Architecture intégrée flexible pour réseaux de neurones

Jamel Eddine Ouali

THESE

présentée par

Jamel Eddine OUALI

pour obtenir le titre de **DOCTEUR**

de l'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE
(arrêté ministériel du 23 Novembre 1988)

Spécialité: **Microélectronique (Conception)**

ARCHITECTURE INTEGREE FLEXIBLE POUR RESEAUX DE NEURONES

Date de soutenance: 11 Juin 1991

Composition du jury :

Madame	Gabrièle SAUCIER	
Messieurs	Jeanny HERAULT	Président
	Gérard DREYFUS	Rapporteur
	Michel WEINFELD	Rapporteur
	Jacques TRILHE	Examinateur
	Rached TOURKI	Invité

Thèse préparée au sein du Laboratoire Conception de Systèmes Intégrés

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

46 avenue Felix Viallet
38031 GRENOBLE cedex

Tél. : 76.57.45.00

Année universitaire 1989

Président de l'Institut :
Monsieur Georges LESPINARD

Professeurs des Universités

BARIBAUD Michel	ENSERG	JAUSSAUD Pierre	ENSIEG
BARRAUD Alain	ENSIEG	JOST Rémy	ENSPG
BAUDELET Bernard	ENSPG	JOUBERT Jean-Claude	ENSPG
BEAUFILS Jean-Pierre	INPG	JOURDAIN Geneviève	ENSIEG
BLIMAN Samuel	ENSERG	LACOUME Jean-Louis	ENSIEG
BOIS Philippe	ENSHMG	LADET Pierre	ENSIEG
BONNETAIN Lucien	ENSEEG	LESIEUR Marcel	ENSHMG
BONNET Guy	ENSPG	LESPINARD Georges	ENSHMG
BRISSONNEAU Pierre	ENSIEG	LONGEQUEUE Jean-Pierre	ENSPG
BRUNET Yves	IUFA	LORET Benjamin	ENSHMG
CAILLERIE Denis	ENSHMG	LOUCHET François	ENSEEG
CAVAIGNAC Jean-François	ENSPG	LUCAZEAU Guy	ENSEEG
CHARTIER Germain	ENSPG	MASSE Philippe	ENSIEG
CHENEVIER Pierre	ENSERG	MASSELOT Christian	ENSIEG
CHERADAME Hervé	UFR PGP	MAZARE Guy	ENSIMAG
CHERUY Arlette	ENSIEG	MOHR Roger	ENSIMAG
CHOVET Alain	ENSERG	MOREAU René	ENSHMG
COHEN Joseph	ENSERG	MORET Roger	ENSIEG
COLINET Catherine	ENSEEG	MOSSIERE Jacques	ENSIMAG
CORNUT Bruno	ENSIEG	OBLED Charles	ENSHMG
COULOMB Jean-Louis	ENSIEG	OZIL Patrick	ENSEEG
COUMES André	ENSERG	PA ULEAU Yves	ENSEEG
CROWLEY James	ENSIMAG	PERRET Robert	ENSIEG
DARVE Félix	ENSHMG	PIAU Jean-Michel	ENSHMG
DELLA-DORA Jean	ENSIMAG	PIC Etienne	ENSERG
DEPEY Maurice	ENSERG	PLATEAU Brigitte	ENSIMAG
DEPORTES Jacques	ENSPG	POUPOT Christian	ENSERG
DEROO Daniel	ENSEEG	RAMEAU Jean-Jacques	ENSEEG
DESRE Pierre	ENSEEG	REINISCH Raymond	ENSPG
DOLMAZON Jean-Marc	ENSERG	RENAUD Maurice	UFR PGP
DURAND Francis	ENSEEG	ROBERT André	UFR PGP
DURAND Jean-Louis	ENSPG	ROBERT François	ENSIMAG
FAUTRELLE Yves	ENSHMG	SABONNADIERE Jean-Claude	ENSIEG
FOGGIA Albert	ENSIEG	SAUCIER Gabrièle	ENSIMAG
FONLUPTJean	ENSIMAG	SCHLENKER Claire	ENSPG
FOULARD Claude	ENSIEG	SCHLENKER Michel	ENSPG
GANDINI Alessandro	UFR PGP	SERMET Pierre	ENSERG
GAUBERT Claude	ENSPG	SILVY Jacques	UFR PGP
GENTIL Pierre	ENSERG	SIRIEYS Pierre	ENSHMG
GENTIL Sylviane	ENSIEG	SOHM Jean-Claude	ENSEEG
GREVEN Hélène	IUFA	SOLER Jean-Louis	ENSIMAG
GUEGUEN Claude	ENSIEG	SOUQUET Jean-Louis	ENSEEG
GUERIN Bernard	ENSERG	TROMPETTE Philippe	ENSHMG
GUYOT Pierre	ENSEEG	VINCENT Henri	ENSPG
IVANES Marcel	ENSIEG	ZADWORNY François	ENSERG

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

BECKER Monique
BINDER Zdenek
CHASSERY Jean-Marc
CHOLLET Jean-Pierre
COEY John
COLINET Catherine
COMMAULT Christian
CORNUEJOLS Gérard
COULOMB Jean- Louis
COURNIL M.
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
LACHENAL D.
LADET Pierre
LATOMBE Claudine
LE HUY H.
LE GORREC Bernard
MADAR Roland
MEUNIER G.
MULLER Jean
NGUYEN TRONG Bernadette
NIEZ J.J.
PASTUREL Alain
PLA Fernand
ROGNON J.P.
ROUGER Jean
TCHUENTE Maurice
VINCENT Henri
YAVARI A.R.

Chercheurs du C.N.R.S

DIRECTEURS DE RECHERCHE CLASSE 0

LANDEAU	Ioan
NAYROLLES	Bernard

Directeurs de recherche 1ère Classe

ANSARA Ibrahim
CARRE René
FRUCHART Robert
HOPFINGER Emile
JORRAND Philippe
KRAKOWIAK Sacha
LEPROVOST Christian
VACHAUD Georges
VERJUS Jean-Pierre

Directeurs de recherche 2ème Classe

ALEMANY Antoine
ALLIBERT Colette
ALLIBERT Michel
ARMAND Michel
AUDIER Marc
BERNARD Claude
BINDER Gilbert
BONNET Roland
BORNARD Guy
CAILLET Marcel
CALMET Jacques
CHATILLON Chritiant
CLERMONT Jean-Robert
COURTOIS Bernard
DAVID René
DION Jean-Michel
DRIOLE Jean
DURAND Robert
ESCUDIER Pierre
EUSTATHOPOULOS Nicolas
GARNIER Marcel
GUELIN Pierre
JOUD Jean-Charles
KAMARINOS Georges
KLEITZ Michel
KOFMAN Walter
LEJEUNE Gérard
MADAR Roland
MERMET Jean
MICHEL Jean-Marie
MEUNIER Jacques
PEUZIN Jean-Claude
PIAU Monique
RENOUARD Dominique
SENATEUR Jean-Pierre
SIFAKIS Joseph
SIMON Jean-Paul
SUERY Michel
TEODOSIU Christian
VAUCLIN Michel
VENNEREAU Pierre
WACK Bernard
YONNET Jean-Paul

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

E.N.S.E.E.G
HAMMOU Abdelkader
MARTIN-GARIN Régina
SARRAZIN Pierre
SIMON Jean-Paul

E.N.S.E.R.G
BOREL Joseph

E.N.S.I.E.G
DESCHIZEAUX Pierre
GLANGEAUD François
PERARD Jacques
REINISCH Raymond

E.N.S.H.M.G
ROWE Alain

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

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

Laboratoires extérieurs :

C.N.E.T
DEVINE Rodericq
GERBER Roland
MERCKEL Gérard
PAULEAU Yves

Situation particulière

PROFESSEURS D'UNIVERSITE

DETACHEMENT

ENSIMAG	LATOMBE	J..Claude	Détachement	21/10/1989
ENSHMG	PIERRARD	J.Marie	Détachement	30/04/1989
ENSIMAG	VEILLON	Gérard	Détachement	30/09/1990
ENSIMAG	VERJUS	J.Pierre	Détachement	30/09/1989
ENSPG	BLOCH	Daniel	Recteur à c/	21/12/1988

SURNOMBRE

INPG	CHIAVERINA	Jean	30/09/1989
ENSHMG	BOUVARD	Maurice	30/09/1991
ENSEEG	PARIAUD	J.Charles	30/09/1991

GRENOBLE 1

UNIVERSITE
JOSEPH FOURIER
SCIENCES. TECHNOLOGIE. MEDECINE

Président de l'Université :
M. NEMOZ Alain

Année universitaire 1988-1990

MEMBRES DU CORPS ENSEIGNANT DE SCIENCES ET DE GÉOGRAPHIE

PROFESSEURS de 1ère Classe

ADIBA Michel	Informatique
ANTOINE Pierre	Géologie I.R.I.G.M.
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
BOELHER Jean-Paul	Mécanique
BRAVARD Yves	Géographie
CARLIER Georges	Biologie Végétale
CASTAING Bernard	Physique
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
FINKE Gerd	Informatique
GAGNAIRE Didier	Chimie Physique
GAUTRON René	Chimie
GENIES Eugène	Chimie
GERMAIN Jean-Pierre	Mécanique
GIDON Maurice	Géologie
GUITTON Jacques	Chimie
HICTER Pierre	Chimie
IDELMAN Simon	Physiologie Animale
JANIN Bernard	Géographie
JOLY Jean-René	Mathématiques Pures

JOSELEAU Jean-Paul	Biochimie
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.
LONGEQUEUE Nicole	Physique
LUNA Domingo	Mathématiques Pures
MACHE Régis	Physiologie Végétale
MASCLE Georges	Géologie
MAYNARD Roger	Physique du solide
OMONT Alain	Astrophysique
OZENDA Paul	Botanique (Biologie Végétale)
PANNETIER Jean	Chimie
PAYAN Jean-Jacques	Mathématiques Pures
PEBAY-PEYROULA Jean-Claude	Physique
PERRIER Guy	Géophysique
PIERRE Jean-Louis	Chimie Organique
RENARD Michel	Thermodynamique
RIEDTMANN Christine	Mathématiques
RINAUDO Marguerite	Chimie CERMAV
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
VAN CUTSEM Bernard	Mathématiques Appliquées
VIALON Pierre	Géologie

PROFESSEURS de 2ème Classe

ARMAND Gilbert	Géographie
ATTANE Pierre	Mécanique
BARET Paul	Chimie
BERTIN José	Mathématiques
BLANCHI J. Pierre	STAPS
BLOCK Marc	Biologie
BLUM Jacques	Mathématiques Appliquées
BOITET Christian	Mathématiques Appliquées
BORNAREL Jean	Physique
BORRIONE Dominique	Automatique Informatique
BOUVET Jean	Biologie
BROSSARD Jean	Mathématiques
BRUANDET Jean-François	Physique
BRUGAL Gérard	Biologie
BRUN Gilbert	Biologie
CASTAING Bernard	Physique
CERFF Rudiger	Biologie
CHIARAMELLA Yves	Mathématiques Appliquées
CHOLLET Jean-Pierre	Mécanique
COLOMBEAU Jean-François	Mathématiques (ENSL)
COURT Jean	Chimie
CUNIN Pierre-Yves	Informatique
DAVID Jean	Géographie

DHOUAILLY Danielle	Biologie
DUFRESNOY Alain	Mathématiques Pures
GASPARD François	Physique
GIDON Maurice	Géologie
GIGNOUX Claude	Sciences Nucléaires
GILLARD Roland	Mathématiques Pures
GIORNI Alain	Sciences Nucléaires
GONZALEZ SPRINBERG Gérardo	Mathématiques Pures
GUIGO Maryse	Géographie
GUMUCHAIN Hervé	Géographie
HACQUES Gérard	Mathématiques Appliquées
HERBIN Jacky	Géographie
HERAULT Jeanny	Physique
HERINO Roland	Physique
JARDON Pierre	Chimie
KERCKHOVE Claude	Géologie
MANDARON Paul	Biologie
MARTINEZ Francis	Mathématiques Appliquées
MOREL Alain	Géographie
NEMOZ Alain	Thermodynamique CNRS - CRTBT
NGUYEN HUY Xuong	Informatique
OUDET Bruno	Mathématiques Appliquées
PAUTOU Guy	Biologie
PECHER Arnaud	Géologie
PELMONT Jean	Biochimie
PELLETIER Guy	Astrophysique
PERRIN Claude	Sciences Nucléaires I.S.N.
PIBOULE Michel	Géologie
RAYNAUD Hervé	Mathématiques Appliquées
REGNARD Jean-René	Physique
RICHARD Jean-Marc	Physique
RIEDTMANN Christine	Mathématiques Pures
ROBERT Danielle	Chimie
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
VINCENT Gilbert	Physique
VIVIAN Robert	Géographie
VOTTERO Philippe	Chimie

MEMBRES DU CORPS ENSEIGNANT DE L'IUT 1

PROFESSEURS de 1ère Classe

BUISSON Roger	Physique IUT 1
CHEHIKIAN Alain	E.E.A. IUT 1
DODU Jacques	Mécanique Appliquée IUT 1
NEGRE Robert	Génie Civil IUT 1
NOUGARET Marcel	Automatique IUT 1
PERARD Jacques	E.E.A. IUT 1

PROFESSEURS de 2ème Classe

BEE Marc	Physique IUT 1
BOUTHINON Michel	E.E.A. IUT 1
CHAMBON René	Génie Mécanique IUT 1
CHENAVAS Jean	Physique IUT 1

CHILO Jean	Physique IUT 1
CHOUTEAU Gérard	Physique IUT 1
CONTE René	Physique IUT 1
FOSTER Panayotis	Chimie IUT 1
GOSSE Jean-Pierre	E.E.A. IUT 1
GROS Yves	Physique IUT 1
HAMAR Roger	Chimie IUT 1
KUHN Gérard, (détaché)	Physique IUT 1
LEVIEL Jean-Louis	Physique IUT 1
MAZUER Jean	Physique IUT 1
MICHOULIER Jean	Physique IUT 1
MONLLOR Christian	E.E.A. IUT 1
PERRAUD Robert	Chimie IUT 1
PIERRE Gérard	Chimie IUT 1
TERRIEZ Jean-Michel	Génie Mécanique IUT 1
TOUZAIN Philippe	Chimie IUT 1
TURGEMAN Sylvain	Génie Civil
VINCENDON Marc	Chimie IUT 1
ZIGONE Michel	Physique IUT 1

PROFESSEURS DE PHARMACIE

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

MEMBRES DU CORPS ENSEIGNANT DE MEDECINE

PROFESSEURS Classe Exceptionnelle et 1ère Classe

AMBLARD Pierre	Dermatologie	C.H.R.G.
AMBROISE-THOMAS Pierre	Parasitologie	C.H.R.G.
BEAUDOING André	Pédiatrie-Puériculture	C.H.R.G.
BEREZ Henri	Orthopédie-Traumatologie	Hôpital 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	Hôpital 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 et Cardiovasculaire	C.H.R.G.
LE NOC Pierre	Bactériologie-Virologie	C.H.R.G.
MALINAS Yves	Gynécologie et Obstétrique	C.H.R.G.
MALLION Jean-Michel	Médecine du Travail	C.H.R.G.

MICOUD Max	Clinique Médicale et 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	Anestésiologie	C.H.R.G.
TANCHE Maurice	Physiologie	Faculté La Merci
VIGNAIS Pierre	Biochimie	Faculté La Merci

PROFESSEURS 2ème Classe

BACHELOT Yvan	Endocrinologie	C.H.R.G.
BARGE Michel	Neurochirurgie	C.H.R.G.
BENABID Alim-Louis	Biophysique	Faculté La Merci
BENSA Jean-Claude	Immunologie	Hôpital 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	Hôpital 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 et Cardiovasculaire	C.H.R.G.
CORDONNIER Daniel	Néphrologie	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 et 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	C.H.R.G.
JUNIEN-LAVILLAUROY Claude	O.R.L.	C.H.R.G.
KOLODIE Lucien	Hématologie Biologique	C.H.R.G.
LETOUBLON Christian	Chirurgie Générale	C.H.R.G.
MACHECOURT Jacques	Cardiologie et Maladies Vasculaires	C.H.R.G.
MAGNIN Robert	Hygiène	C.H.R.G.
MASSOT Christian	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	Hôpital 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.

Remerciements

Je tiens à remercier Madame **Gabrièle SAUCIER**, Professeur à l'ENSIMAG et Directeur du laboratoire Conception de Systèmes Intégrés, pour m'avoir accueilli dans son laboratoire et pour avoir dirigé et guidé les travaux menés dans cette thèse. L'aboutissement de cette thèse doit beaucoup à ses nombreuses remarques, suggestions et discussions.

Je remercie également :

Messieurs **Gérard DREYFUS**, Professeur et Directeur du laboratoire d'Electronique à l'ESPCI Paris, et **Michel WEINFELD**, responsable de l'équipe réseaux de neurones au laboratoire d'informatique à l'Ecole Polytechnique de Paris, d'avoir accepté d'être rapporteurs de cette thèse et d'avoir aidé à ma formation dans ce domaine (pour notre étroite collaboration).

Monsieur **Jacques TRILHE**, Docteur-Ingénieur à la direction technique de SGS-Thomson Microelectronics, d'avoir accepté de faire partie du jury et lui sais gré de ses nombreux conseils et remarques.

Monsieur **Jeanny HERAULT**, Professeur et responsable de l'équipe réseaux de neurones au laboratoire TIRF à l'INP de Grenoble, de me faire l'honneur de présider le jury.

Enfin, je remercie les membres du laboratoire CSI, pour l'agréable ambiance de travail et tous ceux qui, de près ou de loin, ont contribué aux travaux présentés dans cette thèse. Mes remerciements vont en particulier à Monsieur **Rached TOURKI,** Professeur à la Faculté des Sciences et Techniques de Monastir en Tunisie, qui m'a fait connaître la microélectronique et m'a encouragé à poursuivre dans ce domaine.

A mes parents pour tout ce qu'ils ont su m'apporter,

A ma femme Noura, mes enfants Ahmed-Amine et ...,

A mes frères, ma soeur et toute la famille.

Résumé

Ayant rappelé brièvement quelques réalisations matérielles de réseaux de neurones artificiels dans un premier chapitre, cette thèse propose une architecture distribuée, synchrone fondée sur l'existence d'un processeur neurone autonome. Ce processeur pourra être personnalisé suivant les caractéristiques du réseau de neurones à implanter et pourra être connecté à d'autres neurones pour former un réseau de structure et de dimension fixées. Ce neurone se présente comme un circuit dédié fabriqué dans un temps court dans un environnement du type compilateur de silicium. Un tel neurone a été conçu et fabriqué et c'est opérationnel. Il implante sous sa version fabriqué uniquement la phase de reconnaissance.

Dans un troisième chapitre, on montre que sans modification de l'architecture, on peut inclure des possibilités d'apprentissage. Pour ceci un algorithme d'apprentissage par la rétropropagation du gradient a été proposé et étudié et on montre son implantation sur le réseau de neurones proposé en précisant l'adjonction de la partie contrôle du neurone à implanter.

Enfin dans un dernier chapitre, nous explorons la possibilité de réaliser de très grands circuits ce qui serait très judicieux pour faire face à la taille des réseaux de neurones requises pour les applications.

Pour ceci nous explorons les possibilités d'intégration sur tranche entière. En effet, il existe une tolérance aux défauts intrinsèques au calcul neuronal et de plus l'implantation physique régulière doit permettre d'isoler et d'isoler et d'exclure les neurones défaillants. Les possibilités d'implantation physique d'une architecture sur tranche entière sont donc présentées dans ce chapitre.

Mots clés

Réseaux de neurones, Silicium, Architecture neuronique décentralisée, Implantation digitale, Neurone paramétrable.

Summary

After a brief survey of recent realizations of artificial neural network, this thesis proposes a distributed, synchronous architecture, based on the existence of autonomous neural processor. The processor will be customized according to the neural network to be implemented and will be easily connected to other neurons to constitute a network with fixed structure and size. This neuron is presented as a dedicated fabricated circuit in a short time in a silicon compiler environment. Such a neuron has been designed and fabricated and proved completely operationnel. In the third chapter, we show that without any modification of the architecture, we include learning possibilities. For this, a learning algorithm using the back-propagation has been studied and we show its implementation on the proposed neural network, specifying the adjonction in the controller of the neuron to be implemented.

Finaly, in the last chapter, we explore the possibility to realize very large circuits which would be judiciously to face to the size of neural network required for the applications. For this, we explore the possibility of wafer scale integration. In fact, there is an intrinsic defect tolerance to neural computation and regular physical implementation must allow to isolate and to exclude the defectives neurones. The possibilities of physical implementation of wafer scale architecture are presented in this chapter.

Keywords

Neural networks, Silicon, Decentralized neural architecture, Digital implementation, Customizable neuron.

Table des Matières

Introduction Générale

L'idée de réaliser des structures artificielles inspirées par le cerveau humain semble être un rêve qui donne régulièrement naissance à des vagues de recherche ayant comme objectif d'en proposer des modèles abstraits et de simuler son fonctionnement. Il est à noter que les réalisations matérielles, en particulier les ordinateurs numériques des dernières décennies, n'ont paradoxalement presque aucun lien avec de telles structures.

Les machines informatiques, en particulier celles inspirées des modèles de Von Neumann, réalisent ce que le cerveau humain ne peut pas faire (exécution rapide d'algorithmes complexes, opérations arithmétiques avec une grande précision). Le calcul neuromimétique permet au contraire de réaliser aisément certaines fonctions du cerveau humain (traitement parallèle d'un très grand nombre d'informations, apprentissage, classification, association, etc...).

Le regain d'intérêt, dans la dernière décennie, pour l'étude des réseaux de neurones artificiels s'est accompagné d'une volonté plus ferme d'arriver à des réalisations matérielles spécifiques. En effet, vu la complexité des réseaux de neurones en nombre de neurones et connexions requises pour traiter des problèmes type, il semble illusoire de songer à des réalisations efficaces sans matériel spécifique.

Aujourd'hui l'évolution du matériel informatique et de la technologie incluant en particulier des possibilités d'intégration sur silicium tout à fait nouvelles ont contribué à faire avancer très rapidement l'état de l'art.

Tout d'abord, l'existence d'ordinateurs parallèles puissants a permis la simulation d'applications diverses sur des réseaux de neurones, dans un temps raisonnable. Ces simulations ont fourni une information précieuse sur les applications pouvant être traitées et sur les caractéristiques d'implantations numériques requises (nombre de bits utiles pour coder les coefficients et les états, etc...).

En parallèle, des propositions de réalisation matérielle de réseaux de neurones ont fleuri. Les premières propositions ont été fondées sur la technologie analogique, tendant ainsi à être plus proches de certaines fonctionnalités du cerveau humain. Un certain manque de souplesse de la technologie analogique ont conduit à une investigation plus poussée de la technologie numérique.

Dans un premier chapitre, nous décrirons d'abord des machines spécifiques utilisant des boîtiers du commerce. Devant la taille des réseaux de neurones

souhaitée pour certaines applications et par suite de l'évolution des technologies et des méthodes et outils de conception, des réalisations de circuits dédiés ont été entreprises. Eux seuls semblent permettre d'espérer une échelle d'intégration rendant l'approche neuromimétique réaliste.

C'est dans cette dernière orientation que s'inscrit cette thèse. Nous proposons dans un deuxième chapitre une architecture distribuée, synchrone, fondée sur l'existence d'un processeur neurone autonome.

Ce processeur pourra être personnalisé suivant les caractéristiques de divers réseaux de neurones à implanter et pourra facilement être connecté à d'autres neurones pour former un réseau de structure et de dimension fixées. Ce neurone se présente comme un module de base pour un circuit dédié fabriqué dans un temps court dans un environnement du type compilateur de silicium.

Cette architecture est originale par son efficacité pour la circulation des données, en effet, elle fait appel à la fois à des bus et à des principes de décalage de données. Un neurone validant cette architecture a été conçu, fabriqué, et s'est avéré complètement opérationnel. Il implante sous sa version fabriquée uniquement la phase de reconnaissance.

Dans un troisième chapitre, nous montrons que sans modifier l'architecture, il est possible d'inclure des possibilités d'apprentissage. Pour ceci un algorithme d'apprentissage par la rétropropagation du gradient a été étudié et son implantation sur le réseau de neurones proposé est présentée, en précisant les modifications de la partie contrôle.

Enfin, dans un dernier chapitre, nous explorons la possibilité de réaliser de très grands circuits ce qui serait très judicieux pour faire face à la taille des réseaux de neurones requise pour les applications.

Pour ceci, nous étudions les possibilités d'intégration sur tranche entière ; à la lueur des expériences préalablement acquises dans le laboratoire Conception de Systèmes Intégrés, nous pouvons en effet affirmer que les réseaux de neurones constituent des candidats privilégiés pour les réalisations tranche entière et ceci à double titre. En effet, il existe une tolérance aux fautes intrinsèques au traitement neuronal et de plus, l'implantation physique régulière doit permettre d'isoler et d'exclure les neurones défectueux. Les possibilités d'implantation physique d'une architecture tranche entière sont donc présentées dans le dernier chapitre.

Chapitre 1

Les réseaux neuromimétiques

1.1 Réseaux de neurones artificiels

Le cerveau humain [Bal86] [Bie86] est une structure massivement parallèle et comprend entre 10 et 100 milliards de neurones, chaque neurone possédant quelques milliers de connexions ou synapses (10.000 environ). Il s'agit donc d'un réseau à très forte complexité de connexions.

Les réseaux de neurones artificiels cherchent à reproduire les fonctions d'un réseau de neurones réel en général face à une application spécifique. Un réseau de neurones peut être modélisé comme un graphe orienté dont les noeuds sont des neurones qui sont reliés par des arcs pondérés.

Les réseaux de neurones artificiels fonctionnent en 2 phases (i) une phase d'apprentissage pendant laquelle le réseau apprend à reconnaître les prototypes présentés. Il apprend à reconnaître des échantillons en ajustant les poids des interconnexions ou coefficients synaptiques entre neurones. (ii) une phase de reconnaissance pendant laquelle on présente au système des vecteurs d'entrée inconnus qu'il va reconnaître. Dans le cas d'un réseau de Hopfield [Hop82] fonctionnant en mémoire associative, le réseau évolue en général vers un des prototypes déjà appris.

Au sein d'un réseau de neurones artificiels, les neurones sont en général fortement interconnectés et placés selon une topologie régulière. Une classe importante de réseaux de neurones est constituée par les réseaux à couches. Parmi les premiers modèles, notons celui du Perceptron [Ros60] [Lip87], qui est un réseau composé d'une seule couche dont les entrées accèdent à tous les neurones (figure 1.1). L'information circule de l'entrée vers la sortie du réseau.

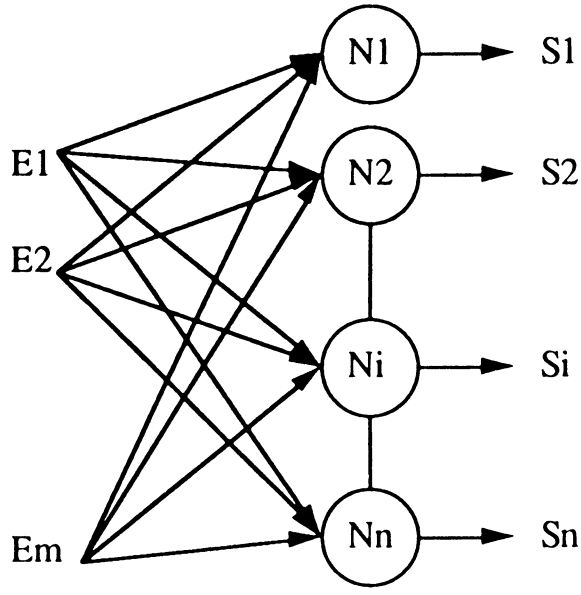


Figure 1.1 Modèle du Perceptron

Dans les réseaux à plusieurs couches, les connexions entre neurones relient les neurones entre couches successives. Elles peuvent être totales ou partielles [Cun89].

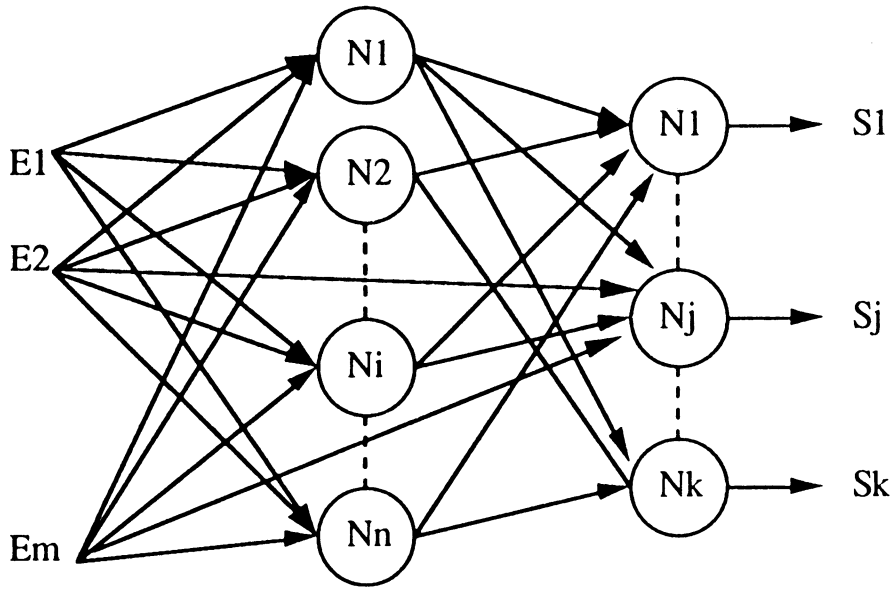


Figure 1.2 : Réseau multicouches

La phase de reconnaissance ainsi que la phase d'apprentissage seront respectivement détaillées dans les chapitres 2 et 3.

On donne dans le tableau 1 des éléments de comparaison entre les réseaux de neurones artificiels et le cerveau humain [Den87] [Sou88] [And88]. Les réseaux de neurones artificiels sont actuellement soit simulés sur des machines existantes, type Von Neumann ou calculateurs parallèles (CRAY, GAPP, Transputers [Pet89] [Rou90] [Wan89]), soit réalisés sur un matériel spécifique avec des composants standard, ou des composants spécifiques (analogiques ou numériques).

Nous énumérerons dans une première partie les principales machines commerciales utilisées pour faire des simulations de résolution de problème par technique neuromimétique.

Dans une seconde partie, nous détaillerons les principes d'une implantation sur circuit dédié et décrirons des exemples pratiques d'implantations en numérique et en analogique. La réalisation de circuits spécifiques permet un gain de vitesse considérable, la contre-partie étant une moindre souplesse d'utilisation.

	Conventionnel	Réseau de neurones	Cerveau
Représentation de l'information	Instructions +Données	Force des connexions dans le réseau	Connexions inter-neurones
Programmation	Instructions +Données initiales	Réseau topologique Apprentissage	Apprentissage
Traitement	Digital	Digital ou analogique	
Nb d'éléments de base	Quelques dizaines	Quelques milliers	Plusieurs milliards
Hardware	Transistor	Transistor	Neurone
Vitesse de commutation	1ns	1ns - 1ms	1ms
Technologie	Silicium	Silicium, optique, chimique	Biologique

Tableau 1 : Diagramme de traitement des informations

1.1.1 Machines permettant de simuler des réseaux neuromimétiques

Plusieurs machines dédiées permettant de simuler des réseaux neuromimétiques sont actuellement disponibles commercialement. Elles permettent de simuler plus rapidement le calcul neuromimétique que les machines d'usage général. Elles se présentent souvent sous forme de cartes à ajouter dans un IBM PC ou une station de travail.

Ces cartes sont des accélérateurs permettant d'effectuer rapidement, la reconnaissance par une multiplication rapide des entrées par les coefficients synaptiques, et d'accélérer également l'apprentissage. Les composants utilisés sont en général des processeurs commerciaux (MC 68020 et MC 68881) et des mémoires de forte capacité (4 Moctets). Citons les systèmes :

- ANZA de Hecht-Nielsen Neurocomputer Corporation [Hec88]

- SIGMA de Science Application International Corp. [Sai88]

- Netsim de Texas Instrument et l'université de Cambridge [Gar87]

- Parallon de Human Device [Tre89]

- Mark III et Mark IV de TRW [Hec86] [Kuc87]

On donne dans le tableau 2 la liste des principaux systèmes proposés avec la dimension maximale du réseau neuromimétique qu'ils permettent de simuler.

Compagnie	Calculateur neuromimétique	Processeurs virtuels	Interconnexions
Hecht-Nielsen Neurocomputer	ANZA ANZA Plus	30 k 1 M	480 k 1,5 M
SAIC	SIGMA	1 M	1 M
Texas Instruments	NETSIM	8 k	250 k
TRW	Mark III (multiple boards) Mark IV	8 k 65 k 236 k	400 k 1,13 M 5,5 M

Tableau 2 : Exemples de circuits existants

Environ 50 types de réseaux neuromimétiques différents ont été publiés, dont la plupart utilisent les structures du perceptron multicouches. Les systèmes mentionnés dans le tableau 2 permettent de simuler différents types de réseaux neuromimétiques.

ANZA est particulièrement adapté à la compression d'image, à l'analyse statistique de données et à la prévision de risques pour l'octroi de prêts bancaires.

SIGMA permet d'implanter des réseaux utilisant la rétropropagation du gradient, les réseaux “mémoire associative”, les réseaux de Hopfield, la machine de Boltzmann et le "self-organizing map" (Kohonen) ; une application particulièrement spectaculaire est un système de détection des explosifs plastiques dans les bagages, système qui est installé à titre expérimental dans un aéroport. Si l'on croit les documents publicitaires fournis par cette société, l'analyse par des méthodes neuromimétiques du rayonnement gamma émis par un bagage irradié donnerait de meilleurs résultats que les techniques conventionnelles.

Netsim est composé de plusieurs cartes, chaque carte contenant un processeur autonome dédié. La taille de la mémoire allouée aux synapses représente de 1 à 2 Moctets. Chaque réseau contient 256 unités avec 1024 entrées par unité. En phase de reconnaissance le réseau peut simuler jusqu'à 450 millions de connexions par seconde. Netsim est capable d'implanter les réseaux de Hopfield et les perceptrons avec rétropropagation.

On peut citer la réalisation par le laboratoire de Traitement d'Images et Reconnaissances de Formes (TIRF) de l'Institut National Polytechnique de Grenoble d'une première machine neuromimétique "CRASY" [Gue87] capable de simuler des architectures de réseau de neurones et de résoudre des systèmes d'équations non linéaires couplées avec des équations différentielles du premier ordre en temps de calcul très court. A titre d'exemple, un système de 128 équations non linéaires est résolu en 20 ms.

Actuellement, une seconde machine en cours de conception appelée SMART [Cha90] permet de traiter les matrices creuses de manière systolique sans effectuer les opérations à argument nul ; ceci permet de compacter des données et est utilisé pour des applications de traitement du signal. Elle offre un temps de calcul de 100 à 1000 fois plus rapide que celui donné par des méthodes algorithmiques.

On peut citer également la machine neuromimétique MIND - 128 (Machine Implementing Neural Devices) réalisée par P. Peretto au Centre d'Etudes Nucléaires de Grenoble [Per90]. C'est une machine mixte (digital analogique) et semi-parallèle. Les coefficients synaptiques codés sur 4 bits et la fonction d'activation à seuil sont stockés dans des mémoires statiques (RAM). Tous les produits partiels des coefficients par les états d'entrée binaires sont convertis à l'aide de convertisseurs digitaux analogiques (CDA). Les états d'entrée sont fournis en série par l'ordinateur hôte permettant à chaque neurone de calculer et de stocker dans une mémoire tous ses produits partiels du potentiel. Les courants délivrés par les convertisseurs DA seront sommés en parallèle pour donner un nouvel état d'entrée au réseau composé de 128 neurones. Les états de sortie finaux du réseau sont directement visibles à l'aide d'une matrice de 8x16 diodes.

Cette machine est particulièrement adaptée aux applications de mémoires associatives, aux problèmes d'optimisation combinatoire et de la physique d'états solides (magnétisme à 2 dimensions). La fréquence de fonctionnement est de 150 KHz et une convergence pour une solution à un problème de mémoire associative est effectuée en 3 ms.

1.1.2 Circuit neuromimétique dédié

Plusieurs groupes de recherche ont travaillé sur l'implantation sur silicium de réseaux neuromimétiques dédiés. Nous décrirons d'abord quelques exemples d'implantations digitales puis des exemples d'implantations analogique.

1.1.2.1 Réalisations

Plusieurs chercheurs ont proposé des circuits neuromimétiques spécifiques digitaux. Citons sans être exhaustif les réalisations de M. Weinfeld [Wei89] de l'Ecole Polytechnique Equipe Réseaux de Neurones en collaboration avec L. Personnaz et G. Dreyfus de l'Ecole Supérieure de Physique et de Chimie Industrielles de Paris (ESPCI) [Per88], Murray de l'Université d'Edinburgh en Ecosse [Mur87], P. Treleaven de l'University College London [Tre89], Ramacher de Siemens [Ram89].

L'architecture la plus répandue est une architecture dite "systolique linéaire" proposée initialement par M. Weinfeld [Wei88] : les états circulent sur une boucle et chaque neurone utilise successivement les différentes valeurs. Le calcul du résultat requiert la circulation de chaque état sur la boucle, le parallélisme du calcul neuronique se réalisant à la fois dans le temps et l'espace. La même architecture adoptée par [Yas90] et [Ham90] utilise des segments de bus pour faire circuler les états entre les neurones.

Le Laboratoire d'Electronique de PHILIPS (LEP) [Dur88] implante le calcul du potentiel des neurones et leur fonction d'activation sur 2 composants distincts.

Dans la première implantation [Wei88], chaque neurone calcule son propre potentiel ainsi que son état de sortie. Dans la deuxième architecture [Dur88] [Ram89], les bits d'état sont présentés successivement à une matrice de portes ET et à un arbre d'additionneurs. Cette matrice reçoit en deuxième entrée les coefficients synaptiques. Tous les produits $C_{ij}.e_j$ des potentiels synaptiques sont effectués simultanément par tous les neurones avec un calcul bit-sériel pour l'état e_j qui défile bit par bit.

Parmi les différentes implantations de circuits numériques [Gas90] [Dur88] [Fau89], nous allons détailler d'avantage la réalisation d'un circuit étudié et conçu par l'Ecole Polytechnique et l'ESPCI.

Ce circuit est composé de 64 neurones totalement connectés. Chaque neurone comporte essentiellement une mémoire de 64 mots codés sur 9 bits, une unité arithmétique à virgule fixe et une sortie binaire.

Le réseau permet de réaliser aussi bien la reconnaissance que l'apprentissage par la règle de Widrow-Hoff. Les états des neurones sont binaires. Ce circuit fabriqué en technologie 1,2 μ est estimé fonctionner à 10 MHz et capable de mettre à jour environ 2.10^8 coefficients par seconde en phase d'apprentissage et il nécessite 30 μs pour converger vers une solution en phase de reconnaissance. Le circuit contient 420.000 transistors environ.

En même temps, des travaux de mise en place d'une plate-forme de test et d'exploitation sont menés. Cette plate-forme est une carte MCP (Macintosh Co-processor Platform) s'installant dans un Mac II. Cette carte, qui possède son propre processeur MC68000 et une mémoire vive, est dotée du système d'exploitation MR-DOS (Minimal Real Time Distributed Operating System) et de circuits d'interface avec le NuBus de la machine hôte.

L'existence de ce système d'exploitation et de cette circuiterie permet d'interfacer le réseau de neurones avec le système du Macintosh à un niveau relativement élevé, et dispense de la conception d'une carte particulière. De plus, cette structure autorise un fonctionnement autonome de la carte, ou de plusieurs cartes analogues installées dans la même machine ou dans des machines du réseau Apple Talk.

1.1.2.2 Réalisations analogiques

Plusieurs chercheurs ont proposé des circuits analogiques [Moo88] [Tsi89] [Ver89]. On peut citer particulièrement les réalisations, de Goser de l'Université de Dortmund en Allemagne [Gos84] [Rue88] et de J. Hérault du TIRF [Ros89] sur l'implantation d'un réseau neuromimétique analogique de 25 synapses utilisable en mémoire associative.

Aux USA, deux implantations analogiques de circuits neuroniques sont particulièrement intéressantes, celle d'AT&T et celle d'INTEL. Nous allons les décrire sommairement. Dans le premier cas, un circuit en technologie CMOS 2,5µ (6,7 x 6,7 mm2 et 75000 transistors) de 54 neurones analogiques avec des interconnexions programmables au moyen de RAM [Gra88] a été réalisé.

La plus grande partie du silicium utile, pratiquement 90%, est utilisée pour les interconnexions. La puce comprend 2916 (54 x 54) dispositifs de connexion. A chaque intersection d'une ligne d'entrée et d'une ligne de sortie, une connexion résistive peut être placée. Toutes les connexions sont programmables.

Un exemple de réseau de résistances est donné en figure 1.3 ; chaque résistance commandée par un bit de la RAM. En sortie, deux inverseurs sont connectés en série et fonctionnent en amplificateur. Les coefficients synaptiques sont programmables et peuvent prendre 3 valeurs +a, 0, -b ; les états sont binaires. Il n'y a pas d'apprentissage possible ; le circuit mémorise un maximum de 54 vecteurs de 54 bits [Gra88].

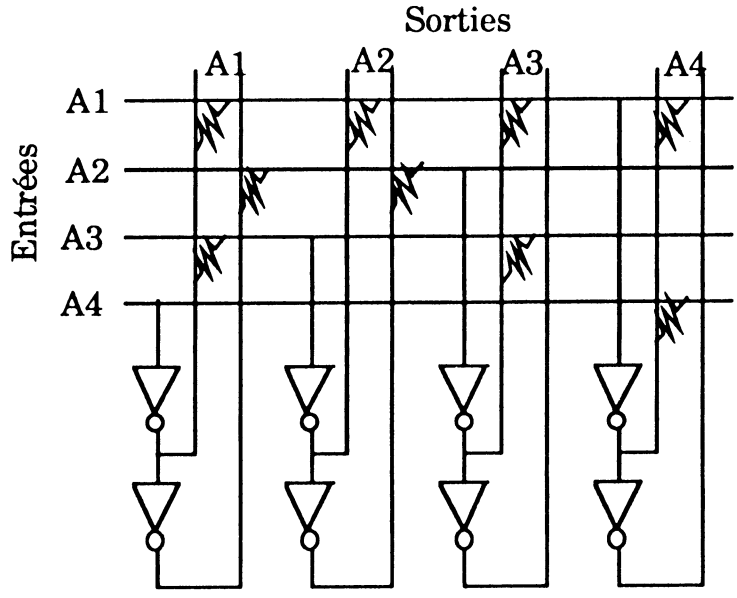


Figure 1.3 : Schéma de 4 neurones illustrant le circuit de AT &T

Le circuit neuromimétique analogique d'INTEL a été présenté pour la première fois à l'IJCNN'89 [Hol89]. C'est la première implantation sur silicium d'un circuit neuromimétique avec stockage des coefficients synaptiques en analogique dans une EEPROM, l'idée ayant été publiée auparavant par Alspector J.et al.[Als87].

Le stockage d'un bit dans une RAM dans le circuit d'AT&T nécessite 6 transistors. Avec le stockage analogique dans la grille flottante d'une EEPROM, INTEL mémorise une donnée sur 7 bits, dans le transistor d'une cellule EEPROM.

Le mécanisme physique du stockage analogique dans une cellule EEPROM n'est pas différent de celui du stockage digital. La programmation analogique consiste à charger partiellement la grille flottante de la cellule EEPROM en utilisant des impulsions plus courtes, de façon à obtenir une valeur parmi 256 sur le courant du transistor de la cellule EEPROM. Si la durée de stockage du coefficient synaptique dépasse 15 ans à une température ambiante, la précision est diminuée à 4 bits.

D'autres essais de stockage analogique ont également été publiés dans la littérature, par stockage de charge dans des CCD [Vit90] [Hoe90].

Le coefficient synaptique est multiplié par le vecteur d'entrée, lui aussi analogique, en utilisant un multiplieur de GILBERT (analogique lui aussi).

Le circuit d'INTEL contient 64 neurones entièrement connectés ; il est dessiné avec des règles de dessin de 1 μ et ses performances typiques sont 10^{10} multiplications par seconde.

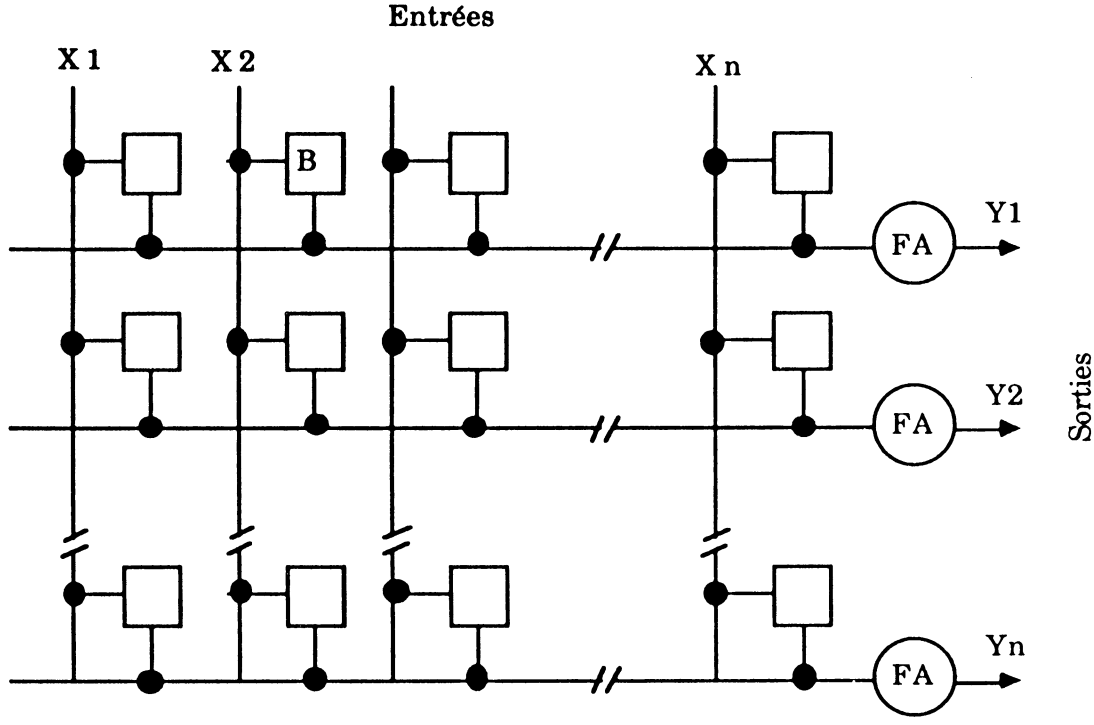


Figure 1.4 : Schéma illustrant le réseau neuromimétique d'INTEL

Les entrées et les sorties du réseau sont notées respectivement par X_i et Y_j, où i varie de 1 à n et j de 1 à m. Chaque carré représente une synapse et FA la fonction d'activation d'un neurone composé d'une ligne de synapse.

1.2 Conclusion

Les implantations de circuits neuromimétiques analogiques sont plus denses que les implantations numériques. Le calcul du potentiel est réalisé de façon complètement parallèle (coefficient . état) ; dans les réalisations digitales, le calcul est en général série parallèle. L'une des 2 données (état ou coefficient) est traitée parallèlement ; le deuxième est traitée sérialement.

En revanche, les circuits analogiques sont très peu souples quant à la précision et ne peuvent être étendus facilement à l'apprentissage. Les circuits digitaux, eux s'avèrent très souples et s'adaptent à des précisions variables et à des algorithmes diversifiés. Ceci sera illustré dans la suite de cette thèse.

Chapitre 2

Circuit neuromimétique dédié

2.1 Introduction

Les réseaux de neurones sont très fortement connectés et peuvent apparaître à priori comme de mauvais candidats pour l'intégration sur silicium, s'ils sont implantés par des méthodes architecturales habituelles [Sei84]. Le très grand nombre de connexions à l'intérieur d'un réseau de neurones et la diversité des modèles de réseaux d'interconnexion nécessitent une stratégie d'interconnexion flexible et efficace.

Il s'agit donc de trouver une architecture de réseaux de neurones artificiels, ayant des communications simples avec le monde extérieur, le minimum de connexions physique entre neurones, une structure répétitive permettant d'implanter des réseaux complexes.

Pour ce faire, un principe de connexions spatio-temporelles a été proposé par beaucoup d'auteurs et initialement par M. Weinfeld [Wei88]. Dans cette approche, une donnée devant être disponible pour plusieurs unités physiques, circule dans un registre à décalage et les unités physiques échantillonent les données en temps opportun. Ces architectures ont été appelées architectures systoliques linéaires car le calcul se fait sur des vagues de données successives [Zub89] [Jon88] [Pac89] [Mor90] [Kun88]. Partant de ce principe, nous avons développé une architecture fondée sur une communication par décalage et segments de bus inspirée de ce principe de connexions spatio-temporelles. Cette architecture est souple et permet d'implanter n'importe quel type de réseaux de neurones.

Il s'agit d'une architecture décentralisée, synchrone, et ne nécessitant aucun contrôle externe après le chargement du réseau. Le processeur neurone est une entité physique identifiée ; il contrôle le traitement des données ainsi que les communications avec les neurones voisins.

Le choix d'un processeur neurone bien identifié a été motivé par des considérations de tolérance aux défauts et aux pannes. En effet, les observations des neurobiologistes montrant que malgrès la destruction d'un certain nombre de neurones (vieillissement par exemple), le cerveau concerve la plupart des informations déjà apprises à l'aide d'une nouvelle mise à jour des poids synaptiques entre les neurones restants.

Ceci nous conduit à penser, par analogie, qu'un réseau distribué de neurones bien identifiés regroupant topologiquement la mémoire des coefficients synaptiques, la fonction de calcul du potentiel, la fonction d'activation, apporte une tolérance intrinsèque aux défauts de fin de fabrication et éventuellement une tolérance aux pannes en cours de fonctionnement.

Cette idée ne sera pas développée dans le cadre de ce travail mais est une base de l'option architecturale. En effet, un réseau neuromimétique réalisé par 3 blocs distincts centralisés à savoir un multiplieur, une mémoire, un opérateur réalisant la fonction d'activation donnera un résultat entièrement faux si un défaut se trouve dans l'un quelconque des 3 éléments précédemment désignés.

Si nous réalisons maintenant ce même calcul sur un matériel composé de processeurs identiques, ayant chacun un multiplieur, une mémoire et une fonction de seuil, un défaut physique dans le circuit intégré, dans la mesure ou il n'est pas fatal pour le circuit, entachera d'erreur la contribution au résultat d'un seul neurone. Le résultat sera donc d'autant moins erroné que l'information sera plus distribuée (nombre de neurones plus grand). Ce raisonnement s'étend également aux défauts de fin de fabrication.

Il est en effet important d'être tolérant aux défauts de fin de fabrication si l'on veut intégrer sur un seul circuit un très grand nombre de neurones. L'extension à de très grands circuits et ce point de vue seront en effet développés dans le chapitre 4.

2.2. Le processeur neurone

A chaque neurone est associée une entité physique appelée processeur neurone. Ce processeur (figure 2.1) contient un contrôleur capable de séquencer toutes les étapes de calcul, aussi bien en phase de reconnaissance qu'en phase d'apprentissage.

Chaque processeur possède en mémoire locale les coefficients relatifs aux neurones qui le relient à ses voisins. Pendant la phase de reconnaissance, il doit effectuer une somme des entrées pondérées par les coefficients synaptiques, le résultat est appelé potentiel synaptique. Ensuite il calcule son nouvel état de sortie par seuillage réalisé par la fonction d'activation choisie pour l'application. Pendant la phase d'apprentissage, il calcule des termes d'erreur et une correction à apporter aux coefficients.

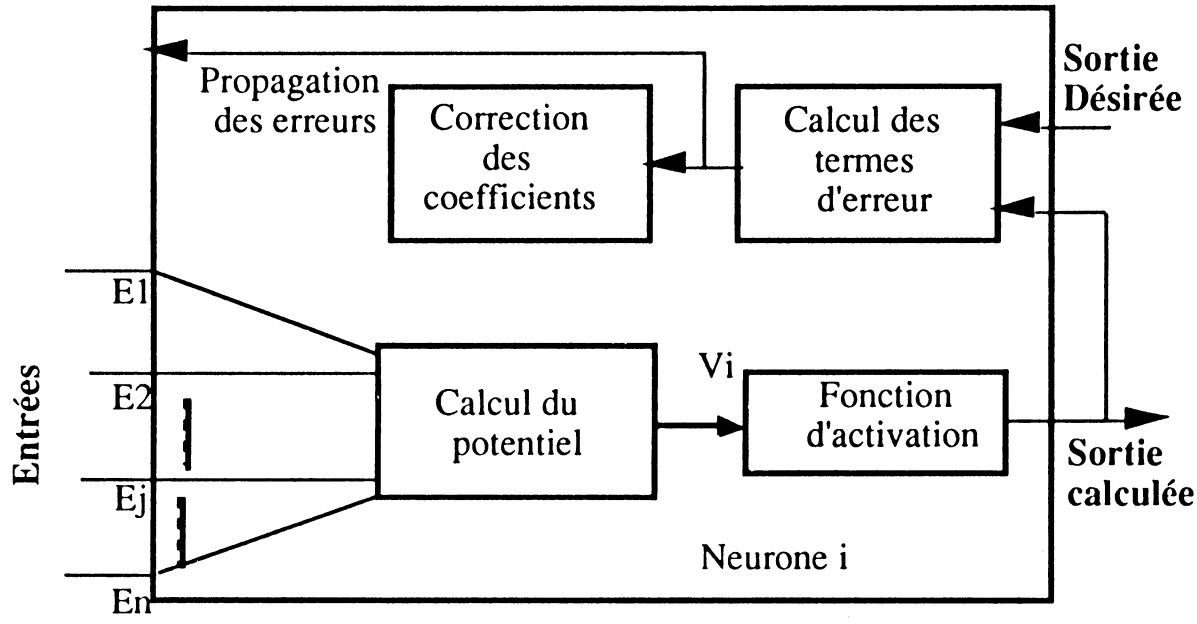


Figure 2.1 : Schéma d'un processeur neurone

2.3.Architectures des réseaux de neurones

L'architecture proposée consiste en un réseau de processeurs neurones autonomes. Les communications sont réalisées d'une part par décalage de la valeur des états des neurones à travers les registres d'entrée des neurones ; d'autre part par des bus d'entrée segmentables permettant en outre un chargement rapide des données.

L'architecture proposée a comme objectif d'être personnalisable afin d'implanter tout réseau de neurones. Ceci veut dire que sans modifier l'architecture de base, on peut générer des réalisations spécifiques en fixant ces paramètres.

Cette personnalisation se fait essentiellement par la modification des organigrammes de contrôle du processeur. Etant donné l'environnement de synthèse automatisée dans lequel nous évoluons, (générateur automatique de contrôleur, compilateur de chemin de données), cette personnalisation se fait à moindre effort. Les paramètres de cette personnalisation sont la taille et la structure de communication du réseau, la précision du code des coefficients et des états (nombre de bits) ainsi que le type de fonction d'activation utilisée et le type de fonction de calcul d'erreurs pour l'apprentissage. L'architecture est d'autre part cascadable pour permettre l'assemblage aisé de circuits dédiés.

2.3.1 Connexions spatio-temporelles

Le principe des connexions spatio-temporelles consiste à remplacer les connexions d'un neurone avec d'autres neurones, par une circulation dans le temps des valeurs des états dans un ensemble de registres montés en pile "rebouclée" (décalage vertical) (figure 2.2.a). Le neurone échantillonne à chaque instant à son entrée une valeur d'état disponible.

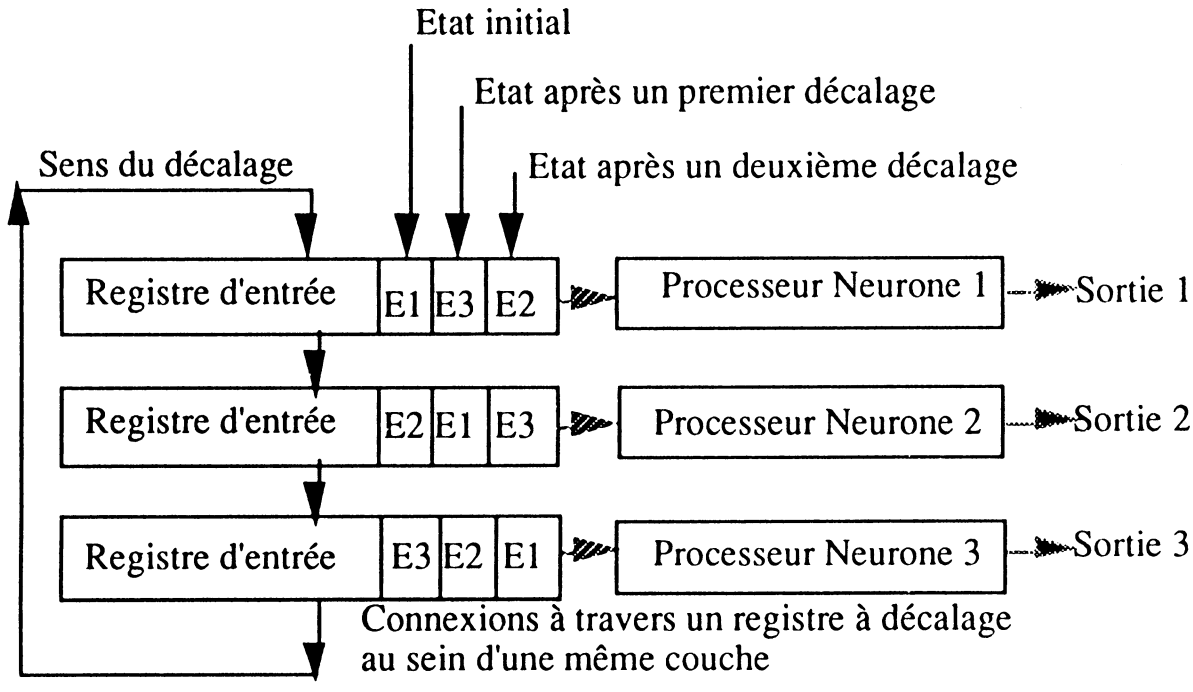


Figure 2.2.a : Réalisation de connexions spatio-temporelles

Considérons un exemple de 3 neurones totalement connectés ayant comme états respectifs E_1, E_2 et E_3.

Chaque neurone doit calculer en premier lieu son potentiel synaptique, respectivement V_1, V_2 et V_3 donnés par :

$$V_1 = W_{11}.E_1 + W_{12}.E_2 + W_{13}.E_3$$

$$V_2 = W_{21}.E_1 + W_{22}.E_2 + W_{23}.E_3$$

$$V_3 = W_{31}.E_1 + W_{32}.E_2 + W_{33}.E_3$$

On présente les entrées E_1, E_2 et E_3 aux entrées respectives des neurones N1, N2 et N3, figure 2.2.b. Ces entrées sont appliquées sur des segments de bus bidirectionnels (figure 2.3).

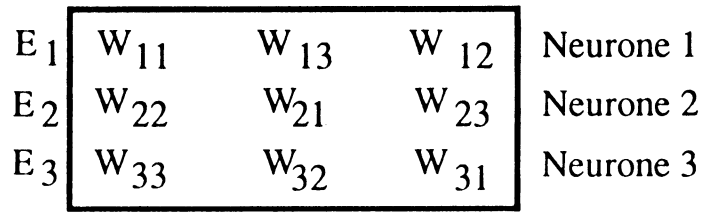


Figure 2.2.b : Initialisation du réseau

Durant la première étape de calcul, figure 2.2.c , chaque neurone calcule une partie du potentiel. Puis, on effectue un décalage vertical des entrées d'un neurone vers le neurone suivant ; dans une seconde étape, les entrées respectives des neurones N1, N2 et N3 seront E_3, E_1 et E_2.

Dans la troisième phase, un autre décalage est effectué et présente les entrées E_2, E_3 et E_1 respectivement aux neurones N1, N2 et N3.

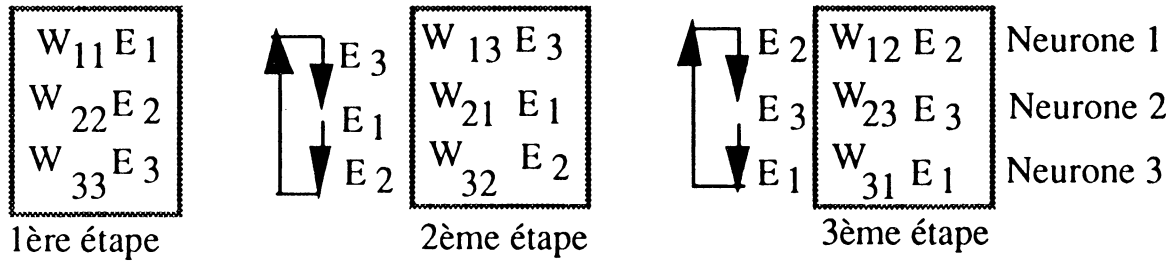


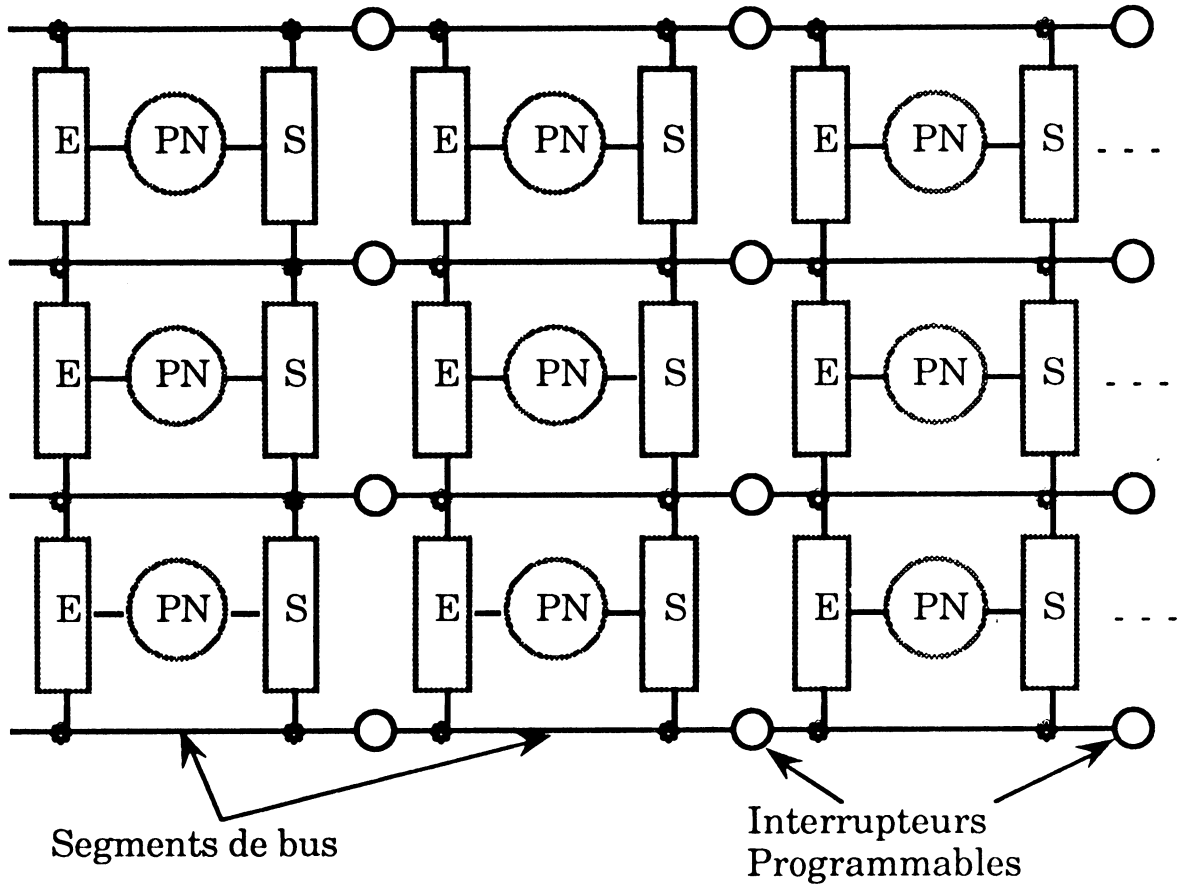
Figure 2.2.c : Décalage effectué pour le calcul du potentiel

2.3.2 Architecture globale :

L'architecture proposée (figure 2.3) est constituée d'un réseau 2-D de neurones, chaque neurone possèdant un registre d'entrée et un registre de sortie et étant connecté en lecture et écriture à deux segments de bus. Les segments de bus sont reliés entre eux par l'intermédiaire d'interrupteurs programmables.

Chaque processeur neurone est capable de réaliser toutes les étapes de calcul dans les phases d'apprentissage et de reconnaissance et de contrôler les communications avec les neurones voisins en commandant les interrupteurs isolant les segments de bus.

Chaque processeur neurone possède aussi un ensemble de registres contenant toutes les informations nécessaires concernant l'architecture globale du réseau et tous ses paramètres de personnalisation, qu'on détaillera ultérieurement.



E : Registre d'entrée
S : Registre de sortie
PN : Processeur neurone

Figure 2.3 : Architecture globale (vue partielle)

Un dessin simplifié du neurone sera utilisé dans la suite du paragraphe (figure 2.4).

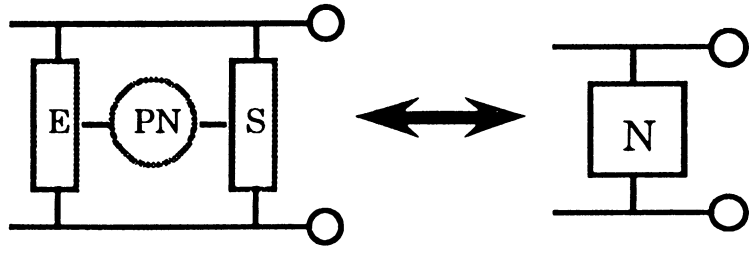


Figure 2.4 : Représentation simplifiée du processeur neurone

2.3.3 Organisation topologique et pliage de couches

Bien que l'architecture proposée soit générale, l'effort essentiel de cette étude est porté sur les réseaux à couches et nous illustrons principalement tous les points de cette étude sur ce type de réseau. Afin d'optimiser la réalisation topologique sur silicium, une couche de neurones peut être "pliée". Notre "pliage" permet de réaliser un circuit rectangulaire ayant un facteur de forme souhaité par le concepteur en évitant l'inconvénient dû à l'existence d'une grande ligne d'interconnexion d'une topologie en serpent. Nous montrerons dans les rubriques c, d et e de ce paragraphe que ceci permet de réaliser de nombreux schémas d'interconnexions et ne crée pas de problème de propagation électrique.

La figure 2.5 montre à titre d'exemple l'organisation topologique de 8 processeurs pour un réseau quelconque. Il y a pliage des couches logiques pour obtenir une grille de 2 x 4 processeurs. La configuration des interrupteurs illustre une architecture à connexions totales dans laquelle les 8 neurones constituent un réseau de Hopfield. Avec diverses positions des interrupteurs, il est possible de configurer un réseau à couches ou un réseau de Hopfield.

2 des 3 bus bidirectionnels sont utilisés pour les entrées/sorties comme indiqué dans la suite. Les interrupteurs 10, 11 et 12 séparent la couche formée des 8 neurones de la couche suivante. Les interrupteurs 4, 5 et 6 partagent la colonne logique en deux parties, chacune est formée de deux colonnes physiques. Le pliage de la couche permet donc d'éviter les longues lignes de rebouclage sur silicium et d'assurer le décalage circulaire des données.

Les interrupteurs 5, 10, 11 et 12 seront toujours ouverts pendant le calcul en phase de reconnaissance. Ils sont fermés au moment du chargement et/ou du transfert des données d'une couche à une autre. Les autres interrupteurs 1, 3, 4, 6, 7, 9 seront en permanence fermés.

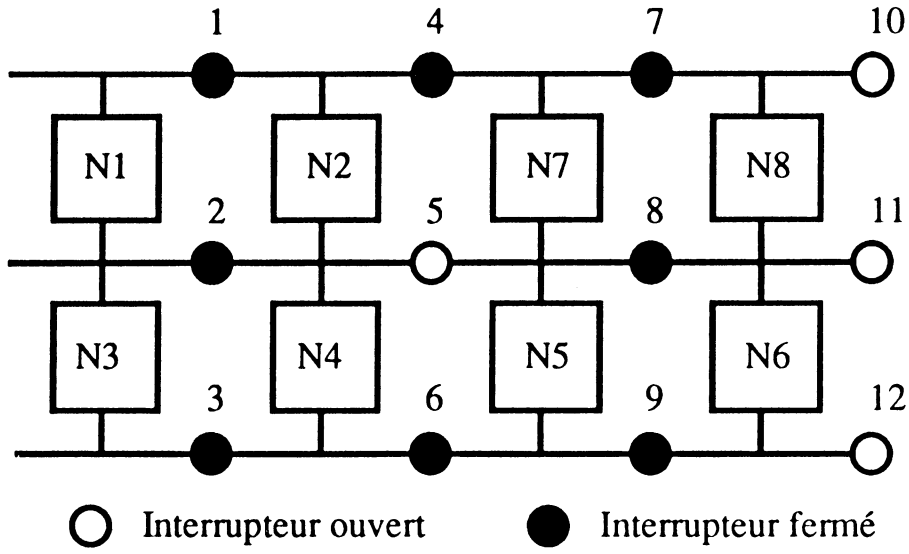


Figure 2.5 : Pliage topologique des couches

a) Circulation des données

Lors de la première étape, les segments de bus horizontaux indiqués en figure 2.5.a, sont chargés par les entrées E_1, E_3, E_5, E_7. Chaque neurone, ayant reçu sa valeur d'entrée par le registre d'entrée, calcule le potentiel partiel $W_{ij}.Ej$. Dans l'exemple considéré, les neurones :

N1, N2 chargés par la valeur E_1 calculent respectivement $W_{11}.E_1$ et $W_{21}.E_1$
N3, N4 chargés par la valeur E_3 calculent respectivement $W_{33}.E3$ et $W_{43}.E_3$
N5, N6 chargés par la valeur E_5 calculent respectivement $W_{55}.E_5$ et $W_{65}.E_5$
N7, N8 chargés par la valeur E_7 calculent respectivement $W_{77}.E_7$ et $W_{87}.E_7$.

Les équations des potentiels V_i montrent en gras les termes calculés pendant cette première phase :

$V_1= \mathbf{W_{11}.E_1} + W_{12}.E_2 + W_{13}.E_3 + W_{14}.E_4 + W_{15}.E_5 + W_{16}.E_6 + W_{17}.E_7 + W_{18}.E_8$

$V_2= \mathbf{W_{21}.E_1} + W_{22}.E_2 + W_{23}.E_3 + W_{24}.E_4 + W_{25}.E_5 + W_{26}.E_6 + W_{27}.E_7 + W_{28}.E_8$

$V_3= W_{31}.E_1 + W_{32}.E_2 + \mathbf{W_{33}.E_3} + W_{34}.E_4 + W_{35}.E_5 + W_{36}.E_6 + W_{37}.E_7 + W_{38}.E_8$

$V_4= W_{41}.E_1 + W_{42}.E_2 + \mathbf{W_{43}.E_3} + W_{44}.E_4 + W_{45}.E_5 + W_{46}.E_6 + W_{47}.E_7 + W_{48}.E_8$

$V_5= W_{51}.E_1 + W_{52}.E_2 + W_{53}.E_3 + W_{54}.E_4 + \mathbf{W_{55}.E_5} + W_{56}.E_6 + W_{57}.E_7 + W_{58}.E_8$

$V_6= W_{61}.E_1 + W_{62}.E_2 + W_{63}.E_3 + W_{64}.E_4 + \mathbf{W_{65}.E_5} + W_{66}.E_6 + W_{67}.E_7 + W_{68}.E_8$

$V_7= W_{71}.E_1 + W_{72}.E_2 + W_{73}.E_3 + W_{74}.E_4 + W_{75}.E_5 + W_{76}.E_6 + \mathbf{W_{77}.E_7} + W_{78}.E_8$

$V_8= W_{81}.E_1 + W_{82}.E_2 + W_{83}.E_3 + W_{84}.E_4 + W_{85}.E_5 + W_{86}.E_6 + \mathbf{W_{87}.E_7} + W_{88}.E_8$

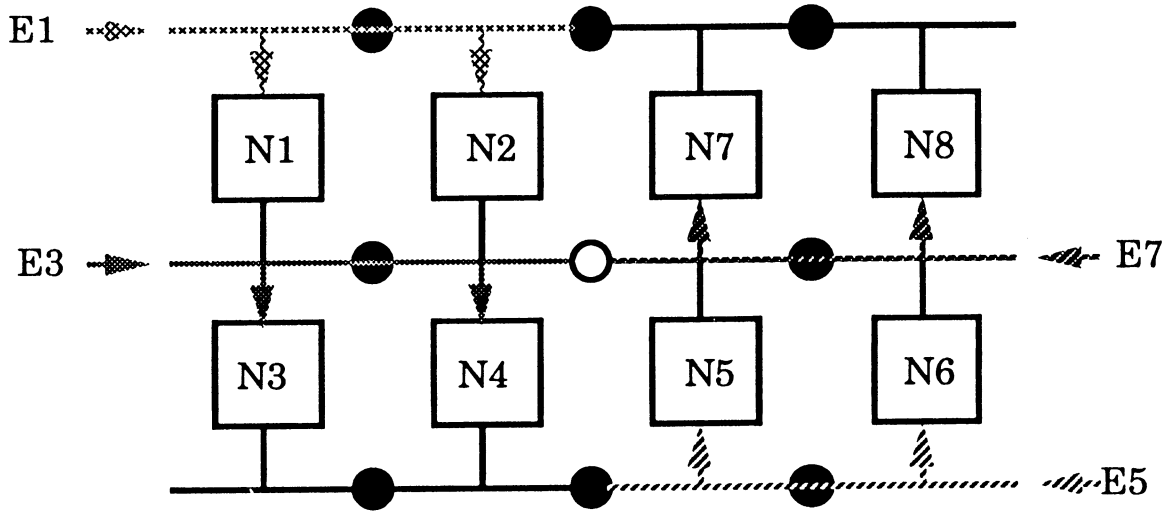


Figure 2.5.a : Pliage de la couche

Pour la suite des calculs, les termes calculés à chaque étape sont indiqués en gras, les termes déjà calculés sont soulignés. A l'étape suivante, figure 2.5.b, on effectue un premier décalage des entrées E_1, E_3, E_5, E_7 vers les segments de bus bas pour les neurone N1, N2, N3, N4 et vers le haut pour les neurones N5, N6, N7, N8. Les neurones calculent les termes suivants :

$$V_1 = \underline{W_{11}.E_1} + W_{12}.E_2 + W_{13}.E_3 + W_{14}.E_4 + W_{15}.E_5 + W_{16}.E_6 + \mathbf{W_{17}.E_7} + W_{18}.E_8$$

$$V_2 = \underline{W_{21}.E_1} + W_{22}.E_2 + W_{23}.E_3 + W_{24}.E_4 + W_{25}.E_5 + W_{26}.E_6 + \mathbf{W_{27}.E_7} + W_{28}.E_8$$

$$V_3 = \mathbf{W_{31}.E_1} + W_{32}.E_2 + \underline{W_{33}.E_3} + W_{34}.E_4 + W_{35}.E_5 + W_{36}.E_6 + W_{37}.E_7 + W_{38}.E_8$$

$$V_4 = \mathbf{W_{41}.E_1} + W_{42}.E_2 + \underline{W_{43}.E_3} + W_{44}.E_4 + W_{45}.E_5 + W_{46}.E_6 + W_{47}.E_7 + W_{48}.E_8$$

$$V_5 = W_{51}.E_1 + W_{52}.E_2 + \mathbf{W_{53}.E_3} + W_{54}.E_4 + \underline{W_{55}.E_5} + W_{56}.E_6 + W_{57}.E_7 + W_{58}.E_8$$

$$V_6 = W_{61}.E_1 + W_{62}.E_2 + \mathbf{W_{63}.E_3} + W_{64}.E_4 + \underline{W_{65}.E_5} + W_{66}.E_6 + W_{67}.E_7 + W_{68}.E_8$$

$$V_7 = W_{71}.E_1 + W_{72}.E_2 + W_{73}.E_3 + W_{74}.E_4 + \mathbf{W_{75}.E_5} + W_{76}.E_6 + \underline{W_{77}.E_7} + W_{78}.E_8$$

$$V_8 = W_{81}.E_1 + W_{82}.E_2 + W_{83}.E_3 + W_{84}.E_4 + \mathbf{W_{85}.E_5} + W_{86}.E_6 + \underline{W_{87}.E_7} + W_{88}.E_8$$

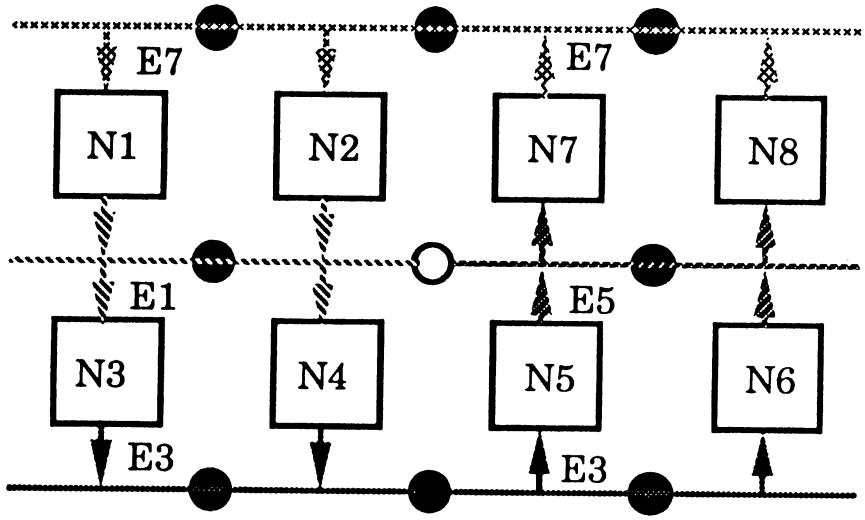


Figure 2.5.b : Circulation des données après un premier décalage

Un deuxième décalage amène les données sur les segments de bus comme indiqué en figure 2.5.c, et nous permet de calculer les termes indiqués

$V_1 = \underline{W_{11}.E_1} + W_{12}.E_2 + W_{13}.E_3 + W_{14}.E_4 + \mathbf{W_{15}.E_5} + W_{16}.E_6 + \underline{W_{17}.E_7} + W_{18}.E_8$

$V_2 = \underline{W_{21}.E_1} + W_{22}.E_2 + W_{23}.E_3 + W_{24}.E_4 + \mathbf{W_{25}.E_5} + W_{26}.E_6 + \underline{W_{27}.E_7} + W_{28}.E_8$

$V_3 = \underline{W_{31}.E_1} + W_{32}.E_2 + \underline{W_{33}.E_3} + W_{34}.E_4 + W_{35}.E_5 + W_{36}.E_6 + \mathbf{W_{37}.E_7} + W_{38}.E_8$

$V_4 = \underline{W_{41}.E_1} + W_{42}.E_2 + \underline{W_{43}.E_3} + W_{44}.E_4 + W_{45}.E_5 + W_{46}.E_6 + \mathbf{W_{47}.E_7} + W_{48}.E_8$

$V_5 = \mathbf{W_{51}.E_1} + W_{52}.E_2 + \underline{W_{53}.E_3} + W_{54}.E_4 + \underline{W_{55}.E_5} + W_{56}.E_6 + W_{57}.E_7 + W_{58}.E_8$

$V_6 = \mathbf{W_{61}.E_1} + W_{62}.E_2 + \underline{W_{63}.E_3} + W_{64}.E_4 + \underline{W_{65}.E_5} + W_{66}.E_6 + W_{67}.E_7 + W_{68}.E_8$

$V_7 = W_{71}.E_1 + W_{72}.E_2 + \mathbf{W_{73}.E_3} + W_{74}.E_4 + \underline{W_{75}.E_5} + W_{76}.E_6 + \underline{W_{77}.E_7} + W_{78}.E_8$

$V_8 = W_{81}.E_1 + W_{82}.E_2 + \mathbf{W_{83}.E_3} + W_{84}.E_4 + \underline{W_{85}.E_5} + W_{86}.E_6 + \underline{W_{87}.E_7} + W_{88}.E_8$

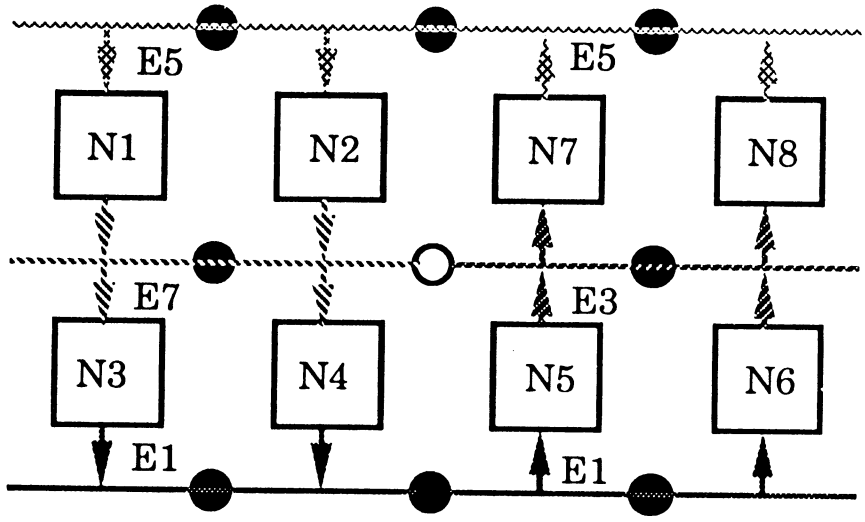


Figure 2.5.c : Position des données après un deuxième décalage

Le troisième décalage permet de finir le traitement de la première vague de données en calculant :

$V_1 = \underline{W_{11}.E_1} + W_{12}.E_2 + \mathbf{W_{13}.E_3} + W_{14}.E_4 + \underline{W_{15}.E_5} + W_{16}.E_6 + \underline{W_{17}.E_7} + W_{18}.E_8$

$V_2 = \underline{W_{21}.E_1} + W_{22}.E_2 + \mathbf{W_{23}.E_3} + W_{24}.E_4 + \underline{W_{25}.E_5} + W_{26}.E_6 + \underline{W_{27}.E_7} + W_{28}.E_8$

$V_3 = \underline{W_{31}.E_1} + W_{32}.E_2 + \underline{W_{33}.E_3} + W_{34}.E_4 + \mathbf{W_{35}.E_5} + W_{36}.E_6 + \underline{W_{37}.E_7} + W_{38}.E_8$

$V_4 = \underline{W_{41}.E_1} + W_{42}.E_2 + \underline{W_{43}.E_3} + W_{44}.E_4 + \mathbf{W_{45}.E_5} + W_{46}.E_6 + \underline{W_{47}.E_7} + W_{48}.E_8$

$V_5 = \underline{W_{51}.E_1} + W_{52}.E_2 + \underline{W_{53}.E_3} + W_{54}.E_4 + \underline{W_{55}.E_5} + W_{56}.E_6 + \mathbf{W_{57}.E_7} + W_{58}.E_8$

$V_6 = \underline{W_{61}.E_1} + W_{62}.E_2 + \underline{W_{63}.E_3} + W_{64}.E_4 + \underline{W_{65}.E_5} + W_{66}.E_6 + \mathbf{W_{67}.E_7} + W_{68}.E_8$

$V_7 = \mathbf{W_{71}.E_1} + W_{72}.E_2 + \underline{W_{73}.E_3} + W_{74}.E_4 + \underline{W_{75}.E_5} + W_{76}.E_6 + \underline{W_{77}.E_7} + W_{78}.E_8$

$V_8 = \mathbf{W_{81}.E_1} + W_{82}.E_2 + \underline{W_{83}.E_3} + W_{84}.E_4 + \underline{W_{85}.E_5} + W_{86}.E_6 + \underline{W_{87}.E_7} + W_{88}.E_8$

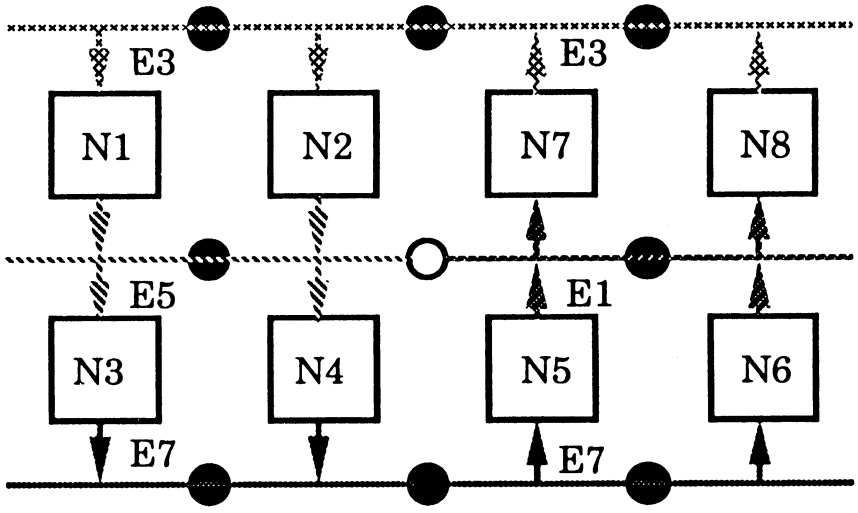


Figure 2.5.d : Circulation des données après un dernier décalage

La deuxième vague de données est chargée de la même façon que la première, figure 2.5.e, les valeurs E_1, E_3, E_5, E_7 sont respectivement remplacées par les valeurs E_2, E_4, E_6, E_8. De la même manière, on calcule le reste des termes des potentiels en effectuant seulement 3 décalages des nouvelles données.

b) Chargement des données

Les valeurs d'entrée de la première vague seront chargées en 2 temps par les mêmes bus d'entrée. Pour le chargement des processeurs situés après le pliage, tous les commutateurs doivent être fermés. Le chargement peut être effectué soit par les deux bus du haut soit par les deux bus du bas. La deuxième vague de données est chargée de la même façon.

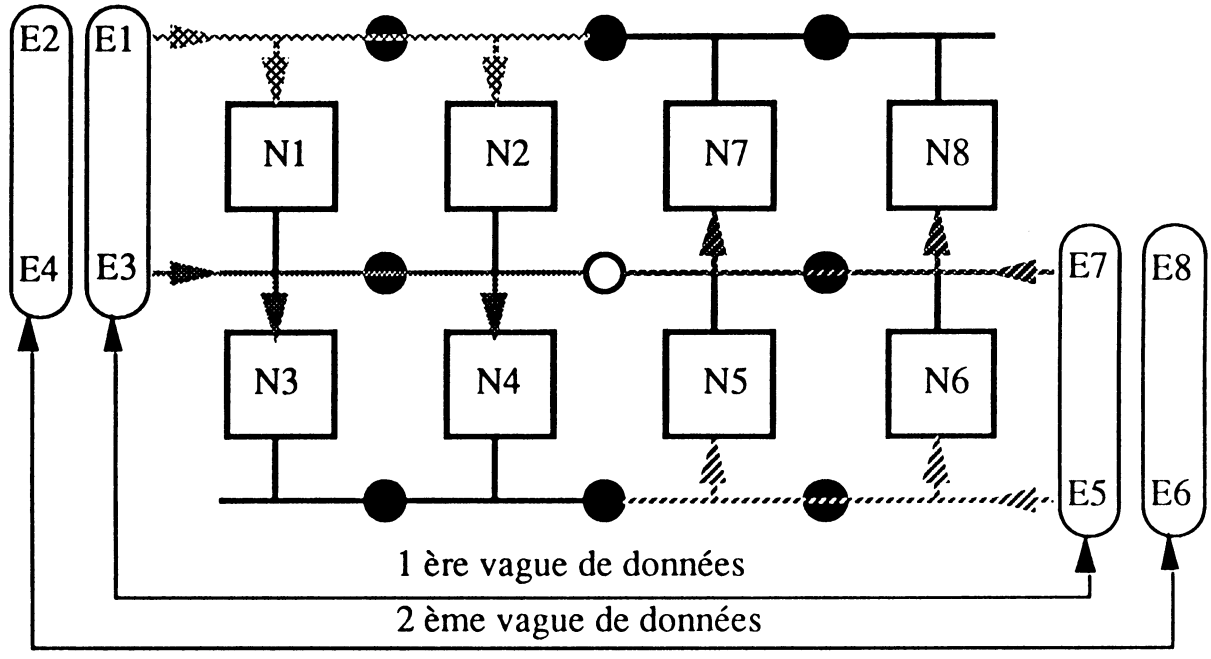


Figure 2.5.e: Chargement des vagues de données

c) Réseaux en couches

On suppose maintenant que les 8 neurones précédents constituent la première couche d'un réseau à couches. Après avoir calculé tous les potentiels synaptiques, chaque processeur neurone calcule son état de sortie. Une fois calculés les états de sortie de la première couche, ils sont transférés vers la couche suivante. On suppose que la couche suivante est constituée de 6 neurones organisés en 2 lignes et 3 colonnes, figure 2.6. Le pliage des couches est choisi arbitrairement dans cet exemple.

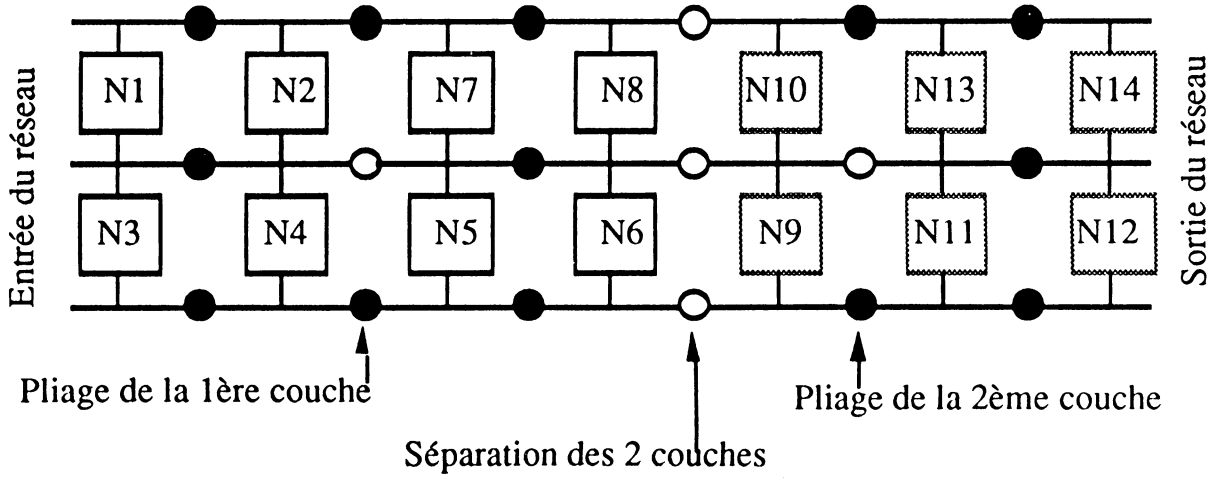


Figure 2.6 : Organisation d'un réseau à 2 couches

Le transfert des valeurs des états de la première couche vers la deuxième se fait également en deux vagues via les deux bus bas des neurones.

Chaque vague est réalisée en 2 étapes, tous les commutateurs étant fermés au moment du transfert.

Première vague :

Etape 1 :

Le processeur N1 transfère son état au processeur N10
Le processeur N3 transfère son état au processeur N9

Etape 2 :

Le processeur N2 transfère son état aux processeurs N13 et N14
Le processeur N4 transfère son état aux processeurs N11 et N12

Deuxième vague :

Etape 1 :

Le processeur N7 transfère son état au processeur N10
Le processeur N5 transfère son état au processeur N9

Etape 2 :

Le processeur N8 transfère son état aux processeurs N13 et N14
Le processeur N6 transfère son état aux processeurs N11 et N12

Le calcul des potentiels synaptiques des processeurs neurones de la 2ème couche est effectué suivant la méthode exposée pour la 1ère couche.

La configuration des commutateurs pendant le calcul est donnée par la figure 2.6. Les états de sortie des 6 neurones de la seconde couche peuvent être lus sur les 2 bus du haut ou les 2 bus du bas.

d) Réseaux à couches à connexions non complètes

Comme dit précédemment, notre architecture est souple et peut s'adapter à divers types de réseaux. Nous allons montrer comment elle permet de réaliser des connexions non complètes entre couches comme celle réalisée dans la figure 2.7.

La solution consiste à conserver l'arrangement topologique des neurones précédemment décrit puis de prévoir un masque, qui permet à la partie contrôle du processeur de prendre ou non la valeur présente dans le registre d'entrée après chaque décalage des données.

Un masque est un registre contenu dans un neurone dont le ième bit vaut 1 si le neurone est connecté au ième élément de la couche précédente et 0 dans l'autre cas.La constitution du masque est rendu difficile par le pliage topologique. Illustrons ceci sur le réseau de la figure 2.7. Dans cette figure la partie droite représente le réseau plié.

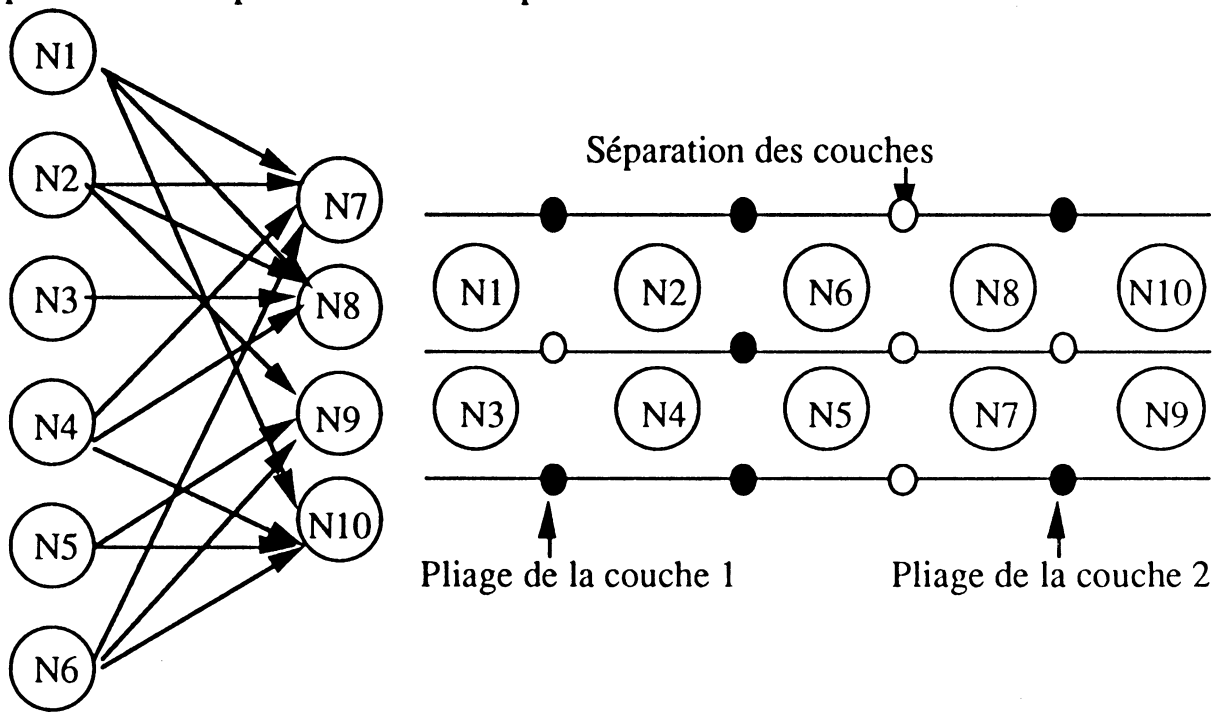


Figure 2.7 : Représentation d'un réseau à connexions non complètes

Dans la première vague de transfert des états de la couche 1 vers la couche 2, on a :

Etape 1 :

Le processeur N1 transfère son état au processeur N8
Le processeur N3 transfère son état au processeur N7

Etape 2 :

Le processeur N2 transfère son état aux processeurs N10
Le processeur N4 transfère son état aux processeurs N9

Le neurone N7 est relié topologiquement aux neurones N1, N2, N4 et N6. Le vecteur qui lui correspond est 110101. Mais en tenant compte du fait que le sens de décalage des données est contraire au sens de rotation des aiguilles d'une montre, que N7 est situé à la deuxième ligne et première colonne dans la seconde couche et que la couche est formée de 2 lignes et 2 colonnes, le masque que doit contenir le neurone N7 est (01110100).

En effet, N7 n'est pas concerné par la valeur de N3, il ne l'échantillonne pas. Lors du premier décalage, N7 reçoit la valeur décalée par N8 (initialement chargée par N1). N7 est relié au neurone N1, il prend la valeur décalée. Au deuxième décalage, c'est la valeur de N10 (initialement chargée par N2) qui se présente, N7 doit l'échantillonner. Au troisième décalage, c'est la valeur de N9 (initialement chargée par N4) qui se présente, N7 doit l'échantillonner. Ceci nous donne la première partie du masque "0111".

Considérons similairement la deuxième vague :

Etape 1 :

Le processeur N6 transfère son état au processeur N8
Le processeur N5 transfère son état au processeur N7
Les processeurs N9 et N10 ne seront pas chargés.

N7 n'est pas relié à N5 donc la valeur qu'il a chargé doit être ignorée. Au premier décalage, N7 voit la valeur décalée de N8 (initialement chargée par N6), qui le concerne ; il doit l'utiliser pour son calcul de potentiel. On remarque qu'il n'y a plus de valeur utilisée par N7, mais il faudrait bien faire 2 décalages supplémentaires afin de permettre aux autres neurones N9 et N10 de compléter leurs calculs des potentiels. On déduit que pour N7, on a "0100" comme deuxième moitié du masque.

De la même façon, on détermine les masques des neurones N8, N9, N10 qui sont respectivement (11110000), (00010110), (01010011).

Une deuxième solution possible consiste à mettre des coefficients nuls pour les neurones non connectés. cette solution est efficace si le nombre de coefficients nuls à rajouter est faible. Sinon, une augmentation, non négligeable, de la surface de la mémoire synaptique, est à déplorer.

e) Neurones fictifs dans une couche

Si le nombre de neurones d'une couche logique ne permet pas d'avoir un arrangement topologique constituant un réseau 2D complet, on rajoute des neurones fictifs afin de compléter la matrice topologique. Ces neurones fictifs sont nécessaires car d'une part leurs registres d'entrée assurent la continuité des décalages effectués pendant le calcul des potentiels synaptiques et d'autre part leurs interrupteurs associés permettent le transfert des états vers la couche suivante.

On montre dans la figure 2.8 un exemple d'une couche à 5 neurones, qui illustre la nécessité du neurone N6, neurone fictif. Ces neurones sont toujours des neurones inactifs. La proportion de neurones inactifs est toujours faible si l'on joue astucieusement sur la répartition en lignes et colonnes, la pénalisation au niveau de la vitesse est donc faible.

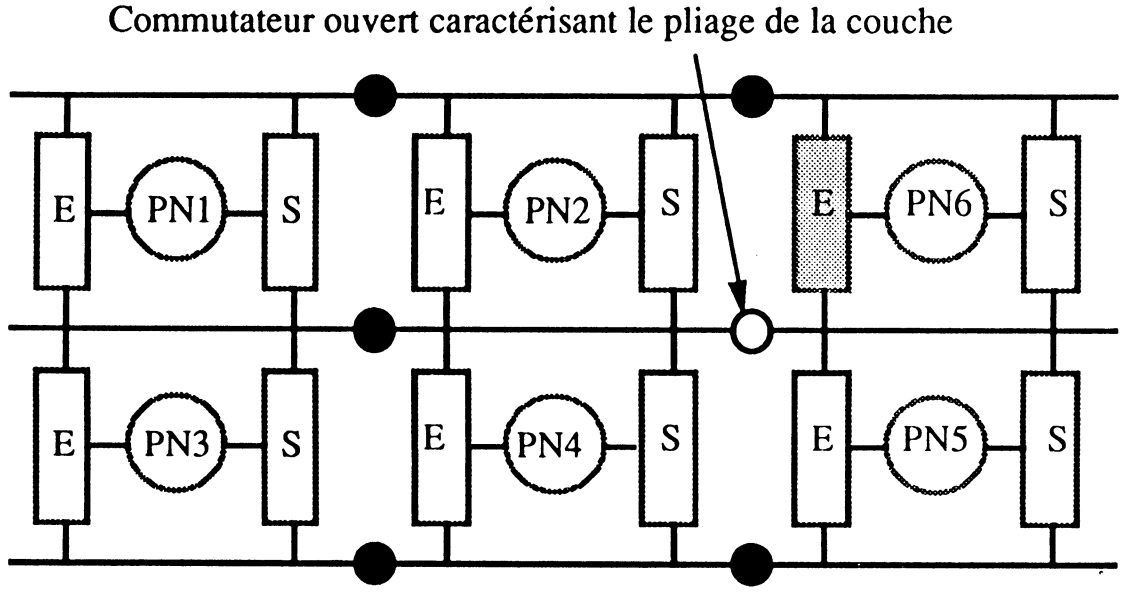


Figure 2.8 : Exemple d'utilisation d'un neurone fictif

2.4. Architecture détaillée

2.4.1 Structure du processeur neurone

Le processeur neurone a été conçu comme un processeur dédié. Il comporte (figure 2.9) :

- Un registre d'entrée qui est utilisé pour le chargement de la valeur initiale si le neurone est dans la première couche, ou bien le chargement des états de sortie des neurones prédécesseurs s'il est dans une couche interne ou dans la couche de sortie. Il est également utilisé comme élément de décalage pour faire circuler (verticalement) les données des entrées,

- Un registre de sortie qui mémorise l'état calculé par le neurone,

- Un chemin de données constitué d'une unité de traitement et de registres dédiés,

- Une mémoire locale qui stocke les coefficients d'interaction du neurone avec ses prédécesseurs.

- Plusieurs registres de personnalisation (Per_R),

- Un contrôleur, qui gère les traitements effectués par le neurone dans les différentes phases et le transfert des données d'une couche à une autre ainsi que le chargement des données initiales.

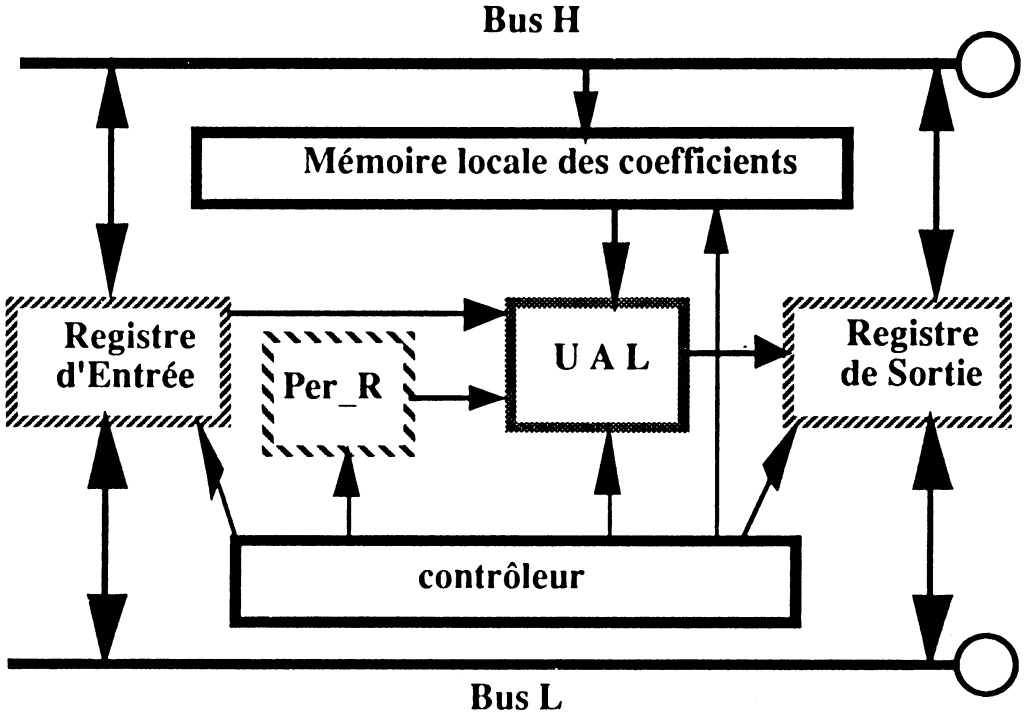


Figure 2.9 : Schéma du neurone.

2.4.2 Démarche générale

La démarche de conception du processeur dédié repose sur une description de ses fonctionnalités par organigramme de contrôle et sur des outils de synthèse très évolués. Un organigramme de contrôle définit à un niveau RTL (transfert de registre) les différentes opérations exécutées par la partie opérative (ou chemin de données) ainsi que leur séquencement [Fle80]. C'est à ce niveau que l'architecture du neurone et des réseaux globaux sont simulés, validés et que le programme de test de validation sera généré.

Les étapes de synthèse d'un processeur neurone sont données dans la figure 2.10. A partir de l'organigramme de contrôle, on extrait la spécification du contrôleur ou machine d'états finis décrit par un automate de Mealy et la partie opérative correspondante. Des outils de synthèse automatique nous permettent de créer, à partir du graphe de la machine d'états finis, un réseau de cellules standard.

De même à partir de la description structurelle de la partie opérative en terme de réseau de blocs, une synthèse automatique est fournie par l'outil de conception (Data Path Compiler). La mémoire est générée par un générateur de mémoire RAM. Le neurone est enfin réalisé par assemblage automatisé de la partie contrôle, du chemin de données et de la mémoire.

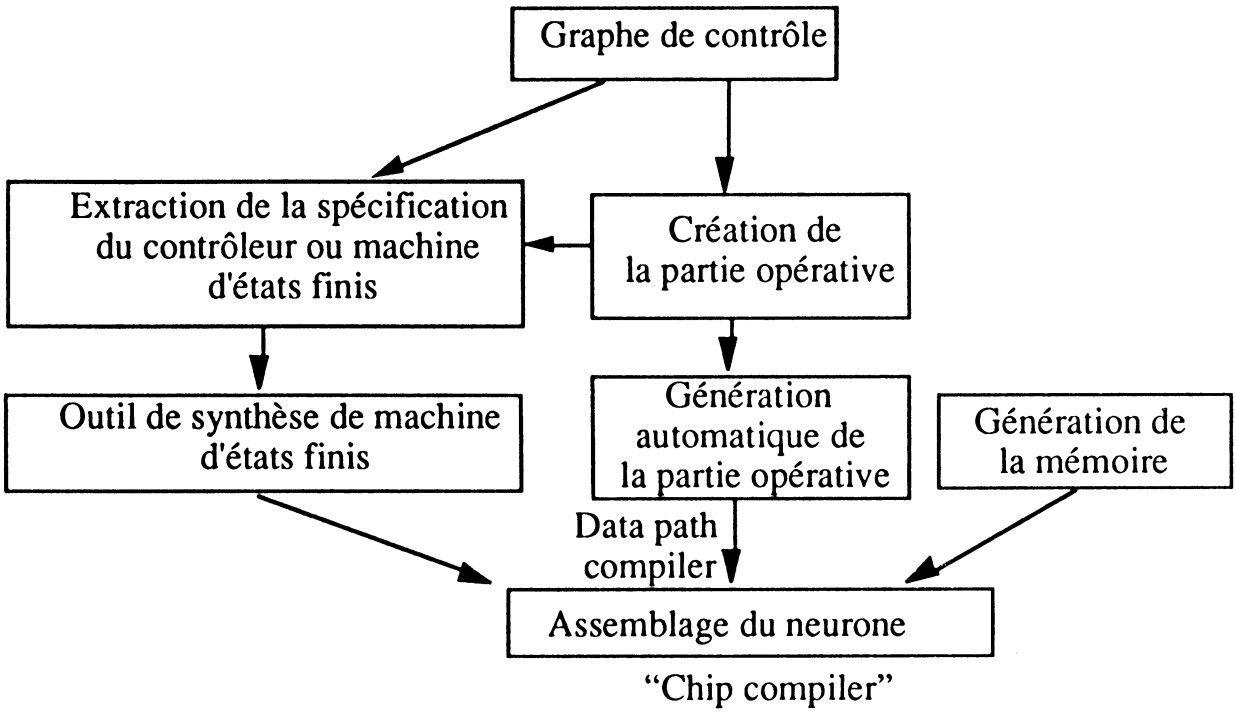


Figure 2.10 : Description des étapes de conception

2.4.3 Chemin de données ou partie opérative

Le chemin de données ou partie opérative est composé d'une unité de traitement, de registres et de compteurs. Les seules opérations de traitement effectuées par la partie opérative pendant l'apprentissage et la reconnaissance sont des multiplications suivant l'algorithme de Booth modifié [Boo61], des additions de produits partiels et des formatages de données. On implantera donc essentiellement dans l'unité de traitement un additionneur/soustracteur. Comme cela était précédemment dit, le chargement des entrées/sorties est fait par des bus bidirectionnels dont la taille sera fixée par le format des données d'entrée/sortie et des coefficients synaptiques.
Les données initiales et les coefficients sont codés sur n bits en complément à deux (n étant un paramètre de l'architecture du processeur).

2.4.3.1 Unité de traitement

Dans le cas de la reconnaissance, les produits partiels concernent les termes du potentiel synaptique. Dans le cas de l'apprentissage, ils concernent les termes d'erreurs introduits par le réseau ainsi que ceux introduits par l'algorithme de correction des coefficients synaptiques. Dans le cas de la reconnaissance, les compteurs sont utilisés pour calculer les adresses mémoires, le nombre de coefficients utilisés, le nombre de vagues de données chargées et le nombre de cycles de la multiplication. L'unité de traitement (figure 2.11) est composée de :

- trois registres R1, R2 et R3. Le registre R3 sera chargé par une valeur d'entrée, le produit sera stocké dans les 2 registres R2 et R3.

- deux additionneurs/soustracteurs de n bits sont utilisés pour réaliser les différentes multiplications et cumuler les produits partiels du potentiels.

- deux accumulateurs des produits partiels.

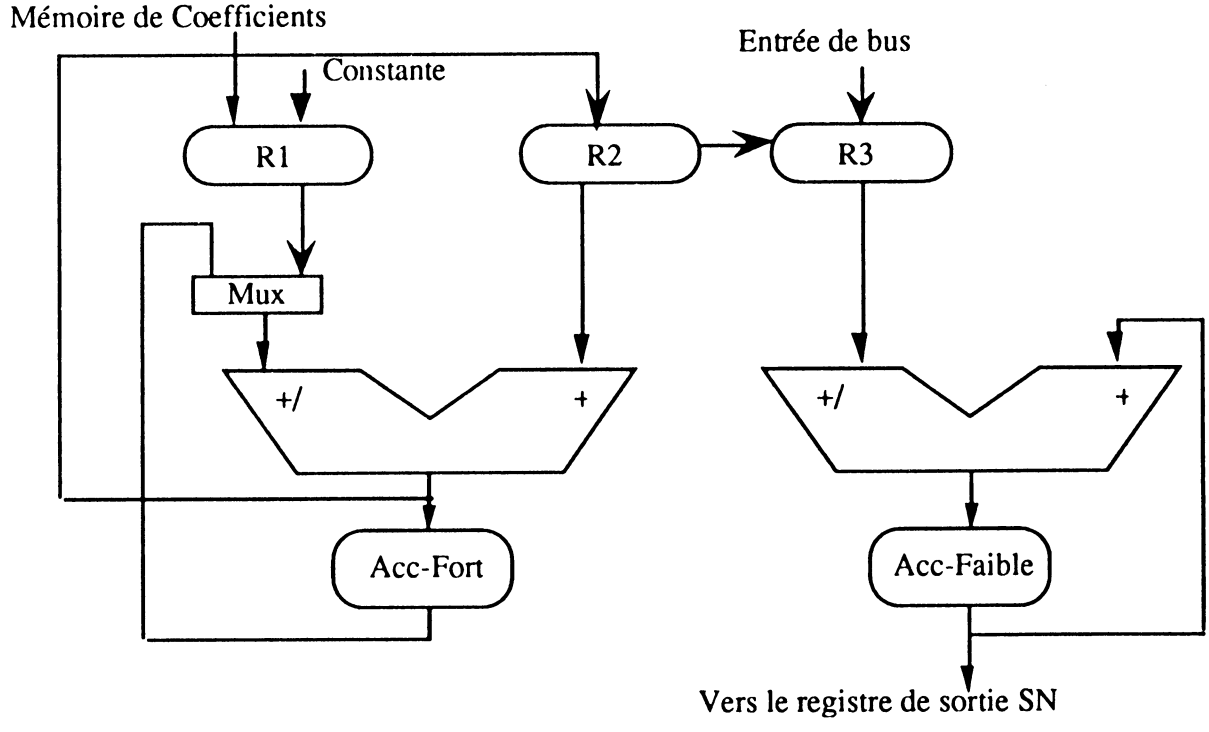


Figure 2.11 : L'unité de traitement

2.4.3.2 Paramètres de personnalisation

Le but principal de cette étude étant la phase de reconnaissance, nous détaillerons essentiellement les paramètres de personnalisation relatifs à cette phase. Les registres de personnalisation utilisés pour la reconnaissance contiennent :

- Le nombre de bits codant l'état et les coefficients,
- La position du neurone dans la couche physique et dans la couche logique,
- Le nombre de neurones prédécesseurs,
- Le nombre de lignes et de colonnes,
- Le sens de circulation des données dans la couche,
- Le nombre de décalages nécessaires pour reformater les résultats des multiplications.

Tous ces paramètres, excepté le dernier, concernent l'architecture du réseau. Par contre, le dernier paramètre concerne le format des données. Peu de travaux sont actuellement disponibles sur le nombre de bits utiles pour coder les données et les coefficients [Joh90]. En réalité, l'apprentissage nécessite beaucoup plus de précision que la reconnaissance. Pour certains types de réseaux, l'apprentissage est fait en virgule flottante.

Pour la reconnaissance, il est inutile de conserver une telle précision. Une représentation en virgule fixe sera alors utilisée pour l'unité de traitement. Ceci présente une limitation de notre circuit si on souhaite l'utiliser à la fois pour l'apprentissage et la reconnaissance. On choisira dans ce cas pour la réalisation sur silicium la plus grande valeur du format d'entrée, à savoir celle de l'apprentissage. En phase de reconnaissance, on considère que les données et les coefficients sont représentés sur un format de n bits en complément à deux avec une virgule fixe. La représentation de la valeur de l'état contient un 1 bit de signe et la partie significative de n-1 bits. Les parties entière et décimale des coefficients varient suivant la précision demandée par l'utilisateur et la position de la virgule est un paramètre donné par le concepteur.

Les neurones binaires ont une fonction d'activation à seuil. Leurs états sont 2 valeurs prises dans l'ensemble {-1, +1} ou {0, +1}. Les neurones dont les coefficients et les états sont codés sur plusieurs bits ont, en général, une fonction d'activation du type pseudo-linéaire ou sigmoïdale.

Leurs états appartiennent aux deux intervalles]0, 1[ou]-1, 1[pour des valeurs réelles du potentiel ($V \in \Re$) (figure 2.17.b et figure 2.17.c).

Pratiquement, pour une conception sur silicium, les bornes infinies de la fonction ne seront jamais atteintes, elles seront considérées finies pour un format de représentation donné. En effet, les états codés sur n bits auront des valeurs comprises dans l'intervalle $[0, 1-1/2^{n-1}]$ ou bien $[-1+1/2^{n-1}, 1-1/2^{n-1}]$.
On rappelle que les coefficients sont aussi codés sur n bits. Le plus petit entier représentable par ce format est égal à $-2^{n-1}-1$; le plus grand entier négatif est égal à -1 ; le plus grand entier positif est égal à $2^{n-1}-1$.
L'introduction de la virgule fixe dans le format des coefficients revient à diviser le nombre entier par 2^j où j représente le nombre de bits après la virgule. Dans ce cas, la plus petite valeur représentable par ce format est égale à $(-2^{n-1}-1)/2^j$; la plus grande valeur négative est égale à $-1/2^j$; la plus grande valeur positive est égale à $(2^{n-1}-1)/2^j$.

Les deux bornes minimales pour des coefficients respectivement négatifs et positifs sont $[(-2^{n-1}-1)/2^j; -1/2^j]$; $[0; (2^{n-1}-1)/2^j]$.

Illustrons ceci par un exemple où n est égal à 8. La valeur de l'état appartiendra à [0, +0,9921875] ou bien [-1, +0,9921875]. $1-2^{n-1}$ sera représenté par 0,1111111 (la virgule est placée au même endroit dans une représentation décimale).
Si les coefficients sont des entiers la valeur négative minimale est égale à - 127 représenté par 10000001 ; le plus grand entier négatif est égal à -1 représenté par 11111110 ; le plus grand entier positif est égal à +127 représenté par 01111111
Si on considère une virgule fixe placée au milieu du format (j=4) la valeur négative minimale est égal à - 8 représenté par 1001,0000 ; la plus grande valeur négative est égale à -0,0625 représenté par 1111,1111 ; la plus grande valeur positive est égale à +7,9375 représenté par 0111,1111.

2.4.4 Partie contrôle

Le fonctionnement de la partie contrôle est défini comme dit précédemment par les organigrammes de contrôle décrivant les opérations effectuées en mode d'apprentissage et/ou reconnaissance. Dans cette partie, on détaillera seulement son fonctionnement en mode de reconnaissance. Le mode d'apprentissage sera développé dans le chapitre 3.

Le contrôleur doit assurer les fonctions suivantes :

- L'initialisation du réseau,
- Le chargement des données à l'entrée du réseau,
- La gestion des interrupteurs programmables,
- La circulation des données entre neurones dans la couche,
- Le calcul du potentiel et de l'état du neurone,
- Le transfert de l'état du neurone à la couche suivante.

L'organigramme est donné macroscopiquement dans la figure 2.12. Les parties d'organigramme relatives à ces différentes fonctionnalités sont discutées ensuite.

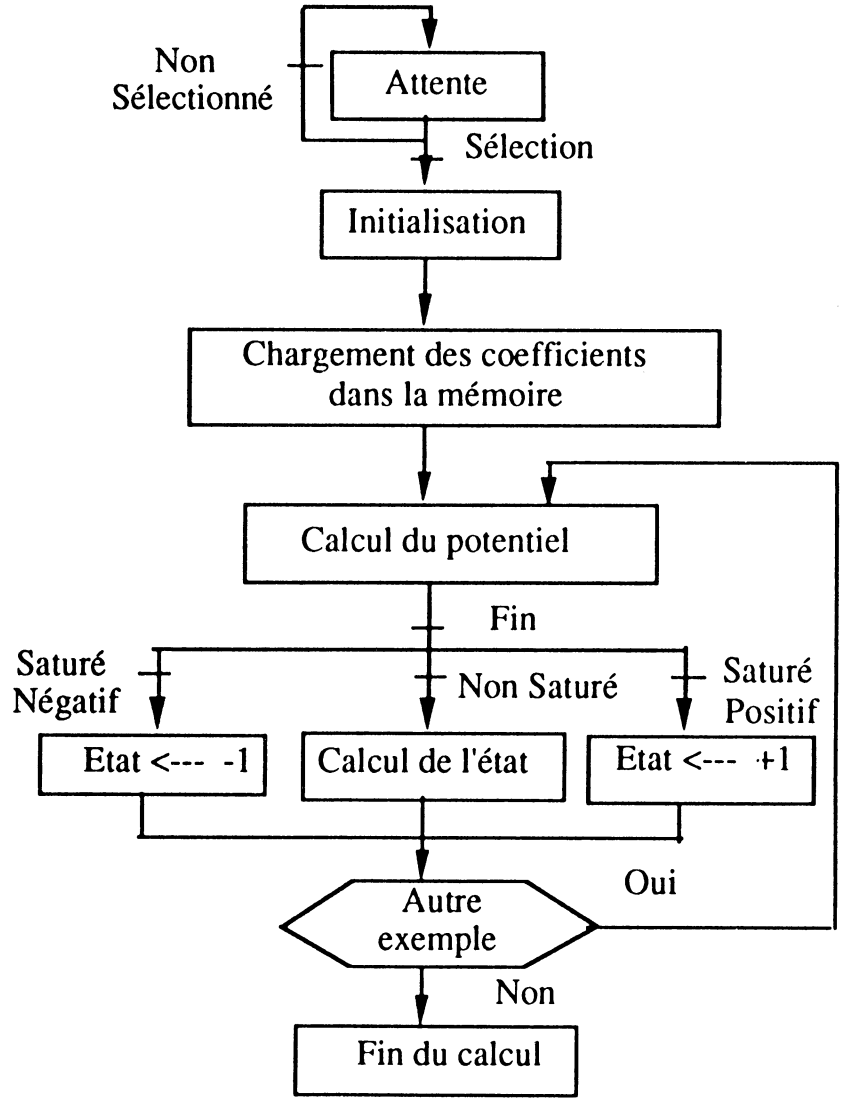


Figure 2.12 : Organigramme de contrôle en phase de reconnaissance

Un organigramme de contrôle s'exprime en terme de ressources de la partie opérative (registres, opérateurs, bus).

A) Initialisation du réseau :
Cette phase nécessite l'utilisation de 6 registres de personnalisation ainsi que de 2 compteurs (notés C2 et C3).

R5 registre contenant le nombre de décalage à effectuer pour formater le potentiel synaptique,

R6 registre contenant le nombre de décalages pour formater le résultat intermédiaire du calcul de la fonction d'activation,

R7 registre contenant le nombre de décalages pour formater le résultat de la dernière multiplication du calcul de la fonction d'activation,

R8 registre contenant le nombre de prédécesseurs du neurone,

R9 registre contenant le nombre de décalages nécessaires pour exécuter une vague de données,

R20 registre contenant les paramètres physiques du neurone, indices ligne - colonne et nature du décalage.

L'étape d'initialisation consiste à charger les registres de personnalisation et la mémoire contenant les coefficients synaptiques ainsi qu'à initialiser les compteurs (figure 2.13.a). Le chargement de ces valeurs s'effectue en sélectionnant une colonne de processeurs à la fois. C'est l'ordinateur hôte qui se charge d'envoyer un bit de sélection suivi de la liste des valeurs à charger. La partie contrôle doit engendrer les signaux nécessaires pour définir le mode d'écriture dans la mémoire (figure 2.13.b). Tous les chargements seront effectués par le bus bas.

On charge le registre R8 contenant le nombre de prédécesseurs du neurone ; puis les registres R5, R9 ; ensuite les registres R6, R7 ; on initialise en parallèle le compteur utilisé pour le calcul de l'adresse mémoire par la valeur de R8, puis un registre R20. L'étape suivante consiste à charger la mémoire par les valeurs des coefficients. Puis, on initialise les compteurs C3 et C2 respectivement par R8 et R9.

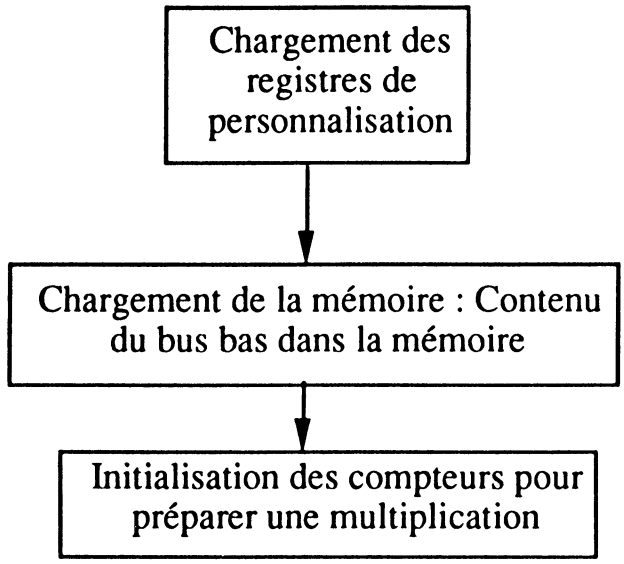


Figure 2.13.a : Algorithme d'initialisation du neurone

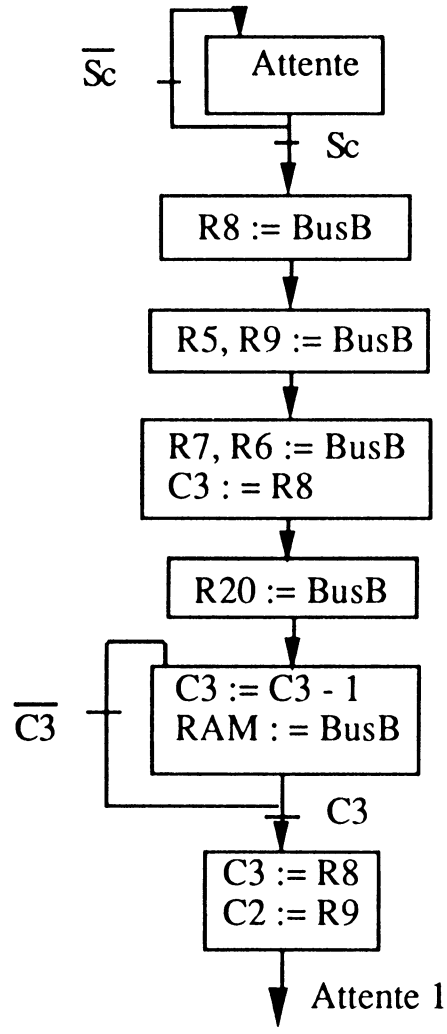


Figure 2.13.b : Organigramme d'initialisation du neurone

Après cette étape d'initialisation, tous les neurones d'une même colonne se mettent en état d'attente pour permettre l'initialisation des autres processeurs neurones.

Chargement des données :
Une fois le réseau initialisé, on procède au chargement des valeurs d'entrée. Ce chargement est effectué en plusieurs vagues. De même que pour les coefficients, on charge les données présentées sur les bus d'entrée, en sélectionnant une colonne de neurones à la fois.

Gestion des interrupteurs :
Les bus utilisés par les neurones sont bidirectionnels. A chaque extrémité de bus sont implantés des interrupteurs programmables. Un interrupteur est réalisé par deux buffers 3 états montés en tête bêche (figure 2.14).
A tout moment, le processeur neurone contrôle l'état de ses interrupteurs, dans l'étape de chargement, au cours du calcul du potentiel et pendant le transfert des données entre couches.

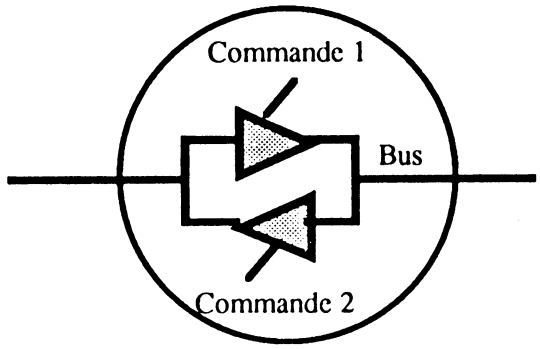


Figure 2.14 : Interrupteur programmable

B) Calcul du potentiel :
Le contrôleur doit assurer toutes les étapes de calcul du potentiel, à savoir (figures 2.15 et 2.16) :
- La lecture des coefficients dans la mémoire et la lecture de la valeur d'entrée sur le bus bas,
- La multiplication du coefficient par la valeur d'entrée,
- Le cumul du résultat de la multiplication dans l'accumulateur,
- La décrémentation des compteurs d'adresse mémoire et du nombre de décalages,
- Le décalage de la valeur d'entrée et simultanément la lecture d'une autre valeur après chaque multiplication,
- Le formatage du résultat final.

La figure 2.15 donne l'organigramme de contrôle de cette partie sans expliciter l'algorithme de Booth associé à la multiplication ; rappelons donc que ce type d'organigramme constitue la spécification de référence de l'architecture et qu'une synthèse quasiment automatique est enchaînée aisément.

Le chemin défini par la condition 1 est suivi lorsque le nombre de neurones et le nombre de vagues n'ont pas été totalement traités. Le neurone déclenche un décalage des données suivant sa position physique vers le haut ou vers le bas.

Dans le cas contraire, c'est le chemin défini par la condition 2 qui est suivi. La condition 3 est vérifiée lorsque la totalité des coefficients n'a pas été utilisée et que les données ont effectué un tour complet de la boucle physique.

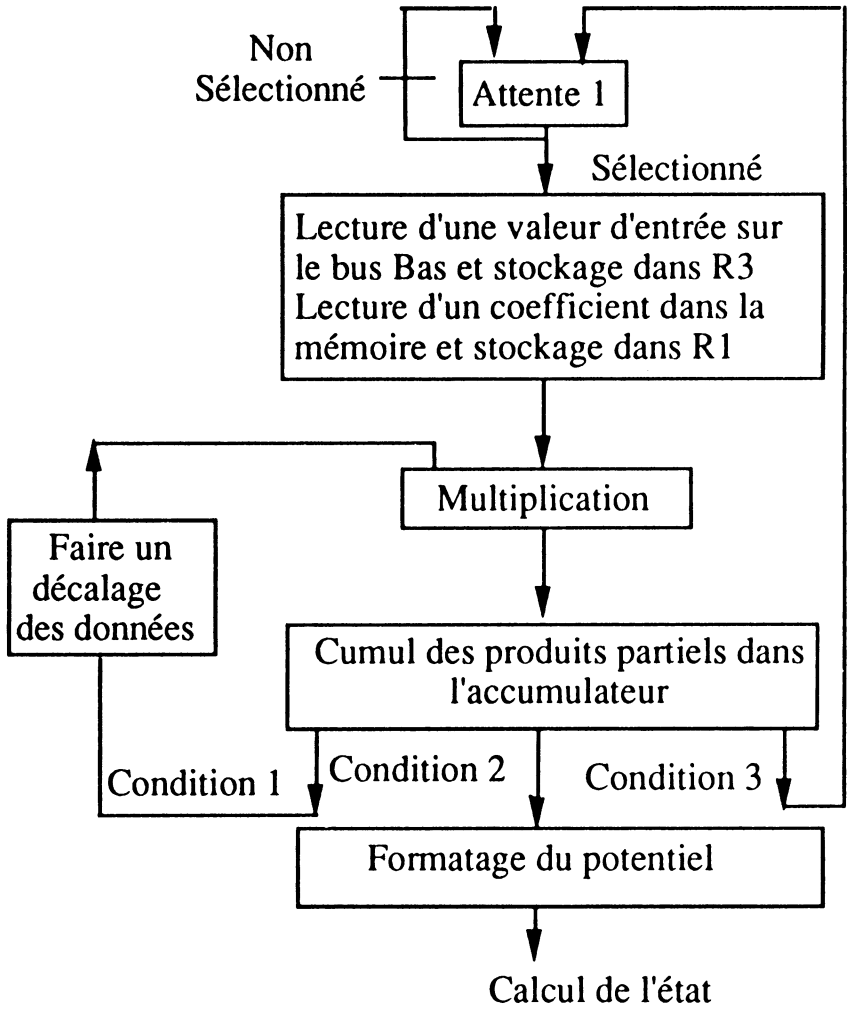


Figure 2.15 : Algorithme de calcul du potentiel

L'organigramme détaillé est donnée en figure 2.16.

Le calcul d'un produit partiel du potentiel est déclenché par le chargement d'un coefficient dans le registre R1 et l'une valeur d'entrée dans le registre R3. Le résultat de la multiplication est cumulé au contenu de l'accumulateur Acc, accumulateur des poids forts Acc_Fort et poids faible Acc_Faible. La donnée stockée dans le registre d'entrée RE est décalée vers le haut via le segment de bus haut (BusH). Si la donnée est décalée vers le bas, alors le segment de bus bas BusB sera utilisé.

Le formatage du résultat du potentiel est effectué par un décalage bit à bit du contenu de R2 vers R3.

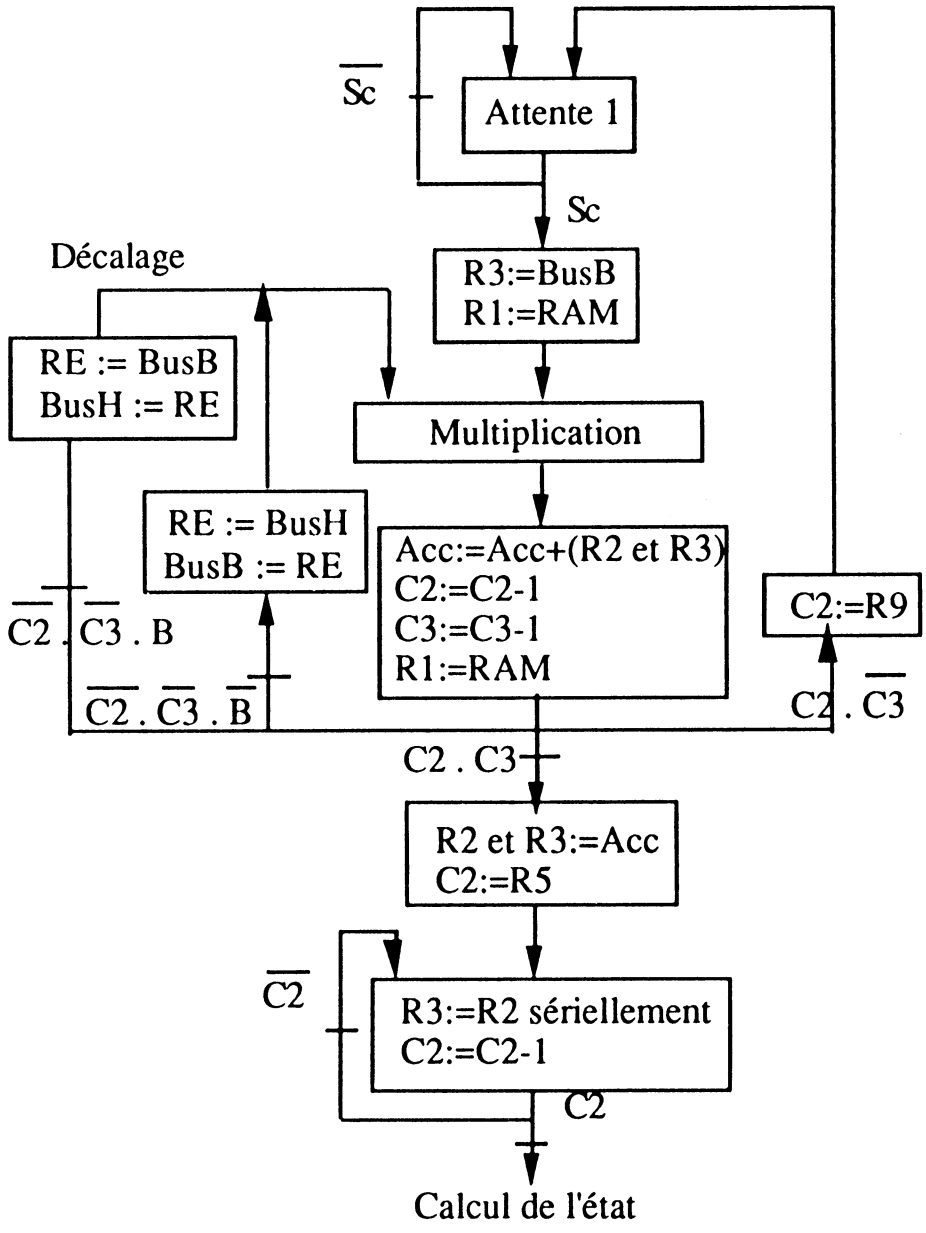


Figure 2.16 : Organigramme de calcul du potentiel

C) Fonction d'activation

Après le calcul du potentiel, le neurone actualise son état de sortie à travers une fonction d'activation. Le choix de cette fonction dépend de l'application à implanter ainsi que du mode d'apprentissage adopté. Les formes les plus connues sont à seuil, pseudo-linéaire et sigmoïdale. La fonction à seuil (figure 2.17.a) est utilisée pour des réseaux simples comme celui de Hopfield [Hop82] ou le perceptron [Lip87] [Tre89a]. Le format des données utilisé par ces réseaux peut être binaire ou sur plusieurs bits.

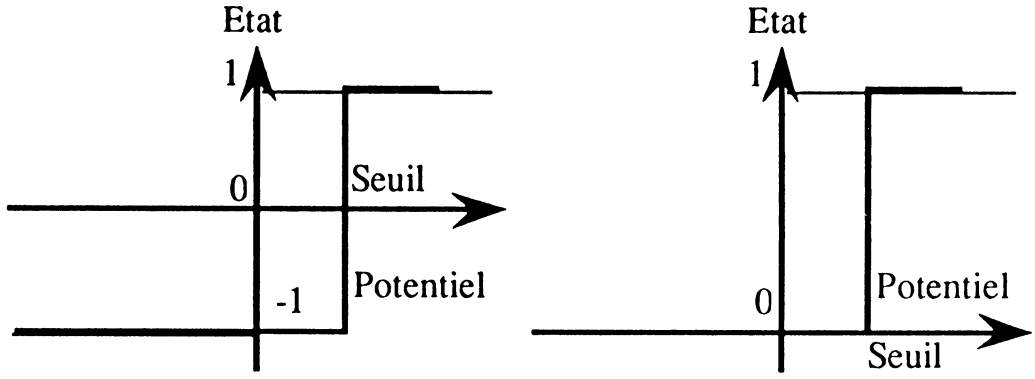


Figure 2.17.a : Fonction d'activation à seuil

La fonction pseudo-linéaire (figure 2.17.b) est utilisée par le réseau à Perceptron utilisant la règle Delta [Hin87] pour l'apprentissage et par le réseau connu sous le nom "Neocognitron" [Fuk88]. Le format des données est uniquement sur plusieurs bits.

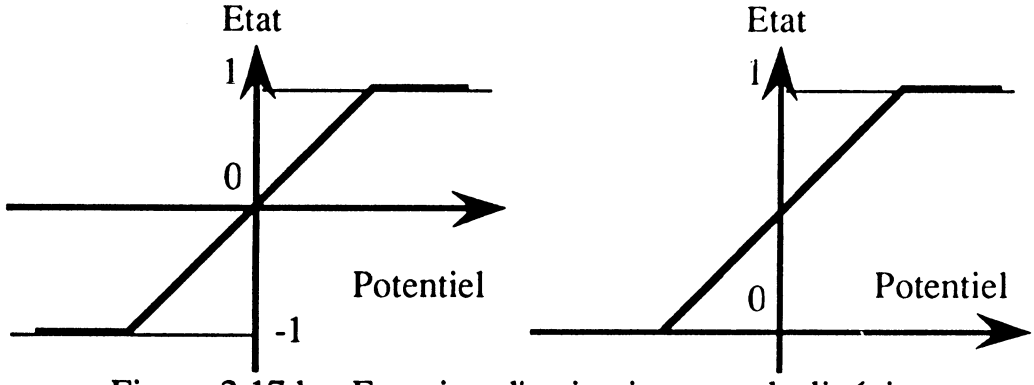


Figure 2.17.b : Fonction d'activation pseudo-linéaire

La fonction sigmoïdale (figure 2.17.c) est souvent utilisée dans les réseaux utilisant la rétropropagation et aussi "self organising map" [Lip87]. Le format des données est sur plusieurs bits.

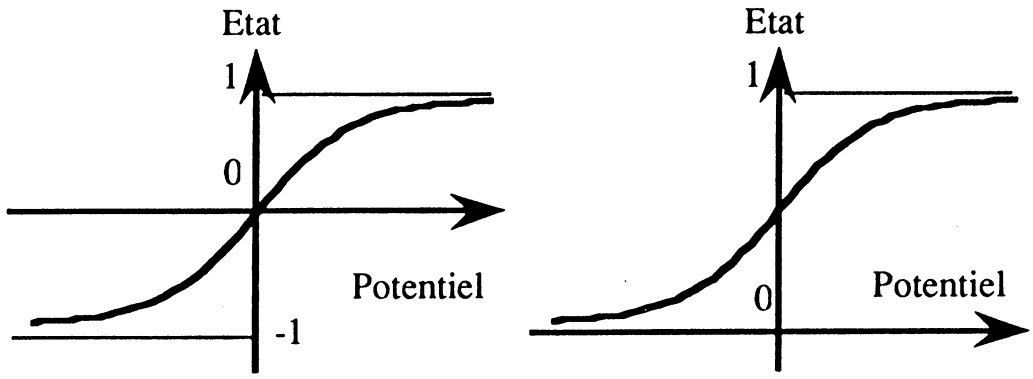


Figure 2.17.c : Fonction d'activation sigmoïdale

Les deux premières fonctions ne présentent aucune difficulté pour l'implantation sur silicium. Par contre, la fonction sigmoïdale pose un problème, car les fonctions exponentielles sont plus difficiles à réaliser. On peut implanter cette fonction, soit à l'aide d'une mémoire tabulée [Mye89] soit la câbler directement [Ali90].

La première solution est très simple et plus directe, tant que la précision demandée sur la valeur de l'état n'est pas très importante. Si on veut avoir une précision plus grande, il faut augmenter le nombre des valeurs échantillonnées, ce qui augmente automatiquement la taille de la mémoire nécessaire pour stocker les valeurs des états.

L'état étant codé sur n bits, le nombre de valeurs d'états possible est de 2^n-1 qui correspond à un potentiel appartenant à l'intervalle [-L, L]. Si l'on souhaite un maximum de flexibilité pour la sigmoïde, il convient de charger ces valeurs dans une RAM statique, adressée directement par la valeur du potentiel calculé. Le nombre d'adresses dépend de la longueur de l'intervalle et du pas d'échantillonage. En supposant que le potentiel est aussi codé sur n bits, alors le nombre d'adresses serait égal à $2^n.L$.

Sachant qu'un point mémoire est constitué de 6 transistors et en prenant une technologie CMOS 2µ, dont on estime que la densité moyenne d'intégration est de l'ordre de 2000 transistors/mm^2, on a tracé sur la figure 2.18 la surface de la mémoire en fonction du nombre de bits et de la borne L.

Pour des précisions plus fines avec un codage de 16 et 32 bits la surface de la mémoire est rédhibitoire. Ainsi, la seconde solution, la fonction câblée, est plus intéressante, puisque, quelle que soit la valeur du potentiel, elle offre une précision optimale en fonction du format utilisé.

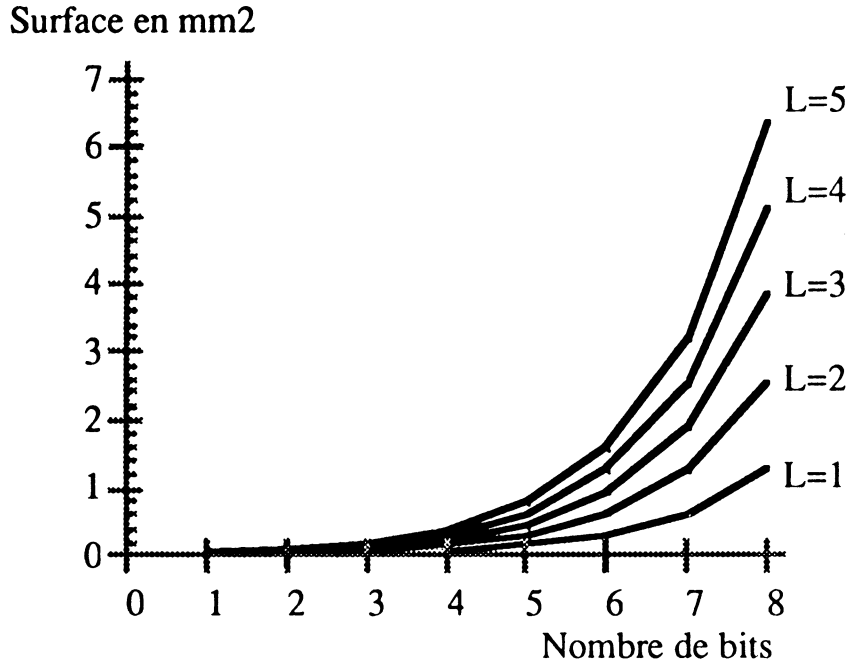


Figure 2.18: Evaluation de la taille de la mémoire tabulée

Le problème est donc de trouver une fonction simple qui puisse approximer la sigmoïde et qui soit facilement intégrable sur silicium. Le choix s'est porté sur une fonction symétrique par rapport à l'origine, dont l'équation est donnée par la formule (1).

$$Ft(V) = (1 - \exp(-\beta V)) / (1 + \exp(-\beta V)) \quad (1)$$

où $\beta/2$ représente la pente de la courbe à l'origine et V le potentiel.

L'approximation la plus simple, répondant aux critères posés, est un polynôme de la forme $\Sigma\, a_i.V^i$ où a_i sont les poids. Ce genre de polynôme donne une bonne approximation de la fonction théorique à partir d'un degré égal à 5. Ceci implique qu'il faut effectuer 5 multiplications, 5 additions et prévoir, aussi, autant de registres pour stocker les paramètres a_i et ceux utilisés pour le formatage après chaque multiplication.

Une deuxième idée consiste à chercher un polynôme de degré inférieur, qui non seulement approxime la fonction théorique de façon satisfaisante, mais aussi optimise les ressources utilisées et le nombres d'opérations effectuées. La fonction approximée proposée (figure 2.19) est une partie de parabole, système d'ordre 2, à laquelle on a imposé des conditions particulières :

- un seuil de saturation du potentiel,

- une valeur saturée de l'état.

$$
\begin{aligned}
F_{ap}(V) &= -a(Vs - V)^2 + 1 && \text{si } 0 \leq V < Vs \\
F_{ap}(V) &= +1 && \text{si } V \geq Vs \\
F_{ap}(V) &= -1 && \text{si } V \leq -Vs \\
F_{ap}(V) &= a(Vs + V)^2 - 1 && \text{si } -Vs < V \leq 0
\end{aligned}
\qquad (2)
$$

Fap représente la fonction d'activation approximée, le paramètre a représentant la pente de la courbe et Vs le potentiel de saturation à partir duquel on estime que la valeur de l'état ne varie plus quelle que soit la valeur du potentiel V.

On peut remarquer que le calcul de cette expression polynomiale ne nécessite que 2 multiplications, 2 additions et seulement 2 registres pour stocker les paramètres a et Vs. Ceci représente une optimisation en ressources et en temps de calcul.

Mais, ce qui est plus intéressant, c'est que ce polynôme peut approximer la fonction théorique quelle que soit sa pente. En effet, avec un choix judicieux des paramètres a et Vs, on peut minimiser l'écart entre la fonction théorique et la fonction approximée.

On sait que la fonction théorique dépend du facteur β, et la fonction approximée dépend de a et de Vs. On a essayé de trouver une solution déterminant l'écart optimum entre les deux fonctions, en tenant compte de ces trois paramètres.

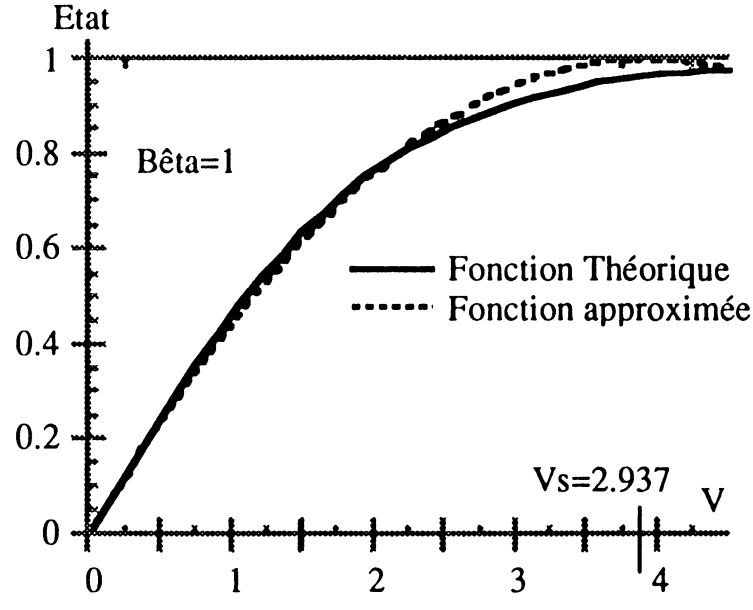


Figure 2.19: Représentation de la partie positive des deux fonctions

Une solution théorique réunissant les trois paramètres à la fois est difficile à formuler. On se propose d'étudier quelques exemples particuliers résolus par des méthodes numériques.

En fixant la pente, on étudie la variation de l'écart entre les deux fonctions théorique et approximée (Fap - Ft) suivant les différentes valeurs du potentiel de saturation. Ce dernier est déterminé à partir de l'erreur E tolérée sur la valeur de l'état à l'infini, Fap(Vs) = 1 - E. La dérivée de la fonction Fap-Ft admet deux extrêmes qui varient en fonction du potentiel de saturation.

Le but est de trouver le potentiel de saturation Vs qui minimise les bornes de l'erreur ; il convient donc d'assurer 2 valeurs d'erreur pour les deux extrêmes. On considère que les exemples traités ci-dessous ont une erreur E qui varie entre 2 et 8% de la valeur maximale de l'état.

Dans le premier cas, on considère que la pente est égale à 0,5 et que le potentiel variant entre 4,2 et 3 (figure 2.20). On peut remarquer qu'il existe un potentiel de saturation, égal à 3,93, pour lequel l'écart de 4% entre les 2 fonctions est minimum et que la courbe correspondante est symétrique par rapport à un potentiel égal à 2.

Pour les autres valeurs de potentiels de saturation, les écarts sont dissymétriques et plus importants en valeurs absolues.

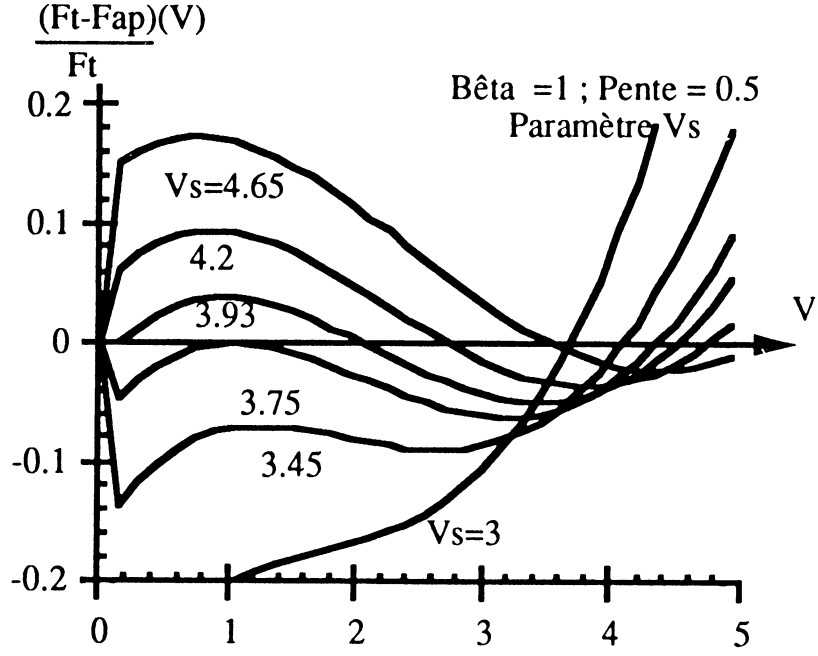


Figure 2.20 : Erreur entre les deux fonctions pour une pente de 0,5

Dans le deuxième cas, la pente est égale à 0,75 et le potentiel varie entre 3,15 et 2,25 (figure 2.21). On remarque qu'à une valeur de Vs égale à 2,55 correspond une erreur minimum de 3,3%.

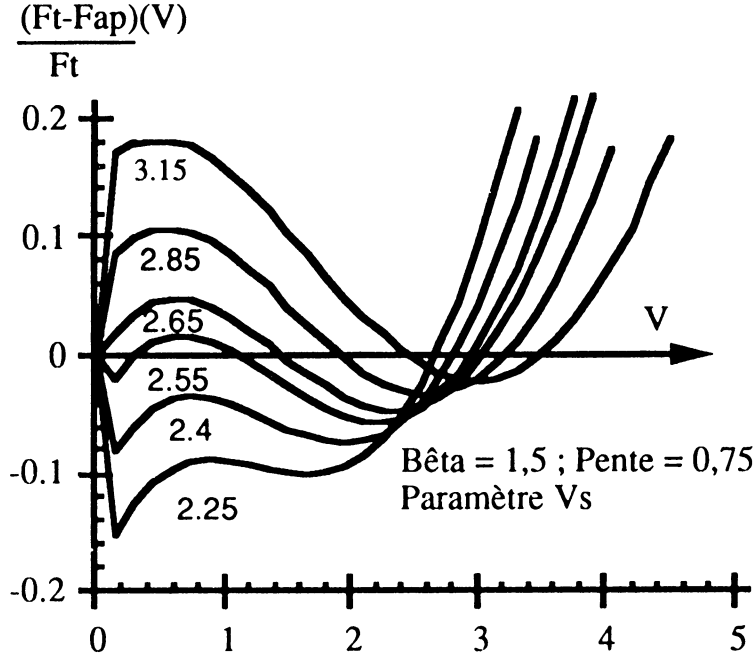


Figure 2.21 : Erreur entre les deux fonctions pour une pente de 0,75

Dans le troisième cas, il est évident que pour les mêmes valeurs d'erreur, et une pente plus forte égale à 1, le potentiel de saturation devient plus faible et varie entre 2,4 et 1,65 (figure 2.22). L'erreur minimum de 4% correspond à un potentiel de saturation égale à 1,95.

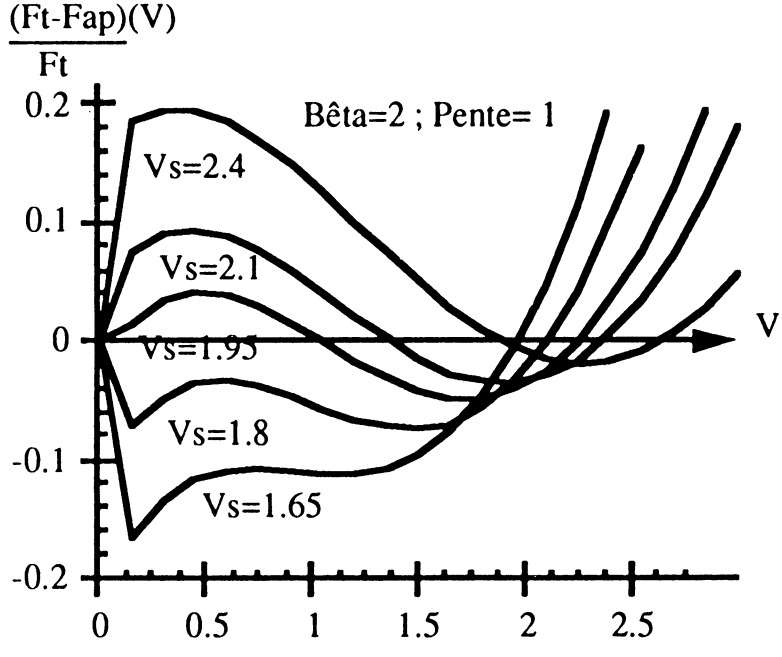


Figure 2.22 : Erreur entre les deux fonctions pour une pente de 1

Pour tous ces exemples, la valeur de l'erreur maximum entre la fonction sigmoïde théorique (Ft) et la fonction approximée (Fap), pour un potentiel de saturation correctement choisi, est de ± 4%. Cette erreur représente une imprécision entre 2^{-5} et 2^{-6}. Un nombre de bits égal à 6 parait donc le minimum pour coder les états de la fonction approximée.

Par ailleurs, des simulations, utilisant cette fonction approximée, de réseaux classifieurs de points dans un plan ont été menées en collaboration avec l'ESPCI. Trois types de réseaux ont pu être étudiés. Le premier type de réseau classifieur de 2 classes apprend 8 exemples (figure 2.23.a), le deuxième apprend 25 exemples (figure 2.23.b). Le troisième réseau classifieur de 5 classes apprend 5 exemples (figure 2.23.c).

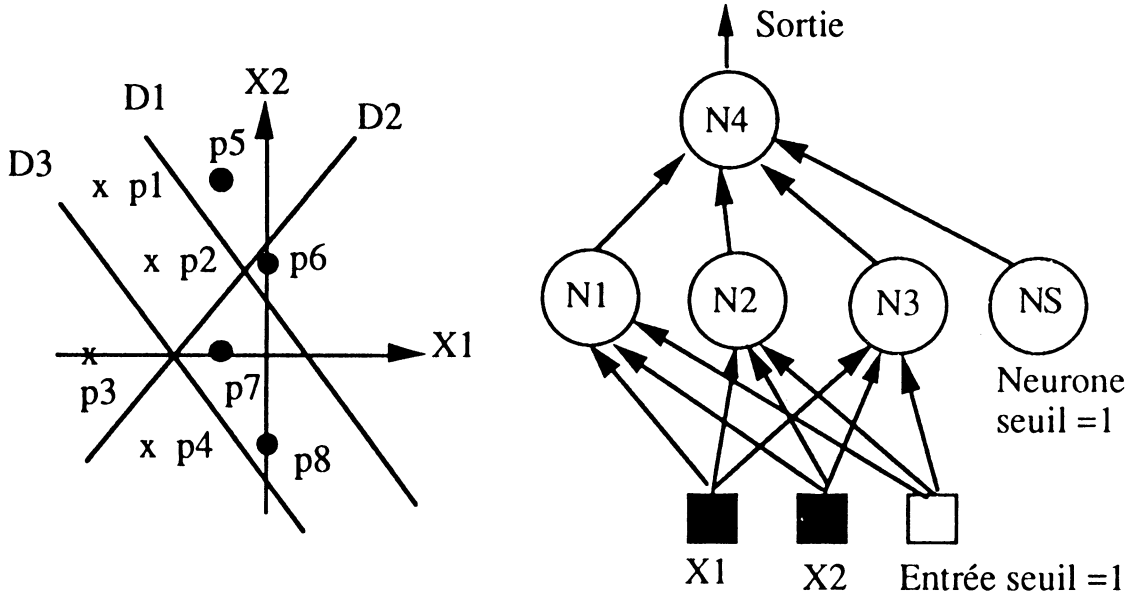


Figure 2.23.a : Réseau classifieur de 2 classes, 8 exemples appris

Les prototypes de p1 à p4 appartiennent à la classe 1 et ceux de p5 à p8 appartiennent à la classe 2. Dans la phase de reconnaissance les vecteurs présentés appartiennent à une partie du plan.

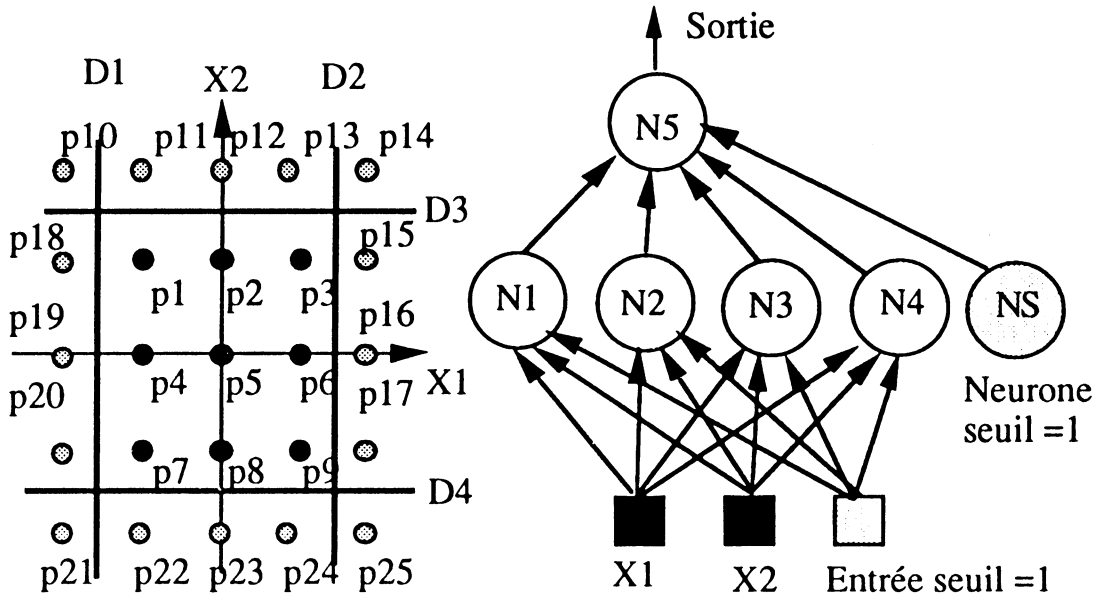


Figure 2.23.b : Réseau classifieur de 2 classes, 25 exemples appris

La sortie est égale à 1 pour les prototypes de p1 à p9 appartenant à la classe 1. La sortie est égale à -1 pour les prototypes de p10 à p25 appartenant à la classe 2.

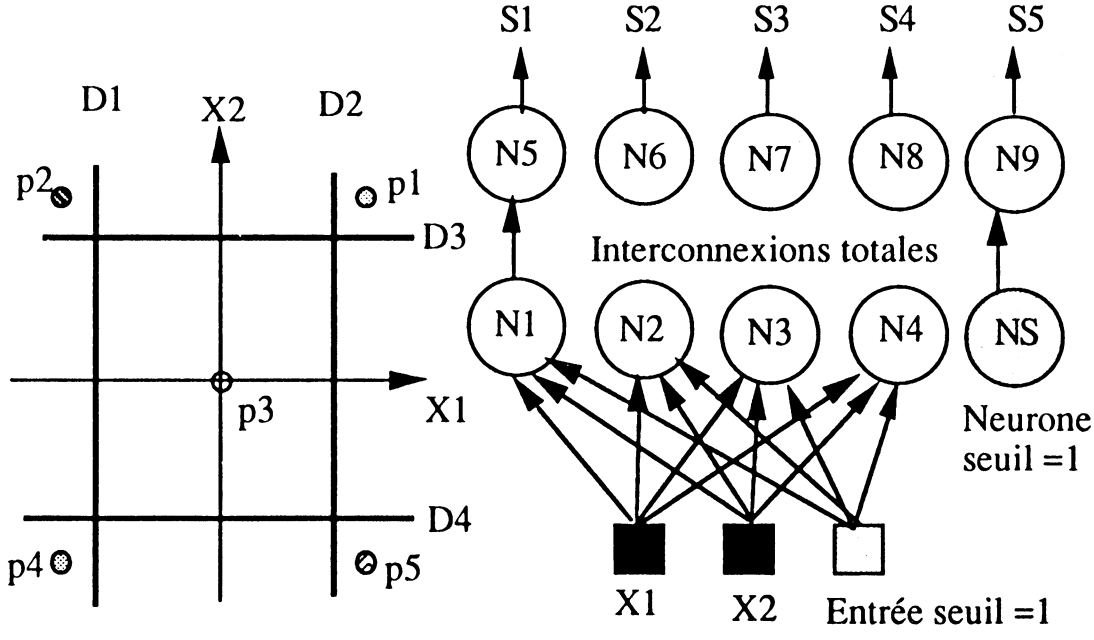


Figure 2.23.c : Réseau classifieur de 5 classes, 5 exemples appris

p1 ∈ à la classe 1, la sortie sera égale à S (+1, -1, -1, -1, -1).
p2 ∈ à la classe 2, la sortie sera égale à S (-1, +1, -1, -1, -1).
p3 ∈ à la classe 3, la sortie sera égale à S (-1, -1, +1, -1, -1).
p4 ∈ à la classe 4, la sortie sera égale à S (-1, -1, -1, +1, -1).
p5 ∈ à la classe 5, la sortie sera égale à S (-1, -1, -1, -1, +1).

Les entrées du réseau sont les coordonnées des points dans le plan. Le nombre de sorties du réseau dépend du nombre de classes à reconnaître. La classification est réalisée à l'aide d'une séparation des classes par des droites. Le nombre de ces droites varie en fonction des zones définissant les classes.

Ces simulations ont montré qu'en phase de reconnaissance utilisant notre approximation de la fonction d'activation, il suffit de coder les états sur 6 bits pour avoir des résultats de classification acceptables.

Si le format des états est codé sur plus de 6 bits, on aura une meilleure précision. Sinon, le nombre de points mal classés sera d'autant plus grand que la précision est faible.

Ceci est seulement valable pour les exemples de classification traités. Etant donné que notre architecture est paramétrable, il est possible d'implanter d'autres applications, en précisant par des simulations précises le nombre de bits et les paramètres de la fonction d'activation.

La comparaison de V à Vs ou -Vs est réalisée simplement par un circuit combinatoire à la sortie de l'accumulateur des poids forts.

Illustrons ce point sur un exemple de calcul de potentiel synaptique. Supposons que n1 bits codent la partie décimale des états, n2 bits codent la partie décimale des coefficients et n3 bits codent la partie décimale du potentiel de saturation.

Le produit de l'état par le coefficient, ainsi que le potentiel final, aura une partie décimale codée sur n1+n2 bits, en supposant que le format choisi peut coder la plus grande valeur déterminée par l'apprentissage. Si la valeur du potentiel calculé est inférieure à la valeur de saturation, le neurone doit calculer son état de sortie, sinon le neurone est saturé et il est inutile de calculer l'état.

Dans les deux cas, le neurone teste la valeur de saturation. Afin de faire une comparaison logique, il faut que le potentiel calculé (codé sur 2.n bits) et le potentiel de saturation (codé sur n bits) aient le même format d'écriture (emplacement de la virgule fixe et nombre de bits). Pour ceci, on est amené à effectuer un formatage du potentiel calculé par un décalage de ces n1+n2-n3 bits de poids fort de l'accumulateur des poids forts (Acc-Fort) vers l'accumulateur des poids faibles (Acc-Faible). De cette façon, en prenant les n3 bits égaux à 1, on vérifie facilement les cas suivants :

-si tous les bits de l'Accu-Fort sont égaux à 0, le potentiel est non saturé et il est positif. Le neurone calcule son état.

-si tous les bits de l'Accu-Fort sont égaux à 1, le potentiel est non saturé et il est négatif. Le neurone calcule son état.

- si les bits de l'Accu-Fort ne sont pas tous identiques, le potentiel est saturé. Le neurone doit afficher sa valeur de saturation suivant son bit de signe.

Il est important de préciser que le formatage du potentiel est fait uniquement sur le résultat final. Par contre, pour d'autres multiplications, par exemple lors du calcul de l'état, le formatage est nécessaire après chaque opération.

Illustrons par un exemple la nécessité du formatage. On suppose qu'un neurone est connecté à 2 prédécesseurs. Ce neurone doit calculer $C_{11}.E_1 + C_{12}.E_2$ avec

$C_{11} = 2,75$ codé sur 8 bits : 0 0 1 0 ,1 1 0 0

$C_{12} = 1,625$ codé sur 8 bits : 0 0 0 1, 1 0 1 0

$E_1 = + 0,75$ codé sur 8 bits : 0, 1 1 0 0 0 0 0

$E_2 = - 0,375$ codé sur 8 bits : 1, 1 0 1 0 0 0 0

Le résultat du potentiel, égal à 3,4375, est codé sur 16 bits comme suit :

0 0 0 0 1, 0 1 1 1 1 1 0 0 0 0 0

Le potentiel de saturation est égal à 3,875 codé en 8 bits : 0 1 1, 1 1 1 1 1

Afin de comparer ces deux dernières valeurs il faut tout d'abord ramener le résultat du potentiel au même format que le potentiel de saturation et non pas l'inverse, ceci pour deux raisons. D'une part, si le neurone est non saturé, on doit calculer l'état du neurone en utilisant un format de donnée accepté par l'unité de traitement (dans le présent exemple 8 bits). Ceci implique automatiquement qu'il faut tronquer les bits les moins significatifs. Et d'autre part, en effectuant le formatage, la comparaison devient très facile à réaliser.

Dans notre exemple, 8 bits sur 16 nous intéressent (00**001,011**11100000) alors on doit effectuer 5 décalages vers la droite. La valeur contenue dans les accumulateurs devient *0 0 0 0 0 0 0 0* **0 0 1, 0 1 1 1 1.**

Accu-Fort ↑ Accu-FAible

Les bits des poids forts sont tous égaux à 0, donc le neurone est non saturé avec un potentiel positif.

La figure 2.24.a donne l'algorithme macroscopique du calcul de l'état. Le formatage du potentiel et le test de la saturation sont effectués. Suivant le résultat, le neurone affiche en sortie un état de saturation si l'un des deux potentiels de saturation est atteint ; sinon le neurone calcule la fonction d'activation.

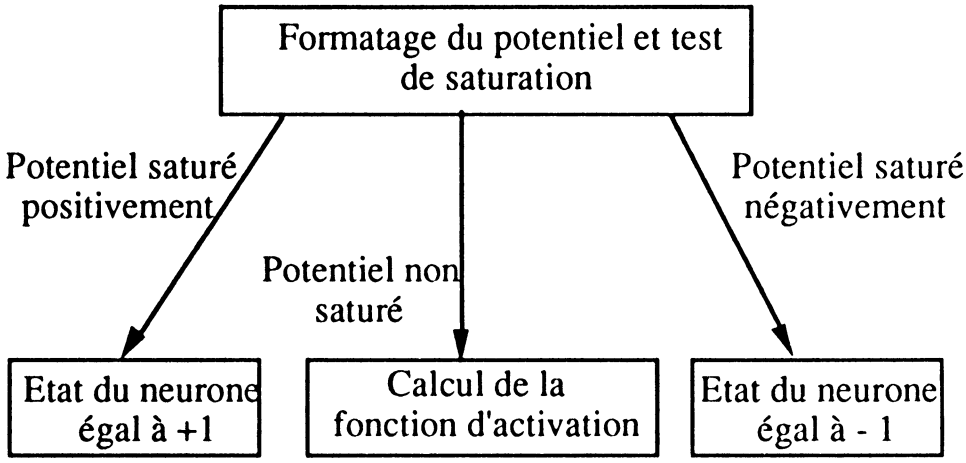


Figure 2.24.a: Organigramme macroscopique du calcul de l'état

Vs désigne le potentiel de saturation, le bit de signe du potentiel est stocké dans le registre SB et F est un registre utilisé pour distinguer les deux multiplications de -a par $(Vs - V)^2$ et a par $(Vs - V)^2$. La figure 2.24.b donne l'algorithme implanté pour le calcul de l'état du neurone.

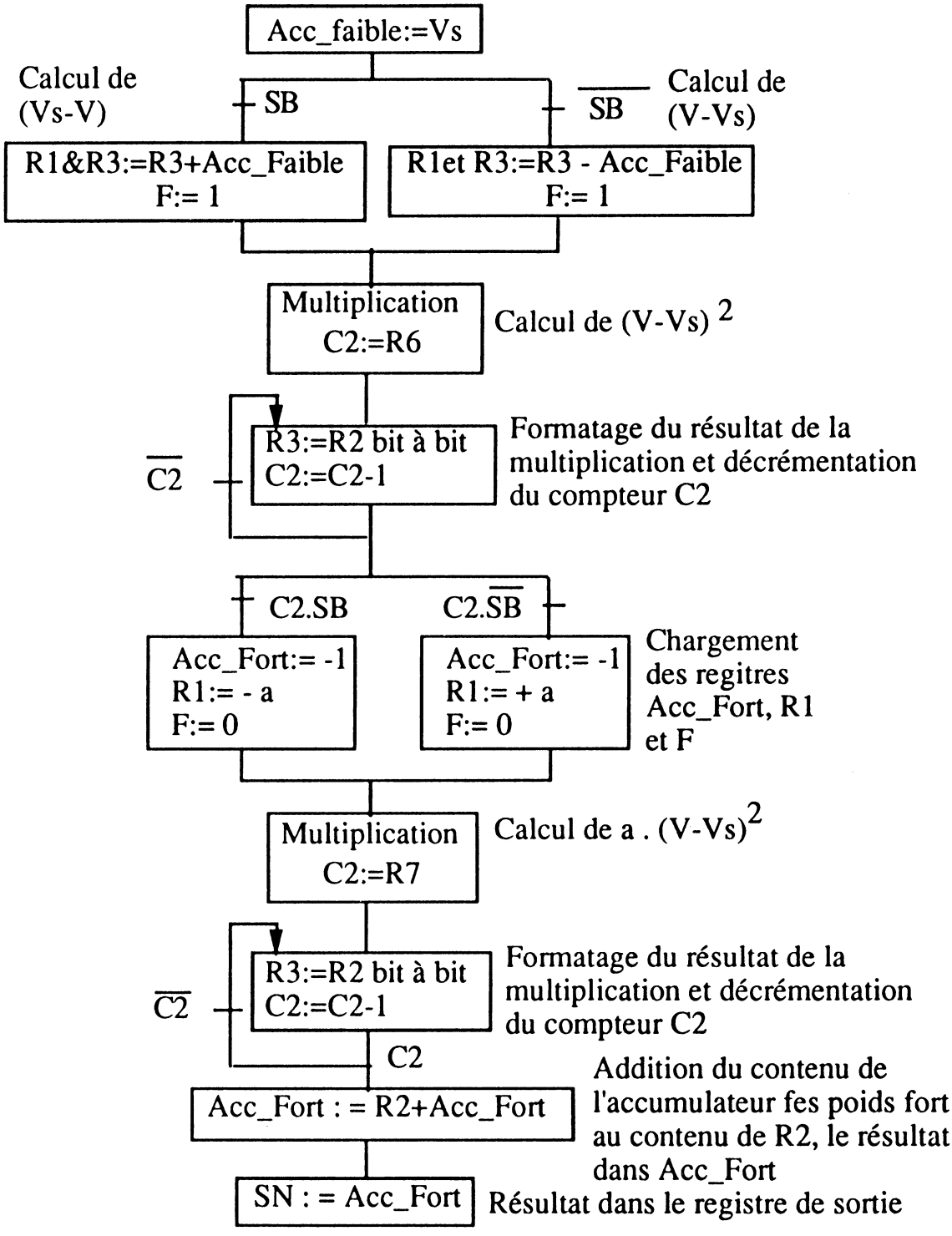


Figure 2.24.b : Organigramme de calcul de l'état du neurone

2.5 Réalisation sur silicium

2.5.1 Outil de conception

Le logiciel de CAO utilisé est celui de VLSI Technology Inc. Il contient des bibliothèques de cellules précaractérisées, des générateurs de RAM, de ROM et de PLA ainsi qu'un outil de synthèse de machines d'états finis et un compilateur de chemin de données.

La technique habituelle d'extraction de graphe d'états finis d'un organigramme de contrôle est illustrée en figure 2.25 ; de façon simplifiée, on montre ce procédé pour un automate de Moore ; les signaux SRi, ERi représentent respectivement les signaux de contrôle commandant les sorties et les entrées des registres, Cop les commandes de la partie opérative.

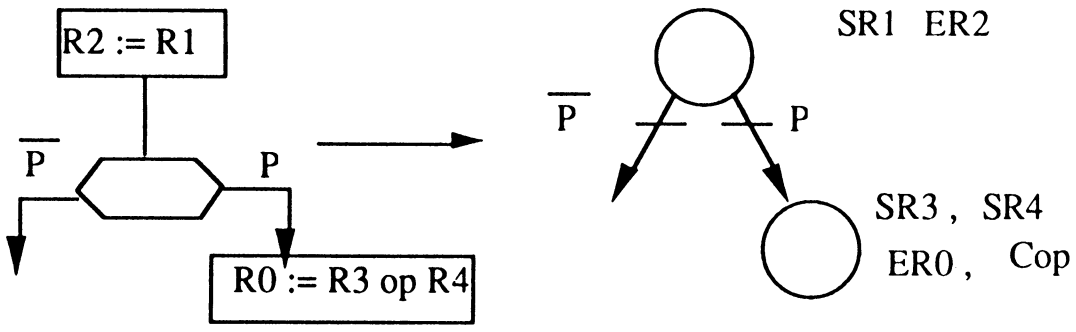


Figure 2.25: Extraction d'un graphe d'état à partir d'un organigramme

Le graphe d'état du processeur neurone est un graphe de 24 états et 39 transitions. A partir de ce graphe d'état, le logiciel de synthèse fournit en sortie un réseau de cellules standard implantant :

- Les équations des sorties,

- Les équations des variables internes,

- Les bascules qui mémorisent l'état,

- Les verrous (latches) synchronisant les entrées/sorties.

Pour la partie "chemin de données", une description en réseaux de blocs de la bibliothèque "chemin de données" (registre, additionneur, soustracteur...) est donnée au système de synthèse.

Cette description est paramétrée en fonction du nombre de bits. Elle ne nécessite que la spécification d'une tranche de 1 bit. Le générateur produit la réalisation finale avec un plan de masse réalisant l'aboutement des tranches de bits.

La mémoire est crée par un générateur de RAM en précisant la taille et le nombre des mots.

Un logiciel appelé "Chip compiler" réalise l'assemblage des blocs et la création du dessin du circuit final. Le placement des blocs est manuel et le routage est automatique. Le concepteur peut définir certaines contraintes entre les différents éléments du circuit pour mieux contrôler le compactage.

2.5.2 Circuit fabriqué

Le premier circuit réalisant un processeur neurone et implantant une mémoire de 64 octets, un contrôleur et une partie opérative de 8 bits (coefficients et états codés sur 8 bits) a été réalisé en technologie CMOS 2 µ. Ce circuit a été fabriqué et fonctionne correctement avec une horloge de 20 MHz. On distingue trois blocs distincts, deux blocs durs représentent la mémoire et la partie "chemin de données" et un bloc de cellules standard représentant la partie contrôle. Les tailles respectives de ces blocs sont 1,16 mm^2, 3,37 mm^2 et 5,53 mm^2. La surface totale incluant le routage est de 16 mm^2 (figure 2.26).

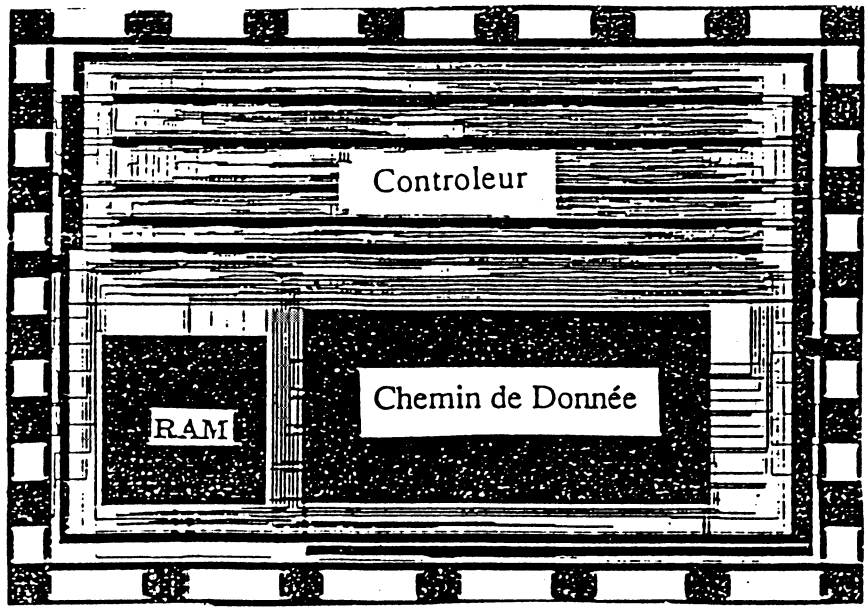


Figure 2.26 : Dessin des masques du circuit fabriqué

2.5.3 Validation et test du circuit

2.5.3.1 Programme de validation

Le test de validation d'un circuit consiste à vérifier la conformité de la réalisation à une spécification de référence. Son but est de détecter des erreurs de conception et en aucun cas de détecter des fautes de fabrication. Il ne s'agit donc pas de test déterministe sur hypothèse de défauts (de fabrication) mais de test de conformité. Il ne s'agit pas non plus de remettre en cause les outils de synthèse automatisés du fondeur qui sont largement validés.

La spécification de référence est ici l'organigramme de contrôle définissant le neurone et le test de validation consistera à passer par tous les chemins du graphe de contrôle. En termes plus intuitifs, toutes les fonctionnalités offertes par le neurone sont validées. En examinant l'organigramme de contrôle, on s'aperçoit qu'il s'agit essentiellement de tester 3 chemins "globaux" correspondant respectivement aux cas de :

- La saturation négative où l'état est égal à -1,
- La saturation positive où l'état est égal à +1,
- La non saturation où l'état est entre -1 et +1.

puis de tester à l'intérieur du calcul du potentiel l'enchaînement du calcul des différents termes partiels.

a) Chemins globaux :
Les vecteurs de test sont choisis de façon à passer par les trois chemins. Nous donnons de façon exacte le programme de validation exécuté en terme de signaux de contrôle appliqués. Ces signaux de contrôle sont les suivants :

- R correspond au signal d'initialisation (reset) général du neurone
- D indique que le décalage des données doit être effectué vers le haut si D=1 ou vers le bas si D=0.

Le numéro du cycle est mentionné dans la colonne "Cycle". On définit aussi les abréviations suivantes :

E Bus B : Entrée des valeurs sur le bus bas,

E Bus H : Entrée des valeurs sur le bus haut,

S Bus B : Sortie des valeurs sur le bus bas,

S Bus H : Sortie des valeurs sur le bus haut.

Dans les exemples choisis, on considère que la partie entière codant les coefficients, respectivement les états, est sur 4 bits et sur 1 bit, sachant que le format des données est sur 8 bits. A partir du paragraphe 2.3.3, on détermine les valeurs des paramètres de formatage dans les registres R5 = R6 = 5 et R7 = 3. Rappelons que :
R5 : registre contenant le nombre de décalages à effectuer pour formater le potentiel synaptique,
R6 : registre contenant le nombre de décalages pour formater le résultat intermédiaire $(V-Vs)^2$ du calcul de la fonction d'activation,
R7 : registre contenant le nombre de décalages pour formater le résultat de la dernière multiplication $a.(V-Vs)^2$ du calcul de la fonction d'activation,
R8 : registre contenant le nombre de prédécesseurs du neurone,
R9 : registre contenant le nombre de décalages nécessaires pour exécuter une vague de données.

Le premier chemin vérifie le cas de saturation positive. Les valeurs du coefficient et de l'entrée sont chargées sur l'entrée du bus bas du neurone. Le résultat est sur le bus de sortie bas (S Bus B). La valeur du coefficient choisie arbitrairement et permettant d'atteindre la saturation positive est égale à 7,9375 (codée sur 8 bits 0111,1111) et celle de l'état est égale à 0,99216 (codée sur 8 bits 0,1111111). Les valeurs de R5 et R9 sont chargées en un seul cycle par le même bus d'entrée des données. De la même façon on charge aussi R7 et R6. Dans cet exemple, la virgule fixe est toujours à la même position.

Cycle	E Bus B	S Bus B	Opérations effectuées
0	xxxxxxxx	xxxxxxxx	Initialisation
1	00**000001**		Chargement de R8
2	**0101**0001		**R5= 5**, R9=1
3	**0011**0101		**R7= 3**, R6=5
4	0111,1111		Coefficient = 7,9375
5	0,1111111		Entrée = 0,99216
29	0,1111111	0,1111111	Sortie = + 0,99216

L'état de sortie du neurone correspond à un état de saturation positive conformément au prévision faite par le choix des 2 valeurs correspondantes respectivement aux valeurs maximales du coefficient et état (codée sur 8 bits 01111111).

Le cas de saturation négative est testé en chargeant la valeur d'entrée à - 0,99216 (codée sur 8 bits 1,0000001) au cycle 5. Cette valeur est choisie pour atteindre un minimum négatif de la fonction d'activation.

Cycle	E Bus B	S Bus B	Opérations effectuées
0	xxxxxxxx	xxxxxxxx	Initialisation
1	00000001		Chargement de R8
2	**01**010001		**R5= 5**, R2=1
3	**00**110101		**R7= 3**, R6=5
4	0111,1111		Coefficient = 7,9375
5	1,0000001		Entrée = - 0,99216
29	1,0000001	10000001	Sortie = - 0,99216

Les valeurs des états saturés négatif (10000001) et positif (01111111) sont cablées dans le chemin de données à l'aide des dispositifs définis dans le §4.2.1.1 ; rappelons que celles ci permettent de figer ces valeurs à l'entrée du registre de sortie du processeur neurone.

Le cas de non saturation a été testé en envoyant deux séquences de vecteurs. La première simule le cas de non saturation positive. Les 2 premières valeurs coefficient (2) et état (0,8515) sont choisis arbitrairement pour simuler le cas de non saturation positive.

Cycle	E Bus B	S Bus B	Opérations effectuées
0	xxxxxxxx	xxxxxxxx	Initialisation
1	00000001		Chargement de R8
2	**01**010001		**R5= 5**, R2=1
3	**00**110101		**R7= 3**, R6=5
4	0010,0000		Coefficient = 2
5	0,1101101		Entrée = + 0,8515
109	0,1010101	0,1010101	Sortie = 0,664

La seconde séquence teste le chemin correspondant à une non saturation négative. L'étape d'initialisation est identique à la précédente, mais les valeurs du coefficient et de l'état correspondent respectivement à 2,437 (codée sur 8 bits 0010,0111) et - 0,6875 (codée sur 8 bits 1,0101000). Le résultat attendu est égal à - 0,6718 (codée sur 8 bits 1,0101010).

b) Test d'enchaînement des calculs dans le processeur
Dans la seconde partie du test, on connecte le neurone à trois autres neurones. Le test consiste à vérifier à la fois les calculs effectués par le processeur neurone ainsi que le bon déroulement de la circulation des données. Les données sont toujours chargés par le bus bas (E Bus B).

Dans ce premier exemple, on suppose que le décalage des données s'effectue de bas en haut. Les coefficients sont chargés à partir du cycle 4. Au cycle 8, on charge la première valeur d'entrée. Après avoir calculé le premier produit partiel, on charge simultanément la deuxième valeur d'entrée sur le bus E Bus B et l'on décale la première valeur utilisée vers le bus haut (E Bus H) au cycle 60. De la même façon, on charge la troisième valeur d'entrée et on décale la deuxième au cycle 112. Au cycle 164, on charge la quatrième valeur d'entrée et on décale la troisième. Le résultat est disponible au cycle 273 sur le bus de sortie bas (S Bus B).

On remarque que le décalage des données est effectué correctement par le neurone de bas (E Bus B) en haut (E Bus H) tous les 52 cycles. Ce décalage correspond effectivement à la fonction décrite dans la partie contrôle du processeur. On a vérifie également que le calcul de l'état du neurone est correct en fonction des valeurs des coefficients chargées dans la mémoire et des états d'entrée du neurone.

C	E Bus H	E Bus B	S Bus H	S Bus B	Opérations effectuées
0	xxxxxxxx	xxxxxxxx			Initialisation
1	00000000	00**000100**			R8 = 4
2	00000000	**01**010100			**R5=5** , R9=4
3	00000000	**00**110101			**R7=3** , R6=5
4	00000000	0010,0100			Coef.1= 2,25
5	00000000	0011,0000			Coef.2= 3,0
6	00000000	0011,0010			Coef.3= 3,125
7	00000000	0001,0101			Coef.4= 1,3125
8	00000000	0,0010000			Entrée 1= 0,625
60	00001000	0,0010000	0,0001000		Entrée 2= 0,125
112	00010000	0,0001010	0,0010000		Entrée 3= 0,078
164	00001010	0,0101001	0,0001010		Entrée 4= 0,32
273	00001010	0,1000000	xxxxxxxx	0,1000000	Etat = **0,5**

Dans le deuxième exemple, on suppose que le décalage des données s'effectue de haut (E Bus H) en bas (E Bus B). Les valeurs des coefficients sont 3,0625, 2, 0,1875 et 1,5625 ; les états respectifs sont 0,0781, 0,125, 0,078 et 0,32. Nous avons vérifié l'exactitude du calcul du potentiel, de l'état et les décalages des données effectués par le processeur neurone. Le résultat final de l'état du neurone est égal à 0,5.

2.5.3.2 Réalisation du test de validation et résultat

Le test a été réalisé sur un testeur IMS HS 1000 (40 MHz maximum). Tous les résultats se sont avérés corrects ; le neurone a donc été déclaré opérationnel et fonctionne parfaitement avec une horloge de 20 MHz sous une tension d'alimentation de 5 volts et une température de 20° C.

2.5.4 Surface et optimisation du circuit obtenu

Dans une première tentative d'optimisation de ce circuit, on estime obtenir un gain en surface de 30%. Ce nouveau circuit sera beaucoup plus rapide. Et pour ceci, on utilise 3 moyens de minimisation ; le chaînage d'opérations, le transfert simultané et la minimisation des ressources. Les résultats de cette optimisation sont donnés dans le paragraphe 2.4.6. On rappelle que les résultats partiel et final d'une multiplication se trouvent dans les registres R2 et R3.

Dans le premier circuit, les étapes d'addition ou de soustraction et de décalage sont effectuées en 2 cycles différents utilisant 3 états (figure 2.27). Dans la nouvelle proposition les additions/soustractions et décalage se font au cours du même cycle de base. Donc, les différentes multiplications seront effectuées en n cycles, au lieu de 2n cycles.

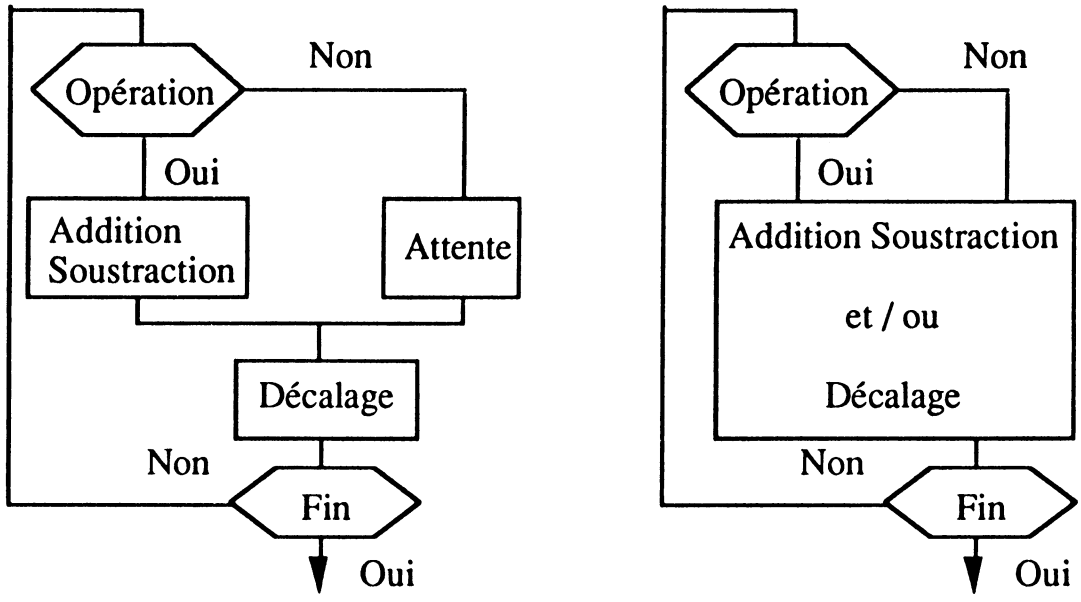


Figure 2.27: Multiplication dans le circuit initial / dans le circuit optimisé

D'un autre côté, le formatage des résultats des multiplications initialement fait dans les registres R2 et R3 (figure 2.28) sera directement effectué dans l'accumulateur, qui contient déjà les résultats. Cette modification permet de réaliser simultanément l'initialisation des registres R2 et R3 pour une nouvelle opération et le formatage du résultat de la multiplication. Cette optimisation permet aussi de supprimer une connexion entre les registres R2 et R3 et l'accumulateur.

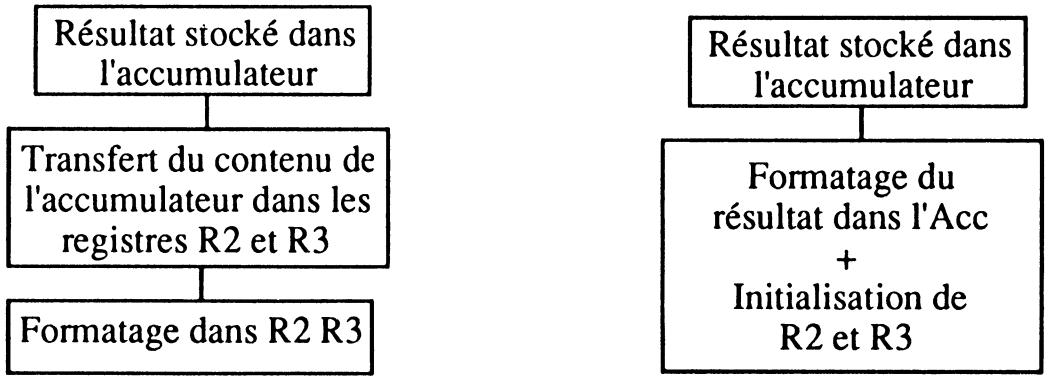


Figure 2.28: Formatage dans le circuit initial / dans le circuit optimisé

Pour toutes les multiplications effectuées, le neurone doit initialiser les registres R3, R2, R0 et lire la valeur du coefficient dans la mémoire. Dans la première version le neurone exécute cette étape en 3 cycles (figure 2.29).

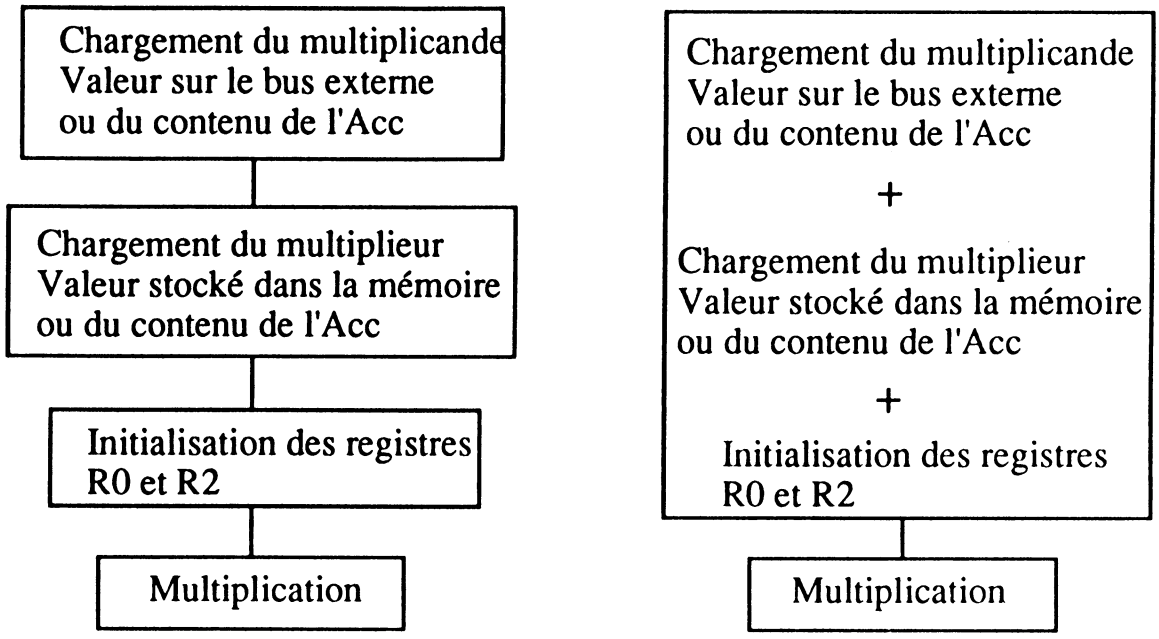


Figure 2.29: Initialisation dans le circuit initial / dans le circuit optimisé

On a remarqué qu'il n'existe aucun conflit entre les connexions utilisées pour les différents transferts de données, ce qui permet de réaliser en un seul cycle cette initialisation.

De la même façon, le décalage entre neurones des valeurs d'entrée, sur le bus externe, et la ré-initialisation pour une nouvelle multiplication seront effectués en un seul cycle.

Ainsi, le chaînage d'opérations a permis de réduire de 2/3 le nombre d'états de la partie contrôle et de moitié le nombre de cycles des multiplications.

Par ailleurs, la phase de minimisation des ressources a permis d'économiser des multiplexeurs, des connexions et des registres.

En effet, les sorties de la mémoire et de l'accumulateur sont multiplexées à l'entrée du registre R1, registre tampon sur n bits utilisé pour la mémorisation du multiplicande. Ces derniers ainsi que la sortie multiplexée sont stables tant qu'aucune instruction d'écriture n'est donnée pour qu'ils changent d'état.

Le registre R1 est en conséquence devenu inutile, puisque l'entrée de l'additionneur (ADD2) sera en permanence stable. Le registre R1 a été supprimé. Le cumul des produits partiels du potentiel, codés sur 2n bits, se fait en 2 cycles, puisque le cycle de base correspond au temps nécessaire pour effectuer une addition sur n bits. Ainsi, on a pu économiser un additionneur n bits en multiplexant les 2 opérations sur un seul additionneur (figure 2.30).

Le premier circuit (1 seul neurone, §2.4.2), qui a été réalisé et fabriqué en utilisant une technologie CMOS 2µ, a une surface de 15 mm^2 sans les plots. La partie contrôle est composée de 24 états et 39 transitions. En passant à une technologie 1,5µ assurant un gain moyen en surface de 36%, ce même circuit aura donc une surface de 9,6 mm^2.

Le circuit optimisé suivant les méthodes décrites dans ce paragraphe (§ 2.4.4), et simulé dans le § 2.4.5 a une surface de 6,2 mm^2, en technologie 1,5µ. La partie contrôle (17 états et 53 transitions) comporte 4000 transistors environ et elle occupe la moitié de la surface du neurone. On peut donc estimer que le nombre total de transistors est de 8000 environ. Ce qui donne une densité moyenne de 1290 transistors/mm^2. Ainsi nous avons atteint l'optimisation prévue de 30% en surface. Quant à la rapidité du neurone est multipliée au moins par 2 étant donné que la multiplication est actuellement effectuée en n cycles au lieu de 2 n cycles (n nombre de bits codant les coefficients et les états d'entrée/sortie du processeur neurone).

Une optimisation manuelle réalisée au niveau dessin des masques en milieu industriel, toujours en technologie 1,5 µ, devrait permettre une nouvelle réduction à 4 mm^2. Le passage à une technologie 1 µ assure un gain de 45% soit une surface de 2,2 mm^2. En conséquence, il sera possible d'intégrer une quarantaine de neurones dans un circuit de 1 cm^2 en technologie 1µ.

Très prochainement VLSI Technology Inc. mettra en service une technologie 0,8 µ. Le gain espéré en surface serait de 30% par rapport à la technologie 1 µ. Ainsi, 64 neurones peuvent être intégrés.

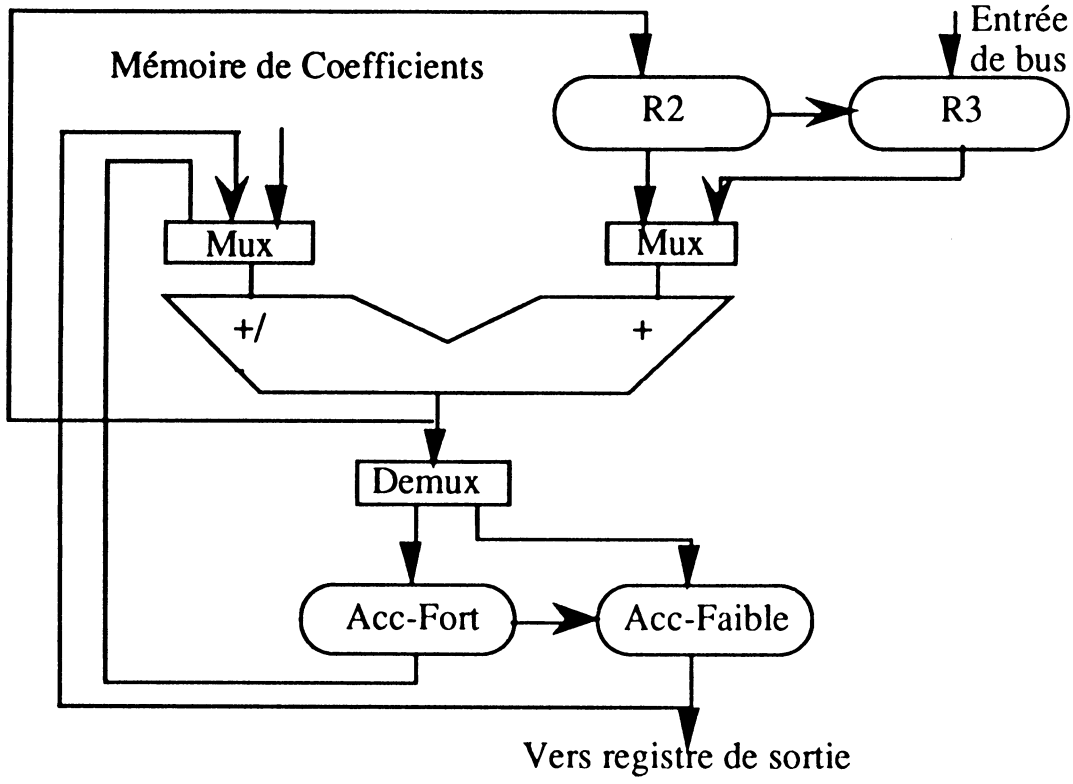


Figure 2.30 : L'unité arithmétique et logique modifiée

2.5.5 Simulation d'un réseau de neurones

Nous avons choisi un exemple de réseau à une seule couche de 8 neurones et 16 cellules d'entrée. Chaque neurone est connecté aux 16 entrées. Les valeurs d'entrée et de sortie sont codées sur 8 bits. Le réseau est organisé topologiquement en 2 lignes et 4 colonnes de processeurs neurones. Chaque processeur possède une mémoire de 64 octets dont seulement 16 coefficients relatifs aux 16 entrées seront chargés.

La validation de l'architecture et des calculs effectués par ce réseau est réalisée au niveau logique avec le logiciel de VLSI Technology Inc. Les modèles simulés sont ceux qui ont servi à la synthèse, aucune erreur due à l'extraction des modèles de plus haut niveau n'est à craindre.

Les résultats complets sont fournis en annexe. Le fichier complet des résultats pour cette simulation comporte plus de 10000 lignes et a donc été filtré et simplifié pour illustrer les étapes les plus importantes.

La simulation comporte les étapes suivantes :
- Initialisation et chargement des coefficients,
- Chargement des vagues de données et calcul du potentiel des processeurs,
- Calcul des états des processeurs et lecture des sorties.

A. Initialisation et chargement de la mémoire

L'initialisation du réseau est réalisée tout d'abord par une mise à "1" logique du signal reset. Ce signal permet à la partie contrôle du neurone de se positionner sur l'état initial (Time = 1:1).

Les registres de personnalisation et les coefficients synaptiques sont chargés successivement par colonne de processeurs (pour la première colonne Time = 5:1).

Les chargements des registres de personnalisation et des coefficients des processeurs neurones sont faits par les bus bas des neurones.

Le premier vecteur chargé contient le nombre de coefficients de la mémoire, soit 16 coefficients (Time = 5:1).

Le deuxième vecteur contient le nombre de décalages à réaliser pour traiter une vague de données et le nombre de décalages à effectuer pour le formatage du potentiel, soit respectivement 3 et 5 (Time = 6:1).

Le troisième vecteur contient les 2 nombres de décalages requis pour formater les résultats des 2 multiplications effectuées par la fonction d'activation, soit respectivement 7 et 3 (Time = 7:1).

Le quatrième vecteur de personnalisation contient le type de décalage (vers le haut ou vers le bas) des données lors du calcul du potentiel et la position physique du neurone, (indices ligne et colonne ; Time = 8:1).

Puis on procède au chargement des coefficients synaptiques dans les mémoires de la première colonne (de Time = 9 à Time = 24).

De la même manière, on charge successivement les neurones des colonnes 2, 3 et 4 (de Time = 25 à Time = 84).

B. Chargement des vagues de données et calcul du potentiel

Une vague de données contient 4 valeurs d'entrée. Etant donné qu'on a 16 valeurs d'entrée, on aura 4 vagues de données à charger. Les 4 valeurs de chaque vague de données seront décalées, §2.2.3, à l'intérieur de la couche pour le calcul des potentiels des neurones.

Au cours du traitement de la première vague de données, on illustre l'exemple d'une multiplication (de Time = 89 à Time = 96). On détaille aussi la manière dont le décalage des données est fait (Time = 100, puis 112 et 124). Les autres vagues, identiques à la première, ne seront pas détaillées.

Les 3 autres vagues sont chargées successivement aux cycles "Time = 189", "Time = 241" et "Time = 291"

La vérification de l'ensemble des calculs a été réalisée après chaque cycle d'horloge.

C. Calcul de l'état de sortie et lecture des sorties

Chaque neurone calcule à travers la fonction d'activation son état de sortie. Quatre cas se présentent :

- Non saturation positive, ce qui correspond aux neurones n11, n21 et n22,

- Non saturation négative, ce qui correspond aux neurones n23,

- Saturation positive, ce qui correspond aux neurones n12 et n13,

- Saturation négative, ce qui correspond aux neurones n14 et n24.

La fonction d'activation est activée uniquement par les neurones n11, n21, n22 et n23. Cette étape est réalisée en 30 cycles (de Time = 299 à Time = 329).

La lecture des sorties sur les bus bas des neurones sera effectuée successivement de la première colonne jusqu'à la quatrième. Ces résultats montrent que l'architecture développée fonctionne comme prévu théoriquement. Nous présenterons aussi une description détaillée des différentes cellules utilisées.

La simulation du réseau a été effectué en 335 cycles, avec une horloge de 100 ns, le chargement des coefficients étant compris. Elle montre que :

- la multiplication se fait en 8 cycles d'horloge de base,

- le cumul des termes partiels du potentiel s'effectue en 2 cycles,

- le calcul de l'état à travers la fonction d'activation prend 30 cycles.

Pour un réseau ayant 2 lignes de neurones, une vague de données est traitée en 49 cycles. Le traitement de toutes les entrées est effectué en 210 cycles. Ce qui correspond à 21 µs en utilisant une horloge de base de 100 ns.

2.5.6 Performances

Considérons un exemple d'application de reconnaissance de chiffres manuscrits [All90]. Le réseau est composé de 256 entrées, une couche de calcul de 45 neurones et une couche finale de 10 neurones. La couche de calcul est entièrement connectée aux entrées du réseau. Un neurone de la couche de sortie est seulement connecté à 10 neurones de la couche de calcul. La couche de calcul est organisée en 4 lignes et 12 colonnes de neurones. La couche de sortie contient des "neurones combinatoires".

Cette application a été simulée et le circuit est en cours de fabrication (retour de fonderie prévu pour fin Juin). Le boîtier contient 6 neurones organisés en 2 lignes et 3 colonnes. La technologie utilisée est une technologie CMOS 1,5µ. La phase de reconnaissance est réalisée en 2180 cycles, avec une horloge de 50 ns ce qui correspond à 109 µs.

Chapitre 3

Insertion de l'apprentissage dans le circuit dédié

3.1. Introduction

Le but de ce chapitre n'est pas de détailler ou de donner une liste exhaustive des algorithmes d'apprentissage, mais de montrer la faisabilité de l'implantation sur silicium d'un algorithme d'apprentissage sur l'architecture développée dans les premiers chapitres. Des thèses antérieures [Cun87] [Guy88] [Thi89] [Wan89] et la communication de [Lip87] sont d'excellentes synthèses sur les algorithmes d'apprentissage.

Nous nous sommes intéressés à l'algorithme de rétropropagation du gradient [Rum86] qui permet de résoudre de façon satisfaisante la plupart des problèmes complexes.

3.2. Algorithme de la rétropropagation du gradient

L'algorithme de la rétropropagation du gradient consiste à corriger les coefficients synaptiques en commençant à partir de la couche de sortie et en remontant vers la couche d'entrée en passant par les couches cachées. Son principe est rappelé sur la figure 3.1.

Nous allons commenter cet algorithme et montrer qu'il peut être implanté sur notre architecture sans modification fondamentale de celle-ci.

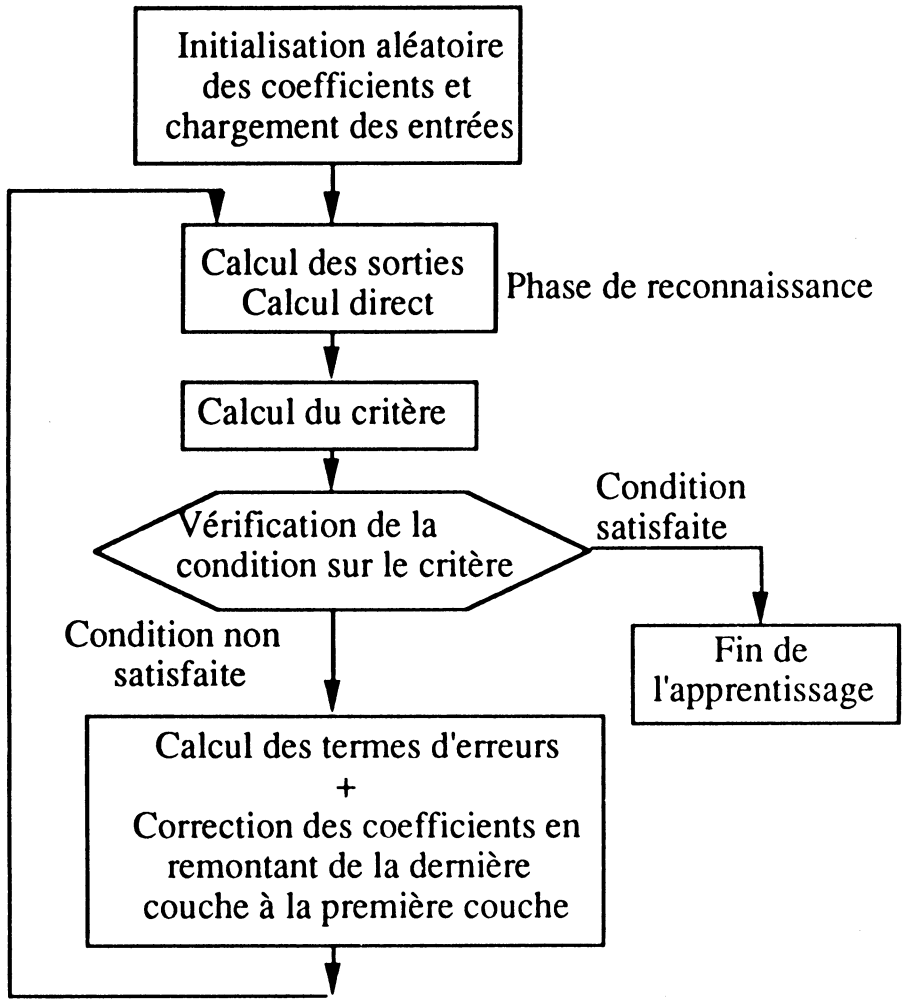


Figure 3.1 : Algorithme d'apprentissage par la méthode de rétropropagation du gradient

3.2.1 Description générale de cet algorithme

On considère un exemple de réseau à plusieurs couches, une couche d'entrée, des couches intermédiaires ou "cachées" et une couche de sortie. Un vecteur d'entrée E est soumis en entrée de la première couche. Un vecteur de sortie, appelé sortie désirée Sd, est supposé donné.

Aucune information n'est donnée initialement aux couches intermédiaires et les coefficients sont initialisés de façon aléatoire. Le réseau calcule, en allant de la couche d'entrée vers la couche de sortie (phase de reconnaissance) le vecteur de sortie qu'on appelle sortie calculée Sc.

Le vecteur de sortie étant calculé, le réseau doit vérifier que le critère C satisfait bien la condition (1), (C est un critère quadratique (2)) :

$$C \leq Er \qquad (1)$$

$$C = \frac{1}{NC_s} \sum_{k=1}^{NC_S} (Sd_k - Sc_k)^2 \qquad (2)$$

où NC_S représente le nombre de neurones dans la couche de sortie et Er représente la borne inférieure vérifiant la condition du critère. Si la moyenne de la somme de toutes les erreurs quadratiques des neurones de la couche de sortie est inférieure à Er, l'apprentissage est fini. Sinon, l'apprentissage est incomplet et la correction des coefficients synaptiques est nécessaire. La modification des coefficients se fait suivant la formule :

$$W(t+1) = W(t) - \eta \nabla C(W(t)) \qquad (3)$$

η est un facteur de gain, ∇C représente le gradient de la valeur du critère par rapport au coefficient synaptique à l'étape t.

On sait que le potentiel synaptique s'écrit comme une somme pondérée des entrées E_j par les coefficients W_{ij} ; i varie comme le nombre de neurones dans la couche et j varie comme le nombre de prédécesseurs auxquels le neurone i est connecté. On désigne par NC_i le nombre de neurones dans la couche i.

$$V_i = \sum_{j=1}^{NC_{i-1}} W_{ij} \cdot E_j \qquad (4)$$

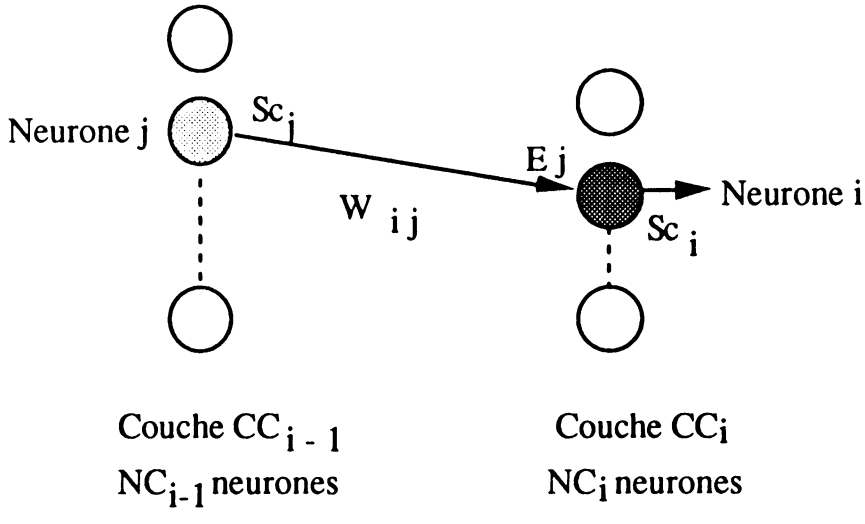


Figure 3.2 : Représentation de dépendance entre 2 neurones

La dérivée du critère par rapport au coefficient peut se décomposer par rapport au potentiel comme suit :

$$\frac{\partial C}{\partial W_{ij}} = \frac{\partial C}{\partial V_i} \frac{\partial V_i}{\partial W_{ij}} = \frac{\partial C}{\partial V_i} E_j \qquad (5)$$

E_j représente l'entrée du neurone i, notée Ni, dans la couche i, notée CC_i. Il représente aussi la sortie calculée Sc_j du neurone Nj dans la couche précédente CC_{i-1}.

En utilisant la linéarité de la fonction dérivée, on peut écrire que la dérivée, par rapport au potentiel, de la somme des termes du critère est égale à la somme des dérivées de ces termes :

$$\frac{\partial C}{\partial V_i} = \frac{1}{NC_s} \sum_{k=1}^{NC_s} \frac{\partial [Sd_k - Sc_k]^2}{\partial E_i} \frac{\partial E_i}{\partial V_i} \qquad (6)$$

En faisant l'hypothèse [Cun87] que les cellules de sortie n'envoient pas leurs états à d'autres cellules du réseau, on peut dire qu'elles ne sont pas interdépendantes. Par conséquent, la dérivée du critère (7) par rapport à l'entrée est partout nulle sauf pour i = k.

$$\frac{1}{NC_s} \sum_{k=1}^{NC_s} \frac{\partial [Sd_k - Sc_k]^2}{\partial E_i} = \frac{1}{NC_s} \frac{\partial [Sd_i - Sc_i]}{\partial E_i} \qquad (7)$$

Or pour la couche de sortie, on peut écrire:

$$\frac{\partial [Sd_i - Sc_i]}{\partial E_i} = -1 \quad \text{et} \quad \frac{\partial E_i}{\partial V_i} = \frac{\partial Sc_i}{\partial V_i} \qquad (8)$$

or Sc_i est la sortie calculée pour le potentiel Vi. Par conséquent, la dérivée partielle de Sc_i par rapport à V_i n'est que la dérivée $F'(V_i)$. L'équation finale utilisée pour la correction des coefficients dans la couche de sortie est :

$$W_{ij}(t+1) = W_{ij}(t) - 2.\eta.(Sd_i - Sc_i).F'(V_i).E_j \qquad (9)$$

En ce qui concerne les couches cachées, l'algorithme de la rétropropagation du gradient fait intervenir, comme son nom l'indique, une propagation de l'erreur de la couche de sortie vers la couche d'entrée. Par conséquent, le calcul de la dérivée du critère par rapport au potentiel pour la couche i (CC_i), peut être relié à l'erreur introduite par la couche en amont CC_{i+1}, constituée de NC_{i+1} neurones.

$$\frac{\partial C}{\partial V_i} = \sum_{k=1}^{NC_{i+1}} \frac{\partial C}{\partial V_k} \frac{\partial V_k}{\partial V_i} \qquad (10)$$

Toutes les dérivés de $\partial C / \partial V_k$ sont connues, puisqu'elles étaient calculées par la couche en amont. Le second terme peut être décomposé par rapport à la sortie calculée du neurone N_i qui est l'état du neurone calculé en phase de reconnaissance.

$$\frac{\partial V_k}{\partial V_i} = \frac{\partial V_k}{\partial Sc_i} \frac{\partial Sc_i}{\partial V_i} = W_{k\ i} \,.\, F'(V_i) \qquad (11)$$

Ainsi, on peut déterminer l'équation relative à l'ajustement des coefficients dans les couches internes et dans la première couche:

$$W_{i\,j}\,(t+1) = W_{i\,j}\,(t) - \eta.E_j.F'(V_i).\sum_{k=1}^{NC_{i+1}} W_{k\,i}\ \frac{\partial C}{\partial V_k} \qquad (12)$$

3.2.2 Approximation de la fonction dérivée

L'actualisation des coefficients nécessite le calcul de la dérivée de la fonction d'activation F. On rappelle que la fonction d'activation est une sigmoïde de la forme :

$$F(V) = \frac{1 - \exp(-\beta V)}{1 + \exp(-\beta V)}$$

La dérivée de cette fonction par rapport à V nous donne

$$F'(V) = \frac{2.\beta.\exp(-\beta V)}{(1 + \exp(-\beta V))^2}$$

Cette fonction dérivée peut être exprimée par :

$$F' = \frac{\beta}{2}(1 - F_{ap}^2) \qquad (13)$$

Cette équation est simple et facile à implanter. Elle comporte trois multiplications, une soustraction et une division par 2 qui se fait par simple décalage à droite du résultat final.

3.3 Implantation de l'algorithme sur l'architecture

L'objectif de cette partie est d'étudier si l'implantation de cet algorithme d'apprentissage nécessiterait la mise en place de modifications fondamentales de notre architecture (structure différente des communications, abandon d'une architecture processeur constituée d'un contrôleur unique et d'un chemin de données). Si ceci était le cas, nous pourrions dire que sur l'architecture précédente l'algorithme décrit ne peut être exécuté. Si l'exécution de cet algorithme laisse la structure inchangée, et ne nécessite que l'adjonction de certains registres ou compteurs dans la partie opérative et bien sûr des parties d'organigramme de contrôle dédiées nous dirons que cette architecture peut également implanter cet algorithme d'apprentissage.

Nous examinerons successivement le problème de la circulation de données et le problème de traitement. Nous verrons que cette architecture peut exécuter cet algorithme et l'on cherchera à estimer l'augmentation en surface ainsi que les performances attendues. Aucune réalisation ni aucune simulation physique n'a été faite à ce jour. Ce sujet fait actuellement l'objet d'une thèse à part entière dans le laboratoire CSI.

3.3.1 Circulation des données

Considérons un exemple simple de réseau constitué de 3 couches, CC_1, CC_2 et CC_3 ayant respectivement un nombre de neurones NC_1, NC_2 et NC_3. Dans notre exemple, on a choisi $NC_1 = 12$, $NC_2 = 8$ et $NC_3 = 8$. Le réseau topologique implantant l'architecture développée sera arbitrairement organisé en lignes et colonnes comme indiqué sur la figure 3.3. Les couches CC_1, CC_2 et CC_3 auront respectivement 6, 4 et 4 colonnes. Toutes les couches auront deux lignes L_1 et L_2 .

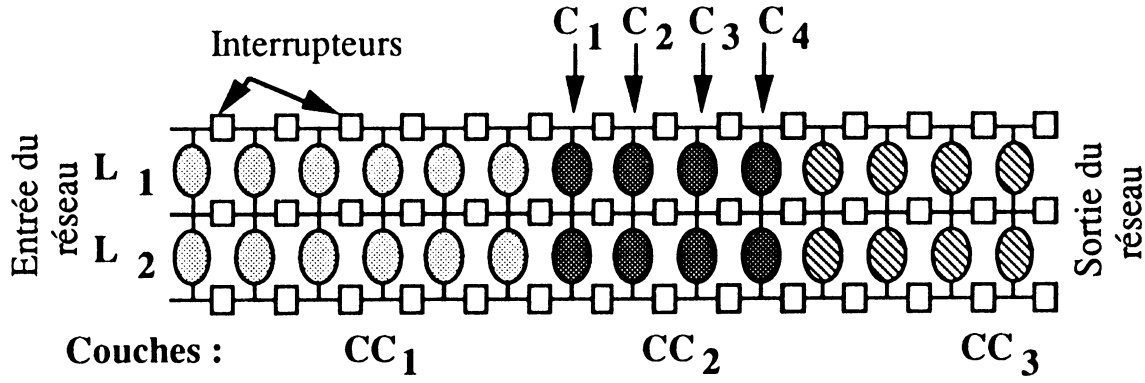


Figure 3.3 : Représentation du réseau

L'algorithme d'apprentissage par la méthode de rétropropagation du gradient est décomposé en plusieurs étapes de calcul :

- Calcul des états des neurones du réseau,

- Calcul du critère et vérification de la condition sur le critère,

- Calcul de l'erreur pondérée et correction des coefficients synaptiques.

A) Calcul des sorties du réseau

Le calcul direct des états des neurones de la couche de sortie est effectué en faisant propager l'information à partir de l'entrée jusqu'à la sortie du réseau en phase de reconnaissance comme expliqué au § 2.3.3.

B) Calcul du critère C

Le calcul du critère, utilisant l'architecture développée est effectué en 3 phases.

Phase 1 : Dans cette première phase, chaque neurone calcule le carré de l'écart entre la sortie désirée et le sortie calculée. Cette erreur est notée δ_i,

$$\delta_i = (Sd_i - Sc_i)^2$$

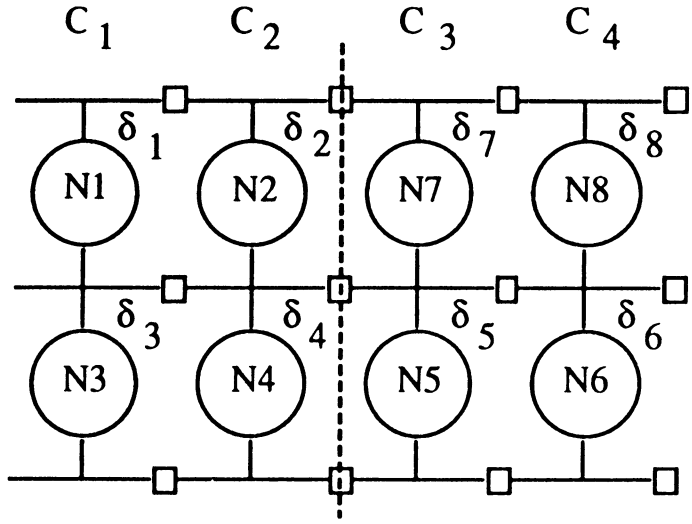


Figure 3.4 : Organisation et pliage de la couche de sortie

Phase 2 : On cumule les erreurs partielles δi calculées par les neurones de la couche de sortie. La ligne pointillée matérialise le pliage physique de la couche. Les deux neurones N1 et N3 de la colonne C_1 cumulent respectivement les termes d'erreur des neurones N2 et N4 (figure 3.5).

De la même façon, les neurones N7 et N5 de la colonne C_3 cumulent respectivement les termes d'erreur des neurones N8 et N6. Le transfert des termes partiels du critère C peut se faire soit par les bus bas des lignes de neurones soit par les bus haut. Pour cet exemple, on choisit un transfert par les bus bas.

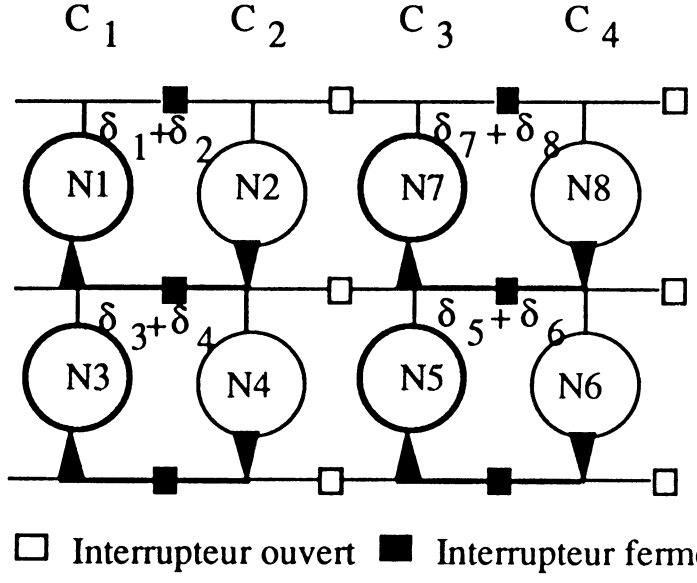


Figure 3.5 : Cumul partiel des termes d'erreur par les colonnes C_1 et C_3

N1 appartenant à la colonne C_1 et à la ligne L_1 calcule $\Delta C_1L_1 = \delta_1 + \delta_2$

N3 appartenant à la C_1 colonne et à la ligne L_2 calcule $\Delta C_1L_2 = \delta_3 + \delta_4$

N7 appartenant à la C_3 colonne et à la ligne L_1 calcule $\Delta C_3L_1 = \delta_7 + \delta_8$

N5 appartenant à la C_3 colonne et à la ligne L_2 calcule $\Delta C_3L_2 = \delta_5 + \delta_6$

De façon plus générale, les neurones topologiquement les plus à gauche d'une colonne physique cumulent les erreurs partielles de tous leurs voisins de droite situés du même côté de la ligne du pliage, (figure 3.6).

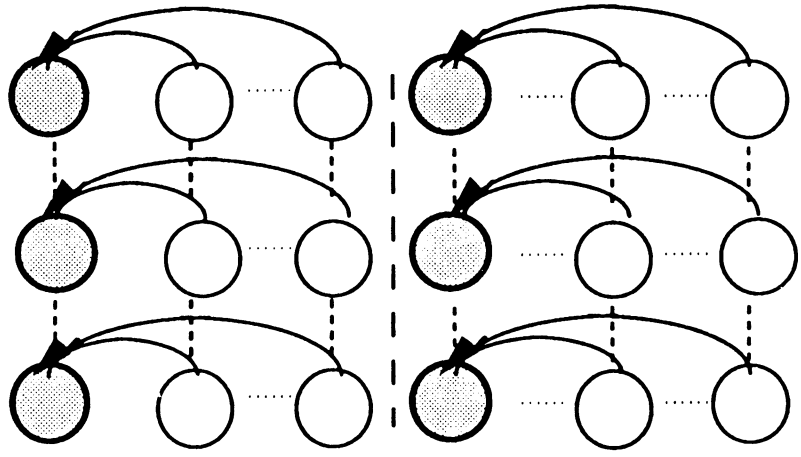


Figure 3.6 : Cumul partiel des termes d'erreur

Phase 3 : On effectue un décalage circulaire des erreurs cumulées (ΔC_1L_1, ΔC_1L_2, ΔC_3L_1, ΔC_3L_2) dans les colonnes C_1 et C_3 selon la figure 3.7.a. Chaque neurone appartenant à ces deux colonnes voit défiler un terme par cycle. Ce qui permet à tous ces neurones de calculer simultanément la valeur finale du critère.

Les neurones situés à gauche de la ligne de pliage décalent vers le bas, via leurs segments de bus bas, les valeurs ΔC_1L_1 et ΔC_1L_2. Les neurones situés après le pliage décalent vers le haut, via les segments de bus haut, les valeurs ΔC_3L_1 et ΔC_3L_2.

Chaque neurone des colonnes C_1 et C_3 calcule la somme des termes d'erreur δ_i :

$$\Delta C_1L_1 + \Delta C_1L_2 + \Delta C_3L_1 + \Delta C_3L_2$$

Le calcul sera réalisé par 3 décalages et 3 additions comme suit :

Dans le premier décalage, les neurones calculent les termes indiqués en gras,

$$N1 : \mathbf{\Delta C_1 L_1} + \Delta C_1 L_2 + \mathbf{\Delta C_3 L_1} + \Delta C_3 L_2$$

$$N3 : \mathbf{\Delta C_1 L_1} + \mathbf{\Delta C_1 L_2} + \Delta C_3 L_1 + \Delta C_3 L_2$$

$$N5 : \Delta C_1 L_1 + \mathbf{\Delta C_1 L_2} + \Delta C_3 L_1 + \mathbf{\Delta C_3 L_2}$$

$$N7 : \Delta C_1 L_1 + \Delta C_1 L_2 + \mathbf{\Delta C_3 L_1} + \mathbf{\Delta C_3 L_2}$$

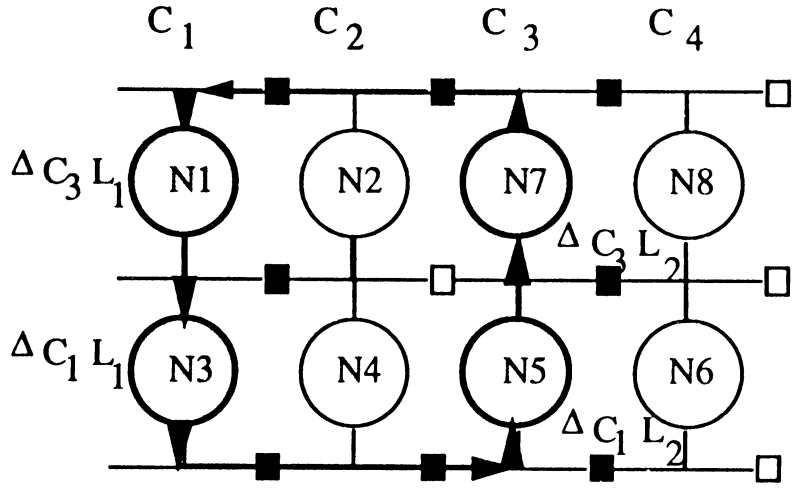


Figure 3.7.a : Premier décalage des termes d'erreur de C_1 et C_3

Dans le deuxième décalage, les neurones cumulent les termes indiqués en gras, les termes déjà utilisés (soulignés) seront décalés (figure 3.7.b).

$$N1 : \underline{\Delta C_1 L_1} + \Delta C_1 L_2 + \underline{\Delta C_3 L_1} + \mathbf{\Delta C_3 L_2}$$

$$N3 : \underline{\Delta C_1 L_1} + \underline{\Delta C_1 L_2} + \mathbf{\Delta C_3 L_1} + \Delta C_3 L_2$$

$$N5 : \mathbf{\Delta C_1 L_1} + \underline{\Delta C_1 L_2} + \Delta C_3 L_1 + \underline{\Delta C_3 L_2}$$

$$N7 : \Delta C_1 L_1 + \mathbf{\Delta C_1 L_2} + \underline{\Delta C_3 L_1} + \underline{\Delta C_3 L_2}$$

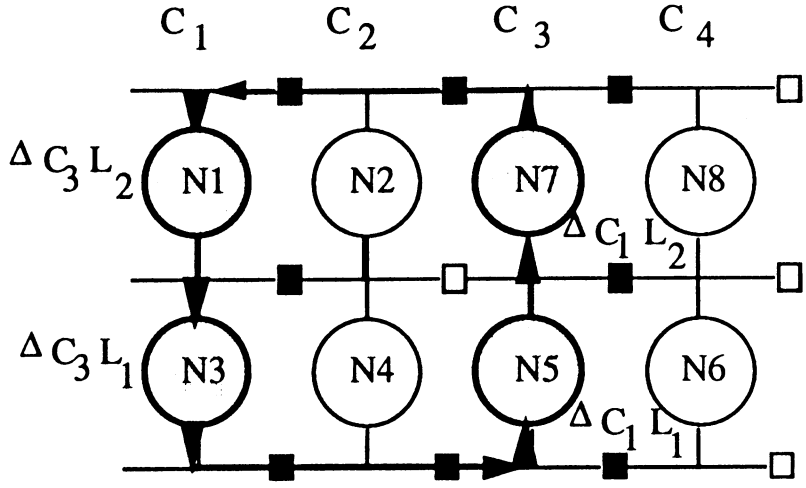


Figure 3.7.b : Deuxième décalage des termes d'erreur de C_1 et C_3

Dans le troisième et dernier décalage, les neurones cumulent les termes en gras, les termes déjà utilisés étant soulignés

N1 : $\underline{\Delta C_1 L_1} + \Delta \mathbf{C_1 L_2} + \underline{\Delta C_3 L_1} + \underline{\Delta C_3 L_2}$

N3 : $\underline{\Delta C_1 L_1} + \underline{\Delta C_1 L_2} + \underline{\Delta C_3 L_1} + \Delta \mathbf{C_3 L_2}$

N5 : $\underline{\Delta C_1 L_1} + \underline{\Delta C_1 L_2} + \Delta \mathbf{C_3 L_1} + \underline{\Delta C_3 L_2}$

N7 : $\Delta \mathbf{C_1 L_1} + \underline{\Delta C_1 L_2} + \underline{\Delta C_3 L_1} + \underline{\Delta C_3 L_2}$

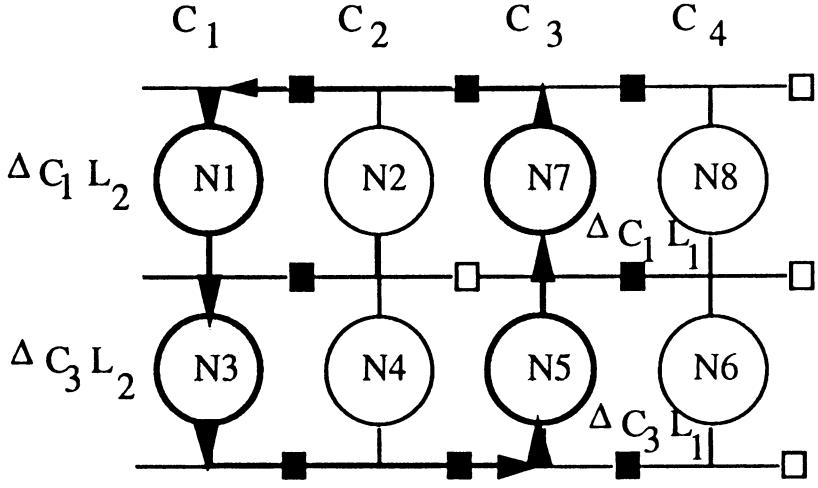


Figure 3.7.c : Troisième décalage des termes d'erreur de C_1 et C_3

On suppose que la condition sur le critère C n'est pas satisfaite et que la mise à jour des coefficients s'impose pour des neurones de la couche de sortie suivant l'équation (9) et pour les couches en aval suivant l'équation (12).

L'expression (14) définit l'erreur pondérée entre les couches CC_{i-1} et CC_i.

$$\Delta p_q = \sum_{i=1}^{NC_i} W_{q\,i} \cdot Sc_i \qquad q \text{ varie de } 1 \text{ à } NC_{i-1} \tag{14}$$

Le procédé de mise à jour est le suivant :

- La couche CC_3 calcule et transfère les termes des erreurs pondérées pour les neurones de la couche CC_2,

- Les neurones de la couche CC_3 corrigent leurs coefficients synaptiques suivant l'équation (9). Simultanément, la couche CC_2 calcule les termes des erreurs pondérées de la couche CC_1,

- Les neurones des couches CC_1 et CC_2 actualisent leurs coefficients suivant l'équation (12).

La couche CC_3, respectivement CC_2, qui calcule les erreurs pondérées pour la couche CC_2, respectivement CC_1, doit avertir CC_2, respectivement CC_1, par un message codé sur un bit, adressé à tous les neurones de la couche, et ceci, avant tout transfert des termes des erreurs pondérées.

Si on généralise ce principe de calcul (figure 8) pour une couche en aval quelconque CC_i, les étapes seront définies comme suit :

- La couche CC_{i+1} calcule et tranfère les termes des erreurs pondérées pour les neurones de la couche CC_i,

- Les neurones de la couche CC_{i+1} corrigent leurs coefficients synaptiques. Simultanément, la couche CC_i calcule les termes des erreurs pondérées de la couche CC_{i-1}.

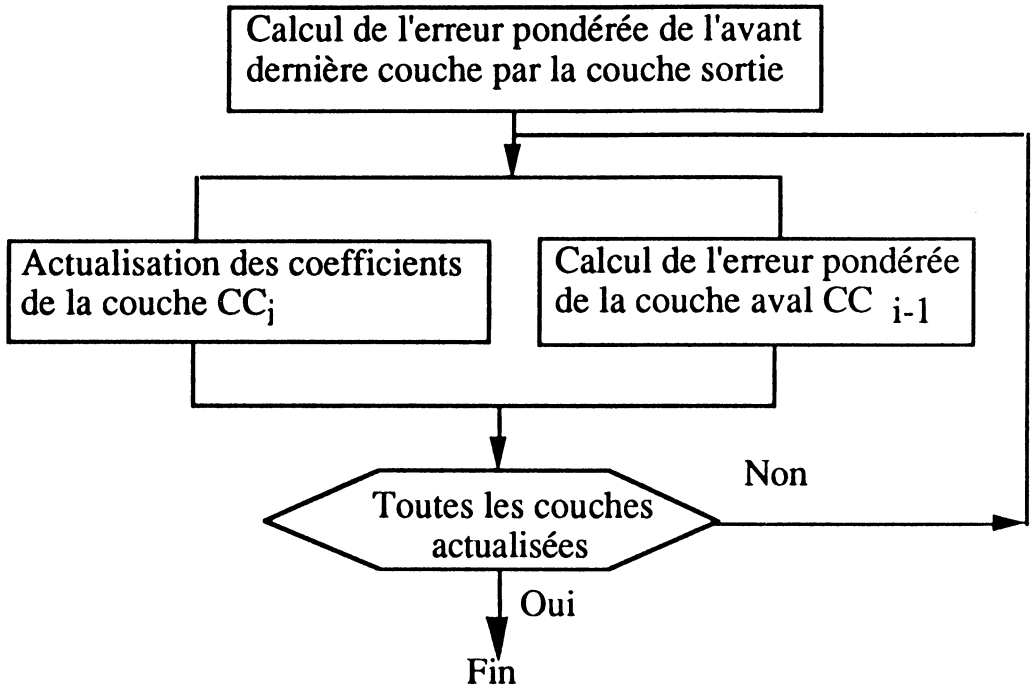


Figure 8 : Organigramme de mise à jour des coefficients

C) Calcul de l'erreur pondérée

Les neurones de la couche de sortie CC_3 calculent les erreurs pondérées des neurones de la couche en aval CC_2, équation (14). Les neurones de la couche CC_3 contiennent leurs coefficients de pondérations avec la couche CC_2, et les états utilisés pour effectuer cette somme.

L'erreur pondérée correspondant au neurone N1 de la couche CC_2 est :

$$\Delta p_1 = W_{11}.Sc_1+W_{12}.Sc_2 +W_{13}.Sc_3+W_{14}.Sc_4+W_{15}.Sc_5+W_{16}.Sc_6+W_{17}.Sc_7+W_{18}.Sc_8$$

Chaque neurone de la couche CC_3 calcule un produit partiel de cette somme.
Les neurones : N1 calcule $W_{11}.Sc_1$,
N2 calcule $W_{12}.Sc_2$,
N3 calcule $W_{13}.Sc_3$, ...

.....

N8 calcule $W_{18}.Sc_8$.

On utilise la même technique de cumul décrite par les phases 2 et 3 pour le calcul du critère.

Les valeurs de ΔC_1L_1, ΔC_1L_2, ΔC_3L_1, ΔC_3L_2 correspondent aux équations suivantes:

$$\Delta C_1L_1 = W_{11}.Sc_1 + W_{11}.Sc_1$$

$$\Delta C_1L_2 = W_{13}.Sc_3 + W_{14}.Sc_4$$

$$\Delta C_3L_1 = W_{17}.Sc_7 + W_{18}.Sc_8$$

$$\Delta C_3L_2 = W_{15}.Sc_5 + W_{16}.Sc_6$$

Ces termes seront transférés au neurone N1 de la couche CC_2, qui calculera la valeur finale de l'erreur pondérée, figure 3.9. L'utilisation des registres d'entrée pour le transfert permet aux neurones de la couche CC_3 d'effectuer en parallèle le calcul des termes d'erreur pondérée du neurone suivant (N2), de la couche CC_2.

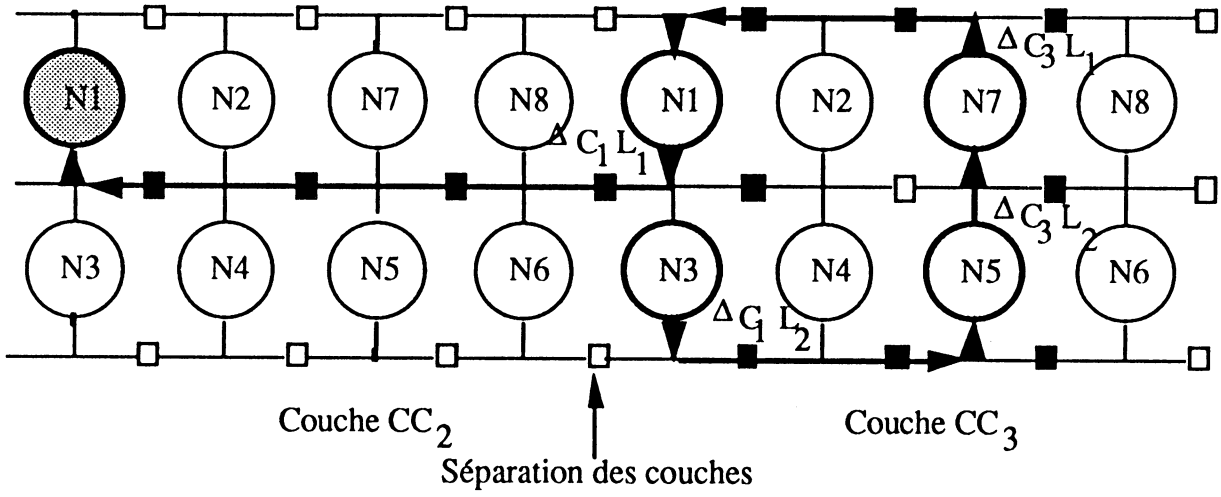


Figure 3.9 : Calcul de l'erreur pondérée du neurone N1 de la couche CC_2

On remarque que le transfert des termes partiels d'erreur du neurone N1 de la couche CC_2 est effectué par le bus bas de la première ligne dans les 2 couches. Il en serait de même pour les neurones N2, N7 et N8.
Pour les neurones N3, N4, N5 et N6 on utilisera le bus bas de la deuxième ligne. Le transfert peut être aussi effectué sur les bus haut. Il faut attendre que tous les neurones de la couche CC_3 aient calculé les erreurs pondérées des neurones de la couche CC_2 pour qu'elle actualise ses propres coefficients. Ce même principe est appliqué pour calculer les erreurs pondérées de la couche CC_1 par la couche CC_2.

D) Mise à jour des coefficents de la couche de sortie

Les équations (9) et (12) montrent que pour l'actualisation des coefficients le processeur neurone a besoin des valeurs de sortie des neurones des couches précédentes. Ainsi pour éviter des transferts supplémentaires entre les deux couches, ces valeurs, reçues en entrée pendant le calcul direct, seront mémorisées dans un banc de registres. Pour la structure de réseau de l'exemple, deux registres supplémentaires par neurone seront suffisants pour mémoriser les 8 valeurs d'entrée de la couche CC_3 et 3 registres pour la couche CC_2.

Nous allons détailler l'actualisation des coefficients des neurones de la couche CC_3. Les neurones de N1 à N8 actualisent leurs coefficients, donnés ci dessous. $W_{11}(E_1)$ représente l'actualisation du coefficient C_{11} en utilisant la valeur d'entrée E_1 dans le neurone N1.

Dans la première étape de calcul, on utilise les valeurs d'entrée E_1, E_3, E_5 et E_7. Les neurones :

N1, N2 utilisent la valeur E_1 pour actualiser respectivement W_{11} et W_{21}

N3, N4 utilisent la valeur E_3 pour actualiser respectivement W_{33} et W_{43}

N5, N6 utilisent la valeur E_5 pour actualiser respectivement W_{55} et W_{65}

N7, N8 utilisent la valeur E_7 pour actualiser respectivement W_{77} et W_{87}.

Les coefficients W indiqués en gras sont actualisés.

N_1:**$W_{11}(E_1)$**, $W_{12}(E_2)$, $W_{13}(E_3)$, $W_{14}(E_4)$, $W_{15}(E_5)$, $W_{16}(E_6)$, $W_{17}(E_7)$, $W_{18}(E_8)$

N_2:**$W_{21}(E_1)$**, $W_{22}(E_2)$, $W_{23}(E_3)$, $W_{24}(E_4)$, $W_{25}(E_5)$, $W_{26}(E_6)$, $W_{27}(E_7)$, $W_{28}(E_8)$

N_3:$W_{31}(E_1)$, $W_{32}(E_2)$, **$W_{33}(E_3)$**, $W_{34}(E_4)$, $W_{35}(E_5)$, $W_{36}(E_6)$, $W_{37}(E_7)$, $W_{38}(E_8)$

N_4:$W_{41}(E_1)$, $W_{42}(E_2)$, **$W_{43}(E_3)$**, $W_{44}(E_4)$, $W_{45}(E_5)$, $W_{46}(E_6)$, $W_{47}(E_7)$, $W_{48}(E_8)$

N_5:$W_{51}(E_1)$, $W_{52}(E_2)$, $W_{53}(E_3)$, $W_{54}(E_4)$, **$W_{55}(E_5)$**, $W_{56}(E_6)$, $W_{57}(E_7)$, $W_{58}(E_8)$

N_6:$W_{61}(E_1)$, $W_{62}(E_2)$, $W_{63}(E_3)$, $W_{64}(E_4)$, **$W_{65}(E_5)$**, $W_{66}(E_6)$, $W_{67}(E_7)$, $W_{68}(E_8)$

N_7:$W_{71}(E_1)$, $W_{72}(E_2)$, $W_{73}(E_3)$, $W_{74}(E_4)$, $W_{75}(E_5)$, $W_{76}(E_6)$, **$W_{77}(E_7)$**, $W_{78}(E_8)$

N_8:$W_{81}(E_1)$, $W_{82}(E_2)$, $W_{83}(E_3)$, $W_{84}(E_4)$, $W_{85}(E_5)$, $W_{86}(E_6)$, **$W_{87}(E_7)$**, $W_{88}(E_8)$

A l'étape suivante, on effectue un décalage circulaire des entrées E_1, E_3, E_5 et E_7. Les neurones :

N1, N2 utilisent la valeur E_7 pour actualiser respectivement W_{17} et W_{27}
N3, N4 utilisent la valeur E_1 pour actualiser respectivement W_{31} et W_{41}
N5, N6 utilisent la valeur E_3 pour actualiser respectivement W_{53} et W_{63}
N7, N8 utilisent la valeur E_5 pour actualiser respectivement W_{75} et W_{85}.

On remarque que ce principe a été détaillé dans le chapitre 2, §2.2.3. On peut dire que l'actualisation des coefficients de la couche de sortie ne nécessite aucune modification du principe de décalage des données proposé par l'architecture. De même, l'actualisation des coefficients dans les autres couches sera aisément réalisée.

3.3.2 Modification de la partie opérative

Les opérations effectuées dans la phase d'apprentissage sont identiques à celles qui sont effectuées en reconnaissance à savoir, la multiplication en complément à deux suivant l'algorithme de Booth, des additions pour cumuler des produits partiels, et des opérations de formatage. Les ressources utilisées sont les mêmes, additionneur/soustracteur, accumulateur, et registre R1, R2, R3 utilisés pendant la multiplication. Le nombre d'opérations en phase d'apprentissage est plus grand que celui en reconnaissance et demande plus de paramètres de personnalisation. Il est donc nécessaire de rajouter des ressources supplémentaires en terme de registres et de compteurs. Les trois compteurs ainsi que les registres de personnalisation utilisés en reconnaissance seront toujours utiles, puisque ces derniers contiennent une partie des informations nécessaires pour l'apprentissage, à savoir le nombre de prédécesseurs, la position physique du neurone dans la couche logique (coordonnées ligne et colonne), le nombre de neurones dans la couche logique, le sens de décalage pour traiter une vague de données, le numéro de la couche.

Pour la phase d'apprentissage,

1) le réseau calcule le critère C, il a donc besoin de connaître les valeurs de sortie du réseau. La sortie désirée du neurone dans la couche de sortie sera stockée dans un registre R11. La valeur que doit satisfaire le critère C sera stockée dans un registre R12.

Etant donné que le neurone n'est pas capable d'effectuer des divisions, la valeur qui normalise le critère C sera un paramètre de personnalisation stocké dans un registre R13. Cette étape nécessite un compteur de multiplication et un compteur de décalages circulaires.

2) le neurone doit connaître le nombre total de lignes du réseau. Ceci sera utilisé pour transférer les termes partiels de l'erreur pondérée au neurone de la couche précédente (figure 3.9). Dans cette étape, trois compteurs seront utilisés, un pour compter les décalages circulaires, deux pour incrémenter les numéros de la ligne et/ou de la colonne après chaque transfert d'erreur pondérée. Puisqu'on calcule en parallèle les termes d'erreur du neurone suivant, un quatrième compteur sera utilisé pour la multiplication.

3) au moment de l'actualisation des coefficients, le neurone aura besoin d'un compteur utilisé pour calculer les adresses mémoire et un compteur de multiplication. Par contre dans cette étape, le neurone aura besoin suivant les équations (9) et (12) des valeurs de β et de η. Afin de gagner un registre on stocke directement la valeur du produit dans un registre R14. On note aussi que la valeur du potentiel calculé doit être conservé ; un registre supplémentaire R15 sera utilisé. Un registre R17, dont on précisera ultérieurement la fonction, contiendra un terme commun à tous les neurones au cours de l'actualisation des coefficients. Les équations (9) et (12) montrent que le neurone doit effectuer 4 multiplications qui nécessitent 4 formatages ; 4 registres de personnalisation seront également ajoutés.

Enfin, on a remarqué précédemment qu'on devrait rajouter des registres dans la couche CC_i, permettant le stockage des états des neurones de la couche CC_{i-1}, et ceci afin d'éviter des transferts supplémentaires lors de l'actualisation des coefficients de la couche CC_i. Mais le nombre de ces registres reste toujours limité. Donc, pour des applications nécessitant un nombre très grand de registres, on sera amené à effectuer de nouveau le transfert de ces états. Dans ce cas, il est préférable de perdre des cycles de calcul, plutôt que d'augmenter de façon importante la taille du circuit.

En conclusion 10 registres et un compteur sont obligatoirement ajoutés à la partie opérative.

3.3.2 Modification de la partie contrôle

L'algorithme d'apprentissage exige 3 opérations essentielles, à savoir le calcul du critère C, le calcul des erreurs pondérées et la mise à jour des coefficients. On rappelle que la couche de sortie est composée de NC_S neurones organisés en L lignes (de L_1 à L_L) et s colonnes (de C_1 à C_S).

A) Le calcul du critère C suivant les phases 1, 2 et 3 décrites dans §3.3.1 peut être généralisé.

Après avoir calculé l'erreur δ_i, les neurones de la colonne la plus à gauche avant la ligne de pliage (la colonne C_1), respectivement après la ligne de pliage (la colonne C_p), cumulent les termes des autres colonnes de C_2 à C_{p-1}, respectivement de C_{p+1} à C_S, (figure 3.10).

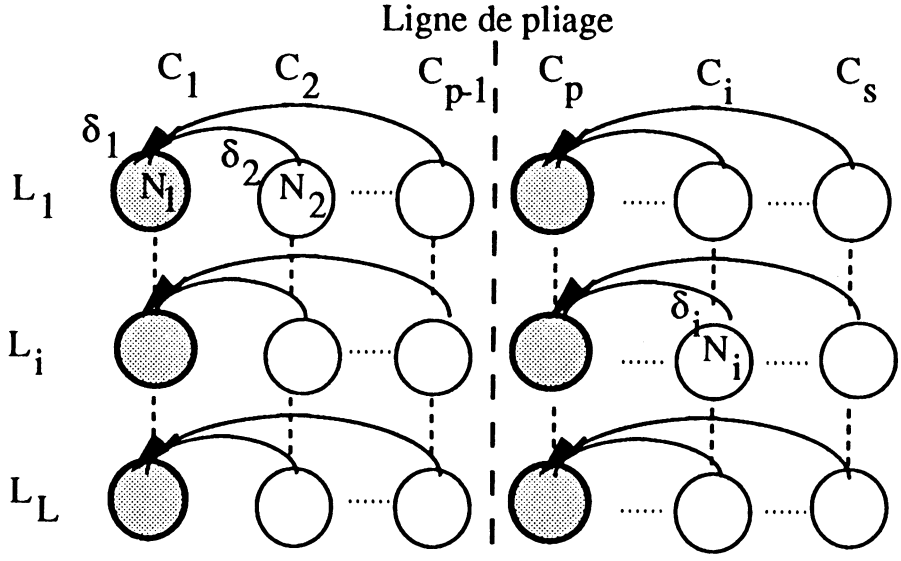


Figure 3.10 : Cumul des termes d'erreur dans C_1 et C_p

Dans cette phase de cumul, chaque neurone avant la ligne de pliage initialise un compteur à la valeur p-1 (p-1 correspond au numéro de la colonne C_{p-1}). De même, chaque neurone après le pliage initialise un compteur à la valeur s (s est le numéro de la dernière colonne). Ce compteur est decrémenté après chaque transfert. Le transfert des états des neurones d'une colonne est effectué si la valeur du compteur est égal au numéro de la colonne. Sinon aucun transfert ne sera effectué (figure 3.11.a).

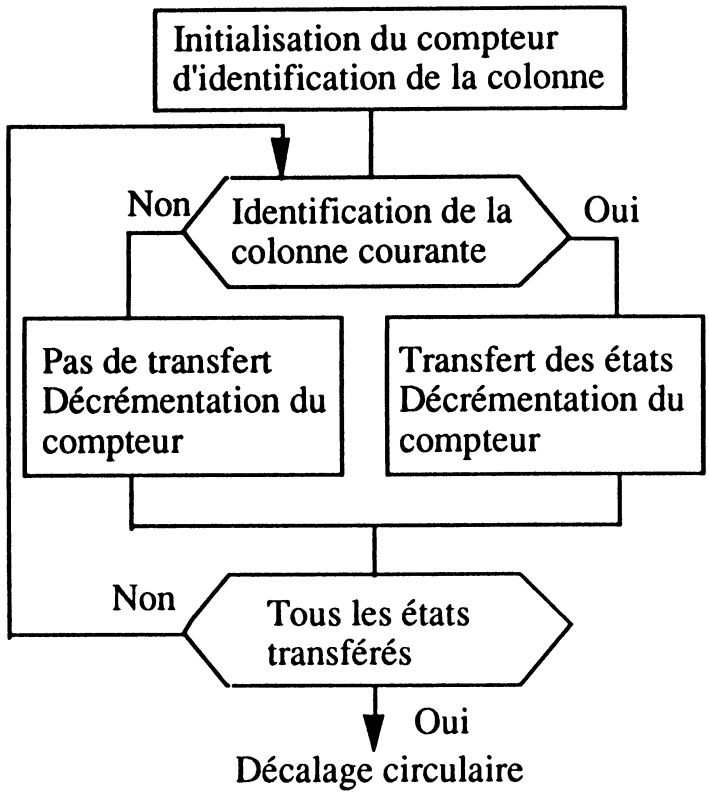


Figure 3.11.a : Organigramme de cumul des termes d'erreur dans C_l et C_p

On note par $CT0_l$, respectivement $CT0_p$, le compteur utilisé par les neurones situés avant la ligne de pliage, respectivement après la ligne de pliage. Le registre R16 (i) contient le numéro de la colonne courante, (paramètre de personnalisation). Le transfert des termes d'erreur est effectué à partir de l'accumulateur des poids faibles du neurone dans la couche de sortie vers le registre d'entrée du neurone de la couche en aval via le bus bas (Bus B) (figure 3.11.b).

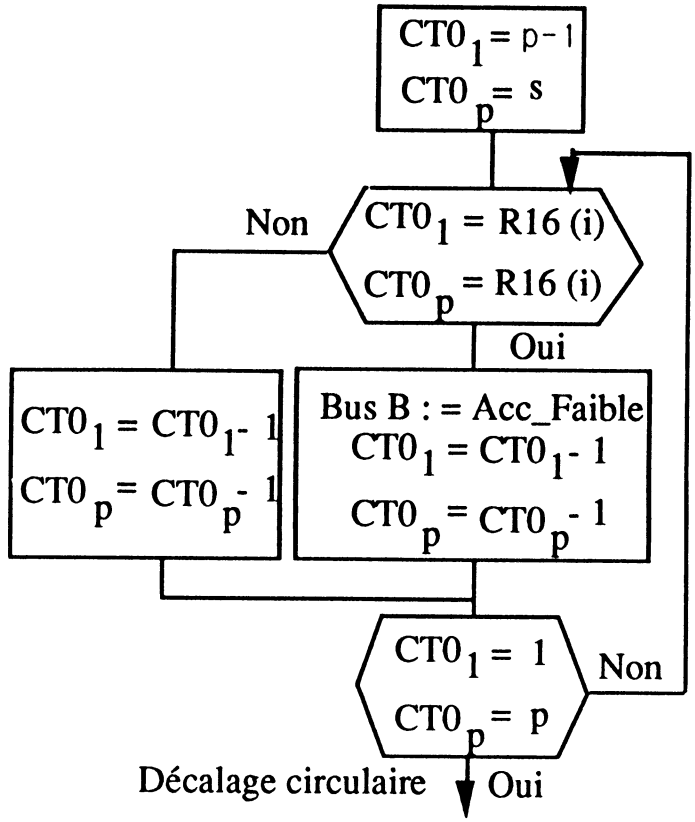


Figure 3.11.b : Algorithme cumulant les termes d'erreur dans C_1 et C_p

L'étape suivante consiste à effectuer un décalage circulaire des termes cumulés dans les colonnes C_1 et C_p. Les termes cumulés dans la colonne C_1 sont notés $\Delta C_1 L_i$, i variant de 1 à L. Les termes cumulés dans la colonne C_p sont notés par $\Delta C_p L_i$ (figure 3.12). Le registre R9 contient le nombre de décalages pour exécuter un tour complet des données dans les colonnes C_1 et C_p.

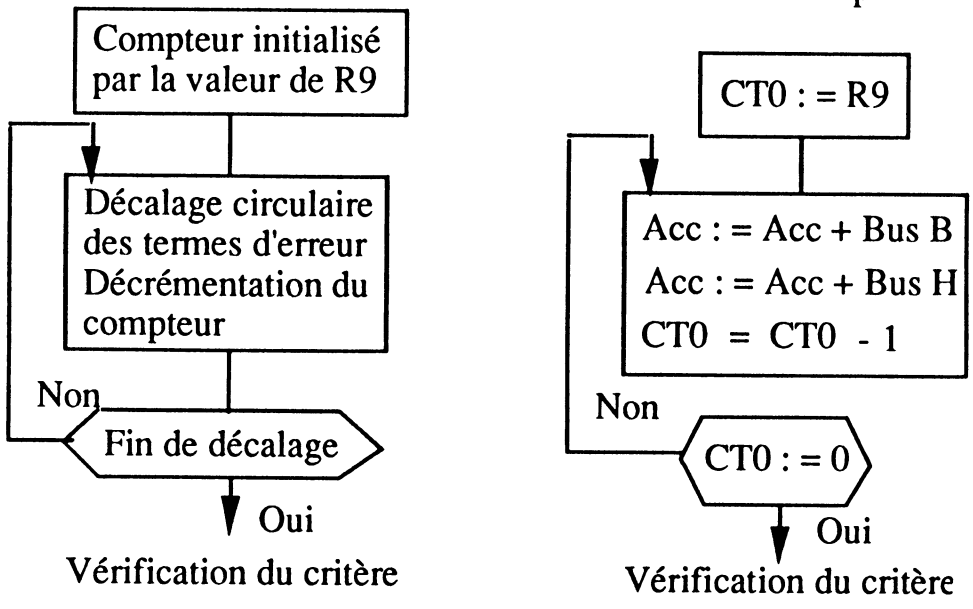


Figure 3.12 : Organigramme et algorithme du décalage circulaire

B) Calcul de l'erreur pondérée

Le principe généralisé du calcul de l'erreur pondérée a été donné dans le paragraphe 3.3.1. Si on considère une couche interne CC_{i+1} qui doit calculer les erreurs pondérées des neurones de la couche en aval CC_i.

Chaque neurone de la couche calcule un produit partiel de cette somme, (équation (14)). En utilisant la même technique de calcul du critère, on cumule ces produits dans les colonnes C_1 et C_p. Les termes cumulés, $\Delta C_1 L_i$ et $\Delta C_p L_i$, i variant de 1 à L, seront transférés au neurone N_i appartenant à la couche CC_i, (figure 3.13.a). Ce transfert est effectué par les registres d'entrée des neurones de la couche CC_{i+1}. En parallèle, les neurones de la couche CC_{i+1} calculent les termes de l'erreur pondérée du neurone suivant (figure 3.13.b). Le registre R4 contient le nombre de cycles nécessaire pour effectuer une multiplication. CT1 est un compteur utilisé pendant cette multiplication.

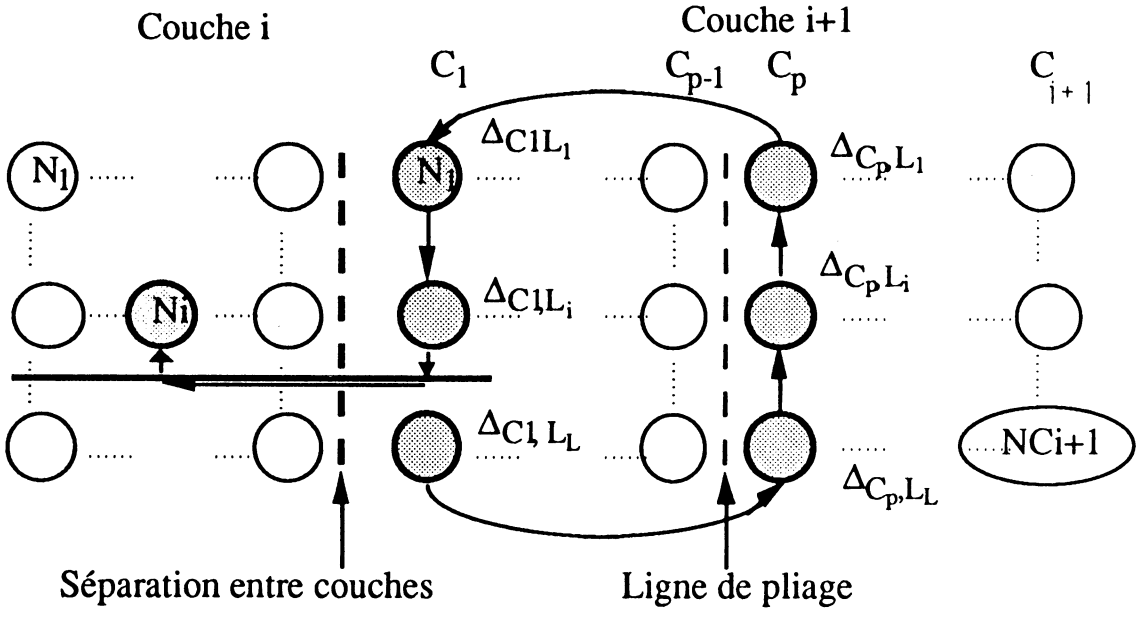


Figure 3.13.a : Cumul de l'erreur pondérée dans la couche en amont

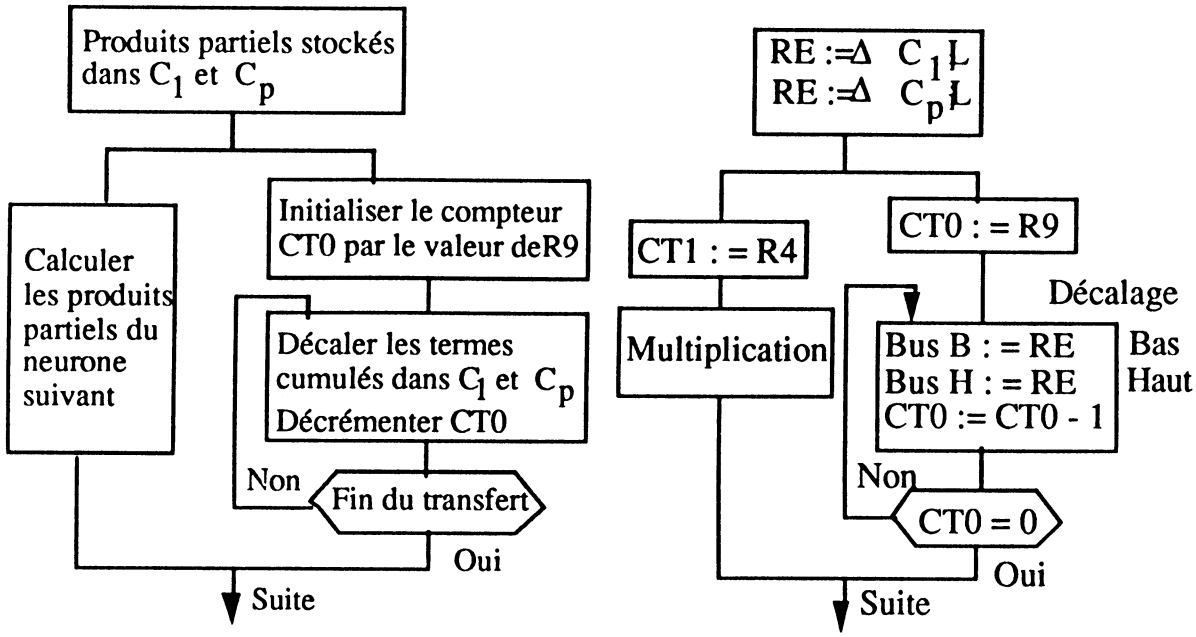


Figure 3.13.b : Organigramme et algorithme du transfert des termes de l'erreur pondérée

C) Actualisation des coefficients

Dans notre cas, deux types de corrections se présentent, la correction des coefficients dans la couche de sortie et celle dans les couches internes. D'après les équations (9) et (12), on remarque qu'il existe dans chacun des cas un terme commun pour la correction des coefficients.

Dans le cas de la couche de sortie ce terme est égal à $2.\eta.F'(V_i).(Sd_i-Sc_i)$ et pour les couches internes il est égal à $\eta.F'(V_i).\Delta p_i$, où Δp_i est l'erreur pondérée du neurone N_i et où η est un paramètre de gain. Chaque neurone calcule son propre terme, qui sera stocké (dans un registre R17) et utilisé pendant l'actualisation des coefficients. L'unité de traitement du neurone permet aisément de calculer ces différents termes. Ce calcul nécessite 4 multiplications, donc 4 registres supplémentaires seront ajoutés pour le formatage du résultat de ces opérations.

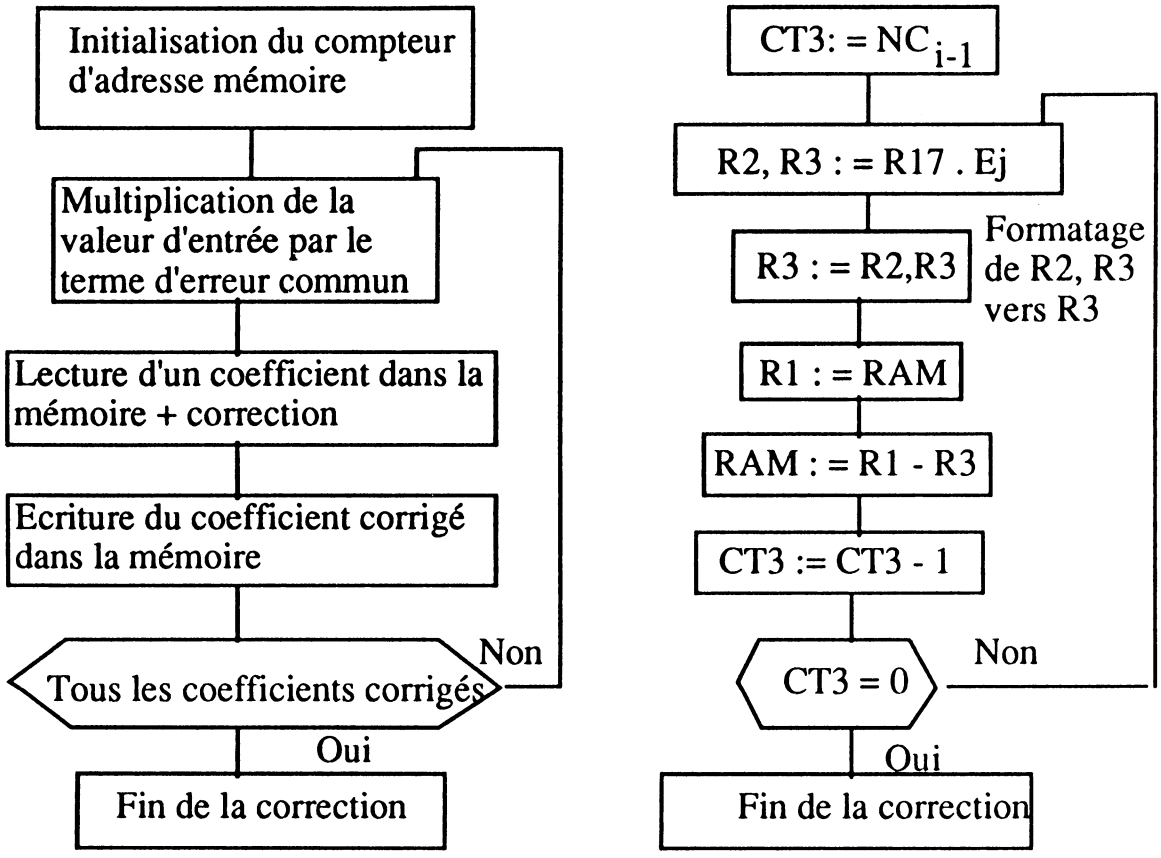


Figure 3.14 : Actualisation des coefficients

3.4 Surface et performance

La taille du circuit permettant de réaliser l'algorithme de la rétropropagation est fonction de l'augmentation de la surface de la partie contrôle et de la partie opérative. La surface de la mémoire des coefficients est inchangée.

Dans le cas du circuit réalisant la phase de reconnaissance, chacune des parties contrôle, opérative et la mémoire représente le tiers de la surface totale du neurone. Pour implanter l'algorithme d'apprentissage, on estime nécessaire une augmentation minimale de la partie contrôle de 2/3 de la surface initiale et la partie opérative de 1/3. Ce qui représente une augmentation de 1/3 de la taille globale du neurone.

Ces estimations sont déduites à partir de l'augmentation du nombre des états de la partie contrôle et du nombre de ressources en terme de registres et de compteurs ajoutés à la partie opérative. Mais l'augmentation réelle de la surface du processeur neurone ne peut être connue qu'après le choix d'une application précise et la définition du format des données.

La complexité de l'algorithme d'apprentissage impose un nombre d'étapes de calcul plus important. On donne aussi une estimation du nombre de cycles nécessaires dans les différentes étapes de calcul.

A) Calcul du critère C

- Le calcul de l'erreur δ_i nécessite **2.M+1** cycles, où M désigne le nombre de cycles pour effectuer une multiplication en complément à deux.

- Le cumul des termes d'erreur δ_i dans les colonnes C_1 et C_p dans la couche de sortie nécessite **(s-2) /2** cycles.

- Le décalage circulaire des termes cumulés dans C_1 et C_p est effectué pendant **2L-1** cycles.

Par conséquent, le calcul du critère C est réalisé en

$$\mathbf{ZC = 2.M + 2.L + (s-2)/2} \text{ cycles.}$$

B) Le calcul de l'erreur pondérée

On désigne par K_i le nombre total de colonnes dans une couche CC_i.

- Un produit partiel de l'erreur pondérée est calculé en **M** cycles.

- Le cumul de ces produits dans les colonnes C_1 et C_p est effectué en $(K_{i+1} - 2) /2$ cycles.

- Le transfert des termes cumulés à la couche en aval, en utilisant les registres d'entrée, nécessite **2.L** cycles. En parallèle, les neurones de la couche C_{i+1} calculent les produits partiels de l'erreur pondérée du neurone suivant en **M** cycles.

Les deux dernières étapes de calcul sont appliquées pour les NC_i neurones de la couche en aval. Le calcul de l'erreur pondérée des neurones de la couche CC_i est effectué en $Z_i = M+(K_{i+1} - 2)/2 + (2.L - M + (K_{i+1} - 2)/2).(NC_i - 1)$ cycles.

$$\mathbf{Z_i = M+NC_i.(K_{i+1} - 2)/2 + (2.L - M).(NC_i - 1)}$$

C) L'actualisation des coefficients d'une couche CC_i comportant NC_i neurones est réalisée comme suit :

- Le calcul de la valeur stockée dans le registre R17 est effectué en **4.M+(F1+F2+F3+F4)** cycles. F1, F2, F3, F4 représentent le nombre de cycles utilisés pour le formatage des résultats des 4 mutiplications.

- L'actualisation réelle des coefficients est réalisée en $\mathbf{3.NC_i}$ cycles. Pour chaque coefficient, le neurone effectue pendant 3 cycles une lecture de la mémoire, une opération (soustraction ou addition) et une écriture dans la mémoire du coefficient actualisé. Donc, une actualisation complète nécessite

$$\mathbf{A_i = 3.NC_i + 4.M+(F1+F2+F3+F4)} \text{ cycles.}$$

Ainsi, on peut établir l'équation permettant de définir le nombre de cycles nécessaires pour la remise à jour des coefficients pour une seule présentation à l'entrée du réseau.

$$ZC + Z_{s-1} + \sum_{i=3}^{s} \mathrm{Sup}(A_i, Z_{i-2}) + \mathrm{Sup}(A_1, A_2)$$

Illustrons ceci par un exemple de réseau ayant 256 entrées et constitué de 3 couches chacune ayant respectivement 150, 50 et 10 neurones. Ainsi, le réseau sera organisé en 5 lignes, 30 colonnes pour la première couche, 10 colonnes pour la deuxième couche et 2 colonnes pour la troisième couche.

On suppose aussi, que les données, coefficients et états, sont codés sur 8 bits. Par conséquent, toute multiplication sera effectuée en 8 cycles, M=8.

Le calcul du critère C est calculé en ZC cycles, ZC = 26.

Le calcul de l'erreur pondérée des neurones de la couche 2 est effectué par la couche 3, en Z_2 cycles , Z_2 = 116.

On pose arbitrairement F1=F2=F3=F4=4. Simultanément, la couche 3 actualise ses coefficients en A_3 cycles, A_3= 198, et la couche 2 calcule les erreurs pondérées de la couche 1 en Z_1 cycles , Z_1 = 1118.

En dernière étape, les couches 1 et 2 actualisent leurs coefficients respectivement en 816 et 498 cycles.

Une actualisation des coefficients du réseau, pour l'exemple présenté ci-dessus, est effectuée en 2076 cycles, (26+116+1118+816). Le réseau est supposé fonctionner, au pire des cas, avec une horloge de base de 20MHz. Donc, le réseau est capable de remettre à jour 4,4 10^6 connexions par secondes (4,4 MCups).

Chapitre 4

Intégration sur tranche entière de réseaux de neurones

4.1 Introduction

Pour les applications de calcul neuromimétique proposées dans la littérature, on peut être frappé par la variation des réseaux neuroniques utilisés ; le nombre de neurones varie de quelques dizaines à quelque milliers ; le nombre de couches varie de 1 à 5 ; la précision des formats de données varie de 4 à 64 bits. On peut donc à juste titre se poser la question de la limite de taille des réseaux de neurones que l'on pourrait implanter sur silicium.

Il est bien connu que l'on ne peut aisément augmenter la taille des circuits pour des raisons de rendement en fin de fabrication. En effet, le rendement d'un circuit intégré sur silicium est une fonction exponentielle décroissante de la surface du circuit. Pour pallier ce problème, on ajoute des éléments redondants. En fin de fabrication, on établit la cartographie des neurones défectueux, on les élimine et on les remplace par les éléments de réserve.

Si l'on examine l'ultime frontière de l'intégration sur silicium qui est l'intégration tranche entière, le nombre de neurones intégrables sur tranche indépendamment de cette redondance dépend essentiellement de la précision des coefficients et des états, de la taille de la mémoire synaptique et du type de calcul.

La figure 4.1 montre le nombre de neurones intégrables sur une tranche de silicium de 10 cm de diamètre (4 pouces) en fonction du nombre de bits codant les coefficients (k bits). Ces courbes sont paramétrées par le nombre de bits codant les états (k1 bits) et représentent une estimation pour une intégration en technologie CMOS 1m. La taille de la mémoire synaptique dépend du nombre de neurones auxquels un neurone est connecté. Dans les cas respectifs de réseaux de Hopfield, de réseaux à 2 couches et à 3 couches, la taille de la mémoire a été considérée comme proportionnelle respectivement au nombre total, à la moitié et au tiers des neurones de la tranche ; on suppose donc que

chaque neurone est connecté à tous ses prédécesseurs et que les couches ont la même taille.

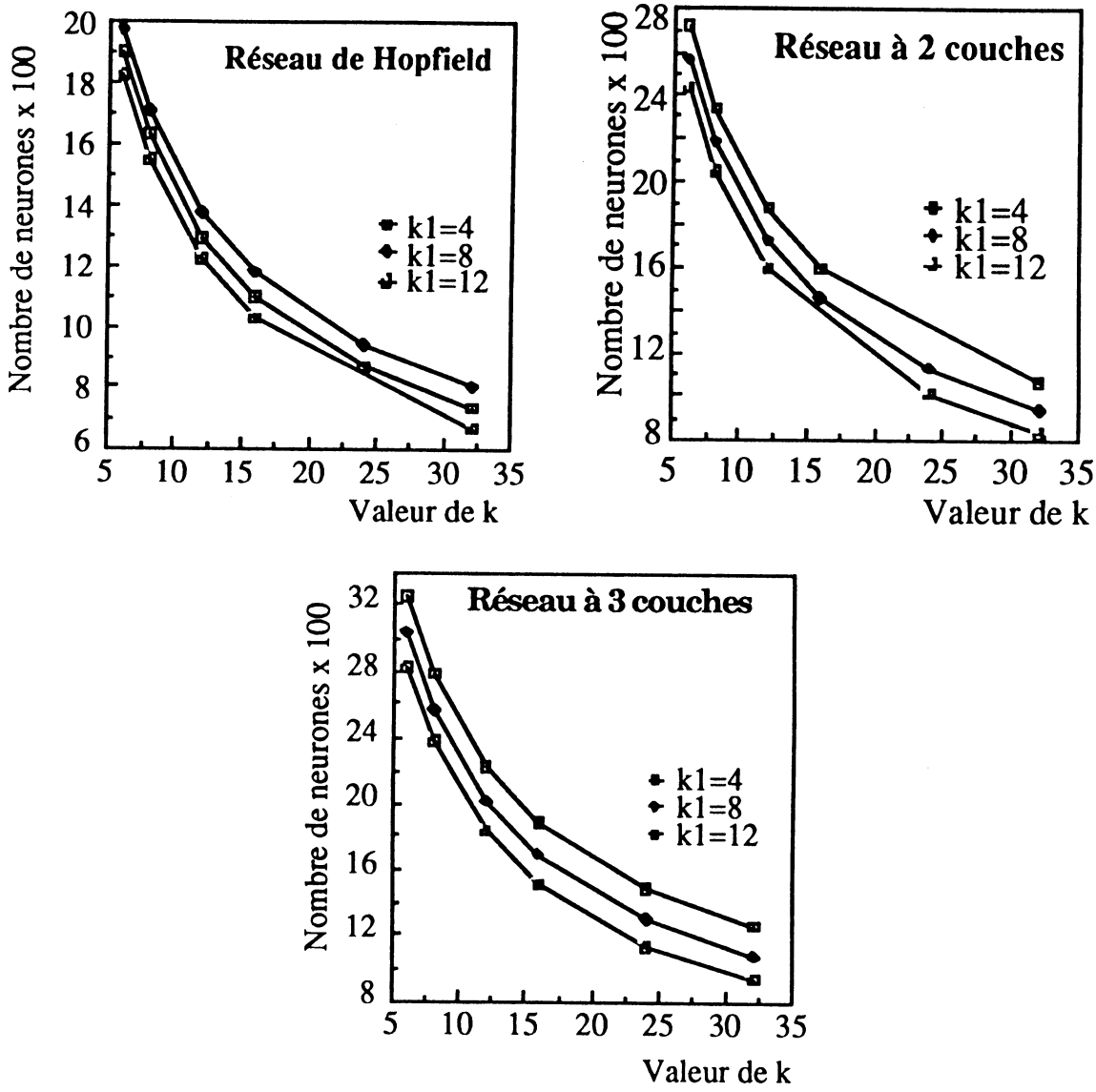


Figure 4.1 : Nombre de neurones intégrables sur tranche de 4 pouces

Nous ne traiterons pas de l'aspect modélisation du rendement de fabrication car les niveaux critiques et la densité de défauts par niveau critique sont des données confidentielles non communiquées par les fabricants de circuits intégrés et nous entraineraient dans des discussions difficiles sur les modèles de rendement.

Par contre nous examinerons les points critiques habituels de faisabilité d'une intégration tranche entière à savoir la possibilité d'identifier en fin de fabrication les neurones défaillants, en d'autres termes d'établir une

cartographie, et la possibilité de les déconnecter physiquement du réseau. Nous montrons donc comment ces deux points pourraient être résolus dans le cadre de l'architecture proposée. Nous proposons également une étude plus poussée de la testabilité en fin de fabrication et en déduisons des modifications mineures que l'on pourrait apporter à l'architecture du processeur neurone pour améliorer la testabilité.

4.2 Stratégie générale

Dans une première étape, la cartographie des neurones et des connexions est établie. Pour ce faire, un autotest des neurones sera effectué (§ 4.2.1.1). Ce test permet une validation partielle éliminant les erreurs fatales qui rendent le circuit irrécupérable. On propose deux tests minimum pour cette première validation.

Dans une seconde étape, les neurones jugés bons sont connectés par des "antifusibles" programmés par laser. Ces éléments sont appelés antifusibles car ils réalisent la fonction complémentaire des fusibles. A titre d'information, dans la technologie développée dans le projet ESPRIT 824, il s'agit de pastilles d'aluminium de 5µ de large déposées à cheval sur deux bandes voisines du deuxième métal des circuits intégrés, de façon à réaliser un court-circuit entre ces deux bandes voisines, et cela au moyen d'une photolytographie utilisant un laser, pour s'affranchir de la réalisation d'un masque sur plaque de quartz pour chaque circuit (les antifusibles sont mis seulement autour des neurones bons). Signalons aussi que cette technique est moins coûteuse que l'écriture directe sur tranche entière par faisceau d'électron, sans aucune perte de performance pour notre application. Dans une troisième étape, le test définitif et exhaustif des neurones et du réseau est effectué.

4.2.1 Cartographie des éléments sains

4.2.1.1 Autotest des neurones

La première étape du test consiste en un autotest, activé au moyen des signaux communs, effectué par le processeur neurone lui même. Cet autotest permet de sélectionner et d'éliminer tous les neurones atteints d'un défaut irréparable tel un court circuit entre les alimentations ou un mauvais fonctionnement grave de la partie opérative.

Un court circuit peut être détecté, d'une part, par la mesure de la consommation du circuit lorsque l'on connecte, par les interrupteurs

programmables, un neurone au réseau, d'autre part, par analyse de la sortie du neurone qui, dans ce cas, est aléatoire et/ou indéterminée pour une mise à zéro de tous les éléments constituant le neurone.

Un test minimum de la partie opérative illustrée en figure 4.2 est effectué. Le test complet du neurone sera réalisé ultérieurement (utilisation des bus). Dès l'initialisation générale du circuit, 2 valeurs fixées (X, Y) seront chargées dans les registres Acc-Fort et R2, propagées dans l'additionneur puis comparées à la valeur exacte du résultat (X+Y cablée dans la partie opérative). Si le résultat est incorrect, un signal S activera une bascule (Flag) observable par faisceau d'électron.

Une utilisation particulière des "swizzles" permet de donner une valeur fixe à chaque fil du bus de donnée, par simple collage à 0 V ou + 5 V.

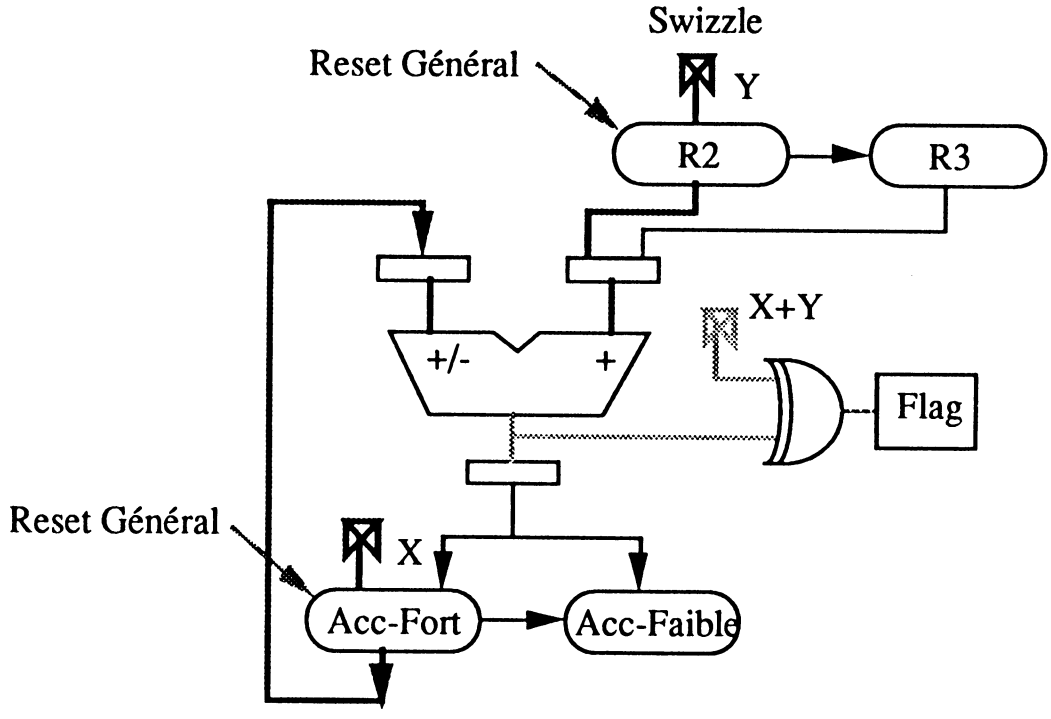


Figure 4.2 : Test minimal du neurone

Les swizzles sont des éléments de bibliothèque de l'outil compilateur de silicium de VLSI Technology Inc. [VLS90] qui permettent d'accéder aux lignes de bus dans le chemin de données. Il sont de deux types. Le premier type permet de sortir certains fils du bus spécifiés à l'extérieur du chemin de données via des connecteurs (figure 4.2.a). Le second type est utilisé pour changer la configuration d'un bus source vers un bus destinataire à l'intérieur

du chemin de donnée (figure 4.2.b). Une utilisation particulière des ces derniers permet de fixer une valeur binaire d'un élément de bus (figure 4.2.c).

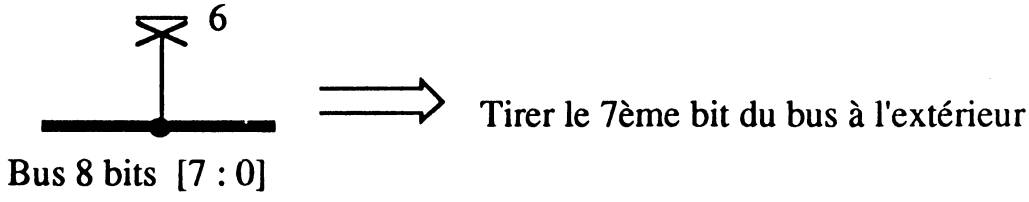


Figure 4.2.a : Représentation du swizzle externe

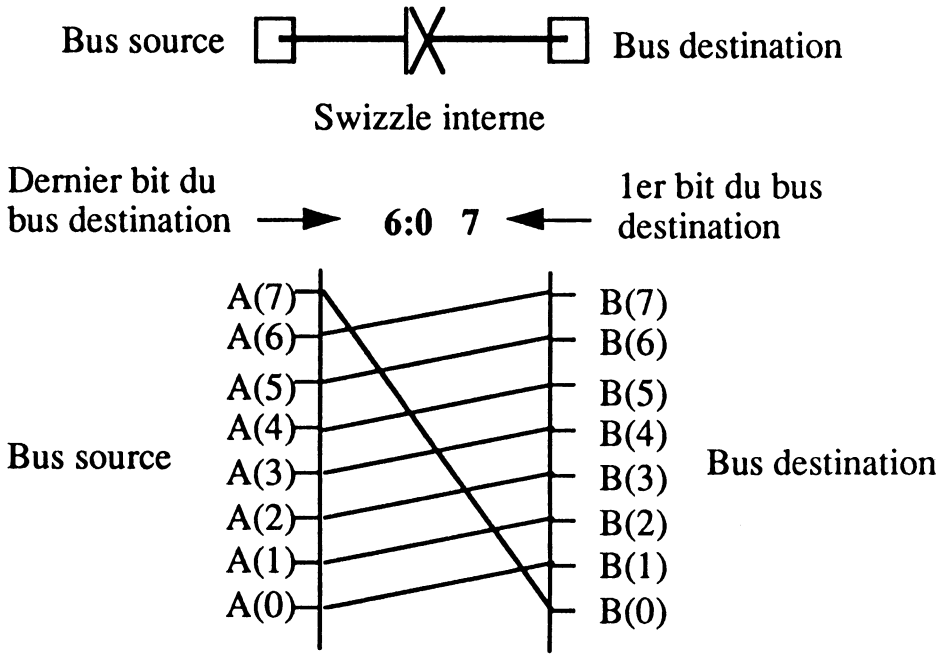


Figure 4.2.b : Exemple de configuration d'un bus du chemin de donnée

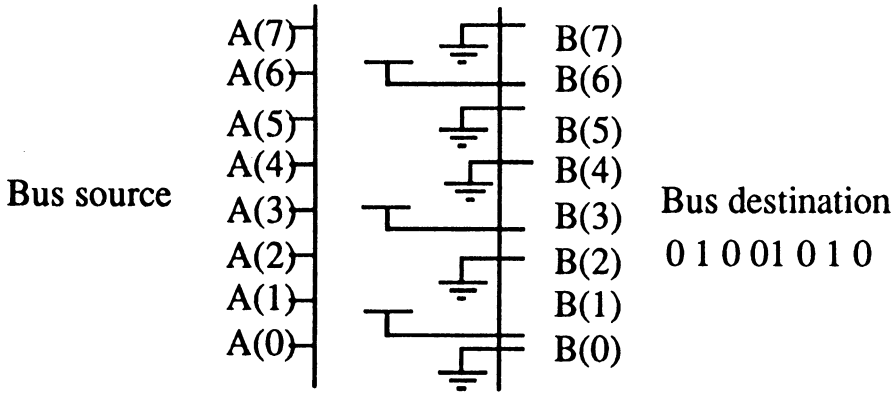


Figure 4.2.c : Mémorisation d'une valeur binaire

4.2.1.2 Test des bus

Les segments de bus sont considérés comme les éléments vitaux du circuit, et ne tolèrent aucun défaut. Rappelons que ces segments de bus (figure 4.3) sont utilisés pour charger initialement les neurones et transférer les états des neurones d'une couche à une autre et éventuellement d'un neurone à un autre dans la même couche. Une réalisation redondante de ces bus est donc nécessaire. Chaque segment de bus contient un commutateur programmable, permettant de réaliser les configurations a, b, c, d et e de la figure 4.4. L'utilisation de ces configurations alliée à l'usage des deux buffers trois états bidirectionnels du neurone permet la reconfiguration des bus.

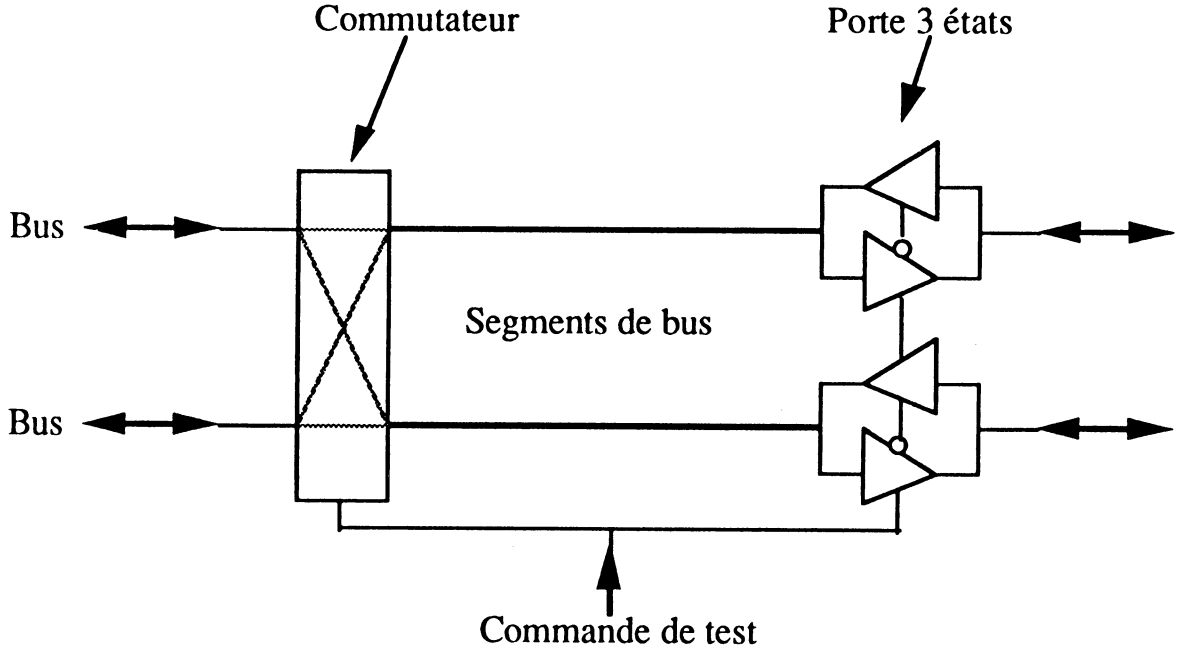


Figure 4.3 : Commutateur programmable

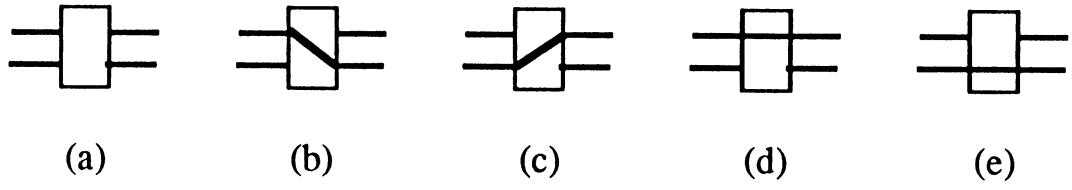


Figure 4.4 : Différentes configurations possibles du commutateur

Les segments de bus étant accessibles des deux cotés de la tranche, il est facile de tester et de détecter l'existence de segments défaillants. En effet, il suffit d'imposer une valeur à l'entrée du premier segment de bus et de lire la sortie du dernier segment dans une configuration initiale (figure 4.4 (d)). Si la sortie

est différente de la valeur d'entrée, un deuxième essai utilisant une configuration des commutateurs (figure 4.4 (e)) est effectué.

Si aucune des 2 configurations n'a donné de résultat positif, on cherche une configuration opérationnelle en faisant intervenir en plus les registres d'entrée et de sortie d'une paire de bus à l'autre.

L'étape suivante est consacrée au test des registres d'entrée et de sortie et des segments de bus haut et bas du neurone. Dans une première phase, on teste le registre d'entrée. On impose une valeur à l'un des segments du bus haut. Cette valeur est propagée dans le registre d'entrée puis lue sur l'un des segments de bus bas (figures 4.5.a et 5.b).

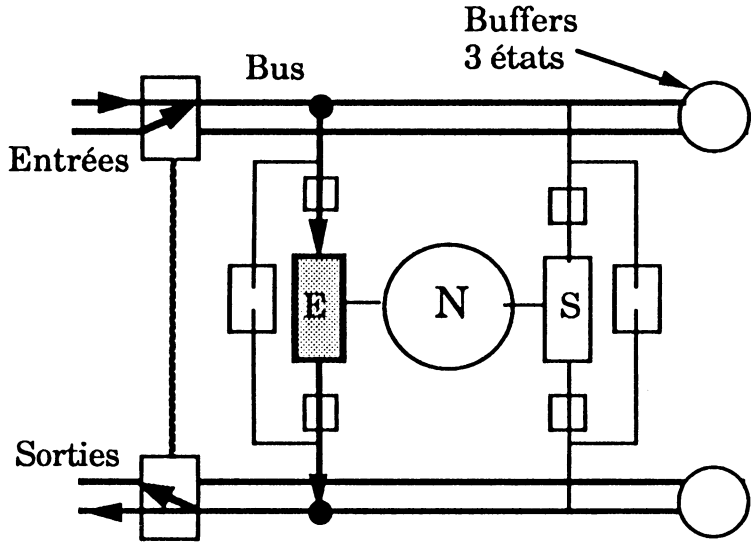


Figure 4.5.a : Ecrire sur le premier bus haut, lecture sur le second bus bas

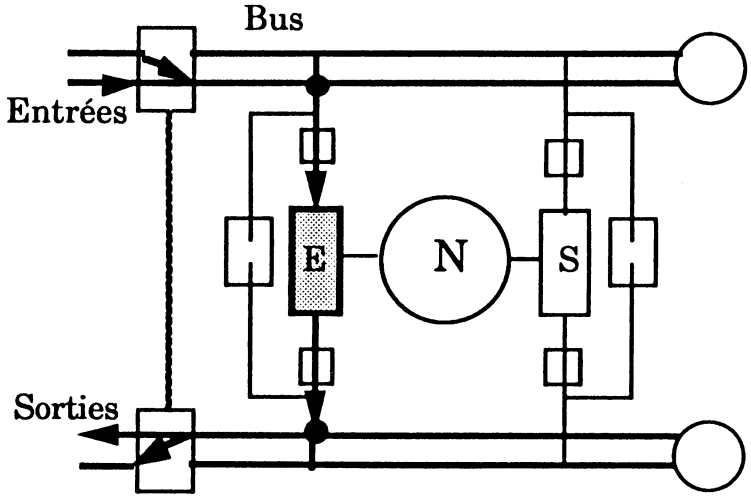


Figure 4.5.b : Ecrire sur le second bus haut, lecture sur le premier bus bas

Une deuxième phase permet de valider le fonctionnement du registre dans l'autre sens, écrire sur le bus bas et lire sur le bus haut (figure 4.6).

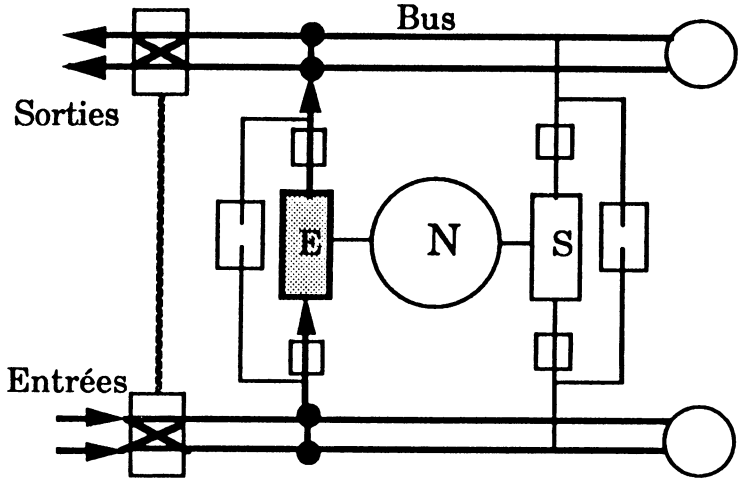


Figure 4.6 : Ecrire sur le bus bas, lecture sur le bus haut

Dans la troisième phase, on teste le registre de sortie. Une valeur déjà figée à l'entrée dans ce dernier (figure 4.7) est lue sur les deux bus du haut et du bas. Si l'un des 2 registres est défaillant, le neurone sera entièrement court circuité.

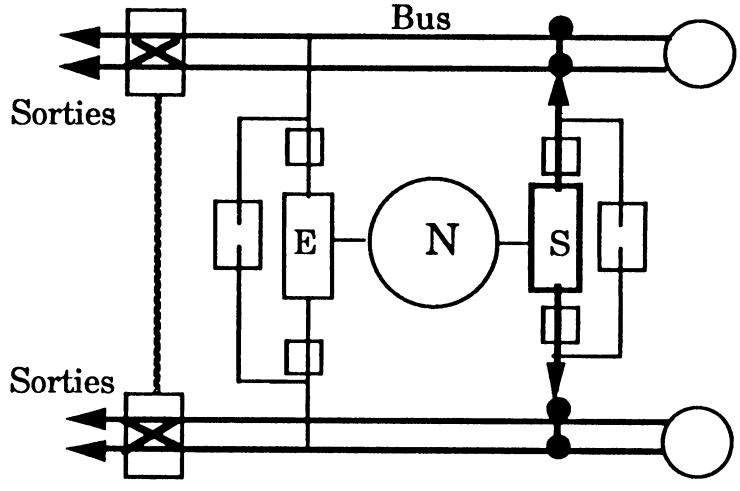


Figure 4.7 : Test du registre de sortie

Un neurone est provisoirement utilisable si ses registres d'entrée et de sortie sont sains. Si les 2 registres sont bons et si le test du neurone est négatif, les 2 registres d'entrée et de sortie n'auront plus aucune utilité et ils seront alors court circuités.

Les registres d'entrée et de sortie sont donc des éléments vitaux du neurone et il serait normal de prévoir des registres de remplacement. Mais en comparant la surface de ces derniers formés de 12.n transistors (n nombre de bits codant

l'état) par rapport à la surface du neurone de 14000 transistors environs, on s'aperçoit que la probabilité d'avoir un défaut atteignant les registres est infime. Nous avons donc choisi de ne pas dupliquer ces registres. De plus si on duplique les registres on augmente le nombre de ressources en multiplexeurs et signaux de contrôle, qui sont elles mêmes susceptibles d'une éventuelle défectuausité.

Une première validation des segments de bus s'effectue en même temps que celle du registre de sortie. Mais, dans le cas d'une défaillance de ce dernier, le test sera confirmé simultanément avec le test du registre d'entrée du neurone suivant, neurone appartenant à la même ligne mais de la colonne suivante.

Toutes les configurations du commutateur (figures 4.4.b.c.d.e) seront utilisées pour valider le segment de bus non défectueux.

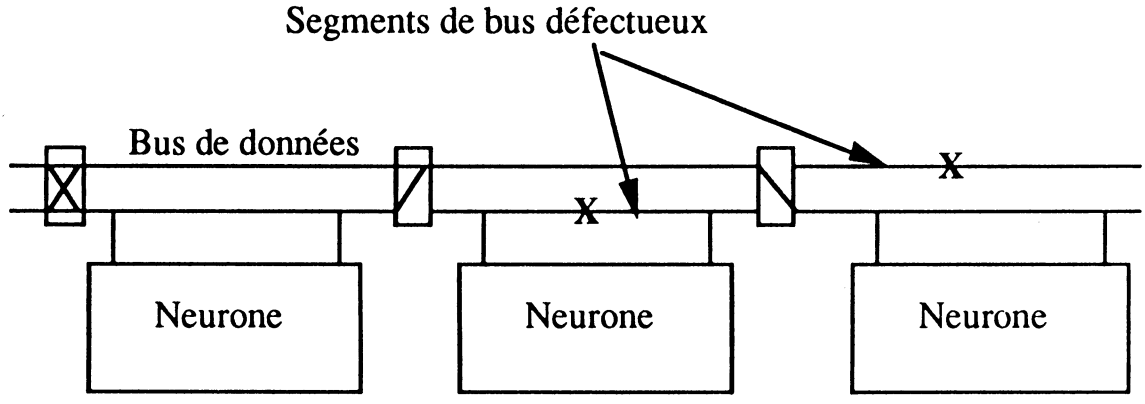


Figure 4.8 : Exemple de configuration possible

4.2.2 Configuration du réseau par déconnexion des neurones défectueux

4.2.2.1 Déconnexion des neurones défectueux

Initialement, tous les processeurs sont déconnectés. La première configuration du réseau consiste à éliminer les processeurs irrécupérables, en court circuitant leurs registres d'entrée et de sortie (figure 4.9) car ils ne peuvent introduire que du retard par des décalages supplémentaires. Le court circuit est réalisé par des anti-fusibles. Les fusibles servent à déconnecter les registres des bus.

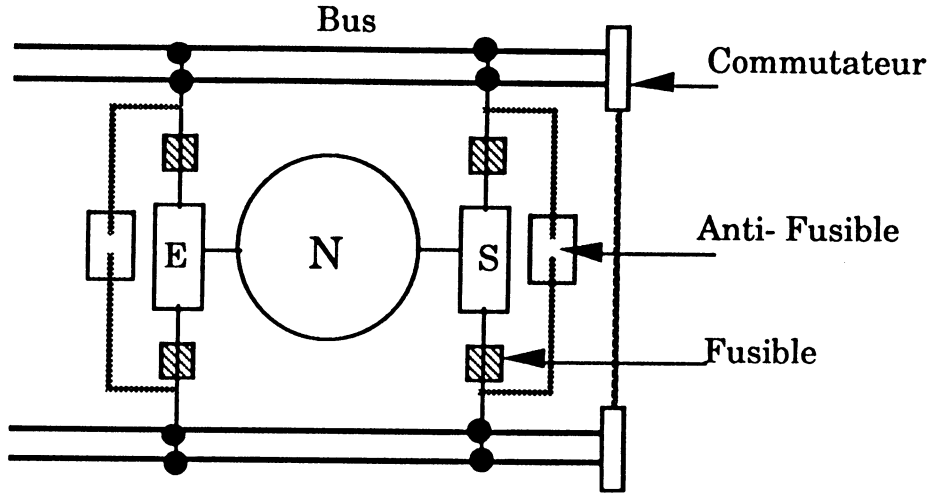


Figure 4.9 : Procédure de court circuit du neurone

4.2.2.2 Connexion des neurones non défaillants

La connexion des neurones aux bus est faite à l'aide des fusibles et des anti-fusibles (figure 4.10.a). Au moment du test du registre de sortie, on lit tout d'abord la valeur sur le bus B1. Si la valeur est erronée, on teste le registre d'entrée du neurone voisin. Si la valeur lue sur le bus B1 est correcte, alors le registre de sortie est défectueux et le bus B1 ne présente aucune anomalie. Le registre de sortie et l'anti-fusible seront court circuités et le fusible du bus B2 sera ouvert. Le bus B1 sera donc utilisé (figure 4.10.b).

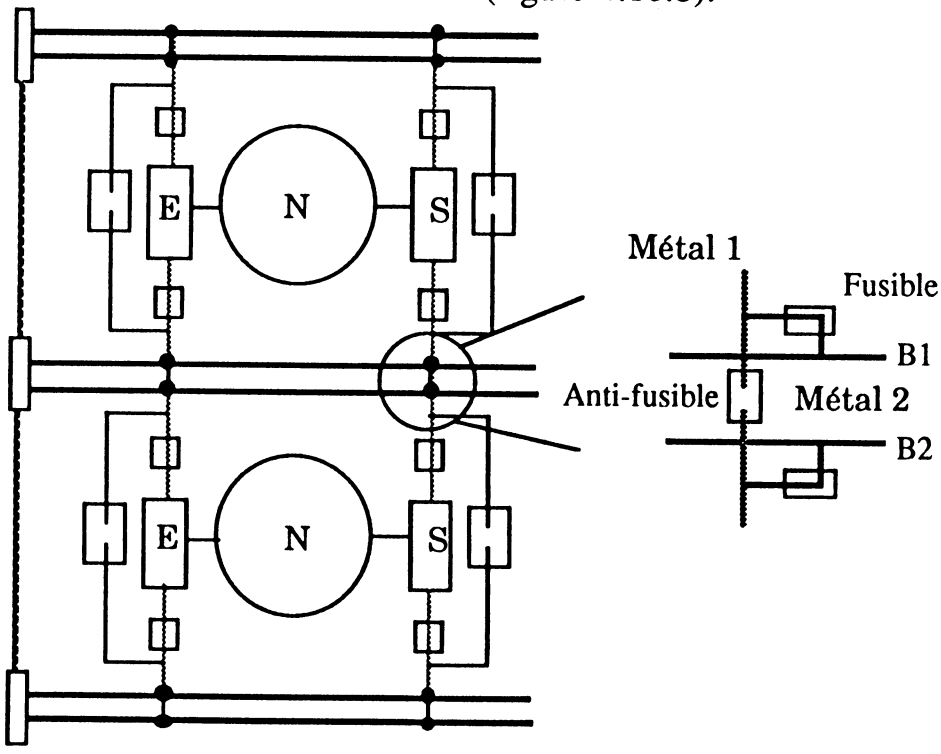


Figure 4.10.a : Connexion des neurones aux segments de bus

Par contre, si le registre de sortie et celui d'entrée du neurone voisin ne donnent pas le résultat voulu, alors le bus B1 est défectueux. L'anti fusible sera fermé et le fusible du bus B1 sera ouvert. On testera, de la même manière, le segment du bus B2 (figure 4.10.c).

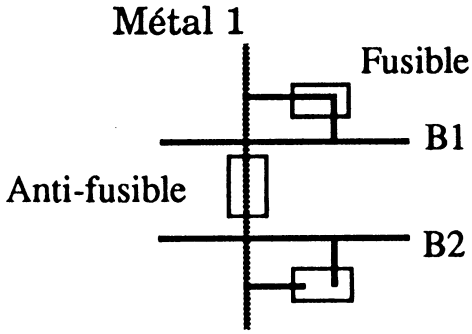


Figure 4.10.b : Utilisation du bus B1

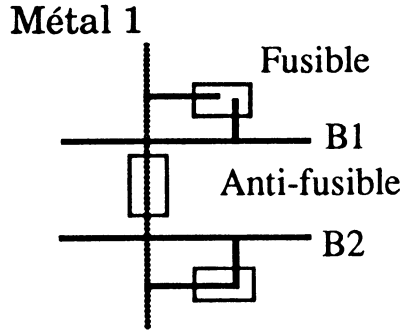


Figure 4.10.c : Utilisation du bus B2

4.2.3 Test définitif et étude générale de la testabilité du neurone

Le réseau définitif étant constitué, il s'agit d'appliquer un test aussi complet que possible. Pour ceci nous avons fait une étude à posteriori de son architecture dans une optique de très haute testabilité et de diagnostic aisé. Cette étude vise une perspective plus générale de test en fin de fabrication.

La testabilité permet l'évaluation de la difficulté de tester un module dans un système. Un bloc est testable si on peut lui appliquer des vecteurs locaux à partir des entrées primaires (contrôlabilité) et observer sa réponse sur les sorties primaires du circuit (observabilité).

Le neurone est constitué de deux parties indépendantes, une partie contrôle et une partie opérative. Le moyen le plus classique et le plus simple pour assurer une bonne observabilité de la partie contrôle et une bonne contrôlabilité de la partie opérative est d'intégrer entre ces deux parties un ensemble de registres montés en "Scan-path" (figure 4.11).

La propriété essentielle de ces registres est d'assurer un fonctionnement en 2 modes, le mode test et le mode de fonctionnement normal du circuit.

Lors du test, ces registres seront chargés en série par un seul plot d'entrée externe et le résultat du test sera aussi observé par une sortie série externe. Un plot d'entrée, un plot de sortie et un plot de contrôle seront suffisants pour permettre l'accès total aux deux grandes parties du circuit, la partie contrôle et la partie opérative.

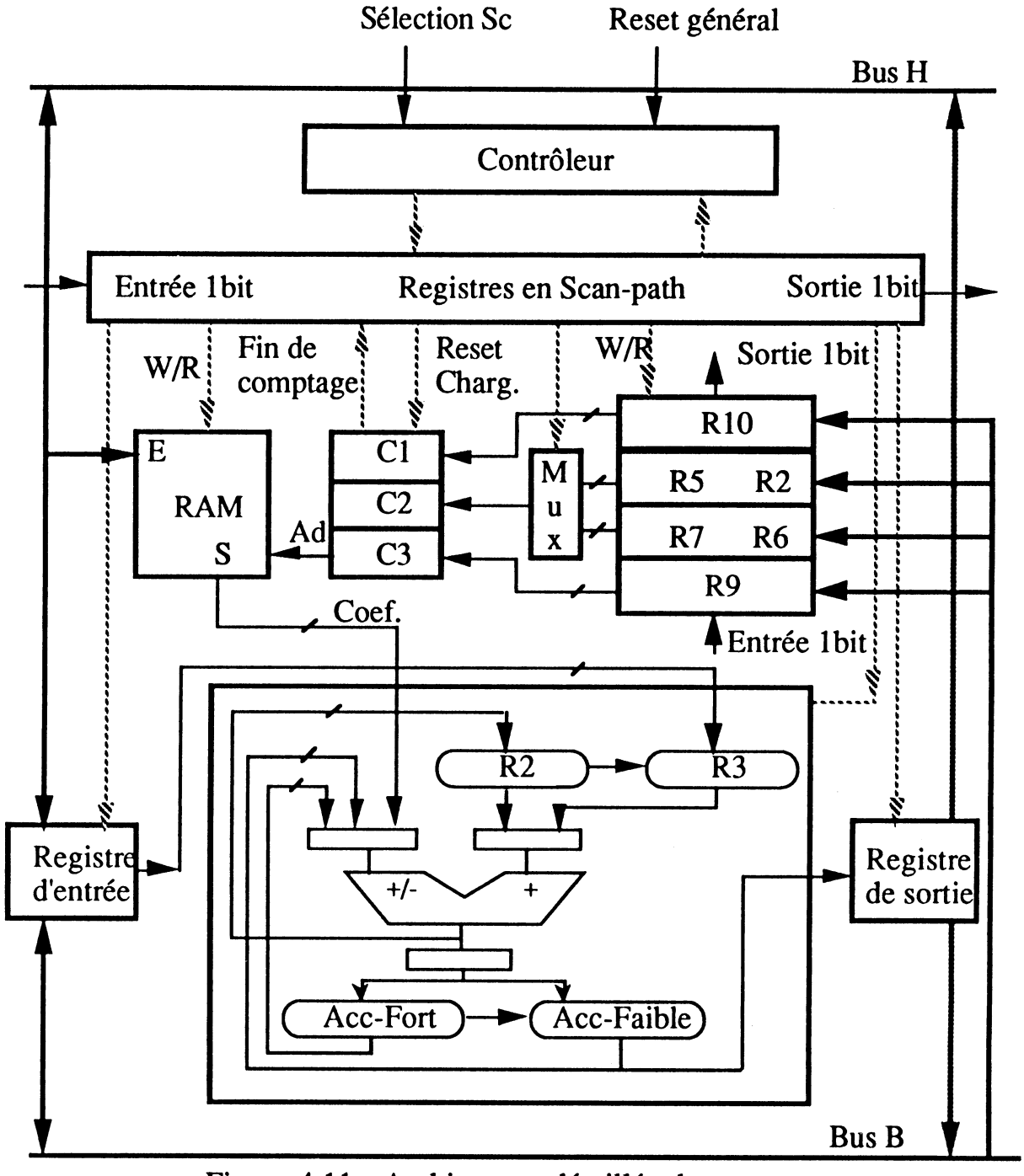


Figure 4.11 : Architecture détaillée du neurone

Pendant le fonctionnement normal du circuit, la liaison série entre ces registres sera coupée par programmation, permettant une liaison directe et parallèle entre les deux parties.

La partie contrôle ne pose aucun problème particulier de testabilité. L'accès direct à ce module par l'intermédiaire des registres Scan-path offre une bonne contrôlabilité et une observabilité à tout moment du test.

Les signaux de contrôle à l'entrée des différents modules de la partie opérative sont directement contrôlables à travers les registres Scan-path. L'étude de la testabilité des modules revient donc à étudier la contrôlabilité de leurs signaux de données et l'observabilité de leurs signaux de sortie. L'entrée des signaux de données vient du bus, les modules de la partie opérative sont donc contrôlables.

Une première étape permet de tester la mémoire synaptique, l'additionneur/soustracteur, le registre R2, les deux accumulateurs des poids forts et faibles.

En effet, on effectue une addition d'un coefficient C, lu dans la mémoire, et d'un zéro, lu dans le registre R2. Le résultat est stocké dans l'accumulateur fort puis décalé dans l'accumulateur faible. L'observation est faite par le registre de sortie (figure 4.12).

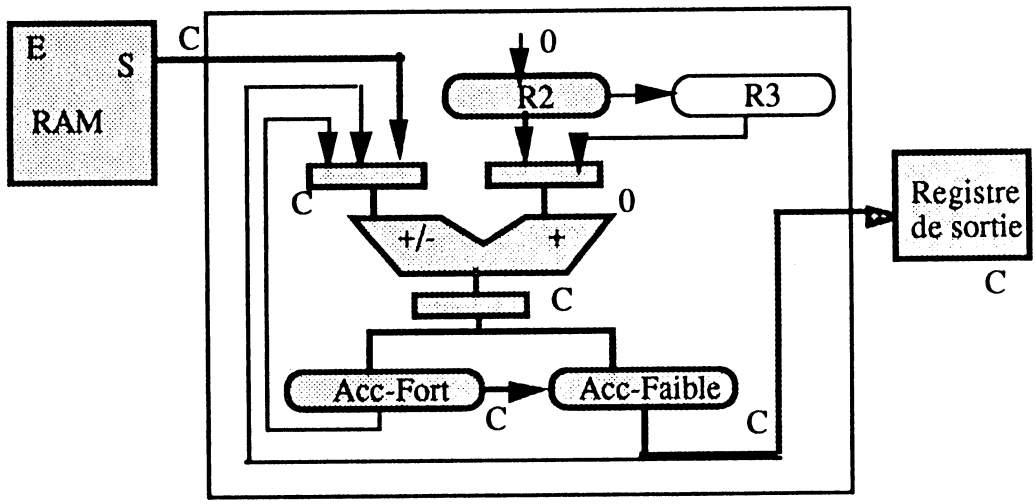


Figure 4.12 : Testabilité de la mémoire, de R2, de l'additionneur soustracteur, des accumulateurs.

Une deuxième étape permet de tester le registre R3. Ce dernier est chargé par le registre d'entrée E. On impose une valeur nulle à l'entrée de l'additionneur par l'intermédiaire de l'accumulateur fort. Le résultat de l'addition sera stocké dans le registre R2 et l'accumulateur faible.

Le contenu de R2 est décalé dans R3. Puis on effectue une addition entre le contenu de l'accumulateur faible et R3 ; le résultat, égal à deux fois la valeur initialement chargée, est observé par le registre de sortie (figure 4.13).

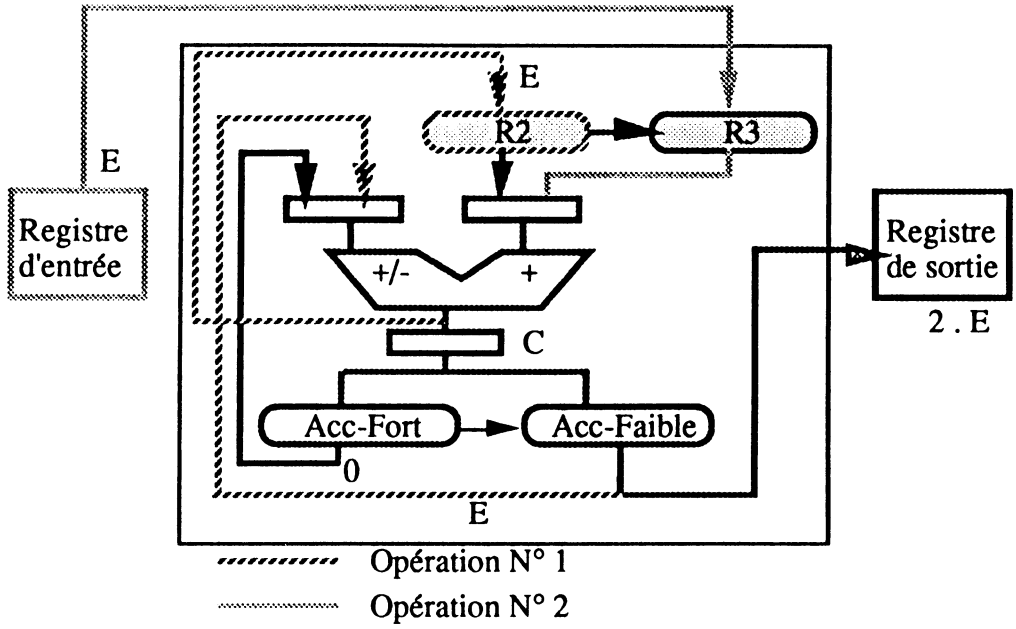


Figure 4.13 : Testabilité du registre R3

Les registres de personnalisation, qui attaquent directement des compteurs et qui n'ont aucune liaison avec l'extérieur, posent un problème d'observabilité. Pour éviter des connexions supplémentaires et éviter d'augmenter le nombre de plots de sortie, on a choisi de les monter en "scan-path" (figure 4.14) moyennant un plot de commande.

Les modules les plus difficiles à tester sont les compteurs. En effet, on ne dispose pas de sortie observable des états des compteurs. Les compteurs C1 et C2 peuvent être contrôlés seulement après la fin du comptage grâce à un bit indiquant la fin de l'opération. Ce bit est lu par l'un des registres en "scan-path" monté entre la partie opérative et la partie contrôle.

Le troisième compteur est testable via la mémoire. En effet, la mémoire est adressée par la sortie du compteur C3. En connaissant à priori la valeur mémorisée dans une adresse donnée, il suffit de lire cette valeur dans la mémoire et de la faire propager jusqu'au registre de sortie.

Remarquons que le neurone fabriqué actuellement n'a pas disposé de ces facilités. Les méthodes de synthèse très automatisées du type compilateur de silicium ont permis d'obtenir des circuits bons dès le premier silicium pour ce qui concerne la conception et bien sûr nous n'avons pas été confrontés à un problème de tri de haute qualité d'un lot important (nous avons reçu 10 circuits du CMP Suisse).

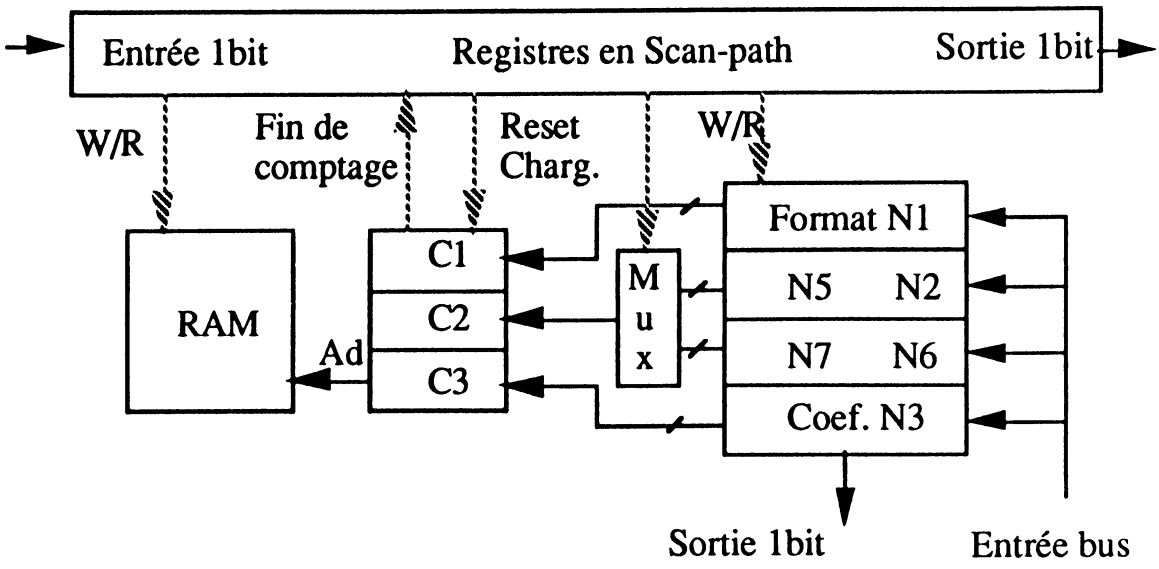


Figure 4.14 : Testabilité des registres de personnalisation et des compteurs

4.3 Organisation physique

Deux procédés de fabrication sont utilisés pour l'intégration à très grande échelle de la tranche entière, la fabrication par photo-répétition ou par faisceau électronique. Le premier procédé consiste à dupliquer un module, appelé réticule, sur toute la tranche. Par contre, le second utilise une insolation directe de la tranche, sans masque, au moyen d'un faisceau d'électrons. Nous avons choisi de détailler le premier pour des raisons de simplicité et à cause du faible coût du procédé de fabrication. La dimension maximale d'un réticule peut atteindre 15 mm de côté. La figure 4.15 montre l'organisation sur tranche des réticules, les bus d'entrée/ sortie des données, les signaux de contrôle externes et les bus d'alimentations. La structure du réticule est constituée par un arrangement 2-D des processeurs neurones (figure 4.16). Les lignes de neurones sont séparées par des lignes de bus traversant toute la tranche.

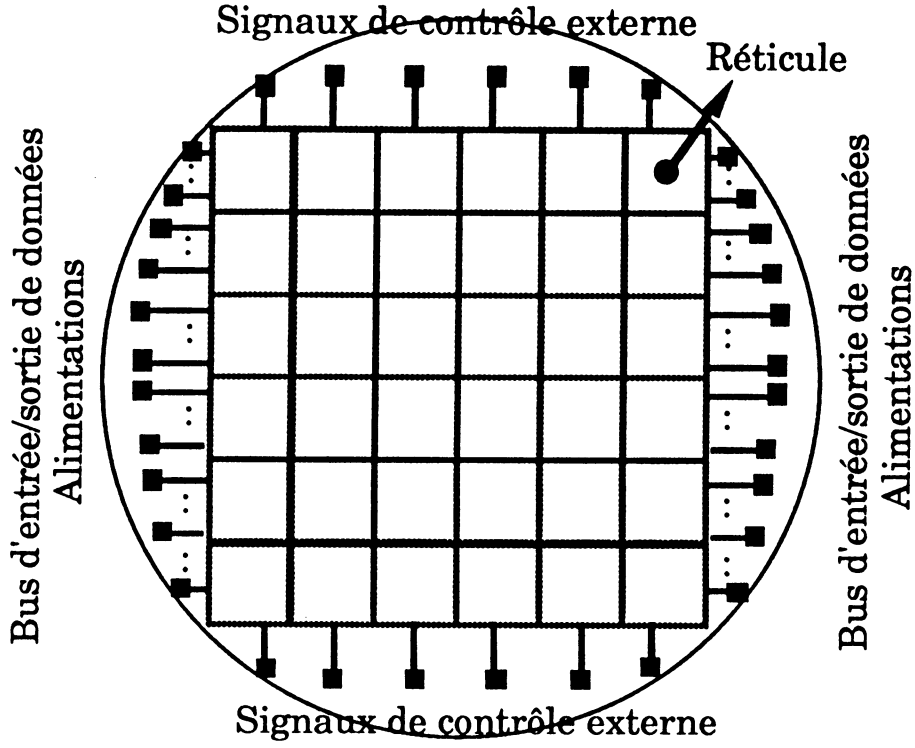


Figure 4.15 : Structure de la tranche de silicium

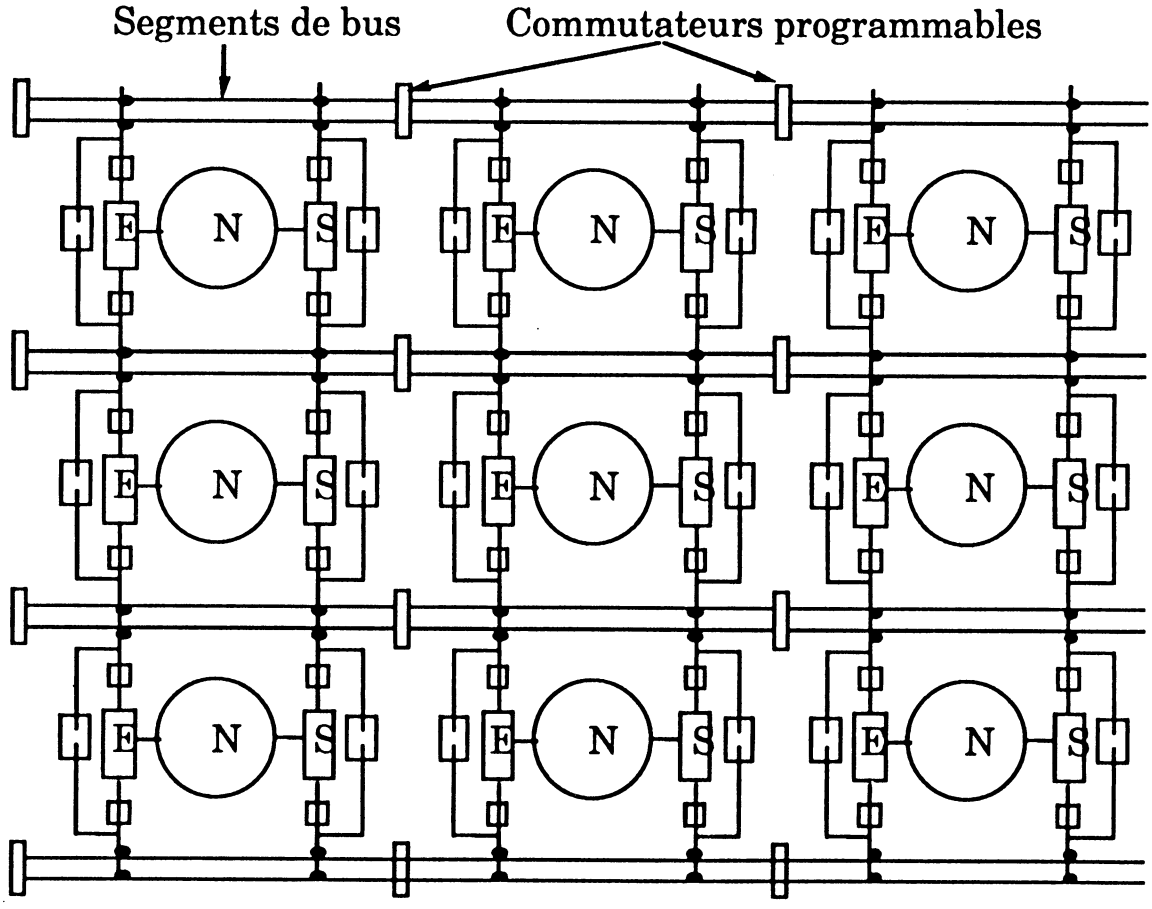


Figure 4.16 :Architecture implantée dans le réticule

4.3.1 Stratégie de distribution d'horloge

Le passage de l'intégration VLSI à l'intégration sur tranche pose également un problème de rendement des lignes communes. Il est en effet clair que la possibilité d'avoir les signaux de contrôle globaux sans défaut est quasi nulle. Il faut ainsi implanter de la redondance sur les lignes vitales [War88].

La stratégie de distribution de signaux d'horloge consiste à utiliser une stratégie à signaux multiples sur le diamètre de la tranche pour générer le signal d'horloge de fonctionnement du processeur neurone. La figure 4.17 montre la distribution des signaux d'horloge [Fri86] [Col86].

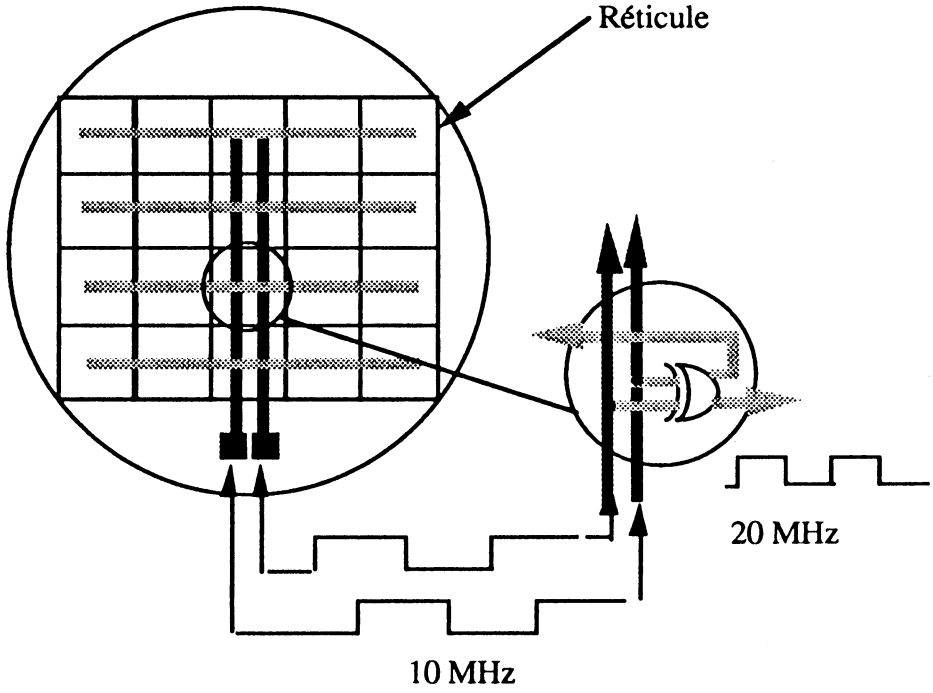


Figure 4.17 : Distribution d'horloge

4.3.2 Stratégie de distribution des signaux de contrôle

Les signaux de contrôle externes sont le signal d'initialisation générale des neurones de la tranche et celui de sélection de la colonne. Le signal commun à tous les neurones sera réalisé par une grille qui offre une multitude de possibilités pour connecter un point du circuit aux plots externes, donc offre une bonne tolérance aux coupures de la ligne d'aluminium, donc une meilleure tolérance aux défauts de fin de fabrication (figure 4.18.a). Cette structure utilise plusieurs chemins pour assurer une bonne alimentation de chaque module.

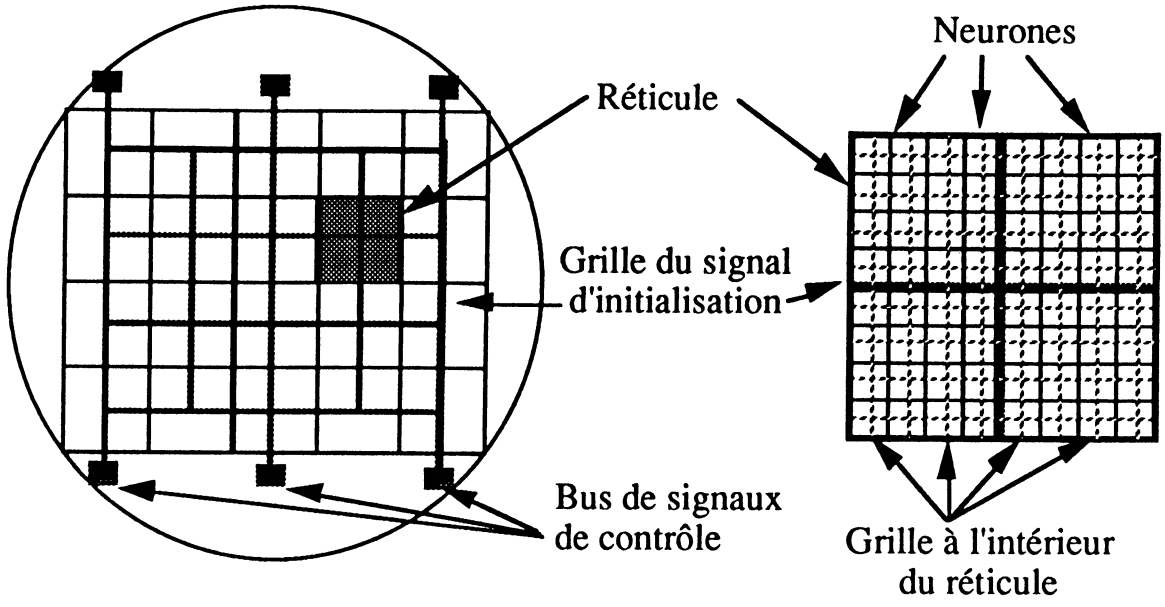


Figure 4.18.a : Distribution du signal d'initialisation

Quant au signal de sélection de la colonne de neurones, il sera dupliqué et alimenté de chaque côté de la tranche. Afin d'augmenter la tolérance aux défauts, les lignes seront reliées à l'intérieur de chaque neurone (figure 4.19.b).

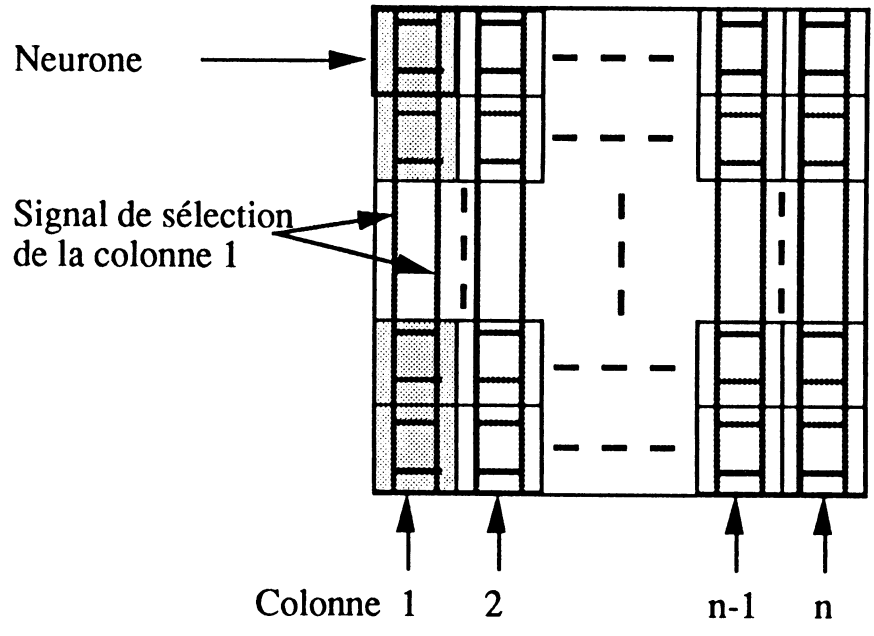


Figure 4.19.b : Distribution des signaux de contrôle de sélection des colonnes

4.3.3 Stratégie de distribution de l'alimentation

Parmi plusieurs stratégies existantes - grille complète, demi-grille, quart de grille et des rails indépendants - employées pour distribuer et alimenter la tranche de silicium, on a choisi la stratégie à rails indépendants (figure 4.20). Cette stratégie fournit une bonne tolérance aux défauts puisque seulement les modules reliés à certains rails seront affectés. Pourtant, des chutes de tension vers le centre de la tranche seront possibles [Joh89].

Les lignes de l'alimentation seront dupliquées, cela permettra également d'alimenter suffisamment le réseau dans un pire cas de chargement. A l'intérieur des réticules, les lignes d'alimentation seront organisées en peigne.

L'intérêt de cette structure est d'éviter tout croisement entre les deux lignes d'alimentation Vdd et Vss. De tels croisements peuvent être à l'origine de court circuits qui sont des défauts fatals et réduisent le rendement de la tranche. C'est pourquoi nous avons choisi des lignes d'alimentation redondantes et organisées en peigne [War86].

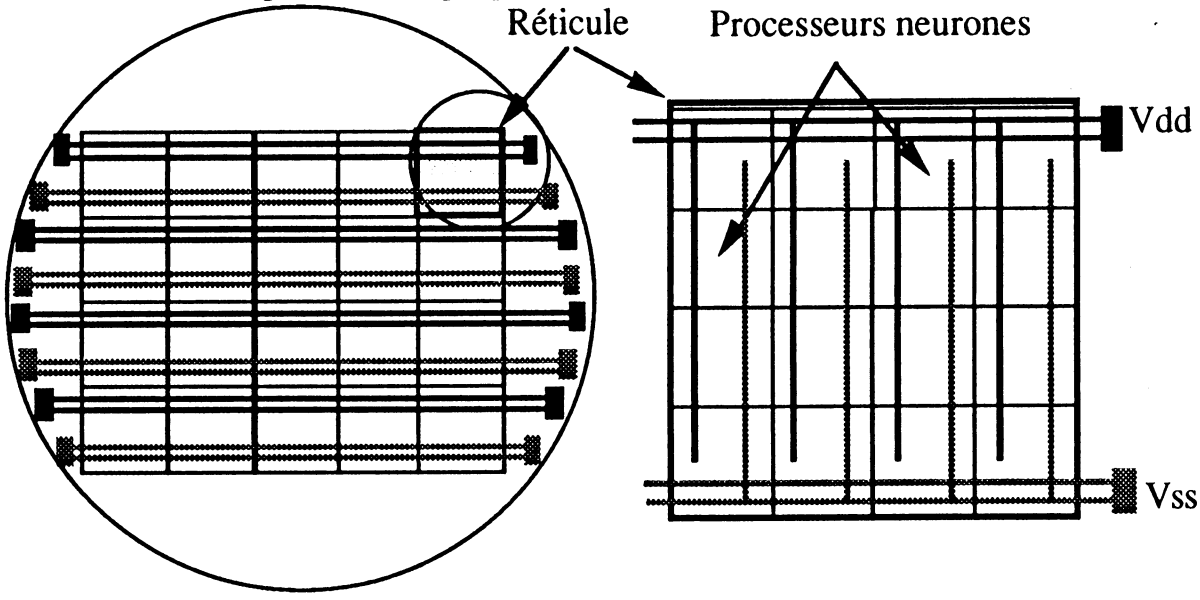


Figure 4.20 : Distribution de l'alimentation de la tranche

Cette pré-étude n'est pas susceptible de prolongement immédiat au laboratoire pour plusieurs raisons, entre-autres le manque de moyens financiers pour une réalisation physique sur tranche entière qui coûte très cher, et le manque de moyen matériel car la simulation d'un réseau en tranche entière est impossible actuellement sur des machines non capables de traiter d'aussi gros réseaux.

Conclusion Générale

L'intérêt des circuits dédiés pour réaliser des réseaux neuromimétiques s'est considérablement accru au cours de la dernière décénie. Dans un premier temps, l'intérêt s'est porté en priorité sur la technologie analogique, plus proche du modèle biologique. Si des réalisations dédiées efficaces ont été rapidement obtenues, le manque de flexibilité a très vite posé des problèmes rédhibitoires. En effet, il est clair que l'état de l'art dans le domaine du calcul neuronique n'est pas du tout stabilisé. Les applications intéressantes ne sont pas complètement identifiées à ce jour et les réalisations sur silicium doivent pouvoir s'adapter à l'évolution des applications.

Les principales recherches consistent à simuler sur des ordinateurs massivement parallèles la phase d'apprentissage pour les applications qui semblent adaptées à l'approche neuronique. Une application sera une bonne candidate si, après le choix de la précision des coefficients et du code des états, des coefficients synaptiques de bonne qualité ont été déterminés ; ceci permettra alors de définir le réseau de neurones réalisant efficacement la phase de reconnaissance pour cette application. Dans le cas des applications pour lesquelles on aura prouvé l'intérêt de l'approche neuromimétique, des circuits dédiés peuvent être réalisés.

Dans ce contexte, il était particulièrement important de s'intéresser à des réalisations digitales flexibles. Le but de cette thèse a donc été de proposer une architecture de processeur neurone dite personnalisable. A partir de l'architecture de base, il sera simple d'en générer des versions pour lesquelles les coefficients et les états seront codés sur un nombre de bits donné et implantant exactement les fonctionnalités souhaitées. Pour ceci, une approche s'appuyant sur des méthodes de synthèse automatisée élaborées a été proposée.

La première partie du travail a consisté à définir un chemin de données, ou partie opérative, relativement simple comprenant essentiellement un additionneur/soustracteur, des registres et des compteurs. Les fonctionnalités pouvant être réalisées sur cette partie opérative sont décrites au niveau de transfert de registres. A chaque fonction de haut niveau (calcul du potentiel, fonction d'activation, etc...) correspond une portion d'organigramme de contrôle ; pour réaliser un neurone dédié, les portions d'organigramme requises sont assemblées. A partir de cette description, une machine d'états

finis peut être extraite automatiquement. Dans un environnement de synthèse sophistiqué, la partie opérative est générée avec un nombre de bits paramétrable et la synthèse de machines d'états finis fait appel à un outil classique.

L'architecture proposée dans cette thèse est originale et apporte un maximum de flexibilité ; le processeur neurone est autonome, personnalisable et cascadable. La flexibilité de l'implantation repose sur les techniques d'avant garde de synthèse automatisée à partir d'organigrammes de contrôle ; c'est seulement grâce à un tel environnement que cette thèse a pu aboutir. Remarquons également que de très bons outils incluant les compilateurs de chemins de données et un générateur de machines d'états finis nous ont permis de nous affranchir de préocupations trop proches du silicium et de réussir d'emblée la réalisation d'un processeur neurone validant notre approche. Une simulation nous a également permis d'affirmer qu'un réseau assemblant de tels neurones cascadables réalise effectivement la phase de reconnaissance.

Nous pensons donc dans cette thèse avoir exploré en profondeur la notion de processeur neurone digital flexible et tout le bénéfice que l'on peut en attendre en phase de reconnaissance. La suite de la thèse a consisté à identifier les limites de cette architecture. Le processeur neurone ayant une structure classique de partie contrôle et partie opérative, il est clair que ces fonctionnalités peuvent être étendues dans une large mesure ; il convient néanmoins d'en limiter la complexité. La partie opérative est supposée avoir une structure globale fixe et seule l'introduction de nouvelles parties d'organigrammes de contrôle a été envisagée. Les communications par les segments de bus étant parfaitement générales, il était supposé que cette architecture pouvait également servir de support à des phases d'apprentissage.

Il s'agissait donc de montrer pour des techniques classiques d'apprentissage (rétropropagation du gradient en particulier) que les moyens de communication entre neurones et la puissance de calcul (opérations de la partie opérative) sont suffisants pour permettre l'implantation de l'apprentissage. Ceci a été démontré dans le chapitre 3. Il est clair qu'il ne s'agit que d'une étude de faisabilité car seuls des circuits implantant la reconnaissance constituaient l'objectif premier de cette thèse. L'étude de la précision des formats des états et des coefficients, ainsi que l'augmentation réelle de la surface du neurone implantant la phase d'apprentissage sont en dehors de l'objectif de cette thèse.

Enfin, dans un dernier chapitre, l'extention à de très gros circuits et la possibilité d'une réalisation sur tranche entière ont été étudiée. Une telle réalisation nécessite l'implantation de techniques de tolérance aux défauts pour pallier les limites de rendement. En conséquence, les points clés d'une telle implantation ont été analysés et on peut en conclure raisonnablement que les réseaux de neurones sont un des meilleurs candidats pour l'intégration sur tranche entière.

Comme suite aux travaux menés dans cette thèse, il est envisagé de mettre à profit la souplesse de l'architecture proposée dans une approche de compilateur de silicium. On propose de réaliser un passage aussi automatisé que possible d'une spécification de haut niveau utilisée, par exemple, à des fins de simulation, à une réalisation matérielle sur silicium. Il serait alors possible de fournir dans un temps court des circuits dédiés implantant la phase de reconnaissance.

L'approche consisterait à extraire les caractéristiques définissant la structure du réseau et la structure du processeur neurone à partir d'une description dans un langage de haut niveau. L'utilisation d'outils de synthèse de haut niveau et de génération de blocs permettrait alors de générer le processeur neurone et le réseau correspondant, comme nous l'avons montré dans un temps court.

Les travaux menés dans cette thèse ont montré que la réalisation rapide de réseaux de neurones dédiés sur silicium est possible et efficace. Une architecture universelle et un processeur neurone simple, capable d'effectuer toutes les opérations arithmétiques du calcul neuromimétique, offrent à l'utilisateur une grande flexibilité d'implantation d'applications spécifiques.

En conclusion, nous pensons que cette thèse aura contribué à avancer l'état de l'art dans la conception de circuits digitaux pour le traitement neuronique et que l'architecture proposée, de par sa flexibilité et les méthodes de conception utilisées, pourra conduire à de nombreuses applications.

Bibliographie

[Ali90] C. Alippi, S. Bonfanti and G. Storti-Gajani, "Approximating sigmoïdal Functions for Digital Implementation of Neural Nets," Proc. of the First Int. Work. on Microelectronics for Neural Networks, Dortmund, Germany, pp.165-170, June 1990.

[All90] P. Alla, G. Dreyfus, J.D. Gascuel, A. Johannet, L. Personnaz, J. Roman and M. Weinfeld, "Silicon Integration of Learning and other auto-Adaptative Properties in a Digital Feedback Neural Network," Proc. of the First Int. Work. on Microelectronics for Neural Networks, Dortmund, Germany, pp.40-46, June 1990.

[Als87] Alspector J. et Al., "A neuromorphic VLSI Learning System," Proc. of the Stanford Conf. Advanced Research in VLSI. pp.313-349, 1987.

[And88] J.A. Anderson and E. Rosenfeld, Neurocomputing, Fundations of Research, MIT Press, Cambridge, MA, 1988.

[Bal86] D.H. Ballard, "Cortical Connections and Parallel Processing : Structure and Function," The Behavioral and Brains Sciences, 9, pp.67-120, 1986.

[Bie86] E. Bienenstock et al, "Disorded Systems and Biological Organization," Springer, 1986.

[Boo61] A.D. Booth, "A Signed Binary Multiplication Technique," Quarterly Journal of Mechanics and Applied Mathematics, Vol.4, part 2, pp.236-240, 1961.

[Cha90] A. Chams, J.C. Lawson et J. Herault, "SMART : Comment simuler de très grands réseaux," Journées Neurosciences et sciences de l'ingénieur, NSI 90, pp.101-104, Aussois, France, Mai 1990.

[Col86] J.N. Coleman and R.M. Lea, "Clock Distribution Techniques for WaferScale Integration," Proc. of a Workshop Held in Southampton, July 1985, pp.47-54. Wafer Scale Integration, Edited by C. Jesshope and W. Moore, Published by A. Hilger1986.

[Cun87] Y. Le Cun, "Modèles Connexionnistes de l'apprentissage," Thèse de Doctorat, Université de Paris 6, 1987.

[Cun89] Y.le Cun, "Generalization and Network Design Strategies" Elsevier Science Publishers, North-Holland,1989.

[Den87] J.S. Denand B.S. Wittner, "Network Generality Training Required and precision required". IEEE Conf. Neural Information Processing Systems-Natural and Synthetic, November 1987.

[Dur88] M. Duranton, J. Gobert and N. Mauduit, " A Digital VLSI Module for Neural Networks" Proc. nEuro88, IDSET, Paris 1988, pp.720-724.

[Fau89] B. Faure and G. Mazaré, "A VLSI Asynchronus Cellular Architecture Dedicated to Multilayered Neural Networks," Proc. nEuro88, IDSET, Paris 1988.

[Fle80] W.I. Fletcher, "Engineering Approach to Digital Design" Prentice-Hall International Editions, Englewood Cliffs, 1988.

[Fri86] J. Fried, "An Analysis of Power and Clock Distribution for WSI Systems," Wafer Scale Integration, pp127-141. Edited bu G.Saucier and J. Trilhe, Published by Elsevier, 1986.

[Fuk88] K. Fukushima, "A Neural for Visual Pattern Recognition" IEEE Computer, pp.65-75, March 1988.

[Gar87] S.C.J. Garth, "A Chipset for High Speed Simulation of Neural Networks Systems" Proc. IEEE First Inter. Conf. Neural Networks, IEEE Press, pp. III-443-452, June 1987.

[Gas90] J.D. Gascuel, P. Alla, J. Roman and M. Weinfeld, "Implementation of internal Annealing and Self-Identification of Successful Prototype Retrieval in VLSI Feedback Neural Network Chip," Eurasip Inter. Work. on Algorithms and Parallel VLSI Architecture, Pont à Mousson, France, June 1990. Elsevier 1991.

[Gos84] K. Goser, C. Folster and U. Rueckert, "Intelligent Memories in VLSI" Information Science, Vol.34, pp.61-82, 1984.

[Gra88] H. Graf, L.Jackel and Hubbard, "VLSI Implementation of Neural Networks Model," Computer, pp.41-49, March 1988.

[Gue87] A. Guérin, "Un Calculateur de réseaux Adaptatifs Systoliques CRASY. Application au Calcul Neuromimétique" Thèse de Docteur Ingénieur en Electronique, INP Grenoble, 1987.

[Guy88] I. Guyon, "Réseaux de Neurones pour la Reconnaissance des Formes : Architecture et Apprentissage," Thèse de Doctorat de l'Université Paris VI, Décembre 1988.

[Ham90] D. Hammerstrom, "A VLSI Architecture for High Performance, Low-cost, on-Chip Learning," Proc. Inter. Joint Conference on Neural Network, San Diego, CA, June 1990.

[Hec86] R. Hecht-Nielsen, "Performance Limits of Optical, Electro-Optical and Electronic Neurocomputers," Optical and Hybrid Computing SPIE, Vol.634, pp.277-306, 1986.

[Hec88] R. Hecht-Nielsen, "Neurocomputing : Picking the Humain Brain" IEEE Spectrum, Vol.25. N°3,March 1988.

[Her80] J. Hérault, "Le traitement de l'information dans les structures nerveuses. Etude par simulation numérique et électronique, applications aux traitement des signaux" Thèse de Docteur ès Sciences, Université I et INP Grenoble, 1980.

[Hin86] G.E. Hinton, "Learning to Recognize Shapes in a Parallel Network," Proc. of the Fyssen Meeting on Vision, Paris, Mars 1986.

[Hin87] G.E. Hinton, "Connectionist Learning Procedures" Technical Report, Computer Science Department, Carnegie-Mellon University, pp.1-46, June 1987.

[Hoe90] J. Hoekstra, "Junction Charge-Coupled Device Technology for Artificial Neural Networks," Proc. of the First Int. Work. on Microelectronics for Neural Networks, Dortmund, Germany, pp.62-68, June 1990.

[Hol89] M. Holler, S. Tam and H. Castro, "An Electrically Trainable Artificial Neural Network with 10240 Floating Gate Synapses" IJCNN, Washington D.C., pp.II.191-196, June 1989.

[Hop82] J.J. Hopfield, "Neural Networks and Physical Systems with emergent Collective Computational Abilities," Proc. Nat. Acad. Sciences, volume 79, pp.2554-2558, 1982.

[Joh89] K.K. Johnstone and J.B. Butcher, "Power Distribution for Highly Parallel WSI Architectures," International Conference on WSI, pp203-214, january 1989, San Francisco, California.

[Joh90] A. Johannet, "Réseaux de neurones Formels : Etude d'un Simulateur à Architecture Parallèle et Conception d'un Circuit Intégré," Thèse de Doctorat de l'Université de Paris VI, 1990.

[Jon88] D.W.Jones, "The Ultimate RISC," Computer Architecture News, Vol.16, N°3, pp.48-55, June 1988.

[Kuc87] R.M. Kuczewski, M.H. Myers and W.J. Crawford, "Neurocomputer Workstations and Processors : Approches and Applications," Proc. IEEE First Int. Conf. on Neural Networks, pp.III-487-500, June 1987.

[Kun88] S.Y. Kung and J.N. Hwang, "Digital VLSI Architectures for Artificial Neural Nets," Neural Networks for Computing, Snowbird, Utah, April, 1988.

[Lip87] R.P. Lippmann, "An Introduction to Computing with Neural Nets" IEEE ASSP Magasine, pp.4-22, April 1987.

[Moo88] A. Moopenn & al, "A Programmable Binary Synaptic Matrix Chip for Electronic Neural Networks," D.Z. Anderson ed. Neural Information Processing Systems, Natural and Synthetic. American Institute of Physics, 1988.

[Mor90] N. Morgan, "Digital Implementation of Artificial Neural Networks," Proc. of the First Int. Work. on Microelectronics for Neural Networks, Dortmund, Germany, pp.187-195, June 1990.

[Mur87] A.F. Murray et al., "Bit-Serial Neural Networks," Proc. IEEE Conf. Neural Information Processing Systems, pp.573-583, November 1987.

[Mye89] D.J. Myers and R.A. Hutchinson, "Efficient Implementation of Piecewise Linear Activation Function for Digital VLSI Neural Networks," Electronics Letters, Vol.25, N°24, pp.1662-1663, November 1989.

[Pac89] M.Pacheco and P.Trealeven, "A VLSI Word-Slice Architecture for Neurocomputing," Proc.Inter. Symp. Computer Architrecture and Digital Signal Processing, IEEE Hong Kong Center, Hong Kong, 1989.

[Per86] L. Personnaz, "Etude de Réseaux de Neurones Formels : Conception, Propriétés et Applications," Thèse de Doctorat d'Etat de l'Université Paris VI, 1986.

[Per88] L. Personnaz, A. Johannet G. Dreyfus and M. Weinfeld, "Towards a Neural Network Chip : A Performance Assessment and a Simple Example" Proc. nEuro88, pp.682-691, IDSET Paris 1988.

[Per90] P. Peretto, R. Van Zurk, A. Mougin and C Gamrat, "The semi-parallel Architectures of Neuro-computers" Conf. NATO ARW, Neurocomputing : Algorithmes Architectures and applications, Les Arcs, France, Février 1990, ed by Springer Verlag 1990.

[Pet89] A. Petrowski, L. Personnaz, G. Dreyfus and C. Girault, "Parallel Implementations of Neural Network Simulations," Hypercube and Distributed Computers, J.P.Verjus, F. André eds, North-Holland,1989.

[Pyg89] ESPRIT II - Pygmalion Conference Brussels, Belgium. November 1989

[Ram89] U. Ramacher and J. Beichter, "Systolic Architectures for Fast Emulation of Artificial Neural Networks" Proc. Inter. Conf. Systolic Arrays, IEEE CS Press, Los Alamitos, California , May 1989.

[Ros60] F. Rosemblatt, "Perceptron Simulation Experiments," Proc. IRE, Vol. 3, No.48, New York, 1960.

[Ros89] O. Rossetto, C. Jutten, J. Hérault, I. Kreuzer, "Analog VLSI Synaptic Matrices as Building Blocks for Neural Networks" IEEE Micro, Vol. 9, n° 6, p. 56, December 1989.

[Rou90] P. Roussel-Ragot and G. Dreyfus, "A Problem-Independent Parallel Implementations of Simulated Annealing : Models and Experients," IEEE Trans. on Computer-Aided, Vol.9, P827, 1991.

[Rue88] U. Rueckert and K. Goser, "VLSI Architectures for Associative Networks" Proc. Int. Symp. Circuits and Systems, pp.755-558, 1988.

[Rum86] D. Rumelhart, G. Hinton and R. Williams, "Learning Internal Representations by Error Propagation," in Parallel Distributed Processing Exploring the Micro-structure of Cognition, D. Rumelhart J. Mc Clelland (eds), MIT Press, 1986

[Sai88] SAIC, "Delta/Sigma/Ansim" Neurocomputers, Vol.2, N° 1, 1988

[Sei84] C.L.Seitz, "Concurrent VLSI Architecture," IEEE Trans. Computers, Vol.C33, N°12, pp.1,247-1,264, Dec.1984.

[Sej86] T.J. Sejnowski and C.R. Rosenberg, "NETtalk, a Parallel Network that Learns to Read Aloud," Tech Report 86-01, Dept of EE-CS, Johns-Hopkins University, Baltimore, MD, 1986.

[Sou88] B. Soucek and M. Soucek, "Neural and Massively Parallel Computers" John Wiley & Sons, 1988.

[Thi89] S. Thiria, "L'apprentissage supervisé dans les Modèles Connexionnistes," Thèse de Doctorat, Université René Descartes Paris 5, 1989.

[Tre89] P. Treleaven and M. Vellasco, "Neural Networks on Silicon" Proc. 3rd IFIP Workshop Wafer Scale Integration, IFIP, Como, Italy, June 1989.

[Tre89] P. Treleaven, M. Pacheco and M. Vellasco, "VLSI Architectures for Neural Networks" IEEE Micro, pp.8-27, December 1989.

[Tsi89] Y.P. Tsividis, D. Anastassiou, "Switched Capacitor Neural Network, Electronics Letters 23, 958-959, 1989.

[Ver89] M. Verleysen, B. Sirletti, P. Jespers, "A New VLSI architecture for Neural Network Associative Memories," Proc. nEuro88, IDSET, Paris 1988.

[Vit90] E. Vittoz & al., "Analog Storage of Adjustable Synaptic Weights," Proc. of the First Int. Work. on Microelectronics for Neural Networks, Dortmund, Germany, pp.69-79, June 1990.

[VLS90] Manuels d'utilisation du logiciel VLSI Technology Inc., 1991

[Wan89] S. Wang, "Réseaux Multicouches de Neurones Artificiels," Thèse de Doctorat, Institut National Polytechnique de Grenoble, 1989.

[War86] K.D. Warren & al, "A Power Distribution Strategy for WSI," Proc. of a Workshop Held in Southampton, July 1985, pp.55-63. Wafer Scale Integration, Edited by C. Jesshope and W. Moore, Published by A. Hilger1986.

[War88] K.D. Warren, "Electrical Design Issues for WSI," Wafer Scale Integration pp191-199. Edited by R.M. Lea and Published by Elsevier, 1988.

[Wei88] M. Weinfeld, "A Fully Digital Integrated CMOS Hopfield Network Including the Learning Algorithm," Int. Workshop on VLSI for Artificial Intelligence, Oxford, England, July 1988.

[Wei89] M. Weinfeld, "Integrated artificial neural networks : components for higher level architectures with new properties," NATO Advanced Research Workshop on Neurocomputing : Algorithms, Architectures and Applications, Les Arcs, France, Springer 1989

[Yas90] M. Yasunaga, N.Masuda, M. Yagyu, M. Asai, M. Yamada, A. Masaki, "Design, Fabrication and Evaluation of a 5 inch Wafer Scale Neural Network LSI Composed of 576 Digital Neurons," Proc. Inter. Joint Conference on Neural Network, San Diego, CA, June 1990.

[Zub89] M. Zubair and B.B. Madan, "Systolic Implementation of Neural Networks," IEEE Inter. Conf. on Computer Design : VLSI in Computers & Processors (ICCD'89), pp.479-482, 1989.

Publications

[1] J. Ouali, G. Saucier and J. Trilhe "Neural Networks on silicon : a myth or a reality ? ", Electronic Design Automation Conference (EDA88), Wembley, UK, June 1988.

[2] J. Ouali, G. Saucier and J. Trilhe, "Réseau neuromimétique sur silicium: mythe ou réalité ?", 1st International Workshop Neural Networks and their Applications (Neuro-Nîmes 88), Nîmes, France, November 1988.

[3] J. Ouali, G. Saucier and J. Trilhe "A Flexible, Universal Wafer Scale Neural Network", 3rd International Workshop on Wafer Scale Integration, Come, Italy, June 1989.

[4] J. Ouali, G. Saucier, "A Distributed Architecture for Neural Networks Based on a Neural Processor". 2nd International Workshop Neural Networks and their Applications (Neuro-Nîmes 89), Nîmes, France, November 1989.

[5] J. Ouali, G. Saucier, "A Flexible Wafer Scale for Neural Networks". International Conference on Computer Design (ICCD'89), Cambridge, USA, October, 1989.

[6] J. Ouali , G. Saucier , J.Trilhe, "Architectures Neuroniques", Journée d'études, Architectures Parallèles et Applications aux traitement d'Images. ENSERG, Grenoble, France, Mai 1990.

[7] J. Ouali, G. Saucier, "Neural Processor for Neural Networks on Silicon", European Conference Specific Integrated Circuits Design, EURO ASIC 90, Paris , France, May 1990.

[8] J. Ouali, G. Saucier and J. Trilhe, "Design of a Neuron processor Including Learning Capabilities", ITG/IEEE Workshop on Microelectronics for Neural Networks, Dortmund, Germany, June 1990.

[9] J. Ouali, G. Saucier, "Fast Generation of Neuro-ASICs", International Neural Network Conference (INNC 90), Paris, France, June 1990.

[10] J. Ouali, G. Saucier, "Silicon Compiler for Neuro-ASICs", International Joint Conference on Neural Networks (IJCNN 90), San Diego, California, June 1990.

[11] J. Ouali, G. Saucier, "Neuron Processor on Silicon", International Conference on Microelectronics ICM'90, Damascus, Syria, October 1990.

[12] J. Trilhe and J. Ouali, "Customizable Artificial Neural Network on Silicon", Revue technique Thomson CSF Vol.22 N°3, ed Gauthier- Villars, Septembre 1990.

[13] J. Ouali, G. Saucier and J. Trilhe, "Distributed Large Neural Networks on Silicon", IFIP Workshop on Silicon Architectures for Neural Nets, St Paul de Vence, France, Nov. 1990.

[14] J. Trilhe and J. Ouali, "Fast Design of Digital Dedicated Neuro Chip", Revue technique Thomson CSF Vol.23 N°1, ed Gauthier-Villars, Mars 1991.

Annexes

Légende

Sc1, Sc2, Sc3 et Sc4 sont les signaux de sélection des colonnes respectives C1, C2, C3 et C4.

Clk : Horloge de base

Toutes les instances visualisées sont automatiquement précédées par le nom du neurone auquelles elles appartiennent. Par exemple, le registres R9 appartenant au neurone n11 est écrit comme n11.R9.

Les bus de chargement sont les bus 12 et bus 13. Et ceux de lecture des états sont les bus 22 et bus 23.

Les compteurs utilisés sont :

- comtc1 utilisé pendant la multiplication

- comtc2 utilisé pour le décalage des vagues de données et le formatage des multiplications

- outc3 compteur d'adresse mémoire

Les commandes d'interrupteurs sont :

- t12L : ouverture du buffer bas dans le sens gauche droite

- t12h : ouverture du buffer haut dans le sens gauche droite

- t21h : ouverture du buffer haut dans le sens droite gauche

La commande de décalage vers le bas est notée buflow et celle de décalage vers le haut est notée bufhigh.

accfbR et accftR correspond aux accumulateurs des poids faibles et forts. Ces accumulateurs contiennent le résultat du potentiel puis les résultats intermédiaires de l'état du neurone.

Le registre insb contient le bit de signe

Le registre st contient le bit de saturation

Le registre sn contient l'état de sortie du neurone

RR2 et RR3 correspond aux registres qui contiennent le résultat partiel des multiplications

Un cycle de simulation est définit comme suit :

Time = numéro du cycle : numéro de phase de l'horloge [temps en ns]

liste des signaux activés ou désactivés

liste des états des instances modifiées au cours du cycle

Fichier de chargement des registres de personnalisation et des coefficients

```
#cell2 * stimuli_8_neurones_bis1 txt * 9 any 0 v8r1.8

# "3-Apr-91 GMT" "14:21:45 GMT" "3-Apr-91 GMT" "14:21:45 GMT" jamel * .
h reset
l sc1
l sc2
l sc3
l sc4
cycle 3
l reset
cycle
# traitement de la première colonne
h sc1
sv bus12 00010000
sv bus13 00010000
cycle
sv bus12 01010100
sv bus13 01010100
cycle
l sc1
sv bus12 00110111
sv bus13 00110111
cycle
sv bus12 10000101
sv bus13 10000110
cycle
#chargement des coefficents dans la première colonne
sv bus12 00000001
sv bus13 00000001
c
sv bus12 00000010
sv bus13 00000010
c
```

```
sv bus12 00000011
sv bus13 00000011
c
sv bus12 00000100
sv bus13 00000100
c
sv bus12 00000101
sv bus13 00000101
c
sv bus12 00000110
sv bus13 00000110
c
sv bus12 00000111
sv bus13 00000111
c
sv bus12 00000000
sv bus13 00000000
c
sv bus12 00000001
sv bus13 00000001
c
sv bus12 00000010
sv bus13 00000010
c
sv bus12 00000011
sv bus13 00000011
c
sv bus12 00000100
sv bus13 00000100
c
sv bus12 00000101
sv bus13 00000101
c
sv bus12 00000110
sv bus13 00000110
c
sv bus12 00000111
sv bus13 00000111
```

```
c
sv bus12 00000000
sv bus13 00000000
c
#traitement de la deuxième colonne
h sc2
sv bus12 00010000
sv bus13 00010000
cycle
sv bus12 01010100
sv bus13 01010100
cycle
l sc2
sv bus12 00110111
sv bus13 00110111
cycle
sv bus12 10000001
sv bus13 10000010
cycle
#chargement des coefficents de la deuxième colonne
sv bus12 01111111
sv bus13 00000000
c  16
#traitement de la troisième colonne
h sc3
sv bus12 10010000
sv bus13 10010000
cycle
sv bus12 01010100
sv bus13 01010100
cycle
l sc3
sv bus12 00110111
sv bus13 00110111
cycle
sv bus12 01011001
sv bus13 01011010
cycle
```

```
#chargement des mémoires de la troisième colonne
sv bus12 01111111
sv bus13 11111101
c  16
#traitement de la quatrième colonne
h sc4
sv bus12 10010000
sv bus13 10010000
cycle
sv bus12 01010100
sv bus13 01010100
cycle
l sc4
sv bus12 00110111
sv bus13 00110111
cycle
sv bus12 01010001
sv bus13 01010010
cycle
# chargement des mémoires de la quatrième colonne
sv bus12 10000001
sv bus13 10000001
c
sv bus12 11111110
sv bus13 11111110
c
sv bus12 11111101
sv bus13 11111101
c
sv bus12 11111100
sv bus13 11111100
c
sv bus12 11111011
sv bus13 11111011
c
sv bus12 11111010
sv bus13 11111010
c
```

```
sv bus12 11111001
sv bus13 11111001
c
sv bus12 11111000
sv bus13 11111000
c
sv bus12 11110111
sv bus13 11110111
c
sv bus12 11110110
sv bus13 11110110
c
sv bus12 11110101
sv bus13 11110101
c
sv bus12 11110100
sv bus13 11110100
c
sv bus12 11110011
sv bus13 11110011
c
sv bus12 11110010
sv bus13 11110010
c
sv bus12 11110001
sv bus13 11110001
c
sv bus12 00000000
sv bus13 00000000
c
```

Fichier de valeurs d'entrées

```
#cell2 * stimuli_fin_8_bis1 txt * 2 any 0 v8r1.8
# "5-Apr-91 GMT" "9:17:57 GMT" "5-Apr-91 GMT" "9:17:57 GMT" jamel * .
# chargement de la première valeur  dans C1 et C2

h sc1 sc2
sv bus12 10001000
sv bus13 10001000
c

h sc3 sc4
l sc1 sc2
sv bus12 01111000
sv bus13 01111000
c

l sc3 sc4
c 50
# normalement 47 cycles

h sc1 sc2
sv bus12 00100000
sv bus13 00100000
c

h sc3 sc4
l sc1 sc2
sv bus12 11100000
sv bus13 11100000
c

l sc3 sc4
c 50

h sc1 sc2
sv bus12 00000011
sv bus13 11111101
c
```

```
h sc3 sc4

l sc1 sc2
sv bus12 00000000
sv bus13 00000000
c

l sc3 sc4
c 50

h sc1  sc2
sv bus12 01010101
sv bus13 10101011
c

h sc3 sc4
l sc1 sc2
sv bus12 01100010
sv bus13 01111111
c

l sc3 sc4
c 87

# Lecture des sorties des neurones colonne par colonne
h sc1
c

l sc1
h sc2
c

l sc2
h sc3
c

l sc3
h sc4
c
l sc4
c
```

Résultats de la simulation

Reading command file '[txt]chargement_couche_8_neurones_bis1'

There are 7649 nodes, 2376 gates, 3200 instances, 0 transistors.

Time = 1:1 [0 ns] **Remise à Zéro et iniatialisation du réseau**

Sc1 --> 0 Sc2 --> 0 Sc3 --> 0 Sc4 --> 0 [+0/0]

Time = 1:1 [0 ns]

Beginning circuit preprocessing... Circuit preprocessing complete.

Time = 1:2 [100 ns] **4 cycles de reset**

Clk --> 1 [+0/0]

Time = 5:1 [800 ns] **Sélection de la première colonne et**
Sc1 --> 1 [+0/0] **chargement des registres de personnalisations et des coefficients**
Clk --> 0 [+0/0] **+ ouverture des interrupteurs des bus de la colonne 1**
n11.t12h --> 1 [+12.4/.6] n11.t12l --> 1 [+15.7/.6]
n21.t12h --> 1 [+12.4/.6] n21.t12l --> 1 [+15.7/.6]

Bus12 = 'B00010000 [+0] **Nombre de coefficients à charger + 1**
Bus13 = 'B00010000 [+0] **qui sera chargé dans le compteur C3**

Time = 5:2 [900 ns]

Clk --> 1 [+0/0]

Time = 6:1 [1000 ns]

Les 4 bits forts indiquent le formatage du potentiel
Les 4 faibles bits indiquent le nombre permutation
Bus12 = 'B01010100 [+0] **pour exécuter une vague de données**
Bus13 = 'B01010100 [+0]

Time = 6:2 [1100 ns]

Clk --> 1 [+0/0]
n11.outc3 = 'B010000 [+13.2] n21.outc3 = 'B010000 [+13.2]
n11.R9 = 'B0100 [+17] n21.R9 = 'B0100 [+17]
n11.R5 = 'B0101 [+17] n21.R5 = 'B0101 [+17]

Time = 7:1 [1200 ns]

Scl --> 0 **Les 4 bits forts indiquent le formatage de la 2 ème**
Clk --> 0 **multiplication de la fonction d'activation**

Les 4 bits faibles indiquent le formatage de la 1ème multiplication de la fonction d'activation

Bus12 = 'B00110111 [+0]
Bus13 = 'B00110111 [+0]

Time = 7:2 [1300 ns]

Clk --> 1 [+0/0]
n11.R6 = 'B0111 [+17] n21.R6 = 'B0111 [+17]
n11.R7 = 'B0011 [+17] n21.R7 = 'B0011 [+17]

Time = 8:1 [1400 ns]

Chargement des registres de personnalisation

Clk --> 0 [+0/0]
Bus12 = 'B10000101 [+0]
Bus13 = 'B10000110 [+0]

Time = 8:2 [1500 ns]

Clk --> 1 [+0/0]

Time = 9:1 [1600 ns] **Début de chargement des mémoires**

Clk --> 0 [+0/0]
Bus12 = 'B00000001 [+0]
Bus13 = 'B00000001 [+0]

Time = 9:2 [1700 ns]

Clk --> 1 [+0/0]

Time = 10:1 [1800 ns]

Valeur chargée	**Adresse mémoire**
Bus12 = 'B00000010 [+0]	n11.outc3 = 'B001111 [+9.2]
Bus13 = 'B00000010 [+0]	n21.outc3 = 'B001111 [+9.2]

Time = 10:2 [1900 ns]

Clk --> 1 [+0/0]

Time = 11:1 [2000 ns]

Clk --> 0 [+0/0]

Bus12 = 'B00000011 [+0]	n11.outc3 = 'B001110 [+9.2]
Bus13 = 'B00000011 [+0]	n21.outc3 = 'B001110 [+9.2]

Time = 11:2 [2100 ns]

Clk --> 1 [+0/0]

Time = 12:1 [2200 ns]

Clk --> 0 [+0/0]

Bus12 = 'B00000100 [+0]	n11.outc3 = 'B001101 [+9.2]
Bus13 = 'B00000100 [+0]	n21.outc3 = 'B001101 [+9.2]

Time = 12:2 [2300 ns]

Clk --> 1 [+0/0]

Time = 13:1 [2400 ns]

Clk --> 0 [+0/0]

Bus12 = 'B00000101 [+0]	n11.outc3 = 'B001100 [+9.2]
Bus13 = 'B00000101 [+0]	n21.outc3 = 'B001100 [+9.2]

Time = 13:2 [2500 ns]

Clk --> 1 [+0/0]

```
Time = 14:1   [2600 ns]

     Clk --> 0     [+0/0]

     Bus12 = 'B00000110 [+0]          n11.outc3 = 'B001011 [+9.2]
     Bus13 = 'B00000110 [+0]          n21.outc3 = 'B001011 [+9.2]

Time = 14:2   [2700 ns]

     Clk --> 1     [+0/0]

Time = 15:1   [2800 ns]

     Clk --> 0     [+0/0]
     Bus12 = 'B00000111 [+0]          n11.outc3 = 'B001010 [+9.2]
     Bus13 = 'B00000111 [+0]          n21.outc3 = 'B001010 [+9.2]

Time = 15:2   [2900 ns]

     Clk --> 1     [+0/0]

Time = 16:1   [3000 ns]

     Clk --> 0     [+0/0]
     Bus12 = 'B00000000 [+0]          n11.outc3 = 'B001001 [+9.2]
     Bus13 = 'B00000000 [+0]          n21.outc3 = 'B001001 [+9.2]

Time = 16:2   [3100 ns]

     Clk --> 1     [+0/0]

Time = 17:1   [3200 ns]

     Clk --> 0     [+0/0]
     Bus12 = 'B00000001 [+0]          n11.outc3 = 'B001000 [+9.2]
     Bus13 = 'B00000001 [+0]          n21.outc3 = 'B001000 [+9.2]

Time = 17:2   [3300 ns]

     Clk --> 1     [+0/0]

Time = 18:1   [3400 ns]

     Clk --> 0     [+0/0]
```

```
    Bus12 = 'B00000010 [+0]          n11.outc3 = 'B000111 [+9.2]
    Bus13 = 'B00000010 [+0]          n21.outc3 = 'B000111 [+9.2]

Time = 18:2  [3500 ns]

    Clk --> 1    [+0/0]

Time = 19:1  [3600 ns]

    Clk --> 0    [+0/0]
    Bus12 = 'B00000011 [+0]          n11.outc3 = 'B000110 [+9.2]
    Bus13 = 'B00000011 [+0]          n21.outc3 = 'B000110 [+9.2]

Time = 19:2  [3700 ns]

    Clk --> 1    [+0/0]

Time = 20:1  [3800 ns]

    Clk --> 0    [+0/0]
    Bus12 = 'B00000100 [+0]          n11.outc3 = 'B000101 [+9.2]
    Bus13 = 'B00000100 [+0]          n21.outc3 = 'B000101 [+9.2]

Time = 20:2  [3900 ns]

    Clk --> 1    [+0/0]

Time = 21:1  [4000 ns]

    Clk --> 0    [+0/0]
    Bus12 = 'B00000101 [+0]          n11.outc3 = 'B000100 [+9.2]
    Bus13 = 'B00000101 [+0]          n21.outc3 = 'B000100 [+9.2]

Time = 21:2  [4100 ns]

    Clk --> 1    [+0/0]

Time = 22:1  [4200 ns]

    Clk --> 0    [+0/0]
    Bus12 = 'B00000110 [+0]          n11.outc3 = 'B000011 [+9.2]
    Bus13 = 'B00000110 [+0]          n21.outc3 = 'B000011 [+9.2]

Time = 22:2  [4300 ns]
```

Clk --> 1 [+0/0]

Time = 23:1 [4400 ns]

Clk --> 0 [+0/0]
Bus12 = 'B00000111 [+0] n11.outc3 = 'B000010 [+9.2]
Bus13 = 'B00000111 [+0] n21.outc3 = 'B000010 [+9.2]

Time = 23:2 [4500 ns]

Clk --> 1 [+0/0]

Time = 24:1 [4600 ns]

Dernière valeur chargée **Dernière adresse**
Bus12 = 'B00000000 [+0] n11.outc3 = 'B000001 [+9.2]
Bus13 = 'B00000000 [+0] n21.outc3 = 'B000001 [+9.2]

Time = 24:2 [4700 ns]

Clk --> 1 [+0/0]

Time = 25:1 [4800 ns]

Sélection de la 2 ème colonne + ouverture des interrupteurs de la colonne 2

Sc2 --> 1 [+0/0] Clk --> 0 [+0/0]
n22.t12h --> 1 [+12.4/.6] n22.t12l --> 1 [+15.7/.6]
n12.t12h --> 1 [+12.4/.6] n12.t12l --> 1 [+15.7/.6]
Bus12 = 'B00010000 [+0]
Bus13 = 'B00010000 [+0]
n11.outc3 = 'B000000 [+9.2]
n21.outc3 = 'B000000 [+9.2]

Time = 25:2 [4900 ns]

Chargement des registres de personnalisation et des mémoires des neurones n21 et n22.

Time = 45:1 [8800 ns]

Sélection de la 3 ème colonne + ouverture des interrupteurs de la colonne 3

Sc3 --> 1 [+0/0] Clk --> 0 [+0/0]
n13.t12h --> 1 [+12.4/.6] n13.t12l --> 1 [+15.7/.6]
n23.t12h --> 1 [+12.4/.6] n23.t12l --> 1 [+15.7/.6]
Bus12 = 'B10010000 [+0]
Bus13 = 'B10010000 [+0]
n12.outc3 = 'B000000 [+9.2]
n22.outc3 = 'B000000 [+9.2]

Time = 45:2 [8900 ns]

Chargement des registres de personnalisation et des mémoires des neurones n31 et n32.

Time = 65:1 [12800 ns]

Sélection de la 4 ème colonne + ouverture des interrupteurs de la colonne 4

Clk --> 0 [+0/0] Sc4 --> 1 [+0/0]
n14.t12h --> 1 [+12.4/.6] n14.t12l --> 1 [+15.7/.6]
n24.t12h --> 1 [+12.4/.6] n24.t12l --> 1 [+15.7/.6]
Bus12 = 'B10010000 [+0] Bus22 = 'B10010000 [+23.8]
Bus13 = 'B10010000 [+0] Bus23 = 'B10010000 [+25.4]
n13.outc3 = 'B000000 [+9.2] n23.outc3 = 'B000000 [+9.2]

Time = 65:2 [12900 ns]

Chargement des registres de personnalisation et des mémoires des neurones n21 et n22.

Reading command file '[txt]stimuli_fin_8_bis1'

Time = 85:1 [16800 ns]

Sc1 --> 1 [+0/0] **Sélection des 2 premières colonnes et**
Sc2 --> 1 [+0/0] **chargement des premières valeurs d'entrées**
Clk --> 0 [+0/0] **par les bus d'entrée bus12 et 13**
Bus12 = 'B10001000 [+0] Bus22 = 'B10001000 [+20.4]
Bus13 = 'B10001000 [+0] Bus23 = 'B10001000 [+21.3]

Time = 85:2 [16900 ns]

Initialisation du compteur d'adresse C3 et de R2 et du compteur de la multiplication C1. Chargement des entrées dans R3

```
Clk --> 1    [+0/0]
n11.comtc1 = 'B000 [+7.7]          n11.RR2 = 'B00000000 [+10.3]
n11.outc3 = 'B010000 [+13.2]       n11.RR3 = 'B10001000 [+11.8]
n21.comtc1 = 'B000 [+7.7]          n21.RR2 = 'B00000000 [+10.3]
n21.outc3 = 'B010000 [+13.2]       n21.RR3 = 'B10001000 [+11.8]
n12.comtc1 = 'B000 [+7.7]          n12.RR2 = 'B00000000 [+10.3]
n12.outc3 = 'B010000 [+13.2]       n12.RR3 = 'B10001000 [+11.8]
n22.comtc1 = 'B000 [+7.7]          n22.RR2 = 'B00000000 [+10.3]
n22.outc3 = 'B010000 [+13.2]       n22.RR3 = 'B10001000 [+11.8]
n14.comtc2 = 'B1110 [+11.9]
n24.comtc2 = 'B1110 [+11.9]
```

Time = 86:1 [17000 ns]

Sélection des 2 dernières colonnes et chargement des premières valeurs d'entrées par les bus d'entrée bus12 et 13

```
Sc3 --> 1   [+0/0]    Sc1 --> 0   [+0/0]    Clk --> 0   [+0/0]
Sc4 --> 1   [+0/0]    Sc2 --> 0   [+0/0]
Bus12 = 'B01111000 [+0]   Bus22 = 'B01111000 [+20.4]
Bus13 = 'B01111000 [+0]   Bus23 = 'B01111000 [+21.3]
```

Time = 86:2 [17100 ns]

Les compteurs C2 sont chargés par des valeurs qui temporisent un début de calcul simultané de tous les neurones

```
Clk --> 1    [+0/0]
n11.comtc2 = 'B1110 [+11.1]     n12.comtc2 = 'B1110 [+11.1]
n21.comtc2 = 'B1110 [+11.1]     n22.comtc2 = 'B1110 [+11.1]
n13.comtc1 = 'B000 [+7.7] n13.RR2 = 'B00000000 [+10.3]
n13.outc3 = 'B010000 [+13.2]    n13.RR3 = 'B01111000 [+11.8]
n23.comtc1 = 'B000 [+7.7] n23.RR2 = 'B00000000 [+10.3]
n23.outc3 = 'B010000 [+13.2]    n23.RR3 = 'B01111000 [+11.8]
n14.comtc1 = 'B000 [+7.7] n14.RR2 = 'B00000000 [+10.3]
n14.outc3 = 'B010000 [+13.2]    n14.RR3 = 'B01111000 [+11.8]
n24.comtc1 = 'B000 [+7.7] n24.RR2 = 'B00000000 [+10.3]
n24.outc3 = 'B010000 [+13.2]    n24.RR3 = 'B01111000 [+11.8]
```

Time = 87:1 [17200 ns]

Sc3 --> 0 [+0/0] Sc4 --> 0 [+0/0] Clk --> 0 [+0/0]

Time = 87:2 [17300 ns]

Clk --> 1 [+0/0] **Attente**

Time = 88:1 [17400 ns]

Clk --> 0 [+0/0]

Time = 88:2 [17500 ns]

Clk --> 1 [+0/0]

Time = 89:1 [17600 ns] **Début des multiplications**

Clk --> 0 [+0/0]

Time = 89:2 [17700 ns]

Clk --> 1 [+0/0]

Etape de multiplication	**Etat des registres R2 et R3**
n11.comtc1 = 'B001 [+7.4]	n11.RR3 = 'B01000100 [+11.8]
n21.comtc1 = 'B001 [+7.4]	n21.RR3 = 'B01000100 [+11.8]
n12.comtc1 = 'B001 [+7.4]	n12.RR3 = 'B01000100 [+11.8]
n22.comtc1 = 'B001 [+7.4]	n22.RR3 = 'B01000100 [+11.8]
n13.comtc1 = 'B001 [+7.4]	n13.RR3 = 'B00111100 [+11.8]
n23.comtc1 = 'B001 [+7.4]	n23.RR3 = 'B00111100 [+11.8]
n14.comtc1 = 'B001 [+7.4]	n14.RR3 = 'B00111100 [+11.8]
n24.comtc1 = 'B001 [+7.4]	n24.RR3 = 'B00111100 [+11.8]

Time = 90:1 [17800 ns]

Clk --> 0 [+0/0]

Time = 90:2 [17900 ns]

Clk --> 1 [+0/0]

n11.comtc1 = 'B010 [+8.7]	n11.RR3 = 'B00100010 [+11.8]
n12.comtc1 = 'B010 [+8.7]	n12.RR3 = 'B00100010 [+11.8]
n22.comtc1 = 'B010 [+8.7]	n22.RR3 = 'B00100010 [+11.8]

```
    n13.comtc1 = 'B010 [+8.7]      n13.RR3 = 'B00011110 [+11.8]
    n23.comtc1 = 'B010 [+8.7]      n23.RR3 = 'B00011110 [+11.8]
    n14.comtc1 = 'B010 [+8.7]      n14.RR3 = 'B00011110 [+11.8]
    n24.comtc1 = 'B010 [+8.7]      n24.RR3 = 'B00011110 [+11.8]

Time = 91:1  [18000 ns]

    Clk --> 0    [+0/0]

Time = 91:2  [18100 ns]

    Clk --> 1    [+0/0]
    n11.comtc1 = 'B011 [+7.4]      n11.RR3 = 'B00010001 [+12.8]
    n21.comtc1 = 'B011 [+7.4]      n21.RR3 = 'B00010001 [+12.8]
    n12.comtc1 = 'B011 [+7.4]      n12.RR3 = 'B00010001 [+12.8]
    n22.comtc1 = 'B011 [+7.4]      n22.RR3 = 'B00010001 [+12.8]
    n13.comtc1 = 'B011 [+7.4]      n13.RR3 = 'B00001111 [+12.8]
    n23.comtc1 = 'B011 [+7.4]      n23.RR3 = 'B00001111 [+12.8]
    n14.comtc1 = 'B011 [+7.4]      n14.RR3 = 'B00001111 [+12.8]
    n24.comtc1 = 'B011 [+7.4]      n24.RR3 = 'B00001111 [+12.8]

Time = 92:1  [18200 ns]

    Clk --> 0    [+0/0]

Time = 92:2  [18300 ns]

    Clk --> 1    [+0/0]
    n11.comtc1 = 'B100 [+8.7]      n11.RR2 = 'B11111111 [+11.3]
                                   n11.RR3 = 'B10001000 [+11.8]
    n21.comtc1 = 'B100 [+8.7]      n21.RR2 = 'B11111111 [+11.3]
                                   n21.RR3 = 'B10001000 [+11.8]
    n12.comtc1 = 'B100 [+8.7]      n12.RR2 = 'B11000000 [+11]
                                   n12.RR3 = 'B10001000 [+11.8]
    n22.comtc1 = 'B100 [+8.7]      n22.RR3 = 'B00001000 [+11.8]
    n13.comtc1 = 'B100 [+8.7]      n13.RR2 = 'B11000000 [+11]
                                   n13.RR3 = 'B10000111 [+11.8]
    n23.comtc1 = 'B100 [+8.7]      n23.RR2 = 'B00000001 [+11.3]
                                   n23.RR3 = 'B10000111 [+11.8]
    n14.comtc1 = 'B100 [+8.7]      n14.RR2 = 'B00111111 [+11.3]
```

```
                                        n14.RR3 = 'B10000111 [+11.8]
    n24.comtc1 = 'B100 [+8.7]           n24.RR2 = 'B00111111 [+11.3]
                                        n24.RR3 = 'B10000111 [+11.8]

Time = 93:1  [18400 ns]

    Clk --> 0    [+0/0]

Time = 93:2  [18500 ns]

    Clk --> 1    [+0/0]
    n11.comtc1 = 'B101 [+7.4]           n11.RR2 = 'B00000000 [+10.3]
                                        n11.RR3 = 'B01000100 [+11.8]
    n21.comtc1 = 'B101 [+7.4]           n21.RR2 = 'B00000000 [+10.3]
                                        n21.RR3 = 'B01000100 [+11.8]
    n12.comtc1 = 'B101 [+7.4]           n12.RR2 = 'B00011111 [+11.3]
                                        n12.RR3 = 'B11000100 [+11.8]
    n22.comtc1 = 'B101 [+7.4]           n22.RR3 = 'B00000100 [+11.8]
    n13.comtc1 = 'B101 [+7.4]           n13.RR2 = 'B11100000 [+10.4]
                                        n13.RR3 = 'B01000011 [+11.8]
    n23.comtc1 = 'B101 [+7.4]           n23.RR2 = 'B00000000 [+10.3]
                                        n23.RR3 = 'B11000011 [+11.8]
    n14.comtc1 = 'B101 [+7.4]           n14.RR2 = 'B00011111 [+9.6]
                                        n14.RR3 = 'B11000011 [+11.8]
    n24.comtc1 = 'B101 [+7.4]           n24.RR2 = 'B00011111 [+9.6]
                                        n24.RR3 = 'B11000011 [+11.8]

Time = 94:1  [18600 ns]

    Clk --> 0    [+0/0]

Time = 94:2  [18700 ns]

    Clk --> 1    [+0/0]
    n11.comtc1 = 'B110 [+8.7]           n11.RR3 = 'B00100010 [+11.8]
    n21.comtc1 = 'B110 [+8.7]           n21.RR3 = 'B00100010 [+11.8]
    n12.comtc1 = 'B110 [+8.7]           n12.RR2 = 'B00001111 [+9.6]
                                        n12.RR3 = 'B11100010 [+11.8]
    n22.comtc1 = 'B110 [+8.7]           n22.RR3 = 'B00000010 [+11.8]
    n13.comtc1 = 'B110 [+8.7]           n13.RR2 = 'B11110000 [+10.4]
                                        n13.RR3 = 'B00100001 [+11.8]
```

```
n23.comtc1 = 'B110 [+8.7]        n23.RR3 = 'B01100001 [+11.8]
n14.comtc1 = 'B110 [+8.7]        n14.RR2 = 'B00001111 [+9.6]
                                 n14.RR3 = 'B11100001 [+11.8]
n24.comtc1 = 'B110 [+8.7]        n24.RR2 = 'B00001111 [+9.6]
                                 n24.RR3 = 'B11100001 [+11.8]

Time = 95:1  [18800 ns]

    Clk --> 0    [+0/0]

Time = 95:2  [18900 ns]

    Clk --> 1    [+0/0]
    n11.comtc1 = 'B111 [+7.4]    n11.RR3 = 'B00010001 [+12.8]
    n21.comtc1 = 'B111 [+7.4]    n21.RR3 = 'B00010001 [+12.8]
    n12.comtc1 = 'B111 [+7.4]    n12.RR2 = 'B00000111 [+9.6]
                                 n12.RR3 = 'B11110001 [+12.8]
    n22.comtc1 = 'B111 [+7.4]    n22.RR3 = 'B00000001 [+12.8]
    n13.comtc1 = 'B111 [+7.4]    n13.RR2 = 'B11111000 [+10.4]
                                 n13.RR3 = 'B00010000 [+11.8]
    n23.comtc1 = 'B111 [+7.4]    n23.RR3 = 'B00110000 [+11.8]
    n14.comtc1 = 'B111 [+7.4]    n14.RR2 = 'B00000111 [+9.6]
                                 n14.RR3 = 'B11110000 [+11.8]
    n24.comtc1 = 'B111 [+7.4]    n24.RR2 = 'B00000111 [+9.6]
                                 n24.RR3 = 'B11110000 [+11.8]

Time = 96:1  [19000 ns]          Dernier cycle de la multiplication

    Clk --> 0    [+0/0]

Time = 96:2  [19100 ns]

    Clk --> 1    [+0/0]
    n11.comtc1 = 'B000 [+8.7]    n11.RR2 = 'B11111111 [+11.3]
                                 n11.RR3 = 'B10001000 [+11.8]
    n21.comtc1 = 'B000 [+8.7]    n21.RR2 = 'B11111111 [+11.3]
                                 n21.RR3 = 'B10001000 [+11.8]
    n12.comtc1 = 'B000 [+8.7]    n12.RR2 = 'B11000100 [+11]
                                 n12.RR3 = 'B01111000 [+11.8]
    n22.comtc1 = 'B000 [+8.7]    n22.RR3 = 'B00000000 [+11.6]
    n13.comtc1 = 'B000 [+8.7]    n13.RR2 = 'B00111011 [+11.3]
```

```
                                        n13.RR3 = 'B10001000 [+11.8]
    n23.comtc1 = 'B000 [+8.7]           n23.RR2 = 'B11111110 [+11]
                                        n23.RR3 = 'B10011000 [+11.8]
    n14.comtc1 = 'B000 [+8.7]           n14.RR2 = 'B11000100 [+11]
                                        n14.RR3 = 'B01111000 [+11.8]
    n24.comtc1 = 'B000 [+8.7]           n24.RR2 = 'B11000100 [+11]
                                        n24.RR3 = 'B01111000 [+11.8]

Time = 97:1  [19200 ns]

    Clk --> 0   [+0/0]      Test de branchement

Time = 97:2  [19300 ns]

    Clk --> 1   [+0/0]

Time = 98:1  [19400 ns]     Cumul des poids faibles

    Clk --> 0   [+0/0]

Time = 98:2  [19500 ns]

    Clk --> 1   [+0/0]
    n11.accftR = 'B11111111 [+11.8] n11.accfbR = 'B10001000 [+12]
    n12.accftR = 'B11000100 [+11.8] n12.accfbR = 'B01111000 [+11]
    n13.accftR = 'B00111011 [+10.9] n13.accfbR = 'B10001000 [+12]
    n14.accftR = 'B11000100 [+11.8] n14.accfbR = 'B01111000 [+11]
    n21.accftR = 'B11111111 [+11.8] n21.accfbR = 'B10001000 [+12]
    n22.accftR = 'B00000000 [5510.7]
    n23.accftR = 'B11111110 [+11.8] n23.accfbR = 'B10011000 [+12]
    n24.accftR = 'B11000100 [+11.8] n24.accfbR = 'B01111000 [+11]

Time = 99:1  [19600 ns]                 Cumul des poids forts

    Clk --> 0   [+0/0]

Time = 99:2  [19700 ns]

    Clk --> 1   [+0/0]

Time = 100:1  [19800 ns]
        Décalage des données + décrémentation de C3
```

```
Clk --> 0     [+0/0]
n23.t12l --> 1     [+10.1/.6]          n11.buflow --> 1     [+10.1/.6]
n12.t21h --> 1     [+10.1/.6]          n12.buflow --> 1     [+10.1/.6]
n13.t21h --> 1     [+10.1/.6]          n13.bufhigh --> 1    [+10.1/.6]
n14.t21h --> 1     [+10.1/.6]          n14.bufhigh --> 1    [+10.1/.6]
n21.t21h --> 1     [+10.1/.6]          n21.buflow --> 1     [+10.1/.6]
n22.t12l --> 1     [+10.1/.6]          n22.buflow --> 1     [+10.1/.6]
n23.t12l --> 1     [+10.1/.6]          n23.bufhigh --> 1    [+10.1/.6]
n24.t12l --> 1     [+10.1/.6]          n24.bufhigh --> 1    [+10.1/.6]

              Bus21 = 'B01111000 [+18.9]
              Bus23 = 'B10001000 [+31.4]

n11.outc3 = 'B001111 [+9.2]       n21.outc3 = 'B001111 [+9.2]
n12.outc3 = 'B001111 [+9.2]       n22.outc3 = 'B001111 [+9.2]
n13.outc3 = 'B001111 [+9.2]       n23.outc3 = 'B001111 [+9.2]
n14.outc3 = 'B001111 [+9.2]       n24.outc3 = 'B001111 [+9.2]

Time = 100:2  [19900 ns]

Clk --> 1     [+0/0]                          Bus22 = 'B10001000 [+14.5]

Initialisation de R2          +  Chargt des valeurs décalées dans R3

n11.RR2 = 'B00000000 [+10.3]   n11.RR3 = 'B01111000 [+11.8]
n21.RR2 = 'B00000000 [+10.3]
n12.RR2 = 'B00000000 [+10.1]
                                n22.RR3 = 'B10001000 [+11.8]
n13.RR2 = 'B00000000 [+10.3]   n13.RR3 = 'B01111000 [+11.8]
n23.RR2 = 'B00000000 [+10.1]   n23.RR3 = 'B10001000 [+10.8]
n14.RR2 = 'B00000000 [+10.1]
n24.RR2 = 'B00000000 [+10.1]   n24.RR3 = 'B10001000 [+11.8]

Time = 101:1  [20000 ns]

Clk --> 0     [+0/0]

Multiplication + Cumul partiel du potentiel

Time = 112:1  [22200 ns]          Décalage + Décrémentation de C3
    n23.t12l --> 1     [+10.1/.6]          n23.bufhigh --> 1    [+10.1/.6]
    n12.t21h --> 1     [+10.1/.6]          n12.buflow --> 1     [+10.1/.6]
```

```
n14.t21h --> 1    [+10.1/.6]        n14.bufhigh --> 1    [+10.1/.6]
n24.t12l --> 1    [+10.1/.6]        n24.bufhigh --> 1    [+10.1/.6]
                                    n11.buflow --> 1     [+10.1/.6]
                                    n21.buflow --> 1     [+10.1/.6]
n13.t21h --> 1    [+10.1/.6]        n13.bufhigh --> 1    [+10.1/.6]
n22.t12l --> 1    [+10.1/.6]        n22.buflow --> 1     [+10.1/.6]
n24.outc3, n14.outc3, n23.outc3, n13.outc3, n22.outc3, n12.outc3, n21.outc3,
n11.outc3 = 'B001110 [+9.2]

Time = 112:2  [22300 ns]

Clk --> 1    [+0/0]

Bus21 = 'B10001000 [+13.2]
Bus23 = 'B01111000 [+26.1]          Chargt des valeurs décalées

Initialisation des R2               n11.RR3 = 'B01111000 [+11.8]
n21.RR2 = 'B00000000 [+10.3]        n21.RR3 = 'B01111000 [+11.8]
n12.RR2 = 'B00000000 [+10.3]        n12.RR3 = 'B01111000 [+11.8]
                                    n22.RR3 = 'B01111000 [+11.8]
n13.RR2 = 'B00000000 [+10.3]
n23.RR2 = 'B00000000 [+10.3]        n23.RR3 = 'B10001000 [+11.8]
n14.RR2 = 'B00000000 [+10.3]        n14.RR3 = 'B10001000 [+11.8]
                                    n24.RR3 = 'B10001000 [+11.8]

Time = 113:1  [22400 ns]

Clk --> 0    [+0/0]

Multiplication + Cumul partiel du potentiel

Time = 124:1  [24600 ns]    Décalage + Décrémentation de C3

Clk --> 0    [+0/0]
n23.t12l --> 1    [+10.1/.6]        n23.bufhigh --> 1    [+10.1/.6]
n12.t21h --> 1    [+10.1/.6]        n12.buflow --> 1     [+10.1/.6]
n14.t21h --> 1    [+10.1/.6]        n14.bufhigh --> 1    [+10.1/.6]
n24.t12l --> 1    [+10.1/.6]        n24.bufhigh --> 1    [+10.1/.6]
                                    n11.buflow --> 1     [+10.1/.6]
                                    n21.buflow --> 1     [+10.1/.6]
n13.t21h --> 1    [+10.1/.6]        n13.bufhigh --> 1    [+10.1/.6]
```

n22.t12l --> 1 [+10.1/.6] n22.buflow --> 1 [+10.1/.6]

n24.outc3, n14.outc3, n23.outc3, n13.outc3, n22.outc3, n12.outc3, n21.outc3, n11.outc3 = 'B001101 [+9.2]

Time = 124:2 [24700 ns]

Clk --> 1 [+0/0] Bus22 = 'B01111000 [+14.5]

Initialisation des R2

n11.RR2 = 'B00000000 [+10.3]
n21.RR2 = 'B00000000 [+10.3]
n12.RR2 = 'B00000000 [+10.3]
n13.RR2 = 'B00000000 [+10.1]
n23.RR2 = 'B00000000 [+10.3]
n14.RR2 = 'B00000000 [+10.3]
n24.RR2 = 'B00000000 [+10.3]

Chargt des valeurs décalées

n11.RR3 = 'B10001000 [+11.8]
n21.RR3 = 'B01111000 [+11.8]
n22.RR3 = 'B01111000 [+11.8]
n13.RR3 = 'B10001000 [+11.8]
n23.RR3 = 'B01111000 [+11.8]
n14.RR3 = 'B10001000 [+11.8]
n24.RR3 = 'B01111000 [+11.8]

Time = 125:1 [24800 ns]

Clk --> 0 [+0/0]

Multiplication + Cumul du potentiel --> Fin du traitement de la première vague

Time = 136:1 [27000 ns] **Chargement de la deuxième Vague**

Clk --> 0 [+0/0]

n23.t12h --> 1	[+10.1/.6]	n23.t12l --> 1	[+10.1/.6]
n12.t12h --> 1	[+10.1/.6]	n12.t12l --> 1	[+10.1/.6]
n14.t12h --> 1	[+10.1/.6]	n14.t12l --> 1	[+10.1/.6]
n24.t12h --> 1	[+10.1/.6]	n24.t12l --> 1	[+10.1/.6]
n11.t12h --> 1	[+10.1/.6]	n11.t12l --> 1	[+10.1/.6]
n21.t12h --> 1	[+10.1/.6]	n21.t12l --> 1	[+10.1/.6]
n13.t12h --> 1	[+10.1/.6]	n13.t12l --> 1	[+10.1/.6]
n22.t12h --> 1	[+10.1/.6]	n22.t12l --> 1	[+10.1/.6]

Bus21 = 'Buuuuuuuu [+35]
Bus22 = 'B01111000 [+36]

n24.outc3, n14.outc3, n23.outc3, n13.outc3, n22.outc3, n12.outc3, n21.outc3, n11.outc3 = 'B001100 [+9.2]

Time = 136:2 [27100 ns]

Clk --> 1 [+0/0]

Time = 137:1 [27200 ns]

Sc1 --> 1 [+0/0] Sc2 --> 1 [+0/0]

Sélection et Chargement des 2 premières colonnes

Bus12 = 'B00100000 [+0] Bus22 = 'B00100000 [+15.2]
Bus13 = 'B00100000 [+0] Bus23 = 'B00100000 [+15.8]

Time = 137:2 [27300 ns]

Clk --> 1 [+0/0]
n11.RR2 = 'B00000000 [+10.1]
n21.RR2 = 'B00000000 [+10.3] n21.RR3 = 'B00100000 [+10.8]
n12.RR2 = 'B00000000 [+10.1] n12.RR3 = 'B00100000 [+10.8]
n22.RR3 = 'B00100000 [+11.8]

Time = 138:1 [27400 ns]

Sc3 --> 1 Sc4 --> 1 Sc1 --> 0 Sc2 --> 0 [+0/0]

Sélection et Chargement des 2 dernières colonnes

Bus12 = 'B11100000 [+0] Bus22 = 'B11100000 [+20.1]
Bus13 = 'B11100000 [+0] Bus23 = 'B11100000 [+20.7]

Time = 138:2 [27500 ns]

Clk --> 1 [+0/0]
n22.comtc2, n12.comtc2, n21.comtc2, n11.comtc2 = 'B1110 [+11.1]
n13.RR2 = 'B00000000 [+10.1] n13.RR3 = 'B11100000 [+11.8]
n23.RR2 = 'B00000000 [+10.1] n23.RR3 = 'B11100000 [+11.8]
n14.RR2 = 'B00000000 [+10.3]
n24.RR2 = 'B00000000 [+10.1] n24.RR3 = 'B11100000 [+11.8]

Time = 139:1 [27600 ns]

Sc3 --> 0 [+0/0] Sc4 --> 0 [+0/0]

Time = 139:2 [27700 ns]

Clk --> 1 [+0/0] **Attente**

n24.comtc2, n14.comtc2, n23.comtc2, n13.comtc2, n22.comtc2, n12.comtc2, n21.comtc2, n11.comtc2 = 'B1111 [+11]

Time = 140:1 [27800 ns]

Clk --> 0 [+0/0]

Time = 140:2 [27900 ns]

Clk --> 1 [+0/0]

n24.comtc2, n14.comtc2, n23.comtc2, n13.comtc2, n22.comtc2, n12.comtc2, n21.comtc2, n11.comtc2 = 'B1011 [+9.2]

Time = 141:1 [28000 ns] **Début de traitement de la 2 ème vague**

Clk --> 0 [+0/0]

Time = 141:2 [28100 ns]

Clk --> 1 [+0/0]

Time = 188:1 [37400 ns] **Fin de traitement de la 2 ème vague**

Clk --> 0 [+0/0]

n23.t12h --> 1	[+10.1/.6]	n23.t12l --> 1	[+10.1/.6]
n12.t12h --> 1	[+10.1/.6]	n12.t12l --> 1	[+10.1/.6]
n14.t12h --> 1	[+10.1/.6]	n14.t12l --> 1	[+10.1/.6]
n24.t12h --> 1	[+10.1/.6]	n24.t12l --> 1	[+10.1/.6]
n11.t12h --> 1	[+10.1/.6]	n11.t12l --> 1	[+10.1/.6]
n21.t12h --> 1	[+10.1/.6]	n21.t12l --> 1	[+10.1/.6]
n13.t12h --> 1	[+10.1/.6]	n13.t12l --> 1	[+10.1/.6]
n22.t12h --> 1	[+10.1/.6]	n22.t12l --> 1	[+10.1/.6]

Bus21 = 'Buuuuuuuu [+35]
Bus22 = 'B11100000 [+35.7]

n24.outc3, n14.outc3, n23.outc3, n13.outc3, n22.outc3, n12.outc3, n21.outc3, n11.outc3 = 'B001000 [+9.2]

Time = 188:2 [37500 ns]

Clk --> 1 [+0/0]

n24.comtc2, n14.comtc2, n23.comtc2, n13.comtc2, n22.comtc2, n12.comtc2, n21.comtc2, n11.comtc2 = 'B1101 [+9.2]

Time = 189:1 [37600 ns]

Sélection et Chargement des 2 premières colonnes

Sc1 --> 1 [+0/0] Sc2 --> 1 [+0/0]
Bus12 = 'B00000011 [+0] Bus22 = 'B00000011 [+20.4]
Bus13 = 'B11111101 [+0] Bus23 = 'B11111101 [+21.3]

Time = 189:2 [37700 ns]

Clk --> 1 [+0/0]
n11.RR3 = 'B00000011 [+12.8] n21.RR3 = 'B11111101 [+12.8]
n12.RR3 = 'B00000011 [+12.8] n22.RR3 = 'B11111101 [+12.8]

Time = 190:1 [37800 ns]

Sélection et Chargement des 2 dernières colonnes

Sc3 --> 1 [+0/0] Sc1 --> 0 [+0/0]
Sc4 --> 1 [+0/0] Sc2 --> 0 [+0/0]
Clk --> 0 [+0/0]
Bus12 = 'B00000000 [+0] Bus22 = 'B00000000 [+15.2]
Bus13 = 'B00000000 [+0] Bus23 = 'B00000000 [+15.8]

Time = 190:2 [37900 ns]

Clk --> 1 [+0/0]
n22.comtc2, n12.comtc2, n21.comtc2, n11.comtc2 = 'B1110 [+11.1]
n13.RR3 = 'B00000000 [+10.8]
n23.RR3 = 'B00000000 [+10.8]
n14.RR3 = 'B00000000 [+10.8]
n24.RR3 = 'B00000000 [+10.8]

Time = 191:1 [38000 ns]

Sc3 --> 0 Sc4 --> 0 Clk --> 0 [+0/0]

Time = 191:2 [38100 ns]

Clk --> 1 [+0/0]

n24.comtc2, n14.comtc2, n23.comtc2, n13.comtc2, n22.comtc2, n12.comtc2, n21.comtc2, n11.comtc2 = 'B1111 [+11]

Time = 192:1 [38200 ns]

Clk --> 0 [+0/0]

Time = 192:2 [38300 ns]

Clk --> 1 [+0/0]

Traitement de 3 ème vague

Time = 240:1 [47800 ns]

Clk --> 0 [+0/0]

n23.t12h --> 1	[+10.1/.6]	n23.t12l --> 1	[+10.1/.6]
n12.t12h --> 1	[+10.1/.6]	n12.t12l --> 1	[+10.1/.6]
n14.t12h --> 1	[+10.1/.6]	n14.t12l --> 1	[+10.1/.6]
n24.t12h --> 1	[+10.1/.6]	n24.t12l --> 1	[+10.1/.6]
n11.t12h --> 1	[+10.1/.6]	n11.t12l --> 1	[+10.1/.6]
n21.t12h --> 1	[+10.1/.6]	n21.t12l --> 1	[+10.1/.6]
n13.t12h --> 1	[+10.1/.6]	n13.t12l --> 1	[+10.1/.6]
n22.t12h --> 1	[+10.1/.6]	n22.t12l --> 1	[+10.1/.6]

Bus21 = 'Buuuuuuuu [+35]
Bus22 = 'B00000000 [+31.2]

n24.outc3, n14.outc3, n23.outc3, n13.outc3, n22.outc3, n12.outc3, n21.outc3, n11.outc3 = 'B000100 [+9.2]

Time = 240:2 [47900 ns]

Clk --> 1 [+0/0]

n22.comtc2, n12.comtc2, n21.comtc2, n11.comtc2 = 'B1101 [+9.2]
n24.comtc2, n14.comtc2, n23.comtc2, n13.comtc2 = 'B1110 [+9.6]

Time = 241:1 [48000 ns]

Sélection et Chargement des 2 premières colonnes

Sc1 --> 1 Sc2 --> 1 Clk --> 0 [+0/0]
Bus12 = 'B01010101 [+0] Bus22 = 'B01010101 [+20.4]
Bus13 = 'B10101011 [+0] Bus23 = 'B10101011 [+21.3]

Time = 241:2 [48100 ns]

Clk --> 1 [+0/0]
n11.RR2 = 'B00000000 [+10.3] n11.RR3 = 'B01010101 [+12.8]
n21.RR3 = 'B10101011 [+12.8]
n12.RR2 = 'B00000000 [+10.1] n12.RR3 = 'B01010101 [+11.8]
n22.RR3 = 'B10101011 [+12.8]

Time = 242:1 [48200 ns]

Sélection et Chargement des 2 dernières colonnes

Sc3 --> 1 Sc1 --> 0 Sc4 --> 1 Sc2 --> 0 Clk --> 0 [+0/0]
Bus12 = 'B01100010 [+0] Bus22 = 'B01100010 [+20.4]
Bus13 = 'B01111111 [+0] Bus23 = 'B01111111 [+21.3]

Time = 242:2 [48300 ns]

Clk --> 1 [+0/0]

n22.comtc2, n12.comtc2, n21.comtc2, n11.comtc2 = 'B1110 [+11.1]
n13.RR2 = 'B00000000 [+10.3] n13.RR3 = 'B01100010 [+11.8]
n23.RR3 = 'B01111111 [+12.8]
n14.RR2 = 'B00000000 [+10.3] n14.RR3 = 'B01100010 [+11.8]
n24.RR3 = 'B01111111 [+12.8]

Time = 243:1 [48400 ns]

Sc3 --> 0 Sc4 --> 0 Clk --> 0 [+0/0]

Time = 243:2 [48500 ns] **Attente**

Clk --> 1 [+0/0]

n24.comtc2, n14.comtc2, n23.comtc2, n13.comtc2, n22.comtc2, n12.comtc2,
n21.comtc2, n11.comtc2 = 'B1111 [+11]

Time = 244:1 [48600 ns]

Clk --> 0 [+0/0]

Time = 244:2 [48700 ns]

Clk --> 1 [+0/0]

n24.comtc2, n14.comtc2, n23.comtc2, n13.comtc2, n22.comtc2, n12.comtc2, n21.comtc2, n11.comtc2 = 'B1011 [+9.2]

Time = 245:1 [48800 ns]

Traitement de la 4 ème vague

Time = 292:1 [58200 ns]

Clk --> 0 [+0/0]

Time = 292:2 [58300 ns]

Clk --> 1 [+0/0]

n11.comtc2 = 'B1010 [+9.6]	n11.RR3 = 'B01111111 [+12.8]
n21.comtc2 = 'B1010 [+9.6]	n21.RR3 = 'B01111111 [+12.8]
n12.comtc2 = 'B1010 [+9.6]	n12.RR2 = 'B00000000 [+10.3]
	n12.RR3 = 'B01111111 [+11.8]
n22.comtc2 = 'B1010 [+9.6]	n22.RR3 = 'B01111111 [+12.8]
n13.comtc2 = 'B1010 [+9.6]	n13.RR2 = 'B00000000 [+9.6]
	n13.RR3 = 'B01111111 [+11.8]
n23.comtc2 = 'B1010 [+9.6]	n23.RR2 = 'B00000000 [+10.1]
	n23.RR3 = 'B01111111 [+12.8]
n14.comtc2 = 'B1010 [+9.6]	n14.RR3 = 'B01111111 [+12.8]
n24.comtc2 = 'B1010 [+9.6]	n24.RR3 = 'B01111111 [+12.8]

Time = 293:1 [58400 ns]

Formatage du potentiel dans les accumulateurs des poids forts et faibles

n11.accftR = 'B00000101 [55710.2]	n11.accfbR = 'B11011111 [+11]
n12.accftR = 'B01101100 [+10.7]	n12.accfbR = 'B00100110 [+11]
n13.accftR = 'B01101100 [+10.2]	n13.accfbR = 'B00100110 [+12]
n14.accftR = 'B10110010 [55710.2]	n14.accfbR = 'B11111000 [+11]
n21.accftR = 'B00000101 [55710.2]	n21.accfbR = 'B10111011 [+12]
n22.accftR = 'B00000000 [5510.7]	n22.accftR = 'B00000000 [10.7]

```
    n23.accftR = 'B11111101 [+9.4]        n23.accfbR = 'B01110010 [+11]
    n24.accftR = 'B10110110 [55710.2]     n24.accfbR = 'B11110100 [+11]

     Clk --> 0    [+0/0]

Time = 293:2  [58500 ns]

    Clk --> 1    [+0/0]
    n11.accftR = 'B00000010 [+10.2]       n11.accfbR = 'B11101111 [+11]
    n12.accftR = 'B00110110 [+10.2]       n12.accfbR = 'B00010011 [+11]
    n13.accftR = 'B00110110 [+10.2]       n13.accfbR = 'B00010011 [+11]
    n14.accftR = 'B11011001 [+10.9]       n14.accfbR = 'B01111100 [+11]
    n21.accftR = 'B00000010 [55710.2]     n21.accfbR = 'B11011101 [+12]
    n22.accftR = 'B00000000 [5510.7]
    n23.accftR = 'B11111110 [+10.2]       n23.accfbR = 'B10111001 [+12]
    n24.accftR = 'B11011011 [+10.2]       n24.accfbR = 'B01111010 [+11]
    n24.comtc2, n14.comtc2, n23.comtc2, n13.comtc2, n22.comtc2, n12.comtc2,
    n21.comtc2, n11.comtc2 = 'B1011 [+11]

Time = 294:1  [58600 ns]

    Clk --> 0    [+0/0]

Time = 294:2  [58700 ns]

    Clk --> 1    [+0/0]
    n11.accftR = 'B00000001 [+10.9]       n11.accfbR = 'B01110111 [+11]
    n12.accftR = 'B00011011 [+10.9]       n12.accfbR = 'B00001001 [+11]
    n13.accftR = 'B00011011 [+10.9]       n13.accfbR = 'B00001001 [+11]
    n14.accftR = 'B11101100 [+10.2]       n14.accfbR = 'B10111110 [+12]
    n21.accftR = 'B00000001 [55710.2]     n21.accfbR = 'B01101110 [+11]
    n22.accftR = 'B00000000 [5510.7]
    n23.accftR = 'B11111111 [+10.2]       n23.accfbR = 'B01011100 [+11]
    n24.accftR = 'B11101101 [+10.9]       n24.accfbR = 'B10111101 [+11]

    n24.comtc2, n14.comtc2, n23.comtc2, n13.comtc2, n22.comtc2, n12.comtc2,
    n21.comtc2, n11.comtc2 = 'B1100 [+11.1]

Time = 295:1  [58800 ns]

    Clk --> 0    [+0/0]
```

```
Time = 295:2   [58900 ns]

     Clk --> 1     [+0/0]
     n11.accftR  = 'B00000000 [+10]        n11.accfbR  = 'B10111011 [+12]
     n12.accftR  = 'B00001101 [+10.2]      n12.accfbR  = 'B10000100 [+12]
     n13.accftR  = 'B00001101 [+10.2]      n13.accfbR  = 'B10000100 [+12]
     n14.accftR  = 'B11110110 [+10.9]      n14.accfbR  = 'B01011111 [+11]
     n21.accftR  = 'B00000000 [55710.2]    n21.accfbR  = 'B10110111 [+11]
     n22.accftR  = 'B00000000 [5510.7]
     n23.accftR  = 'B11111111 [+10.9]      n23.accfbR  = 'B10101110 [+11]
     n24.accftR  = 'B11110110 [+10.2]      n24.accfbR  = 'B11011110 [+12]

     n24.comtc2, n14.comtc2, n23.comtc2, n13.comtc2, n22.comtc2, n12.comtc2,
     n21.comtc2, n11.comtc2  = 'B1101  [+11]

Time = 296:1   [59000 ns]

     Clk --> 0     [+0/0]

Time = 296:2   [59100 ns]

     Clk --> 1     [+0/0]
     n11.accftR  = 'B00000000 [58710]      n11.accfbR  = 'B01011101 [+11]
     n12.accftR  = 'B00000110 [+10.2]      n12.accfbR  = 'B11000010 [+11]
     n13.accftR  = 'B00000110 [+10.2]      n13.accfbR  = 'B11000010 [+11]
     n14.accftR  = 'B11111011 [+10.2]      n14.accfbR  = 'B10101111 [+12]
     n21.accftR  = 'B00000000 [55710.2]    n21.accfbR  = 'B01011011 [+11]
     n22.accftR  = 'B00000000 [5510.7]
     n23.accftR  = 'B11111111 [58710.9]    n23.accfbR  = 'B11010111 [+12]
     n24.accftR  = 'B11111011 [+10.2]      n24.accfbR  = 'B01101111 [+11]

     n24.comtc2, n14.comtc2, n23.comtc2, n13.comtc2, n22.comtc2, n12.comtc2,
     n21.comtc2, n11.comtc2  = 'B1110  [+11.1]

Time = 297:1   [59200 ns]

          Clk --> 0     [+0/0]

Time = 297:2   [59300 ns]

     Clk --> 1     [+0/0]
     n11.accftR  = 'B00000000 [58710]      n11.accfbR  = 'B00101110 [+11]
```

n12.accftR = 'B00000011 [+10.9] n12.accfbR = 'B01100001 [+11]
n13.accftR = 'B00000011 [+10.9] n13.accfbR = 'B01100001 [+11]
n14.accftR = 'B11111101 [+10.2] n14.accfbR = 'B11010111 [+11]
n21.accftR = 'B00000000 [55710.2] n21.accfbR = 'B00101101 [+11]
n22.accftR = 'B00000000 [5510.7]
n23.accftR = 'B11111111 [58710.9] n23.accfbR = 'B11101011 [+11]
n24.accftR = 'B11111101 [+10.2] n24.accfbR = 'B10110111 [+11]

n24.comtc2, n14.comtc2, n23.comtc2, n13.comtc2, n22.comtc2, n12.comtc2, n21.comtc2, n11.comtc2 = 'B1111 [+11]

Time = 298:1 [59400 ns]

Clk --> 0 [+0/0]

Time = 298:2 [59500 ns]

Clk --> 1 [+0/0]
N21.insb --> 0 [+10.7/3.2] N21.st --> 1 [+18/1.7]
N23.insb --> 0 [+10.7/3.2] N23.st --> 1 [+18/1.7]
N22.insb --> 1 [+10.7/3.2] N22.st --> 1 [+19.5/1.7]
N24.insb --> 1 [+10.7/3.2] N11.st --> 1 [+19.5/1.7]
n11.RR3 = 'B01101110 [+11.6]
n21.RR3 = 'B01111100 [+11.6]
n23.RR3 = 'B01000010 [+11.6]

n12.sn = 'B01111111 [+9.2] **N12 Saturé positivement**

n13.sn = 'B01111111 [+9.2] **N13 Saturé positivement**

n14.sn = 'B10000001 [+9.2] **N14 Saturé négativement**

n24.sn = 'B10000001 [+9.2] **N24 Saturé négativement**

Time = 299:1 [59600 ns]

Pour les autres neurones on calcule l'état de sortie On calcule $(V-Vs)^2$

Time = 306:1 [61000 ns]

Clk --> 0 [+0/0]

Time = 306:2 [61100 ns]

Résultat de la multiplication dans R2 et R3

Time = 307:1 [61200 ns]

Formatage du produit $(V-Vs)^2$

Clk --> 0 [+0/0]

Time = 307:2 [61300 ns]

Clk --> 1 [+0/0]

n11.accftR = 'B00010000 [+10.9] n11.accfbR = 'B10010000 [+11]
n21.accftR = 'B00011010 [+9.4] n21.accfbR= 'B01000100 [+10.9]
n22.accftR = 'B00111000 [+9.4] n22.accfbR= 'B01000000 [+10.4]
n23.accftR = 'B00101011 [+10.9] n23.accfbR = 'B11101110 [+11]

n23.comtc2, n22.comtc2, n21.comtc2, n11.comtc2 = 'B1000 [+9.6]

Time = 314:1 [62600 ns]

Clk --> 0 [+0/0]

Time = 314:2 [62700 ns] **Résultat formaté**

Clk --> 1 [+0/0]
n11.accftR = 'B00000000 [62510] n11.accfbR = 'B00100001 [+11]
n21.accftR = 'B00000000 [62510] n21.accfbR = 'B00110100 [+11]
n22.accftR = 'B00000000 [62510] n22.accfbR = 'B01110000 [+11]
n23.accftR = 'B00000000 [62510] n23.accfbR = 'B01010111 [+11]
n23.comtc2, n22.comtc2, n21.comtc2, n11.comtc2 = 'B1111 [+11]

Time = 315:1 [62800 ns]

Clk --> 0 [+0/0]

Time = 315:2 [62900 ns] **Multiplication par + a ou - a**

Clk --> 1 [+0/0]
n11.RR2 = 'B00000000 [+10.3] n11.RR3 = 'B00100000 [+11.8]
n21.RR3 = 'B00011101 [+11.8]
n22.RR2 = 'B00000000 [+10.3] n22.RR3 = 'B01110000 [+11.8]
n23.RR3 = 'B01000010 [+11.8]

Time = 316:1 [63000 ns] **Début de multiplication**

Clk --> 0 [+0/0]

Time = 316:2 [63100 ns]

Time = 324:1 [64600 ns] **Fin de multiplication**

Clk --> 0 [+0/0]

Time = 324:2 [64700 ns] **Chargt. de +1 ou -1 dans la fonction d'activation**

	Valeur chargée dans R3
n11.comtc2 = 'B1100 [+9.6]	n11.RR3 = 'B01111111 [+12.8]
n21.comtc2 = 'B1100 [+9.6]	n21.RR3 = 'B01111111 [+12.8]
n22.comtc2 = 'B1100 [+9.6]	n22.RR3 = 'B01111111 [+12.8]
n23.comtc2 = 'B1100 [+9.6]	n23.RR3 = 'B10000001 [+10.8]

Time = 325:1 [64800 ns]

Clk --> 0 [+0/0]

Time = 325:2 [64900 ns] **Formatage de la dernier multiplication**

Clk --> 1 [+0/0]
n11.accftR = 'B11111110 [+10.2] n11.accfbR = 'B11111000 [+12]
n21.accftR = 'B11111110 [+9.4] n21.accfbR = 'B01100000 [+11]
n22.accftR = 'B11111100 [+10.2] n22.accfbR = 'B10000000 [+11]
n23.accftR = 'B00000010 [+10.9] n23.accfbR = 'B10111000 [+11]

n23.comtc2, n22.comtc2, n21.comtc2, n11.comtc2 = 'B1101 [+11]

Time = 326:1 [65000 ns]

Clk --> 0 [+0/0]

Time = 326:2 [65100 ns]

Clk --> 1 [+0/0]
n23.comtc2, n22.comtc2, n21.comtc2, n11.comtc2 = 'B1110 [+11.1]

Time = 327:1 [65200 ns]

Clk --> 0 [+0/0]

Time = 327:2 [65300 ns] **Fin du dernier formatage**

Clk --> 1 [+0/0]
N21.st --> 0 [+14.3/1.3]
N23.st --> 0 [+14.3/1.3]
N22.st --> 0 [+14.4/1.3]
N11.st --> 0 [+14.4/1.3]
n23.comtc2, n22.comtc2, n21.comtc2, n11.comtc2 = 'B1111 [+11]

Time = 328:1 [65400 ns]

Clk --> 0 [+0/0]

Time = 328:2 [65500 ns] **Les résultats dans les registres de sortie sn**

Clk --> 1 [+0/0]
n11.sn = 'B01010111 [+9.2]
n21.sn = 'B01000100 [+8.5]
n22.sn = 'B00001000 [+9.2]
n23.sn = 'B11011111 [+9.2]

Time = 329:1 [65600 ns]

Préparation à la lecture des états des neurones sur les bus 22 et bus 23

Clk --> 0 [+0/0]
n22.t12l --> 1 [+10.1/.6] n22.t12h --> 1 [+10.1/.6]
n23.t12l --> 1 [+10.1/.6] n23.t12h --> 1 [+10.1/.6]

Time = 329:2 [65700 ns]

Clk --> 1 [+0/0]

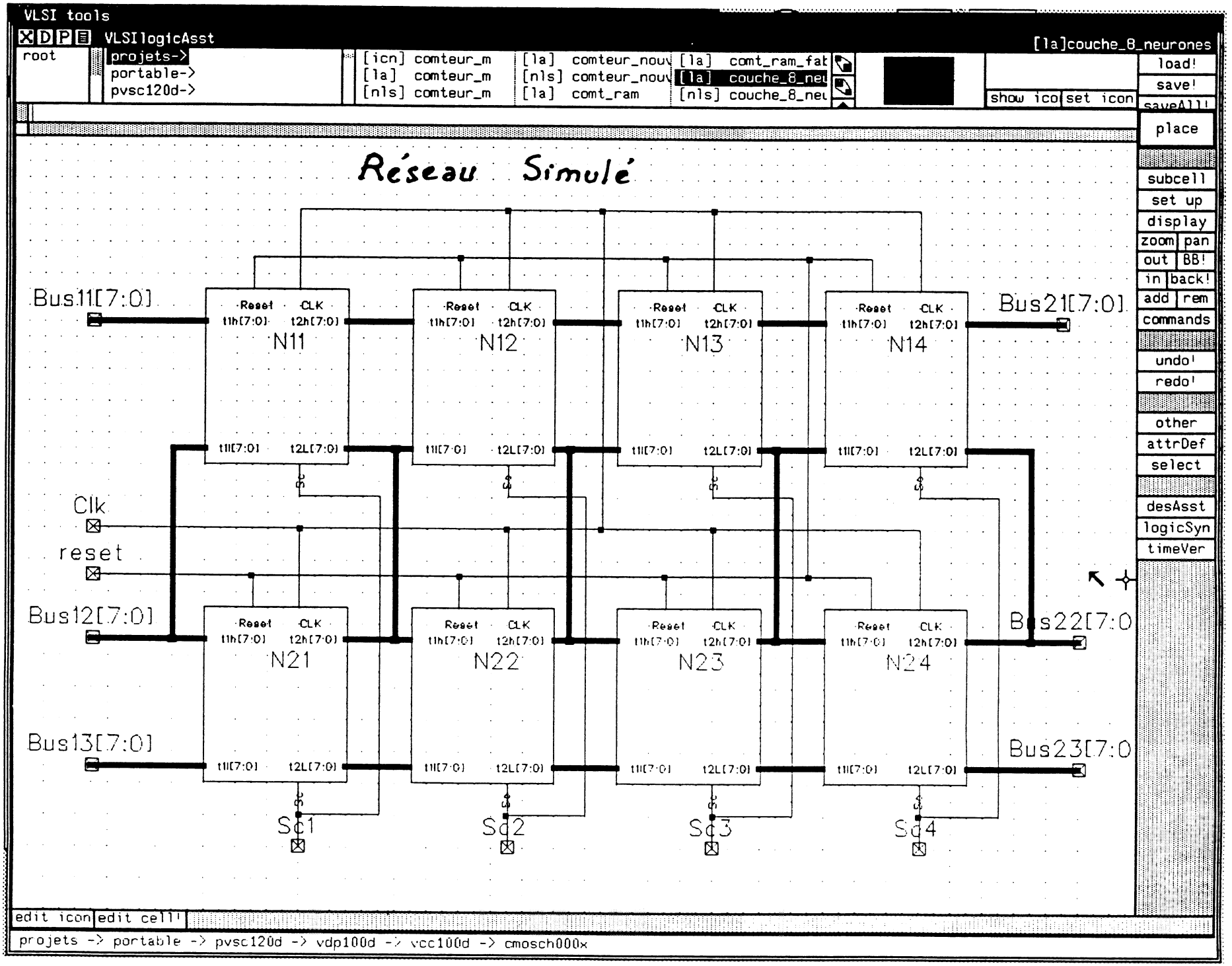
Time = 330:1 [65800 ns] **Lecture de la première colonne**

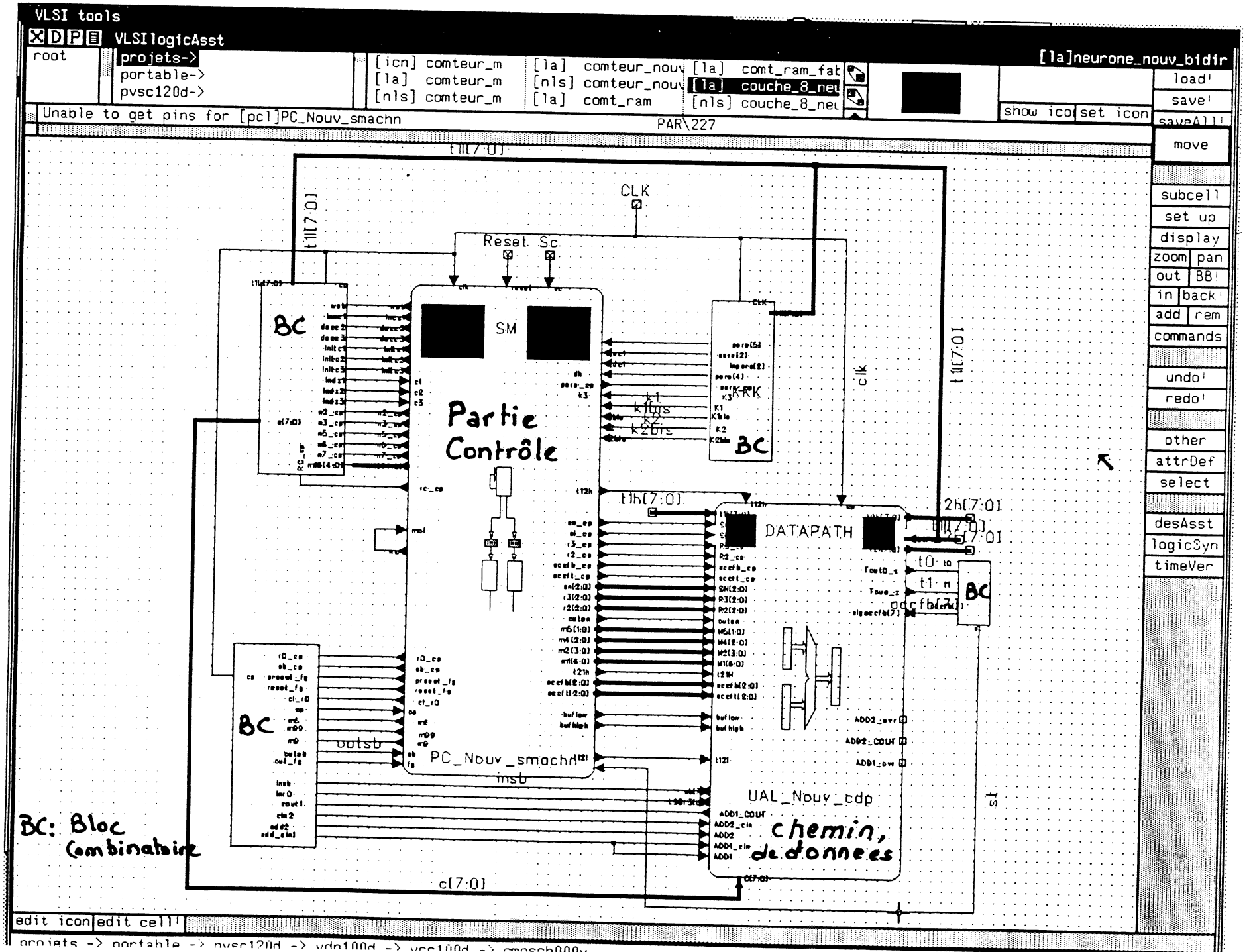
Sc1 --> 1 [+0/0]
Clk --> 0 [+0/0]
Bus22 = 'B01010111 [+40.7] **Etat du neurone n11**
Bus23 = 'B01000100 [+31.1] **Etat du neurone n21**

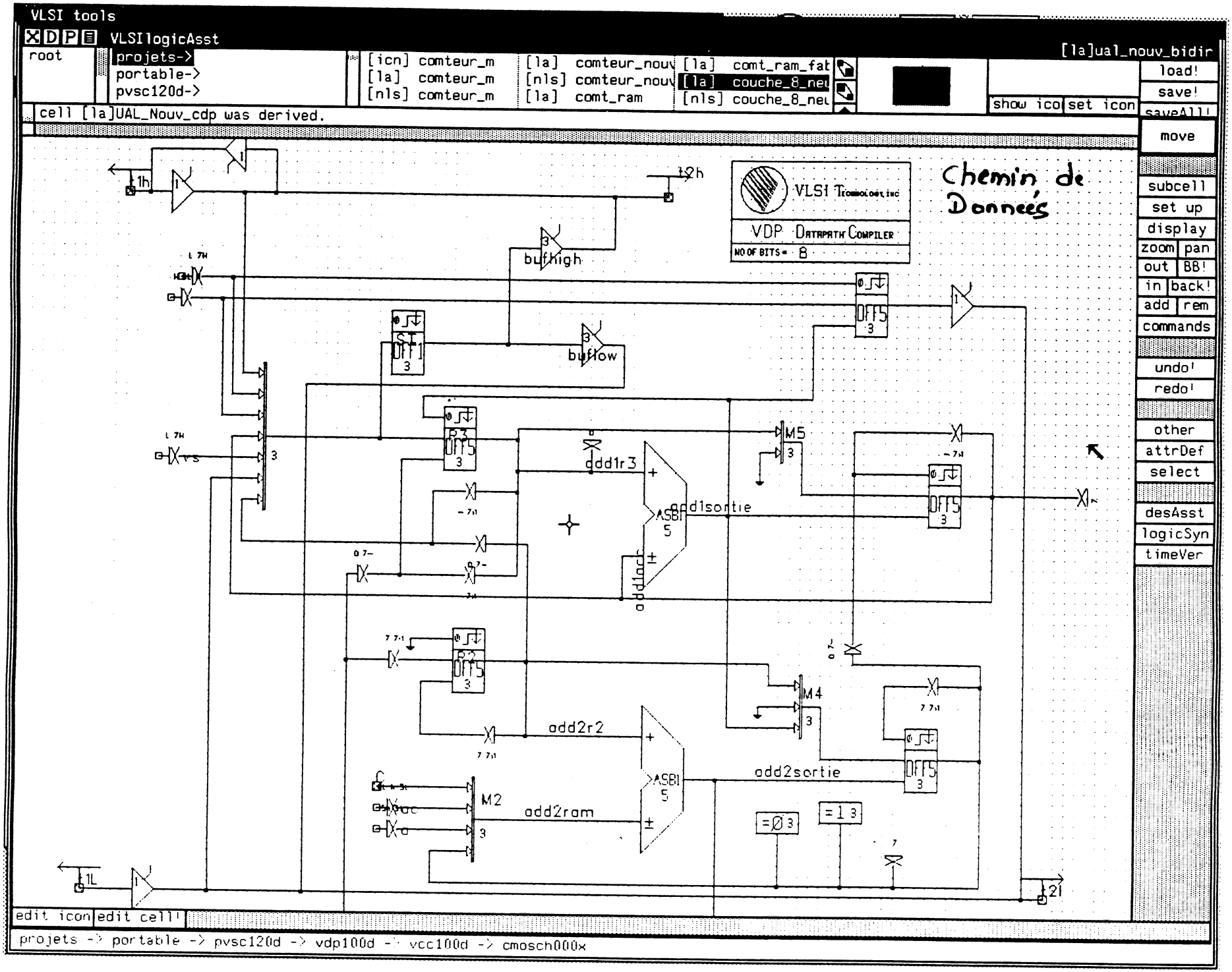
Time = 330:2 [65900 ns]

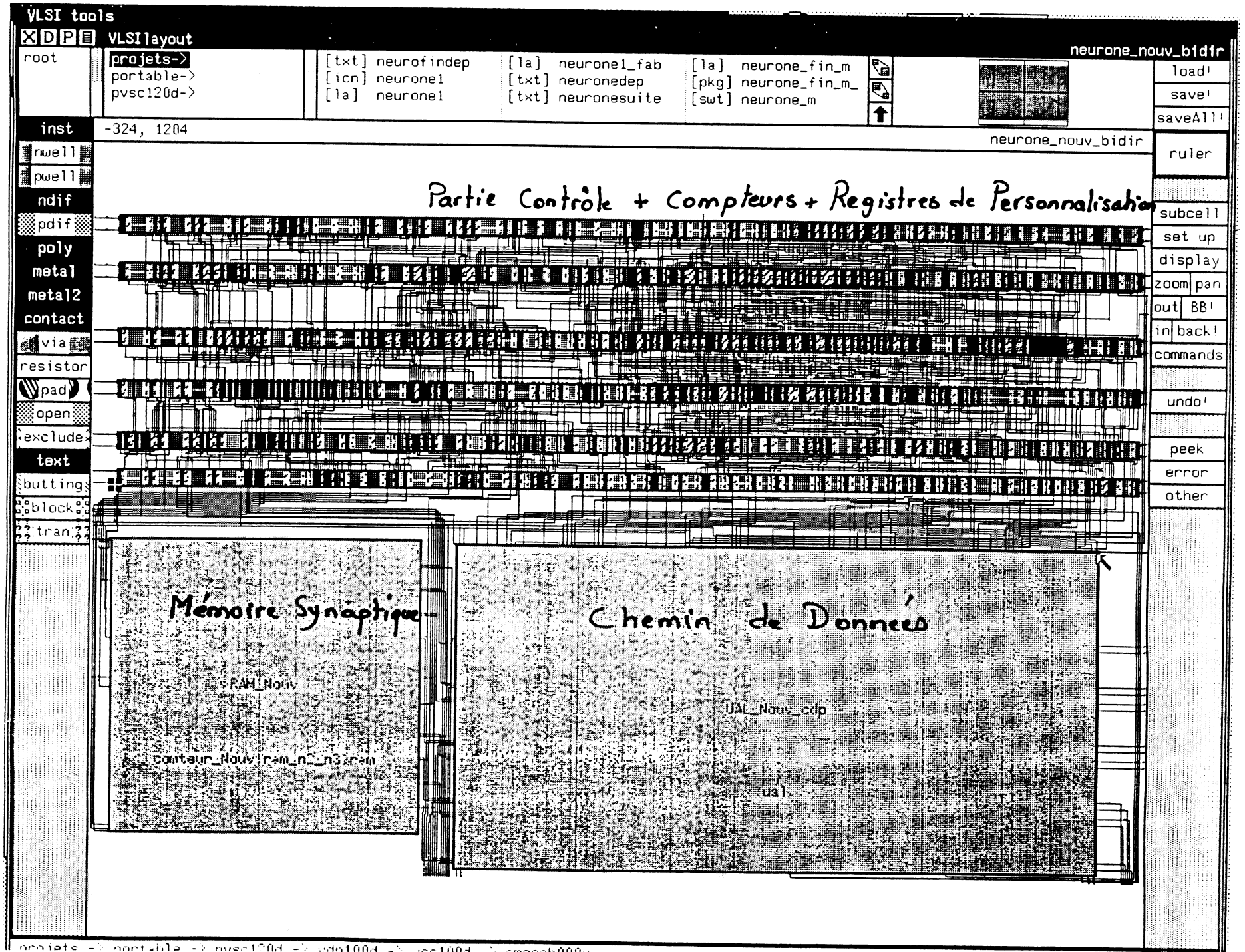
```
    Clk --> 1    [+0/0]
    N11.insb --> 0    [+10.7/3.2]
    N11.st --> 1    [+13.5/1.7]
    N11.st --> 0    [+21.6/1.3]
    n11.comtc2 = 'B1011  [+9.2]        n11.outc3 = 'B010000  [+13.2]
    n21.comtc2 = 'B1011  [+9.2]        n21.outc3 = 'B010000  [+13.2]

Time = 331:1  [66000 ns]
    Sc1 --> 0    [+0/0]
    Sc2 --> 1    [+0/0]        Sélection et lecture de la colonne 2
    Clk --> 0    [+0/0]
    n12.t12h --> 0    [+13.1/.5]
    n22.t12h --> 0    [+13.1/.5]
    n12.t12l --> 0    [+15.7/.5]
    n22.t12l --> 0    [+15.7/.5]
    Bus22 = 'B01111111  [+40.2]          Etat du neurone n12
    Bus23 = 'B00001000  [+36.4]          Etat du neurone n22

Time = 331:2  [66100 ns]

    Clk --> 1    [+0/0]
    N22.insb --> 0    [+10.7/3.2]
    N22.st --> 1    [+13.5/1.7]
    N22.st --> 0    [+21.6/1.3]
    N12.st --> 0    [+21.6/1.3]
    n12.comtc2 = 'B1011  [+9.2]        n12.outc3 = 'B010000  [+13.2]
    n22.comtc2 = 'B1011  [+9.2]        n22.outc3 = 'B010000  [+13.2]

Time = 332:1  [66200 ns]
    Sc2 --> 0    [+0/0]
    Sc3 --> 1    [+0/0]        Sélection et lecture de la colonne 3
    Clk --> 0    [+0/0]
    n23.t12h --> 0    [+13.1/.5]
    n13.t12h --> 0    [+13.1/.5]
    n23.t12l --> 0    [+15.7/.5]
    n13.t12l --> 0    [+15.7/.5]
    Bus22 = 'B01111111  [+30.7]          Etat du neurone n13
    Bus23 = 'B11011111  [+30.7]          Etat du neurone n23

Time = 332:2  [66300 ns]
```

```
    Clk --> 1     [+0/0]
    N13.st --> 0     [+21.6/1.3]
    n13.comtc2 = 'B1011  [+9.2]        n13.outc3 = 'B010000  [+13.2]
    n23.comtc2 = 'B1011  [+9.2]        n23.outc3 = 'B010000  [+13.2]

Time = 333:1   [66400 ns]

    Sc3 --> 0     [+0/0]
    Sc4 --> 1     [+0/0]        Sélection et lecture de la colonne 4
    Clk --> 0     [+0/0]
    n24.t12h --> 0     [+13.1/.5]
    n14.t12h --> 0     [+13.1/.5]
    n24.t12l --> 0     [+15.7/.5]
    n14.t12l --> 0     [+15.7/.5]
    Bus22 = 'B10000001  [+28.3]        Etat du neurone n14
    Bus23 = 'B10000001  [+24]          Etat du neurone n24

Time = 333:2   [66500 ns]

    Clk --> 1     [+0/0]
    N24.insb --> 0     [+10.7/3.2]  N14.st --> 0     [+21.8/1.3]
    N14.insb --> 0     [+10.7/3.2]  N24.st --> 0     [+21.8/1.3]
    n14.comtc2 = 'B1011  [+9.2]        n14.outc3 = 'B010000  [+13.2]
    n24.comtc2 = 'B1011  [+9.2]        n24.outc3 = 'B010000  [+13.2]

Time = 334:1   [66600 ns]

    Sc4 --> 0     [+0/0]
    Clk --> 0     [+0/0]

Time = 334:2   [66700 ns]

    Clk --> 1     [+0/0]

Time = 335:1   [66800 ns]

VLSIsim> VLSI>
```


VLSI tools
VLSIlogicAsst
root
projets->
portable->
pvsc120d->
[icn] comteur_m
[la] comteur_m
[nls] comteur_m
[la] comteur_nouv
[nls] comteur_nouv
[la] comt_ram
[la] comt_ram_fat
[la] couche_8_neu
[nls] couche_8_neu
[la]couche_8_neurones
load!
save!
saveAll!
show ico
set icon
place
subcell
set up
display
zoom
pan
out
BB!
in
back!
add
rem
commands
undo!
redo!
other
attrDef
select
desAsst
logicSyn
timeVer
Réseau Simulé
Bus11[7:0]
Bus12[7:0]
Bus13[7:0]
Bus21[7:0]
Bus22[7:0]
Bus23[7:0]
Clk
reset
Reset
CLK
t1h[7:0]
t2h[7:0]
t1l[7:0]
t2L[7:0]
N11
N12
N13
N14
N21
N22
N23
N24
Sc1
Sc2
Sc3
Sc4
edit icon
edit cell!
projets -> portable -> pvsc120d -> vdp100d -> vcc100d -> cmosch000x


VLSI tools
VLSIlogicAsst
root
projets->
portable->
pvsc120d->
[la]neurone_nouv_bidir
Unable to get pins for [pc]PC_Nouv_smachn
PAR\227
load!
save!
saveAll!
move
subcell
set up
display
zoom pan
out BB!
in back!
add rem
commands
undo!
redo!
other
attrDef
select
desAsst
logicSyn
timeVer
show ico set icon
CLK
Reset Sc
SM
Partie Contrôle
BC
DATAPATH
PC_Nouv_smachn
UAL_Nouv_cdp
chemin de donnees
BC: Bloc Combinatoire
t1l[7:0]
t1h[7:0]
2h[7:0]
c[7:0]
edit icon
edit cell!


VLSI tools
VLSIlogicAsst
root
projets->
portable->
pvsc120d->
[la]ual_nouv_bidir
load!
save!
saveAll!
show ico
set icon
move
cell [la]UAL_Nouv_cdp was derived.
VLSI TECHNOLOGY, INC
VDP DATAPATH COMPILER
NO OF BITS = 8
Chemin de Donnees
subcell
set up
display
zoom
pan
out
BB!
in
back!
add
rem
commands
undo!
redo!
other
attrDef
select
desAsst
logicSyn
timeVer
buthigh
butlow
ddd1r3
add1sortie
add1acc
add2r2
add2ram
add2sortie
M5
M4
M2
ASBI
edit icon
edit cell!
projets -> portable -> pvsc120d -> vdp100d -> vcc100d -> cmosch000x


VLSI tools
VLSIlayout
root
projets->
portable->
pvsc120d->
[txt] neurofindep
[icn] neurone1
[la] neurone1
[la] neurone1_fab
[txt] neuronedep
[txt] neuronesuite
[la] neurone_fin_m
[pkg] neurone_fin_m_
[swt] neurone_m
neurone_nouv_bidir
load!
save!
saveAll!
inst
-324, 1204
neurone_nouv_bidir
ruler
nwell
pwell
ndif
pdif
poly
metal
metal2
contact
via
resistor
pad
open
exclude
text
butting
block
tran
Partie Contrôle + Compteurs + Registres de Personnalisation
subcell
set up
display
zoom pan
out BB!
in back!
commands
undo!
peek
error
other
Mémoire Synaptique
RAM_Nouv
compteur_Nouv
UAL_Nouv_cdp
ual
projets -> portable -> pvsc120d -> vdp100d ->

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Grenoble, le 28 Mai 1991

DÉPARTEMENT DES ÉTUDES DOCTORALES

Affaire suivie par
Tél : 76.57.

V/Réf. :

Objet :

AUTORISATION de SOUTENANCE

Vu les dispositions de l'arrêté du 23 Novembre 1988 relatif aux Etudes Doctorales
Vu les rapports de présentation de :

- Monsieur WEINFELD
- Monsieur BREYFUS

Monsieur OUALI Jamel Eddine

est autorisé(e) à présenter une thèse en soutenance en vue de l'obtention du diplôme de DOCTEUR de l'INSTITUT NATIONAL POLYTECHNIQUE de GRENOBLE, spécialité :

"Microélectronique"

Pour le Président de l'I.N.P.-G.
et par délégation,
le Vice-Président
M. GARNIER

Résumé

Ayant rappelé brièvement quelques réalisations matérielles de réseaux de neurones artificiels dans un premier chapitre, cette thèse propose une architecture distribuée, synchrone fondée sur l'existence d'un processeur neurone autonome. Ce processeur pourra être personnalisé suivant les caractéristiques du réseau de neurones à implanter et pourra être connecté à d'autres neurones pour former un réseau de structure et de dimension fixées. Ce neurone se présente comme un circuit dédié fabriqué dans un temps court dans un environnement du type compilateur de silicium. Un tel neurone a été conçu et fabriqué et c'est opérationnel. Il implante sous sa version fabriqué uniquement la phase de reconnaissance.

Dans un troisième chapitre, on montre que sans modification de l'architecture, on peut inclure des possibilités d'apprentissage. Pour ceci un algorithme d'apprentissage par la rétropropagation du gradient a été proposé et étudié et on montre son implantation sur le réseau de neurones proposé en précisant l'adjonction de la partie contrôle du neurone à implanter.

Enfin dans un dernier chapitre, nous explorons la possibilité de réaliser de très grands circuits ce qui serait très judicieux pour faire face à la taille des réseaux de neurones requises pour les applications.

Pour ceci nous explorons les possibilités d'intégration sur tranche entière. En effet, il existe une tolérance aux défauts intrinsèques au calcul neuronal et de plus l'implantation physique régulière doit permettre d'isoler et d'isoler et d'exclure les neurones défaillants. Les possibilités d'implantation physique d'une architecture sur tranche entière sont donc présentées dans ce chapitre.

Mots clés

Réseaux de neurones, Silicium, Architecture neuronique décentralisée, Implantation digitale, Neurone paramétrable.