



# Architecture massivement parallèle : un réseau de cellules intégré pour la reconstruction d'images

Didier Lattard

## ► To cite this version:

Didier Lattard. Architecture massivement parallèle : un réseau de cellules intégré pour la reconstruction d'images. Modélisation et simulation. Institut National Polytechnique de Grenoble - INPG, 1989. Français. NNT : . tel-00335786

**HAL Id: tel-00335786**

**<https://theses.hal.science/tel-00335786>**

Submitted on 30 Oct 2008

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# THESE

présentée par

Didier LATTARD

pour obtenir le titre de **DOCTEUR**

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

(arrêté ministériel du 23 novembre 1988)

---

*ARCHITECTURE MASSIVEMENT PARALLELE :*  
*UN RESEAU DE CELLULES INTEGRE*  
*POUR LA RECONSTRUCTION D'IMAGES*

---

Date de soutenance : 8 novembre 1989

Composition du jury :

|           |                     |             |
|-----------|---------------------|-------------|
| Messieurs | Jacques MOSSIERE    | Président   |
|           | Jean-Louis AMANS    |             |
|           | Guy MAZARE          |             |
|           | Jean-Luc BASILLE    | Rapporteurs |
|           | Jean-Paul SANSONNET |             |

Thèse préparée au sein du Laboratoire de **G**énie Informatique



# INSTITUT NATIONAL POLYTECHNIQUE GRENOBLE

46 avenue F. Viallet - 38031 GRENOBLE Cedex -

Tél : 76.57.45.00

ANNEE UNIVERSITAIRE 1989

Président de l'Institut  
Monsieur Georges LESPINARD

## PROFESSEURS DES UNIVERSITES

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

.../...



SITUATION PARTICULIERE

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   | Récteur 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 |

.../...

PERSONNES AYANT OBTENU LE DIPLOME  
d'habilitation à diriger des recherches

|            |      |          |      |           |    |              |      |
|------------|------|----------|------|-----------|----|--------------|------|
| BECKER     | M.   | DANES    | F.   | GHIBAUDO  | G. | MULLER       | J.   |
| BINDER     | Z.   | DEROO    | D.   | HAMAR     | S. | NGUYEN TRONG | B.   |
| CHASSERY   | J.M. | DIARD    | J.P. | HAMAR     | R. | NIEZ         | J.J. |
| CHOLLET    | J.P. | DION     | J.M. | LACHENAL  | D. | PASTUREL     | A.   |
| COEY       | J.   | DUGARD   | L.   | LADET     | P. | PLA          | F.   |
| COLINET    | C.   | DURAND   | M.   | LATOMBE   | C. | ROGNON       | J.P. |
| COMMAULT   | C.   | DURAND   | R.   | LE HUY    | H. | ROUGER       | J.   |
| CORNUEJOLS | G.   | GALERIE  | A.   | LE GORREC | B. | TCHUENTE     | M.   |
| COULOMB    | J.L. | GAUTHIER | J.P. | MADAR     | R. | VINCENT      | H.   |
| COURNIL    | M.   | GENTIL   | S.   | MEUNIER   | G. | YAVARI       | A.R. |
| DALARD     | F.   |          |      |           |    |              |      |

CHERCHEURS DU C.N.R.S.

DIRECTEURS DE RECHERCHE CLASSE 0

|           |         |
|-----------|---------|
| LANDAU    | Ioan    |
| NAYROLLES | Bernard |

DIRECTEURS DE RECHERCHE 1ère CLASSE

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

DIRECTEURS DE RECHERCHE 2ème CLASSE

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

**PERSONNALITES AGREEES A TITRE PERMANENT A DIRIGER DES TRAVAUX DE RECHERCHE**  
**( DECISION DU CONSEIL SCIENTIFIQUE )**

|               |                                             |                                             |
|---------------|---------------------------------------------|---------------------------------------------|
| <u>ENSEEG</u> | HAMMOU<br>MARTIN-GARIN<br>SARRAZIN<br>SIMON | Abdelkader<br>Régina<br>Pierre<br>Jean-Paul |
|---------------|---------------------------------------------|---------------------------------------------|

|               |       |        |
|---------------|-------|--------|
| <u>ENSERG</u> | BOREL | Joseph |
|---------------|-------|--------|

|               |                                                |                                          |
|---------------|------------------------------------------------|------------------------------------------|
| <u>ENSIEG</u> | DESCHIZEAUX<br>GLANGEAUD<br>PERARD<br>REINISCH | Pierre<br>François<br>Jacques<br>Raymond |
|---------------|------------------------------------------------|------------------------------------------|

|               |      |       |
|---------------|------|-------|
| <u>ENSHMG</u> | ROWE | Alain |
|---------------|------|-------|

|                |         |         |
|----------------|---------|---------|
| <u>ENSIMAG</u> | COURTIN | Jacques |
|----------------|---------|---------|

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

**Laboratoire extérieurs :**

|                 |                                        |                                      |
|-----------------|----------------------------------------|--------------------------------------|
| <u>C.N.E.T.</u> | DEVINE<br>GERBER<br>MERCKEL<br>PAULEAU | Rodericq<br>Roland<br>Gérard<br>Yves |
|-----------------|----------------------------------------|--------------------------------------|

XXXXXXXXXXXXXXXXXXXX



Président de l'Université :

M. NEMOZ Alain

ANNEE UNIVERSITAIRE 1988 - 1989

MEMBRES DU CORPS ENSEIGNANT DE SCIENCES ET DE GEOGRAPHIE

PROFESSEURS DE 1ERE CLASSE

ADIBA Michel  
ANTOINE Pierre  
ARNAUD Paul  
ARVIEU Robert  
AURIAULT Jean Louis  
AYANT Yves  
BARJON Robert  
BARNOUD Fernand  
BARRA Jean René  
BECKER Pierre  
BEGUIN Claude  
BELORISKY Elie  
BENZAKEN Claude  
BERARD Pierre  
BERNARD Alain  
BERTRANDIAS Françoise  
BERTRANDIAS Jean Paul  
BILLET Jean  
BOEHLER Jean Paul  
BRAVARD Yves  
CARLIER Georges  
CASTAING Bernard  
CAUQUIS Georges  
CHARDON Michel  
CHIBON Pierre  
COHEN ADDAD Jean Pierre  
COLIN DE VERDIERE Yves  
CYROT Michel  
DEBELMAS Jacques  
DEGRANGE Charles  
DEMAILLY Jean Pierre  
DENEUVILLE Alain  
DEPORTES Charles  
DOLIQUE Jean Michel  
DOUCE Roland  
DUCROS Pierre  
FINKE Gerde  
GAGNAIRE Didier  
GAUTRON René  
GENIES Eugène  
GERMAIN Jean Pierre  
GIDON Maurice  
GUITTON Jacques  
HICTER Pierre  
IDELMAN Simon  
JANIN Bernard  
JOLY Jean René  
JOSELEAU Jean Paul

Informatique  
Géologie I.R.I.G.M.  
Chimie Organique  
Physique Nucléaire I.S.N.  
Mécanique  
Physique Approfondie  
Physique Nucléaire I.S.N.  
Biochimie Macromoléculaire Végétale  
Statistiques - Mathématiques Appliquées  
Physique  
Chimie Organique  
Physique  
Mathématiques Pures  
Mathématiques Pures  
Mathématiques Pures  
Mathématiques Pures  
Mathématiques Pures  
Géographie  
Mécanique  
Géographie  
Biologie Végétale  
Physique  
Chimie Organique  
Géographie  
Biologie Animale  
Physique  
Mathématiques Pures  
Physique du Solide  
Géologie Générale  
Zoologie  
Mathématiques Pures  
Physique  
Chimie Minérale  
Physique des Plasmas  
Physiologie Végétale  
Cristallographie  
Informatique  
Chimie Physique  
Chimie  
Chimie  
Mécanique  
Géologie  
Chimie  
Chimie  
Physiologie Animale  
Géographie  
Mathématiques Pures  
Biochimie

KAHANE André, détaché  
 KAHANE Josette  
 KRAKOWIAK Sacha  
 LAJZEROWICZ Jeanine  
 LAJZEROWICZ Joseph  
 LAURENT Pierre Jean  
 LEBRETON Alain  
 DE LEIRIS Joël  
 LHOMME Jean  
 LLIBOUTRY Louis  
 LOISEAUX Jean Marie  
 LONGEQUEUE Nicole  
 LUNA Domingo  
 MACHE Régis  
 MASCLE Georges  
 MAYNARD Roger  
 OMONT Alain  
 OZENDA Paul  
 PANNETIER Jean  
 PAYAN Jean Jacques  
 PEBAY PEYROULA Jean Claude  
 PERRIER Guy  
 PIERRE Jean Louis  
 RENARD Michel  
 RIEDTMANN Christine  
 RINAUDO Marguerite  
 ROSSI André  
 SAXOD Raymond  
 SENDEL Philippe  
 SERGERAERT Francis  
 SOUCHIER Bernard  
 SOUTIF Michel  
 STUTZ Pierre  
 TRILLING Laurent  
 VAN CUTSEM Bernard  
 VIALON Pierre

PROFESSEURS DE 2EME CLASSE

ARMAND Gilbert  
 ATTANE Pierre  
 BARET Paul  
 BERTIN José  
 BLANCHI Jean Pierre  
 BLOCK Marc  
 BLUM Jacques  
 BOITET Christian  
 BORNAREL Jean  
 BORRIONE Dominique  
 BOUVET Jean  
 BROSSARD Jean  
 BRUANDET Jean François  
 BRUGAL Gérard  
 BRUN Gilbert  
 CERFF Rudiger  
 CHIARAMELLA Yves  
 CHOLLET Jean Pierre  
 COLOMBEAU Jean François  
 COURT Jean  
 CUNIN Pierre Yves

Physique  
 Physique  
 Mathématiques Appliquées  
 Physique  
 Physique  
 Mathématiques Appliquées  
 Mathématiques Appliquées  
 Biologie  
 Chimie  
 Géophysique  
 Sciences Nucléaires I.S.N.  
 Physique  
 Mathématiques Pures  
 Physiologie Végétale  
 Géologie  
 Physique du Solide  
 Astrophysique  
 Botanique (Biologie Végétale)  
 Chimie  
 Mathématiques Pures  
 Physique  
 Géophysique  
 Chimie Organique  
 Thermodynamique  
 Mathématiques  
 Chimie C.E.R.M.A.V.  
 Biologie  
 Biologie Animale  
 Biologie Animale  
 Mathématiques Pures  
 Biologie  
 Physique  
 Mécanique  
 Mathématiques Appliquées  
 Mathématiques Appliquées  
 Géologie

Géographie  
 Mécanique  
 Chimie  
 Mathématiques  
 S.T.A.P.S.  
 Biologie  
 Mathématiques Appliquées  
 Mathématiques Appliquées  
 Physique  
 Automatique informatique  
 Biologie  
 Mathématiques  
 Physique  
 Biologie  
 Biologie  
 Biologie  
 Mathématiques Appliquées  
 Mécanique  
 Mathématiques (ENSL)  
 Chimie  
 Informatique

DAVID Jean  
 DHOUAILLY Danielle  
 DUFRESNOY Alain  
 GASPARD François  
 GIDON Maurice  
 GIGNOUX Claude  
 GILLARD Roland  
 GIORNI Alain  
 GONZALEZ SPRINBERG Gérardo  
 GUIGO Maryse  
 GUMUCHIAN Hervé  
 HACQUES Gérard  
 HERBIN Jacky  
 HERAULT Jeanny  
 HERINO Roland  
 JARDON Pierre  
 KERCKHOVE Claude  
 MANDARON Paul  
 MARTINEZ Francis  
 MOREL Alain  
 NEMOZ Alain  
 NGUYEN HUY Xuong  
 OUDET Bruno  
 PAUTOU Guy  
 PECHER Arnaud  
 PELMONT Jean  
 PELLETIER Guy  
 PERRIN Claude  
 PIBOULE Michel  
 RAYNAUD Hervé  
 REGNARD Jean René  
 RICHARD Jean Marc  
 RIEDTMANN Christine  
 ROBERT Danielle  
 ROBERT Gilles  
 ROBERT Jean Bernard  
 SARROT REYNAULD Jean  
 SAYETAT Françoise  
 SERVE Denis  
 STOECKEL Frédéric  
 SCHOLL Pierre Claude  
 SUBRA Robert  
 VALLADE Marcel  
 VIDAL Michel  
 VINCENT Gilbert  
 VIVIAN Robert  
 VOTTERO Philippe

Géographie  
 Biologie  
 Mathématiques Pures  
 Physique  
 Géologie  
 Sciences Nucléaires  
 Mathématiques Pures  
 Sciences Nucléaires  
 Mathématiques Pures  
 Géographie  
 Géographie  
 Mathématiques Appliquées  
 Géographie  
 Physique  
 Physique  
 Chimie  
 Géologie  
 Biologie  
 Mathématiques Appliquées  
 Géographie  
 Thermodynamique C.N.R.S. C.R.T.B.T.  
 Informatique  
 Mathématiques Appliquées  
 Biologie  
 Géologie  
 Biochimie  
 Astrophysique  
 Sciences Nucléaires I.S.N.  
 Géologie  
 Mathématiques Appliquées  
 Physique  
 Physique  
 Mathématiques Pures  
 Chimie  
 Mathématiques Pures  
 Chimie Physique  
 Géologie  
 Physique  
 Chimie  
 Physique  
 Mathématiques Appliquées  
 Chimie  
 Physique  
 Chimie Organique  
 Physique  
 Géographie  
 Chimie

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

### PROFESSEURS DE 1ERE CLASSE

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

### PROFESSEURS DE 2EME CLASSE

|                      |                           |
|----------------------|---------------------------|
| BEE Marc             | Physique I.U.T. 1         |
| BOUTHINON Michel     | E.E.A. I.U.T. 1           |
| CHAMBON René         | Génie Mécanique I.U.T. 1. |
| CHENAVAS Jean        | Physique I.U.T. 1         |
| CHILO Jean           | Physique I.U.T. 1         |
| CHOUTEAU Gérard      | Physique I.U.T. 1         |
| CONTE René           | Physique I.U.T. 1         |
| FOSTER Panayotis     | Chimie I.U.T. 1           |
| GOSSE Jean Pierre    | E.E.A. I.U.T. 1           |
| GROS Yves            | Physique I.U.T. 1         |
| HAMAR Roger          | Chimie I.U.T. 1           |
| KUHN Gérard, détaché | Physique I.U.T. 1         |
| LEVIEL Jean Louis    | Physique I.U.T. 1         |
| MAZUER Jean          | Physique I.U.T. 1         |
| MICHOULIER Jean      | Physique I.U.T. 1         |
| MONLLOR Christian    | E.E.A. I.U.T. 1           |
| PERRAUD Robert       | Chimie I.U.T. 1           |
| PIERRE Gérard        | Chimie I.U.T. 1           |
| TERRIEZ Jean Michel  | Génie Mécanique I.U.T. 1  |
| TOUZAIN Philippe     | Chimie I.U.T. 1           |
| TURGEMAN Sylvain     | Génie civil               |
| VINCENDON Marc       | Chimie I.U.T. 1           |
| ZIGONE Michel        | Physique I.U.T. 1         |







Je tiens à remercier

Monsieur Jacques MOSSIERE, Directeur de l'ENSIMAG, de me faire l'honneur de présider le jury de cette thèse ;

Monsieur Guy MAZARE, Professeur à l'ENSIMAG, de m'avoir accueilli dans son équipe et conseillé tout au long de la préparation de cette thèse ;

Messieurs Jean-Luc BASILLE, Professeur à l'INP de Toulouse (ENSEEIH), et Jean-Paul SANSONNET, Directeur de Recherche au CNRS et Directeur du PRC "Architecture de Machines Nouvelles", d'avoir accepté d'être rapporteurs de cette thèse ;

Monsieur Jean-Louis AMANS, Docteur Ingénieur à la Division LETI (CEA), pour l'intérêt qu'il a porté à mon travail et pour ses nombreux conseils.

Je remercie également toutes les personnes que j'ai côtoyées pendant les trois années de préparation de cette thèse. Grâce à leur aide et à leur amabilité, j'ai pu mener à bien ce travail dans de bonnes conditions.



## **Résumé**

Depuis quelques années, l'intérêt pour des problèmes de grande complexité tels que le traitement du signal et de l'image, et la limitation de puissance due à la séquentialité des opérations des machines basées sur le modèle de Von Neumann, ont poussé les informaticiens à étudier un nouveau modèle de traitement caractérisé par l'exécution parallèle des opérations. Nous proposons dans cette thèse une architecture massivement parallèle basée sur un réseau régulier de cellules, qui ont la particularité d'être totalement asynchrones et de pouvoir communiquer entre elles grâce à un mécanisme d'acheminement de messages. Chaque cellule comprend une partie de traitement élémentaire réalisant les fonctions nécessitées par l'application et une partie routage permettant d'acheminer les messages. Nous avons validé cette architecture en développant une machine complète dédiée à la reconstruction d'images.

Pour cette application particulière, chaque cellule traite une sous-image, et le réseau est utilisé pour réaliser des opérations d'épandage et de projection, qui sont essentielles dans les principales méthodes de reconstruction. Les différents paramètres de la cellule sont définis en fonction des contraintes temporelles, de manière à obtenir d'excellentes performances et une bonne activité globale du réseau. L'interface dans un environnement hôte d'une telle machine a été étudiée. Nous avons réalisé un circuit intégré comprenant une cellule, afin de développer une maquette de démonstration.

## **Mots-clés**

Architecture parallèle, Réseau cellulaire, Communication, Circuit intégré, Reconstruction d'images, Epandage, Projection.



## **Abstract**

For a few years, the interest in great complexity problems such as signal and image processing, and the limited power caused by the operation sequentiality of the machines based upon the Von Neumann model, have induced computer scientists to study a new processing model characterized by parallel operation execution. We propose in this thesis a highly parallel architecture based upon a regular network of cells which have the characteristics to be completely asynchronous, and to be able to communicate between themselves using a message routing mechanism. Each cell includes a simple processing part which realize the necessary functions of the application, and a routing part which insures messages transportation. The architecture has been validated by developing a complete machine dedicated to image reconstruction.

For this particular application, each cell deals with a sub-image, and the cellular array is used to realize back-projection and projection processings, which are essential in reconstruction techniques. The different cell parameters are defined according to temporal constraints, in such a way to obtain excellent performances, and good global activity of the network. The interface of such an architecture in a host environment has been studied. An integrated circuit including one cell has been designed, in order to develop a demonstration machine.

## **Keywords**

Parallel architecture, Cellular network, Communication, Integrated circuit, Image reconstruction, Back-projection, projection.



## ***Table des matières***

---



## Table des matières

|                                                     |            |
|-----------------------------------------------------|------------|
| <b>Introduction.....</b>                            | <b>25</b>  |
| <b>Chapitre 1 - La reconstruction d'images.....</b> | <b>35</b>  |
| <b>1.1. Présentation du problème .....</b>          | <b>39</b>  |
| 1.1.1. Introduction .....                           | 39         |
| 1.1.2. L'aspect physique.....                       | 39         |
| 1.1.3. Position détaillée du problème .....         | 41         |
| <b>1.2. Les méthodes de reconstruction .....</b>    | <b>46</b>  |
| 1.2.1. Historique.....                              | 46         |
| 1.2.2. Les méthodes de reconstruction .....         | 48         |
| 1.2.3. Discussion .....                             | 56         |
| <b>1.3. Vers une machine parallèle.....</b>         | <b>58</b>  |
| 1.3.1. Pourquoi une machine parallèle?.....         | 58         |
| 1.3.2. Les machines existantes.....                 | 59         |
| 1.3.3. Le réseau cellulaire .....                   | 62         |
| <b>Chapitre 2 - La machine cellulaire.....</b>      | <b>65</b>  |
| <b>2.1. Introduction .....</b>                      | <b>69</b>  |
| 2.1.1. Stratégie globale.....                       | 69         |
| 2.1.2. L'opération d'épandage .....                 | 70         |
| 2.1.3. L'opération de projection.....               | 71         |
| 2.1.4. La machine parallèle .....                   | 73         |
| <b>2.2. Le réseau cellulaire .....</b>              | <b>74</b>  |
| 2.2.1. Architecture générale .....                  | 74         |
| 2.2.2. Mécanisme de communication.....              | 75         |
| 2.2.3. La cellule .....                             | 77         |
| 2.2.4. La machine cellulaire .....                  | 78         |
| 2.2.5. Les applications.....                        | 79         |
| 2.2.6. La reconstruction d'images .....             | 79         |
| <b>2.3. La cellule .....</b>                        | <b>80</b>  |
| 2.3.1. Sa fonction .....                            | 80         |
| 2.3.2. L'algorithme .....                           | 84         |
| 2.3.3. La structure des messages .....              | 86         |
| 2.3.4. Amélioration de l'aspect routage.....        | 95         |
| 2.3.5. Le codage des informations géométriques..... | 97         |
| 2.3.6. Traitement d'une sous-image.....             | 104        |
| <b>2.4. L'ordinateur hôte .....</b>                 | <b>112</b> |
| 2.4.1. Son rôle .....                               | 112        |
| 2.4.2. L'algorithme .....                           | 113        |
| <b>2.5. Les entrées et sorties des images.....</b>  | <b>114</b> |
| 2.5.1. Introduction .....                           | 114        |
| 2.5.2. Sortie de l'image .....                      | 116        |
| 2.5.3. Entrée d'une image dans le réseau .....      | 119        |

## **Chapitre 3 - Les simulations.....125**

|                                                          |            |
|----------------------------------------------------------|------------|
| <b>3.1. Introduction .....</b>                           | <b>129</b> |
| <b>3.2. Les simulations fonctionnelles OCCAM .....</b>   | <b>129</b> |
| 3.2.1. Le langage OCCAM .....                            | 129        |
| 3.2.2. Le Transputer.....                                | 130        |
| 3.2.3. Structure du programme de simulation.....         | 131        |
| 3.2.4. Les résultats : images.....                       | 134        |
| 3.2.5. Conclusion.....                                   | 147        |
| <b>3.3. Les simulations comportementales HELIX .....</b> | <b>148</b> |
| 3.3.1. Silvar_Lisco et le simulateur HELIX .....         | 148        |
| 3.3.2. Structure du programme de simulation.....         | 149        |
| 3.3.3. Les résultats : performances du réseau .....      | 149        |
| 3.3.4. Conclusion.....                                   | 155        |

## **Chapitre 4 - Architecture de la cellule.....157**

|                                                                 |            |
|-----------------------------------------------------------------|------------|
| <b>4.1. Introduction .....</b>                                  | <b>161</b> |
| 4.1.1. Définition des paramètres de la cellule .....            | 161        |
| 4.1.2. Méthode de conception .....                              | 163        |
| <b>4.2. Définition de l'architecture .....</b>                  | <b>163</b> |
| 4.2.1. Les différentes fonctions de la cellule .....            | 163        |
| 4.2.2. L'architecture proposée.....                             | 164        |
| 4.2.3. Les tampons d'entrée et sortie .....                     | 166        |
| 4.2.4. La partie opérative .....                                | 168        |
| 4.2.5. Le séquençement.....                                     | 184        |
| <b>4.3. Les simulations .....</b>                               | <b>188</b> |
| 4.3.1. Présentation des simulations .....                       | 188        |
| 4.3.2. Validation de l'architecture.....                        | 192        |
| <b>4.4. Implantation.....</b>                                   | <b>192</b> |
| 4.4.1. Le logiciel SOLO1200.....                                | 192        |
| 4.4.2. Réalisation du circuit .....                             | 193        |
| 4.4.3. Le brochage du circuit.....                              | 200        |
| 4.4.4. Performances.....                                        | 202        |
| 4.4.5. L'interfaçage du circuit dans l'environnement hôte ..... | 207        |

## **Conclusion .....213**

## **Bibliographie.....219**

|                                                                             |            |
|-----------------------------------------------------------------------------|------------|
| <b>Annexe 1 - Aspect mathématique de la reconstruction d'images.....</b>    | <b>233</b> |
| 1. Les méthodes analytiques.....                                            | 235        |
| 2. Les méthodes algébriques.....                                            | 238        |
| <b>Annexe 2 - Systèmes d'acquisition et de reconstruction .....</b>         | <b>245</b> |
| 1. Introduction.....                                                        | 247        |
| 2. Les différentes générations de machines.....                             | 247        |
| 3. L'unité d'acquisition.....                                               | 250        |
| 4. L'unité de calcul.....                                                   | 250        |
| 5. L'affichage des résultats.....                                           | 251        |
| 6. La géométrie des machines et les techniques de reconstruction.....       | 251        |
| <b>Annexe 3 - Les simulations du réseau .....</b>                           | <b>255</b> |
| 1. Les simulations fonctionnelles OCCAM .....                               | 257        |
| 2. Les simulations comportementales HELIX .....                             | 261        |
| <b>Annexe 4 - Description de l'architecture de la cellule de base .....</b> | <b>273</b> |
| 1. Les simulations comportementales HELIX .....                             | 275        |
| 2. Simulations électriques d'un additionneur .....                          | 285        |
| 3. Simulations logiques.....                                                | 293        |
| 4. Dessins des masques.....                                                 | 305        |
| 5. Plan de montage.....                                                     | 310        |



## ***Introduction***

---

## Introduction

## **Les architectures parallèles**

Depuis l'avènement de l'informatique, la plupart des machines ont été réalisées en suivant le modèle de Von Neumann. Bien que ce type de machines ait subi une évolution améliorant considérablement leurs performances, elles restent néanmoins très limitées de par l'enchaînement séquentiel des opérations qu'elles exécutent.

Or, l'intérêt pour des classes de problèmes nécessitant des puissances de calcul très supérieures à celles des machines séquentielles a engendré un engouement certain pour le traitement parallèle des informations. Parmi ces applications se prêtant particulièrement bien à une exécution parallèle, nous pouvons citer le traitement du signal et de l'image.

Des architectures parallèles composées d'un nombre limité de processeurs généraux puissants, dites à gros grains, ont été réalisées (Supernode, FPS, ...). Ces machines disposent pour l'instant de systèmes d'exploitation et de langages de programmation encore trop primitifs pour utiliser leur puissance globale et attirer les programmeurs.

Une autre solution pour réaliser des traitements parallèles consiste à utiliser des architectures massivement parallèles composées de processeurs très simples, dites à grains fins. La puissance de ces machines qui dépend directement du nombre d'éléments de base ( $10^4$  ou plus) est énorme. Les progrès technologiques d'intégration sur silicium permettent de réaliser des circuits comprenant de plus en plus de processeurs élémentaires, et par conséquent, donnent à cette approche un crédit certain. Le problème essentiel de ces architectures massivement parallèles réside dans la réalisation des communications entre processeurs. Fort heureusement, en général, les graphes de connexion ne sont jamais complets et il en découle, en utilisant des placements efficaces, des communications souvent locales (seulement quelques nœuds à traverser). L'utilisation de bus de communication est en contradiction totale avec la structure à processeurs les plus simples possibles, car ils occuperaient une surface trop grande par rapport à la surface des



processeurs et ils nécessiteraient l'ajout d'arbitres et de protocoles complexes. Pour les raisons contraires, les communications par messages semblent mieux adaptées et plus efficaces.

### **Notre proposition**

L'architecture massivement parallèle que nous proposons est basée sur un réseau de cellules asynchrones organisées en matrice. Les cellules de base sont toutes identiques, et elles peuvent effectuer des traitements simples, et spécifiques pour chaque application, de manière à obtenir d'excellentes performances. L'asynchronisme des cellules réside dans le fait que l'architecture est de type MIMD, et donc chaque cellule travaille indépendamment des autres. Grâce à cette structure cellulaire, il est possible de traiter des applications qui manquent de régularité pour être traitées par des réseaux systoliques [KUN82]. Le mécanisme de communication a été choisi de manière à permettre, en plus des communications locales, des communications éloignées tout en gardant une bonne efficacité. Une cellule peut envoyer un message à n'importe quelle autre : l'acheminement est réalisé par les cellules elles-mêmes, en plus de leur tâche spécifique de traitement. Chaque cellule est donc organisée en deux parties distinctes : une partie traitement qui réalise les fonctions nécessitées par l'application, et une partie routage qui permet les communications éloignées en recopiant les messages présents dans ses tampons d'entrée vers des tampons de sortie.

Cette étude entre dans le cadre d'un vaste projet. Elle a été commencée il y a quelques années par J.P. Bernard, qui a travaillé sur l'aspect général du réseau [BER85]. Pour valider cette architecture, nous avons développé des machines spécifiques pour plusieurs applications. La machine pour la simulation logique a été étudiée et simulée par Y. Ansade, et la réalisation a été menée à bien par R. Cornu-Emieux et P. Objois [COR88], [OBJ88]. Actuellement, quelques autres applications sont en cours d'étude au sein de notre équipe de recherche.

Mon travail a consisté à définir et réaliser une machine de traitements d'images, et plus particulièrement de reconstruction

[LAT88b]. Ce problème est tout à fait bien adapté au traitement cellulaire, et l'étude menée montre l'efficacité de l'architecture que nous proposons.

## **La reconstruction d'images**

La reconstruction d'images permet d'étudier la structure interne d'objets et de corps vivants sans avoir à les détériorer. Le but est donc d'arriver à connaître cette structure en reconstruisant des images représentant des sections de l'objet, à partir d'informations caractérisant celui-ci. Les seules informations qu'il est possible d'obtenir sur un corps, sans pour cela le détruire, sont des images de transmission, appelées projections, générées par des sources de radiation externes. La connaissance de la structure interne d'objets intéresse beaucoup de secteurs d'activité et de recherche, en particulier la médecine avec l'imagerie médicale [PER82].

Les applications de la reconstruction d'images sont très nombreuses et diffèrent en fonction du type d'objets à étudier, et du type de radiations utilisées pour les explorer. Outre le domaine médical, le procédé est courant en microscopie électronique afin d'étudier les structures moléculaires biologiques (par transmission d'électrons ou de rayons X), en astronomie et radioastronomie pour l'étude des corps célestes, étoiles, plasmas, nuages et couronne solaire (par émission de lumière et signaux radio), et en géophysique afin d'examiner la structure terrestre [FRI87]. Il y a d'autres applications possibles, notamment en sonographie pour analyser morphologiquement et sédimentologiquement les fonds marins (l'image est recueillie à partir d'un navire et obtenue par échos latéraux provoqués par le balayage transversal d'un faisceau d'émission), en interférométrie holographique (méthode de photographie en relief utilisant les interférences produites par la superposition de faisceaux laser), et optique (étude de la propagation de la lumière) [BAT83], [HER79]. Un nouveau champ d'application qui est de plus en plus important est le test de pièces en industrie, cela permet de juger de la bonne ou mauvaise qualité de leur composition sans avoir à les détériorer.

### L'imagerie médicale

La médecine est cependant le secteur le plus important pour la reconstruction d'images. Depuis de nombreuses années, nous avons cherché à connaître la structure du corps humain afin de mieux comprendre l'anatomie et la physiologie, et déceler les éventuelles anomalies pour traiter les patients le plus efficacement possible.

L'imagerie médicale a débuté avec la radiologie traditionnelle. Elle consiste à examiner un objet éclairé par transparence par un faisceau de rayons X, la visualisation se fait par fluorescence d'un écran (radioscopie) ou par fixation de l'image sur un film photographique (radiographie). Les limites de cette technique (l'image obtenue correspond à la superposition de tous les plans de l'objet, et il est par conséquent très difficile de reconnaître les différentes structures) ont poussé les chercheurs et médecins à développer d'autres procédés plus efficaces, pour avoir des informations plus précises sur les parties cachées du corps humain, en reconstituant des images représentant les différentes densités des tissus (en général, les tissus malades ont une densité différente de celle des tissus sains). L'étape suivante de l'imagerie médicale est la tomographie (*tomos* veut dire tranche en grec) ; il s'agit d'isoler une tranche de l'objet à examiner. La tomographie diffère de la radiologie par le fait que le générateur de rayons X et le film sensible subissent des mouvements de déplacement par rapport à l'organe à analyser, tels que les images contenues dans un plan privilégié (où se situe l'axe de ce déplacement) demeurent seules lisibles. Les objets placés en dehors de ce plan privilégié sont pratiquement effacés. La tomographie donne ainsi des images d'une fine tranche d'un organe. La tomographie numérique (Computerized Tomography) est le dernier stade du développement de l'imagerie médicale. Le film est remplacé par des détecteurs très sensibles, et les renseignements qu'ils récupèrent sont traités à l'aide d'un ordinateur, qui reconstruit la tranche de l'objet observé par une méthode mathématique. Ce procédé de reconstruction informatique a été inventé par G. N. Hounsfield en 1973, et la dénomination de scanner apparaît pour la première fois.

L'imagerie est une excellente aide au diagnostic médical, car elle permet de détecter d'éventuelles tumeurs, et de les localiser très précisément pour les traiter ou envisager leur ablation grâce à la chirurgie [AXE83], [BLO83]. Les principales techniques d'imagerie utilisées sont la tomographie par transmission de rayons X (radiologie) ou rayons gamma (médecine nucléaire), et la tomographie par résonance nucléaire magnétique (RNM) [AMA85]. La tomographie par rayons X est aussi appelée tomodensitométrie ou scanographie. La résonance nucléaire magnétique est l'étude de la résonance nucléaire de certains atomes lorsqu'une fréquence de radiation électromagnétique et une intensité de champ magnétique sont appliquées (c'est l'étude des interactions d'impulsions électromagnétiques sur les noyaux d'hydrogène) [HIN83]. L'échographie est un autre procédé basé sur l'émanation d'ultrasons provenant des tissus organiques [GRE83].

Dans la suite du travail, nous considérons uniquement le problème de la tomographie numérique par rayons X. C'est la partie la plus importante de la reconstruction d'images et elle représente bien le problème. Les autres applications ne présentent pas de différences fondamentales en ce qui concerne l'aspect mathématique du problème. Le travail exposé peut se généraliser aux autres applications car en fait il y a seulement des changements dans le type de radiations utilisées pour obtenir les projections.

## **Organisation du document**

Le but de mon travail a donc consisté à analyser les principales fonctions de la reconstruction d'images et en particulier celles qui sont les plus gourmandes en quantité de calcul, puis à proposer une implémentation cellulaire. Le document est organisé comme suit.

Le chapitre 1 pose clairement le problème de la reconstruction d'images et fait une revue des méthodes existantes, afin de présenter une ouverture vers une machine parallèle.

Le chapitre 2 décrit précisément l'architecture du réseau cellulaire pour la reconstruction d'images. La fonctionnalité de la cellule est donnée, et les problèmes de communication avec le monde extérieur sont étudiés. En particulier est donnée une évaluation de la précision nécessaire pour coder les informations et pour les calculer.

Le chapitre 3 présente les simulations fonctionnelles de la machine de reconstruction basée sur le réseau cellulaire. Les résultats montrent l'influence des divers paramètres (en particulier la précision) lors du traitement, et permettent d'évaluer l'activité moyenne des cellules.

Dans le chapitre 4, est exposée la structure interne de la cellule de base du réseau, ainsi que l'implantation de cette architecture et la réalisation du circuit intégré. L'interface avec un ordinateur hôte est étudiée, afin de développer une maquette de démonstration de notre machine de reconstruction d'images.

Pour conclure, nous faisons le point sur le travail effectué, et dressons l'avenir du projet.

On trouvera en annexe 1 des informations complémentaires sur l'aspect mathématique de la reconstruction d'images, et en annexe 2 une description des systèmes d'acquisition et de reconstruction. L'annexe 3 présente quelques programmes et résultats des simulations fonctionnelles du réseau cellulaire. Quant à l'annexe 4, elle présente la structure interne de la cellule (simulations comportementales et logiques), et la réalisation du circuit (dessins des masques et plan de montage).





## ***Chapitre 1***

---

# La reconstruction d'images





|                                                                  |           |
|------------------------------------------------------------------|-----------|
| <b>1.1. Présentation du problème .....</b>                       | <b>39</b> |
| 1.1.1. Introduction .....                                        | 39        |
| 1.1.2. L'aspect physique .....                                   | 39        |
| 1.1.3. Position détaillée du problème .....                      | 41        |
| 1.1.3.1. Génération des projections .....                        | 41        |
| 1.1.3.2. L'espace de reconstruction .....                        | 42        |
| 1.1.3.3. Formulation continue du problème .....                  | 44        |
| 1.1.3.4. Formulation discrète du problème .....                  | 45        |
| <b>1.2. Les méthodes de reconstruction .....</b>                 | <b>46</b> |
| 1.2.1. Historique .....                                          | 46        |
| 1.2.2. Les méthodes de reconstruction .....                      | 48        |
| 1.2.2.1. Classification .....                                    | 48        |
| 1.2.2.2. Les méthodes par sommation .....                        | 49        |
| 1.2.2.3. Les méthodes utilisant les transformées de Fourier..... | 51        |
| 1.2.2.4. Les méthodes purement analytiques.....                  | 51        |
| 1.2.2.5. Les méthodes par approximations.....                    | 53        |
| 1.2.2.5.1. Les méthodes ART.....                                 | 54        |
| 1.2.2.5.2. Les méthodes SIRT.....                                | 56        |
| 1.2.3. Discussion .....                                          | 56        |
| 1.2.3.1. Les méthodes les plus efficaces .....                   | 56        |
| 1.2.3.2. Comparaison .....                                       | 57        |
| <b>1.3. Vers une machine parallèle .....</b>                     | <b>58</b> |
| 1.3.1. Pourquoi une machine parallèle? .....                     | 58        |
| 1.3.2. Les machines existantes.....                              | 59        |
| 1.3.2.1. Les machines pipelines .....                            | 59        |
| 1.3.2.2. Les machines parallèles à mémoire partagée .....        | 59        |
| 1.3.2.3. Les machines parallèles à mémoire distribuée.....       | 60        |
| 1.3.3. Le réseau cellulaire .....                                | 62        |

## Chapitre 1 : La reconstruction d'images

## 1.1. PRESENTATION DU PROBLEME

### 1.1.1. Introduction

Avant de définir une implantation matérielle très parallèle de traitement, il est important de bien poser le problème et exposer les algorithmes. Nous allons donc commencer par expliciter le problème de la reconstruction d'images, en présentant la tomographie par rayons X, et en précisant les notions essentielles [AMS88].

L'objectif de ces méthodes est de construire la carte représentative de la distribution de densité interne d'un objet, à partir d'un nombre fini de ses projections, obtenues par irradiation de celui-ci par des rayons X.

### 1.1.2. L'aspect physique

Les rayons X traversant un objet, s'atténuent en suivant une loi bien connue de la physique des radiations : la loi d'absorption. Elle s'exprime ainsi : lorsqu'un faisceau monochromatique (une seule longueur d'onde ou un seul niveau d'énergie) de rayons X traverse un objet de densité uniforme, il est atténué suivant la relation exponentielle :

$$I_t = I_0 e^{-\mu l} , \quad [1]$$

où  $I_0$  est l'intensité incidente du faisceau,  $I_t$  l'intensité transmise,  $l$  la longueur du chemin du faisceau à travers l'objet, et  $\mu$  le coefficient d'absorption de l'objet ( $\mu$  est souvent appelé coefficient d'atténuation linéaire en tomographie). Ce coefficient est déterminé par la densité physique et la composition atomique de l'objet, et il est indépendant de l'énergie du faisceau de rayons X.

Habituellement, la densité de l'objet sur le chemin du faisceau n'est pas uniforme. Mais alors il est possible de discrétiser le problème en découpant le chemin du faisceau de longueur totale  $L$  en plusieurs

segments de longueurs différentes, sur lesquels les structures sont homogènes. L'atténuation totale se cumule, et nous pouvons dire que :

$$I_t = I_0 e^{-\sum_L \mu_i l_i}, \quad [2]$$

où  $l_i$  est la longueur du segment  $i$ , et  $\mu_i$  le coefficient d'atténuation sur ce segment.

En découpant la longueur totale du faisceau  $L$  en des segments de même longueur ( $l_i = l$ , pour tout segment), et en s'assurant toujours que les coefficients d'atténuation  $\mu_i$  sont constants sur ces petits segments, alors l'équation [2] peut s'écrire :

$$\sum_L \mu_i = \frac{1}{l} \ln \frac{I_0}{I_t}, \quad [3]$$

où  $\ln \frac{I_0}{I_t}$ , le logarithme népérien de  $\frac{I_0}{I_t}$  représente l'atténuation du faisceau.

L'atténuation totale, qui est la somme des coefficients d'atténuation sur le chemin du faisceau, peut être facilement calculé si  $I_0$ ,  $I_t$  et  $l$  sont connus.  $l$  est choisie arbitrairement petit, de manière à ce que la densité soit uniforme sur les segments à l'intérieur de l'objet. En tomographie,  $I_t$  et  $I_0$  sont mesurées pour beaucoup d'angles différents par des détecteurs aux rayons X ; ces mesures donnent des équations multiples ayant les mêmes inconnues  $\mu_i$ , qui peuvent être résolues par diverses approches mathématiques.

Le résultat est la carte des coefficients d'atténuation de l'objet, qui peut être représentée comme une image, en assignant les valeurs de brillance (niveaux de gris) des pixels, proportionnellement aux coefficients d'atténuation en chaque point de l'objet.

Un émetteur classique de rayons X produit un faisceau polychromatique, ayant une distribution de rayons X d'énergies différentes (ou différentes longueurs d'onde). Les basses énergies

(grandes longueurs d'onde) sont absorbées plus rapidement que les hautes énergies (petites longueurs d'onde) ; celles-ci pénètrent donc plus loin dans les structures de l'objet irradié. Ce phénomène augmente la moyenne énergétique du faisceau sortant, ainsi l'objet apparaît moins dense, que s'il avait été traité avec un faisceau monochromatique d'énergie, égale à la moyenne énergétique du faisceau polychromatique incident. Ce problème de durcissement du faisceau ("beam hardening") est habituellement corrigé, par homogénéisation des radiations incidentes au moyen d'un filtre, ainsi nous nous ramenons à la loi d'absorption d'un faisceau monochromatique [ROB82], [TRE86].

### **1.1.3. Position détaillée du problème**

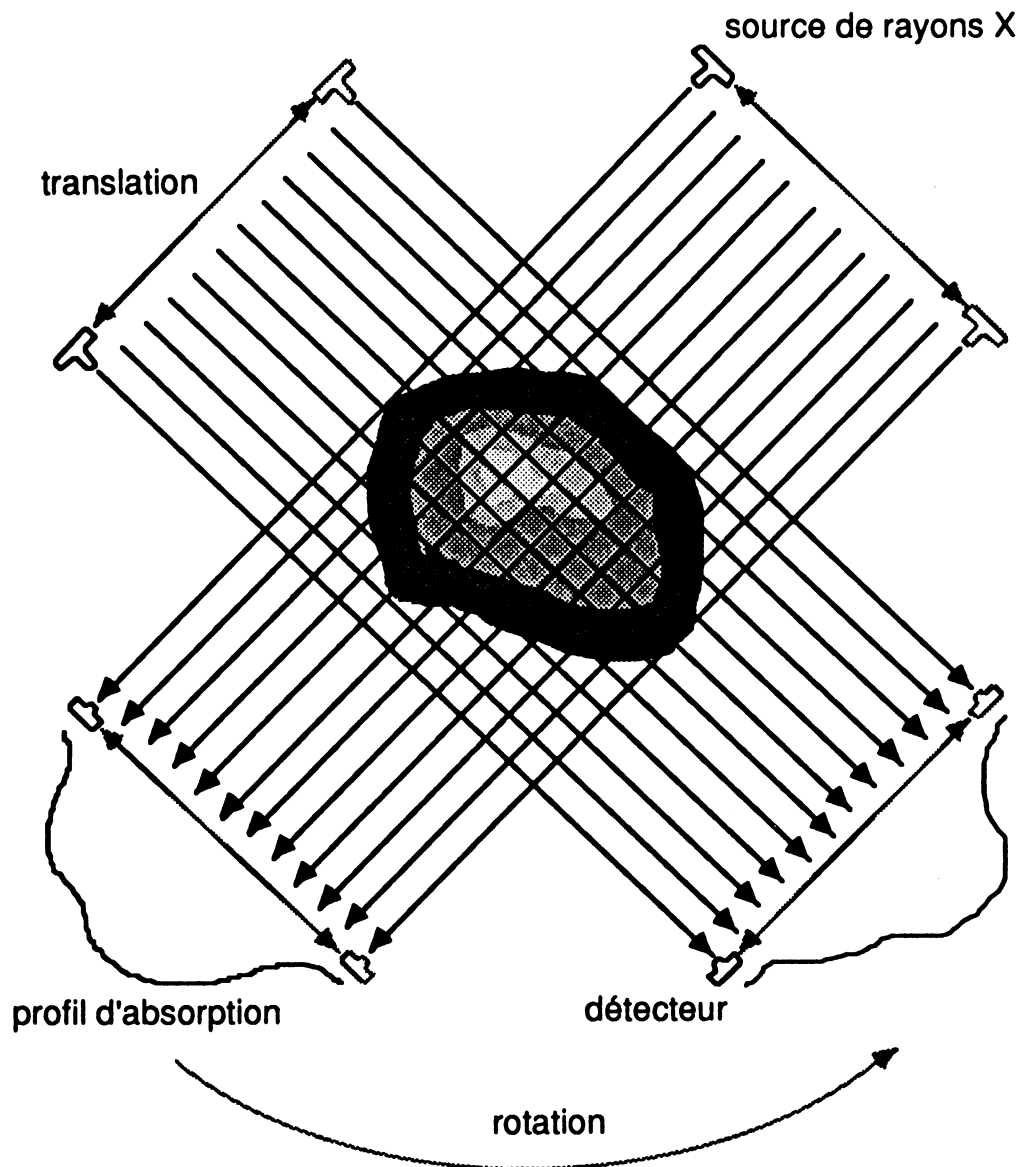
#### **1.1.3.1. Génération des projections**

La génération des projections d'un objet est réalisée à l'aide de faisceaux de rayons X traversant l'objet, et de détecteurs les collectant après leur passage. Ces mesures divisées par les intensités incidentes des rayons, permettent d'obtenir les profils d'absorption.

La figure 1.1. décrit une machine à géométrie parallèle, où les mesures sont acquises grâce à une seule source de rayons et un seul détecteur, lesquels subissent un mouvement de translation, et de rotation afin d'obtenir l'ensemble des informations.

Les mesures sont donc effectuées tout autour de l'objet sur  $180^\circ$ , en faisant tourner le système d'émetteur et récepteur, elles forment un ensemble pour chaque angle qui s'appelle projection.

La valeur de la projection à chaque point est proportionnelle à l'intensité du rayon X transmise le long du chemin, entre la source de rayons X, et le détecteur correspondant. L'intensité lue est convertie en atténuation, en prenant le logarithme népérien, du rapport de l'intensité du rayon incident, par l'intensité du rayon transmis.

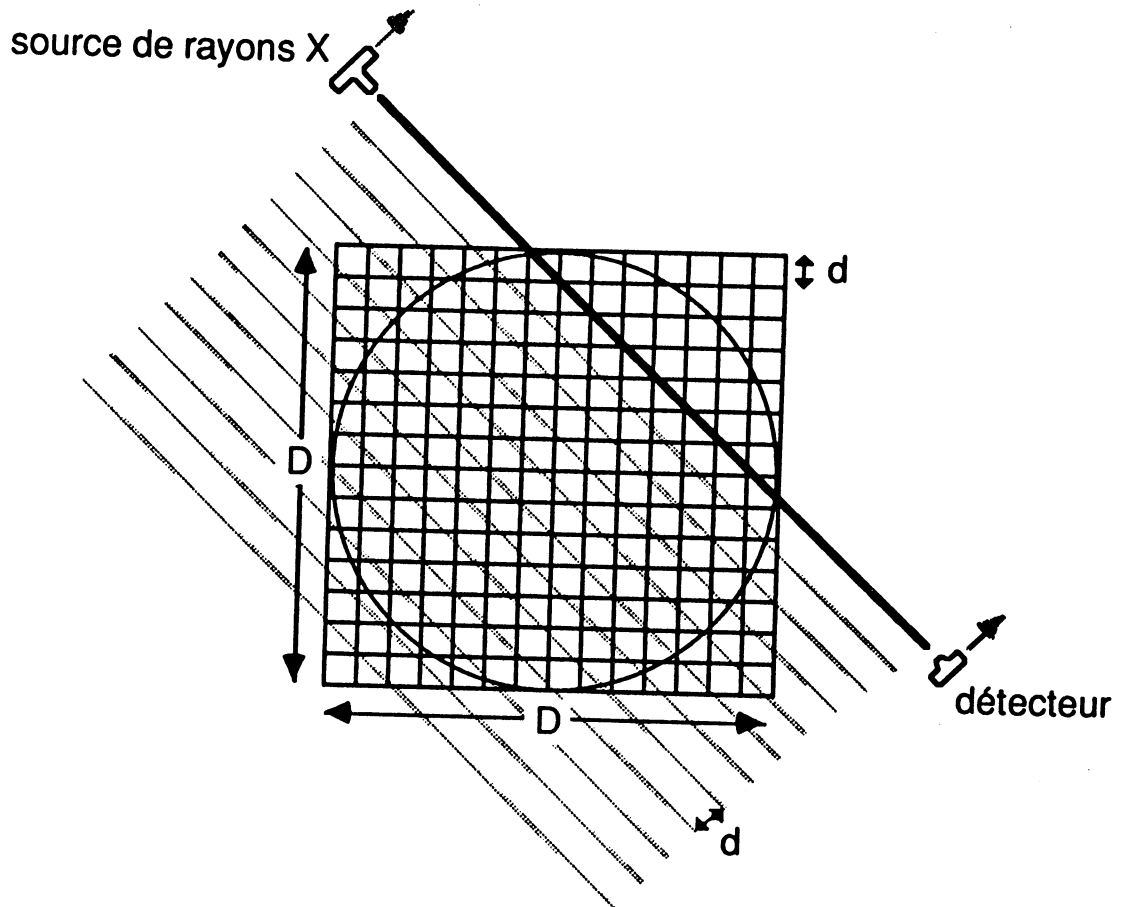


**Figure 1.1.** Principe de génération des projections

### **1.1.3.2. L'espace de reconstruction**

Avant d'introduire les méthodes de reconstruction, nous définissons l'espace de reconstruction par une région plane dans l'espace irradié, contenant les structures que nous souhaitons analyser. Comme le montre la figure 1.2., cette région plane est bornée par un carré, qui est partitionné en une matrice de  $N \times N$  petites régions non recouvrantes, habituellement appelées pixels, chacun d'eux ayant

un coefficient d'atténuation uniforme aux rayons X (ou densité constante).



**Figure 1.2.** Découpage de l'espace de reconstruction

Le nombre et la taille des pixels est basé sur la résolution des projections et la résolution désirée de l'image. Si nous souhaitons reconstruire une région de diamètre  $D$ , à partir de  $M$  projections contenant chacune d'elle  $\frac{D}{d}$  rayons parallèles, cela amène à travailler avec une image de taille  $N \times N$  avec  $N$  défini par le rapport  $\frac{D}{d}$  et des pixels de taille  $d \times d$ . La qualité de l'image augmente avec le nombre de projections  $M$ , mais nous pouvons dire qu'il faut théoriquement au moins  $\frac{\pi}{2} N = M$  projections, pour reconstruire une image de  $N \times N$  pixels [ROB82].



Il est possible d'afficher la matrice des coefficients d'atténuation comme une grille de niveaux de brillance (ou niveaux de gris), pour visualiser une image des structures de la région de reconstruction.

### 1.1.3.3. Formulation continue du problème

Le problème de la reconstruction peut se décrire sous forme d'équation intégrale telle que:

$$P(r,\varnothing) = \int_{-\infty}^{+\infty} A(x,y) dl, \quad [4]$$

où  $P(r,\varnothing)$  est la valeur du point projeté de la projection d'angle  $\varnothing$  relativement à l'axe  $x$  du système de coordonnées cartésiennes  $(O,x,y)$  et  $r$  la distance du rayon à l'origine  $O$ . Cette fonction  $P$  à deux variables ( $r$  et  $\varnothing$ ) peut être représentée comme une image habituellement appelée sinogramme.  $A(x,y)$  est la fonction donnant les coefficients d'atténuation (ou densités) à chaque point  $(x,y)$ , et  $dl$  est la longueur différentielle du chemin d'un rayon quelconque.

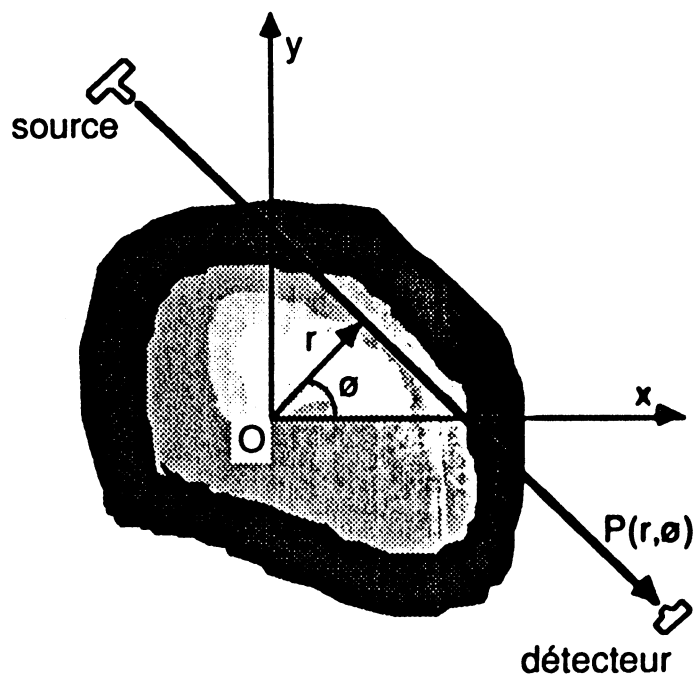


Figure 1.3. Repérage des projections

Cette équation forme la base de toutes les approches de reconstruction d'images, c'est la formule mathématique de projection.

Le problème est d'inverser cette équation, afin de déterminer la fonction  $A(x,y)$ .

### 1.1.3.4. Formulation discrète du problème

Le problème peut être posé sous sa forme discrète plutôt que sous sa forme continue. En considérant la région de reconstruction discrétisée sous forme d'un tableau de  $n$  pixels ( $n = N \times N$ ), il est clair que les faisceaux de rayons  $X$  n'intersectent pas tous les pixels qu'ils traversent de la même façon, ceci veut dire que la contribution d'un rayon  $X$  aux pixels traversés sera plus ou moins importante.

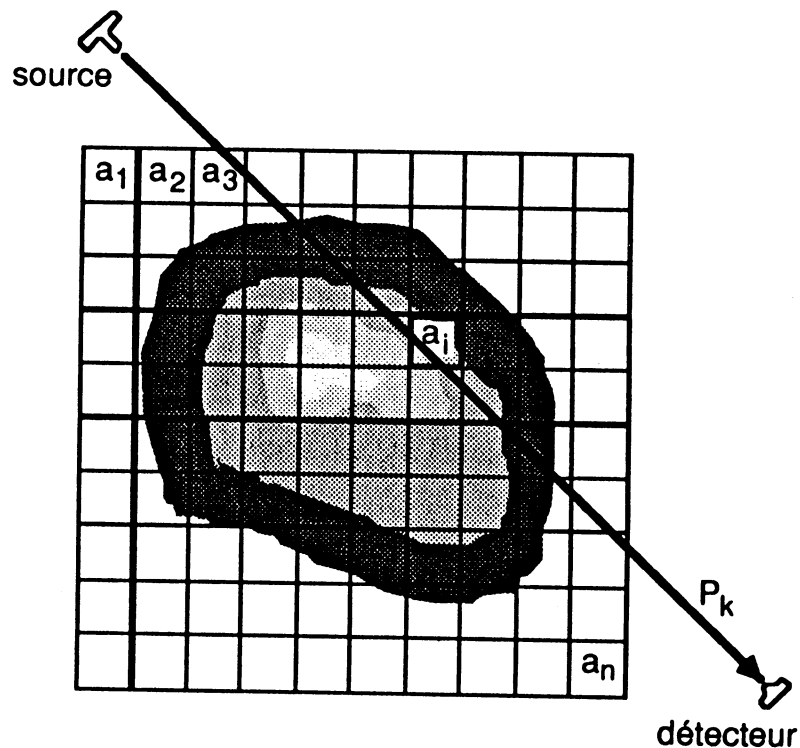


Figure 1.4. Partitionnement de la section de l'objet

Si  $m$  est le nombre total de mesures ( $m = M \times \text{nombre de rayons parallèles par projection}$ ), la forme discrète de la formule de projection s'écrit ainsi :

$$P_k = \sum_{i=1}^{i=n} w_{ik} a_i \quad \text{pour } k = 1, \dots, m, \quad [5]$$

où  $a_i$  est le coefficient d'atténuation du pixel  $i$ ,  $P_k$  la valeur du point projeté correspondant au rayon  $k$ , et  $w_{ik}$  un facteur pondérateur qui caractérise la contribution du rayon  $k$  au pixel  $i$  ( $w_{ik}$  est nul si le rayon  $k$  ne traverse pas le pixel  $i$ ).

Le problème est donc de résoudre le système linéaire [5] de  $m$  équations à  $n$  inconnues.

Une technique évidente de résolution d'un tel système consiste à utiliser les méthodes classiques d'analyse numérique conduisant à inverser directement la matrice  $W$  des coefficients ( $w_{ik}$ ). Le problème en terme de vecteurs et matrice, se pose de la façon suivante :  $P = W * A$ , où  $P$  est le vecteur des projections ( $P_k$ ), et  $A$  le vecteur des densités des pixels ( $a_i$ ). La solution évidente de cette équation matricielle est  $A = W^{-1} * P$ , et le problème initial est équivalent à l'inversion de  $W$ . Or, nous nous apercevons très vite, que cette solution est à écarter, du fait de la taille des données à traiter. Dans le cas pratique, où nous voulons reconstruire une image de  $n = 10^6$  pixels, à partir d'environ  $m = 10^6$  points projetés, la matrice  $W$  a  $10^{12}$  éléments. Sachant que le nombre d'opérations nécessaires, pour inverser une matrice carrée de  $q^2$  éléments, est de l'ordre de  $q^3$  (par la méthode de triangularisation de Gauss ou diagonalisation de Jordan), il faudrait  $(10^6)^3$  opérations dans le cas présent ; et en supposant qu'une opération prenne une microseconde, il faudrait 317 siècles !!! [SCU78].

## 1.2. LES METHODES DE RECONSTRUCTION

### 1.2.1. Historique

Les méthodes de reconstruction ont intéressé les mathématiciens dès le début du siècle, le problème a alors été analytiquement défini et résolu par l'autrichien J. Radon. Bien que les méthodes actuellement connues ne sont pas toutes basées sur ce travail, il est important de l'expliquer brièvement.

J. Radon, a prouvé qu'un objet bidimensionnel ou tridimensionnel peut être reconstruit, à partir de l'ensemble infini de ses projections [RAD17]. L'équation intégrale  $P(r, \vartheta) = \int_{-\infty}^{+\infty} A(x, y) dl$  [4], illustrée au paragraphe 1.1.3.3., a été résolue par Radon, et la solution exprimée en coordonnées polaires est :

$$A(t, \beta) = \frac{1}{2\pi^2} \int_0^\pi \int_{-\infty}^{+\infty} -\frac{1}{d} \frac{\partial P(r, \vartheta)}{\partial r} dr d\vartheta, \quad [6]$$

où  $d$  est la distance perpendiculaire du point  $M$  au rayon défini par  $r$  et  $\vartheta$  ( $d = r - t \cos(\vartheta - \beta)$ ),  $M$  étant un point quelconque de coordonnées  $(t, \beta)$ .

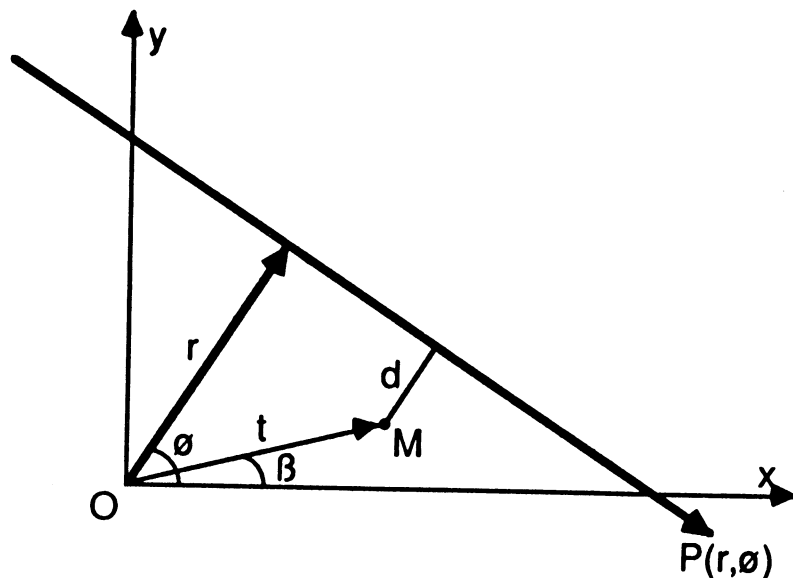


Figure 1.5. Illustration des notations

Dans la littérature, l'ensemble des projections défini par la relation intégrale [4] (l'équation à résoudre) est appelée "transformée de Radon", et la fonction densité définie par la relation [6] (la solution) "transformée inverse de Radon" [HOR78], [HOR79].

Ce résultat a été successivement découvert par les mathématiciens, les radioastronomes, les "microscopistes", et les radiologistes.

Toutes les méthodes de reconstruction ne s'appliquent pas à l'ensemble infini des projections d'un objet (comme dans la solution de Radon), mais seulement à un nombre limité de projections. Les reconstructions, qu'il est possible de faire, sont seulement des estimations de la structure de l'objet.

L'exploitation des techniques de reconstruction a dû attendre l'arrivée des ordinateurs modernes, à cause du nombre très important de calculs nécessaires pour obtenir une reconstruction.

### **1.2.2. Les méthodes de reconstruction**

#### **1.2.2.1. Classification**

Les méthodes de reconstruction peuvent se classer en deux grands groupes : les méthodes analytiques, et les méthodes algébriques. Elles sont liées au type de formulation du problème, qu'elles cherchent à résoudre. Les premières ont leur base dans l'équation intégrale [4] de la formulation continue (approche de Radon), et les secondes cherchent à résoudre le système linéaire [5] de la formulation discrète [AMA85], [HER80a], [HER80b].

Une autre classification en quatre catégories, dont les trois premières sont analytiques, est couramment utilisée :

- Les méthodes par sommation,
- les méthodes utilisant les transformées de Fourier,
- les méthodes purement analytiques,
- et les méthodes par approximations successives [JAM85].

Une présentation mathématique plus détaillée, qu'elle ne l'est dans ce chapitre, des méthodes de reconstruction, est donnée en annexe 1. Une description des systèmes d'acquisition et de reconstruction et leurs performances sont présentées en annexe 2.

### 1.2.2.2. Les méthodes par sommation

Ces méthodes ont une base intuitive évidente. Leur principe est très élémentaire, la technique consiste à estimer les densités de chaque pixel en additionnant ensemble toutes les valeurs des projections, dont les rayons correspondants passent dans le pixel.

La tomographie dite "classique" (l'image est recueillie sur un support sensible) est une méthode par sommation. Une plaque photographique est placée parallèlement à la section d'un côté de l'objet et une source de rayons X de l'autre côté. La source et la plaque sont déplacées parallèlement à la section et en sens opposé, de manière à ce que chaque point de la section soit projeté sur un même point de la plaque (les points hors de la section sont également projetés, mais de façon plus aléatoire). Nous obtenons ainsi la distribution de densité de la section avec un certain flou de l'image, dû à l'ensemble des points projetés se trouvant hors de la section [TRE86].

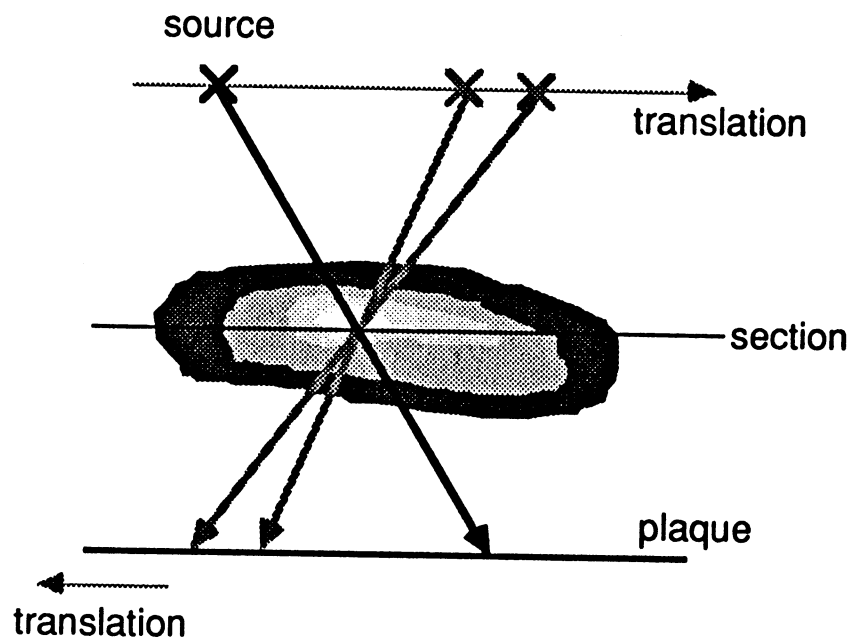


Figure 1.6. Principe de la tomographie classique

Pour revenir à la tomographie numérique, les valeurs des projections sont distribuées uniformément aux pixels traversés par les rayons. Chaque pixel de l'image reconstruite est calculé, en effectuant la somme de tous les points projetés le traversant, cette opération s'appelle un épandage. De même qu'en tomographie "classique", illustrée auparavant, il y a un phénomène de flou de l'image, dû au fait que les points hors d'un point pixel considéré interviennent dans la somme finale de la valeur du pixel.

Nous pouvons formuler discrètement l'opération d'épandage par la relation :

$$a_i = \sum_{k=1}^{k=m} w_{ik} P_k \quad \text{pour } i = 1, \dots, n, \quad [7]$$

en reprenant les notations précédentes pour les valeurs des pixels ( $a_i$ ), les valeurs des points projetés ( $P_k$ ), et les facteurs pondérateurs ( $w_{ik}$ ). Ces poids ( $w_{ik}$ ) étant les mêmes que ceux de la formulation discrète du problème, il est clair que les méthodes utilisant cette technique de sommation (un seul épandage des projections) est très approximative, puisque les valeurs des pixels ainsi calculées ne peuvent vérifier parfaitement le système linéaire [5], que nous avons à résoudre initialement :

$$P_k = \sum_{i=1}^{i=n} w_{ik} a_i \quad \text{pour } k = 1, \dots, m \quad [5].$$

L'effet de flou des images obtenues grâce à un seul épandage peut être faiblement corrigé par application d'un filtre. Le contraste peut être amélioré en soustrayant à l'ensemble des pixels une constante, calculée de façon à ce que la moyenne de densité de l'image reconstruite soit identique à la moyenne "réelle" de l'objet (qui peut être calculé grâce à l'ensemble des projections). Quel que soit le type de traitement utilisé pour l'amélioration de l'image, les résultats obtenus par ces méthodes sont loin d'être satisfaisants.

### **1.2.2.3. Les méthodes utilisant les transformées de Fourier**

Ces techniques découlent d'un théorème bien connu dans le milieu des instrumentistes en imagerie médicale : le théorème Coupe-Projection.

Le principe de ces méthodes consiste à transformer les projections dans l'espace de Fourier, afin d'obtenir quelques valeurs de transformée de Fourier de l'image complète ; les autres valeurs sont estimées par interpolation, et la reconstruction est obtenue en prenant la transformée inverse de l'image [LEW83].

Ces méthodes sont coûteuses du fait de leurs traitements bidimensionnels (interpolation, transformée inverse), et sont par conséquent peu utilisées.

### **1.2.2.4. Les méthodes purement analytiques**

Comme nous l'avons vu auparavant, la relation entre l'image et ses projections s'exprime en un ensemble d'équations intégrales, qui ont été résolues analytiquement. Les pixels sont calculés directement grâce à la solution analytique de Radon [SZU86].

Les méthodes diffèrent dans la manière d'évaluer cette solution à partir des équations intégrales.

La méthode, dite par convolution, peut être considérée comme une implémentation de Radon, elle est aussi connue sous le nom d'épandage des projections filtrées (Filtered Back Projection : FBP). Comme son nom l'indique, elle ressemble à la méthode du simple épandage, mais avec une différence significative, qui consiste à faire subir un filtrage aux projections avant d'effectuer l'opération d'épandage. Ce pré-filtrage est réalisé par convolution des projections avec une certaine fonction pondératrice.

L'épandage des projections filtrées est la méthode analytique la plus performante, car elle n'inclut pas de traitement bidimensionnel outre l'épandage. Une autre qualité de cette méthode est qu'elle permet de ne reconstruire qu'une partie de l'image. En plus de cette modularité de la reconstruction, sa mise en œuvre est simple : il est



possible d'effectuer le filtrage d'une projection, pendant l'épandage de la précédente, et pendant l'acquisition de la suivante (mode pipeline).

La formulation continue de la technique par épandage des projections filtrées peut s'exprimer par :

$$A(x,y) = \int_0^{\pi} P'(r,\vartheta) d\vartheta , \quad [8]$$

où  $P'(r,\vartheta)$  sont les valeurs filtrées des projections mesurées  $P(r,\vartheta)$ , et  $A(x,y)$  la fonction densité de l'objet.

La formulation discrète se représente par le système linéaire :

$$a_i = \sum_{k=1}^{k=m} w_{ik} P'_k \quad \text{pour } i = 1, \dots, n , \quad [9]$$

où  $a_i$  est la valeur du pixel  $i$ ,  $P'_k$  la valeur du point projeté filtré correspondant au rayon  $k$ , et  $w_{ik}$  un facteur pondérateur caractérisant la contribution du rayon  $k$  au pixel  $i$ .

L'opération de filtrage peut être vu comme un système linéaire. La mission d'un filtre est de modifier un signal d'entrée multifréquence, en laissant passer certaines fréquences, et en atténuant d'autres. Une méthode d'épandage des projections filtrées très connue est la méthode par convolution. En nommant les points mesurés du vecteur de projection d'angle  $\vartheta$  par  $P_{\vartheta}(k)$ ,  $k$  variant de 1 à  $K$ , qui est le nombre de points par projection (en reprenant les notations précédentes  $K = \frac{m}{M}$ ), le filtre qu'elle utilise est décrit par le système :

$$P'_{\vartheta}(k) = \frac{P_{\vartheta}(k)}{4} - \left( \frac{1}{\pi^2} \sum_{t \text{ impair}} \frac{P_{\vartheta}(k+t)}{t^2} \right) \quad \text{pour } k = 1, \dots, K , \quad [10]$$

où  $P'_{\vartheta}(k)$  est la valeur filtrée de la valeur de projection  $P_{\vartheta}(k)$ , et  $t$  un entier impair tel que  $1 \leq (k+t) \leq K$  [ROB82].

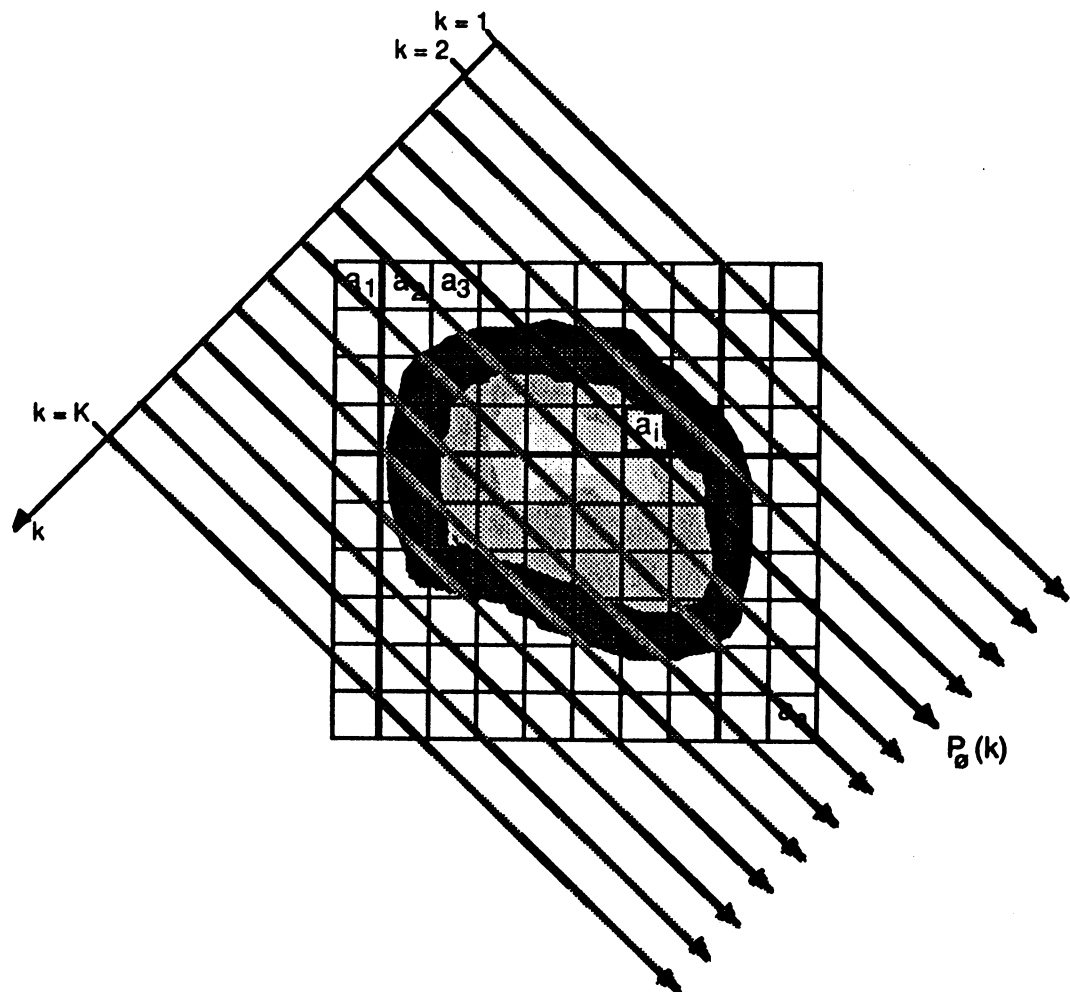


Figure 1.7. Illustration de la projection d'angle  $\theta$

Cette méthode est très efficace, et peut être implémentée facilement, parce que c'est la seule, qui en dehors de l'épandage, n'a pas de traitement bidimensionnel. En effet, l'opération de filtrage est un traitement unidimensionnel (les vecteurs de projections sont traités indépendamment les uns des autres), donc peu coûteux. Elle est la méthode analytique la plus utilisée. La qualité des images reconstruites repose sur la manière, dont est effectué l'épandage.

### 1.2.2.5. Les méthodes par approximations

Ces techniques sont purement algébriques et itératives. Le problème traité ici est celui de la résolution d'un système d'équations,

où les inconnues sont les valeurs des pixels, toutes les méthodes ont pour base la technique d'inversion de matrice de Kaczmarz [EGG81].

Ces méthodes se rapportent à la formulation discrète du problème, elles essaient de résoudre le système linéaire :

$$P_k = \sum_{i=1}^{i=n} w_{ik} a_i \quad \text{pour } k = 1, \dots, m. \quad [5]$$

Une classe d'algorithmes itératifs regroupe les techniques algébriques de reconstruction (Algebraic Reconstruction Techniques : ART). L'algorithme de base consiste à évaluer les valeurs des pixels, puis à les corriger de manière itérative, grâce à un facteur tenant compte de la différence entre les valeurs de projections mesurées et celles calculées. Il y a deux types de techniques ART : des méthodes additives (le facteur correcteur est une différence), et des méthodes multiplicatives (le facteur correcteur est un rapport) [GOR70], [GOR74b], [HER73a].

Les techniques de reconstruction itératives simultanées (Simultaneous Iterative Reconstruction Techniques : SIRT) forment une autre classe d'algorithmes, elles diffèrent des méthodes ART par le fait que toutes les équations sont traitées simultanément à chaque itération [HER76].

Les méthodes itératives sont très performantes, mais moins rapides que les solutions analytiques. Par contre, elles présentent des avantages dans certains cas d'application, notamment lorsque le nombre de projections est limité, ou lorsque les projections sont très bruitées.

### 1.2.2.5.1. Les méthodes ART

Nous partons d'une image initiale, puis par un traitement itératif, nous allons l'affiner ; ainsi de proche en proche, la solution calculée tend vers la solution réelle du système linéaire. Pour cela, à chaque itération, un seul angle de projection est pris en compte, et les valeurs des pixels sont modifiées de manière à ce que les valeurs mesurées de

cette projection correspondent aux valeurs calculées à partir de l'image précédant la mise à jour [CEN83], [GOR74a], [GOR75]. Le facteur de correction intervenant à chaque pas de l'algorithme détermine le type de méthode ART utilisée.

Il est prouvé mathématiquement que la solution converge vers la solution réelle du système. Le procédé est effectué un nombre entier de fois le nombre de projections (M), et le traitement est stoppé lorsque les variations de l'image deviennent négligeables, ou que l'ensemble des valeurs des projections calculées sous tous les angles possibles se rapproche beaucoup de l'ensemble des valeurs des projections mesurées. Alors nous sommes sûr d'avoir une image proche de la réalité, et que l'ensemble des pixels la formant vérifie avec une erreur très faible le système linéaire caractérisant le problème.

Pratiquement, le nombre d'itérations nécessaires pour obtenir une image de qualité satisfaisante est de l'ordre de  $(5 \cdot M)$ .

### La méthode additive

En prenant la convention de noter  $P_{\theta}(k)$  les valeurs des points projetés mesurés de la projection d'angle  $\theta$ ,  $R_{\theta}(k)$  les valeurs des points projetés calculés sous le même angle  $\theta$  ( $k$  pouvant varier de 1 à  $K$ ),  $a_i$  la valeur du pixel  $i$  ( $i$  pouvant varier de 1 à  $n$ ),  $N_{\theta}(k)$  le nombre de pixels traversés par le point projeté de valeur  $R_{\theta}(k)$ , et  $w_{ik}$  la contribution du rayon  $k$  au pixel  $i$ , la correction effectuée au cours d'une itération ( $q+1$ ) de cette méthode peut se décrire en fonction des données à l'itération précédente  $q$  par la relation :

$$a_i^{(q+1)} = a_i^{(q)} + \sum_{k=1}^{K} \left( w_{ik} \frac{P_{\theta}(k) - R_{\theta}^{(q)}(k)}{N_{\theta}(k)} \right) \quad \text{pour } i = 1, \dots, n$$

Le facteur de correction est ici la différence entre la valeur mesurée du point projeté et sa valeur calculée, pondérée par le nombre de pixels traversés par le rayon correspondant (pour la distribuer de manière uniforme à tous les pixels).

### 1.2.2.5.2. Les méthodes SIRT

Contrairement aux méthodes ART, où la modification des pixels de l'image à chaque itération ne tient compte que d'une projection, les méthodes SIRT effectuent, à chaque itération, une mise à jour des pixels en tenant compte de l'ensemble des projections de l'objet, d'où leur nom de techniques "simultanées". Ainsi à chaque pas de ces techniques, tous les points projetés de toutes les projections sont calculés afin de les comparer aux valeurs mesurées, puis en fonction de ces différences, les valeurs des pixels sont modifiées.

Les critères d'arrêt des algorithmes sont identiques à ceux des techniques ART. Le nombre d'itérations nécessaire, constaté de façon expérimentale est de l'ordre de 15.

#### La méthode additive

Le facteur de correction est ici proportionnel à la différence entre les valeurs calculées et mesurées.

En reprenant les mêmes conventions d'écriture, la relation récurrente définissant le traitement de l'itération d'ordre (q+1) en fonction de celle d'ordre q s'écrit :

$$a_i^{(q+1)} = a_i^q + \sum_{\theta=0}^{\theta=\pi} \left( \sum_{k=1}^{k=K} (w_{ik}) \frac{P_{\theta}(k) - R_{\theta}^q(k)}{N_{\theta}(k)} \right) \quad \text{pour } i = 1, \dots, n$$

### 1.2.3. Discussion

#### 1.2.3.1. Les méthodes les plus efficaces

Après avoir fait le tour des différents types de méthodes de reconstruction, nous pouvons dire que trois techniques sont supérieures en ce qui concerne l'efficacité et la facilité de mise en œuvre. Elles sont actuellement les plus utilisées, ces trois méthodes sont :

- l'épandage des projections filtrées FBP,
- la technique ART,
- la technique SIRT.

### 1.2.3.2. Comparaison

Pour comparer ces diverses méthodes, nous allons considérer les critères de temps de reconstruction, de précision de l'image, et de portabilité (utilisation dans des situations particulières) [HER73b], [GOR74a].

Les méthodes itératives ART et SIRT sont moins rapides, le temps de reconstruction par épandage des projections filtrées (FBP) est de l'ordre du temps nécessaire à un cycle ( $m$  itérations) des techniques ART, ou d'une itération des techniques SIRT (c'est à dire un épandage et une projection). Donc les techniques ART demandent environ 5 fois plus de temps de la méthode FBP, et les techniques SIRT environ 15 fois plus. Le nombre d'opérations (multiplications ou additions) de la méthode FBP est proportionnel à  $M(K^2+n)$ , avec  $M$  le nombre de projections,  $K$  le nombre de points par projection, et  $n = N^2$  le nombre de pixels de l'image. Le temps nécessaire au filtrage est de l'ordre de  $MK^2$  et celui de l'épandage de l'ordre de  $Mn$ . Le nombre d'opérations des méthodes ART ou SIRT est de l'ordre de  $Mnl$ , où  $l = \alpha m$  est le nombre de cycles ( $\alpha$  entier) pour les techniques ART, ou le nombre d'itérations pour les techniques SIRT.

Les trois méthodes sont équivalentes quant à la précision de l'image reconstruite avec l'hypothèse que nous disposons d'un nombre suffisant de projections ( $M \geq \frac{\pi}{2} N$ ).

Par contre, les techniques itératives ART et SIRT présentent des avantages dans certains cas, notamment lorsque le nombre de projections est restreint ( $M < \frac{\pi}{2} N$ ), et quand les projections sont incomplètes ou fortement bruitées [SMI73].

Finalement, nous pouvons dire que la méthode par épandage des projections filtrées est la "meilleure", en considérant les trois critères précédents de façon globale. Mais en fait, les trois méthodes sont complémentaires, et le choix d'une méthode plutôt qu'une autre est à effectuer en fonction de l'application à traiter.

### **1.3. VERS UNE MACHINE PARALLELE**

#### **1.3.1. Pourquoi une machine parallèle?**

Comme nous l'avons vu précédemment, la reconstruction d'images est difficile à implémenter parce que le nombre d'opérations est très important, et ceci d'autant plus que la précision des images est grande. Le traitement d'épandage demande de l'ordre de  $Mn$  opérations élémentaires ( $M$  : nombre de projections,  $n$  : nombre de pixels de l'image), soit environ  $10^9$  opérations pour reconstruire une image de  $(10^3 \times 10^3)$  pixels à l'aide de  $10^3$  projections.

Il ne faut pas perdre de vue que la principale application de la reconstruction d'images se trouve en imagerie médicale, et les traitements doivent être très rapides pour qu'ils soient effectués en temps réel (le temps de reconstruction doit être du même ordre de grandeur que le temps d'acquisition).

Bien que les microprocesseurs soient de plus en plus puissants et rapides, une architecture monoprocesseur de type SISD (Single Instruction, Single Data) ne peut assumer une telle charge de travail dans des délais acceptables. Il convient donc de diviser les tâches à effectuer afin de les répartir entre différents processeurs, d'où la nécessité d'une architecture multiprocesseur, ou parallèle.

Une architecture de type SIMD (Single Instruction, Multiple Data) ne peut être utilisée pour résoudre le problème, à cause l'irrégularité des données à traiter (les projections sont acquises sous un grand nombre de directions différentes). C'est ce qui fait de la reconstruction,

un traitement d'images très particulier, qui ne peut être résolu par les machines classiques de traitement d'images [OFF85].

### **1.3.2. Les machines existantes**

#### **1.3.2.1. Les machines pipelines**

Dans la plupart des machines récentes qui travaillent avec l'algorithme d'épandage des projections filtrées, certains traitements sont exécutés dans un mode pipeline. Nous avons vu que cette méthode comprenait deux phases : une phase de filtrage des vecteurs de projection, et une phase d'épandage.

Une architecture pipeline comprend trois processeurs, auxquels sont assignées trois tâches distinctes et indépendantes : le premier gère l'acquisition (le déplacement de l'instrumentation, et la réception de projections brutes), le deuxième réalise le filtrage des projections (en général, c'est un circuit spécifique implémentant un algorithme de transformation de Fourier rapide FFT), et le troisième épand les projections filtrées. Donc à un instant quelconque de la reconstruction, trois opérations sont effectuées simultanément ; pendant le filtrage d'une projection, la suivante est acquise, et la précédente est épandue.

Le gain en temps de ce type de machines est évident, mais cependant il est loin d'être suffisant pour concevoir des tomographes temps réel très performants. Ceci s'explique par le fait que les trois tâches citées ne sont pas de la même taille, l'opération d'épandage est beaucoup plus coûteuse en temps que le filtrage.

#### **1.3.2.2. Les machines parallèles à mémoire partagée**

D'autres machines réalisent l'épandage grâce à un tableau de processeurs.

Le travail est alors réparti entre les processeurs, il peut l'être de deux façons différentes en associant à un processeur, soit une partie



de l'image à reconstruire, soit une projection (et les projections sont pipelinées). La seconde approche est une approche dite "projection-driven" : les projections sont traitées les unes après les autres, au fur et à mesure qu'elles sont acquises. Dans l'autre approche possible, dite "pixel-driven", les pixels sont évalués après avoir recueilli toutes les projections. Cette deuxième approche ne permet pas un traitement très rapide, car le temps d'acquisition (qui n'est pas négligeable) n'est pas utilisé [GIL81].

Le point faible de ce type d'architecture est que les processeurs sont connectés à une unique mémoire image partagée, via un bus unique. Les tâches ont été réparties, mais la mémoire globale a été conservée, donc les accès mémoire sont aussi nombreux, et le bus autant encombré qu'avec un seul processeur de traitement [ALE83].

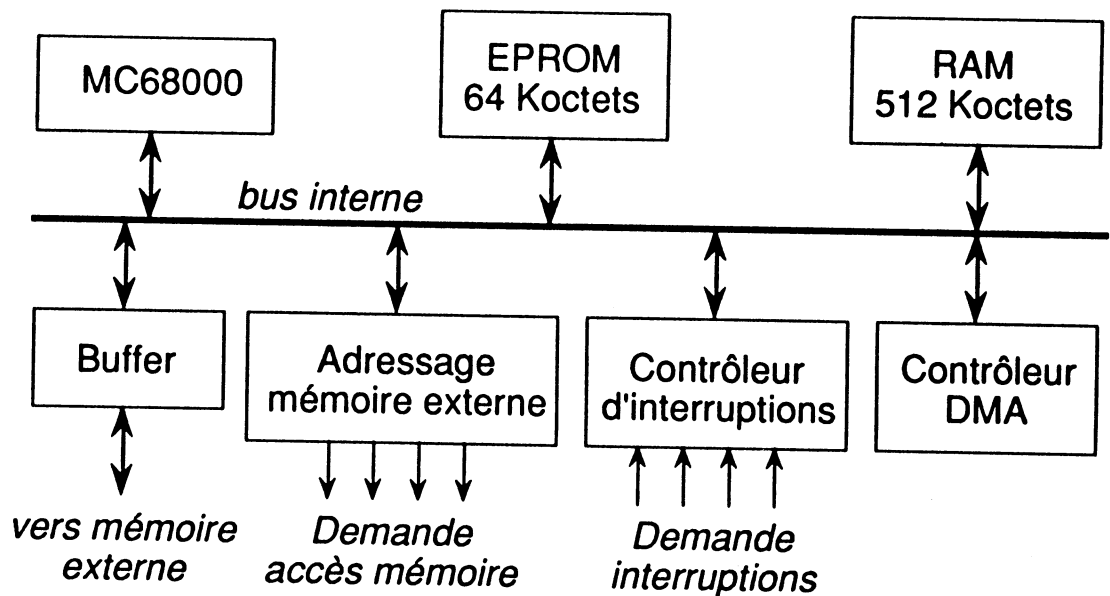
### **1.3.2.3. Les machines parallèles à mémoire distribuée**

Une autre classe de machines reprend la notion de parallélisme précédent, mais inclut en plus une division de la mémoire image globale en un certain nombre de petites mémoires locales aux processeurs. Ainsi un processeur traite une partie de l'image à reconstruire, celle-ci étant stockée dans une mémoire qui lui est propre. Les processeurs concurrents peuvent alors vraiment être efficaces, et les problèmes d'accès mémoire sont réduits.

Parmi ce type d'architecture, nous trouvons des machines travaillant avec des modules processeur, incluant des processeurs spécifiques ou classiques, et de la mémoire.

A l'université d'Aveiro (Portugal), une machine parallèle comprenant des modules processeurs à base de 68000 a été construite [BOR88]. Cette réalisation permet de reconstruire des images de  $128 \times 128$  pixels en une quarantaine de secondes, grâce à la méthode par épandage des projections filtrées. Les modules processeurs sont disposés sur une grille 2D, et dans cette première version il y en a seulement 4. Ils sont connectés via des ports d'entrée

sortie à module mémoire comprenant une RAM d'un mégaoctets et un arbitre gérant les accès. Un module processeur possède un microprocesseur 68000, une mémoire RAM de 512 kilooctets, une mémoire EPROM de 64 kilooctets, et des éléments s'occupant des interruptions et communications (voir figure 1.18.). Les modules processeurs peuvent communiquer avec le module mémoire, mais aussi entre eux.



**Figure 1.18.** Module processeur

Une machine de reconstruction basée sur un tableau de Transputers a été étudiée à l'université de Bristol (Grande-Bretagne). La méthode utilisée est la méthode itérative, et chaque Transputer réalise les traitements d'épandage et de projection sur une partie de l'image, qui est stockée dans sa mémoire interne. Un ordinateur hôte pilote le tableau de Transputers, et lui fournit les informations à traiter. Les performances de cette machine dépend du nombre de Transputers utilisés, et de la taille de l'image affectée à chacun d'eux. [KIN86a], [KIN86b].

### **1.3.3. Le réseau cellulaire**

Depuis quelques années, notre équipe travaille sur une architecture cellulaire, où chaque cellule est un petit processeur capable de réaliser des opérations simples. Les cellules sont disposées sur une grille carrée, et elles peuvent communiquer entre elles par un mécanisme d'acheminement de messages original.

L'idée qui vient spontanément à l'esprit, pour réaliser les traitements d'épandage ou de projection (opération duale), des techniques de reconstruction d'images, à l'aide de cette architecture hautement parallèle, est d'associer une cellule à un pixel de l'image à reconstruire.

La suite du travail a consisté à définir l'architecture du réseau, et à implémenter des algorithmes de reconstruction d'images sur cette machine originale.





## ***Chapitre 2***

---

### La machine cellulaire

|                                                   |           |
|---------------------------------------------------|-----------|
| <b>2.1. Introduction .....</b>                    | <b>69</b> |
| 2.1.1. Stratégie globale.....                     | 69        |
| 2.1.2. L'opération d'épandage.....                | 70        |
| 2.1.3. L'opération de projection .....            | 71        |
| 2.1.4. La machine parallèle .....                 | 73        |
| <b>2.2. Le réseau cellulaire.....</b>             | <b>74</b> |
| 2.2.1. Architecture générale .....                | 74        |
| 2.2.2. Mécanisme de communication .....           | 75        |
| 2.2.3. La cellule .....                           | 77        |
| 2.2.4. La machine cellulaire .....                | 78        |
| 2.2.5. Les applications .....                     | 79        |
| 2.2.6. La reconstruction d'images.....            | 79        |
| <b>2.3. La cellule .....</b>                      | <b>80</b> |
| 2.3.1. Sa fonction .....                          | 80        |
| 2.3.1.1. L'aspect routage.....                    | 81        |
| 2.3.1.2. L'aspect traitement.....                 | 84        |
| 2.3.2. L'algorithme .....                         | 84        |
| 2.3.3. La structure des messages .....            | 86        |
| 2.3.3.1. Les informations de reconstruction ..... | 86        |
| 2.3.3.2. Les informations géométriques.....       | 87        |
| 2.3.3.2.1. Passage du cosinus et du sinus .....   | 87        |
| 2.3.3.2.2. Passage de la tangente .....           | 88        |
| 2.3.3.2.2.1. Calcul approché de la longueur ..    | 90        |
| 2.3.3.2.2.2. Traitement des projections .....     | 92        |
| 2.3.3.3. Structure définitive des messages .....  | 94        |

|                                                             |                |
|-------------------------------------------------------------|----------------|
| <b>2.3.4. Amélioration de l'aspect routage.....</b>         | <b>95</b>      |
| <b>2.3.5. Le codage des informations géométriques .....</b> | <b>97</b>      |
| 2.3.5.1. Le codage .....                                    | 98             |
| 2.3.5.2. Les erreurs de calcul.....                         | 99             |
| 2.3.5.2.1. Le cheminement du message .....                  | 99             |
| 2.3.5.2.2. La longueur du rayon.....                        | 104            |
| <b>2.3.6. Traitement d'une sous-image.....</b>              | <b>104</b>     |
| 2.3.6.1. Principe.....                                      | 104            |
| 2.3.6.2. L'algorithme de la cellule.....                    | 106            |
| 2.3.6.3. Le codage des informations géométriques..          | 108            |
| <br><b>2.4. L'ordinateur hôte.....</b>                      | <br><b>112</b> |
| 2.4.1. Son rôle.....                                        | 112            |
| 2.4.2. L'algorithme.....                                    | 113            |
| 2.4.2.1. L'épandage des projections filtrées.....           | 113            |
| 2.4.2.2. La technique itérative .....                       | 114            |
| <br><b>2.5. Les entrées et sorties des images.....</b>      | <br><b>114</b> |
| 2.5.1. Introduction .....                                   | 114            |
| 2.5.2. Sortie de l'image .....                              | 116            |
| 2.5.2.1. Principe.....                                      | 116            |
| 2.5.2.2. L'ordinateur hôte .....                            | 118            |
| 2.5.2.3. La cellule .....                                   | 119            |
| 2.5.3. Entrée d'une image dans le réseau .....              | 119            |
| 2.5.3.1. Principe.....                                      | 119            |
| 2.5.3.2. L'ordinateur hôte .....                            | 121            |
| 2.5.3.3. La cellule .....                                   | 122            |



## Chapitre 2 : La machine cellulaire

## 2.1. INTRODUCTION

Dans ce chapitre, nous présentons l'implémentation d'algorithmes parallèles de reconstruction d'images sur une machine parallèle, basée sur un réseau cellulaire.

Nous nous intéressons aux méthodes de reconstruction d'images, qui sont actuellement les plus utilisées : l'épandage des projections filtrées (FBP), et les techniques itératives (ART et SIRT) présentées au chapitre précédent.

Rappelons que l'épandage des projections filtrées comprend deux phases : le filtrage des projections brutes, puis l'épandage de celles-ci. Le filtrage est l'application d'un système linéaire à chaque vecteur de projection, l'épandage est l'évaluation de l'image en distribuant les valeurs des points projetés aux pixels le long des chemins des rayons.

Par contre, les méthodes itératives consistent en partant d'une image fixée à priori, à calculer les projections de celle-ci, puis à épandre la différence entre les valeurs mesurées et les valeurs calculées de projection, ceci jusqu'à convergence de l'image. A chaque itération, une seule projection est prise en compte (techniques ART), ou l'ensemble des projections est considéré (techniques SIRT). Les valeurs de projection utilisées sont les valeurs brutes.

### 2.1.1. Stratégie globale

Nous allons détailler la méthode de reconstruction par épandage des projections filtrées, et la méthode itérative SIRT. Les algorithmes décrits sont ceux exécutés dans le cas d'un traitement séquentiel.

L'algorithme de la méthode par épandage des projections filtrées est très simple :

| filtrage des vecteurs de projection  
| *épandage* des projections filtrées

L'algorithme de la technique itérative est le suivant :

```
épandage des projections mesurées brutes
REPETER
  reprojection
  épandage de la différence entre les valeurs mesurées et
    les valeurs calculées
JUSQU'A convergence de l'image
```

Le premier épandage détermine l'image initiale, qui servira dans le procédé itératif. L'opération de reprojection consiste, en partant d'une image, à calculer des pseudo-projections (valeurs calculées), elle correspond à une opération de projection.

### 2.1.2. L'opération d'épandage

Chaque pixel de l'image est calculé, en effectuant la somme pondérée de tous les points projetés le traversant. La région de reconstruction est discrétisée, sous la forme d'un tableau de  $n$  pixels ( $n = N \times N$ ).

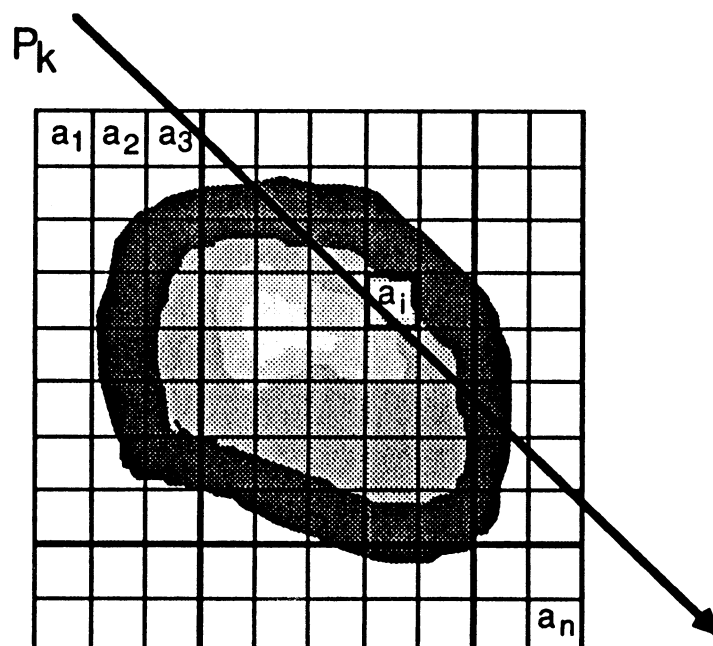


Figure 2.1. Coupe de l'objet

Si  $m$  est le nombre total de points projetés mesurés ( $m = M \cdot \text{nombre de points par angle de projection}$ ), nous pouvons formuler discrètement, l'opération d'épandage, par la relation :

$$a_i = \sum_{k=1}^{k=m} w_{ik} P_k \quad \text{pour } i = 1, \dots, n,$$

où  $a_i$  est la valeur du pixel  $i$ ,  $P_k$  la valeur du point projeté correspondant au rayon  $k$ , et  $w_{ik}$  un facteur pondérateur qui caractérise la contribution du rayon  $k$  au pixel  $i$  ( $w_{ik}$  est nul si le rayon  $k$  ne traverse pas le pixel  $i$ ).

### 2.1.3. L'opération de projection

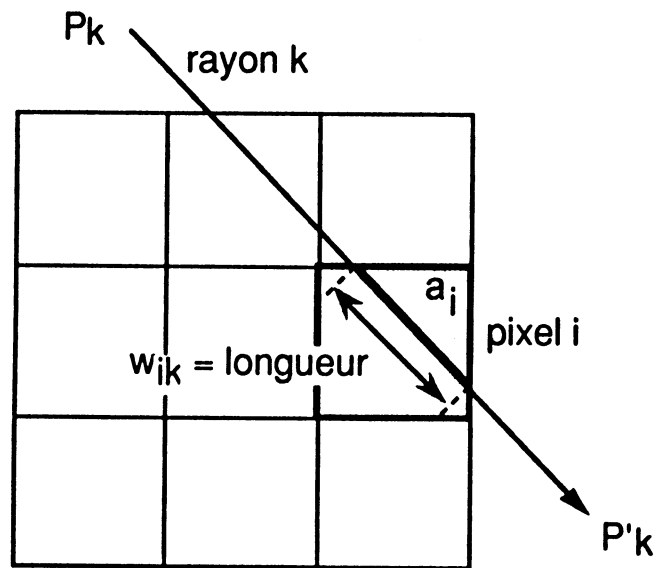
La valeur de projection de chaque rayon est calculée, en effectuant la somme pondérée de toutes les valeurs des pixels, qu'il traverse.

La formulation discrète de l'opération de projection s'écrit ainsi :

$$P'_k = \sum_{i=1}^{i=n} w_{ik} a_i \quad \text{pour } k = 1, \dots, m,$$

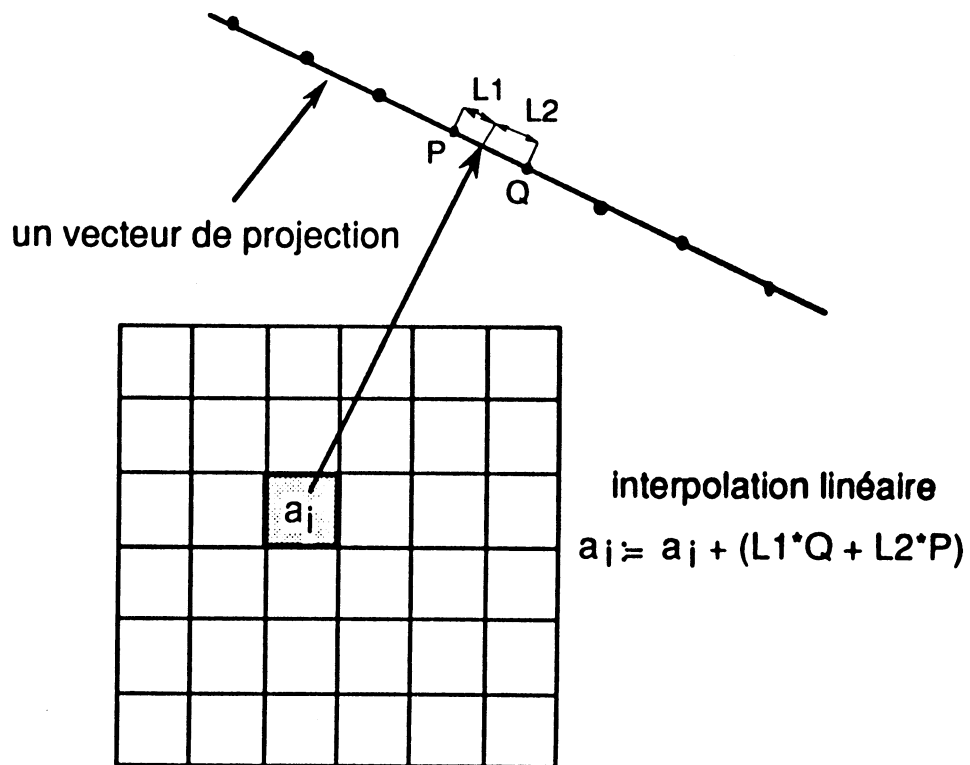
où  $a_i$  est la valeur du pixel  $i$ ,  $P'_k$  est la valeur calculée du point projeté correspondant au rayon  $k$  (pseudo-projection), et  $w_{ik}$  un facteur pondérateur qui caractérise la contribution du pixel  $i$  au rayon  $k$  ( $w_{ik}$  est nul si le rayon  $k$  ne traverse pas le pixel  $i$ ).

Le facteur  $w_{ik}$  intervenant dans les deux traitements (épandage et projection) est choisi en fonction de la précision demandée sur les opérations. Pour calquer parfaitement la réalité, il faut associer à chaque rayon une certaine largeur et donc le facteur  $w_{ik}$  correspond à la surface du rayon  $k$  dans le pixel  $i$ . Mais il s'avère que nous n'avons pas besoin d'une information aussi précise : la longueur du rayon dans le pixel est suffisante (voir figure 2.2.).



**Figure 2.2.** Contribution des pixels aux projections

Il est intéressant de remarquer que dans la plupart des machines existantes, les phases d'épandage et de reprojection ne sont pas réalisées de cette manière.



**Figure 2.3.** Principe de mise à jour des pixels par interpolation

Pour épandre une projection sur une image, chaque pixel est projeté sur le vecteur de projection, et la valeur à ajouter au pixel considéré est évaluée par interpolation. Cette interpolation peut être simplement linéaire : seuls les deux points voisins sont pris en compte, ou plus complexe. La figure 2.3. illustre un épandage par interpolation linéaire.

La méthode utilisée, consistant à mettre à jour la valeur du pixel en fonction de la longueur du rayon dans celui-ci et la méthode par interpolation linéaire permet de reconstruire des images de qualité équivalente (à en juger par les résultats de simulation).

### 2.1.4. La machine parallèle

Les machines de reconstruction actuelles du type architecture monoprocesseur sont très vite dépassées devant la quantité de calculs, que demandent les opérations d'épandage et de projection, et ceci d'autant plus que les images doivent être de plus en plus précises (1024\*1024 points au moins).

Il est par conséquent très intéressant de concevoir une machine complète, où les phases d'épandage et de projection sont réalisées par un co-processeur à structure massivement parallèle. Une telle machine comprend un ordinateur hôte gérant les différentes étapes de l'algorithme de reconstruction, et un co-processeur utilisé pour les phases critiques d'épandage et de projection, afin de soulager le travail de l'hôte.

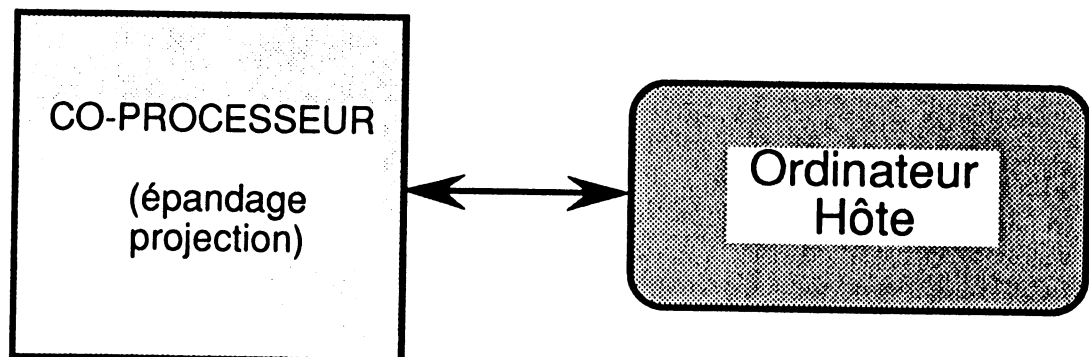


Figure 2.4. La machine parallèle

## 2.2. LE RESEAU CELLULAIRE

### 2.2.1. Architecture générale

Cette machine cellulaire est basée sur un réseau régulier de cellules asynchrones organisées en une matrice plane. Elle est caractérisée par deux propriétés essentielles. D'une part, les processeurs élémentaires sont totalement asynchrones, chaque cellule travaille lorsqu'elle a des messages à traiter, indépendamment des autres, et éventuellement à sa propre cadence. Pour l'application reconstruction d'images, les messages circulent dans le réseau selon différentes directions (les rayons ne sont pas tous parallèles), ce qui crée des surcharges de travail pour certaines cellules. D'autre part, le réseau possède un mécanisme de communication par messages, permettant à une cellule quelconque d'échanger des informations avec n'importe quelle autre. Ce mécanisme privilégie les communications locales, mais permet également des communications éloignées.

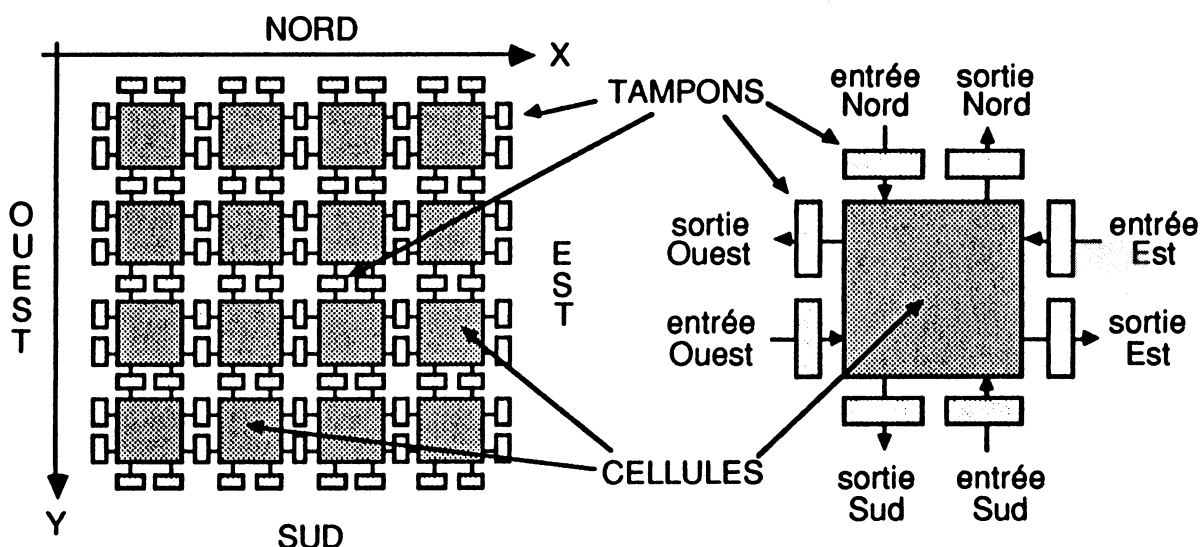


Figure 2.5. Architecture du réseau cellulaire

Cette machine entre dans la catégorie des machines cellulaires non systoliques, c'est à dire de type MIMD (Multiple Instruction, Multiple Data) d'après la classification de Flynn [FLY72]. Elle permet de traiter des algorithmes particuliers et non réguliers, qui ne pourraient pas l'être grâce aux réseaux systoliques [COR86].

L'architecture proposée est d'autant plus performante que le nombre d'éléments de base est grand ; de ce fait la cellule entité du réseau est un processeur assez élémentaire capable de réaliser des opérations simples. La puissance de la machine réside dans sa répétitivité, et dans la spécificité de la cellule de base pour une application particulière.

### **2.2.2. Mécanisme de communication**

L'implantation des cellules sur silicium laisse disponible un périmètre important de la cellule, qui autorise de transmettre les bits des messages en parallèle. Chaque cellule est donc entourée par 8 tampons unidirectionnels : 4 en entrée, et 4 en sortie, qui sont utilisés pour les communications entre cellules adjacentes. Le protocole repose sur le principe de boîte à lettres, en prenant soin d'assurer l'exclusion mutuelle des deux cellules directement connectées à un tampon donné.

Chaque tampon peut contenir un seul message, il est rempli par une cellule et vidé par la cellule voisine. Le tampon intègre un drapeau, qui assure l'exclusion mutuelle dans le cas de cellules asynchrones, et qui rend compte aux deux cellules connectées de l'état du tampon : état plein (le tampon contient un message), ou état vide (pas de message).



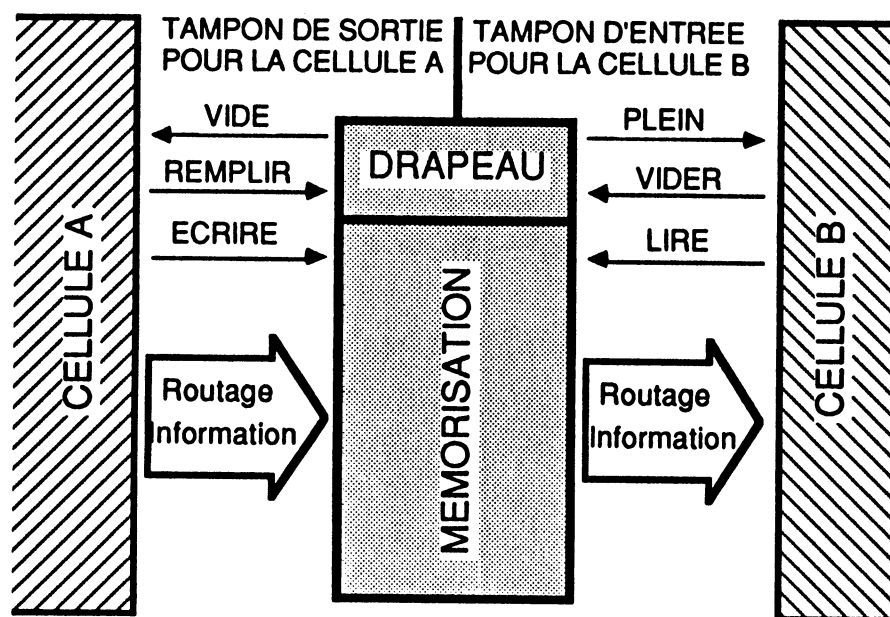
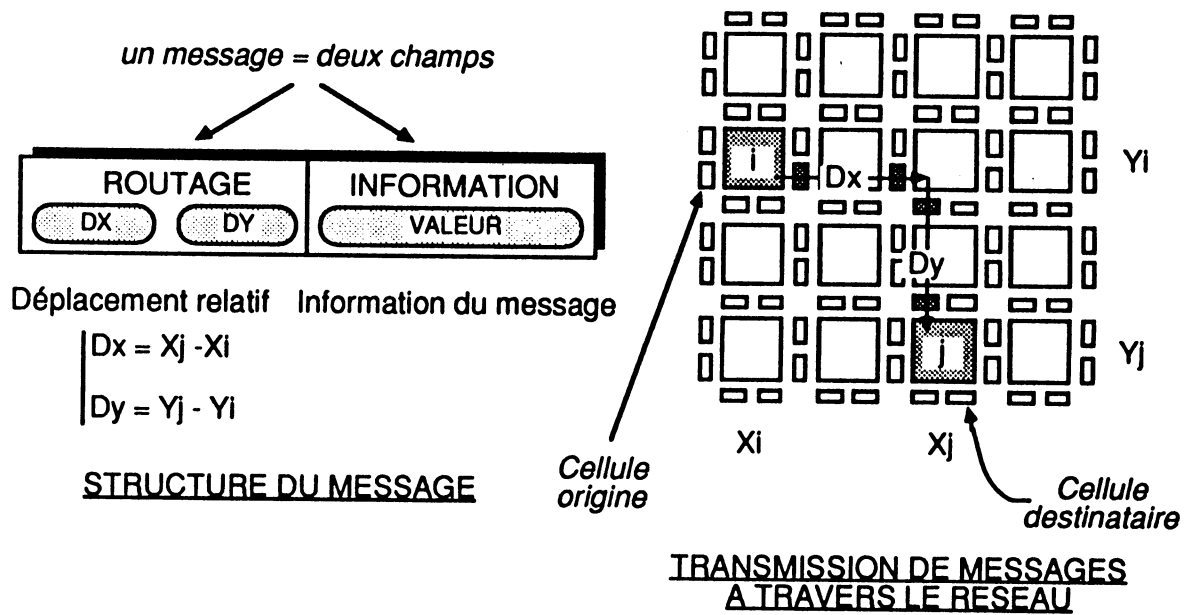


Figure 2.6. Un tampon

Les communications entre cellules éloignées peuvent avoir lieu, bien qu'il n'y ait pas de connexions matérielles entre elles. Pour cela, chaque cellule intègre un mécanisme d'acheminement de messages. Un message est composé de deux champs : un champ adresse, qui contient la position relative de la cellule destinataire par rapport à la position courante du message, et un champ information, qui contient les données à transmettre. Les messages transitent de la cellule émettrice à la cellule destinataire par un ensemble de cellules, formant un chemin entre les deux cellules qui veulent communiquer. Ce chemin est calculé au fur et mesure de la progression du message en fonction de la politique de routage choisie, et le champ adresse est alors mis à jour.

La figure 2.7. illustre une politique de routage très simple. Le message contient deux valeurs ( $D_x$ ,  $D_y$ ), représentant les déplacements relatifs entre les deux cellules qui veulent communiquer. L'acheminement se fait d'abord suivant l'axe X, puis suivant l'axe Y. Le message est arrivé à destination lorsque  $D_x=0$  et  $D_y=0$ , et il est alors transmis à la partie traitement de la cellule.



**Figure 2.7.** Principe du mécanisme de communication

L'aiguilleur présenté ici est entièrement séquentiel : un seul message peut être acheminé à la fois. Cette solution n'est pas optimale, et peut causer un effondrement des performances du réseau, dans le cas où l'application traitée possède un graphe de connexions de grande taille (c'est à dire avec beaucoup de connexions, ou de longues connexions). Pour remédier à cela, nous étudions actuellement un aiguilleur combinatoire capable d'acheminer plusieurs messages en parallèle dans des temps très courts [KAR89].

### 2.2.3. La cellule

La cellule est la structure de base du réseau. Elle est composée de deux parties distinctes, fonctionnant en parallèle.

La partie aiguilleur assure, et gère l'acheminement des messages. La partie traitement réalise l'algorithme spécifique d'une application. Ces parties concurrentes communiquent entre elles par l'intermédiaire de tampons, selon le même principe que deux cellules adjacentes.

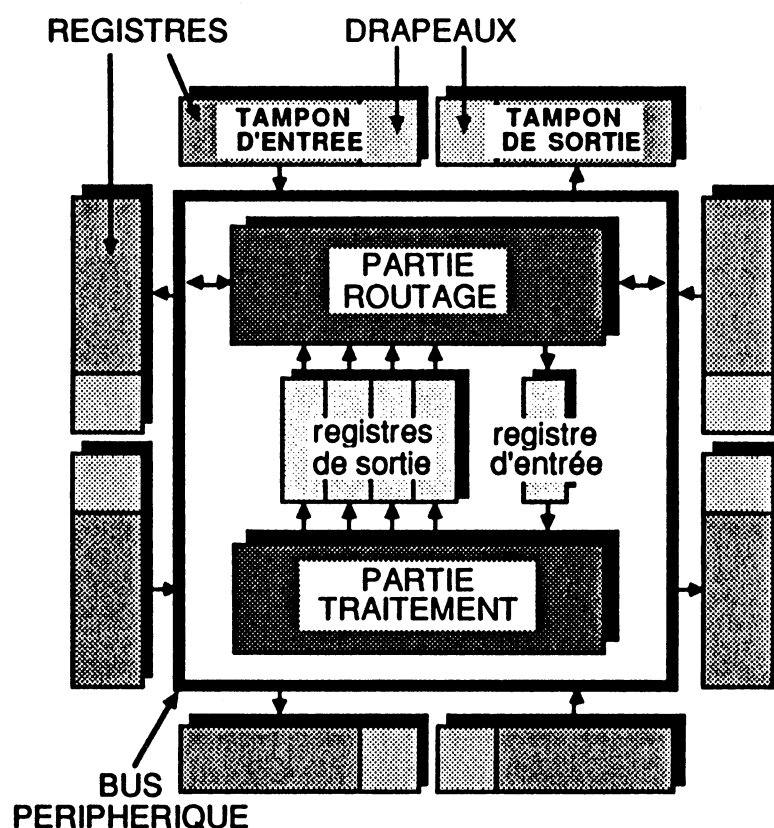


Figure 2.8. Schéma fonctionnel de la cellule

Pour la plupart des applications étudiées, les algorithmes implémentés en partie traitement comportent une boucle infinie qui consiste à recevoir des messages, calculer un résultat, et générer des messages.

## 2.2.4. La machine cellulaire

Le réseau cellulaire est piloté par un ordinateur hôte, qui gère les entrées et sorties, et les synchronisations entre les différentes phases de calcul. Outre les commandes de synchronisation globale (*reset*, ...), et divers signaux spécifiques à une application donnée, l'ordinateur hôte communique avec le réseau par l'intermédiaire des tampons périphériques, en échangeant des messages.

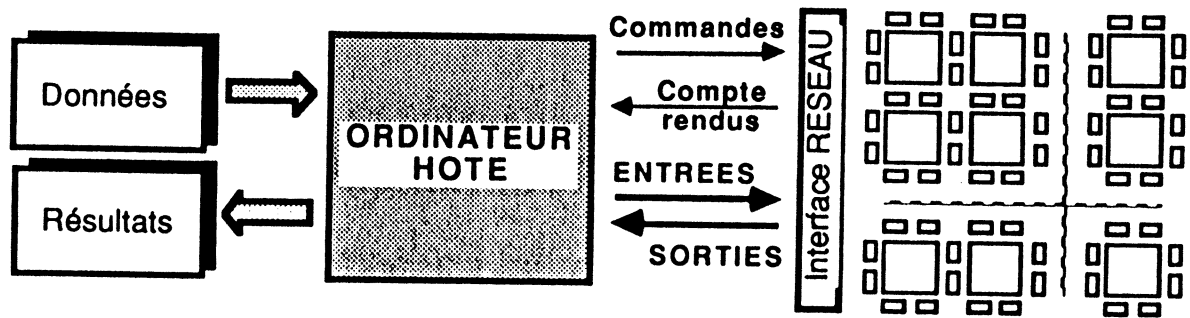


Figure 2.9. La machine cellulaire

### 2.2.5. Les applications

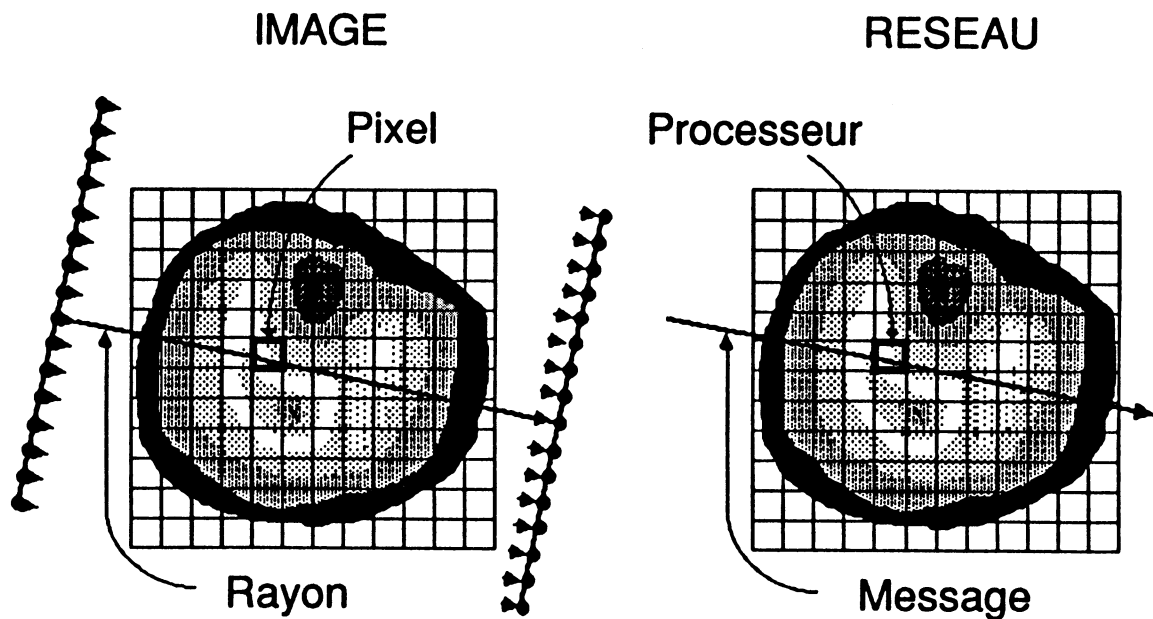
Cette machine cellulaire est actuellement étudiée pour d'autres applications que la reconstruction d'images notamment :

- ⇒ un simulateur logique à délai unitaire et trois états (0, 1 et X) [COR87a], [COR87b], [OBJ87b],
- ⇒ un placeur fonctionnant par échange de paires [COR88],
- ⇒ un réseau neuronique multicouche, basé sur un algorithme d'apprentissage par rétro-propagation des gradients [FAU88a],
- ⇒ un réseau d'opérateurs programmables [PAY88].

### 2.2.6. La reconstruction d'images

La reconstruction d'images est une application de notre machine cellulaire, qui ne nécessite pas toute sa puissance de communication. En effet, de part la régularité des traitements d'images auxquels nous nous intéressons, une cellule a besoin de communiquer uniquement avec ses quatre voisines les plus proches. Tout message arrivant à une cellule est traité par celle-ci, donc la partie aiguillage est ici inutile.

Une manière évidente de décomposer les traitements d'épandage et de projection en tâches élémentaires, pour les faire exécuter par un réseau cellulaire, est de distribuer les opérations mettant en jeu un pixel à une cellule [LAT88a].



**Figure 2.10.** Association pixel processeur

De la même façon que les rayons se propagent dans le plan de l'objet étudié, lors de l'acquisition, pour collecter les valeurs de projection, des messages contenant ces informations circulent dans le tableau de processeurs, afin de réaliser la reconstruction.

## 2.3. LA CELLULE

### 2.3.1. Sa fonction

La cellule est l'élément de base du réseau. C'est un petit processeur capable de réaliser des opérations simples, et pouvant échanger des informations avec ses quatre voisins les plus proches, sous forme de messages, par l'intermédiaire des tampons l'entourant.

Le rôle fonctionnel de la cellule comprend deux aspects :

→ un aspect routage : elle doit recevoir les messages qui lui sont destinés, et les transmettre à ses voisins de manière à respecter les chemins des rayons,

→ un aspect traitement : en fonction du type de message reçu (épandage ou projection), elle doit exécuter le traitement correspondant (mise à jour de la valeur du pixel ou de la valeur de la pseudo-projection contenue dans le message). Explicitement, si la cellule  $i$  traite le pixel  $i$  de valeur  $a_i$ , elle effectuera l'une des deux opérations suivantes :

$$\text{épandage} \quad \Rightarrow \quad a_i := a_i + (\text{LONG} * P_k)$$

$$\text{projection} \quad \Rightarrow \quad P'_k := P'_k + (\text{LONG} * a_i)$$

où LONG est la longueur du rayon dans le pixel,  $P_k$  la valeur de projection à épandre (information contenue dans le message, et restant constante tout au long de la traversée du message dans le tableau de pixels), et  $P'_k$  la valeur de pseudo-projection en cours de calcul (information contenue dans le message, et mise à jour à chaque traversée de pixel).

### 2.3.1.1. L'aspect routage

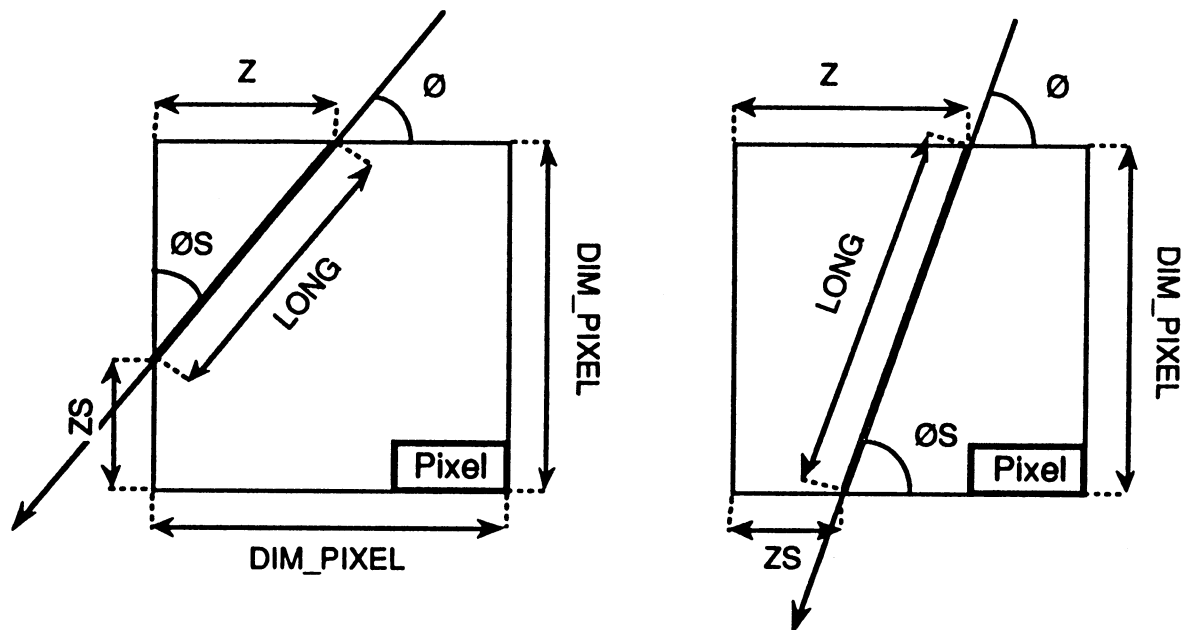
La cellule doit avoir des informations géométriques caractérisant le rayon pour router correctement le message, c'est à dire assurer la coïncidence du cheminement du message avec celui du rayon correspondant.

Ces informations sont de deux types :

→ celles concernant le point d'entrée du rayon dans le pixel, pour cela une seule suffit : la coordonnée d'entrée par rapport à la face d'entrée du rayon Z,

→ celles concernant l'angle d'incidence  $\varnothing$  du rayon dans le pixel : le sinus SIN, ou le cosinus COS, ou la tangente TG, ou la cotangente COTG (le choix des fonctions trigonométriques, plutôt que d'autres est fait et expliqué plus loin).

En fonction des informations géométriques reçues, la cellule génère d'autres informations géométriques, qu'elle passera à sa voisine. Elle doit calculer le point de sortie du rayon ZS (qui est le point d'entrée du rayon dans la cellule destinataire du message), et les informations caractérisant l'angle de sortie  $\emptyset S$  (COSS, SINS, TGS, COTGS).



**Figure 2.11. Principe de routage**

Les relations entre les informations d'entrée et de sortie dans le cas, où la face de sortie du rayon est adjacente à la face d'entrée (schéma gauche de la figure 2.11.) sont :

$$\rightarrow ZS = DIM\_PIXEL - (Z * TG)$$

$$\rightarrow \emptyset S = \frac{\pi}{2} - \emptyset$$

ou en termes de fonctions trigonométriques :

$$\Rightarrow SINS = COS$$

$$\Rightarrow COSS = SIN$$

$$\Rightarrow TGS = COTG$$

$$\Rightarrow COTGS = TG$$

Les relations entre les informations d'entrée et de sortie dans le cas, où la face de sortie du rayon est opposée à la face d'entrée (schéma droite de la figure 2.11.) sont :

$$\rightarrow ZS = Z - (DIM\_PIXEL * COTG)$$

$$\rightarrow \emptyset S = \emptyset$$

ou en termes de fonctions trigonométriques :

$$\Rightarrow SINS = SIN$$

$$\Rightarrow COSS = COS$$

$$\Rightarrow TGS = TG$$

$$\Rightarrow COTGS = COTG$$

La figure 2.12. illustre les différents calculs que la cellule doit effectuer en fonction de Z et  $\emptyset$ .

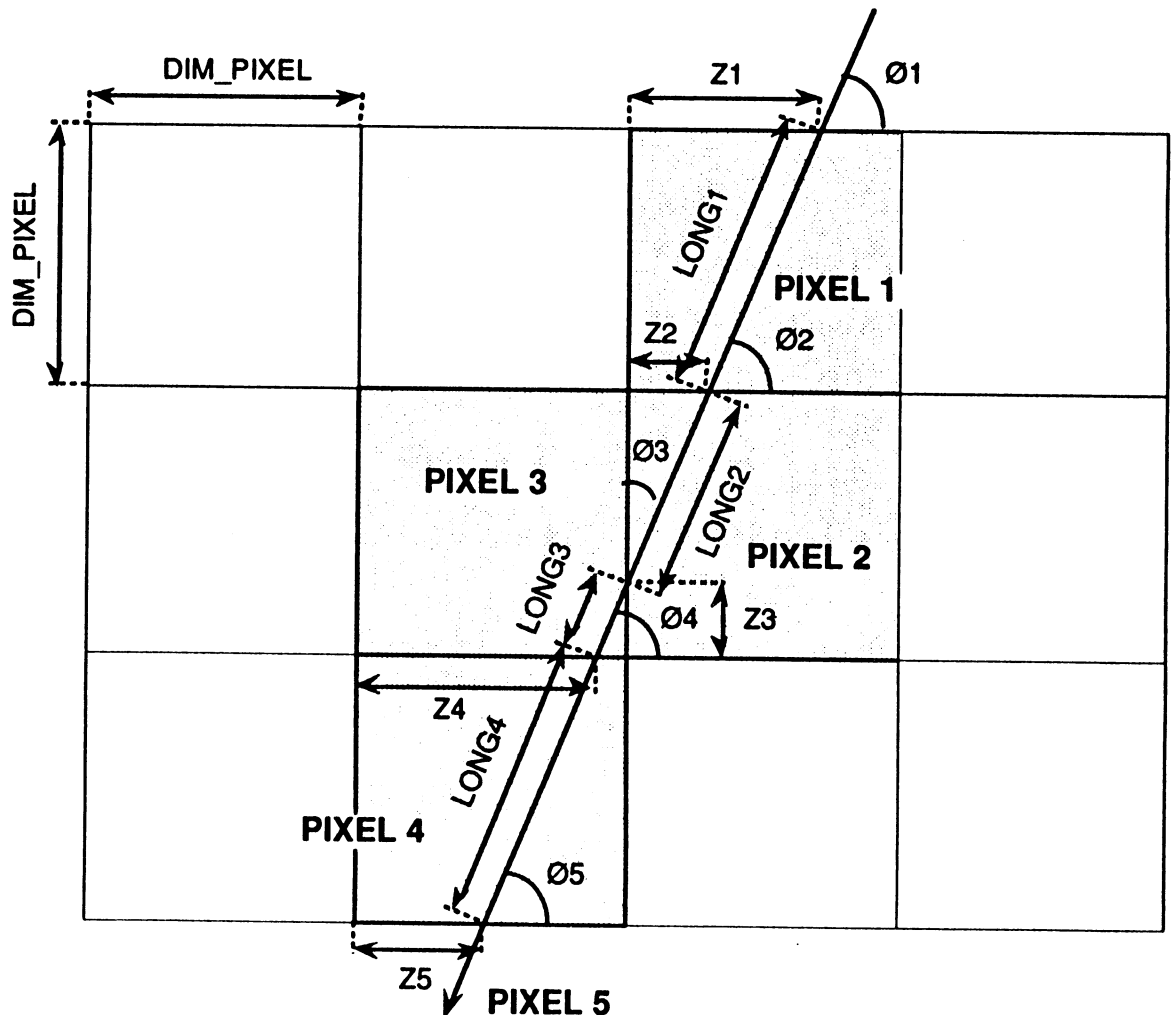


Figure 2.12. Les différents points d'entrée du rayon



Le pixel  $i$  reçoit les informations :  $Z_i$  et  $\emptyset_i$ , et calcule les informations :  $Z_{i+1}$  et  $\emptyset_{i+1}$  pour le routage, et  $LONG_i$  pour le traitement (voir paragraphe suivant).

### 2.3.1.2. L'aspect traitement

La cellule doit, quel que soit le type du message à traiter (épandage ou projection), calculer à partir des informations géométriques reçues, la longueur du rayon dans le pixel  $LONG$ .

Dans le cas où la face de sortie du rayon est adjacente à la face d'entrée, le calcul de  $LONG$  se fait par la relation :

$$\rightarrow LONG = \frac{Z}{\cos}$$

et dans le cas où la face de sortie du rayon est opposée à la face d'entrée, par :

$$\rightarrow LONG = \frac{DIM\_PIXEL}{\sin}$$

### 2.3.2. L'algorithme

Le principe de l'algorithme exécuté par une cellule élémentaire est simple. Après sélection d'un tampon d'entrée plein, la cellule lit le message afin de déterminer la cellule destinataire, et si la transmission peut se faire (tampon de sortie vide) alors elle traite le message, sinon elle laisse le message en attente dans le tampon d'entrée.

La politique consistant à ne traiter que les messages qui peuvent être transmis à l'issue du traitement, permet d'éviter tout blocage du réseau.

L'algorithme est :

```

sélectionner un tampon d'entrée
TANT_QUE < VRAI > FAIRE
    tester le drapeau du tampon d'entrée sélectionné
    SI < tampon d'entrée sélectionné est plein > ALORS
        lire le message contenu dans le tampon d'entrée
        calculer l'adresse de sortie
        tester le drapeau du tampon de sortie destinataire
        SI < tampon de sortie destinataire est vide > ALORS
            vider le tampon d'entrée
            traiter le message
            écrire le message dans le tampon de sortie
            remplir le tampon de sortie
        FIN_SI
    FIN_SI
    considérer le tampon d'entrée suivant
FIN_TANT_QUE
    
```

Les phases les plus complexes à implémenter sont le calcul de l'adresse, et le traitement du message.

Le calcul de l'adresse a plusieurs objectifs :

- déterminer la cellule destinataire (le côté de sortie du message),
- calculer les informations géométriques du message, qui sera éventuellement transmis après le traitement (ZS, ØS) ,
- calculer la longueur du chemin du rayon dans la cellule (LONG).

Les deux derniers calculs sont faits par anticipation, bien qu'ils soient inutiles dans le cas où le message ne serait pas traité par la cellule (tampon destinataire plein). Nous verrons plus loin qu'ils pourront être effectués en parallèle avec le calcul du côté de sortie, et donc la cellule ne sera pas ralentie par des opérations inutiles.

Le traitement du message dépend de son type, comme nous l'avons vu précédemment. Etant donné que le calcul de la longueur a déjà été effectué par anticipation lors du calcul de l'adresse, la cellule doit uniquement mettre à jour la valeur du pixel dans le cas d'un message d'épandage, ou la valeur de la pseudo-projection circulant avec le message de projection.

L'algorithme qui vient d'être présenté, est une version simplifiée volontairement pour mieux expliquer le fonctionnement de la cellule. Les messages d'épandage peuvent être transmis avant d'être traités complètement d'où une bien meilleure activité globale du réseau. Cette transmission précoce du message ne peut être réalisée pour les messages de type projection, car dans ce cas, la cellule doit évidemment évaluer la pseudo-projection afin de pouvoir la passer.

### **2.3.3. La structure des messages**

Nous retrouvons dans la structure des messages les informations liées aux deux aspects de la cellule :

- pour le traitement des informations propres au problème de la reconstruction,
- et pour le routage des informations géométriques utilisées dans la phase calcul d'adresse de l'algorithme.

#### **2.3.3.1. Les informations de reconstruction**

Il y a deux informations que la cellule doit connaître pour traiter le message : le type du message TYPE, qui caractérise le traitement (épandage ou projection), et la valeur de projection à distribuer dans le cas d'un message d'épandage ( $P_k$ ), ou la valeur de pseudo-projection en cours de calcul dans le cas d'un message de projection ( $P'_k$ ). Cette deuxième information est stockée dans le champ INFO du message.

### 2.3.3.2. Les informations géométriques

Outre la coordonnée du point d'entrée dans le pixel Z, le message doit contenir des informations sur l'angle d'incidence du rayon par rapport au côté d'entrée de celui-ci.

Le choix des fonctions trigonométriques (SIN, COS, TG, COTG) dépend des calculs que la cellule devra effectuer. Pour le calcul du point de sortie ZS, la cellule a besoin de la tangente (TG), ou de la cotangente (COTG) ; et pour le calcul de la longueur du rayon (LONG), elle a besoin du sinus (SIN) et du cosinus (COS). Dans une première étude fonctionnelle, nous avons utilisé le sinus et le cosinus ; puis en cherchant à minimiser les calculs de la cellule, nous nous sommes aperçu que le passage de la tangente étant suffisant, et donnait d'aussi bons résultats.

#### 2.3.3.2.1. Passage du cosinus et du sinus

Dans cette première version fonctionnelle de la cellule, les messages contiennent trois informations géométriques : Z, COS, SIN (valeurs non signées), et un bit de signe de la tangente S (pour caractériser  $\varnothing \geq \frac{\pi}{2}$  ou  $\varnothing < \frac{\pi}{2}$ ). Avant d'effectuer le calcul du point de sortie ZS, la cellule calcule la tangente (TG) si  $\varnothing$  est inférieur à  $\frac{\pi}{4}$ , ou la cotangente (COTG) si  $\varnothing$  supérieur à  $\frac{\pi}{4}$ .

Les opérations réalisées lors du calcul d'adresse sont :

**SI  $\langle \varnothing \leq \frac{\pi}{4} \rangle$  > ALORS { sortie : côté adjacent à l'entrée }**

|      |                                          |
|------|------------------------------------------|
| TG   | $:= \frac{\text{SIN}}{\text{COS}}$       |
| ZS   | $:= \text{DIM\_PIXEL} - (Z * \text{TG})$ |
| COSS | $:= \text{SIN}$                          |
| SINS | $:= \text{COS}$                          |
| SS   | $:= \text{complément (S)}$               |
| LONG | $:= \frac{Z}{\text{COS}}$                |

## SINON

TG :=  $\frac{\text{COS}}{\text{SIN}}$

SI  $\langle ((\text{DIM\_PIXEL} * \text{TG}) \geq Z) \rangle$  ALORS { sortie : côté adjacent }

ZS :=  $\text{DIM\_PIXEL} - \frac{Z}{\text{TG}}$

COSS := SIN

SINS := COS

SS := complément (S)

LONG :=  $\frac{Z}{\text{COS}}$

SINON { sortie : côté opposé à l'entrée }

ZS :=  $Z - (\text{DIM\_PIXEL} * \text{TG})$

COSS := COS

SINS := SIN

SS := S

LONG :=  $\frac{\text{DIM\_PIXEL}}{\text{SIN}}$

FIN\_SI

FIN\_SI

Dans le cas où le côté de sortie du message est adjacent au côté d'entrée, le tampon destinataire est déterminé par le bit de signe de la tangente S. Si  $\emptyset$  est inférieur à  $\frac{\pi}{2}$ , le côté de sortie est le côté adjacent en tournant dans le sens trigonométrique, et si  $\emptyset$  est supérieur à  $\frac{\pi}{2}$ , le côté de sortie est le côté adjacent en tournant dans le sens inverse du sens trigonométrique.

### 2.3.3.2.2. Passage de la tangente

Les informations COS et SIN sont directement utilisées, uniquement dans le calcul de la longueur du rayon, donc le fait de transmettre la tangente (TG) si  $\emptyset$  est inférieur à  $\frac{\pi}{4}$ , ou la cotangente (COTG) si  $\emptyset$  est supérieur à  $\frac{\pi}{4}$ , au lieu des deux informations : COS et SIN, ne perturbe pas le cheminement du message (ZS dépend de TG ou COTG), mais implique de faire un calcul approché pour la longueur

(LONG). Le message contient deux informations géométriques : Z, TG, un bit de signe de la tangente S, et un bit TC spécifiant si nous passons la tangente ( $0 \leq \frac{\pi}{4}$ ), ou la cotangente ( $0 > \frac{\pi}{4}$ ). Il est clair que dans tous les cas nous aurions pu utiliser uniquement la tangente, mais celle-ci étant une fonction non bornée de 0 à  $+\infty$ , elle est très difficilement codable sur un nombre limité de bits avec une bonne précision, et elle engendrait par conséquent de grossières erreurs dans les calculs.

En comparaison au cas précédent, qui illustre les calculs lors du passage de COS et SIN, la cellule n'a plus le calcul de la tangente ou cotangente à effectuer.

Les opérations réalisées lors du calcul d'adresse sont :

```

SI < ( $0 \leq \frac{\pi}{4}$ ) > ALORS           { sortie : côté adjacent à l'entrée }
    ZS      := DIM_PIXEL - (Z * TG)
    TGS     := TG
    TCS     := complément (TC)
    SS      := complément (S)
    { calcul approché de LONG }
SINON
    SI < ((DIM_PIXEL * TG) ≥ Z) > ALORS { sortie : côté adjacent }
        ZS      := DIM_PIXEL -  $\frac{Z}{TG}$ 
        TGS     := TG
        TCS     := complément (TC)
        SS      := complément (S)
        { calcul approché de LONG }
        SINON                                     { sortie : côté opposé à l'entrée }
            ZS      := Z - (DIM_PIXEL * TG)
            TGS     := TG
            TCS     := TC
            SS      := S
            { calcul approché de LONG }
        FIN_SI
    FIN_SI

```

### 2.3.3.2.2.1. Calcul approché de la longueur

Dans ce paragraphe, nous présentons une méthode pour calculer de manière approchée la longueur LONG. Cette solution est relativement complexe à mettre en œuvre, et elle sera délaissée pour une autre qui simplifiera les traitements (voir paragraphe 2.3.3.2.2.2.).

Le problème du calcul de LONG revient à approcher COS par une fonction de TG. D'après les relations entre les fonctions trigonométriques, dans le cas où nous avons le calcul  $LONG = \frac{Z}{\cos}$  à effectuer, nous pouvons écrire  $LONG = Z * \sqrt{1 + TG^2}$ .

Le calcul de la racine carrée est trop complexe pour envisager d'intégrer un opérateur spécifique, le réalisant dans la cellule. Pour remédier à cela, il est possible d'effectuer le développement limité de  $\sqrt{1 + TG^2}$ , car dans le cas de ce calcul, TG est inférieur à 1; nous obtenons :

$$LONG := Z * (1 + \frac{TG^2}{2} - \frac{TG^4}{8} + \frac{TG^8}{16} + \dots)$$

en explicitant les trois premiers termes du développement limité.

L'erreur de calcul est d'autant plus grande, que le nombre de termes est petit, et que TG est grand (TG proche de 1).

⇒ Le calcul approché effectué en considérant un seul terme :

$$LONG := Z * (1 + \frac{TG^2}{2}) \quad [\text{approximation n°1}]$$

donne un résultat (LONG), dont seuls les quatre bits de poids forts sont corrects pour des valeurs de TG comprises entre 0 et 1.

⇒ Le calcul approché effectué en considérant deux termes :

$$LONG := Z * (1 + \frac{TG^2}{2} - \frac{TG^4}{8}) \quad [\text{approximation n°2}]$$

ou trois termes :

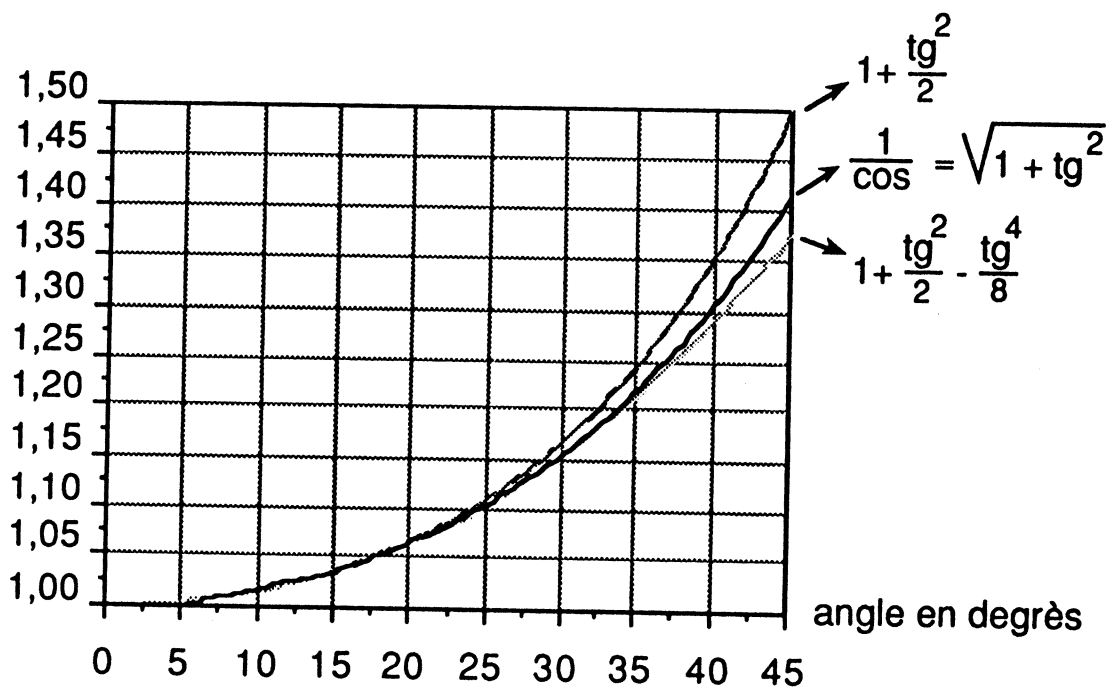
$$LONG := Z * (1 + \frac{TG^2}{2} - \frac{TG^4}{8} + \frac{TG^8}{16}) \quad [\text{approximation n°3}]$$

donne un résultat (LONG), dont seuls les cinq bits de poids forts sont corrects pour des valeurs de TG comprises entre 0 et 1.

⇒ Par contre, il est possible d'obtenir plus simplement un meilleur résultat (sept bits corrects) avec le calcul approché suivant :

$$\text{LONG} := Z * \left(1 + \frac{\text{TG}^2}{2} - \frac{\text{TG}^4}{12}\right) \quad [\text{approximation n}^\circ 4]$$

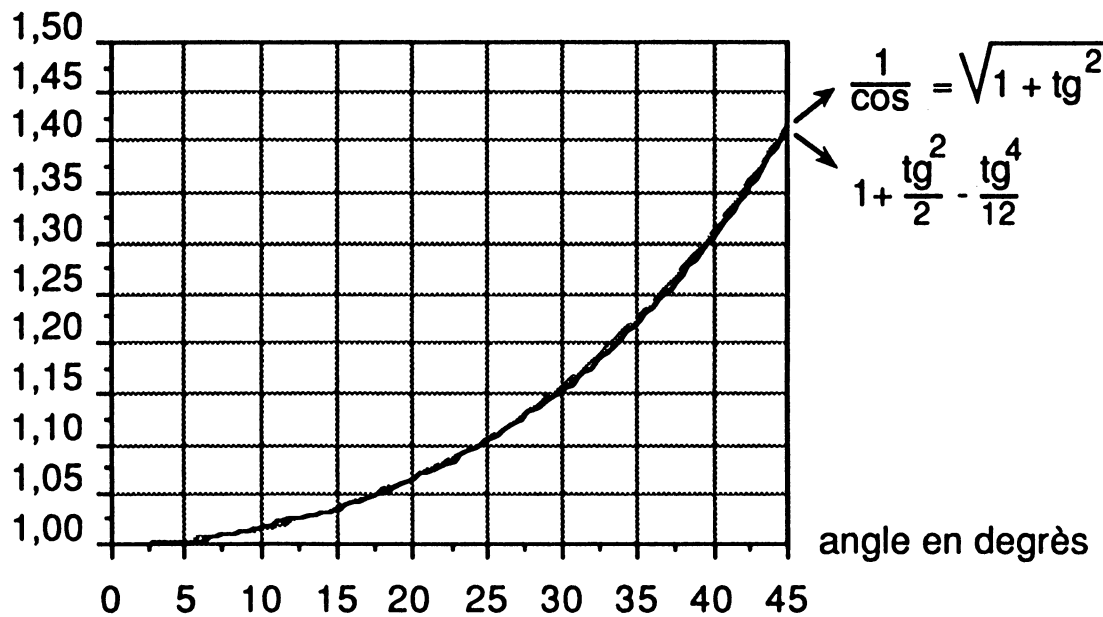
La figure 2.13. illustre les deux premières approximations [n°1 et n°2] et la fonction initiale  $\sqrt{1 + \text{TG}^2}$ , pour un angle variant de 0° à 45° (TG variant de 0 à 1).



**Figure 2.13.** Courbes des fonctions d'approximation

La figure 2.14. illustre la dernière approximation [n°4], et la fonction initiale  $\sqrt{1 + \text{TG}^2}$ , pour un angle variant de 0° à 45° (TG compris entre 0 et 1).





**Figure 2.14.** Meilleure approximation

Le calcul approché de la longueur à partir de la tangente, demande un certain nombre d'opérations complexes (multiplications et divisions), suivant la qualité d'approximation souhaitée. Sachant que nous voulons faire une réalisation matérielle de la cellule, cette solution n'est pas très satisfaisante, car elle est coûteuse en temps (si nous choisissons des algorithmes séquentiels), ou en surface (pour des implantations d'opérateurs parallèles).

### 2.3.3.2.2. Traitement des projections

Nous avons vu que l'information contenue dans le champ INFO du message, correspond à la valeur de projection  $P_k$  à distribuer ( lorsque TYPE = épandage ), ou à la valeur de pseudo-projection  $P'_k$  en cours de calcul ( lorsque TYPE = projection ).

Il est plus astucieux de passer, pour l'épandage la valeur de projection divisée par le maximum du cosinus et du sinus de l'angle d'incidence du rayon. C'est à dire  $INFO = \frac{P_k}{\text{Max}(\text{COS}, \text{SIN})}$ , et alors la mise à jour de la valeur du pixel pour l'épandage est réalisée par :

$$a_i := a_i + ( \text{LONG} * \text{Max} (\text{COS}, \text{SIN}) * \frac{P_k}{\text{Max} (\text{COS}, \text{SIN})} )$$

Auparavant dans le cas où la face de sortie du rayon était adjacente à la face d'entrée, le calcul de LONG se faisait par la relation  $\text{LONG} = \frac{Z}{\text{COS}}$ , et dans le cas où la face de sortie du rayon était opposée à la face d'entrée, par  $\text{LONG} = \frac{\text{DIM\_PIXEL}}{\text{SIN}}$ . Avec ce nouveau procédé, nous ne calculons plus LONG (la longueur réelle), mais le produit  $(\text{LONG} * \text{Max}(\text{COS}, \text{SIN}))$ , qui dans le premier cas est Z si  $\varnothing$  est inférieur à  $\frac{\pi}{4}$ , ou  $\frac{Z}{\text{COTG}}$  si  $\varnothing$  est supérieur à  $\frac{\pi}{4}$ , et dans le second cas est DIM\_PIXEL.

Nous n'avons plus besoin de faire de calcul approché pour la longueur réelle du rayon (elle n'est pas calculée), et le facteur à multiplier à la valeur de projection dépend uniquement de COTG.

Pour le traitement de projection, le facteur qu'il faut multiplier à la valeur de pseudo-projection en cours de calcul est le même (c'est à dire Z, ou  $\frac{Z}{\text{COTG}}$ , ou DIM\_PIXEL suivant le cas). Mais la valeur sortante du réseau de cellules devra être divisée par le maximum du cosinus et du sinus de l'angle d'incidence. Ce travail de pré-traitement pour l'épandage, et de post-traitement pour la projection est réalisé par l'ordinateur hôte fournissant et collectant les informations au réseau.

Les opérations réalisées lors du calcul d'adresse sont :

```

SI  $(\varnothing \leq \frac{\pi}{4})$  > ALORS      { sortie : côté adjacent à l'entrée }
    |
    | ZS      := DIM_PIXEL - (Z * TG)
    | TGS     := TG
    | TCS     := complément (TC)
    | SS      := complément (S)
    | LONG    := Z
SINON
    | SI  $((\text{DIM\_PIXEL} * \text{TG}) \geq Z)$  > ALORS { sortie : côté adjacent }
    | | ZS    := DIM_PIXEL -  $\frac{Z}{\text{TG}}$ 
    |

```

```

|   TGS  := TG
|   TCS  := complément (TC)
|   SS   := complément (S)
|   LONG :=  $\frac{Z}{TG}$ 
|   SINON                                { sortie : côté opposé à l'entrée }
|       ZS  := Z - (DIM_PIXEL * TG)
|       TGS := TG
|       TCS := TC
|       SS  := S
|       LONG := DIM_PIXEL
|   FIN_SI
FIN_SI

```

Les opérations réalisées lors du traitement du message sont :

```

SI < (TYPE = épandage) > ALORS
|    $a_i := a_i + LONG * INFO$ 
SINON                                { TYPE = projection }
|    $INFO := INFO + LONG * a_i$ 
FIN_SI

```

### 2.3.3.3. Structure définitive des messages

En ce qui concerne les informations géométriques du message, le choix de passer TG (tangente ou cotangente) est meilleur, puisqu'il permet de diminuer la taille du message par rapport au passage de COS et SIN. De plus, le fait de travailler avec des informations de reconstruction pré-traitées ou post-traitées limitent considérablement le nombre d'opérations de multiplication et division, car il n'est pas utile d'évaluer LONG par calcul approché.

La figure 2.15. donne la composition finale du message.

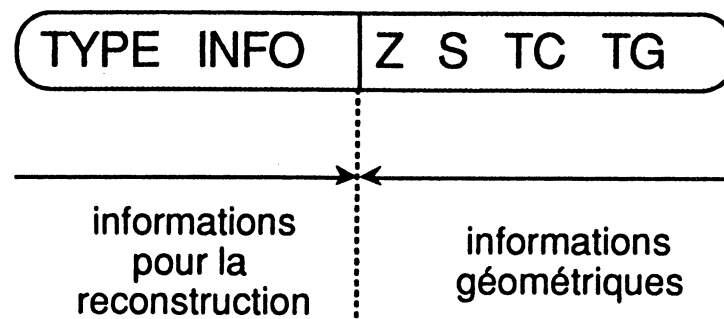


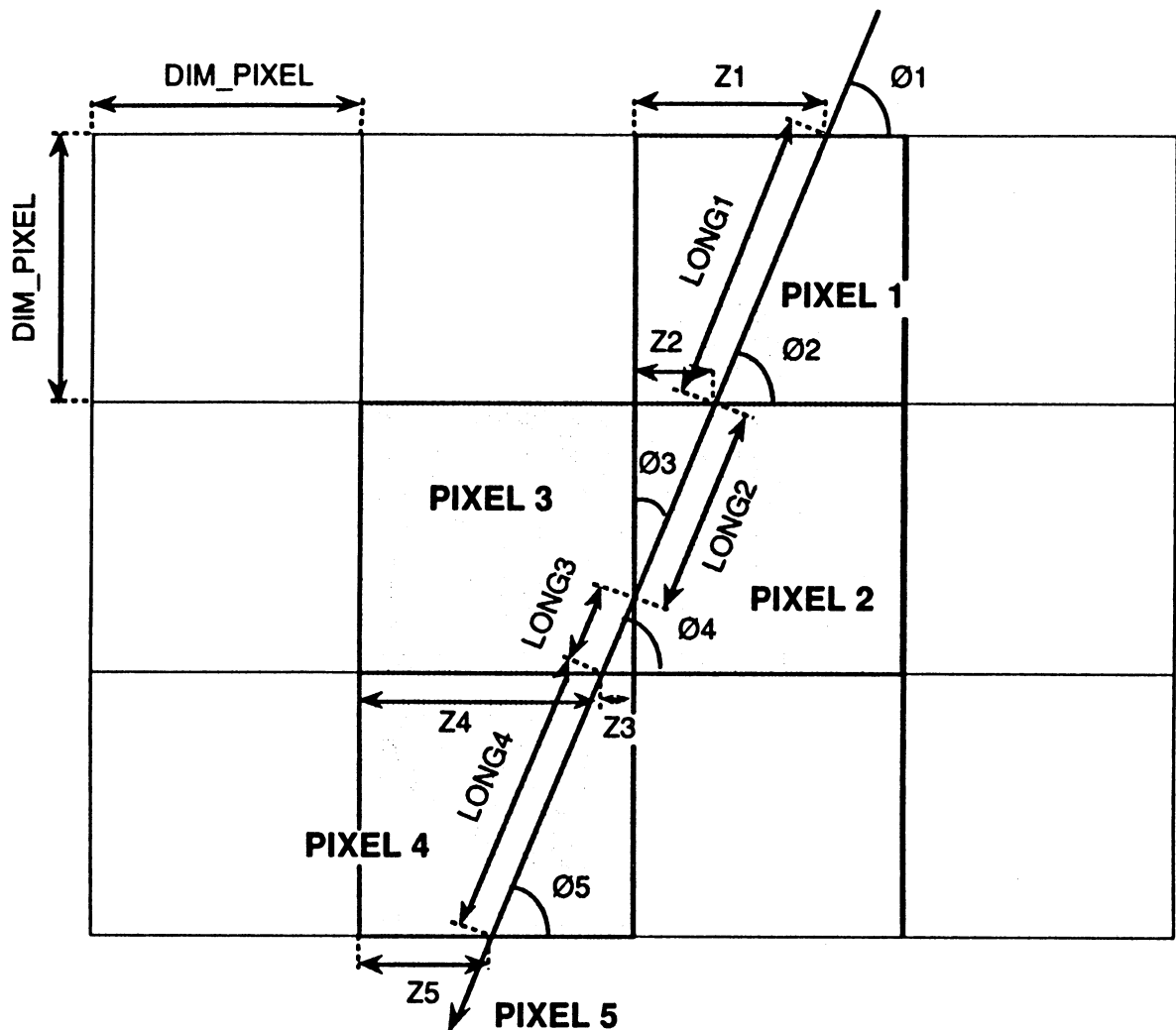
Figure 2.15. Structure du message

### 2.3.4. Amélioration de l'aspect routage

Jusqu'ici l'élément cellulaire : le pixel, était vu comme une entité. Pour chaque pixel, à partir de l'information : coordonnée du point d'entrée du rayon, nous calculons la coordonnée du point de sortie du rayon. La totalité de ces informations n'est pas utile pour acheminer correctement les messages.

En effet, il est préférable de considérer l'image comme un ensemble de lignes ou de colonnes, et d'évaluer la coordonnée de sortie du rayon uniquement lorsqu'il y a changement de ligne ou de colonne. Lorsque l'angle d'incidence du rayon sur le réseau est inférieur à  $\frac{\pi}{4}$ , le réseau est traité comme un ensemble de colonnes, et lorsqu'il est supérieur à  $\frac{\pi}{4}$ , comme un ensemble de lignes. Grâce à cette modification apportée aux calculs des coordonnées des points d'entrée dans les pixels successifs, la cellule n'a plus le produit  $Z \cdot TG$ , ou le rapport  $\frac{Z}{TG}$  à évaluer.

La figure 2.16. illustre les calculs que doit effectuer la cellule.



**Figure 2.16.** L'image est un ensemble de colonnes

Compte-tenu de cette hypothèse, les opérations réalisées lors du calcul d'adresse sont :

**SI**  $(\varnothing \leq \frac{\pi}{4})$  **> ALORS** { sortie : côté adjacent à l'entrée }

|      |                    |
|------|--------------------|
| ZS   | := DIM_PIXEL - Z   |
| TGS  | := TG              |
| TCS  | := complément (TC) |
| SS   | := complément (S)  |
| LONG | := $\frac{Z}{TG}$  |

**SINON**

```

SI < ((DIM_PIXEL * TG) ≥ Z) > ALORS { sortie : côté adjacent }
    ZS      := (DIM_PIXEL * TG) - Z
    TGS     := TG
    TCS     := complément (TC)
    SS      := complément (S)
    LONG    :=  $\frac{Z}{TG}$ 
SINON                                     { sortie : côté opposé à l'entrée }
    ZS      := Z - (DIM_PIXEL * TG)
    TGS     := TG
    TCS     := TC
    SS      := S
    LONG    := DIM_PIXEL
FIN_SI
FIN_SI

```

Dans les deux derniers cas, l'évaluation de ZS demande la réalisation du produit (DIM\_PIXEL \* TG) ; nous verrons plus loin, lors du codage de TG, que ce produit est supprimé.

Cette version de la mise en œuvre du calcul de l'adresse, et du traitement du message, engendre le minimum de opérations de multiplication et division. Ce qui est très important, car elles sont très coûteuses en temps ou surface. Pour le calcul de l'adresse, dans le pire cas, la cellule devra faire une division ( $\frac{Z}{TG}$ ) pour évaluer LONG, et pour le traitement, dans tous les cas, une multiplication (LONG\*INFO ou LONG\*a<sub>i</sub>).

### 2.3.5. Le codage des informations géométriques

Le cheminement des messages doit être très précis, ceci afin qu'ils traversent les "bonnes" cellules, et que le facteur LONG soit calculé avec le moins d'erreur possible.

Sachant que nous travaillons avec des valeurs à précision limitée, le cheminement des messages dépend évidemment du nombre de bits utilisés pour coder les informations géométriques Z et TG. Il est par

conséquent intéressant d'établir une relation entre le codage des informations, et la précision des calculs, en fonction de la taille de l'image.

### 2.3.5.1. Le codage

Les informations Z et TG sont codées à priori, sur un même nombre de bits : l, avec un bit de poids faible de valeur  $2^0$ , et un bit de poids fort de valeur  $2^{l-1}$ . Elles peuvent prendre  $2^l$  valeurs différentes.

L'information contenue dans le champ TG correspond, soit à la tangente de l'angle d'incidence  $\varnothing$  du rayon sur la face d'entrée ( $\varnothing \leq \frac{\pi}{4}$ ), soit à la cotangente ( $\varnothing > \frac{\pi}{4}$ ). Il en découle que cette valeur est comprise entre 0 et 1. Pour coder cette information sur l bits, nous choisissons de coder 0 par la valeur 0 (les l bits à 0), et 1 par  $(2^l - 1)$  (les l bits à 1), ce qui revient à coder le réel x compris entre 0 et 1, par la valeur arrondie de  $(x * (2^l - 1))$ . Nous obtenons ainsi un codage de TG, tel que sa valeur est entière et comprise entre 0 et  $(2^l - 1)$ .

Le codage de Z sur l bits définit la dimension du pixel DIM\_PIXEL par la valeur  $2^l$ .

En tenant compte de ces considérations, et en notant FACT la valeur  $(2^l - 1)$ , les opérations réalisées lors du calcul d'adresse sont :

```

SI  $(\varnothing \leq \frac{\pi}{4})$  > ALORS           { sortie : côté adjacent à l'entrée }
    ZS      := DIM_PIXEL - Z
    TGS      := TG
    TCS      := complément (TC)
    SS       := complément (S)
    LONG     :=  $\frac{Z * FACT}{TG}$ 
SINON
    SI  $(\frac{DIM\_PIXEL * TG}{FACT} \geq Z)$  > ALORS { sortie : côté adjacent }
        ZS      :=  $\frac{(DIM\_PIXEL * TG)}{FACT} - Z$ 
        TGS      := TG
    
```

```

| TCS := complément (TC)
| SS  := complément (S)
| LONG :=  $\frac{Z * FACT}{TG}$ 
| SINON { sortie : côté opposé à l'entrée }
|   ZS := Z  $\frac{(DIM\_PIXEL * TG)}{FACT}$ 
|   TGS := TG
|   TCS := TC
|   SS  := S
|   LONG := DIM_PIXEL
| FIN_SI
FIN_SI

```

### 2.3.5.2. Les erreurs de calcul

Nous avons évalué les erreurs de calcul dues à la précision limitée des informations pour le cheminement du message, et tiré les conséquences sur le calcul de LONG.

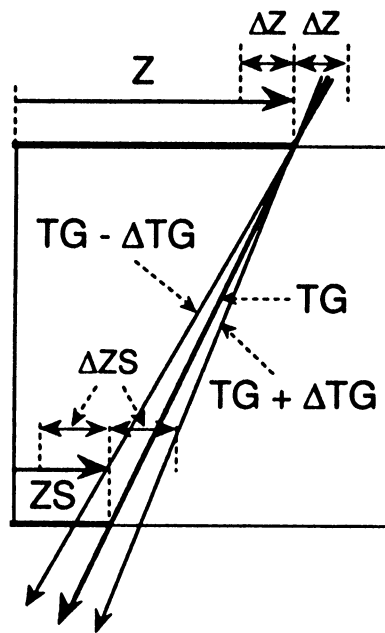
#### 2.3.5.2.1. Le cheminement du message

##### Traversée d'une cellule :

Le point de sortie du rayon (ZS) dans un pixel est fonction du point d'entrée du rayon (Z), et de la tangente de l'angle d'incidence du rayon (TG). Le codage de ces informations sur un nombre limité de bits entraînant une approximation, nous convenons de noter  $\Delta Z$ ,  $\Delta TG$ , et  $\Delta ZS$ , les erreurs maximales respectives de Z, TG, et ZS. Nous avons donc les relations :

$$\begin{array}{llll}
 (Z - \Delta Z) & \leq & Z_{\text{réel}} & \leq (Z + \Delta Z) \\
 (TG - \Delta TG) & \leq & TG_{\text{réel}} & \leq (TG + \Delta TG) \\
 (ZS - \Delta ZS) & \leq & ZS_{\text{réel}} & \leq (ZS + \Delta ZS)
 \end{array}$$





**Figure 2.17.** Erreur sur le cheminement du rayon

L'évaluation du point de sortie du pixel est réalisé suivant le cas, par l'un des trois calculs suivants :

$$ZS = DIM\_PIXEL - Z \quad [\text{calcul n}^\circ 1]$$

$$ZS = \frac{(DIM\_PIXEL * TG)}{FACT} - Z \quad [\text{calcul n}^\circ 2]$$

$$ZS = Z - \frac{(DIM\_PIXEL * TG)}{FACT} \quad [\text{calcul n}^\circ 3]$$

Le calcul n°1 n'engendre pas d'erreur supplémentaire,  $\Delta ZS = \Delta Z$ .

Pour les deux autres calculs [n°2 et n°3], l'erreur maximale engendrée correspond à la somme des erreurs des deux termes de la soustraction, donc

$$\Delta ZS = \left( \frac{DIM\_PIXEL}{FACT} * \Delta TG \right) + \Delta Z.$$

En ne tenant pas compte du rapport  $\frac{DIM\_PIXEL}{FACT}$ , qui est très peu différent de 1, nous obtenons  $\Delta ZS = \Delta TG + \Delta Z$ .

Il est possible de dire, que dans tous les cas, l'erreur maximale sur la coordonnée du point de sortie ZS est :

$$\Delta ZS = \Delta TG + \Delta Z.$$

### Traversée d'un réseau :

En supposant que nous travaillons avec une image de  $N \times N$  pixels (donc un réseau de  $N \times N$  cellules), et que l'angle d'incidence du rayon sur le réseau est supérieur à  $\frac{\pi}{4}$  (le réseau est considéré comme un ensemble de lignes), nous notons :

→  $Z_0$  est la coordonnée du point d'entrée dans la cellule de la première ligne du réseau (et  $\Delta Z_0$  son erreur),

→  $Z_1$  est la coordonnée du point de sortie de la cellule de la première ligne (et  $\Delta Z_1$  son erreur),

→  $Z_i$  est la coordonnée du point de sortie de la cellule de la  $i$ ème ligne (et  $\Delta Z_i$  son erreur),

→  $Z_N$  est la coordonnée du point de sortie de la cellule de la  $N$ ème ligne (et  $\Delta Z_N$  son erreur).

Les points de sortie successivement calculés pour un rayon, dans le cas où  $N = 4$ , sont illustrés par la figure 2.18.

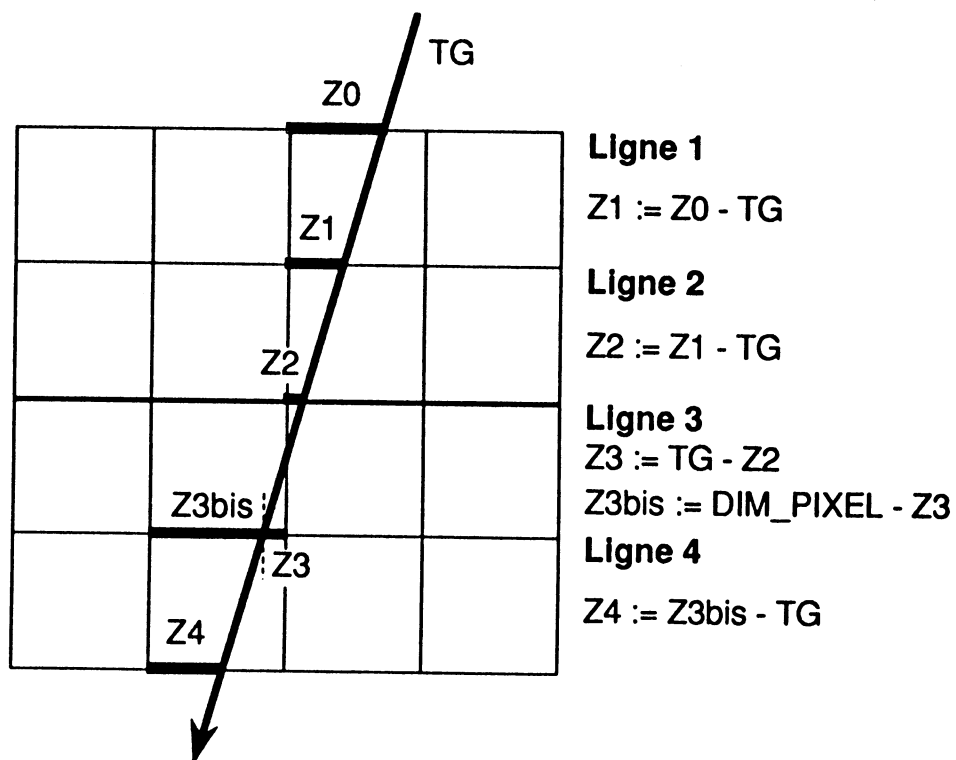


Figure 2.18. Les points d'entrée calculés

Les erreurs maximales suivent les relations :

$$\Delta Z_1 = \Delta TG + \Delta Z_0$$

$$\Delta Z_2 = \Delta TG + \Delta Z_1$$

.....

$$\Delta Z_i = \Delta TG + \Delta Z_{i-1}$$

.....

$$\Delta Z_N = \Delta TG + \Delta Z_{N-1}$$

Nous pouvons donc écrire :  $\Delta Z_N = N \cdot \Delta TG + \Delta Z_0$ .

L'erreur sur la coordonnée du point d'entrée de n'importe quelle cellule du réseau est inférieure à  $\Delta Z_N$ . Cette relation de  $\Delta Z_N$  montre que la précision de TG est beaucoup plus importante que celle de  $Z_0$ . Il semble donc très intéressant de coder la tangente TG sur un nombre de bits plus grand que Z.

La figure 2.19. illustre le nouveau codage utilisé : TG est codé sur l bits, et de Z sur (l - x) bits.

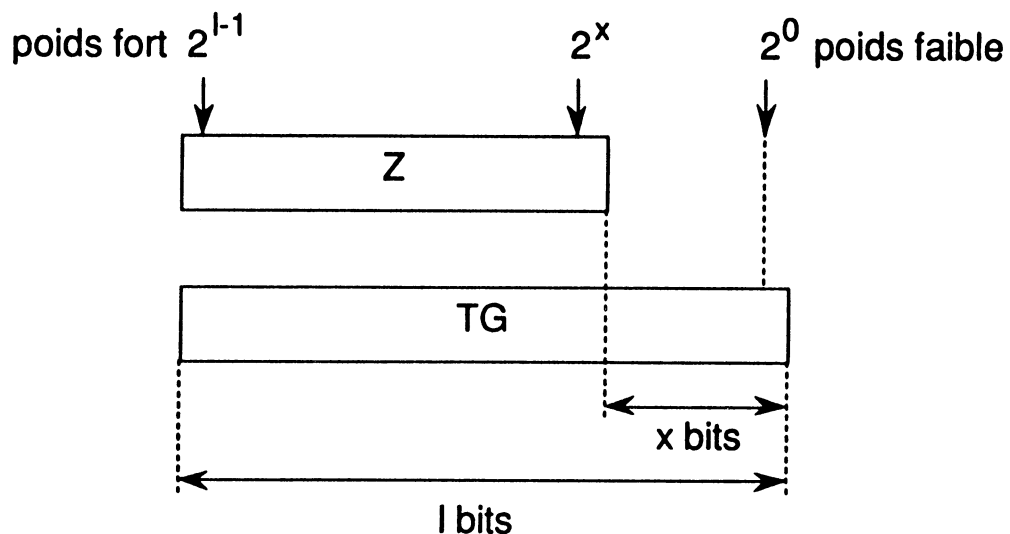


Figure 2.19. Codage de Z et TG

$\Delta TG$  et  $\Delta Z_0$  sont dues uniquement au codage initial des informations sur un nombre limité de bits, et donc nous pouvons dire que ( $\Delta TG = 2^{-1}$ ) et ( $\Delta Z_0 = 2^{x-1}$ ).

Il en découle la relation :  $\Delta Z_N = \frac{N}{2} + 2^{x-1}$ ,

si  $x$  est tel que ( $2^{x-1} = N$ ), c'est à dire ( $x = 1 + \log_2 N$ ), alors :

$$\Delta Z_N = 2^{x-2} + 2^{x-1}$$

( $2^{x-2} + 2^{x-1}$ )  $\leq 2^x$  quelle que soit la valeur de  $x$ , et donc :  $\Delta Z_N \leq 2^x$ .

$\Delta Z_N \leq 2^x - 1$ , si  $x \geq 2$ , c'est à dire  $N \geq 2$ .

Si nous souhaitons que les informations contenues dans le champ  $Z$  aient au moins  $p$  bits corrects, il faut que  $\Delta Z_N < \frac{\text{DIM\_PIXEL}}{2^p}$  ou encore que  $\Delta Z_N \leq \frac{2^l}{2^p} - 1$ .

Il faut donc que  $\Delta Z_N \leq 2^x - 1 \leq \frac{2^l}{2^p} - 1$

c'est à dire :  $2^x \leq \frac{2^l}{2^p}$

Nous arrivons à la relation :  $l \geq p + x$ , ou  $l \geq p + 1 + \log_2 N$ . Donc le nombre minimum de bits nécessaire pour coder  $TG$  afin que  $p$  bits de  $Z$  soient corrects est :  $l = p + 1 + \log_2 N$ . Et le nombre minimum de bits nécessaire pour coder  $Z$  est  $p$  bits.

Le codage des informations dépend du logarithme de la taille du réseau, ou de l'image.

Le tableau suivant donne le nombre de bits minimum pour le codage de  $TG$ , en fonction de la précision requise ( $p$  : axe vertical) sur  $Z$ , et de la taille du réseau ( $N$  : axe horizontal).

| $p \setminus N$ | 32 | 64 | 128 | 256 | 512 | 1024 |
|-----------------|----|----|-----|-----|-----|------|
| 4               | 10 | 11 | 12  | 13  | 14  | 15   |
| 5               | 11 | 12 | 13  | 14  | 15  | 16   |
| 6               | 12 | 13 | 14  | 15  | 16  | 17   |
| 7               | 13 | 14 | 15  | 16  | 17  | 18   |
| 8               | 14 | 15 | 16  | 17  | 18  | 19   |

### 2.3.5.2.2. La longueur du rayon

Par contre l'évaluation du facteur LONG donne lieu à des erreurs beaucoup plus grandes. Ce facteur est calculé par l'une des deux expressions suivantes :

$$\begin{aligned}\text{LONG} &:= \text{DIM\_PIXEL} \\ \text{LONG} &:= \frac{Z * \text{FACT}}{\text{TG}}\end{aligned}$$

Le deuxième calcul engendre sur LONG une erreur proportionnelle à  $\frac{1}{\text{TG}^2}$  (ceci est une conséquence de la loi de dérivation d'un rapport  $\frac{\partial u}{\partial v} = \frac{\partial u \cdot v - u \cdot \partial v}{v^2}$ ). Fort heureusement le cas où l'erreur est très grande (TG petit et  $Z < \text{TG}$ ) se produit très peu fréquemment, et de plus le facteur étant borné par DIM\_PIXEL, les quelques mises à jour erronées portant sur un pixel donné ne produisent pas une erreur importante sur sa valeur finale. Des simulations ont été effectuées pour évaluer la précision du codage du facteur LONG, et il en résulte qu'un codage de l'ordre de 8 bits est très suffisant.

## 2.3.6. Traitement d'une sous-image

### 2.3.6.1. Principe

Une première étude architecturale et temporelle de la cellule, nous a permis de constater que le réseau était très mal utilisé : seulement un très petit nombre de cellules fonctionnent en parallèle. En effet, la cellule est beaucoup trop rapide par rapport à la vitesse d'acquisition des projections d'une telle machine de reconstruction. Cette vitesse est de l'ordre de 10 millisecondes par vecteur de projection (1000 points), et le temps de traitement d'un message par une cellule est d'environ 5 microsecondes.

Ceci nous a conduit à associer à un processeur, non plus un pixel, mais une sous-image, c'est à dire un petit tableau de  $k \times k$  pixels. L'aspect fonctionnel de la cellule change uniquement en ce qui concerne la phase de traitement du message : la cellule réalise un procédé itératif de manière à traiter séquentiellement tous les pixels

traversés par le rayon correspondant au message. L'algorithme cellulaire et la structure des messages sont identiques.

La figure 2.20. illustre les calculs de la cellule, et le traitement séquentiel des pixels, en fonction du point d'entrée du rayon.

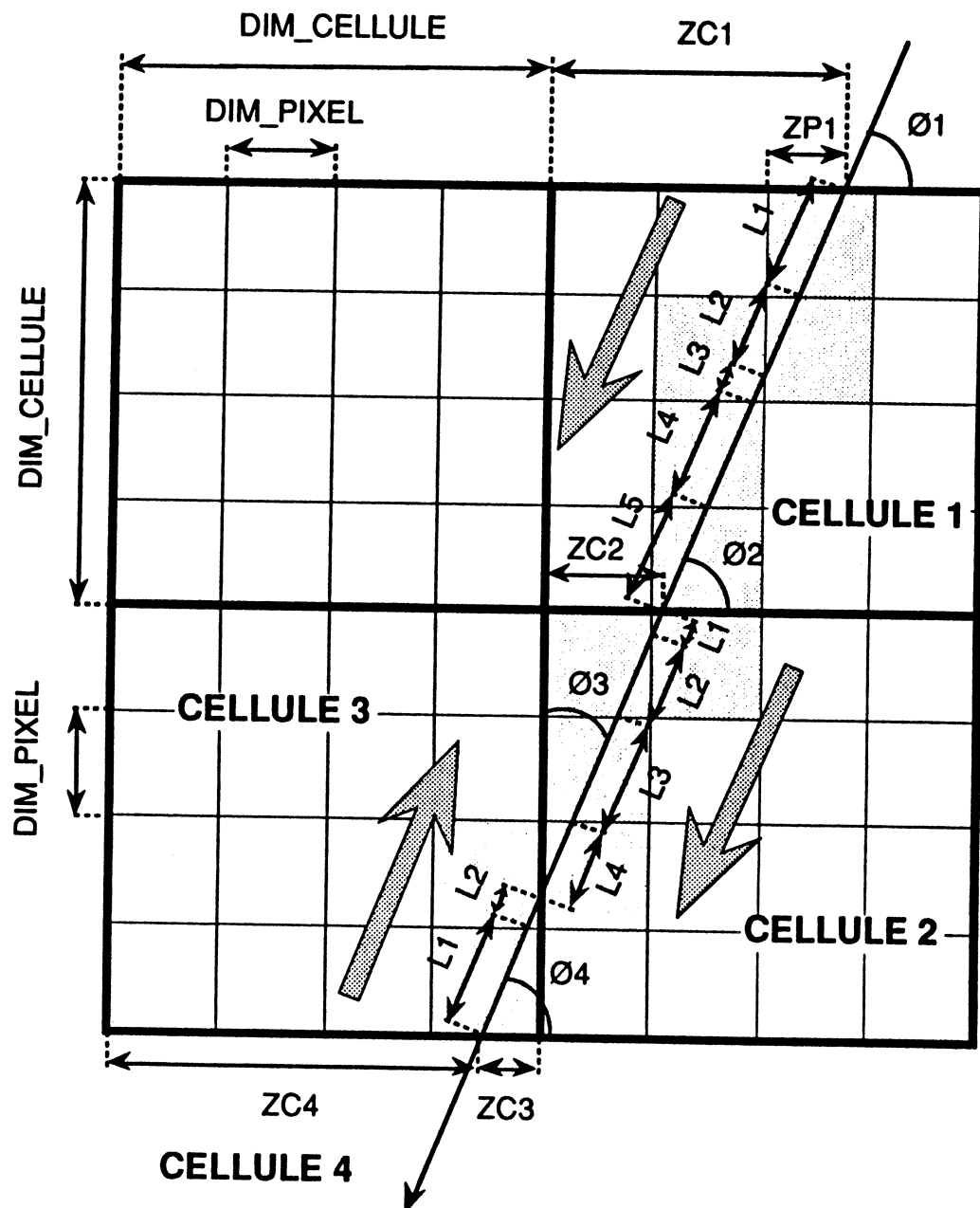


Figure 2.20. Traitements de la cellule

Les informations géométriques du message arrivant dans la cellule  $i$  sont  $ZC_i$  (point d'entrée dans la cellule), et des informations concernant l'angle  $\emptyset_i$ . La cellule  $i$ , après avoir déterminé la cellule destinataire du message (cellule  $i+1$ ), et calculé le point de sortie  $ZC_{(i+1)}$ , et les informations sur l'angle  $\emptyset_{(i+1)}$ , met à jour les pixels suivant le même sens que le rayon les traversent, ou le sens inverse, ceci en fonction des informations concernant l'angle  $\emptyset_i$ . Dans le cas, où l'angle d'incidence sur la face d'entrée du rayon dans le réseau est supérieur à  $\frac{\pi}{4}$ , nous ne calculons que les points d'entrée quand le rayon change de ligne ; ceci explique le fait que dans le cas où l'angle d'incidence sur la face d'entrée du rayon dans une cellule est inférieur à  $\frac{\pi}{4}$ , les pixels sont traités dans le sens contraire du sens de déplacement du message (cellule 3).  $ZP_1$  est la coordonnée du point d'entrée dans le premier pixel de la cellule 1.

### 2.3.6.2. L'algorithme de la cellule

L' algorithme est le même que lorsqu'un seul pixel est traité par cellule. Seules les phases de calcul d'adresse, et de traitement du message sont modifiées.

Dans la phase de calcul d'adresse, la cellule n'a plus à calculer la longueur du rayon dans la cellule ; cette information est inutile, nous nous intéressons ici aux longueurs du rayon dans les différents pixels. Par contre, la cellule doit initialiser correctement les informations concernant le premier pixel à traiter (ENTREE\_PIXEL : face d'entrée dans le pixel), et l'angle d'incidence sur celui-ci (TG, TC, S).

Le calcul d'adresse est :

**SI**  $\langle \emptyset \leq \frac{\pi}{4} \rangle$  **ALORS**      { sortie : côté adjacent à l'entrée }

|     |                    |
|-----|--------------------|
| ZS  | := DIM_CELLULE - Z |
| TGS | := TG              |
| TCS | := complément (TC) |
| SS  | := complément (S)  |

```

| ENTREE_PIXEL := SORTIE          { face de sortie du rayon}
| TC      := TCS
| S       := SS
SINON
| SI < ( $\frac{DIM\_CELLULE * TG}{FACT} \geq Z$ ) > ALORS {sortie:côté adjacent}
|   | ZS := ( $\frac{DIM\_CELLULE * TG}{FACT}$ ) - Z
|   | TGS := TG
|   | TCS := complément (TC)
|   | SS  := complément (S)
|   SINON { sortie : côté opposé à l'entrée }
|   | ZS := Z -  $\frac{(DIM\_CELLULE * TG)}{FACT}$ 
|   | TGS := TG
|   | TCS := TC
|   | SS  := S
|   FIN_SI
| ENTREE_PIXEL := ENTREE          { face d'entrée du rayon}
FIN_SI

```

La phase de traitement du message comprend plusieurs étapes :

- calculer d'adresse du premier pixel traversé  
en fonction de (ENTREE\_PIXEL, S et ZC)
- calculer le point d'entrée dans ce pixel ZP en fonction de Z
- tant que les pixels sur le chemin ne sont pas tous traités
  - ⇒ traiter le pixel :
    - calcul du point de sortie, des informations sur l'angle,
    - calcul de la longueur du rayon dans le pixel,
    - traitement suivant le type du message  
(épandage ou projection),
    - calcul de l'adresse du pixel suivant

L'algorithme détaillé du traitement du message est :

```

| { calcul de l'adresse du premier pixel }
| ZP := Z MODULO DIM_PIXEL

```



```

TANT_QUE < les pixels ne sont pas tous traités > FAIRE
    SI <  $(\emptyset \leq \frac{\pi}{4})$  > ALORS { sortie : côté adjacent à l'entrée }
        LONG :=  $\frac{(ZP * FACT)}{TG}$ 
        ZP    := DIM_PIXEL - ZP
        TC    := complément (TC)
        S     := complément (S)
    SINON
        SI <  $(\frac{DIM\_PIXEL * TG}{FACT} \geq ZP)$  > ALORS
            LONG :=  $\frac{(ZP * FACT)}{TG}$  { sortie : côté adjacent }
            ZP    :=  $\frac{(DIM\_PIXEL * TG)}{FACT} - ZP$ 
            TC    := complément (TC)
            S     := complément (S)
        SINON { sortie : côté opposé à l'entrée }
            LONG := DIM_PIXEL
            ZP    :=  $ZP - \frac{(DIM\_PIXEL * TG)}{FACT}$ 
        FIN_SI
    FIN_SI
    SI < (TYPE = épandage) > ALORS
         $a_i := a_i + LONG * INFO$ 
    SINON { TYPE = projection }
         $INFO := INFO + LONG * a_i$ 
    FIN_SI
    { calcul de l'adresse du pixel suivant}

```

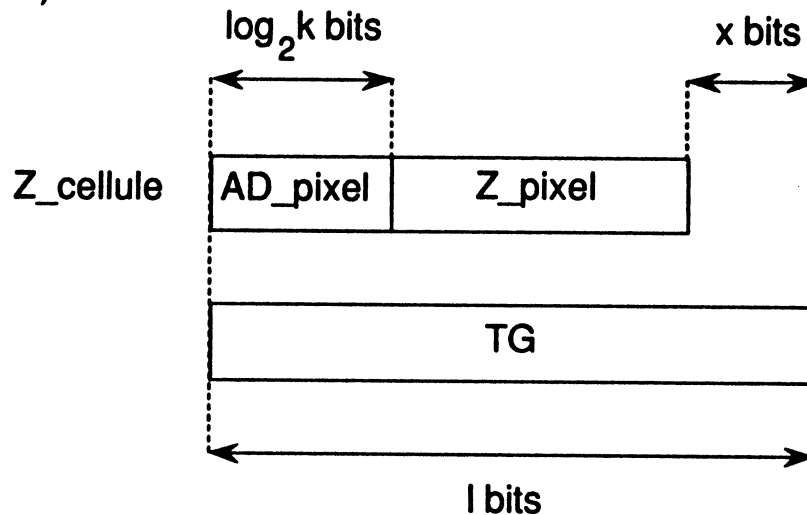
A l'intérieur d'une cellule, les calculs successifs des points d'entrée et des informations sur l'angle est fait de la même manière que les calculs entre les cellules.

### 2.3.6.3. Le codage des informations géométriques

De même que pour le cas où une cellule traite un seul pixel, nous allons établir une relation entre le codage des informations géométriques (Z et TG), la précision des calculs, et la taille de l'image

reconstruite (c'est à dire le nombre de cellules du réseau  $N^2$ , et le nombre de pixels par cellule  $k^2$ ).

L'information  $Z$  circulant entre les cellules, représente les coordonnées des points d'entrée dans les cellules ( $Z_{\text{cellule}}$ ). Elle est composée de deux parties : l'adresse du pixel, sur lequel arrive le rayon ( $AD_{\text{pixel}}$ ), et de la coordonnée du point d'entrée du rayon dans ce pixel ( $Z_{\text{pixel}}$ ).



**Figure 2.21.** Codage des informations géométriques

Nous supposons que l'information  $TG$  est codée sur  $l$  bits,  $Z_{\text{cellule}}$  sur  $(l - x)$  bits, et  $AD_{\text{pixel}}$  sur  $\log_2 k$  bits.

Les erreurs maximales sur les points d'entrée des pixels d'une cellule suivent les relations :

$$\Delta Z_1 = \Delta TG + \Delta Z_0$$

$$\Delta Z_2 = \Delta TG + \Delta Z_1$$

.....

$$\Delta Z_i = \Delta TG + \Delta Z_{i-1}$$

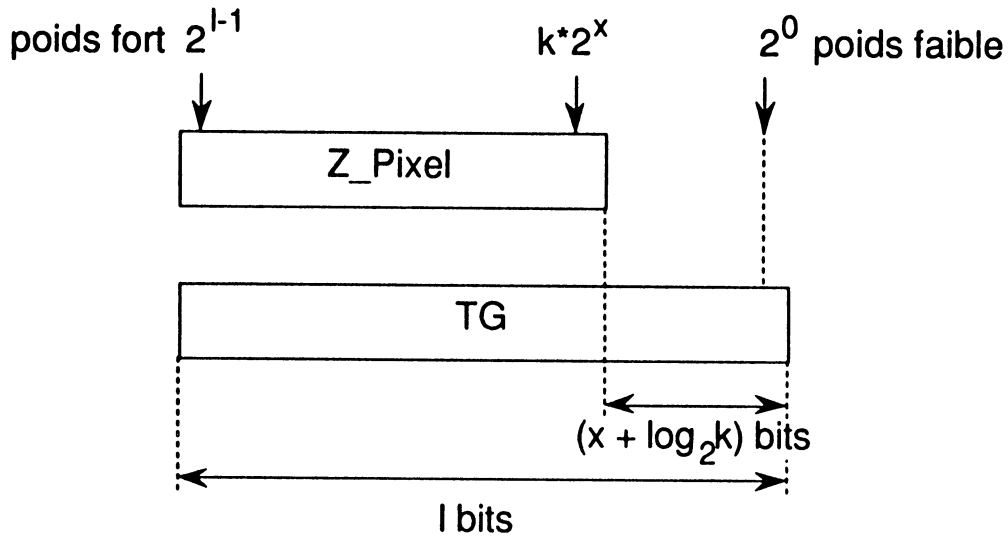
.....

$$\Delta Z_k = \Delta TG + \Delta Z_{k-1}$$

où  $Z_i$  est la coordonnée du point de sortie du pixel de la  $i^{\text{ème}}$  ligne, et  $\Delta Z_i$  son erreur.

Nous pouvons donc écrire :  $\Delta Z_k = k \cdot \Delta TG + \Delta Z_0$ .

TG est codé sur  $l$  bits, AD\_pixel sur  $\log_2 k$  bits, Z\_cellule sur  $(l-x)$  bits, et Z\_pixel sur  $(l-x-\log_2 k)$  bits (voir figure 2.22.).



**Figure 2.22.** Codage de TG et Z\_pixel

L'erreur maximale sur la tangente est  $\Delta TG = 2^{-1}$ .

L'erreur maximale sur le point d'entrée du premier pixel  $\Delta Z_0$ , dépend de l'erreur maximale sur le point d'entrée dans une cellule quelconque du réseau  $\Delta Z_N$ , calculé dans le cas où une cellule traite un seul pixel. Nous avons vu que  $\Delta Z_N \leq 2^x - 1$ , en posant  $x = 1 + \log_2 N$ , donc  $\Delta Z_0 \leq k * (2^x - 1)$ .

Il en découle la relation :  $\Delta Z_k \leq \frac{k}{2} + k * (2^x - 1)$ ,

ou encore  $\Delta Z_k \leq (k * 2^x) - \frac{k}{2}$

Si nous souhaitons que les informations contenues dans le champ Z aient au moins  $p$  bits corrects, il faut que  $\Delta Z_k < \frac{\text{DIM\_PIXEL}}{2^p}$  ou encore que  $\Delta Z_k \leq \frac{2^l}{2^p} - 1$ .

Il faut donc que  $\Delta Z_k \leq (k * 2^x) - \frac{k}{2} \leq \frac{2^l}{2^p} - 1$

ou  $k * 2^x + (1 - \frac{k}{2}) \leq \frac{2^l}{2^p} - 1$

Si  $k \geq 2$  alors  $k * 2^x + (1 - \frac{k}{2}) \leq k * 2^x$ , donc il faut que :

$$k * 2^x \leq \frac{2^l}{2^p}$$

Nous obtenons la relation :  $l \geq p + x + \log_2 k$ , ou  $l \geq p + 1 + \log_2 N + \log_2 k$ . Donc le nombre minimum de bits nécessaire pour coder TG, afin que p bits de Z soient corrects est :  $l = p + 1 + \log_2 N + \log_2 k$ . Et le nombre minimum de bits nécessaire pour coder Z\_pixel est p bits.

Le codage de l'information TG dépend donc du logarithme de la taille de l'image reconstruite ( $N * k$ ).

|                        |                                         |
|------------------------|-----------------------------------------|
| TG est codé sur        | $[(p + 1) + \log_2 k + \log_2 N]$ bits, |
| Z_cellule est codé sur | $[p + \log_2 k]$ bits,                  |
| AD_pixel est codé sur  | $[\log_2 k]$ bits,                      |
| Z_pixel est codé sur   | $[p]$ bits,                             |

Par exemple, si nous reconstruisons des images  $1024 * 1024$  à partir d'un réseau de  $32 * 32$  cellules ( $N = 32$ ), où chacune d'elle traite une image de  $32 * 32$  pixels ( $k = 32$ ), TG doit être codé sur au moins 16 bits pour avoir une précision de 5 bits sur les coordonnées des points d'entrée au niveau du pixel.

Lors de la réalisation de la cellule, le codage de la tangente a été choisi légèrement différent de celui présenté jusqu'à maintenant. L'information TG est codée sur l bits, et ses valeurs sont comprises entre 0 et  $2^{l-1}$  (le bit de poids fort à 1, et les autres à 0), et non pas  $2^l - 1$ . Le fait de coder la valeur 1 de la tangente réelle par une puissance de 2 :  $2^{l-1}$  simplifie grandement les calculs de la cellule, car FACT qui est défini comme étant la valeur maximale de TG est aussi une puissance de 2 ( $FACT = 2^{l-1}$ ). Il en découle que les évaluations de  $(ZP * FACT)$ , de  $\frac{DIM\_PIXEL * TG}{FACT}$ , et de  $\frac{DIM\_CELLULE * TG}{FACT}$  se ramènent à de simples décalages (FACT et DIM\_PIXEL sont des puissances de 2).

Par contre, la précision ayant diminuée d'un bit sur la tangente, elle a également diminuée d'un bit sur les points d'entrée des pixels.

Le tableau précédent (paragraphe 2.3.5.2.1.), définissant le codage, a été établi avec l'évaluation maximale de l'erreur au niveau

de chaque pixel, et en réalité les simulations fonctionnelles montrent qu'un codage moindre (de quelques bits) est suffisant pour obtenir des images de bonne qualité.

## 2.4. L'ORDINATEUR HÔTE

### 2.4.1. Son rôle

La machine de reconstruction comprend deux parties :

- le réseau cellulaire, qui permet de traiter de manière parallèle les opérations d'épandage et de projection,
- l'ordinateur hôte, qui pilote le réseau en gérant ses entrées et sorties.

L'ordinateur hôte réalise l'algorithme de reconstruction, il communique avec le réseau, en lui envoyant et en recevant les messages par l'intermédiaire des tampons périphériques, il synchronise les différentes étapes de calcul.

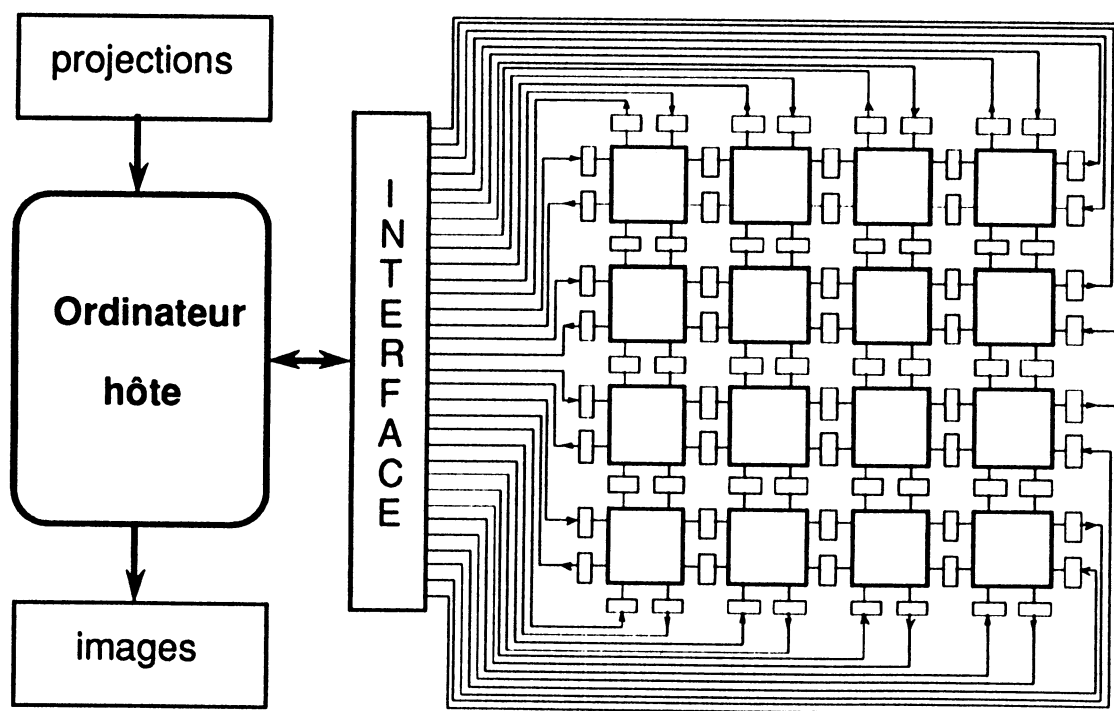


Figure 2.23. La machine de reconstruction

## 2.4.2. L'algorithme

Nous allons détailler l'algorithme de l'ordinateur hôte pour les méthodes de reconstruction suivantes : l'épandage des projections filtrées, et la technique itérative.

### 2.4.2.1. L'épandage des projections filtrées

Avant de décrire cet algorithme, il est nécessaire d'introduire un constructeur parallèle (PAR, FIN\_PAR). Les instructions comprises entre ces deux mots clés, sont exécutées en parallèle.

L'algorithme est :

```

initialiser du réseau
PAR
    envoyer tous les messages d'épandage
    recevoir tous les messages sortants du réseau
FIN_PAR
recevoir les résultats (image)
    
```

L'initialisation du réseau permet de mettre à zéro les valeurs des pixels de l'image complète.

L'ordinateur hôte envoie les messages d'épandage, au fur et à mesure que la partie acquisition lui fournit les valeurs de projection. Il doit générer les parties géométriques des messages, et écrire chaque message dans le "bon" tampon périphérique d'entrée. En même temps, il doit scruter les tampons périphériques de sortie afin de recevoir les messages ; les informations contenues dans ces messages d'épandage sortants ne lui sont pas utiles, mais cette réception est nécessaire pour libérer ces tampons.

Une fois le traitement réalisé, il reste à récupérer l'image stockée dans le réseau afin de la visualiser. Le procédé utilisé pour sortir l'image du réseau, ou charger le réseau avec une image est expliqué plus loin dans le rapport (voir paragraphe 2.5.2.).

### 2.4.2.2. La technique itérative

Pour la technique itérative, l'ordinateur réalise une succession d'itérations, comprenant une phase d'épandage, et une phase de projection. Il doit également, à chaque itération, décider de l'arrêt ou de la poursuite du procédé répétitif.

L'algorithme est :

```
initialiser du réseau
REPETER
  PAR
    envoyer tous les messages d'épandage
    recevoir tous les messages sortants du réseau
  FIN_PAR
  PAR
    envoyer tous les messages de projection
    recevoir tous les messages sortants du réseau
  FIN_PAR
  comparer les valeurs calculées aux valeurs mesurées
JUSQU'A < valeurs calculées  $\approx$  valeurs mesurées >
recevoir les résultats (image)
```

Les informations contenues dans les messages de projection sortants du réseau sont lues par l'ordinateur hôte, en particulier le champ INFO qui contient la valeur calculée de projection. Cette information est comparée à la valeur mesurée de projection, pour chaque rayon, et selon cette différence, l'ordinateur poursuit ou stoppe le procédé.

## 2.5. LES ENTREES ET SORTIES DES IMAGES

### 2.5.1. Introduction

La sortie de l'image stockée dans le réseau est une opération évidemment nécessaire en fin de reconstruction, pour pouvoir la

visualiser. Le chargement d'une image sur le réseau peut être intéressant dans le cas, où nous voulons simplement calculer ses projections ; donc cette opération doit aussi être possible grâce à notre architecture cellulaire.

Le réseau n'a aucune orientation fixée à priori, les images entrent et sortent du réseau par l'intermédiaire des tampons, ligne par ligne, ou colonne par colonne.

Nous avons besoin d'introduire de nouveaux types de message afin de réaliser ces opérations d'entrée et sortie. Des messages de type `entrée_image` et `sortie_image` sont nécessaires, ce sont des messages de commande véhiculés jusqu'à chaque cellule, qui vont déclencher un traitement spécifique. Des messages de type transparence sont également utiles pour acheminer les messages, contenant les valeurs de pixels. Un message de type transparence est simplement recopié dans le tampon de sortie opposé au tampon d'entrée, qui le contenait (voir figure 2.24.).

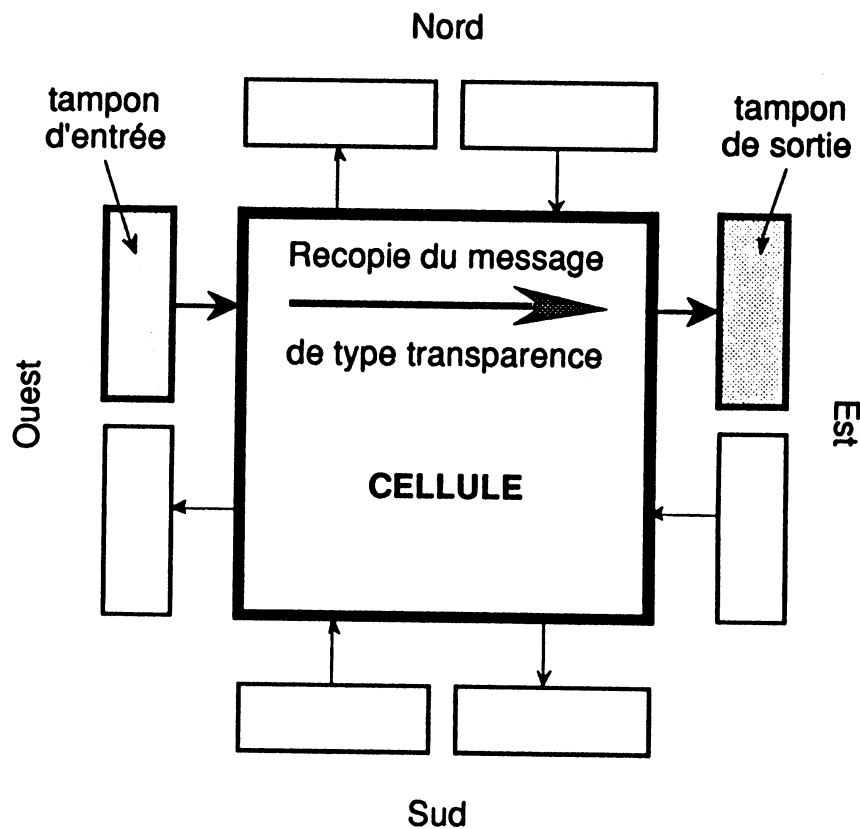


Figure 2.24. Message de type transparence



La phase de traitement du message de l'algorithme de la cellule n'est pas effectuée, lorsque le message reçu est du type transparence.

## **2.5.2. Sortie de l'image**

### **2.5.2.1. Principe**

La sortie de l'image contenue dans le réseau s'effectue ligne par ligne, ou colonne par colonne. Ce choix a été fait pour ne pas privilégier une orientation du réseau plutôt qu'une autre.

Pour sortir la  $q^{\text{ième}}$  ligne dans l'ordre Ouest-Est, l'ordinateur hôte envoie un message de type `sortie_image` au réseau, en l'écrivant dans le tampon d'entrée périphérique de la cellule contenant cette ligne.

Le message doit avoir un champ TG de valeur nulle (pour que le message traverse le réseau horizontalement,  $\emptyset = \frac{\pi}{2}$ ), et un point d'entrée dans la cellule Z, tel que cette coordonnée permet d'accéder à la ligne souhaitée. Le routage des messages de type `sortie_image` est fait de même façon que le routage des autres messages.

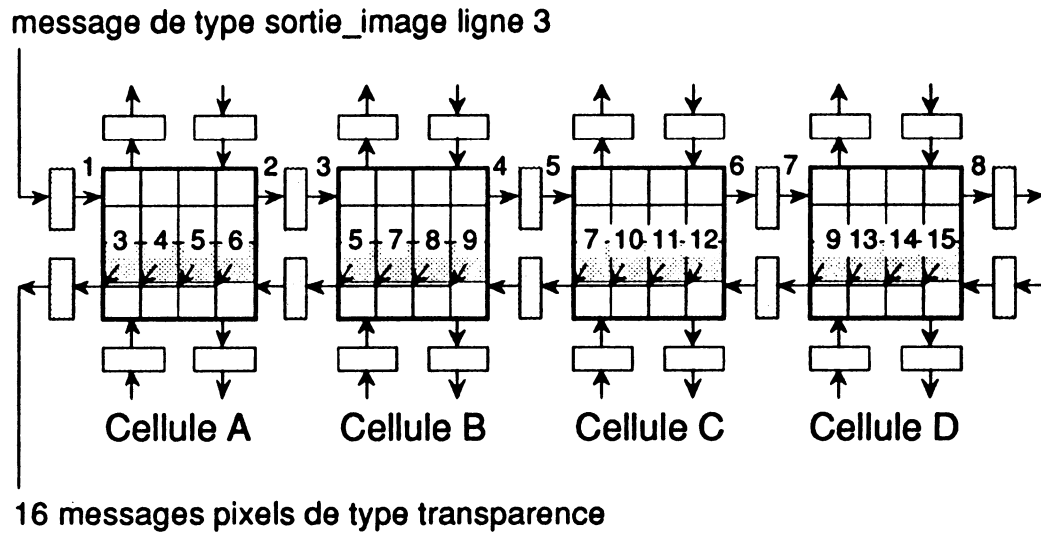
Une cellule recevant un message de type `sortie_image`, le passe à sa voisine avant tout traitement. Ensuite, elle émet par le tampon de sortie situé du même côté que le tampon d'entrée, par lequel elle a reçu le message de type `sortie_image`, une série de k messages de type transparence (en supposant qu'une cellule traite  $k \times k$  pixels), dont le champ INFO contient la valeur d'un pixel de la ligne adressée par la valeur du champ Z.

Après l'envoi d'un message de type `sortie_image`, l'ordinateur hôte n'a plus qu'à recevoir par le tampon de sortie adjacent à celui qui lui a servi pour l'envoi, m messages contenant les m valeurs des pixels de la ligne ( $m \times m$  est la taille de l'image traitée,  $m = N \times k$ , où  $N \times N$  est le nombre de cellules, et  $k \times k$  le nombre de pixels par cellule). Il doit aussi récupérer le message initial de type `sortie_image`, qui a traversé tout le réseau.

Pour sortir l'image entière (les m lignes), l'ordinateur hôte répète m fois ce procédé. Il est possible de sortir N lignes de l'image en parallèle,

et de cette manière, le temps nécessaire à la sortie totale est égal à  $k$  fois le temps nécessaire pour sortir une ligne.

La figure 2.25. illustre la sortie d'une ligne de l'image dans le cas où  $m = 16$  (image  $16 \times 16$ ), et  $k = 4$  (une cellule =  $4 \times 4$  pixels). Les chiffres inscrits sur le schéma indiquent l'ordre chronologique des envois et réceptions de messages.



**Figure 2.25.** Principe de sortie de l'image

Le réseau dans ce cas a un fonctionnement qui s'apparente au fonctionnement de type systolique, bien que les durées de chaque opération puissent être différentes. Nous allons expliquer plus précisément le déroulement de la sortie d'une ligne. Les pixels de la ligne considérée sont nommés de la gauche vers la droite par  $a_1, a_2, \dots, a_{16}$ . Le tableau suivant donne le séquençement des actions.

| instants | actions                                                            |
|----------|--------------------------------------------------------------------|
| $t=0$    | le maître écrit le message de type sortie_image                    |
| $t=1$    | la cellule A lit le message de type sortie_image                   |
| $t=2$    | la cellule A passe le message à sa voisine                         |
| $t=3$    | la cellule A sort $a_1$ , et B lit le message de type sortie_image |
| $t=4$    | A : sortie de $a_2$ , B : passage du message                       |
| $t=5$    | A : sortie de $a_3$ , B : sortie de $a_5$ , C : lecture du message |
| $t=6$    | A : sortie de $a_4$ , C : passage du message                       |

|       |                                                                                                           |
|-------|-----------------------------------------------------------------------------------------------------------|
| t=7   | A : passage de a <sub>5</sub> , B : sortie de a <sub>6</sub> , C : sortie de a <sub>9</sub> , D : lecture |
| t=8   | A : passage de a <sub>6</sub> , B : sortie de a <sub>7</sub> , D : passage du message                     |
| t=9   | A : passage de a <sub>7</sub> , B : sortie de a <sub>8</sub> , D : sortie de a <sub>13</sub>              |
| t=10  | A : passage de a <sub>8</sub> , B : passage de a <sub>9</sub> , C : sortie de a <sub>10</sub>             |
| t=11  | A : passage de a <sub>9</sub> , B : passage de a <sub>10</sub> , C : sortie de a <sub>11</sub>            |
| ..... | ..... etc                                                                                                 |

Le fait que chaque cellule transmette le message de type `sortie_image`, avant de sortir les valeurs des pixels, permet d'anticiper sur l'ensemble du travail. Il y a ainsi un plus grand parallélisme, et un gain de temps sur la sortie de la ligne. Les valeurs des pixels commencent à sortir très peu de temps après l'envoi du message de type `sortie_image`, car ils sortent du même côté. Cette solution est meilleure que celle consistant à les faire sortir du côté opposé, car elle évite un temps de latence important correspondant au temps de traversée du réseau du message de type `sortie_image`.

### 2.5.2.2. L'ordinateur hôte

Détaillons l'algorithme de l'ordinateur hôte réalisant la sortie de l'image.

```

POUR numéro_cellule := 1 A N FAIRE
  POUR numéro_ligne := 1 A k FAIRE
    envoyer un message de type sortie_image
    POUR numéro_colonne := 1 A m FAIRE
      recevoir un message contenant une valeur de pixel
    FIN_POUR
    recevoir le message de type sortie_image
  FIN_POUR
FIN_POUR
  
```

Cet algorithme est purement séquentiel. Il est clair que N lignes peuvent être sorties en parallèle (une par étage de cellules).

### 2.5.2.3. La cellule

L'algorithme de la cellule reste le même, seule la phase de traitement du message est modifiée afin de prendre en compte, et de traiter les messages de type *sortie\_image* et *entrée\_image*.

Les pixels de la ligne concernée sont adressés par le même calcul itératif effectué pour les messages d'épandage ou de projection.

Les calculs effectués lors du traitement au niveau du pixel sont :

#### CHOIX

```

< TYPE = épandage >
  | ai := ai + LONG * INFO
< TYPE = projection >
  | INFO := INFO + LONG * ai
< TYPE = sortie_image >
  | émettre un message contenant la valeur du pixel ai
    par le côté du message reçu
< TYPE = entrée_image >
  | { traitement entrée_image }
```

#### FIN\_CHOIX

### 2.5.3. Entrée d'une image dans le réseau

#### 2.5.3.1. Principe

L'entrée d'une image sur le réseau s'effectue de la façon analogue à la sortie.

Pour entrer la *q*<sup>ième</sup> ligne dans l'ordre Ouest-Est, l'ordinateur hôte envoie un message de type *entrée\_image* au réseau, en l'écrivant dans le tampon d'entrée périphérique de la cellule contenant la ligne. Le message doit contenir un champ TG de valeur nulle (pour que le message traverse le réseau horizontalement,  $\emptyset = \frac{\pi}{2}$ ), et la coordonnée d'accès de la ligne concernée dans le champ Z.

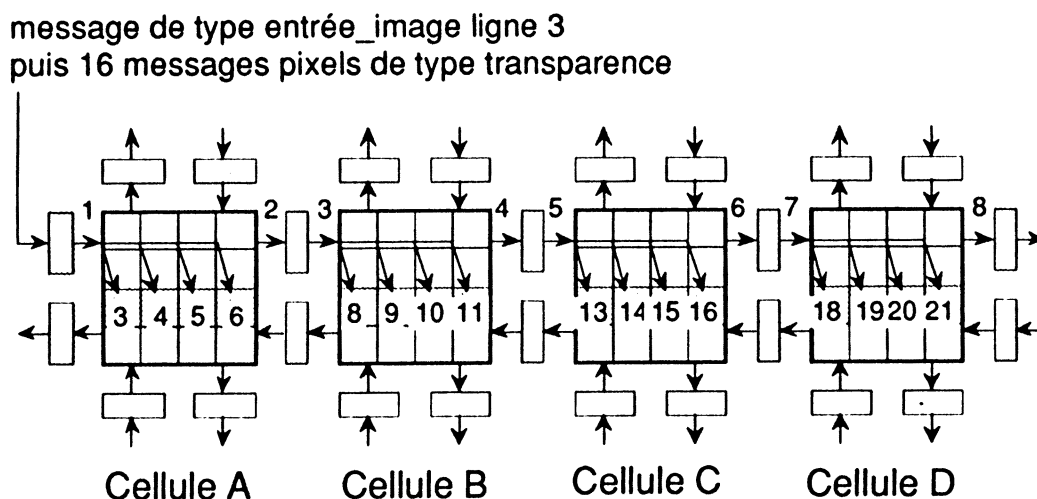
Après l'envoi d'un message de type `entrée_image`, l'ordinateur hôte émet  $m$  messages de type transparence contenant les  $m$  valeurs des pixels de la ligne ( $m \times m$  est la taille de l'image traitée,  $m = N \times k$ , où  $N \times N$  est le nombre de cellules, et  $k \times k$  le nombre de pixels par cellule).

Une cellule recevant un message de type `entrée_image`, le passe à la cellule voisine sur la ligne, reçoit une série de  $k$  messages de type transparence contenant chacun d'eux la valeur d'un pixel de la ligne concernée dans le champ INFO. Ensuite, elle transmet les autres messages de type transparence contenant les valeurs des autres pixels de la ligne.

L'ordinateur hôte doit aussi récupérer le message initial de type `entrée_image`, qui a traversé tout le réseau.

Pour entrer une image entière (les  $m$  lignes), l'ordinateur hôte répète  $m$  fois ce procédé. Il est également possible d'entrer  $N$  lignes de l'image en parallèle, et de cette manière, le temps nécessaire à l'entrée complète est égal à  $k$  fois le temps nécessaire pour entrer une ligne.

La figure 2.26. illustre l'entrée d'une ligne de l'image dans le cas où  $m = 16$  (image  $16 \times 16$ ), et  $k = 4$  (une cellule =  $4 \times 4$  pixels). Les chiffres inscrits sur le schéma indiquent l'ordre chronologique des envois et réceptions de messages.



**Figure 2.26.** Principe d'entrée d'une image

Nous allons expliquer plus précisément le déroulement de l'entrée d'une ligne. Les pixels de la ligne considérée sont nommés de la

gauche vers la droite par  $a_1, a_2, \dots, a_{16}$ . Le tableau suivant donne le séquençement des opérations réalisées.

| temps | actions                                                                    |
|-------|----------------------------------------------------------------------------|
| t=0   | le maître écrit le message de type entrée_image                            |
| t=1   | la cellule A lit le message de type entrée_image                           |
| t=2   | la cellule A passe le message à sa voisine                                 |
| t=3   | la cellule A reçoit $a_1$ , et B lit le message de type entrée_image       |
| t=4   | A : réception de $a_2$ , B : passage du message                            |
| t=5   | A : réception de $a_3$ , C : lecture du message                            |
| t=6   | A : réception de $a_4$ , C : passage du message                            |
| t=7   | A : passage de $a_5$ , D : lecture du message                              |
| t=8   | A : passage de $a_6$ , B : réception de $a_5$ , D : passage du message     |
| t=9   | A : passage de $a_7$ , B : réception de $a_6$                              |
| t=10  | A : passage de $a_8$ , B : réception de $a_7$                              |
| t=11  | A : passage de $a_9$ , B : réception de $a_8$                              |
| t=12  | A : passage de $a_{10}$ , B : passage de $a_9$                             |
| t=13  | A : passage de $a_{11}$ , B : passage de $a_{10}$ , C : réception de $a_9$ |
| ..... | ..... etc                                                                  |

### 2.5.3.2. L'ordinateur hôte

L'algorithme de l'ordinateur hôte réalisant le chargement d'une image est :

```
POUR numéro_cellule := 1 A N FAIRE
  POUR numéro_ligne := 1 A k FAIRE
    envoyer un message de type entrée_image
    POUR numéro_colonne := 1 A m FAIRE
      envoyer un message contenant une valeur de pixel
    FIN_POUR
    recevoir le message de type entrée_image
  FIN_POUR
FIN_POUR
```

De même que pour la sortie de l'image, il est envisageable d'entrer N lignes en parallèle (une par étage de cellules).

### 2.5.3.3. La cellule

Les calculs effectués lors du traitement au niveau du pixel sont :

#### CHOIX

```
< TYPE = épandage >  
  |  $a_i := a_i + \text{LONG} * \text{INFO}$   
< TYPE = projection >  
  |  $\text{INFO} := \text{INFO} + \text{LONG} * a_i$   
< TYPE = sortie_image >  
  | émettre un message contenant la valeur du pixel  $a_i$   
< TYPE = entrée_image >  
  | recevoir un message contenant la valeur du pixel  $a_i$ 
```

#### FIN\_CHOIX

Les méthodes exposées, et les algorithmes qui viennent d'être décrits ont été simulés. Les principaux résultats obtenus font l'objet de chapitre suivant.







## ***Chapitre 3***

---

### **Les simulations**



|                                                           |            |
|-----------------------------------------------------------|------------|
| <b>3.1. Introduction.....</b>                             | <b>129</b> |
| <b>3.2. Les simulations fonctionnelles OCCAM.....</b>     | <b>129</b> |
| 3.2.1. Le langage OCCAM .....                             | 129        |
| 3.2.2. Le Transputer.....                                 | 130        |
| 3.2.3. Structure du programme de simulation .....         | 131        |
| 3.2.4. Les résultats : Images.....                        | 134        |
| 3.2.4.1. Reconstruction d'un cercle homogène centré ..... | 134        |
| 3.2.4.1.1. Etude suivant la méthode utilisée .....        | 136        |
| 3.2.4.1.2. Etude suivant le nombre de mesures....         | 141        |
| 3.2.4.1.3. Etude suivant la précision du codage..         | 143        |
| 3.2.4.2. Reconstruction d'images complexes.....           | 146        |
| 3.2.5. Conclusion.....                                    | 147        |
| <b>3.3. Les simulations comportementales HELIX..</b>      | <b>148</b> |
| 3.3.1. Silvar-Lisco le simulateur HELIX .....             | 148        |
| 3.3.2. Structure du programme de simulation.....          | 149        |
| 3.3.3. Les résultats : performances du réseau .....       | 149        |
| 3.3.3.1. Etude du temps de reconstruction .....           | 149        |
| 3.3.3.2. Influence du nombre de projections .....         | 151        |
| 3.3.3.3. Influence du type de traitement .....            | 152        |
| 3.3.3.4. Traitement d'une sous-image.....                 | 153        |
| 3.3.4. Conclusion.....                                    | 155        |



## **3.1. INTRODUCTION**

Pour valider notre machine de reconstruction, nous avons fait deux types de simulation.

Les premières ont été de pures simulations fonctionnelles sans tenir compte de l'aspect temporel. Elles ont été réalisées avec le langage parallèle OCCAM, et nous ont permis d'une part de valider l'architecture cellulaire, et d'autre part de vérifier et d'étudier l'aspect algorithmique des méthodes parallèles de reconstruction [ANS86], [OBJ87a]. Grâce aux images reconstruites lors de ces simulations, nous avons pu comparer l'efficacité des méthodes proposées, et observer l'influence de la précision du codage des informations.

Les secondes ont été des simulations comportementales, afin d'examiner l'aspect temporel de l'architecture et ses performances. Elles ont permis d'étudier l'activité du réseau et le cheminement des messages en fonction de la fréquence d'envoi des messages, et de déterminer les utilisations optimales du réseau. Elles ont été réalisées grâce au simulateur HELIX, du système intégré de conception assistée par ordinateur (CAO) : Silvar-Lisco.

## **3.2. LES SIMULATIONS FONCTIONNELLES OCCAM**

### **3.2.1. Le langage OCCAM**

OCCAM est un langage parallèle issu de CSP ("Communicating Sequential Processes"), il permet de décrire très simplement des processus concurrents [HOA78]. Les synchronisations, et communications entre processus sont réalisés grâce à des canaux unidirectionnels. Nous avons choisi ce langage pour réaliser les simulations fonctionnelles de notre machine, car notre architecture de type matricielle se prête bien à une description en terme de processus

concurrents. En plus de cette facilité de programmation, l'efficacité de simulation est très bonne.

Dans ce langage parallèle, il y a en plus des constructeurs habituels des langages de type séquentiel (par exemple Pascal), quelques constructeurs spécifiques [INM84]. Ces constructeurs sont PAR, SEQ et ALT :

- ⇒ PAR annonce une série d'instructions, qui seront exécutées de manière concurrente,

- ⇒ SEQ annonce une suite séquentielle d'instructions (constructeur implicite dans un langage de type séquentiel)

- ⇒ ALT est un constructeur alternatif permettant d'activer tel ou tel processus en fonction d'événements extérieurs et de conditions.

Parmi les autres constructeurs, il y a un constructeur conditionnel (IF), des constructeurs répétitifs (WHILE et REPEAT), et un constructeur de multiplexage (FOR).

### 3.2.2. Le Transputer

Dans un premier temps, pour réaliser les simulations fonctionnelles, nous avons utilisé le VAX780 disponible au CIME (Centre Interuniversitaire de Micro-Electronique). Mais à cause du manque d'efficacité dans la génération de code à partir des sources OCCAM, les simulations étaient très longues et donc coûteuses ; c'est pourquoi nous avons opté pour une autre solution.

Nous nous sommes alors procurés une carte B004 comprenant un Transputer, et deux mégaoctets de mémoire RAM, que nous avons monté sur un compatible IBM/PC-AT. Ce matériel constitue grâce au microprocesseur Transputer qui a été spécialement développé pour le langage OCCAM par INMOS afin d'obtenir d'excellentes performances d'exécution, et au système de développement du Transputer (Transputer Development System), un excellent outil très efficace pour simuler fonctionnellement une machine de type parallèle [INM86a]. Le Transputer possède de la mémoire locale, et quatre liens de communication permettant de le connecter à d'autres Transputers. Le Transputer de cette carte est un B414 comprenant une unité centrale

32 bits de performance 7,5 MIPS, deux kilooctets de RAM, et fonctionnant à une vitesse d'horloge de 15 MHz [INM86b].

### 3.2.3. Structure du programme de simulation

La machine complète de reconstruction a été simulée. La cellule de base de l'architecture parallèle, et l'ordinateur hôte ont été décrits en tant que processus OCCAM.

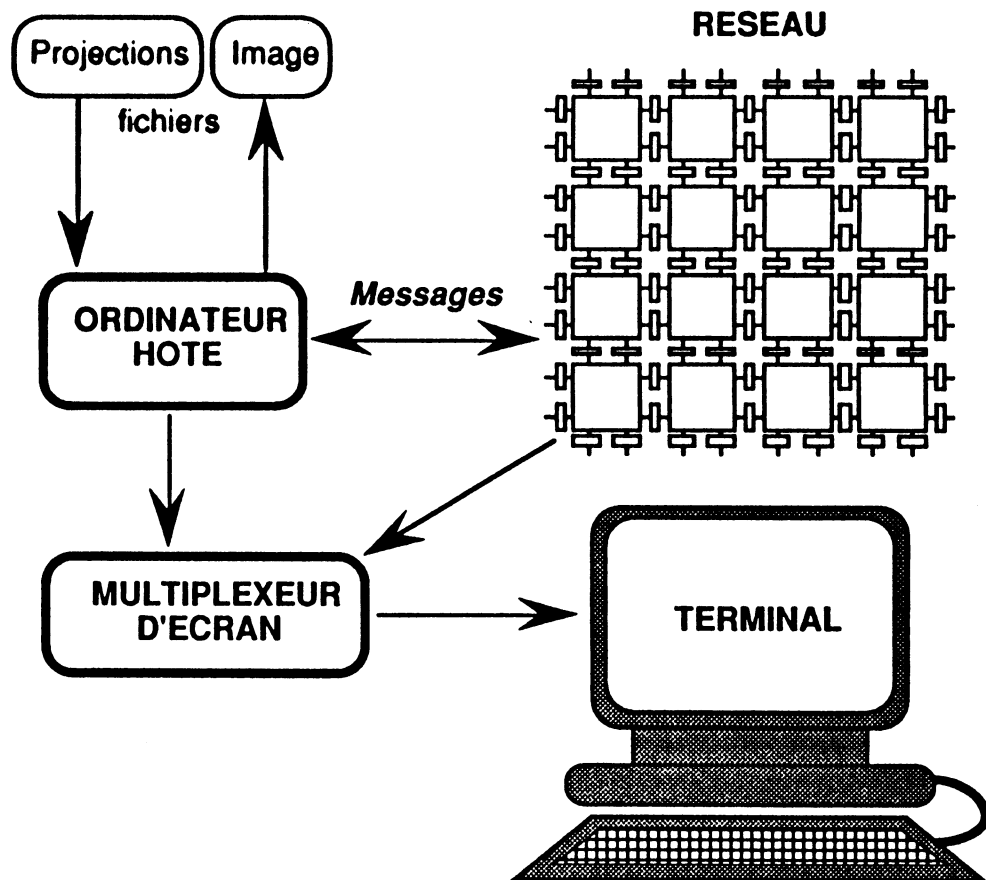


Figure 3.1. Les différents processus

La figure 3.1. illustre les différents processus du programme qui sont :

→ Le processus "ordinateur\_hôte", qui à partir d'un fichier source contenant les projections de l'objet à reconstruire, génère les messages à transmettre au réseau, et lui les communique. Il réalise



l'algorithme de reconstruction voulu, et en fin traitement récupère l'image calculée par le réseau, et l'écrit dans un fichier.

→ Le processus "réseau" composé d'un ensemble de processus parallèles "cellule" et "tampon" réalise la reconstruction.

→ Le processus "multiplexeur\_d'écran" permet à tous les processus ("ordinateur\_hôte", "cellule", et "tampon") d'accéder à l'écran, et d'afficher des traces afin de suivre le déroulement du programme.

La structure générale du programme est la suivante :

```

PROC prog_simulation_machine (CHAN KEYBOARD, SCREEN, [4]CHAN from, to)
  {{{ paramètres (réseau, reconstruction) }}}
  {{{ déclarations (constantes, variables et canaux) }}}
  {{{ procédures écran/clavier
      (affichage, lecture clavier et écriture écran) }}}
  {{{ procédures fichiers
      (ouverture, fermeture, lecture et écriture) }}}
  {{{ fonctions trigonométriques (cosinus, sinus et tangente) }}}
  PROC multiplexeur_d'écran :
  PROC tampon :
  PROC cellule (adresse) :
  PROC ordinateur_hôte :
  PAR                                     {{{ programme principal }}}
    multiplexeur_d'écran
    SEQ
      PAR
        PAR i = 0 FOR n                 {{{ réseau n*n }}}
          PAR j = 0 FOR n
            cellule (i,j)
          ordinateur_hôte
        arrêter le multiplexeur_d'écran
      :
  :

```

PROC introduit la définition d'un processus, l'appel étant réalisé par le nom du processus.

La structure du processus "cellule" est la suivante :

```

PROC cellule (VAL INT n°_ligne, n°_colonne)
  PAR
    tampons_d'entrée
    SEQ                                     {{{ programme principal }}}
      sélection d'un tampon d'entrée
      WHILE TRUE
        SEQ
          test du drapeau du tampon d'entrée sélectionné
          IF le tampon d'entrée sélectionné est plein
            SEQ
              lecture du message du le tampon d'entrée
              calcul de l'adresse de sortie
              test du drapeau du tampon de sortie destinataire
              IF le tampon de sortie est vide
                SEQ
                  vidage du tampon d'entrée
                  traitement du message
                  écriture du message
                  remplissage du tampon de sortie
              sélection du tampon d'entrée suivant
  :
```

Nous retrouvons exactement l'algorithme de la cellule décrit au chapitre 2, paragraphe 2.3.2., aux primitives OCCAM près.

Le processus complet "cellule" est donné en annexe 3.

Le programme est articulé autour de plusieurs paramètres, permettant de spécifier les informations caractéristiques du réseau :

- sa taille : le nombre de cellules est  $n \times n$ , [n]
- le nombre de pixels traités par une cellule ( $k \times k$ ), [k]
- et le nombre de bits utilisés pour le codage des informations géométriques (Z et TG); [nb\_bits\_z, nb\_bits\_tg]

et de définir très simplement les informations de reconstruction :

- le nombre de projections, *[nbpro]*
- le nombre de rayons par projection, *[nbray]*
- et le traitement : reconstruction par épandage des projections filtrées ou par la méthode itérative, ou simple projection. *[traitement]*

### **3.2.4. Les résultats : images**

Avant la présentation des résultats, il faut dire que le programme ne peut être exécuté qu'avec un réseau, et une image de tailles limitées, ceci est dû à la limite de la mémoire de la carte B004 : deux mégaoctets. Le réseau de taille maximale est  $64 \times 64$  cellules, où chacune d'elle traite un seul pixel. L'image de taille maximale que nous pouvons reconstruire est de  $512 \times 512$  pixels (par exemple, avec un réseau  $4 \times 4$ , où chaque cellule traite  $128 \times 128$  pixels).

Pour étudier les méthodes de reconstruction, et l'influence des paramètres sur les images reconstruites, il a été fait toute une gamme de reconstructions, à partir des projections d'un cylindre homogène centré. Ensuite, nous présenterons des résultats plus complexes.

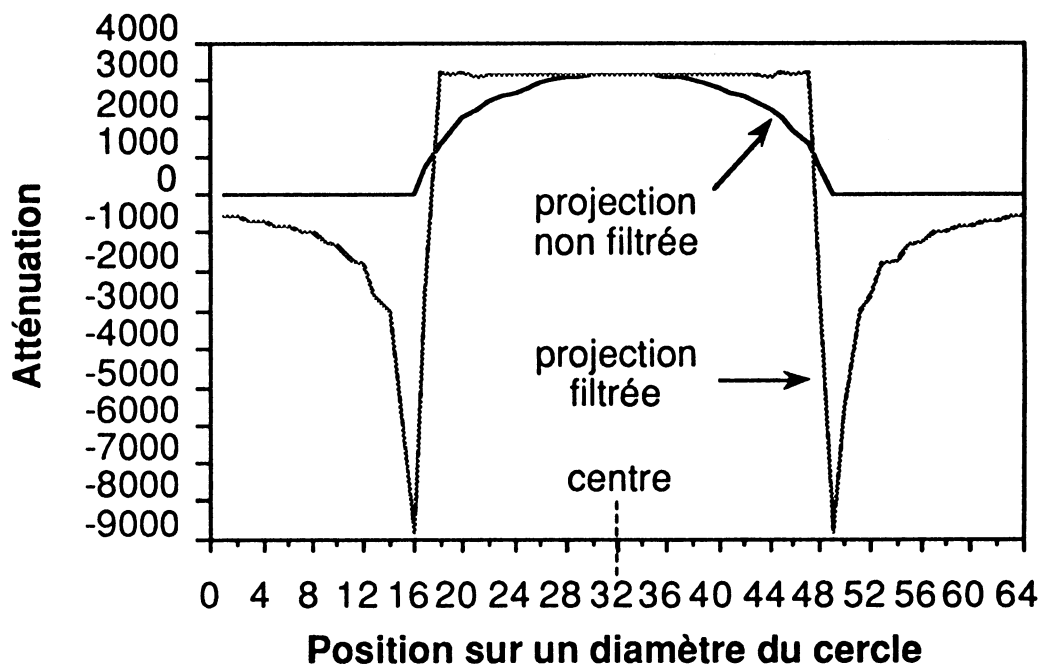
#### **3.2.4.1. Reconstruction d'un cercle homogène centré**

Les reconstructions suivantes ont été faites avec les projections d'un cylindre homogène centré de diamètre 32 (les projections sont identiques, quel que soit l'angle de projection). Les projections sont envoyées au réseau sur  $180^\circ$  à intervalle d'angle régulier. Le réseau comprend  $4 \times 4$  cellules ( $n = 4$ ), et chaque cellule traite  $16 \times 16$  pixels ( $k = 16$ ). Les images reconstruites sont donc de taille  $64 \times 64$  ( $m = n \times k = 64$ ), elles représentent un cercle homogène centré de diamètre 32 [LAT88c]. Les traitements d'épandage, ou de projection, pour des images de cette taille nécessite environ 2 à 3 minutes sur le simulateur développé en OCCAM.

Le fait que ces images soient de taille réduite, et que l'objet à reconstruire soit symétrique par rapport au centre de l'image simplifie

considérablement la mise en œuvre de la reconstruction. Cependant, les résultats obtenus sont très significatifs, et leur extrapolation à des images plus grandes et plus complexes est possible.

La figure 3.2. illustre une projection du cercle centré homogène sous un angle quelconque (une projection brute non filtrée, et une filtrée). L'abscisse du graphique donne la position sur un axe quelconque, passant par le centre du cercle de diamètre 64, inscrit dans l'image 64\*64. Le point 32 correspond au centre de l'image, et au centre du cercle de diamètre 32 que nous voulons reconstruire. Les valeurs en ordonnée représentent les valeurs de projection ; ce sont des atténuations relatives. Les valeurs de la projection non filtrée correspondent aux mesures faites à partir un cylindre réel, elles sont toutes positives pour les rayons le traversant, d'autant plus qu'ils passent proches du centre (abscisse 32), et nulles quand ils ne traversent pas le cylindre. La projection filtrée a été calculée en appliquant un filtre sur la projection non filtrée (voir chapitre 1, paragraphe 1.2.2.4.).



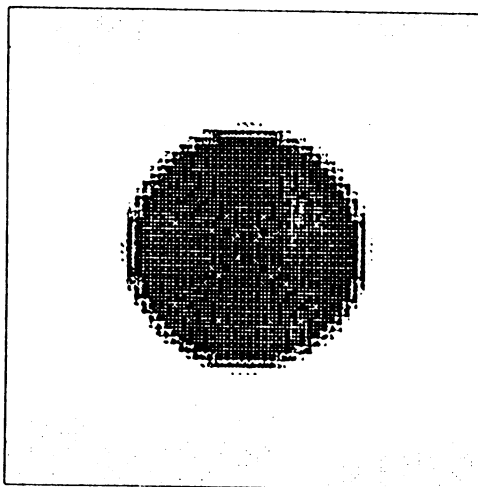
**Figure 3.2.** Les projections du cercle homogène centré

L'affichage des images 64\*64 reconstruites est réalisée grâce à une palette de 64 niveaux de gris. Chaque niveau de gris correspond à une densité de l'objet, la couleur blanche signifie une densité nulle (pas de matière), et une couleur noire une densité maximale. Chaque pixel de l'image est composé de 8\*8 points, et en fonction de la valeur du pixel, un certain nombre de points choisis de manière aléatoire, parmi les 64 points du pixel sont affichés en noir.

### 3.2.4.1.1. Etude suivant la méthode utilisée

Dans ce paragraphe, nous présentons et comparons les résultats obtenus à partir des deux méthodes de reconstruction implémentées sur le réseau : l'épandage des projections filtrées, et la technique itérative de reconstruction. Ces simulations ont été réalisées pour comparer les deux méthodes afin d'en choisir éventuellement une seule, et estimer le nombre d'itérations nécessaires pour la technique itérative.

Le nombre de projections est 64 (nbpro), le nombre de rayons par projection est 64 (nbray), et les informations sont codées sur 10 bits (nb\_bits\_z et nb\_bits\_tg).

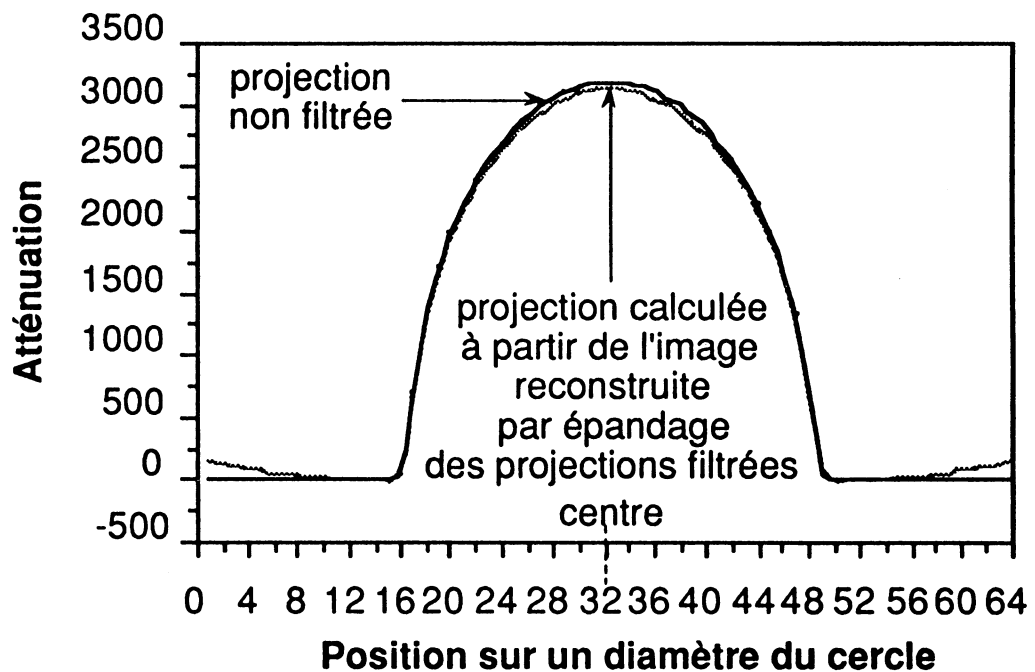


**Figure 3.3.** Image obtenue par épandage des projections filtrées

La figure 3.3. est l'image résultat obtenue par épandage des projections filtrées. Il ne faut pas tenir compte des pixels hors du cercle

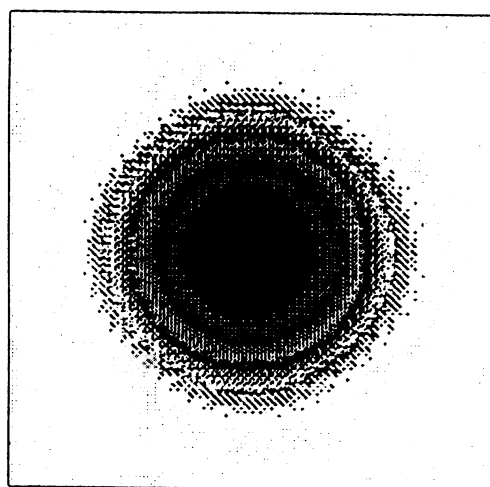
de diamètre 64 inscrit dans l'image ; en effet pour la reconstruction, nous utilisons des vecteurs de 64 points, et par conséquent pour les projections d'angle proche de  $\frac{\pi}{4}$ , et de  $\frac{3\pi}{4}$ , ces zones ne sont pas couvertes.

La figure 3.4. caractérise la qualité de l'image reconstruite (les axes du graphique ont la même signification que ceux de la figure 3.2.). Elle représente les projections calculées, à partir de l'image reconstruite (figure 3.3.), et les projections de départ avant le filtrage, et l'épandage. Le calcul des projections a été fait grâce au réseau. Les deux courbes sont très proches, l'erreur relative maximale est de l'ordre de 1%.

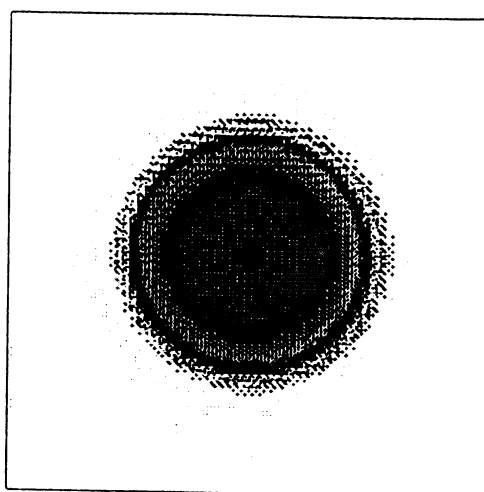


**Figure 3.4.** Caractérisation de la méthode par épandage des projections filtrées

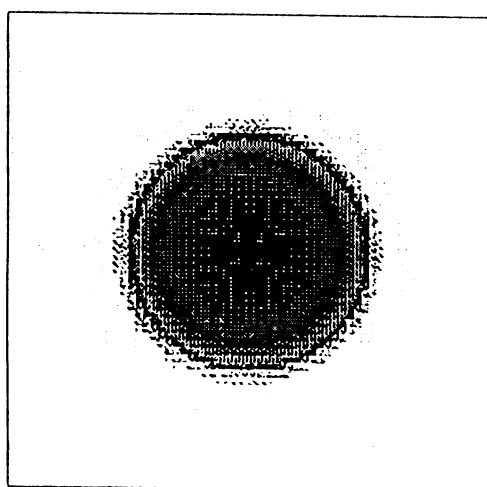
La figure 3.5. présente les résultats obtenus grâce à la technique itérative pour des nombres d'itérations différents. A chaque itération, les valeurs de projection épandues sont les différences entre les valeurs de projection mesurées brutes (sans filtrage) et les valeurs de projection calculées à l'itération précédente.



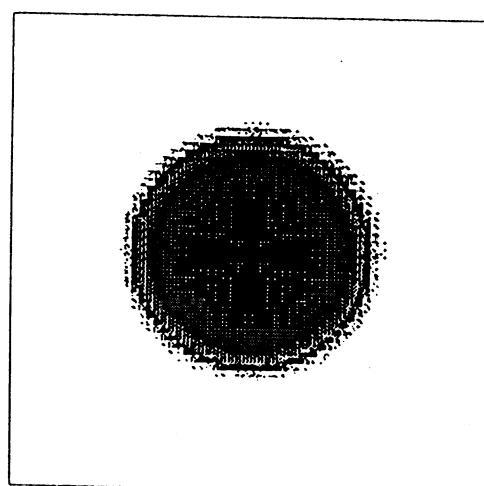
5 itérations



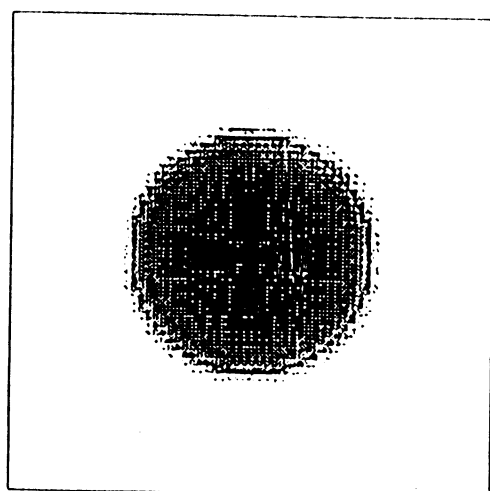
10 itérations



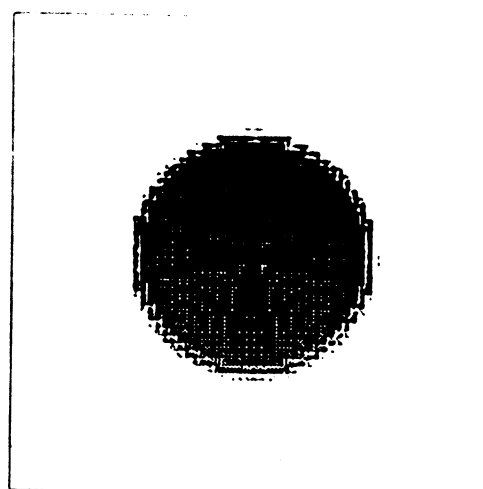
15 itérations



20 itérations



30 itérations

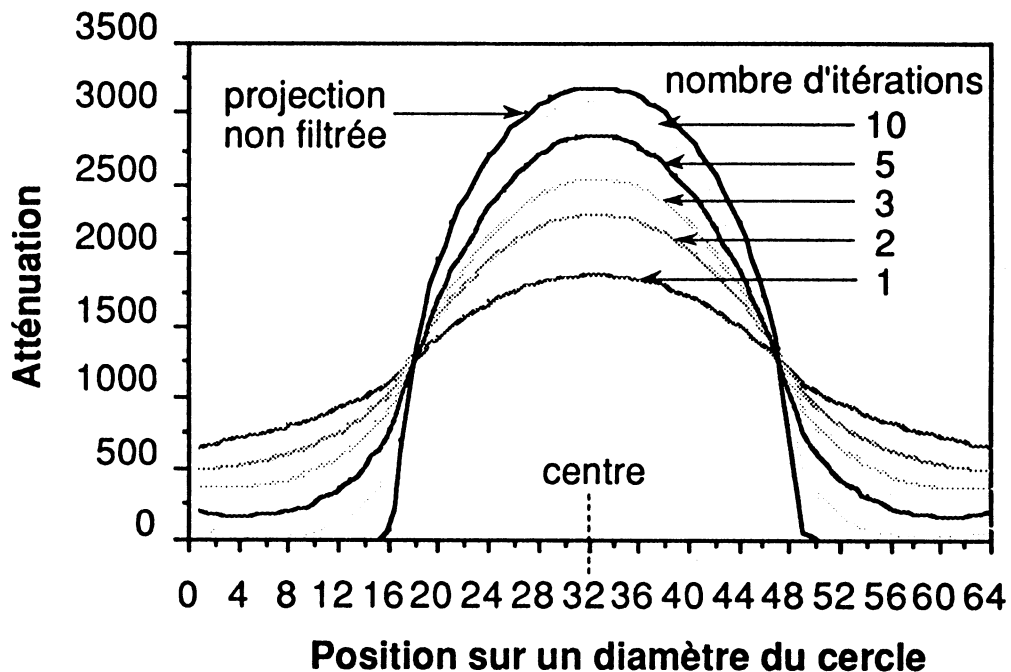


50 itérations

**Figure 3.5.** Images obtenues par la méthode itérative

La convergence de l'image est relativement lente, ceci peut s'expliquer par le fait que l'objet est très contrasté.

La figure 3.6. illustre cette convergence à travers les valeurs de projection. Les valeurs calculées à partir des images reconstruites tendent vers les valeurs de projection mesurées non filtrées, en fonction d'un nombre d'itérations.

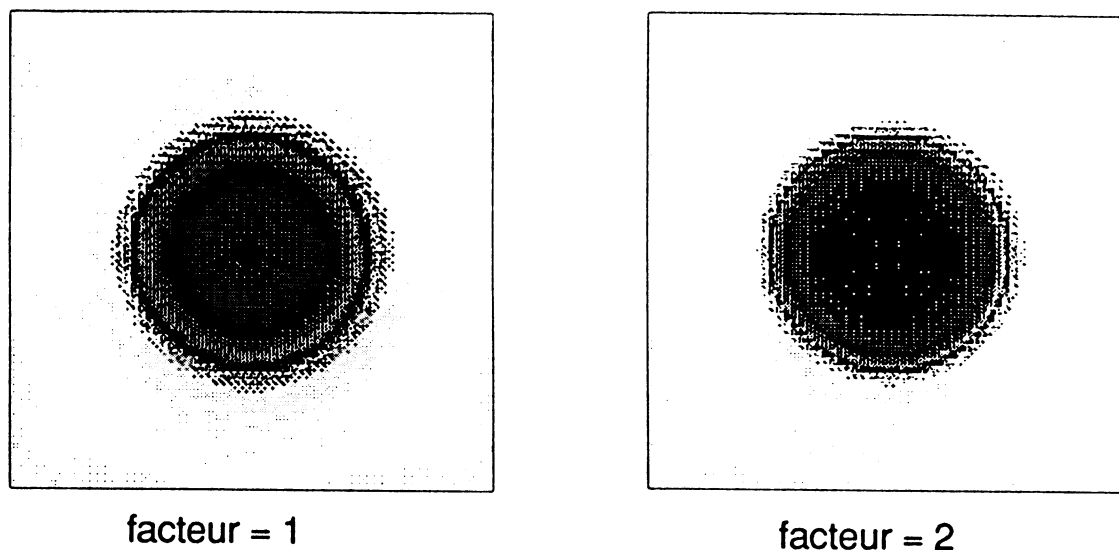


**Figure 3.6.** Convergence de la méthode itérative

La convergence est plus rapide en appliquant la méthode, itérative avec un facteur pondérateur supérieur à 1 (voir annexe 1, l'aspect mathématique de la reconstruction d'images, paragraphe 2.2.3.2.). En effet, un facteur pondérateur sur les valeurs re-projetées conduit en fonction de certaines propriétés de l'image à une convergence plus rapide, donc un temps de reconstruction moindre. A chaque itération, les valeurs de projection épanouies sont les différences pondérées par le facteur, entre les valeurs de projection mesurées brutes (sans filtrage) et les valeurs de projection calculées à l'itération précédente.

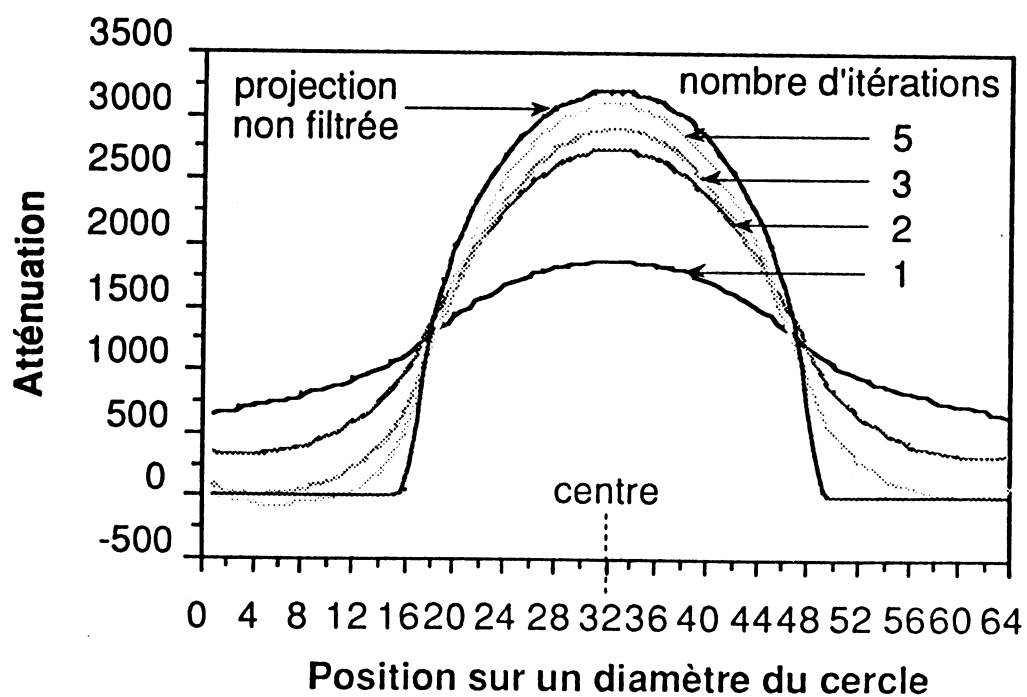


La figure 3.7. présente deux images reconstruites avec la méthode itérative après 10 itérations. L'image de gauche a été obtenue avec un facteur 1, et la droite avec un facteur 2.



**Figure 3.7.** Influence du facteur pondérateur

La figure 3.8. illustre la convergence des valeurs de projection pour un facteur pondérateur de 2 en fonction du nombre d'itérations.



**Figure 3.8.** Convergence de la méthode itérative pondérée

En comparant les courbes à celles de la figure 3.6., nous nous rendons compte de l'amélioration de la convergence, et donc de la vitesse de reconstruction.

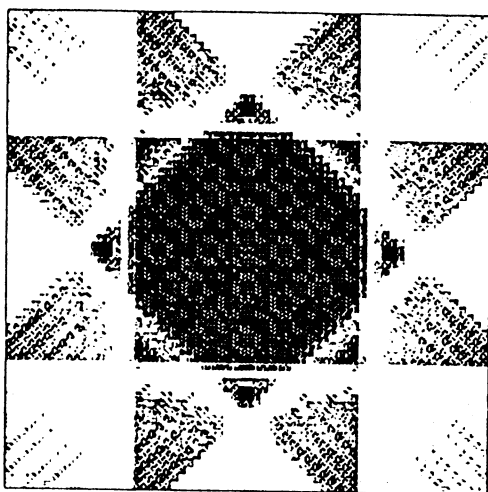
Les deux méthodes de reconstruction produisent des résultats comparables. Le nombre d'itérations nécessaires pour la deuxième méthode dépend des propriétés de l'objet à reconstruire et de l'application du facteur pondérateur, il se situe entre 20 et 50. Nous ne pouvons pas dire que l'une est meilleure que l'autre, elles sont complémentaires et le choix est à effectuer en fonction de l'application.

### **3.2.4.1.2. Etude suivant le nombre de mesures**

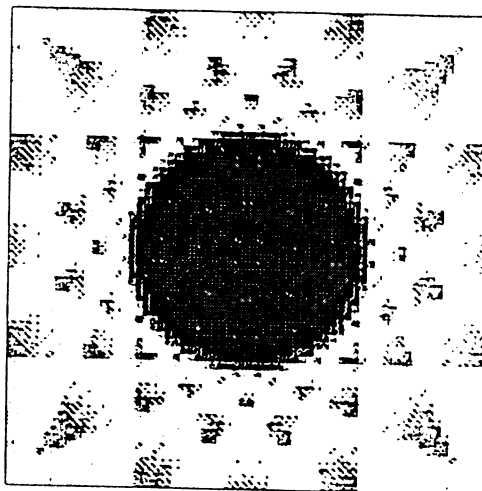
Une étude a été faite pour montrer l'influence du nombre de mesures, sur la qualité de l'image reconstruite. Nous avons étudié l'importance du nombre de projections, et du nombre de points par projection afin de déterminer le nombre de mesures nécessaire et suffisant pour réaliser des reconstructions correctes.

Les images de la figure 3.9. montrent l'influence du nombre de projections utilisées sur la qualité des images reconstruites. La méthode utilisée est l'épandage des projections filtrées, le nombre de rayons par projection est 64 (nbray), et les informations sont codées sur 10 bits (nb\_bits\_z et nb\_bits\_tg).

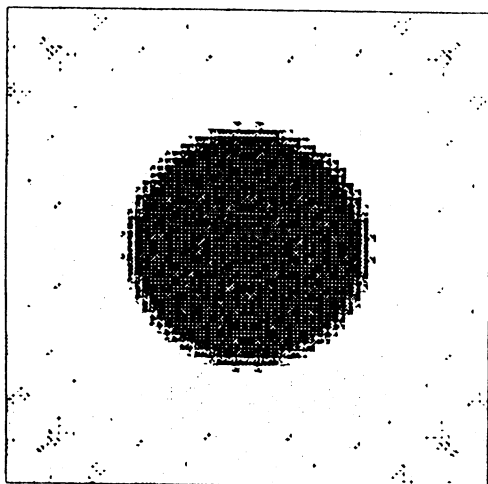
La qualité de l'image augmente avec le nombre de projections ; pour un nombre de projections supérieures à 64 (ce qui correspond à la taille de l'image) les résultats sont tout à fait corrects. Pour 100 projections l'image est très peu parasitée ; ce nombre de 100 correspond au nombre nécessaire défini mathématiquement pour obtenir une bonne définition,  $100 \approx \frac{\pi}{2} * 64$  (voir chapitre 1, paragraphe 1.1.3.2.).



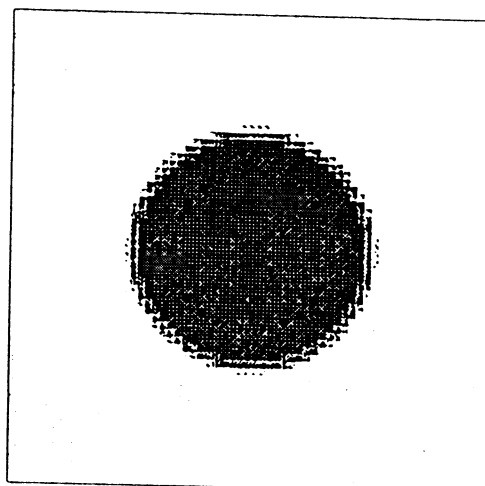
4 projections ( $0, \frac{\pi}{4}, \frac{\pi}{2}, \text{ et } \frac{3\pi}{4}$ )



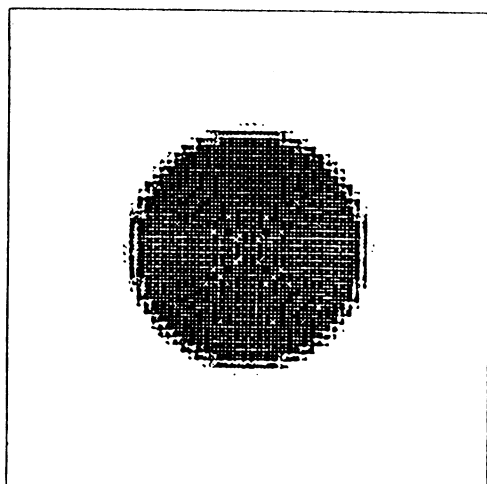
8 projections



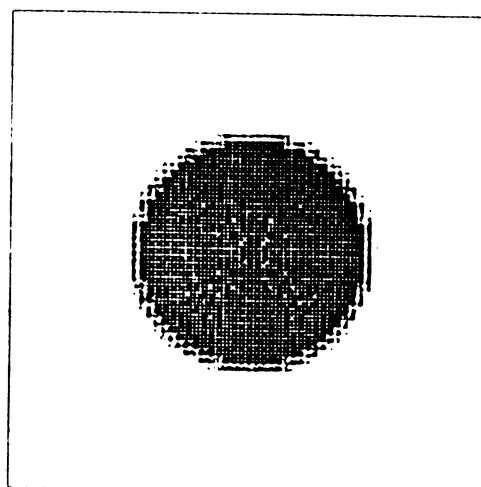
16 projections



32 projections



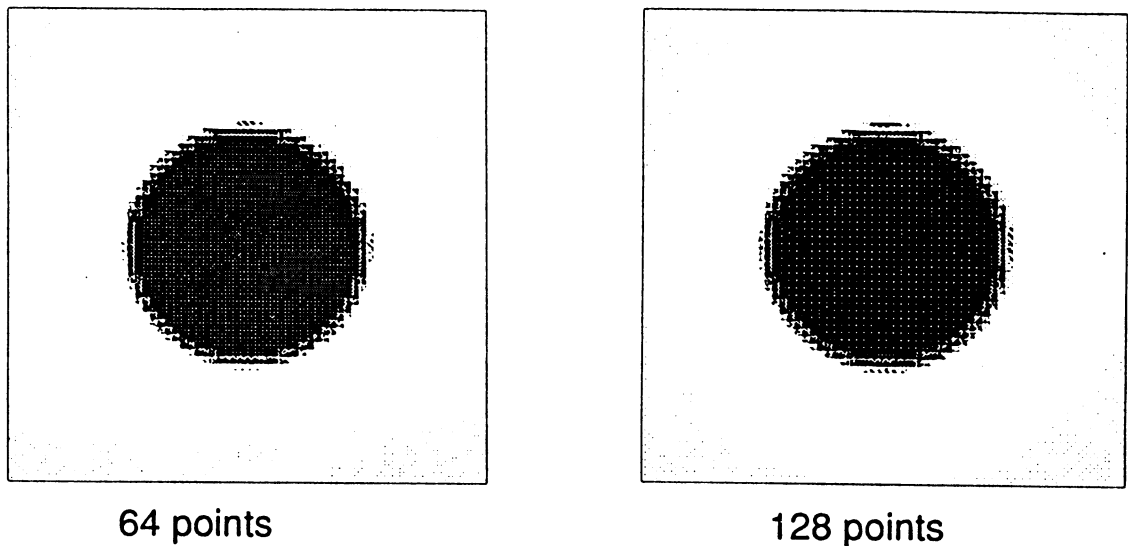
64 projections



100 projections

Figure 3.9. Influence du nombre de projections

Les images de la figure 3.10. montrent l'influence du nombre de points par projection, sur la qualité des images reconstruites. La méthode utilisée est l'épandage des projections filtrées, le nombre de projections est 64 (nbpro), et les informations sont codées sur 10 bits (nb\_bits\_z et nb\_bits\_tg).

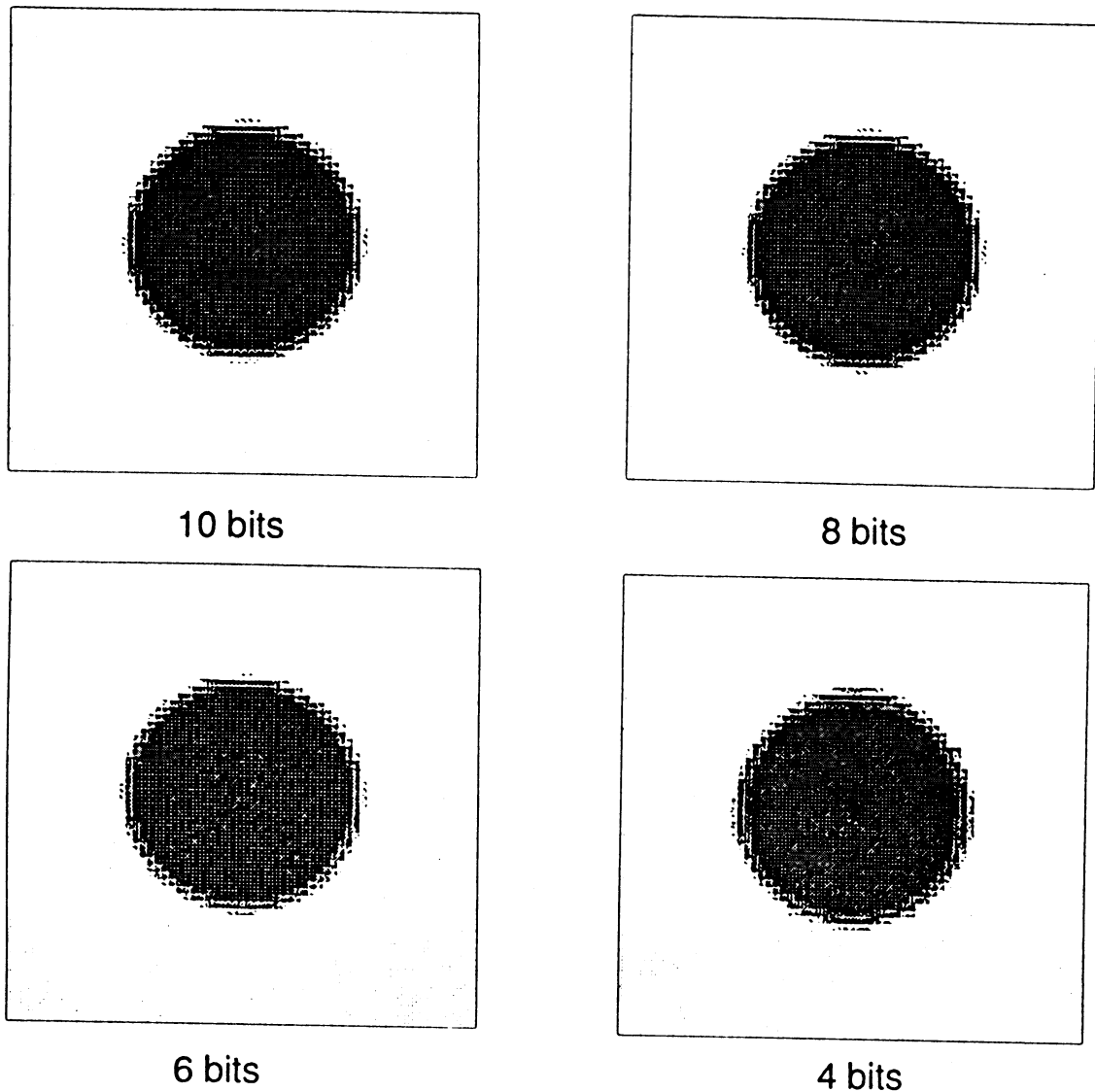


**Figure 3.10.** Influence du nombre de points par projection

Le fait d'avoir plus de points par projection que la taille de l'image (64) produit des meilleurs résultats.

### **3.2.4.1.3. Etude suivant la précision du codage**

Les images de la figure 3.11. montrent l'influence de la précision du codage des informations géométriques Z et TG (nb\_bits\_z et nb\_bits\_tg), sur la qualité des images reconstruites. La méthode utilisée est l'épandage des projections filtrées, le nombre de projections est 64 (nbpro), et le nombre de rayons par projection est 64 (nbray).



**Figure 3.11.** Influence du codage de Z et TG

Les images de la figure 3.12. montrent l'influence de la précision du point d'entrée dans un pixel Z (`nb_bits_z`), sur la qualité des images reconstruites. La méthode utilisée est l'épandage des projections filtrées, le nombre de projections est 64 (`nbpro`), le nombre de rayons par projection est 64 (`nbray`), et l'information TG est codé sur 12 bits (`nb_bits_tg`).

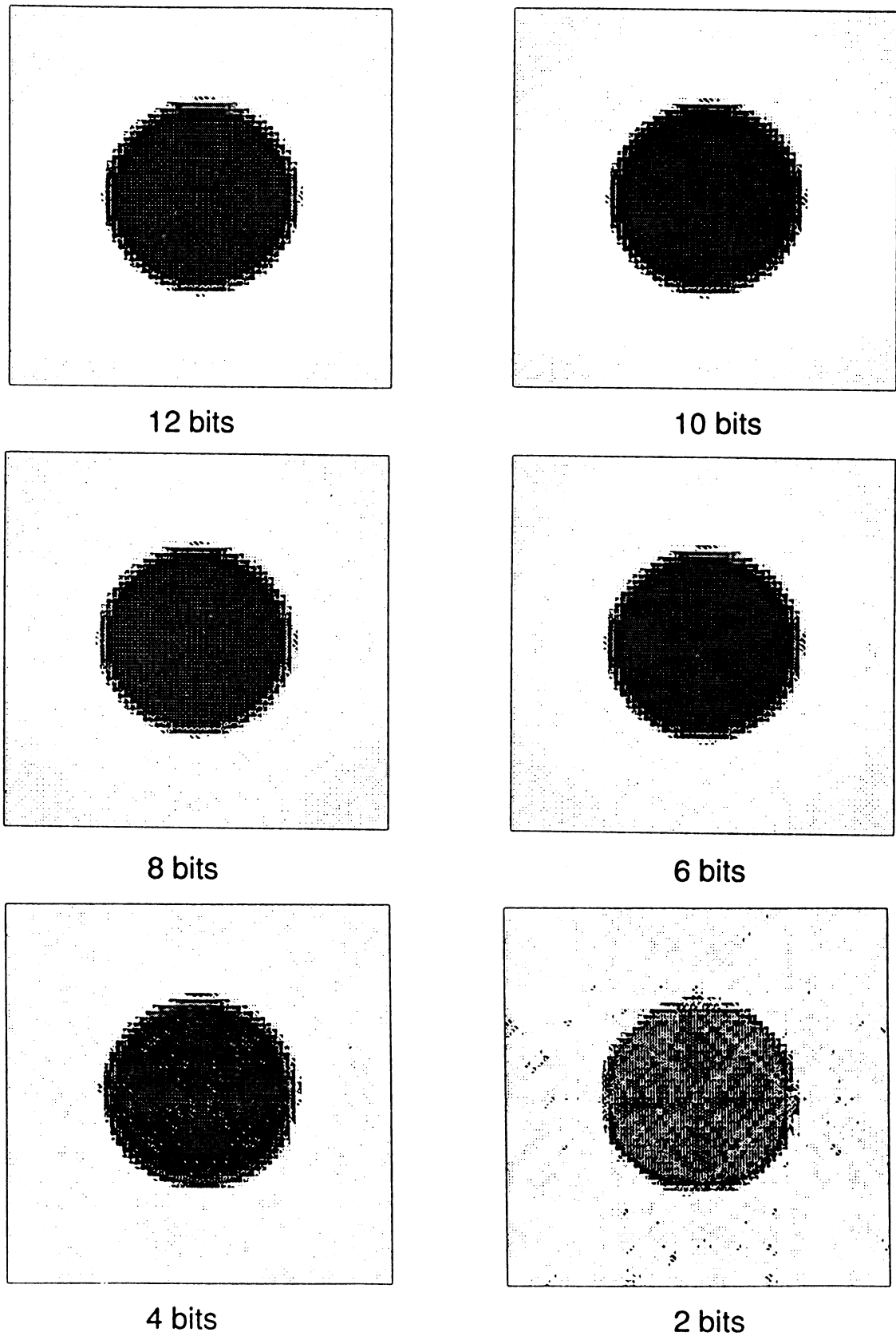


Figure 3.12. Influence du codage de  $z$

Les résultats précédents montrent que la précision du codage de TG est plus importante que celle de Z. Il est préférable d'utiliser un codage beaucoup plus précis pour TG, ainsi nous obtenons une structure minimale des messages sans dégradation de la qualité de l'image. Ceci correspond à l'étude théorique faite à propos du codage au chapitre 2, paragraphe 2.3.5. et paragraphe 2.3.6.3..

### **3.2.4.2. Reconstruction d'images complexes**

Nous avons également fait des reconstructions d'images plus complexes à partir de projections réelles. L'image présentée en figure 3.13. a été reconstruite par épandage des projections filtrées grâce à un réseau de  $4 \times 4$  cellules ( $n = 4$ ). C'est une image  $400 \times 400$ , c'est à dire que chaque cellule traite  $100 \times 100$  pixels ( $k = 100$ ). Les informations ont été codées sur 12 bits, et 512 projections de 400 points ont été utilisées. Pour information, la reconstruction d'une telle image à l'aide du simulateur écrit en OCCAM, sur un compatible IBM/PC, équipée d'une carte B004, prend environ 3 heures.

Ces images ont été visualisées grâce à une palette de 12 couleurs représentant la distribution de densité de l'objet (la palette est continue, du bleu foncé : pas de matière, au rouge : densité maximale). La figure 3.13. est une photocopie, et les niveaux de gris générés ne forment pas une palette continue.

Quoi qu'il en soit, les images obtenues sont de qualité comparable aux images reconstruites par interpolation à partir des mêmes projections au D.LETI (Division d'Electronique, de Technologie et d'Instrumentation) du CEA (Commissariat à l'Energie Atomique) à Grenoble.

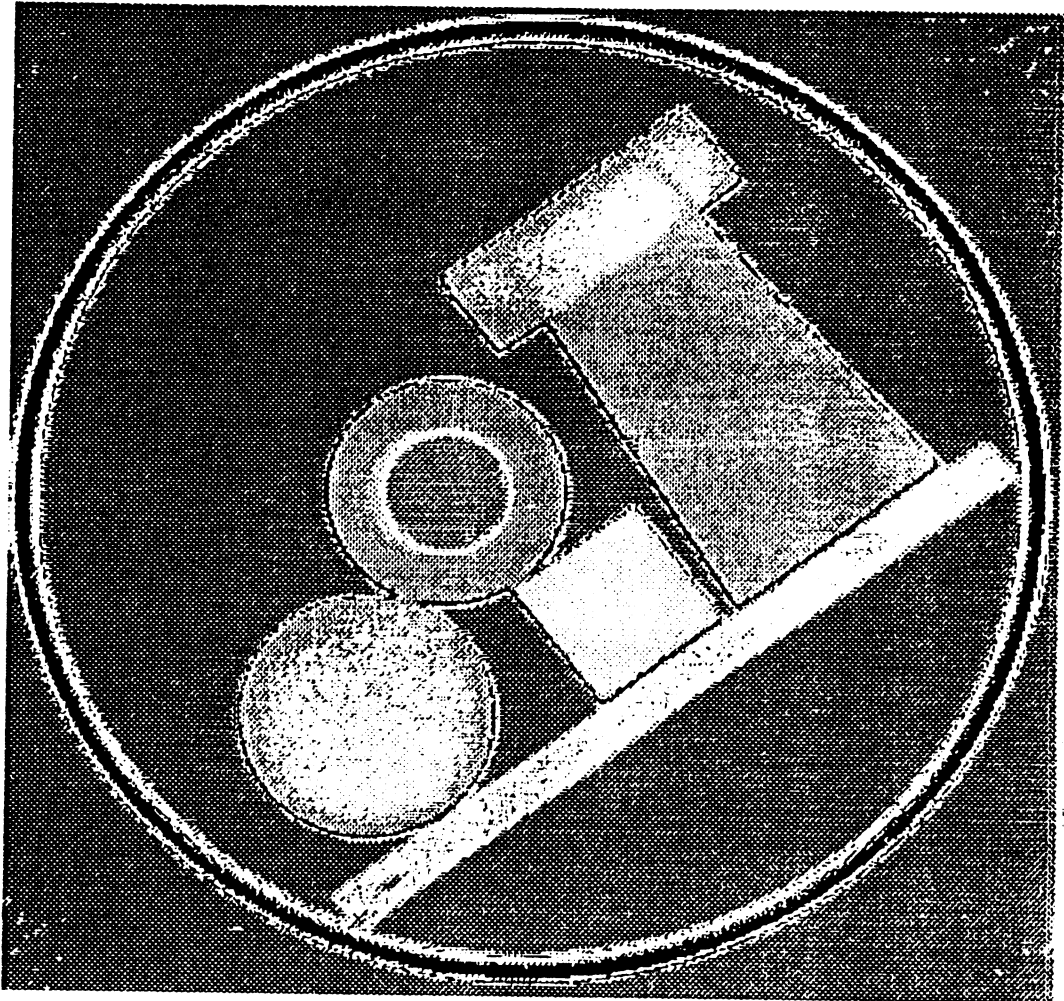


Figure 3.13. Image 400\*400

### 3.2.5. Conclusion

Cette étude fonctionnelle nous a permis de valider les algorithmes parallèles de reconstruction d'images pour notre architecture cellulaire. Les traitements d'épandage et de projection sont réalisables grâce au réseau, donc les méthodes par épandage des projections filtrées et itératives peuvent être facilement implémentées sur la machine cellulaire.



### **3.3. LES SIMULATIONS COMPORTEMENTALES HELIX**

#### **3.3.1. Silvar-Lisco et le simulateur HELIX**

Silvar-Lisco est un logiciel intégré de CAO, il inclut tous les programmes et utilitaires classiques d'une chaîne de conception : éditeurs de schémas (Structured Design System) et masques (Princess), simulateurs logique (Bimos) et comportemental (Hélix), interfaces pour simulateurs électrique (Spice) et logique (Hilo), extracteurs, vérificateurs (Drc, Erc).

Le simulateur HELIX a été choisi pour réaliser les simulations comportementales du réseau car c'est un outil de simulation de matériel très performant [SIL86]. Il permet de simuler le fonctionnement d'architectures à des niveaux de description plus ou moins fins (processeurs, partie contrôle et partie opérative, UAL, RAM, ROM, PLA, registres, portes logiques).

Les éléments de base de l'architecture que nous voulons simuler sont modélisés dans un langage HHDL (Hierarchical Hardware Description Language) : les sorties d'un modèle sont exprimées en fonction des entrées et du temps. La connectique de la machine globale à simuler est décrite schématiquement, en utilisant les "boîtes" associées aux modèles HHDL, et en les reliant entre elles. Le langage de description est du type langage Pascal, auquel ont été rajoutées des primitives concernant les entrées et sorties des modèles. Les informations circulant sur les connexions entre les différents modèles, sont des structures Pascal plus ou moins complexes (booléen, entier, réel, tableau, enregistrement, ... ).

Ces simulations ont été réalisées sur des stations de travail Vax\_Station GPX II (de configuration : 5 mégaoctets de RAM, et 2 disques durs de 450 mégaoctets).

### **3.3.2. Structure du programme de simulation**

Une machine de reconstruction basée sur un réseau cellulaire de 8\*8 cellules a été simulée. L'ordinateur hôte a été construit en deux parties : un émetteur qui calcule les messages et les envoie au réseau, et un récepteur qui collecte les messages en sortie du réseau. La cellule de base est construite à l'aide de deux modèles : un modèle tampon, et un modèle cellule qui correspond à la partie traitement.

Pour ces simulations, les modèles fonctionnels n'ont pas été définis aussi précisément que pour les simulations OCCAM. Leur intérêt est de permettre d'étudier le cheminement des messages, et non de valider les algorithmes de reconstruction, en examinant les images reconstruites. Elles nous ont également permis d'étudier l'activité du réseau, en fonction de certains choix algorithmiques de la cellule. Une seule phase de calcul a été simulée (épandage ou projection).

Le modèle de la cellule est donnée en annexe 3, ainsi que plusieurs schémas illustrant la machine cellulaire simulée.

### **3.3.3. Les résultats : performances du réseau**

#### **3.3.3.1. Etude du temps de reconstruction**

Une première série de mesures a permis d'observer le temps de reconstruction et l'activité du réseau, en fonction du temps de traitement de la cellule.

Chaque cellule du réseau 8\*8 traite un pixel ; 8 projections sont envoyées sur 180°, et chaque vecteur de projection possède 8 points. Les messages sont envoyés à la vitesse d'une microseconde par message, ce qui correspond environ au temps nécessaire à l'acquisition d'un point sur les machines actuelles (voir annexe 2, paragraphe 3.2.). La cellule traite un message uniquement si le tampon destinataire est vide, le tampon d'entrée est vidé au début du traitement, et le message est transmis en fin de traitement. Un délai est

associé à chaque étape de calcul : lecture du message, calcul du tampon de sortie, .... ; ces délais sont constants à l'exception du temps de traitement.

Une telle simulation a nécessité un temps de calcul de 20 minutes sur une station de travail GPX II.

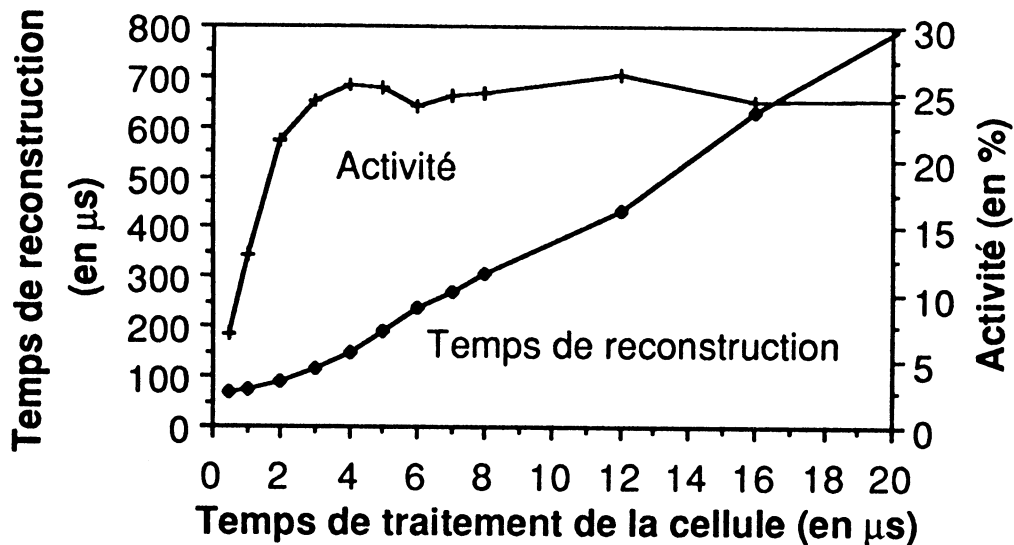


Figure 3.14. Résultats pour 8 projections

L'activité du réseau pour 8 projections est de l'ordre de 25%, pour un temps de traitement de la cellule supérieur à 4 microsecondes (16 cellules sur 64 sont actives en moyenne). Ce résultat est bon, puisqu'il comprend les phases de chargement et de déchargement du réseau. Nous pouvons comparer ce résultat à l'activité d'un réseau systolique de  $8 \times 8$  processeurs traitant un flot de données de 8 vecteurs de 8 points, qui est de 50%. Pour notre réseau, la direction du flot de données n'est pas constante mais tourne sur  $180^\circ$  ; il y a donc des croisements de flots à l'intérieur du réseau et des surcharges de travail sur certaines cellules, ce qui explique le rapport 2 qu'il existe entre les activités comparées.

La zone intéressante des courbes de la figure 3.14. se situe pour un temps de traitement de la cellule de 4 microsecondes. En effet, dans cette partie le réseau est très bien utilisé (bonne activité), et le temps de reconstruction est de l'ordre du double du temps d'acquisition des

mesures (qui est de 64 microsecondes), c'est à dire 150 microsecondes.

Ces résultats sont généralisables à un réseau de dimension  $N \times N$ , traitant une image  $N \times N$ , à l'aide de  $N$  projections de  $N$  points.

### 3.3.3.2. Influence du nombre de projections

La figure 3.15. représente trois courbes donnant l'activité du réseau sur le temps total de reconstruction, en fonction du temps de traitement de la cellule et du nombre de projections. La taille du réseau et les choix algorithmiques sont les mêmes qu'au paragraphe précédent.

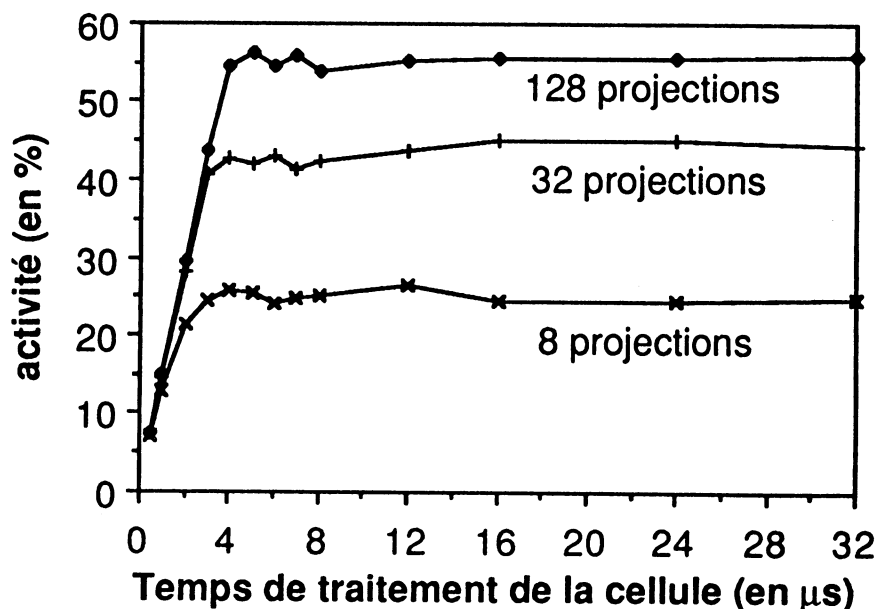


Figure 3.15. Rendement du réseau

Le palier des 25% augmente avec le nombre de projections, ceci est dû au fait, que le temps de chargement et déchargement du réseau diminue par rapport au temps total de reconstruction, et que les flots de données sont plus réguliers. Avec 32 projections, il est à 45%, et avec 128 projections, à 55%. Nous pouvons affirmer que cette activité tend vers 100%, lorsque le nombre de projections tend vers l'infini.

En extrapolant, cette constatation doit se retrouver quand une cellule traite une sous-image au lieu d'un seul pixel : l'activité du réseau croît avec la taille de la sous-image associée à chaque cellule. Il faut bien sûr trouver un juste équilibre entre le traitement des projections en temps réel et l'utilisation optimale du réseau, pour calculer la taille de la sous-image associée à chaque cellule.

Pour information, nous pouvons préciser qu'une simulation de ce type, mettant en jeu 8 projections (64 messages), a demandé 20 minutes de calcul sur la station de travail. Ce temps est passé à 80 minutes pour 32 projections (256 messages), et 5 heures pour 128 projections (1024 messages).

### **3.3.3.3. Influence du type de traitement**

Jusqu'à présent, nous avons supposé que le message était passé en fin de traitement, or cette hypothèse ne s'applique qu'aux messages de type projection. Pour les messages de type différent (épandage, entrée\_image, sortie\_image, et transparence), ils peuvent être passer avant le traitement, dès que le tampon de sortie a été calculé.

Par exemple, pour un réseau 8\*8 (dont chaque cellule est associée à un pixel, et traite un message en 8 microsecondes), une phase de calcul à partir de 8 projections de 8 points nécessite un temps de 307 microsecondes, en supposant une vitesse d'émission des messages d'un par microseconde, et le passage des messages en fin de traitement (messages de type projection). Ce qui correspond à une activité de 25% (voir paragraphe 3.3.3.1.). Par contre, la même phase de calcul, avec le passage des messages en début de traitement (avec des messages de type épandage) demande 145 microsecondes, soit une activité de 53%.

Ce résultat montre qu'il est très important de transmettre les messages le plus tôt possible, lorsque ceci est possible.

### 3.3.3.4. Traitement d'une sous-image

Nous avons simulé un réseau de  $8 \times 8$  cellules, où chacune d'elle traite  $4 \times 4$  pixels. La reconstruction porte donc sur une image  $32 \times 32$ , et elle est obtenue à l'aide de 32 projections de 32 points. La génération des 1024 messages à envoyer au réseau demande 1024 microsecondes (à raison d'une toute les microsecondes).

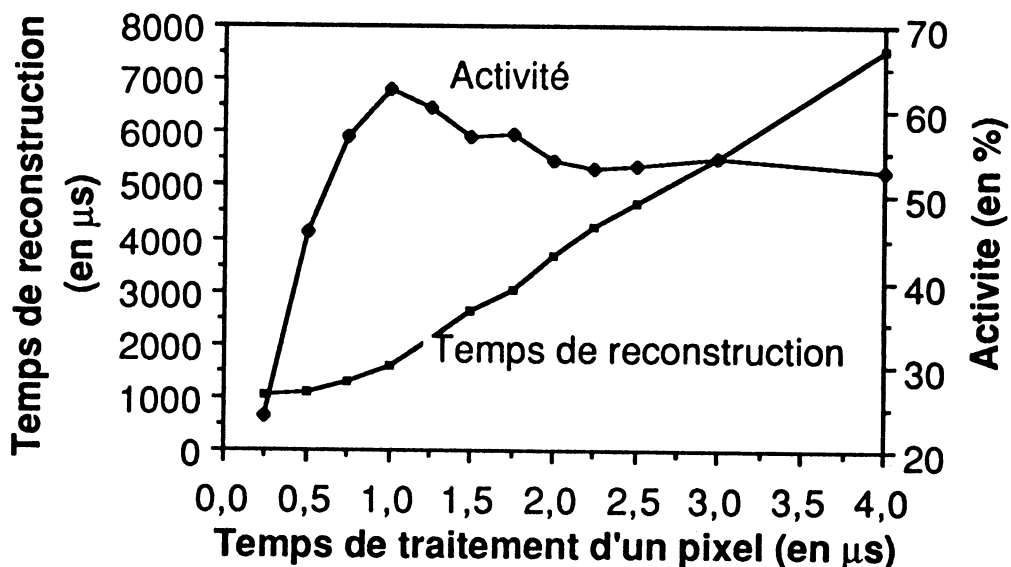


Figure 3.16. Une cellule traite  $4 \times 4$  pixels

La figure 3.16. donne le temps total de reconstruction, et l'activité du réseau en fonction du temps de traitement d'un pixel. La meilleure activité est de 60%, lorsque le temps de traitement du pixel est d'une microseconde, et alors la reconstruction a demandé 1,5 millisecondes.

Pour ces simulations, le passage des messages est fait à la fin du traitement. Il est clair que dans le cas des messages de type épandage, où il est possible de les transmettre avant le traitement, l'activité est supérieure à 60% (voir paragraphe 3.3.3.3.).

Ces résultats peuvent être extrapolés, afin d'en tirer une relation générale entre le nombre de pixels traités par cellule, le temps de traitement du pixel, et la vitesse d'acquisition des projections (qui correspond à la vitesse d'envoi des messages), pour qu'avec un réseau de taille  $N \times N$  (où chaque cellule traite  $k \times k$  pixels) permettant de

travailler avec des images  $m \times m$  (avec  $m = N \times k$ ), nous avons un excellent rendement et un temps de reconstruction rapide.

Nous supposons que le nombre de mesures est de l'ordre du nombre de pixels de l'image ( $m$  projections de  $m$  points).

De manière intuitive, nous pouvons dire qu'une cellule doit traiter  $k$  messages pendant l'acquisition d'un vecteur de projection ( $m$  points). Pour un message, le nombre de pixels à traiter est au maximum de  $(2k - 1)$ , donc il en suit la relation :

$$\text{Temps d'acquisition d'un vecteur} \approx 2 * k^2 * \text{Temps de traitement d'un pixel}$$
$$(T_{\text{vecteur}}) \qquad \qquad \qquad (T_{\text{pixel}})$$

Cette relation est vérifiée pour nos simulations présentées dans ce paragraphe.

Sachant qu'un vecteur comprend  $m$  points,  $T_{\text{vecteur}} = m * T_{\text{point}}$  ( $T_{\text{point}}$  est le temps d'acquisition d'un point), nous pouvons écrire :

$$N * T_{\text{point}} \approx 2 * k * T_{\text{pixel}}$$

L'intérêt de cette relation est qu'elle permet pour un problème donné, de calculer la taille du réseau et le nombre de pixels, qu'une cellule devra traiter pour avoir une utilisation optimale. La taille de l'image est fixée d'après l'utilisation de la machine de reconstruction,  $T_{\text{point}}$  est défini par le système d'acquisition des mesures,  $T_{\text{pixel}}$  dépend de l'architecture interne de la cellule.

La taille de la sous-image traitée par une cellule est donnée par :

$$k^2 \approx \frac{m * T_{\text{point}}}{2 * T_{\text{pixel}}}$$

Par exemple, si nous voulons reconstruire des images  $1000 \times 1000$  à l'aide de  $1000 \times 1000$  valeurs de projections acquises à la vitesse d'un point par microseconde, et en supposant que notre architecture assure le traitement d'un pixel en 5 microsecondes, alors la sous-image que devra traiter chaque cellule pour une utilisation optimale du réseau est définie par  $k^2 \approx \frac{1000 * 1}{2 * 5}$ , c'est à dire  $k = 10$ , et  $N = 100$ . Il faudra donc utiliser un réseau de  $100 \times 100$  cellules, chacune d'elle traitant  $10 \times 10$

pixels. Le temps d'acquisition total est d'une seconde, et le temps de calcul d'une phase d'épandage, ou de projection est inférieur à 2 secondes.

### **3.3.4. Conclusion**

Toutes ces simulations comportementales faites grâce au simulateur HELIX ont permis d'étudier l'aspect temporel de l'architecture cellulaire. La reconstruction est un problème qu'il faut résoudre en temps réel ; nous avons montré que la solution consistant à associer un seul pixel à un processeur n'était pas optimale, et nous savons maintenant définir une configuration du réseau, telle que son activité soit d'au moins 60%.





## ***Chapitre 4***

---

# Architecture de la cellule

|                                                      |            |
|------------------------------------------------------|------------|
| <b>4.1. Introduction.....</b>                        | <b>161</b> |
| 4.1.1. Définition des paramètres de la cellule.....  | 161        |
| 4.1.2. Méthode de conception .....                   | 163        |
| <b>4.2. Définition de l'architecture.....</b>        | <b>163</b> |
| 4.2.1. Les différentes fonctions de la cellule ..... | 163        |
| 4.2.2. L'architecture proposée.....                  | 164        |
| 4.2.3. Les tampons d'entrée et sortie .....          | 166        |
| 4.2.4. La partie opérative .....                     | 168        |
| 4.2.4.1. Accès aux tampons.....                      | 169        |
| 4.2.4.2. Mémorisation du message.....                | 171        |
| 4.2.4.3. Module de calcul arithmétique .....         | 173        |
| 4.2.4.4. Module de calcul combinatoire.....          | 179        |
| 4.2.4.5. Mémoire de l'image .....                    | 183        |
| 4.2.5. Le séquençement.....                          | 184        |
| <b>4.3. Les simulations.....</b>                     | <b>188</b> |
| 4.3.1. Présentation des simulations.....             | 188        |
| 4.3.2. Validation de l'architecture.....             | 192        |

## **4.4. Implantation.....192**

### **4.4.1. Le logiciel SOLO 1200..... 192**

### **4.4.2. Réalisation du circuit ..... 193**

#### 4.4.2.1. Principe..... 193

#### 4.4.2.2. La partie opérative ..... 194

#### 4.4.2.3. Le séquenceur ..... 195

#### 4.4.2.4. Le circuit complet..... 198

### **4.4.3. Le brochage du circuit..... 200**

### **4.4.4. Performances ..... 202**

#### 4.4.4.1. Surface..... 202

#### 4.4.4.2. Temps ..... 202

#### 4.4.4.3. Performance d'un réseau construit avec le circuit..... 203

### **4.4.5. Interfaçage du circuit dans l'environnement hôte .. 207**

#### 4.4.5.1. Principe de l'interface ..... 207

#### 4.4.5.2. La carte ..... 207

#### 4.4.5.3. Vitesse de communication..... 211

## Chapitre 4 : Architecture de la cellule

## 4.1. INTRODUCTION

Dans ce chapitre, nous présentons la structure interne de la cellule de base du réseau pour la reconstruction d'images. En fonction de l'algorithme final décrit au chapitre 2, nous proposons une architecture, qui découle d'un bon compromis temps et surface. Il est vrai que les contraintes temporelles ne sont pas très fortes : la vitesse d'acquisition des projections étant relativement lente, la cellule peut traiter un tableau de pixels, et non pas un seul pixel. D'autre part, étant donné que le nombre de pixels à associer à une cellule dépend du temps de traitement du pixel, il convient de ne pas négliger l'aspect temporel, puisqu'il permet un gain en taille du réseau.

### 4.1.1. Définition des paramètres de la cellule

Pour définir l'architecture de la cellule, nous avons fixé tous les paramètres intervenant dans son fonctionnement : codage des informations, taille de l'image traitée. Une cellule nécessitant des valeurs de paramètres autres que ceux choisis, pour une application bien particulière, peut facilement s'extrapoler à partir de cette description architecturale.

L'étude faite sur les performances du réseau au chapitre 3, nous a conduit à fixer la taille de la sous-image traitée par chaque cellule à 8 par 8 pixels. L'étude portant sur le codage des informations géométriques (voir chapitre 2), et les résultats des simulations fonctionnelles (voir chapitre 3), nous permettent de définir parfaitement la structure des messages, notamment le codage de la tangente TG et du point d'entrée dans le pixel  $Z_{\text{pixel}}$ . Ils sont donc composés de trois champs de 16 bits :

#### ⇒ champ n°1

|                                                      |        |
|------------------------------------------------------|--------|
| type du message TYPE                                 | 3 bits |
| adresse du pixel dans la cellule $AD_{\text{pixel}}$ | 3 bits |
| point d'entrée dans le pixel $Z_{\text{pixel}}$      | 8 bits |

|                                             |       |
|---------------------------------------------|-------|
| bit de signe de la tangente ou cotangente S | 1 bit |
| bit désignant le passage de tg ou cotg TC   | 1 bit |

⇒ champ n°2

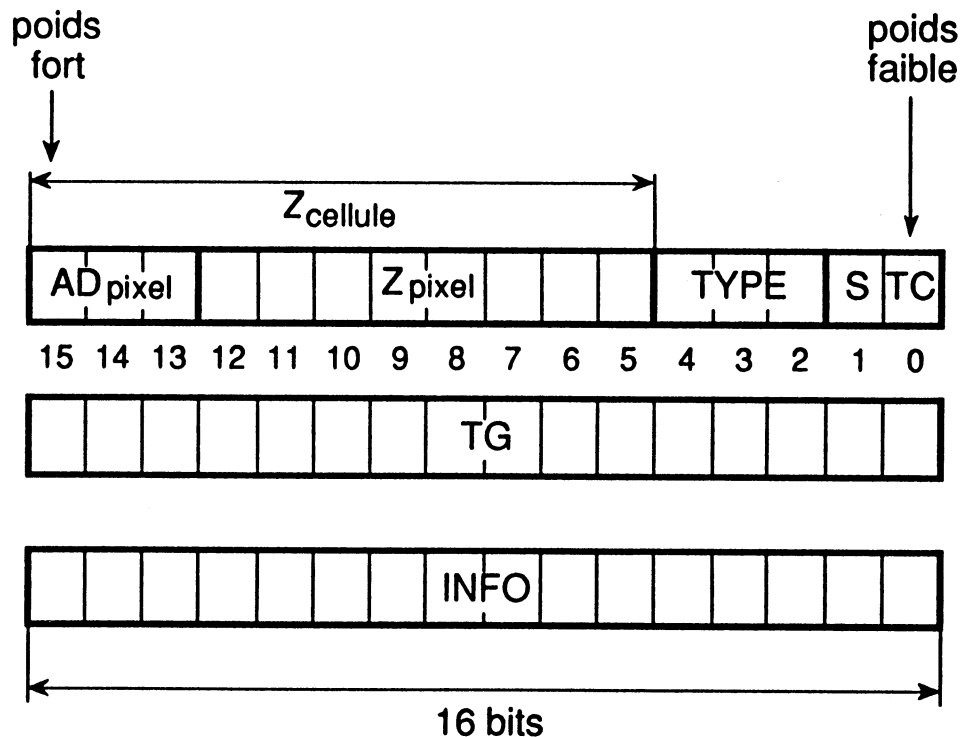
|                             |         |
|-----------------------------|---------|
| tangente (ou cotangente) TG | 16 bits |
|-----------------------------|---------|

⇒ champ n°3

|                                           |         |
|-------------------------------------------|---------|
| information (projection ou de pixel) INFO | 16 bits |
|-------------------------------------------|---------|

La valeur de TG est comprise entre 0 et 215, et celle de  $Z_{\text{pixel}}$  entre 0 et  $2^8 - 1$ .

La figure 4.1. illustre les trois mots de 16 bits formant le message.



**Figure 4.1.** Structure du message

Le tableau suivant donne le codage du type du message. Nous avons cinq types différents à coder (épandage, projection, entrée\_image, sortie\_image, et transparence), il faut donc trois bits pour coder cette information : type2, type1 et type0.

| type2 | type1 | type0 | type du message |
|-------|-------|-------|-----------------|
| 0     | X     | X     | transparence    |
| 1     | 0     | 0     | entrée_image    |
| 1     | 0     | 1     | sortie_image    |
| 1     | 1     | 0     | épandage        |
| 1     | 1     | 1     | projection      |

La valeur X représente la valeur quelconque (0 ou 1). Ce codage permet le traitement d'un message en fonction de son type avec un minimum de facilité quant à la réalisation des tests.

### 4.1.2. Méthode de conception

La méthode de conception de l'architecture est une méthode descendante. A partir de la fonction maintenant figée de la cellule, nous étudions les principaux blocs fonctionnels, puis cette description est affinée jusqu'aux portes logiques.

Par contre, une fois que cette phase de conception est faite, la réalisation du circuit est ascendante. En partant de la découpe proposée lors de la conception, nous construisons chaque élément de la cellule, depuis les portes les plus élémentaires jusqu'au circuit lui-même.

## 4.2. DEFINITION DE L'ARCHITECTURE

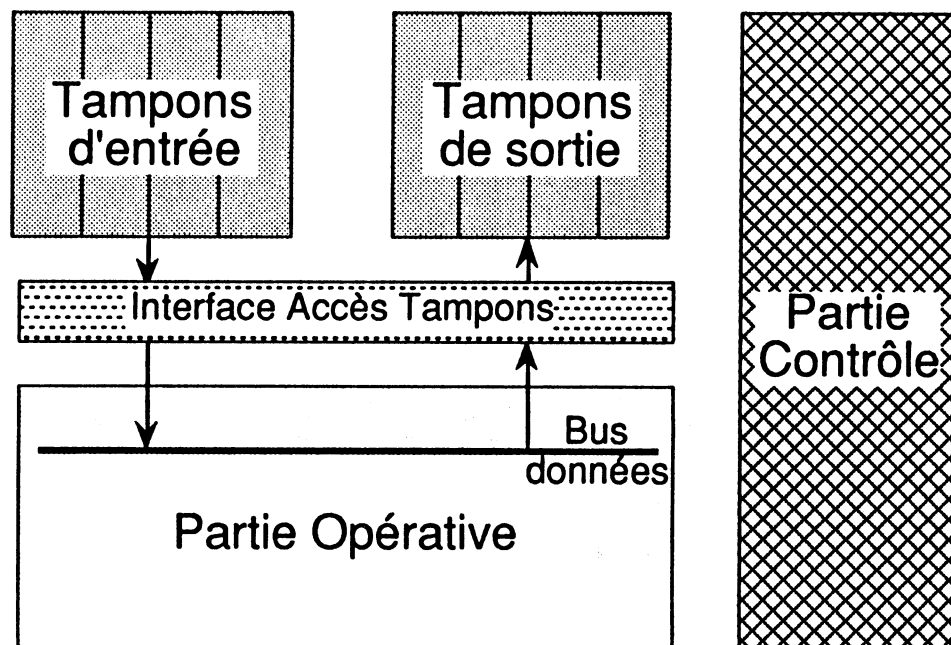
### 4.2.1. Les différentes fonctions de la cellule

D'après l'algorithme de la cellule, les fonctions qu'elle doit réaliser sont les suivantes : elle doit pouvoir accéder aux tampons d'entrée et sortie (lecture et écriture des messages), effectuer les opérations arithmétiques d'addition, soustraction, multiplication et division (calcul des différents points de sortie, du facteur LONG, et mise à jour des valeurs des pixels), et stocker la sous-image qu'elle traite (8\*8 pixels).



### 4.2.2. L'architecture proposée

L'architecture proposée est tout à fait classique, elle reprend la structure partie opérative et partie contrôle, et la figure 4.2. en présente le schéma [MAZ88]. Le chemin de données comprend un unique bus de données, et l'exécution des opérations est commandée par un séquenceur à un seul niveau. La liaison du processeur au monde extérieur représenté par les tampons d'entrée et de sortie, est possible grâce à un module d'interface, permettant un échange de données facile.



**Figure 4.2.** Aspect général de l'architecture de la cellule

Nous allons à présent préciser l'architecture en expliquant les choix effectués.

Les tampons mémorisent les messages lors des transferts de cellules aux voisines. Ils comprennent un drapeau indiquant l'état et réalisant l'exclusion mutuelle, et un ensemble de registres permettant le stockage d'un message de 48 bits.

Les messages traités par la cellule sont de grande taille, et il n'est donc pas envisageable pour une question de surface de réaliser un bus de données de cette largeur. Mais sachant que beaucoup d'opérations arithmétiques portent sur 16 bits, l'architecture de la partie opérative est construite autour d'un bus données de 16 bits. Par conséquent, l'accès aux tampons est fait en 3 cycles pour acquérir ou transmettre un message complet, et il est possible de lire ou écrire un seul des trois champs du message.

La scrutation des tampons d'entrée pour détecter l'éventuelle présence de messages est réalisée de manière circulaire, grâce à un compteur, qui les parcourt successivement. Ce principe est très facilement implantable, et permet de résoudre le problème de la famine. Par contre, il n'est pas très performant, puisqu'il ne permet pas de sélectionner un tampon plein à chaque cycle d'horloge, et par conséquent, un message peut attendre au plus 3 cycles avant d'être choisi. Cette mauvaise performance n'est que relative, puisque le traitement d'un message étant assez long (de l'ordre de 300 cycles), la perte de 3 cycles à la réception du message est tout à fait négligeable, par rapport aux 300, que nécessite le traitement.

La partie opérative comprend une unité arithmétique et logique (UAL) qui permet de réaliser les 4 opérations arithmétiques élémentaires sur des opérandes de 16 bits, et rend compte à la partie contrôle des éventuels résultats négatifs lors de soustractions (pour permettre facilement les tests). L'UAL, et notamment les algorithmes qu'elle réalise, seront détaillés plus loin. La mémorisation des valeurs des pixels de l'image 8\*8 traitée par la cellule est faite grâce à une mémoire RAM (Random Acces Memory). Cette solution est la moins coûteuse en surface et la plus pratique, par contre le temps d'accès n'est pas négligeable (par rapport à des registres par exemple), mais nous verrons que ce n'est pas le point critique de l'architecture proposée. D'autre part, il y a des modules combinatoires qui permettent de calculer certains bits du message de sortie, le côté de sortie, l'adresse du premier pixel à traiter et l'adresse du pixel suivant.

La figure 4.3. illustre plus précisément la structure de la cellule.

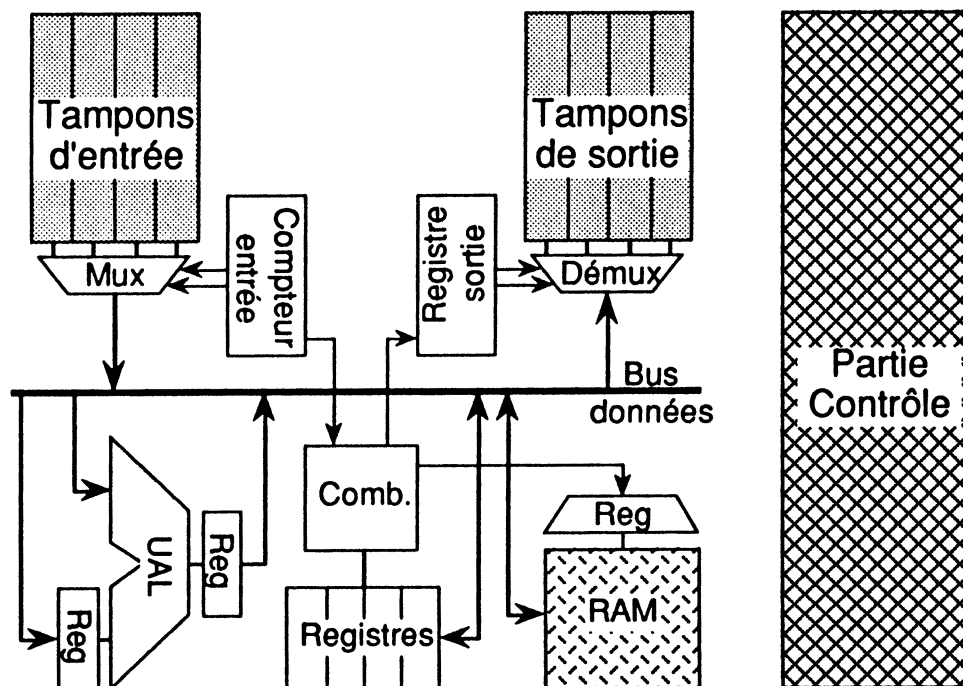


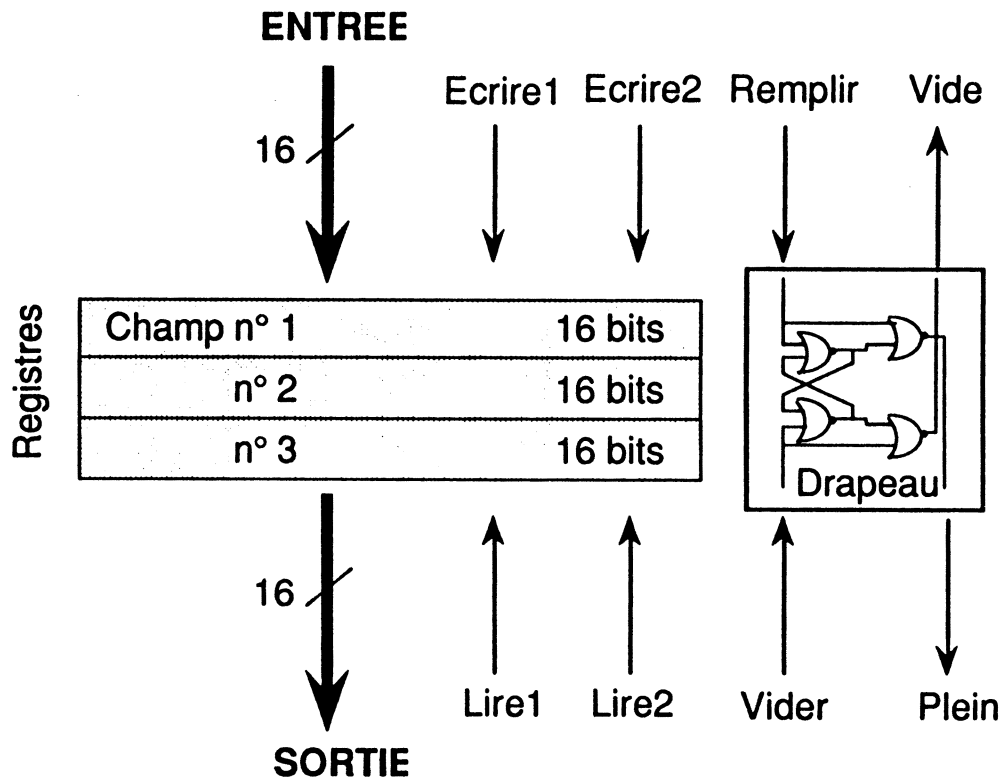
Figure 4.3. Synoptique de la cellule

### 4.2.3. Les tampons d'entrée et sortie

Un tampon de communication est une interface unidirectionnelle entre une cellule productrice et une cellule consommatrice. Son drapeau gère l'exclusion mutuelle des deux cellules intervenantes, afin que tout message soit consommé après qu'il soit produit, et qu'aucun ne soit perdu.

Le drapeau est mémorisé par une bascule de type RS, et une petite combinatoire assure le non-recouvrement des signaux d'état.

La mémorisation du message est faite par des registres simples regroupés en mots de 16 bits, et l'accès à l'un quelconque des trois est réalisé grâce à deux commandes d'écriture et deux commandes de lecture. Un tampon est destiné à être connecté à un bus en entrée et à un bus en sortie, c'est pourquoi se trouvent en sortie des portes 3 états passantes uniquement pendant les phases de lecture du tampon.



**Figure 4.4.** Un tampon de communication

La figure 4.4. illustre en détail le tampon, et notamment ses différentes commandes.

Le protocole de communication se déroule de la façon suivante.

Du côté producteur (ou émetteur), le tampon émet un signal *Vide* qui lui indique qu'il peut passer un message. Grâce aux commandes d'écriture *Ecrire1* et *Ecrire2*, il transmet les trois parties du message (*Ecrire1 . Ecrire2* → champ n°1, *Ecrire1 . Ecrire2* → champ n°2, *Ecrire1.Ecrire2* → champ n°3). Ensuite il envoie le signal *Remplir* qui va modifier l'état du drapeau pour activer la consommation.

Du côté consommateur (ou récepteur), le signal émis par le tampon *Plein* indique la présence d'un message. Le consommateur va récupérer les trois champs composant le message à l'aide des commandes *Lire1* et *Lire2* (*Lire1 . Lire2* → champ n°1, *Lire1 . Lire2* → champ n°2, *Lire1.Lire2* → champ n°3). Ensuite il envoie le signal *Vider* qui va libérer le tampon au producteur.

### 4.2.4. La partie opérative

Le chemin de données de la cellule comprend plusieurs parties distinctes articulées autour du bus 16 bits. Il inclut un module réalisant l'accès aux tampons, une zone de mémorisation permettant de stocker le message qui a été lu dans un tampon d'entrée, un module de calcul arithmétique pour la réalisation des diverses opérations, un module de calcul combinatoire, et une mémoire sauvegardant les valeurs des pixels (voir figure 4.5.).

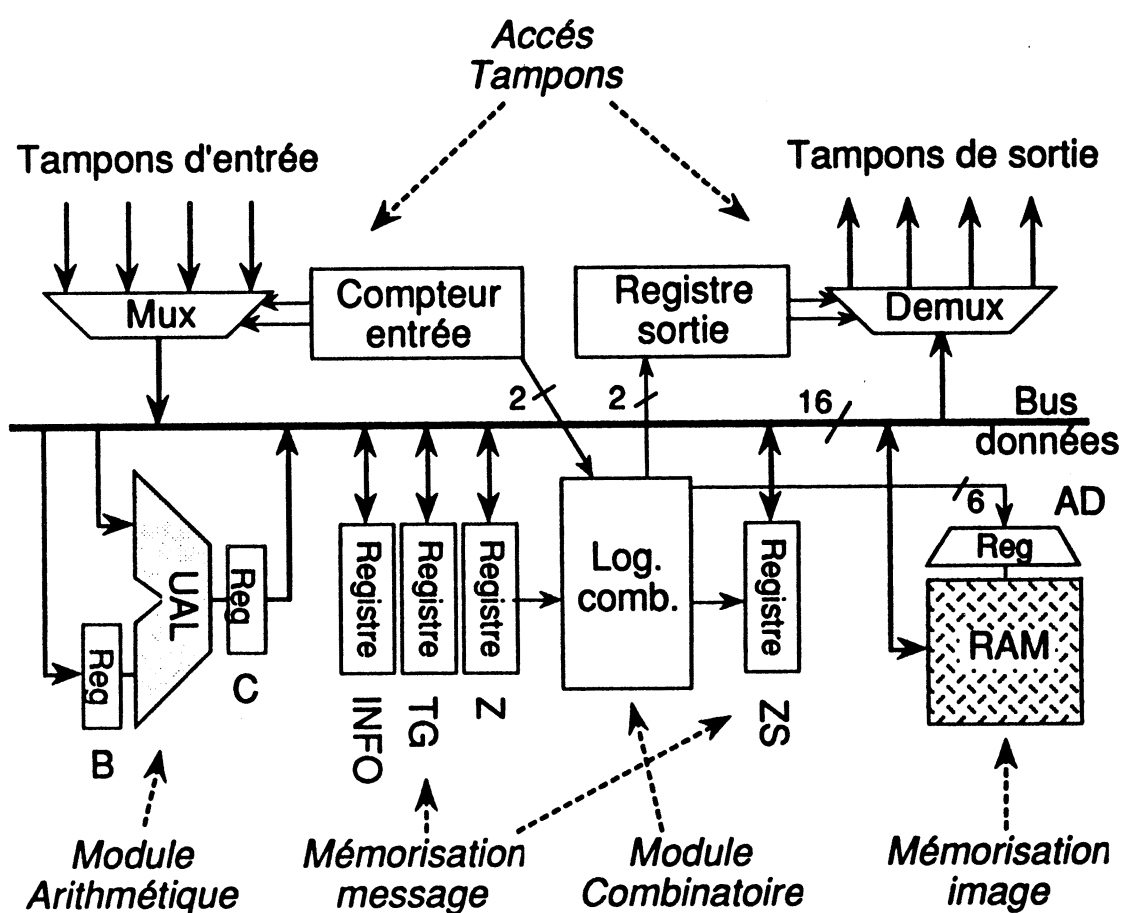


Figure 4.5. La partie opérative

Maintenant nous allons affiner la description de ces différents modules en présentant leur fonctionnement et leur réalisation.

### 4.2.4.1. Accès aux tampons

La sélection d'un tampon d'entrée est réalisée par un compteur modulo 4, qui permet une scrutation circulaire. Ce compteur de 2 bits (Ce1, Ce0) adresse un tampon d'entrée par l'intermédiaire d'un multiplexeur et démultiplexeur. Le tableau suivant donne le tampon sélectionné en fonction de la valeur du compteur ; nous avons utilisé un codage circulaire dans le sens trigonométrique en commençant arbitrairement par le nord.

| Ce1 | Ce0 | valeur | tampon sélectionné |
|-----|-----|--------|--------------------|
| 0   | 0   | 0      | Nord               |
| 0   | 1   | 1      | Ouest              |
| 1   | 0   | 2      | Sud                |
| 1   | 1   | 3      | Est                |

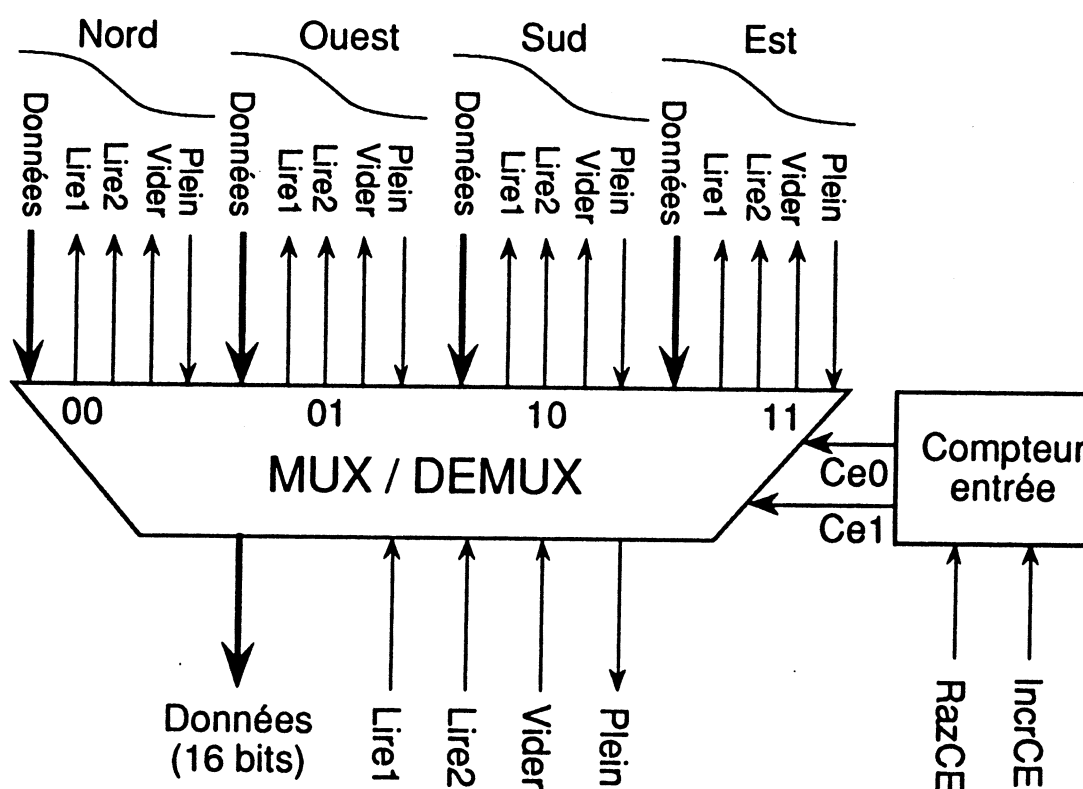


Figure 4.6. Accès aux tampons d'entrée

La figure 4.6. illustre le principe d'adressage des tampons d'entrée. Le compteur est commandé par 2 signaux : un signal de remise à zéro *razCE* et une commande d'incrémentatation *incrCE*. Il est composé d'un registre maître/esclave de 2 bits bouclé sur lui-même, avec une partie combinatoire réalisant l'incrémentatation.

L'adressage du tampon de sortie destinataire est fait suivant le même principe (voir figure 4.7.). Le mécanisme permettant de multiplexer les commandes des tampons est commandé par un registre de 2 bits (S1, S0) contenant le tampon sélectionné. Cette valeur provient d'une partie combinatoire calculant le côté de sortie du message, elle est enregistrée dans le registre à l'aide de la commande *EcrS*. Le codage des 4 tampons de sortie est le même que celui utilisé pour les tampons d'entrée.

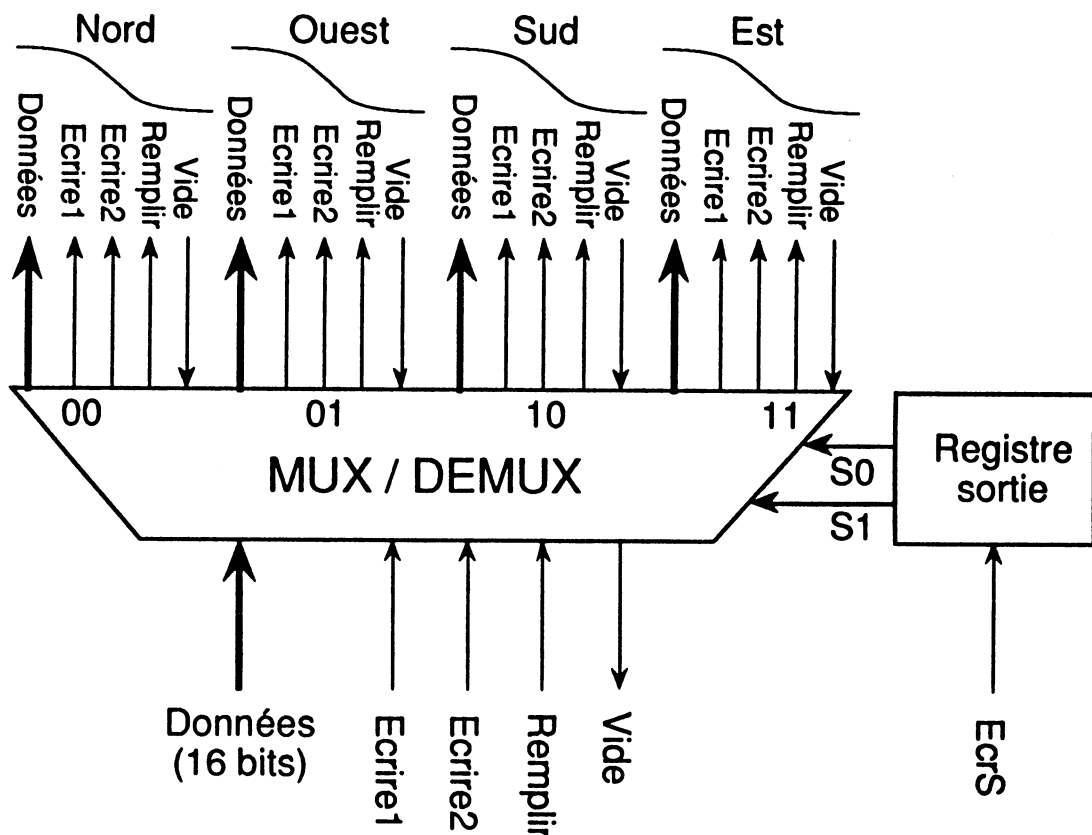


Figure 4.7. Accès aux tampons de sortie

#### 4.2.4.2. Mémorisation du message

Après avoir détecté la présence d'un message dans un tampon d'entrée, la cellule va lire ce message et le stocker dans la partie opérative, afin de vider le tampon au plus vite si le message peut être acheminé et traité.

Le fait de libérer les tampons d'entrée au début du traitement permet une bien meilleure activité globale du réseau cellulaire. Le message lu est sauvegardé dans 3 registres de 16 bits, qui sont directement connectés au bus interne de la cellule (voir figure 4.8.).

Les deux registres stockant les champs n°2 et 3 (TG et INFO) du message sont de simples registres ayant une commande d'écriture, et une de lecture.

Par contre, le registre stockant le champ n°1 ( $Z_{\text{cellule}}$ , TYPE, S et TC) est plus complexe.

Il est muni en entrée d'un multiplexeur permettant d'écrire, soit tout le champ (grâce à la commande *ecrZ*), soit uniquement les bits de  $Z_{\text{pixel}}$  (grâce à la commande *ecrZP*). Ceci est nécessaire, car ce registre est également utilisé pour stocker les points d'entrée successifs dans les pixels traversés de la cellule. Pour la même raison, il est muni d'une commande *invSTC* permettant d'inverser les 2 bits de poids faibles (S et TC), et donc le registre à ce niveau est un maître/esclave bouclé sur lui-même. Pour la lecture, le registre est commandé par deux signaux : un signal *lirZP* qui écrit la valeur de  $Z_{\text{pixel}}$  sur le bus, et un signal *lirZC* qui écrit la valeur de  $Z_{\text{cellule}}$  (ce qui permet de calculer les points de sortie au niveau du pixel, ou au niveau de la cellule).

Nous avons besoin d'un autre registre pour sauvegarder les coordonnées du point de sortie du message. En effet, le calcul est effectué au tout début du traitement, mais il se peut que le message soit passé tout à fait en fin de traitement (pour un message de type projection par exemple).



Ce simple registre de 16 bits est commandé par plusieurs signaux. Le signal *ecrZCS* permet l'enregistrement du point de sortie de la cellule (cette information provient du module de calcul arithmétique par l'intermédiaire du bus), et le signal *ecrSTCTYPE* permet la sauvegarde du type du message (cette information vient directement du champ n°1 du message reçu), et de S et TC caractérisant l'angle de sortie du rayon (cette information est calculée par le module combinatoire). De plus, la commande *lirZS* permet l'écriture du mot complet sur le bus afin de le transmettre à la cellule destinataire. La commande *razTYPE* permet de remettre à zéro le type du message, ce qui correspond à la génération d'un message de type transparence.

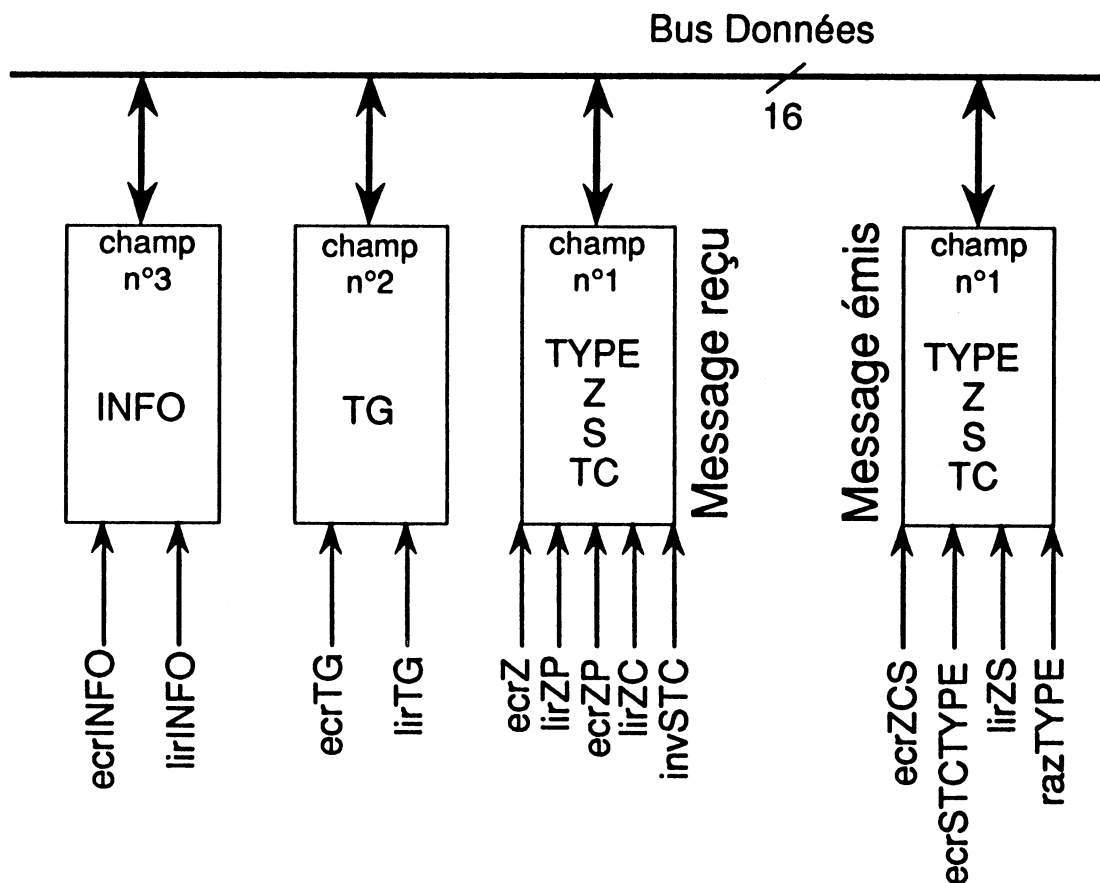


Figure 4.8. Registres mémorisant le message

### 4.2.4.3. Module de calcul arithmétique

#### Les opérations à réaliser

Ce module de la partie opérative est l'un des plus importants, puisqu'il va être utilisé pour la réalisation des opérations arithmétiques, que doit exécuter la cellule pour traiter les messages.

Dans le cas où l'angle d'incidence du rayon est inférieur à  $\frac{\pi}{4}$ , la cellule détermine le côté de sortie du message en comparant la tangente à la coordonnée du point d'entrée dans la cellule par soustraction. Dans tous les cas, le point de sortie est évalué grâce à une soustraction. Le calcul du facteur LONG demande la réalisation d'une division, et la mise à jour du pixel (pour un message de type épandage) ou de la projection en cours de calcul (pour un message de type projection) nécessite une multiplication et une addition.

En résumé, les opérations d'addition et de soustraction portent sur des opérandes de 16 bits, la multiplication met en jeu un multiplicande de 16 bits et un multiplieur de 8 bits pour un produit de 16 bits (les poids forts), et la division est réalisée avec un dividende de 8 bits, un diviseur de 16 bits et un quotient de 8 bits (le dividende est décalé de manière à être considéré comme un mot de 24 bits).

#### Choix des algorithmes

L'algorithme de multiplication employé découle de la classique méthode par additions et décalages. Cette technique demande un matériel élémentaire (un additionneur et des registres à décalage), et un temps de calcul de l'ordre du nombre de bits du résultat [BAE80]. L'usage d'un multiplieur de type cellulaire a été écarté car il nécessite une implantation matérielle coûteuse, ce qui est en contradiction avec l'esprit de notre architecture à grains les plus fins possibles.

En ce qui concerne la division, c'est le même problème : elle est réalisée de manière séquentielle, en utilisant l'algorithme à base de soustractions et décalages, à partir d'un soustracteur et de registres.

Les additionneurs peuvent se classer en trois catégories : l'additionneur séquentiel à propagation de retenue (qui a un temps d'exécution de l'ordre du nombre de bits à additionner :  $O(n)$ ), l'additionneur à retenue bondissante ( $O(\sqrt{n})$ ), et l'additionneur à anticipation de retenue ( $O(\log_2 n)$ ) [MUL86].

L'efficacité de notre additionneur est d'autant plus importante, qu'il est utilisé pour réaliser toutes les opérations arithmétiques (addition, soustraction, multiplication et division). Les opérandes traités sont sur 16 bits, par conséquent l'additionneur à anticipation de retenue n'est pas intéressant pour nous, puisque  $\sqrt{n} = \log_2 n = 4$ , quand  $n = 16$ .

Les deux premiers additionneurs ont été étudiés en détail ; l'additionneur à retenue bondissante permet d'obtenir de bonnes performances, si les cellules de base utilisées pour le réaliser sont rapides. Des simulations électriques à partir d'une bibliothèque dans la technologie CMOS 2 microns et 2 niveaux de métal du CNET (Centre National d'Etudes des Télécommunications) donnent un temps d'addition sur 16 bits de 10 nanosecondes pour l'additionneur séquentiel, et de 7 nanosecondes pour l'additionneur à retenue bondissante (voir annexe 4).

L'additionneur implanté lors de la réalisation du circuit est un additionneur séquentiel, car avec la bibliothèque utilisée, il est plus rapide. Ceci s'explique par le fait, que les portes nécessaires pour engendrer les rebonds de la retenue ont une entrance relativement élevée (4 et 5), et donc le temps de traversée de ces portes est beaucoup plus grand que celui des portes utilisées pour réaliser l'addition (portes d'entrance 2 et 3).

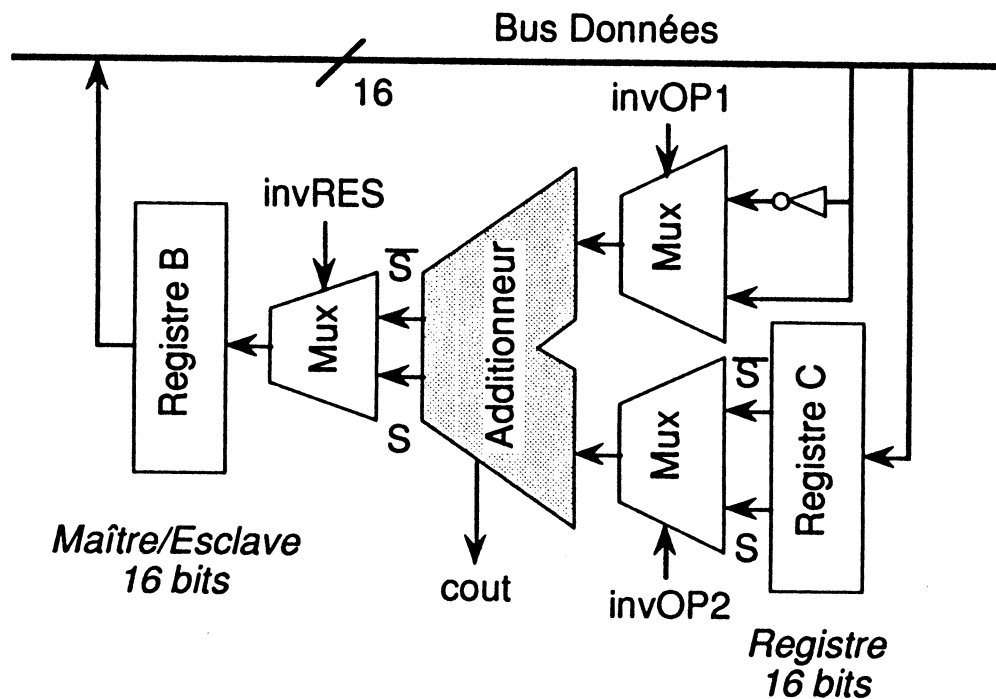
## **Le module**

Nous allons présenter les parties du module en fonction des opérations que nous souhaitons réaliser.

Nous travaillons avec des valeurs non signées positives ; la soustraction est effectuée en utilisant l'additionneur, et en complémentant bit à bit, le premier des opérandes et le résultat. Cette technique demande en entrée et en sortie de l'additionneur, des

multiplexeurs permettant de sélectionner les valeurs ou les valeurs complémentées (voir figure 4.9.).

L'additionneur émet un signal *cout*, qui rend compte du débordement, dans le cas d'une addition, et d'un résultat négatif, dans le cas d'une soustraction.



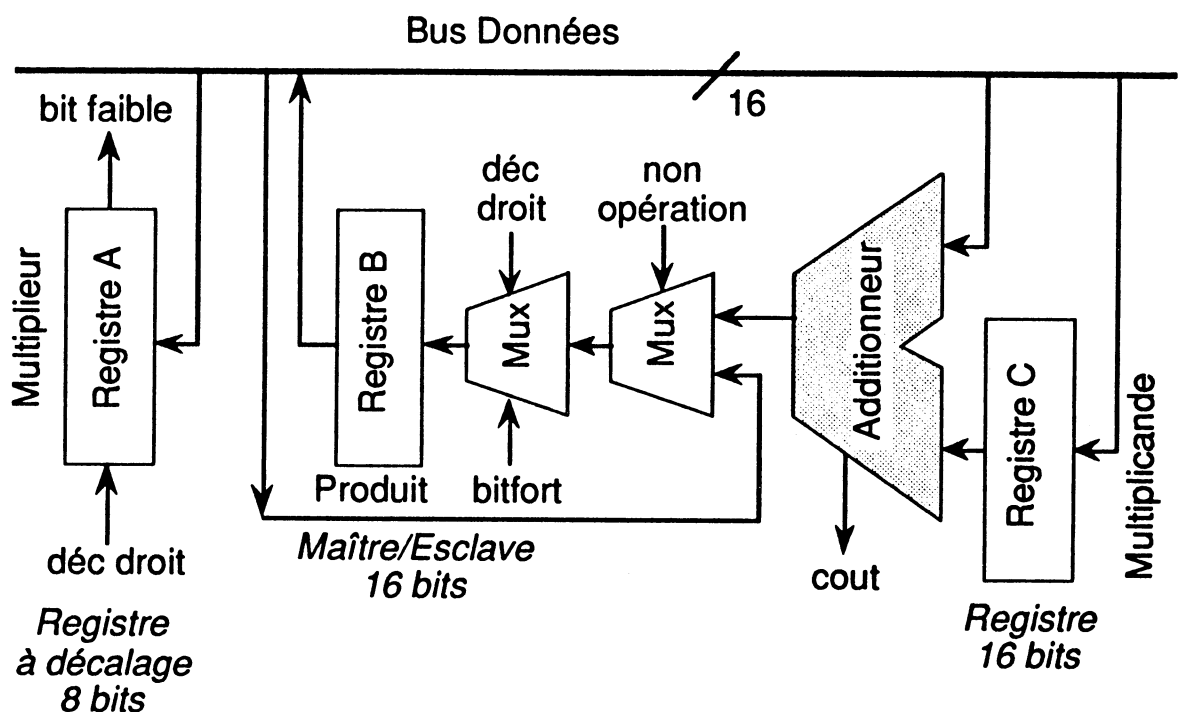
**Figure 4.9.** Réalisation des additions et des soustractions

Ce matériel permet une grande souplesse d'utilisation, notamment dans la réalisation des soustractions, puisqu'elles peuvent être faites sans distinction entre les deux opérands : il en découle donc un gain de temps, dû à l'absence de manipulation des opérands.

L'algorithme de multiplication demande quelque matériel supplémentaire, dont un registre à décalage pour stocker le multiplieur (8 bits), un multiplexeur en sortie d'additionneur qui va réaliser le décalage du produit partiel, et un autre qui inhibera l'addition, lorsque le bit testé du multiplieur est nul (non opération) (voir figure 4.10.).

Au départ de l'opération, le multiplicande est rangé dans le registre C, et le registre B contient la valeur nulle. A chaque cycle d'horloge, la commande de lecture du registre C, la commande d'écriture et de

lecture du registre B, la commande de décalage à droite (division par 2) du registre A et du multiplexeur sont activées. Le bit de poids faible du multiplieur (registre A) commande le multiplexeur, qui permet de choisir la somme provenant de l'additionneur s'il est à 1, ou la non opération s'il est à 0. Il se peut que l'addition génère une retenue *cout*, alors dans ce cas, pour conserver ce bit de l'information, le bit fort entrant dans le multiplexeur réalisant le décalage à droite est forcé à 1 (*bitfort*). En un seul cycle d'horloge, sont donc effectuées une opération d'addition et décalage, et la multiplication est réalisée en 8 cycles seulement.

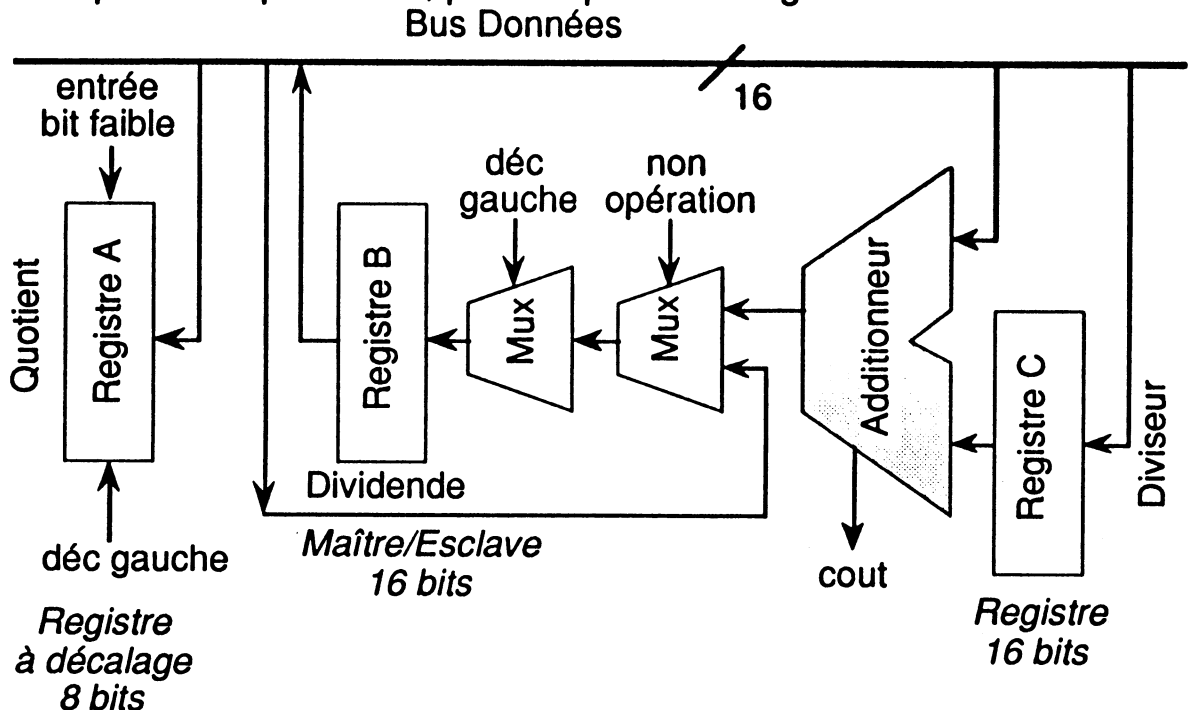


**Figure 4.10.** Réalisation de la multiplication

L'algorithme de division implémenté est également très simple.

Au départ de l'opération, le diviseur est stocké dans le registre C et le dividende dans le registre B (les 8 bits de poids fort, et les autres étant nuls). A chaque cycle, la commande de lecture du registre C, les commandes de lecture et d'écriture du registre B, la commande du multiplexeur réalisant le décalage à gauche (multiplication par 2), et la commande de décalage à gauche du registre A sont activées. Si la

soustraction (dividende - diviseur) est réalisable ( $cout = 0$ ), il faut stocker le nouveau dividende, sinon ( $cout = 1$ ) la commande du multiplexeur forçant la non opération est activée (elle est générée directement par  $cout$ ). Chaque valeur de  $cout$  donne un bit inversé du produit, qui est rangé dans le registre A. La figure 4.11. illustre ce fonctionnement, les multiplexeurs permettant d'inverser les opérandes et le résultat de l'addition, afin de réaliser les soustractions successives n'ont pas été représentés, pour ne pas surcharger le schéma.

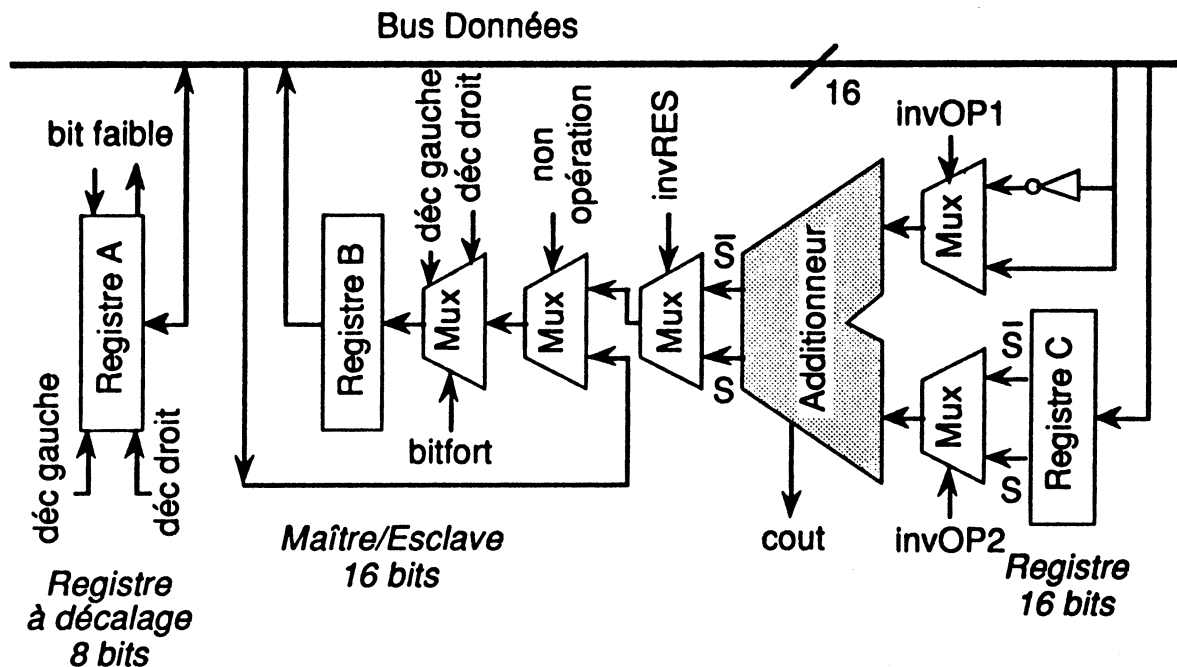


**Figure 4.11.** Réalisation de la division

Une division demande 8 cycles d'horloge, puisque les opérations de soustraction et de décalage sont effectuées en chacun d'eux (pour un quotient de 8 bits). Ce n'est pas tout à fait vrai, parce qu'au début de l'opération, il faut décaler à gauche le dividende du fait du codage des informations TG et Z, il faut donc 9 cycles d'horloge pour réaliser la division.

La figure 4.12. présente le module complet, avec tout le système de multiplexage permettant la réalisation des 4 opérations arithmétiques. Nous avons regroupé les deux multiplexeurs réalisant le

décalage à gauche et le décalage à droite en un seul, ceci a été possible, parce que ces opérations ne sont jamais faites en même temps.



**Figure 4.12.** Le module arithmétique

Dans le séquençement des opérations, la multiplication est réalisée juste après la division. Le quotient calculé sert par la suite de multiplicateur, et cette valeur est déjà présente dans le registre A au moment de commencer la multiplication. Par ce fait, le registre A n'a pas besoin d'être connecté au bus ; mais cependant cette connexion existe, parce que ce registre est utilisé au cours du traitement de la cellule, pour sauvegarder la coordonnée du point d'entrée du pixel traité. Le registre A est commandé par le signal *ecrA* qui permet de stocker la valeur présente sur le bus, et par le signal *lirA* qui permet d'accéder à la valeur enregistrée dans le registre. Le registre C est muni d'une commande d'écriture *ecrC*, et le registre B d'une commande d'écriture *ecrB*, et d'une commande de lecture *lirB*. D'autre part, les registres B et C possèdent une commande de remise à zéro (*razB* et *razC*), et les registres A et C, une commande de mise à un (*setA* et *setC*).

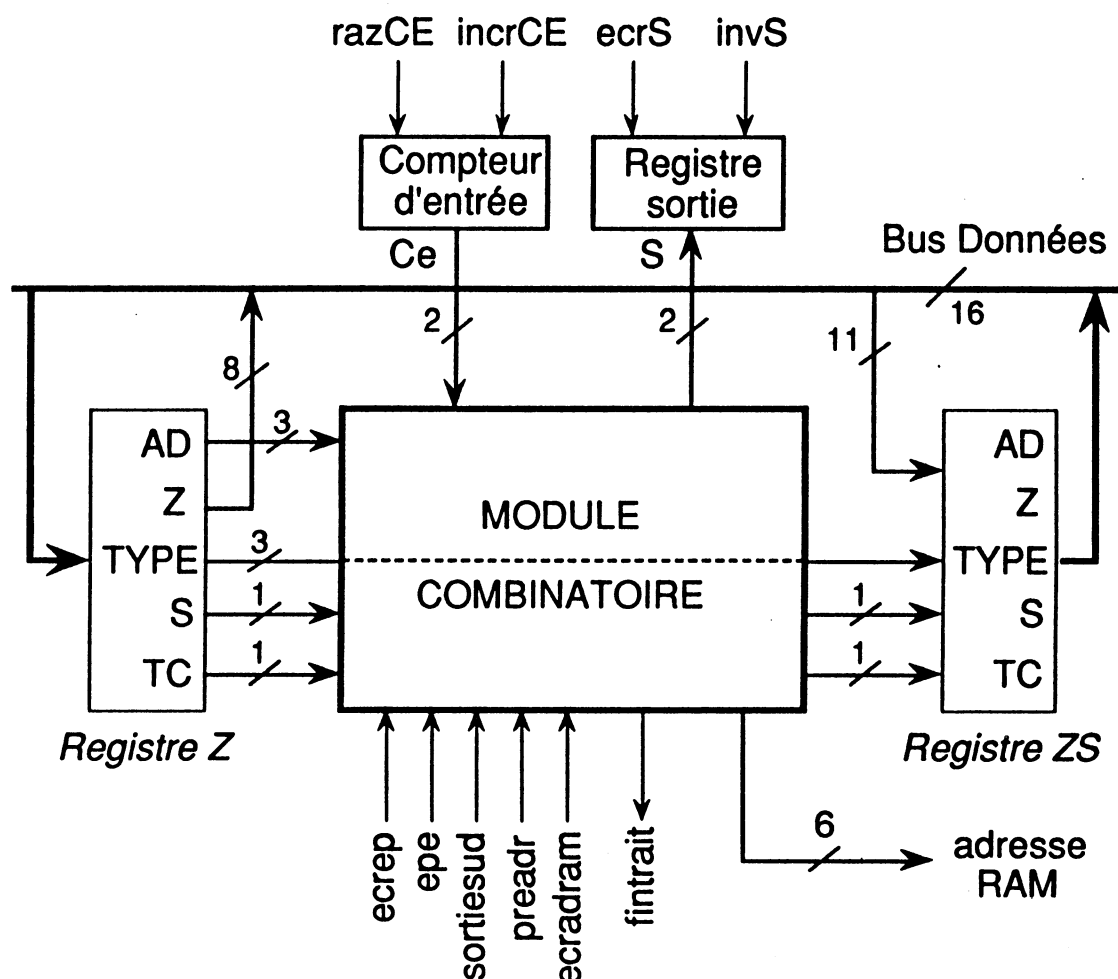
Un compteur modulo 8 est nécessaire pour les opérations de multiplication et de division, afin de fournir au séquenceur un signal lui précisant que le nombre d'itérations a été atteint. Ce compteur est réalisé à l'aide d'un registre maître/esclave de 3 bits, bouclé sur lui-même avec une partie combinatoire réalisant l'incrémentation. Il est commandé par les signaux *razcpt* (remise à zéro) et *incrcpt* (incrémentation), et il fournit le signal *fincpt*.

Ce module possède un chemin critique relativement long, les opérandes provenant de registres doivent traverser l'additionneur et quatre multiplexeurs, pour fournir un résultat correct arrivant dans le registre B. Les simulations montrent que le temps nécessaire à un calcul arithmétique est tout à fait correct, si chaque élément possède de bonnes caractéristiques. Il faut rappeler que les multiplications et divisions sont effectuées en un temps qui est de l'ordre de  $n$  (nombre de bits du multiplieur et du quotient), ce qui est excellent pour les algorithmes utilisés, car en général elles sont réalisées en un temps d'ordre de  $2n$ . De plus pour nous  $n$  est petit ( $n = 8$ ), et donc il était inutile de mettre en œuvre des réalisations plus sophistiquées, et coûteuses en surface.

#### 4.2.4.4. Module de calcul combinatoire

Ce module a plusieurs fonctions. Il permet de calculer le côté de sortie du message traité, et le côté de sortie pour tous les pixels traversés par le rayon associé au message. Il sert également à mettre à jour certaines parties du message (les informations sur l'angle : S et TC), et à calculer les adresses successives des pixels à traiter. La figure 4.13. montre les entrées et sorties, et les principales commandes du module.



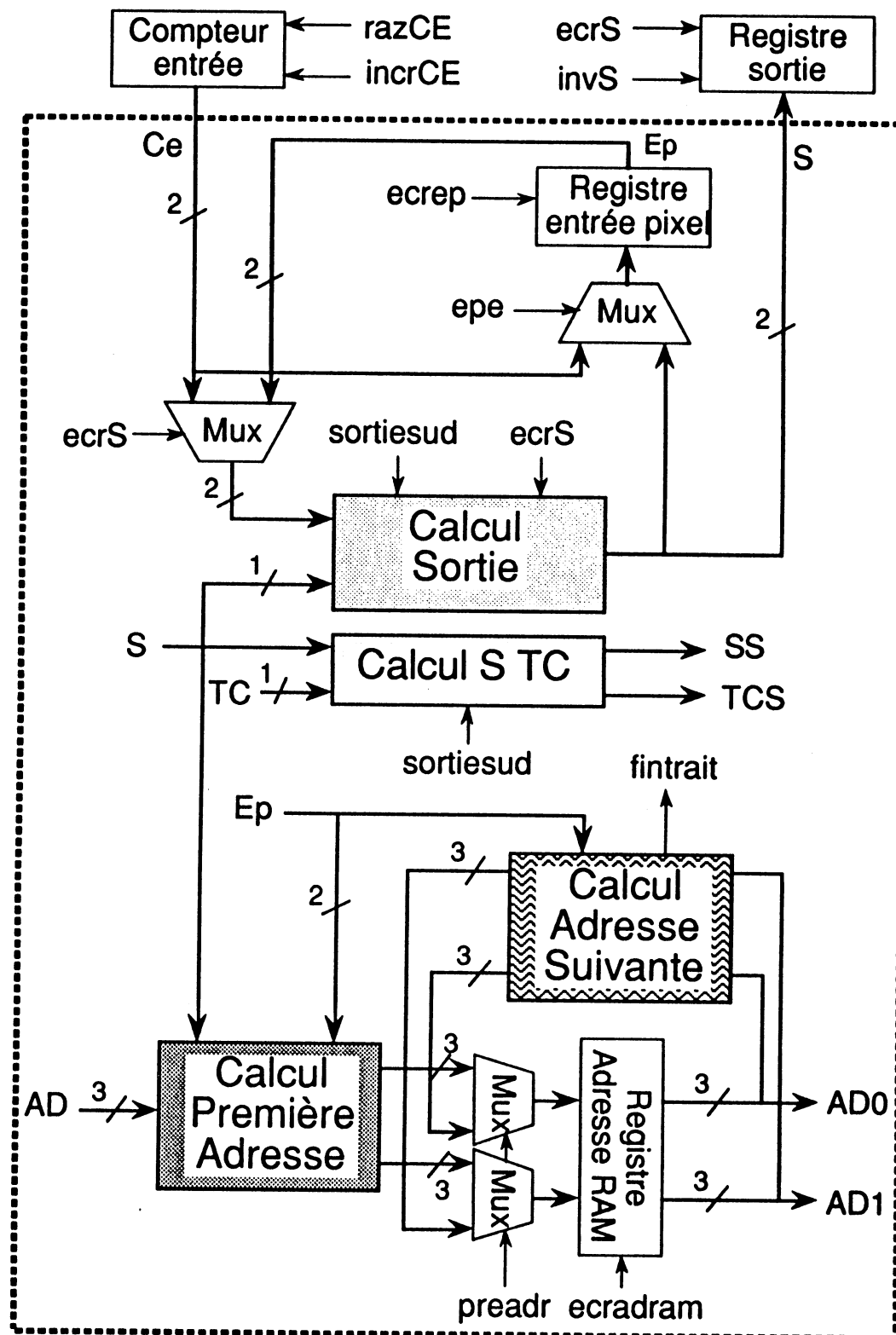


**Figure 4.13.** Le module combinatoire

Avant de donner la signification précise de chacune des commandes du module, nous allons détailler la structure interne de cette partie.

La figure 4.14. présente cette structure, qui peut se décomposer en trois blocs plus ou moins distincts.

Un premier bloc fonctionnel permet de déterminer le côté de sortie du message, en fonction du côté d'entrée contenu dans le compteur, et de commandes générées par la partie contrôle de la cellule.



**Figure 4.14. Structure du module combinatoire**

Ce bloc est articulé autour d'une logique combinatoire *Calcul Sortie*.

Un registre maître/esclave de 2 bits permet de stocker le côté d'entrée du pixel courant, et un système bouclé à base de multiplexeurs permet d'initialiser ce registre et de calculer les valeurs successives. Le côté de sortie de la cellule est enregistré grâce à la commande *ecrS*, le côté d'entrée dans le pixel courant  $E_p$  est stocké grâce à la commande *ecrep*, et la commande *epe* sélectionne la valeur avant ou après la combinatoire selon le cas. Le registre sauvegardant le côté de sortie de la cellule possède une commande *invS*, qui inverse le bit fort, ce qui permet de générer un message se propageant dans le sens inverse du message reçu lors de la sortie de l'image.

Un second bloc est constitué uniquement par de la logique combinatoire : *Calcul STC*.

Cette partie évalue les 2 bits *SS* donnant le signe de la tangente de l'angle d'incidence dans la cellule suivante, et *TCS* indiquant le passage de la tangente ou de la cotangente à cette cellule, en fonction des informations reçues *S* et *TC*. Ce bloc, ainsi que le précédent, est commandé par le signal *sortiesud*, indiquant une sortie du message du côté opposé à celui de l'entrée.

Un troisième bloc détermine l'adresse mémoire du pixel en cours de traitement. Il est organisé autour de deux parties combinatoires, l'une permet de calculer l'adresse du premier pixel traité (*Calcul Première Adresse*), et l'autre de calculer l'adresse suivante (*Calcul Adresse Suivante*).

L'adresse du pixel est stocké dans un registre maître/esclave de 6 bits, dont les 3 de poids fort *AD1* déterminent la ligne du pixel, et les 3 de poids faible *AD0* la colonne du pixel. Par un système de boucle, les adresses successives peuvent être évaluées. La commande *preadr* permet de choisir la première valeur, et la commande *ecradram* provoque l'enregistrement. La première valeur dépend des 3 bits *AD* du message reçu, du signe de la tangente de l'angle d'incidence du rayon *S*, et du côté d'entrée du message dans la cellule. Une valeur suivante dépend de la valeur précédente, et du côté d'entrée du rayon dans le

pixel  $E_p$ , cette valeur est calculée à l'aide d'un additionneur-soustracteur, qui incrémente ou décrémente un des 2 champs de l'adresse du pixel (AD1 ou AD0), pour réaliser la progression dans le tableau de pixels. La partie combinatoire *Calcul Adresse Suivante* émet un signal *fintrait*, qui indique au séquenceur la fin de traitement des pixels traversés par le rayon.

Les schémas logiques de quelques parties sont donnés en annexe 4.

#### 4.2.4.5. Mémoire de l'image

L'image traitée par la cellule est de taille 8 par 8 pixels, elle est stockée dans une mémoire RAM de 64 mots de 16 bits. Chaque valeur de pixel est un mot de 16 bits, et l'adresse permettant l'accès à un pixel est sur 6 bits, organisés en 2 mots de 3 bits (AD1, et AD0) afin de conserver la structure des lignes et des colonnes de la sous-image traitée (voir figure 4.15).

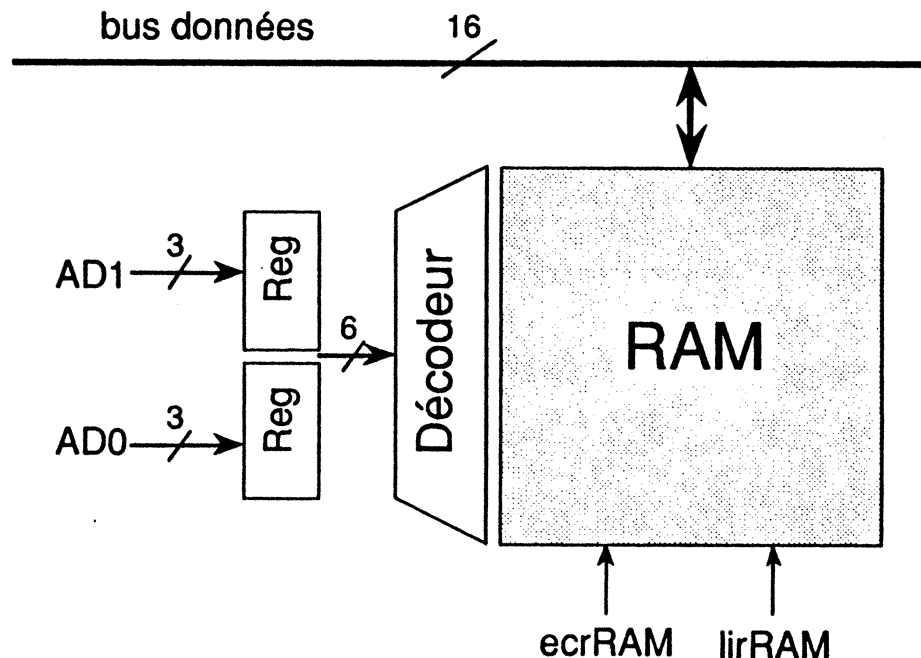


Figure 4.15. Mémorisation de l'image

La commande *ecrRAM* permet d'enregistrer dans la mémoire la valeur présente sur le bus, et la commande *lirRAM* permet d'écrire la valeur d'un pixel contenue dans la mémoire sur le bus.

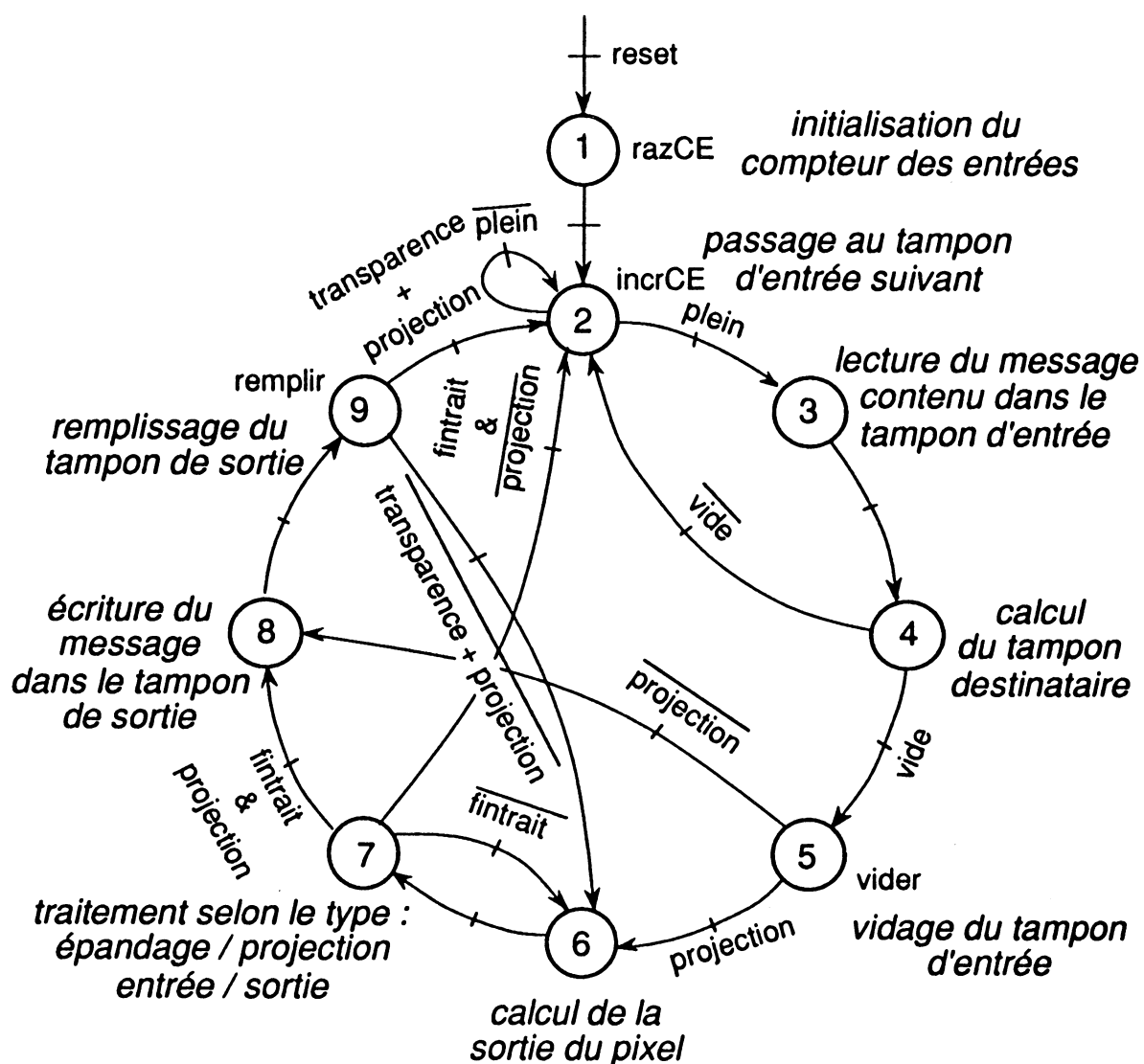
Lors du *reset* de la machine, nous devons avant tout traitement remettre à zéro tous les mots de la mémoire. Pour faire cela, le séquenceur va parcourir toute la mémoire grâce à un compteur modulo 64 permettant d'adresser tous les mots, afin d'y enregistrer le mot nul.

### 4.2.5. Le séquencement

La partie opérative de la cellule est pilotée par un séquenceur, qui assure le bon ordonnancement des opérations nécessaires au traitement cellulaire.

La figure 4.16. donne le graphe simplifié des principales tâches que le séquenceur distribue. Cet ordonnancement a été fait de manière à transmettre le message en cours de traitement, le plus vite possible. C'est pourquoi le type est testé très tôt, afin de distinguer les messages de type projection, car uniquement pour les messages de ce type, le passage du message se fait en fin de traitement.

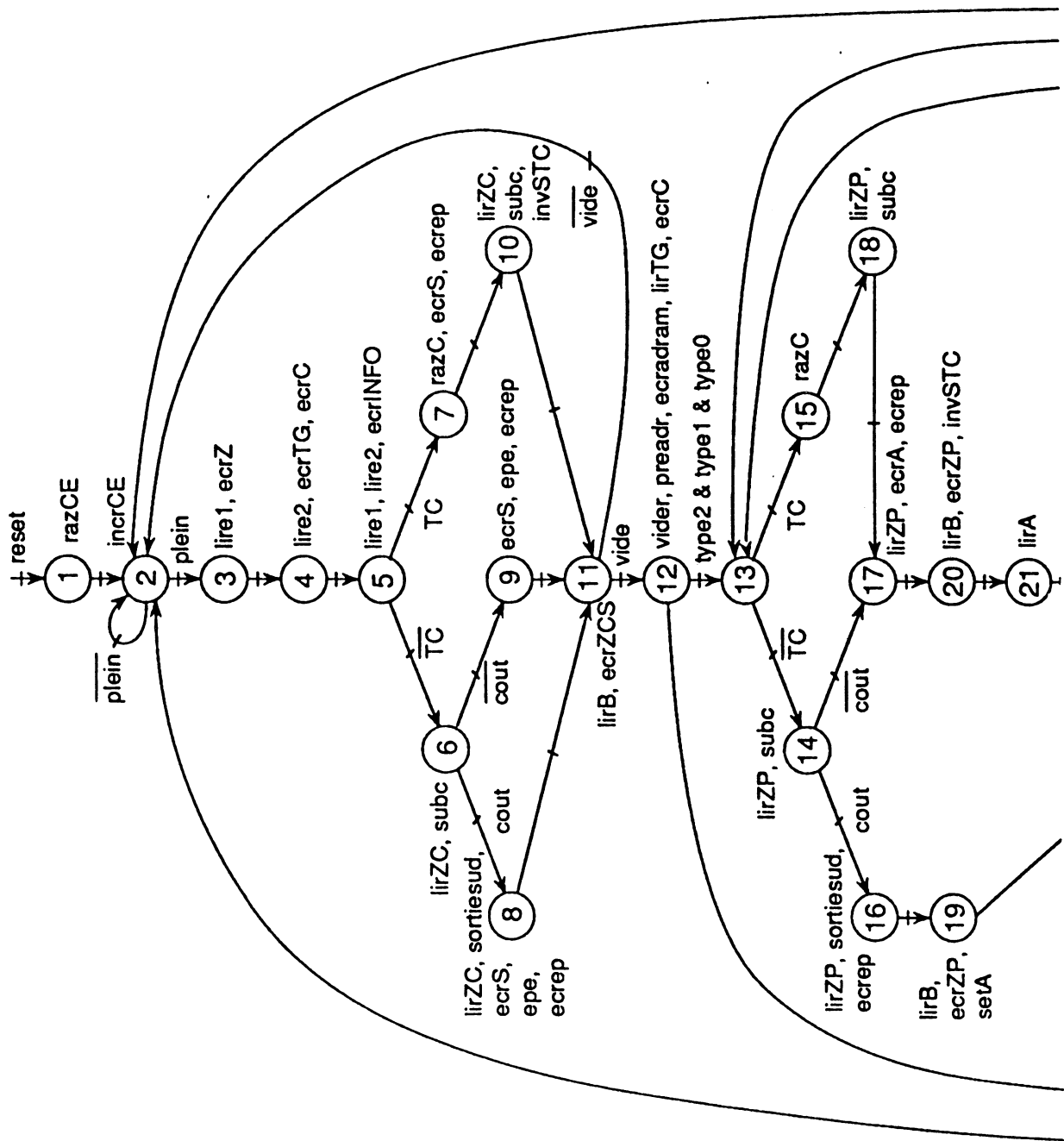
Sur ce graphe, il n'apparaît que quelques commandes (*razCE*, *incrCE*, *vider*, *remplir*), et quelques transitions (*plein* : le tampon d'entrée est plein, *vide* : le tampon de sortie est vide, *fintrait* : les pixels sont tous traités). Chaque nœud est en fait formé par un sous-graphe, qui réalise la tâche associée.

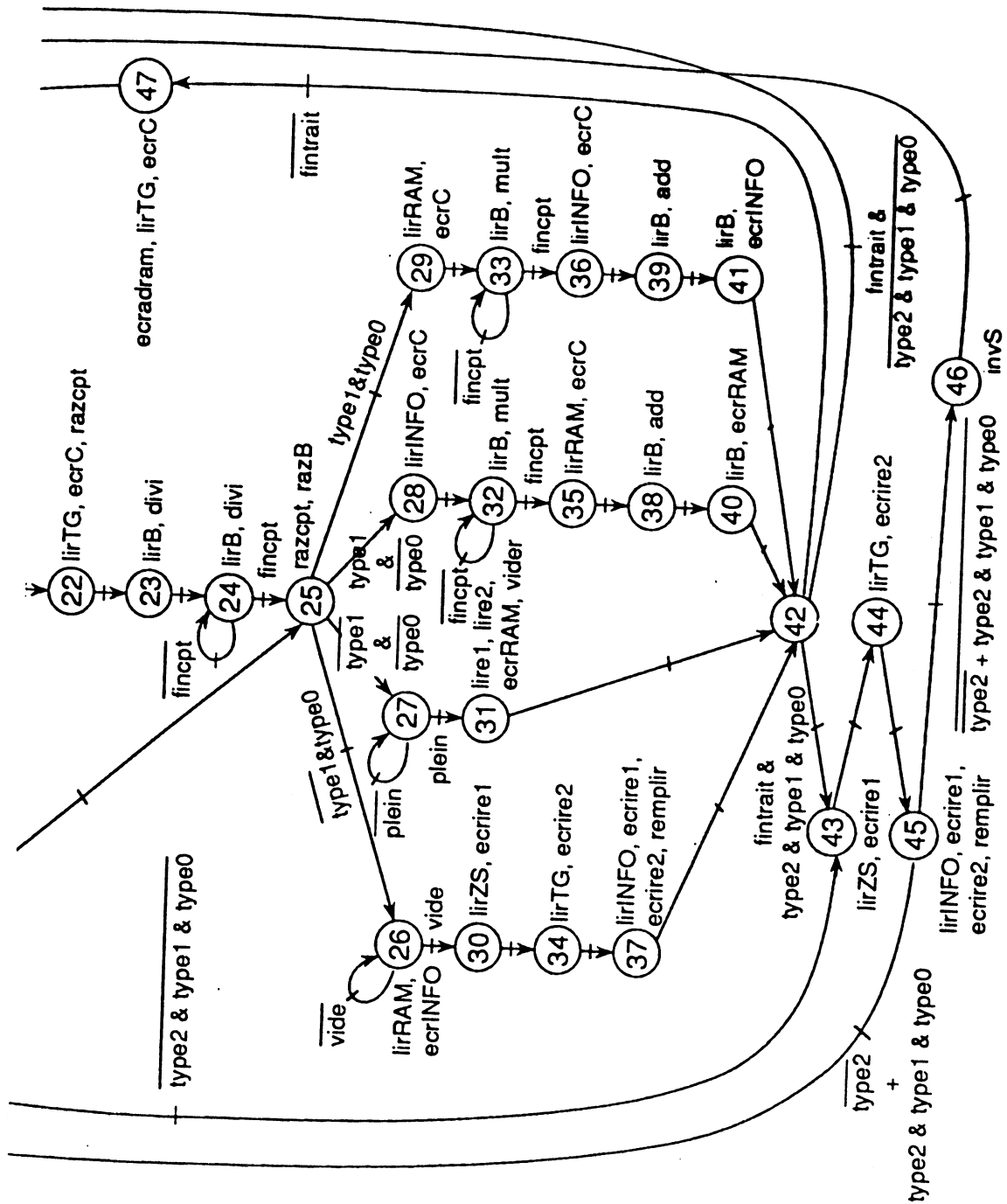


**Figure 4.16.** Séquencement des opérations

La figure 4.17. présente le graphe de séquencement complet. Il possède 47 états, génère 42 commandes et dispose de 12 signaux d'entrée.

Figure 4.17. Graphe de séquençement







## **4.3. LES SIMULATIONS**

### **4.3.1. Présentation des simulations**

L'architecture complète de la cellule, avec ses 8 tampons d'entrée et sortie, a été simulée grâce à HELIX, le simulateur comportemental de Silvar-Lisco. Elle a été décrite en terme de modules élémentaires : registres, tampons, compteur, mémoire, blocs combinatoires et arithmétiques, séquenceur. Toutes les données ont été manipulées en tant qu'entiers, et les commandes et compte-rendus en tant que booléens.

Ces simulations ont permis de mettre au point l'aspect architectural de la cellule, du fonctionnement du module le plus élémentaire, au séquençement complet des opérations, jusqu'au traitement d'une série de messages. L'architecture de la cellule étant relativement complexe, cette étape a été indispensable à la conception. En annexe 4, sont présentés les schémas descriptifs de l'architecture simulée et la description HELIX de quelques blocs élémentaires (mémoire, registres, multiplexeurs).

Une série de messages a été définie, afin de tester l'ensemble de l'architecture : les différents blocs fonctionnels, mais aussi le séquençement des opérations. Ce jeu de tests a été utilisé par la suite, lors de la validation logique de la cellule.

Un premier jeu de messages a permis de tester le traitement de la cellule pour un message de type épandage, dans tous les cas possibles. Pour un point d'entrée fixe sur un côté de la cellule, nous avons étudié le message sortant de la cellule, en fonction du côté d'entrée et de la valeur de la tangente, et exploré la mise à jour des pixels traversés.

La figure 4.18. illustre un tel message, la coordonnée d'entrée dans la cellule est  $Z_{\text{cellule}} = 875$  (la dimension de la cellule est 2048), l'adresse du pixel est  $AD_{\text{pixel}} = 3$  (le nombre de pixels adressable est 8), et la coordonnée d'entrée dans le pixel est  $Z_{\text{pixel}} = 107$  (la dimension du pixel est 256). Ce message est de type épandage (TYPE = 6), l'angle d'incidence du rayon est compris entre 0 et  $\frac{\pi}{4}$  (S = 0, et TC = 1), et la valeur de la tangente est TG = 16251 (c'est à dire 0,496 en valeur réelle). La valeur à épandre est INFO = 100.

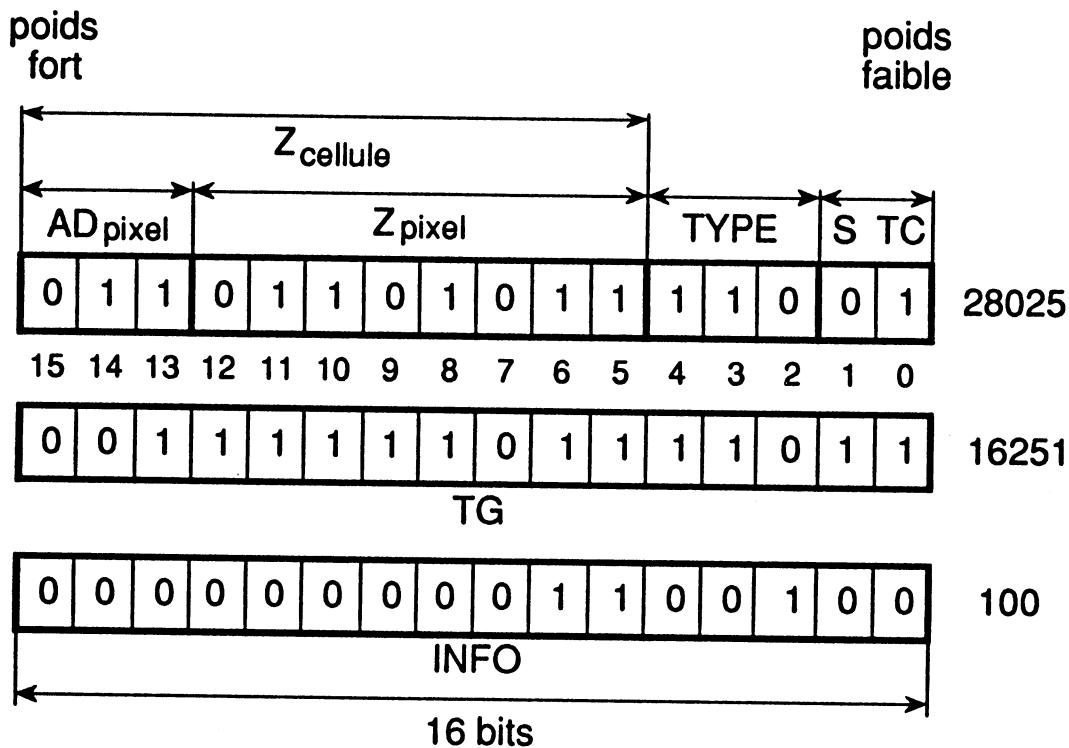
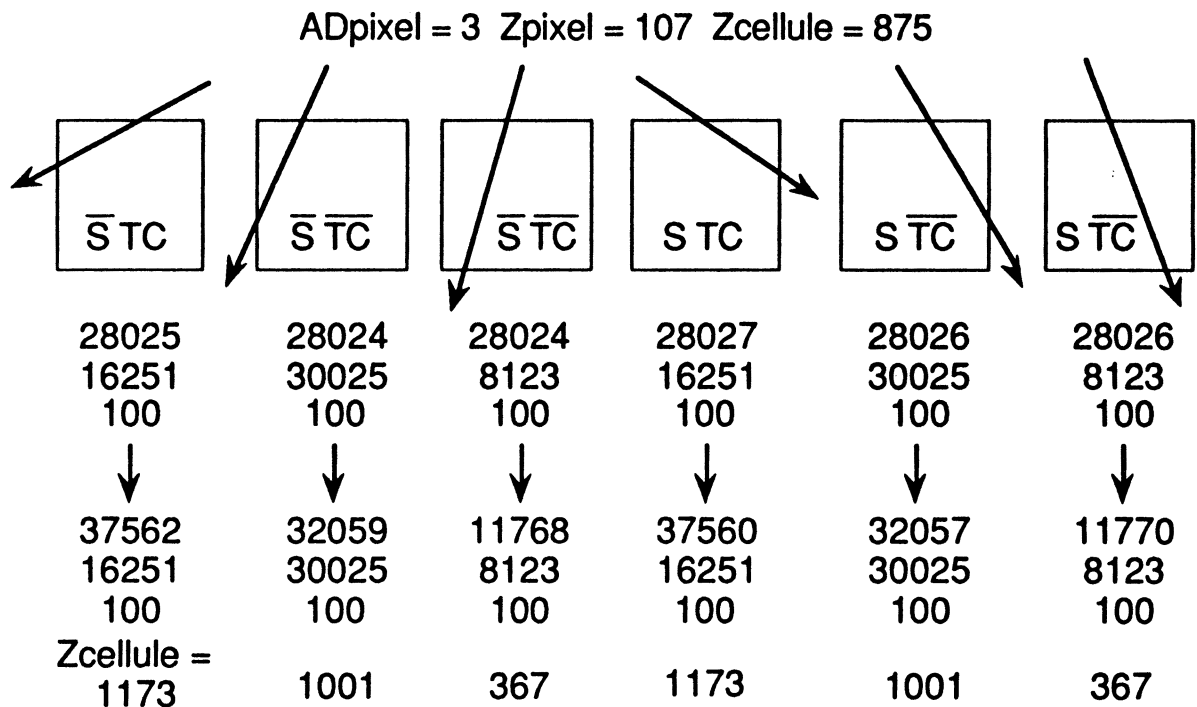


Figure 4.18. Un message test

Ce message (28025, 16251, 100) écrit dans un tampon d'entrée quelconque de la cellule, produira un message sortant du côté adjacent au côté d'entrée dans le sens trigonométrique, dont les 3 champs ont les valeurs (37562, 16251, 100). La figure 4.19. indique d'autres cas de figures, pour des angles d'incidence du rayon différents.



**Figure 4.19.** Messages utilisés pour tester le calcul de la sortie

Après avoir mis au point les modules permettant l'accès aux tampons et le calcul de la destination et du message, nous nous sommes intéressés au traitement au niveau des pixels, pour vérifier le bon fonctionnement de la mise à jour, et aussi les algorithmes de division et multiplication implémentés.

La figure 4.20. illustre cette mise à jour, pour l'envoi du message de type épandage (28025, 16251, 100) par le côté nord ; la cellule formée de 8 par 8 pixels est adressable à l'aide des informations AD1 et AD0. Dans ce cas, les pixels traversés sont traités dans le sens inverse de la circulation du message. Pour tous ces pixels, le tableau joint à la figure indique le coefficient LONG et la valeur du pixel.

Ces vérifications ont été faites également avec des messages de type épandage possédant d'autres propriétés.

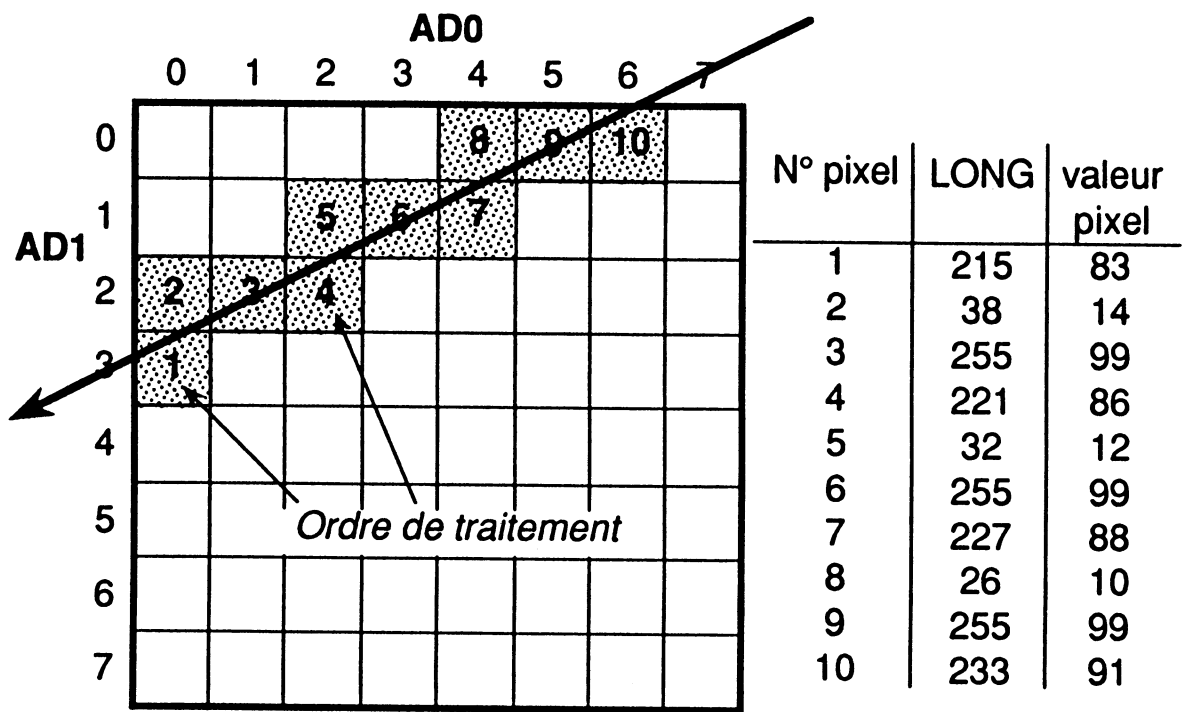


Figure 4.20. Mise à jour des pixels

Un deuxième jeu de messages a permis de tester le traitement des messages de type projection. Après avoir envoyé le message de type épandage exposé précédemment (28025, 16251, 100), nous avons envoyé sur le même chemin un message de type projection (28029, 16251, 0), qui a généré un message sortant de la cellule (37566, 16251, 601). La valeur de projection calculée est INFO = 601, et les calculs au niveau des pixels ont été satisfaisants.

Un dernier jeu de tests a été élaboré pour examiner le traitement des messages de type entrée\_image, sortie\_image et transparence. Après avoir envoyé à la cellule un message de type entrée\_image (12306, 0, 0), et une série de 8 messages de type transparence ((12290, 0, 10), ...) contenant chacun d'eux une valeur de pixel de la ligne adressée (la troisième ici), nous avons envoyé un message de type sortie\_image (12310, 0, 0) sur la même ligne et récupéré du même côté, 8 messages de type transparence possédant les valeurs précédemment émises. Le cheminement des messages de type transparence a été également testé.

### **4.3.2. Validation de l'architecture**

Il faut réaffirmer l'intérêt des simulations comportementales HELIX, qui ont permis de définir et de figer l'architecture de la cellule. De plus, la structure proposée a pu être simulée, et donc validée. La description des modules élémentaires en HELIX reprend quasiment les équations logiques associées, et par conséquent, nous pouvons dire que cette validation est également logique.

## **4.4. IMPLANTATION**

### **4.4.1. Le logiciel SOLO 1200**

Un circuit intégrant une cellule de base du réseau a été réalisé, grâce au logiciel SOLO 1200, dans la technologie CMOS 2 microns et 2 niveaux de métal de ES2 (European Silicon Structures) [ES288]. Il a été dessiné sur du matériel du CIME (Centre Interuniversitaire de Micro-Electronique) : stations de travail Sun, et fondu par l'intermédiaire du CMP (Circuits Multi-Projets). Ce logiciel de CAO permet de concevoir des circuits VLSI (Very Large Scale Integration) précaractérisés.

La solution d'utiliser une méthode automatique de conception, grâce à un tel outil, plutôt qu'une méthode plus manuelle pour dessiner les masques (full-custom) a été choisie pour plusieurs raisons. D'une part, dans le temps qui nous était imparti, la réalisation d'un circuit d'une telle complexité (plusieurs milliers de transistors) n'était pas envisageable. Et d'autre part, bien qu'un outil automatique produise des circuits moins denses, il assure une plus grande sûreté dans la conception, ce qui est essentiel pour un circuit de recherche.

L'utilisation du logiciel SOLO1200 est très simple. Nous dessinons le circuit sous forme de schémas logiques, à partir de cellules de bibliothèques fournies par le fondeur, et les étapes de génération des dessins des masques : placement des portes et routage sont entièrement automatiques. Un simulateur permet de valider les descriptions schématiques à un niveau plus fin que le niveau logique,

puisque'il tient compte des capacités des interconnexions et de la sortance des portes.

Les programmes de placement et routage génèrent un circuit structuré en plusieurs colonnes, séparées par des canaux verticaux de routage. Chaque colonne est formée de rangées contenant les portes et de canaux horizontaux de routage. Chaque rangée est organisée autour de deux lignes de diffusion (une de type N et une de type P), que traversent verticalement des barrettes de silicium polycristallin, afin de réaliser les transistors. Deux rails d'aluminium (de premier niveau : alu1) entourent la rangée pour amener l'alimentation et la masse. Les connexions entre les différentes portes transitent par les canaux de routage en aluminium (de premier et second niveau : alu1 et alu2).

Les bibliothèques disponibles au moment de la réalisation du circuit contenaient toutes les portes logiques élémentaires (inverseur, ou, nonou, nonet, ...), des points de mémorisation (registres simples, et de type maître/esclave, bascules de type D, RS, et JK), des portes spécifiques (porte 3 états, porte de passage, amplificateurs), des blocs de mémoire RAM de taille fixe et des plots d'entrée et sortie.

Les dessins des masques de quelques cellules de base utilisées pour réaliser le circuit sont présentés en annexe 4.

## **4.4.2. Réalisation du circuit**

### **4.4.2.1 Principe**

Le circuit réalisé inclut une cellule et uniquement les 4 tampons d'entrée. En effet, le circuit étant prévu pour construire un réseau par simple juxtaposition sur une matrice carrée, les tampons de sortie d'une cellule sont les tampons d'entrée adjacents des 4 cellules voisines.

Le travail d'implantation a consisté à reprendre les schémas utilisés pour les simulations comportementales HELIX, et à les compléter. La validation logique a été faite de façon ascendante, des

éléments de base jusqu'au circuit total, grâce aux messages tests déjà utilisés lors des simulations comportementales.

#### **4.4.2.2 La partie opérative**

Les différents modules du chemin de données de la cellule ont été conçus indépendamment avant l'assemblage final. Ils ont été simulés et placés, afin de connaître leur surface, pour en déduire leur densité d'intégration.

Un tampon réalisé à partir des cellules de base de la bibliothèque comporte 1142 transistors pour une surface de 2,5 mm<sup>2</sup>, soit une densité de 457 transistors par mm<sup>2</sup>.

Le système d'accès aux tampons d'entrée et sortie comprend 244 transistors pour une surface de 0,5 mm<sup>2</sup>, soit une densité de 488 transistors par mm<sup>2</sup>.

La mémorisation du message demande 5 mm<sup>2</sup> pour 2214 transistors, soit une densité de 443 transistors par mm<sup>2</sup>.

Le module arithmétique est le module le plus important avec une surface de 6 mm<sup>2</sup> pour 2694 transistors, soit une densité de 449 transistors par mm<sup>2</sup>. L'additionneur (400 transistors) a été réalisé à partir de la cellule additionneur 1 bit de la bibliothèque, et le temps de traversée de la retenue dans un tel additionneur 16 bits est de 30 nanosecondes. Un additionneur à retenue rebondissante a été étudié, mais il possédait de moins bonnes performances (voir annexe 4). Le chemin critique de ce module est parcouru en un temps de 60 nanosecondes.

Le module combinatoire contient 854 transistors pour une surface de 1,8 mm<sup>2</sup>, soit une densité de 474 transistors par mm<sup>2</sup>.

La mémorisation de l'image traitée par la cellule est réalisée à l'aide de 2 blocs mémoire disponibles en bibliothèque, chacun d'eux d'une capacité de 64 mots de 8 bits. Un bloc permet de stocker les bits de poids fort des valeurs des pixels de l'image 8 par 8, et l'autre les bits de poids faible. Ces cellules de base sont des mémoires RAM de type statique, où chaque bit est stocké grâce à une bascule D (2 inverseurs

bouclés) et demande 6 transistors. Ce module contient environ 7000 transistors pour une surface de 4 mm<sup>2</sup>, soit 1750 transistors par mm<sup>2</sup>.

La partie opérative complète sans les tampons d'entrée et sortie, contient 13362 transistors pour 18 mm<sup>2</sup>, soit une densité de 742 transistors par mm<sup>2</sup>. Avec les 4 tampons d'entrée, cette partie comporte 17930 transistors pour 28 mm<sup>2</sup>, soit une densité de 640 transistors par mm<sup>2</sup>.

Le tableau de la figure 4.23. (paragraphe 4.4.2.4.) donne la récapitulation du nombre de transistors des différentes parties du circuit. En annexe 4, sont présentés les schémas logiques de certaines parties du circuit et des résultats de simulation.

### 4.4.2.3 Le séquenceur

Pour réaliser la partie contrôle, nous avons utilisé un codage "1 parmi n" des états du graphe de séquençement. Cette solution a été adoptée, dans un premier temps uniquement à cause de la facilité de réalisation. En effet, le logiciel de conception utilisé SOLO 1200 ne comporte pas de générateur de séquenceur, ni de PLA (Programming Logic Array), donc la solution consistant à coder les états de façon minimale et à générer les commandes grâce à un PLA a été écartée.

Notre solution consiste à utiliser un point mémoire de type maître/esclave par place du graphe, et à reconstruire la connectique du graphe à l'aide de ces points mémoire. Au cours du traitement, un jeton et un seul, se promène dans le graphe, et il est utilisé pour générer les commandes associées à cet état.

La figure 4.21. illustre ce procédé, et présente un exemple de réalisation à partir d'un petit graphe de séquençement à 5 places et 4 commandes.

Les points mémoire possèdent 2 commandes : une commande d'écriture correspondant au signal d'horloge H, et une commande permettant l'initialisation activée par le *reset* du séquenceur. Il s'agit d'une commande de mise à 1 pour la première place active (p1), et une



commande de mise à 0 pour les autres. La génération des commandes est réalisée à l'aide de portes logiques et des valeurs contenues dans les différents points mémoire ; elles sont synchronisées sur la phase haute de l'horloge.

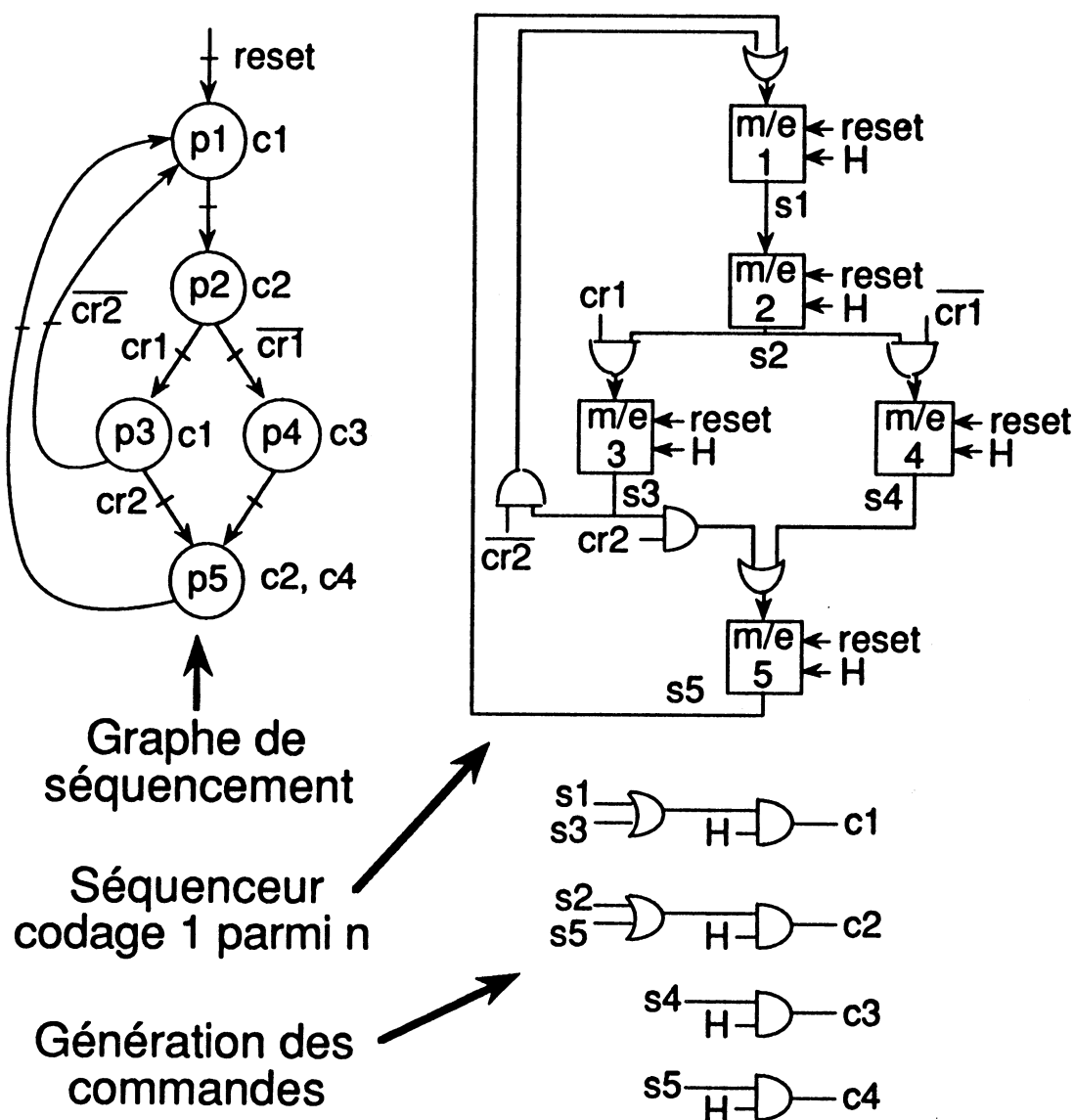
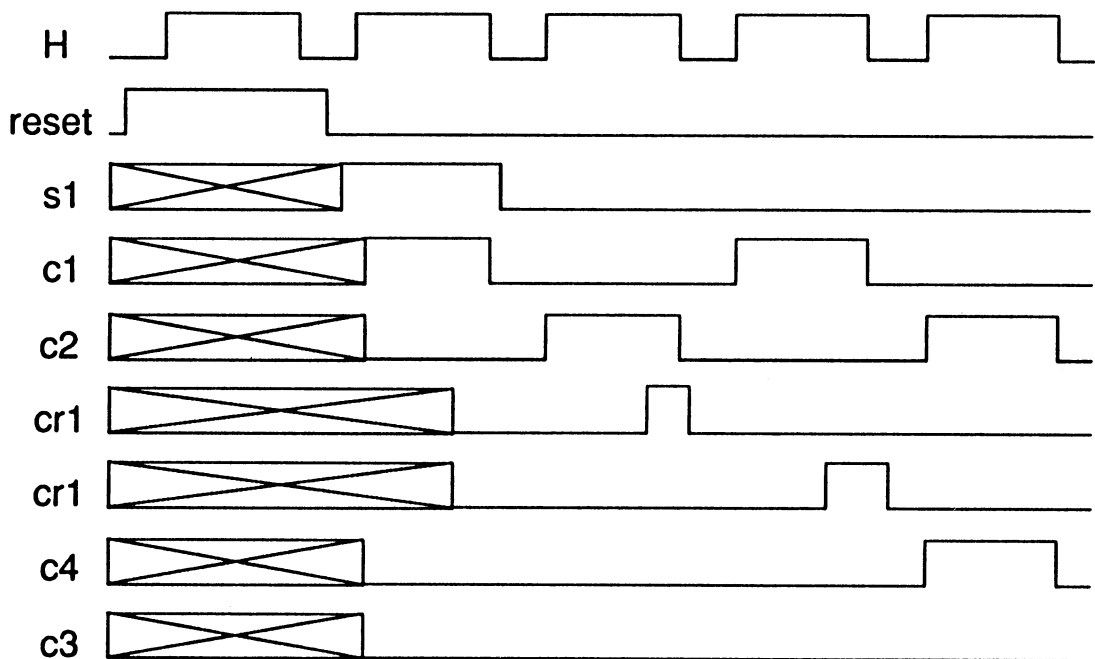


Figure 4.21. Principe du séquenceur

La figure 4.22. représente un chronogramme illustrant le fonctionnement du séquenceur donné précédemment en exemple. Pendant la phase haute de l'horloge, la partie opérative de la machine est activée, et le jeton passe de l'esclave du point mémoire courant au

maître du suivant. Pendant la phase basse de l'horloge, le jeton n'a qu'à passer du maître à l'esclave du point mémoire courant ; c'est pourquoi, il est astucieux d'utiliser une horloge irrégulière (possédant une phase haute longue et une phase basse courte) pour améliorer le temps total de calcul.

Une commande générée au cours d'un état, produira une condition de transition (un compte-rendu de la partie opérative), qui peut être considéré pour sortir de cet état. Ceci n'est en général pas vrai avec les séquenceurs à base de PLA, car souvent les commandes et les compte-rendus ont besoin d'être mémorisés, ce qui produit un décalage entre les commandes et les transitions de 2 états.



**Figure 4.22.** Exemple de chronogrammes

Il s'avère que la réalisation de ce type de séquenceur n'est pas plus coûteuse que celle de l'autre solution envisagée à base de PLA. Cela s'explique par le fait que le nombre de places et de commandes est important (47 et 42), et donc qu'il aurait fallu un PLA de grande taille et beaucoup de points mémoire pour stocker les commandes.

Le séquenceur réalisé, possède 2110 transistors pour une surface de 4 mm<sup>2</sup>, soit une densité de 517 transistors par mm<sup>2</sup>.

#### 4.4.2.4 Le circuit complet

Les cellules de base sont les suivantes :

|                        |                                                   |
|------------------------|---------------------------------------------------|
| not                    | inverseur,                                        |
| or2, or3, or4, or5     | portes <u>ou</u> à 2, 3, 4, et 5 entrées,         |
| and2, and3             | portes <u>et</u> à 2 et 3 entrées,                |
| nor2, nor3, nor4, nor5 | portes <u>nonou</u> à 2, 3, 4, et 5 entrées,      |
| nand2, nand3           | portes <u>nonet</u> à 2 et 3 entrées,             |
| txpair                 | multiplexeur 2 vers 1,                            |
| xor                    | porte <u>ou-exclusif</u> ,                        |
| eqv                    | porte <u>égalité</u> (ou-exclusif inversé),       |
| buffer                 | amplificateur inversant,                          |
| sumadder               | additionneur 1 bit,                               |
| latcha                 | registre 1bit,                                    |
| ni tri buffer          | porte 3 états amplificatrice de 16 bits,          |
| bf003                  | porte logique $S = \overline{A1A2 + B1B2}$ ,      |
| bf008                  | porte logique $S = \overline{A1 + B1B2 + C1C2}$ , |
| bf013                  | porte $S = \overline{A1A2+B1B2B3+C1C2C3}$ ,       |
| ram64*8                | bloc RAM (64 mots de 8 bits).                     |

Pour information, nous pouvons donner les temps de traversée de quelques portes élémentaires : il est de 0,75 nanoseconde pour l'inverseur (not), de 1,5 nanosecondes pour le nonou et le nonet à 2 entrées (nor2 et nand2), de 2,75 nanosecondes pour le ou-exclusif (xor), et le bloc de RAM a un temps de lecture et d'écriture supérieur à 20 nanosecondes.

La figure 4.23. présente le nombre de cellules de base utilisées pour réaliser le circuit, le nombre de portes et la complexité (nombre de transistors) des principaux modules.

| modules<br>portes | tampon | accès<br>tampons | mémorisation<br>message | module<br>arithmétique | module<br>combinatoire | mémoire image | partie opérative | partie contrôle | circuit | nb de transistors<br>des portes |
|-------------------|--------|------------------|-------------------------|------------------------|------------------------|---------------|------------------|-----------------|---------|---------------------------------|
| not               | 18     | 6                | 31                      | 209                    | 40                     | 31            | 317              | 206             | 797     | 2                               |
| or2               |        |                  |                         |                        | 1                      |               | 1                | 10              | 17      | 6                               |
| or3               |        |                  |                         |                        |                        |               |                  | 2               | 3       | 8                               |
| or4               |        |                  |                         |                        |                        |               |                  | 1               | 2       | 14                              |
| or5               |        |                  |                         |                        |                        |               |                  | 1               | 1       | 20                              |
| and2              | 1      |                  |                         | 2                      | 2                      | 5             | 9                | 50              | 64      | 6                               |
| and3              |        |                  |                         |                        |                        |               |                  | 3               | 3       | 8                               |
| nor2              | 7      |                  | 2                       | 1                      | 7                      | 3             | 13               | 44              | 85      | 4                               |
| nor3              |        | 24               |                         |                        |                        |               | 24               | 7               | 31      | 6                               |
| nor4              |        |                  |                         |                        |                        |               |                  | 10              | 10      | 12                              |
| nor5              |        |                  |                         |                        |                        |               |                  | 2               | 2       | 18                              |
| nand2             |        |                  |                         |                        | 17                     |               | 17               | 1               | 18      | 4                               |
| nand3             |        |                  |                         |                        |                        |               |                  | 1               | 1       | 6                               |
| txpair            |        | 6                | 24                      | 193                    | 7                      | 18            | 248              | 96              | 344     | 4                               |
| xor               |        |                  |                         | 2                      | 12                     | 5             | 19               |                 | 19      | 12                              |
| eqv               |        |                  |                         |                        | 11                     |               | 11               |                 | 11      | 12                              |
| buffer            | 6      | 8                | 27                      | 42                     | 11                     | 9             | 97               | 2               | 123     | 8                               |
| sumadder          |        |                  |                         | 16                     |                        |               | 16               |                 | 16      | 24                              |
| latcha            | 48     |                  | 60                      | 4                      | 18                     |               | 82               | 48              | 322     | 10                              |
| ni tri buffer     | 1      |                  | 4                       | 2                      |                        |               | 6                |                 | 10      | 288                             |
| bf003             |        |                  | 10                      | 16                     | 11                     | 6             | 43               |                 | 43      | 8                               |
| bf008             |        |                  |                         |                        |                        |               |                  |                 | 1       | 10                              |
| bf013             | 16     |                  |                         |                        |                        |               |                  |                 | 64      | 16                              |
| ram64*8           |        |                  |                         |                        |                        | 2             | 2                |                 | 2       | 3500                            |
| nb de transistors | 1142   | 244              | 2214                    | 2694                   | 854                    | 7356          | 13362            | 2110            | 20616   |                                 |

Figure 4.23. Complexité du circuit

Après assemblage des modules composant le circuit, la surface totale (sans les plots) donnée par les programmes de placement et routage de SOLO 1200 est de 38 mm<sup>2</sup> (5,98 mm x 6,35 mm) pour 20616 transistors. La densité d'intégration du circuit est donc de 542

transistors par mm<sup>2</sup>. Les dessins des masques de certains modules sont présentés en annexe 4.

La figure 4.24. présente les entrées et sorties du circuit. Les données sont groupées en 4 flots d'entrée de 16 bits arrivant aux 4 tampons d'entrée de la cellule, et 1 seul flot de sortie de 16 bits qui sera connecté aux 4 circuits voisins lors de la réalisation du réseau. Ce flot de sortie correspond au bus de données de la partie opérative de la cellule ; et de ce fait, le test du circuit en sera grandement facilité.

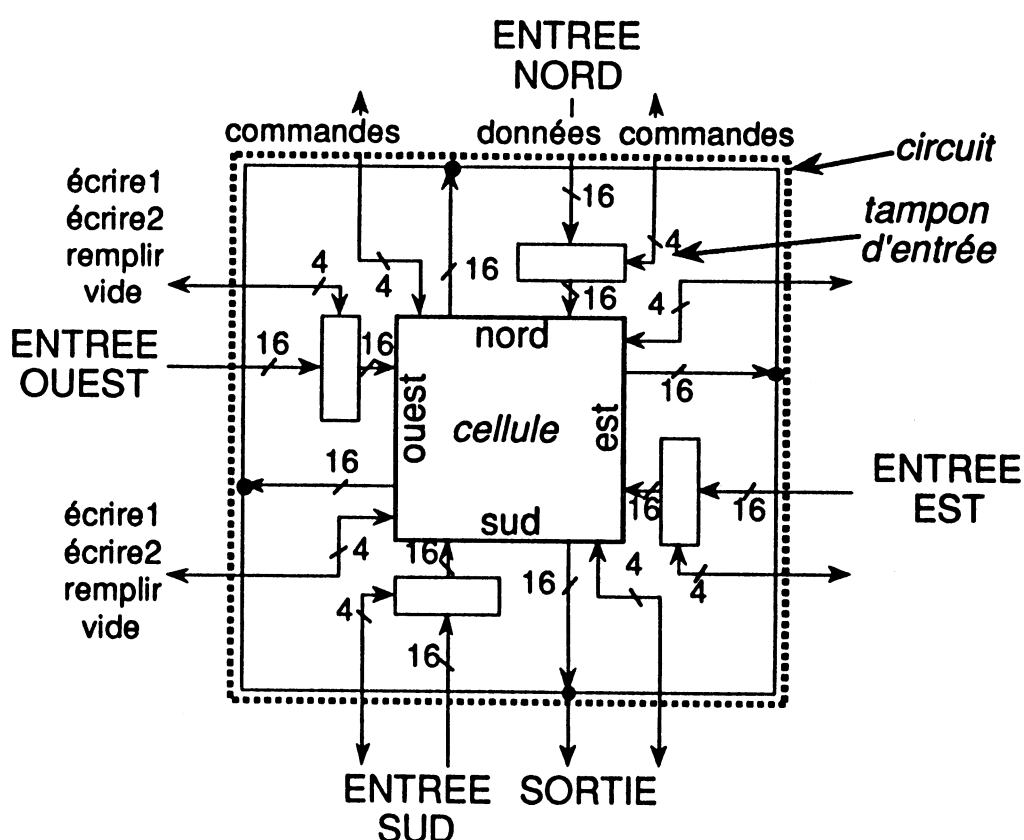


Figure 4.24. Les entrées et sorties du circuit

### 4.4.3. Le brochage du circuit

Le circuit possède 118 plots, dont 34 plots d'entrée, 80 plots de sortie, 2 plots d'alimentation de 5 volts et 2 plots de masse. La figure 4.25. donne le plan de brochage du circuit et la liste des plots.

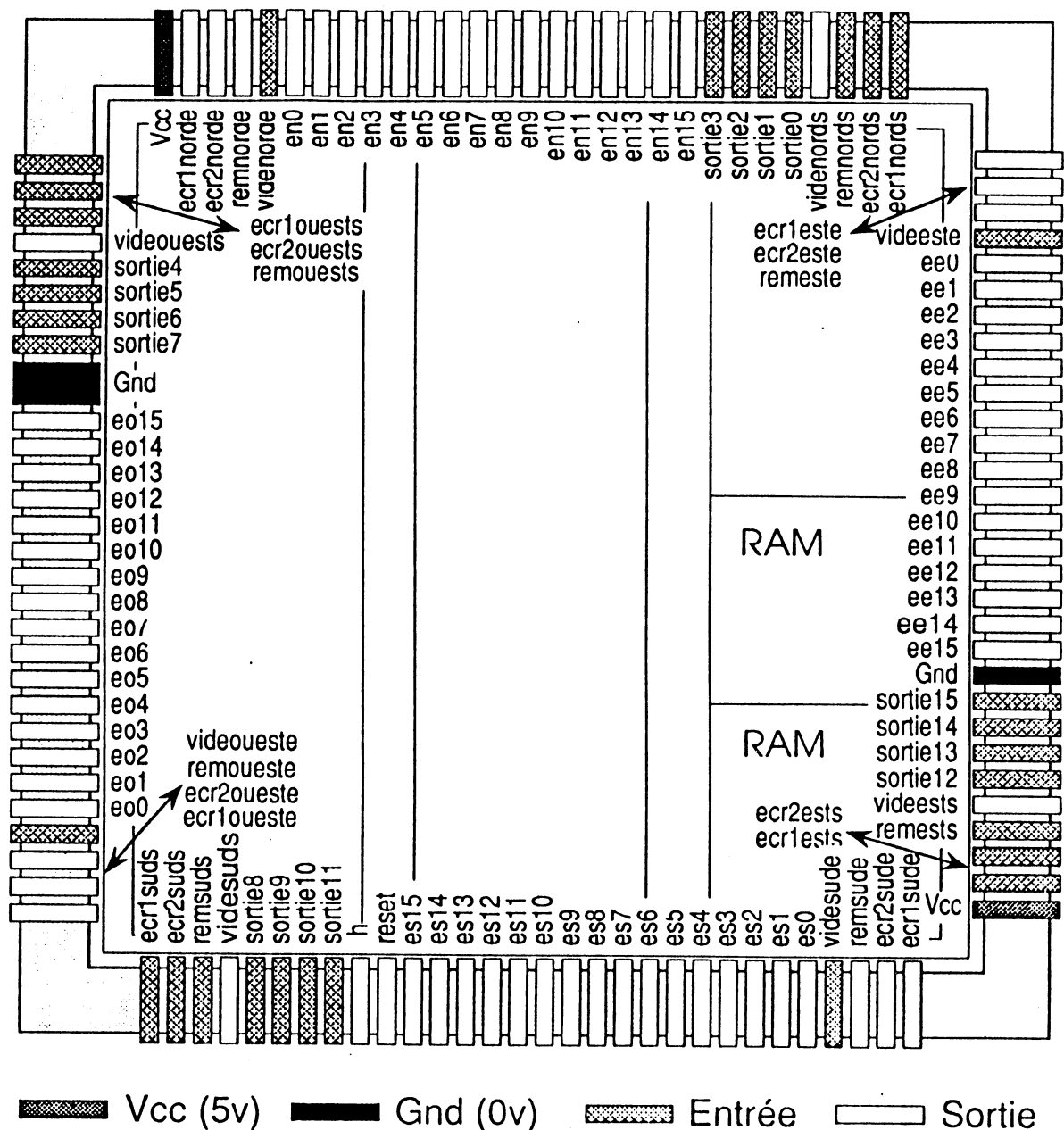


Figure 4.25. Brochage du circuit

Le circuit complet avec la couronne de plots occupe la surface de 51,5 mm<sup>2</sup> (6,97 mm x 7,39 mm), ce qui donne une densité d'intégration de 400 transistors par mm<sup>2</sup> en reprenant le nombre de 20616 transistors.

Le circuit est monté dans un boîtier de type Pin Grid Array de 124 pattes mesurant 3,3 cm de côté. En annexe 4, sont donnés le plan de montage et la liste des pattes du boîtier.

Le test du circuit a été effectué grâce à un analyseur logique et un générateur de mots. Nous avons développé une carte pour regrouper les entrées et sorties du circuit afin de réaliser les connections à l'analyseur le plus simplement possible. Comme vecteurs de test nous avons repris les messages tests utilisés lors des simulations. Sur les 10 circuits actuellement montés en boîtier, 3 ont un fonctionnement satisfaisant.

#### **4.4.4. Performances**

##### **4.4.4.1. Surface**

Le circuit a une bonne densité d'intégration (400 transistors par  $\text{mm}^2$ ), compte tenu du fait que c'est un circuit précaractérisé (les dessins des masques ont été générés automatiquement), et qu'il a été réalisé avec une technologie relativement peu performante (CMOS 2 microns).

L'objectif de cette intégration n'était pas de produire un circuit très performant et très dense, comme nous aurions pu le faire avec une technologie plus fine (1 micron, ou moins) et une technique de fabrication industrielle. Nous aurions pu alors obtenir un circuit d'une densité de l'ordre de 4000 transistors par  $\text{mm}^2$ , et le circuit mesurerait 5,15  $\text{mm}^2$ , ce qui aurait permis d'intégrer sur la même surface un réseau de 3\*3 cellules. Mais le but était de réaliser rapidement et avec un maximum de sûreté de conception, un circuit de recherche, afin de développer une maquette de démonstration pour valider notre concept d'architecture cellulaire.

##### **4.4.4.2. Temps**

Lors des simulations effectuées sous SOLO 1200, le circuit fonctionne avec une fréquence d'horloge de 10 MHz (85 nanosecondes pour la phase haute et 15 nanosecondes pour la phase basse). Le temps de traitement du pixel, pour des messages de type épandage ou projection, est au maximum de 3 microsecondes. Un tel

message est traité en une trentaine de microsecondes, suivant le nombre de pixels traversés par le rayon.

Le temps de transmission d'un message de type transparence est d'une microseconde, ce qui correspond aussi au temps nécessaire à la génération d'un message contenant une valeur de pixel (après réception d'un message de type sortie\_image), ou temps nécessaire à la cellule pour acquérir un message contenant la valeur d'un pixel (après réception d'un message de type entrée\_image). L'égalité de ces trois temps est essentiel pour assurer une excellente activité du réseau lors des phases de chargement et de sortie d'une image (voir chapitre 2, paragraphe 2.5.).

#### **4.4.4.3. Performance d'un réseau construit avec le circuit**

Nous allons présenter les performances d'un réseau cellulaire, dont la cellule de base est le circuit réalisé. Les résultats suivants découlent des simulations fonctionnelles HELIX portant sur l'architecture cellulaire (voir chapitre 3, paragraphe 3.3.), et des performances du circuit réalisé. Un réseau de  $N$  par  $N$  cellules traitant chacune d'elle  $k$  par  $k$  pixels est utilisé pour reconstruire des images de  $m$  par  $m$  pixels ( $m = k * N$ ).

La figure 4.26. présente plusieurs courbes donnant le nombre de pixels par cellule  $k$  pour différentes vitesses d'acquisition, en fonction du temps de traitement du pixel, ceci dans le cadre d'une utilisation optimale du réseau traitant une image de taille 1024 par 1024 pixels. La vitesse d'acquisition est caractérisé par  $T_{\text{point}}$ , qui est le temps nécessaire pour obtenir un point projeté.

Pour une reconstruction avec une vitesse d'envoi d'un message par microseconde et un temps de traitement du pixel de 3 microsecondes, la solution optimale donne  $k \approx 13$ .



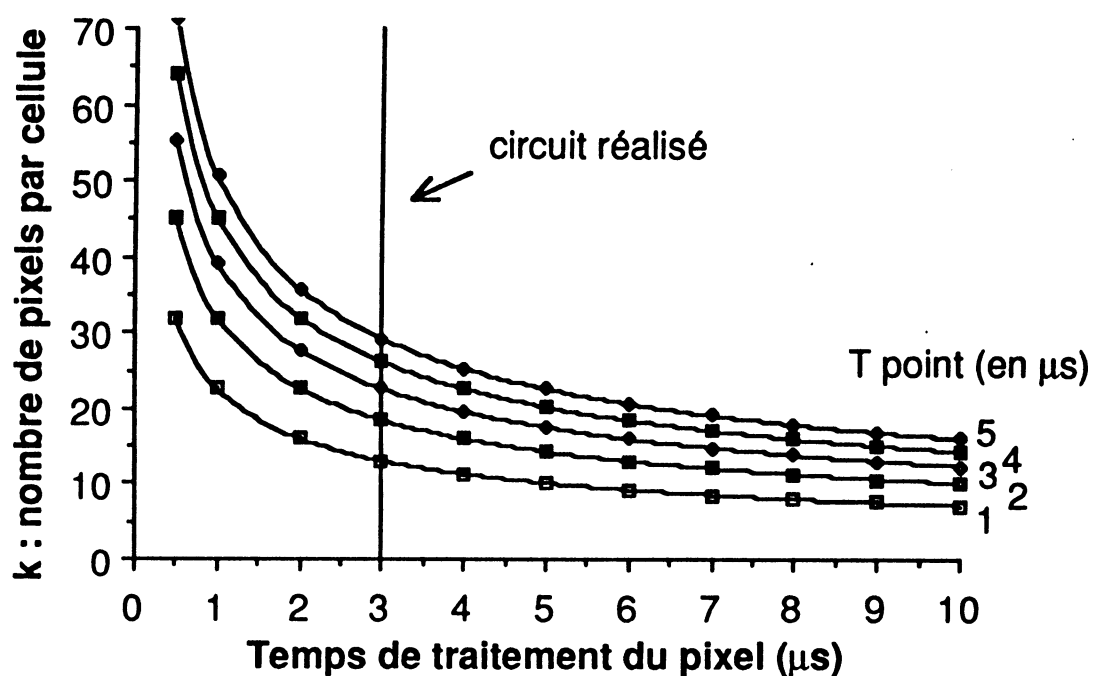


Figure 4.26. Taille de la cellule en fonction du temps de traitement

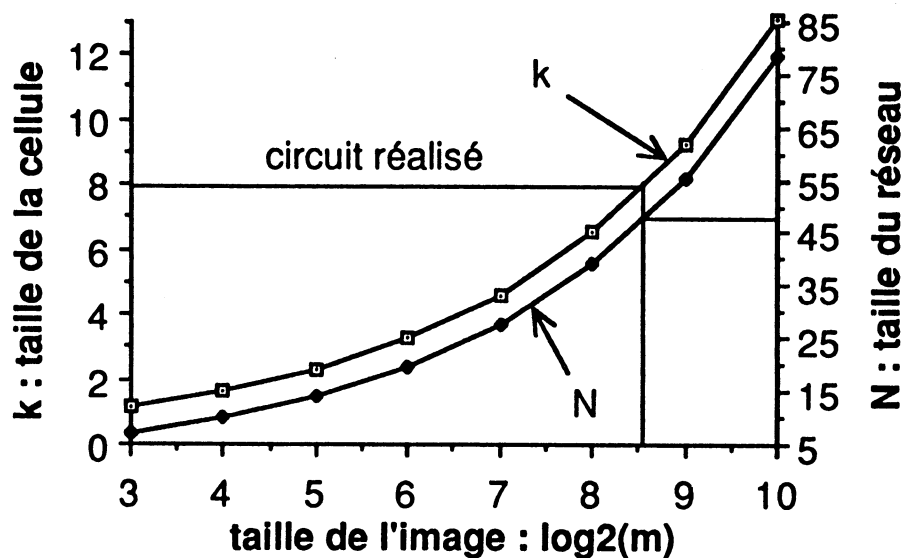


Figure 4.27. Taille de la cellule et du réseau en fonction de la taille de l'image

Les 2 courbes de la figure 4.27. présentent la taille du réseau  $N$  et la taille de la sous-image traitée par chaque cellule  $k$ , en fonction du

logarithme de la taille des images que nous souhaitons reconstruire ( $\log_2 m$ ), pour plusieurs temps de traitement du pixel et une vitesse d'envoi d'un message par microseconde. Le circuit réalisé traite 8 par 8 pixels ( $k=8$ ), et son utilisation est optimale pour reconstruire une image de taille 384 par 384 pixels ( $m=384$ ), grâce à un réseau de 48 par 48 cellules ( $N=48$ ).

La courbe de la figure 4.28. caractérise l'utilisation optimale du circuit réalisé.

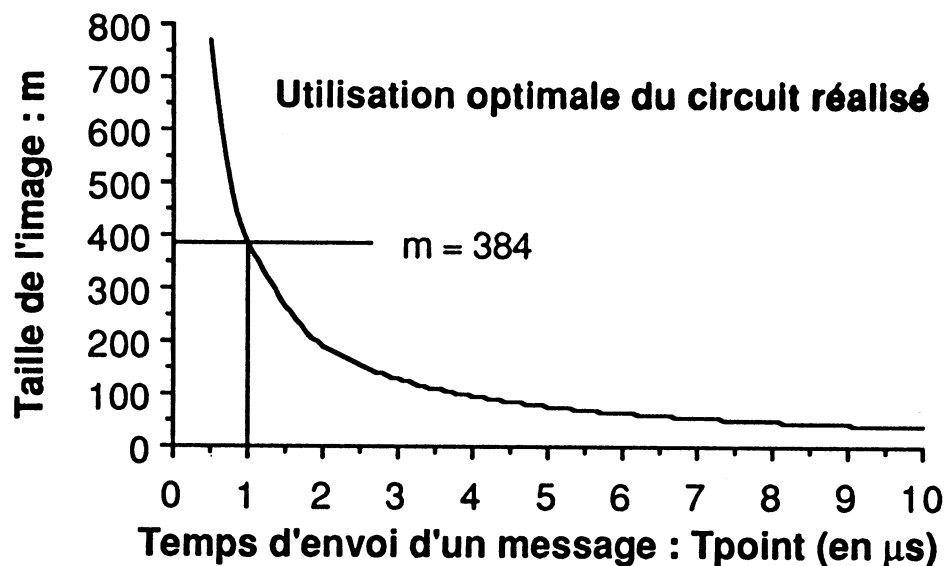


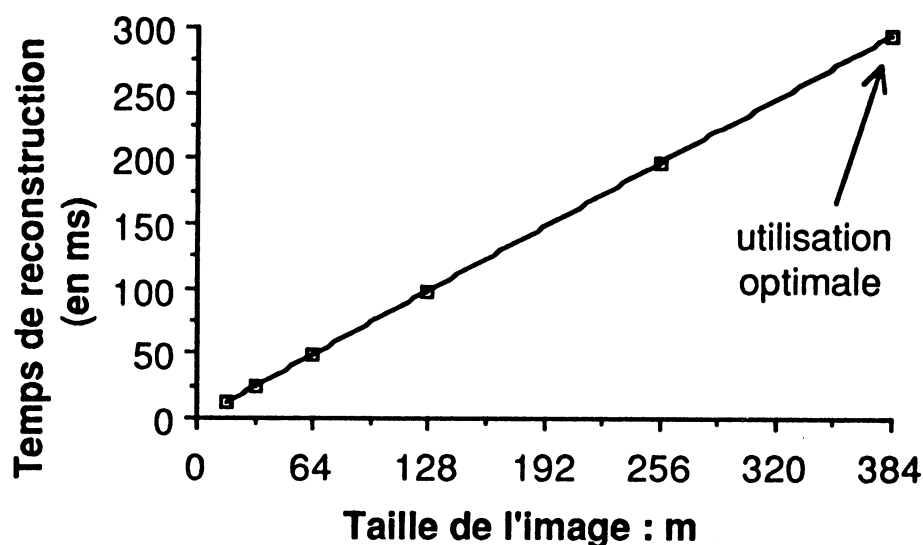
Figure 4.28. Taille de l'image en fonction de la vitesse d'acquisition

### Temps de reconstruction

Pour une vitesse d'acquisition d'un point par microseconde, l'utilisation du circuit est optimale pour des images de 384 par 384 pixels ( $m = 2 \cdot k^2 \cdot T_{pixel} = 384$ ). Alors le temps d'acquisition est de 147 millisecondes, et le temps de reconstruction (une phase d'épandage ou de projection), qui nécessite moins du double du temps nécessaire à l'acquisition est inférieur à 296 millisecondes.

Pour une image plus petite, le réseau est "surchargé" (l'activité est bonne, mais il n'arrive pas à traiter les messages en temps réel). Pour

une image 4 fois plus petite, le temps de reconstruction est divisé par 2 seulement (voir courbe de la figure 4.29.). Par exemple, pour un réseau 4\*4 traitant 32\*32 pixels, le temps de reconstruction est de 30 millisecondes. Si le circuit avait été fait spécialement pour une image de cette taille, une cellule aurait traité beaucoup moins de pixels pour permettre un traitement en temps réel, et le temps de reconstruction aurait été de 2 millisecondes.



**Figure 4.29.** Temps de reconstruction en fonction de la taille de l'image

### Temps de sortie de l'image

Après l'envoi d'un message de type `sortie_image` à une rangée de cellules, la sortie des messages contenant les valeurs des pixels s'effectue à la vitesse d'un message par microseconde.

Pour une image 1000\*1000, le temps de sortie est d'une seconde pour un temps de reconstruction de moins de 2 secondes (utilisation optimale). Pour une image 32\*32, le temps de sortie est d'une milliseconde pour un temps de reconstruction de 2 millisecondes (circuit optimal).

Le temps de sortie ne dépend pas des conditions d'utilisation du circuit lors des phases de reconstruction (peu utilisé, optimal, saturé).

La vitesse de sortie de l'image peut être beaucoup plus rapide, en sortant simultanément une ligne par rangée de cellules ; pour cela il faut que l'ordinateur hôte soit capable de récupérer les messages beaucoup plus vite.

### **4.4.5. Interfaçage du circuit dans l'environnement hôte**

#### **4.4.5.1 Principe de l'interface**

Comme nous l'avons vu au chapitre 2, paragraphe 2.4., le réseau cellulaire est piloté par un ordinateur hôte, qui a la charge de lui envoyer des messages via les tampons périphériques d'entrée, et de recevoir les messages sortants du réseau via les tampons périphériques de sortie.

Pour réaliser une machine de reconstruction d'images, il faut assembler les circuits en matrice carrée sur une carte, afin de l'installer sur un micro ou mini-ordinateur standard. Le microprocesseur de celui-ci assure la gestion des entrées et sorties du réseau grâce au bus données, qui permet le transfert de messages (16 bits), chacun des tampons périphériques étant adressable via le bus adresses, de la même manière qu'une mémoire.

#### **4.4.5.2 La carte**

La figure 4.30. illustre le principe de réalisation de la carte. La carte comprend un réseau de  $N$  par  $N$  cellules, et une partie décodage permettant d'accéder à tous les tampons périphériques (une adresse RAM par tampon).

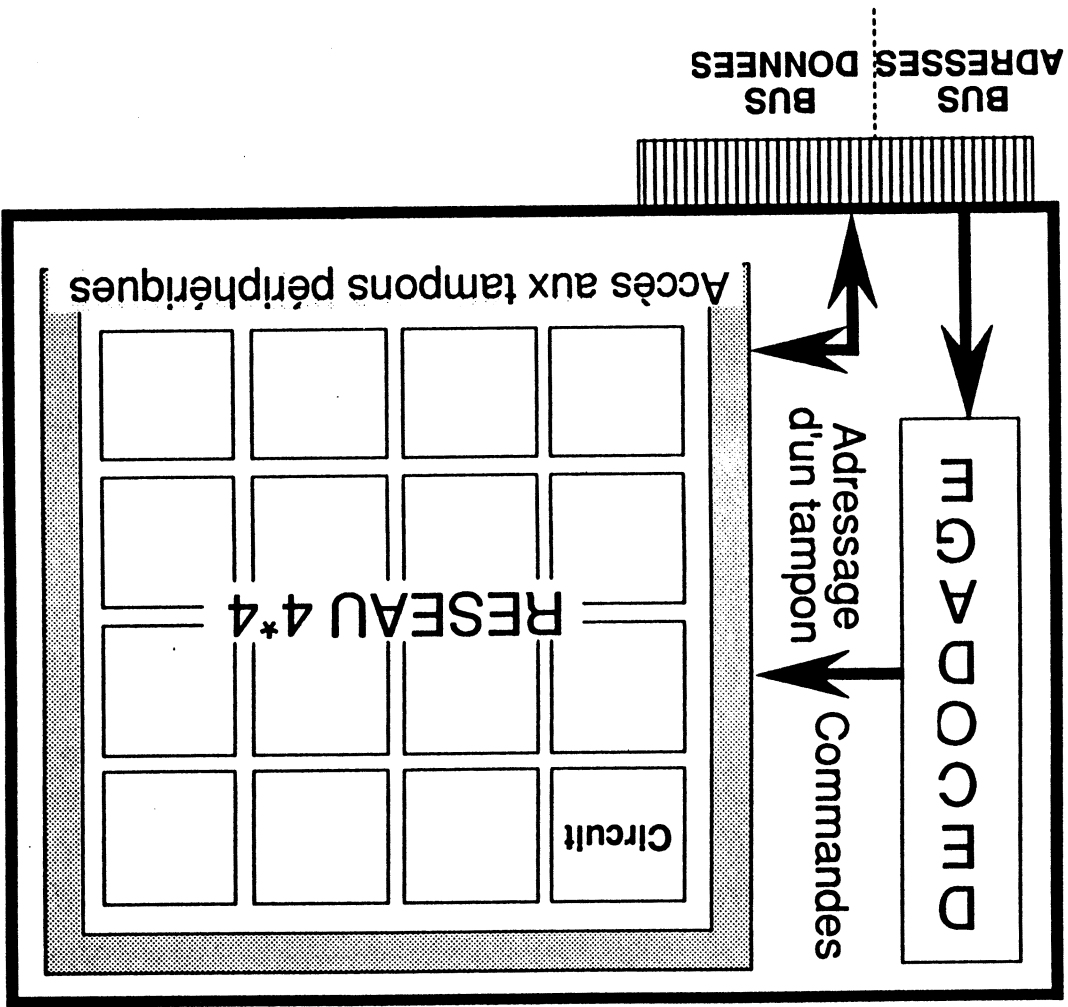


Figure 4.30. Carte d'interfaçage du réseau

Pour une machine à base du circuit réalisé (qui contient uniquement les tampons d'entrée de la cellule), il faut remplacer les tampons de sortie périphériques manquants par des composants discrets (par exemple 3 registres de 16 bits par tampon). Mais il est beaucoup plus astucieux d'utiliser des piles de type FIFO (First In First Out) permettant de stocker plusieurs messages, ainsi l'ordinateur peut recevoir les messages du réseau seulement en fin de traitement, ou à des moments bien précis du traitement.

La figure 4.31. présente la structure du tableau cellulaire, avec les tampons d'entrée périphériques, et les piles remplaçant les tampons

de sortie manquants. Les commandes proviennent de la partie décodage d'adresse.

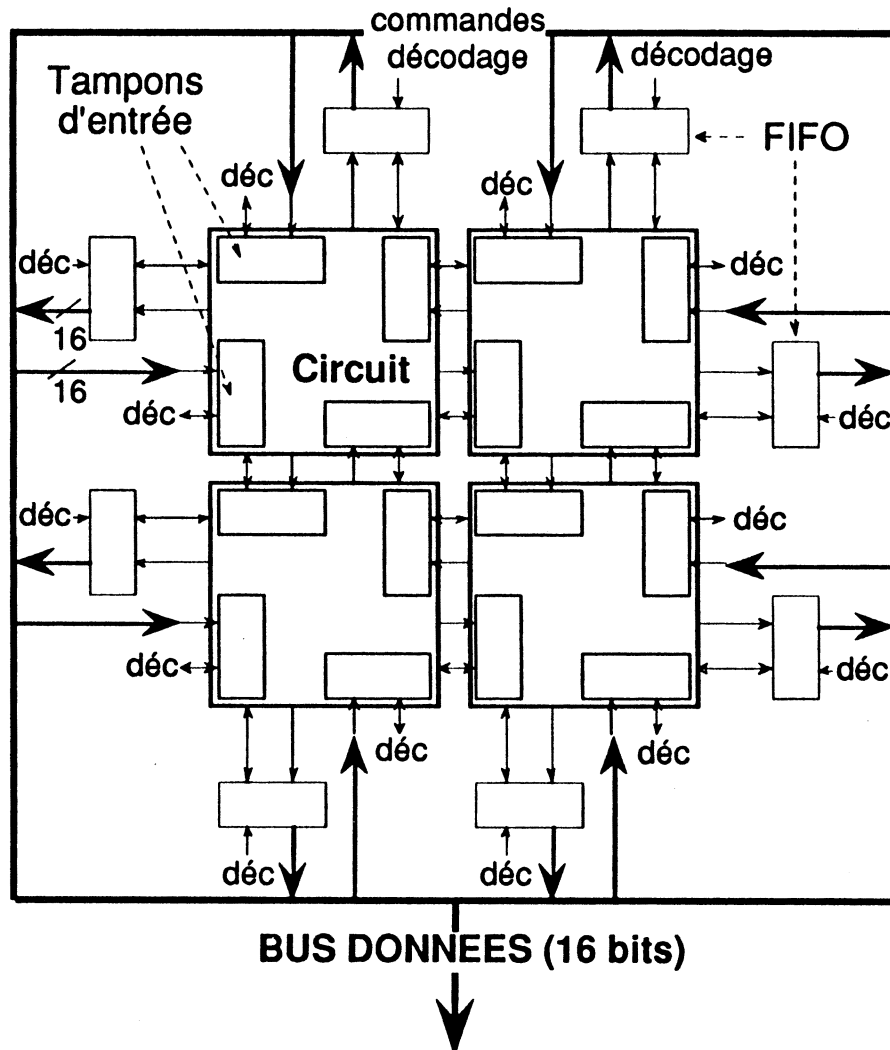


Figure 4.31. Le tableau cellulaire

Grâce au mécanisme de décodage, il est possible d'adresser tous les tampons périphériques et de générer les commandes d'écriture et de lecture pour les différents champs des messages, ainsi que les commandes de remplissage et de vidage des drapeaux (voir figure 3.32.). Le test des drapeaux est réalisé par une lecture ; de plus, si les drapeaux sont regroupés afin de former des mots de 16 bits, la vitesse de test est accrue.

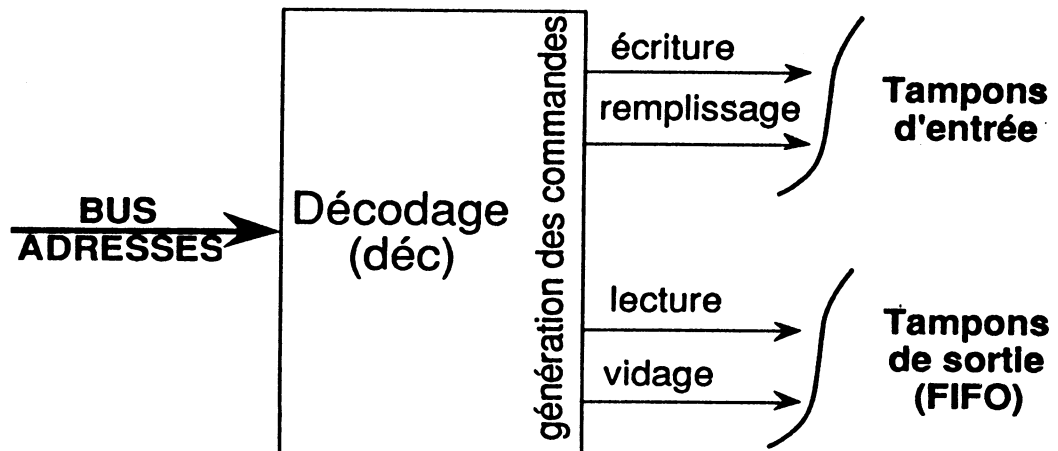


Figure 4.32. Décodage des adresses

La figure 4.33. décrit plus précisément les commandes de l'interface entre la périphérie du réseau et le bus de données.

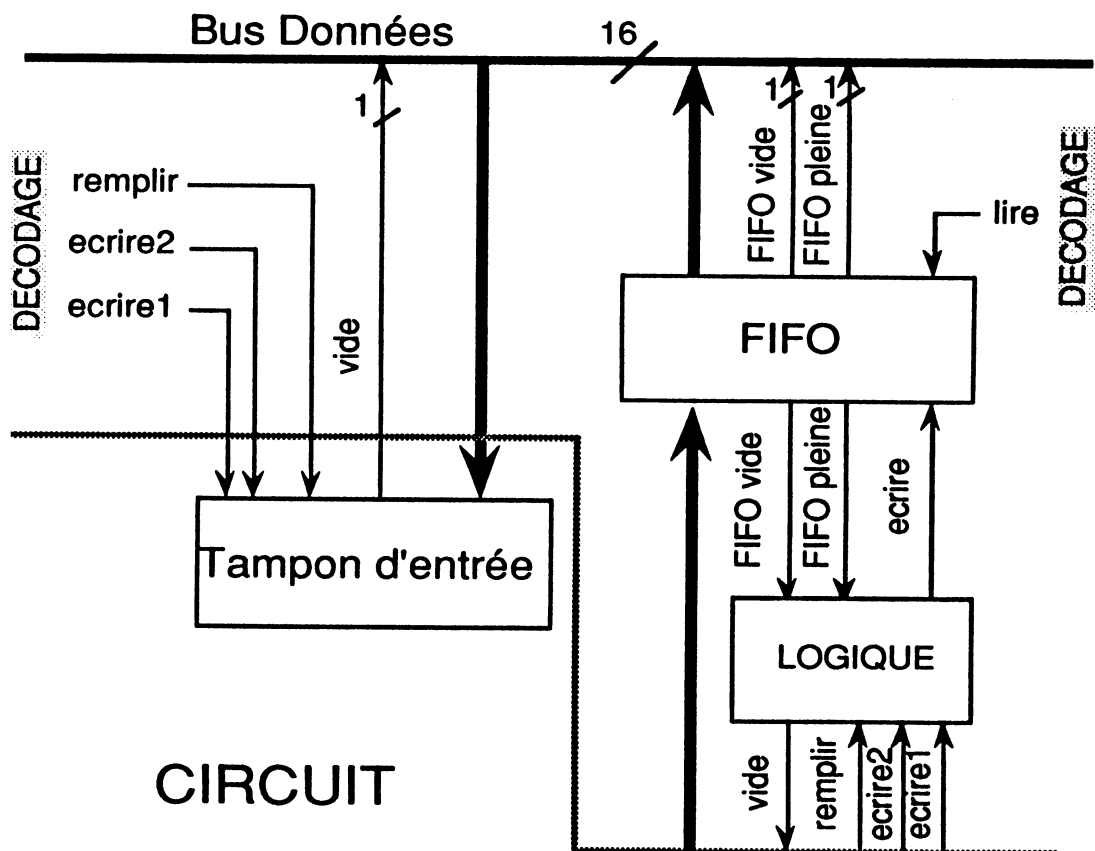


Figure 4.33. Détail des commandes d'interface

Pour l'opération d'épandage, l'ordinateur hôte n'a pas besoin de collecter les messages sortants du réseau, il suffit donc qu'il vide les piles FIFO de temps en temps au cours du traitement. Pour l'opération de projection, l'ordinateur hôte récupère les messages sortants du réseau contenus dans les FIFO, en lisant les mots de 16 bits, 3 par 3, pour reconstituer les messages. Pour la phase de sortie de l'image, l'ordinateur hôte n'est intéressé que par le troisième champ du message, qui est la valeur d'un pixel.

### **4.4.5.3. Vitesse de communication**

La plupart des microprocesseurs courants (Motorola 68030, Intel 80386) ont une fréquence de fonctionnement comprise entre 16 et 33 MHz (c'est à dire un cycle dure de 30 à 60 nanosecondes). Les instructions d'écriture et de lecture en mémoire prennent suivant les machines de 2 à 3 cycles d'horloge, et les temps d'accès sont classiquement compris entre 50 et 150 nanosecondes [IBM84]. Les communications par bus varient de 1 à 16 MHz.

De toutes ces informations générales sur les micro-ordinateurs, nous en déduisons que l'envoi d'un message (3 mots de 16 bits) à la carte demande environ entre 100 et 500 nanosecondes, suivant le type de machine hôte. Mais l'ordinateur hôte doit, avant d'envoyer les messages au réseau, calculer les points d'entrée des rayons (3 multiplications ou divisions par vecteur de projection, et une soustraction par point), et pré-traiter les informations de reconstruction (1 division). Compte tenu de cette charge de travail supplémentaire, la cadence d'émission des messages vers la carte se situe entre 0,5 et 1,5 microsecondes, c'est à dire une fréquence de l'ordre du MHz (1 million de messages par seconde). Cette fréquence correspond à la vitesse d'acquisition des projections par les machines actuelles, et le traitement est donc bien exécuté en temps réel.

Pour l'instant, la réalisation de la carte n'en est qu'à sa première étape, nous avons seulement montré sa faisabilité à l'aide de boîtiers standards.





## ***Conclusion***

---

## Conclusion

Le travail effectué lors de la préparation de cette thèse a concerné l'étude d'une architecture parallèle pour la reconstruction d'images. La réalisation d'un circuit intégré a été menée à bien en vue de développer une maquette.

### **L'architecture parallèle proposée**

Le réseau cellulaire que nous proposons est original par le fait du fonctionnement asynchrone des cellules, et du mécanisme de communication par acheminement de messages, permettant à des cellules éloignées de communiquer entre elles. Cette architecture générale est dédiée à des applications très particulières se prêtant bien au traitement parallèle, telles que la simulation logique de circuits intégrés [MAZ87], la simulation neuronique [FAU88b], le traitement du signal [PAY89] et la reconstruction d'images [LAT89]. La puissance du réseau cellulaire proposé réside dans la grande répétitivité de l'élément entité : la cellule. La technologie des circuits devenant de plus en plus efficace en densité d'intégration, les machines construites autour du réseau deviennent de plus en plus puissantes par le simple fait de l'augmentation du nombre de processeurs élémentaires. De plus, de par la structure du tableau cellulaire, nous pouvons facilement envisager de faire de l'intégration sur tranche (Wafer Scale Integration), en prenant cependant quelques précautions au niveau du routage [FAU86].

La validation de l'architecture a été faite grâce à l'application étudiée lors de ce travail : la reconstruction d'images, qui consiste à visualiser les distributions de densité d'une coupe d'un objet, en reconstruisant des images représentatives, à partir d'informations de projection obtenues par irradiation. Nous nous sommes intéressés à la tomographie par rayons X (avec un système d'acquisition à géométrie parallèle), et plus particulièrement aux méthodes de reconstruction par épandage des projections filtrées et aux techniques itératives. L'architecture cellulaire intervient pour réaliser les traitements d'épandage et de projection en temps réel (le temps de calcul est du même ordre de grandeur que le temps d'acquisition). Cette utilisation du réseau cellulaire illustre bien le genre d'algorithmes que nous

pouvons implémenter sur notre architecture. En effet les informations circulant dans le tableau ne sont pas régulières (les rayons traités à un instant donné ne sont pas tous parallèles).

### **Les performances obtenues**

Pour cette application de reconstruction, le réseau est très performant, puisqu'en moyenne lors d'une phase de traitement (épandage ou projection), l'activité des processeurs est d'au moins 60%. En fonction des contraintes temps réel dues à l'acquisition des projections, nous sommes capables de définir une structure cellulaire optimale quant à l'activité et au temps de reconstruction.

Le circuit précaractérisé réalisé dans une technologie CMOS 2 microns avec le logiciel SOLO 1200, intègre une cellule élémentaire traitant un tableau de 8 par 8 pixels. Le temps de traitement d'un pixel est inférieur à 3 microsecondes ; ce qui est comparable au temps nécessaire pour effectuer les mêmes opérations sur les microprocesseurs actuels développés dans des technologies plus performantes. La densité du circuit : 400 transistors par mm<sup>2</sup> (20000 transistors pour une surface de 50 mm<sup>2</sup>) est relativement faible, mais l'objectif de cette implantation était avant tout de réaliser rapidement un circuit de recherche avec un maximum de sûreté de conception, pour développer une machine de démonstration. Cependant, si notre circuit avait été conçu avec une méthode moins automatique et dans une technologie plus fine, nous aurions de meilleures performances de traitement (un temps de traitement du pixel de l'ordre de la microseconde), et une densité d'intégration d'environ 4000 transistors par mm<sup>2</sup>. Le gain en temps permettrait de traiter plus de pixels par cellule (environ 16 par 16), et ainsi pour la même surface, le circuit pourrait intégrer un réseau 3\*3 traitant une image de 48 par 48 pixels. Il suffirait alors d'une plaque d'une centaine de circuits pour construire une machine de reconstruction très performante (une image 512\*512 reconstruite en moins d'une seconde).

L'intérêt de ces développements ne réside pas bien sûr dans la puissance brute de l'opérateur réalisé, mais constitue une démonstration de faisabilité de l'architecture proposée.

### **L'avenir du projet**

A partir du circuit réalisé dont le test a mis en évidence son bon fonctionnement, nous allons pouvoir développer une carte supportant une matrice de processeurs, afin de construire une machine de reconstruction complète. La maquette pourra traiter seulement de petites images ( $32 \times 32$ , ou  $40 \times 40$ ), mais il sera toujours possible de reconstruire des images de grande taille, en les découpant, et en travaillant sur les différentes parties par logiciel.

Aujourd'hui, du fait de la durée nécessaire pour développer un circuit spécifique, notre projet s'oriente vers la définition d'un réseau programmable [RUB89]. Cette nouvelle machine générale permettra bien sûr de traiter toutes les applications étudiées, mais elle constituera, de plus, un excellent outil de simulation pour aborder d'autres problèmes. L'objectif est de concevoir une machine assez générale pour être utilisable dans beaucoup de domaines, mais en gardant d'excellentes performances pour pouvoir rivaliser avec les opérateurs spécifiques.



## ***Bibliographie***

---



## Bibliographie

- [ALE83] *"Array processors in medical imaging"*,  
P. Alexander,  
IEEE Computer, 1983, pp. 17-30.
  
- [AMA85] *"Imagerie médicale : méthodes de reconstruction et techniques instrumentales"*,  
J.L. Amans, R.E. Campagnolo, P. Garderet,  
Journée de Synthèse Imagerie Bio-Médicale, 5<sup>ième</sup> Congrès AFCET,  
Grenoble, 1985.
  
- [AMS88] *"Bases physiques et mathématiques du scanner aux rayons X"*,  
A. Amsalem,  
Bulletin de l'Association des Professeurs de Mathématiques de  
l'Enseignement Public, N°366, décembre 1988, pp. 563-578.
  
- [ANS86] *"OCCAM for functional simulation of highly parallel architectures"*,  
Y. Ansade, R. Cornu-Emieux, B. Faure,  
ESPRIT Project Meeting, Saint Pierre de Chartreuse, février 1986.
  
- [AXE83] *"Applications of computerized tomography to diagnostic radiology"*,  
L. Axel, P.H. Arger, R.A. Zimmerman,  
IEEE Proceedings, Vol. 71, N°3, 1983, pp. 293-297.
  
- [BAE80] *"Computer systems architecture, Chapter 3 : Arithmetic algorithms"*,  
J.L. Baer,  
Computer Science Press, 1980, pp. 76-131.
  
- [BAT83] *"Overview of computerized tomography with emphasis on future developements"*,  
R.H.T. Bates, K.L. Garden, T.M. Peters,  
IEEE Proceedings, Vol. 71, N°3, 1983, pp. 356-372.

## Bibliographie

- [BER85] *"Etude d'une machine cellulaire pour la simulation logique de circuits intégrés"*,  
J.P. Bernard,  
Thèse de Docteur Ingénieur, Grenoble, 1985.
- [BLO83] *"Application of computerized tomography to therapy and surgical planning"*,  
P. Bloch, J.K. Udupa,  
IEEE Proceedings, Vol. 71, N°3, 1983, pp. 351-355.
- [BOR88] *"A multi-microprocessor architecture for image reconstruction in CT"*,  
A.R. Borges, A.M.B. Ferrari de Almeida,  
Parallel Processing for Computer Vision and Display, Leeds, 1988,  
Edition Addison Welsey, 1989, pp. 489-496.
- [CEN83] *"Finite series-expansion reconstruction methods"*,  
Y. Censor,  
IEEE Proceedings, Vol. 71, N°3, 1983, pp. 409-419.
- [COR86] *"Some highly parallel algorithms processed by a regular network of asynchronous cells"*,  
R. Cornu-Emieux, Y. Ansade, B. Faure, G. Mazaré, P. Objois,  
Proceedings of the International Workshop on Parallel Algorithms Architectures, Luminy, North Holland 1986.
- [COR87a] *"Réseau de cellules asynchrones dédié à la simulation logique : conception et réalisation"*,  
R. Cornu-Emieux, D. Lattard, G. Mazaré, P. Objois,  
Journées Architectures de C3, Sophia Antipolos, juillet 1987,  
revue Bigre et Globule n°56, novembre 1987.

- [COR87b] *"An integrated highly parallel architecture to accelerate logical simulation"*,  
R. Cornu-Emieux, G. Mazaré, P. Objois,  
Proceedings of the IEEE International Symposium on Electronic  
Devices, Circuits and Systems (ISELDECS 87), Kharagpur, décembre  
1987.
- [COR88] *"Réseau de cellules intégré : Etude d'architectures pour des  
applications de CAO"*,  
R. Cornu-Emieux,  
Thèse de l'INPG, Grenoble, septembre 1988.
- [EGG81] *"Iterative algorithms for large partitioned linear systems, with  
applications to image reconstruction"*,  
P.P.B. Eggermont, G.T. Herman, A. Lent,  
Linear Algebra and its Applications, Vol. 40, 1981, pp. 37-67.
- [ES288] *"SOLO 1200 reference manual"*,  
European Silicon Structures limited, 1988.
- [FAU86] *"WSI asynchronous cell network"*,  
B. Faure, Y. Ansade, R. Cornu-Emieux, G. Mazaré,  
Proceedings of IFIP Workshop on WSI, Grenoble, mars 1986.
- [FAU88a] *"A VLSI asynchronous cellular architecture dedicated to multilayered  
neural networks"*,  
B. Faure, G. Mazaré,  
Proceedings of nEuro'88, Paris, juin 1988.
- [FAU88b] *"A VLSI implementation of multilayered neural networks"*,  
B. Faure, G. Mazaré,  
International Workshop on VLSI for Artificial Intelligence, Oxford, juillet  
1988.

## Bibliographie

- [FLY72] *"Some computer organizations and their effectiveness"*,  
M.J. Flynn,  
IEEE Transactions and Computers, Vol. C 21, N°9, 1972.
- [FRI87] *"Image reconstruction from multiple 1-D scans using filtered localized projection"*,  
B.R. Frieden, H.H.G. Aumann,  
Applied optics, Vol. 26, N°17, 1987, pp. 3615-3621.
- [GIL81] *"Rapid execution of fan beam image reconstruction algorithms using efficient computational techniques and special-purpose processors"*,  
B.K. Gilbert, S.K. Kenue, R.A. Robb, A. Chu, A.H. Lent, E.E. Swartzlander,  
IEEE Trans. on Biomed. Eng., Vol. BME-28, N°2, 1981, pp. 98-116.
- [GOR70] *"Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography"*,  
R. Gordon, R. Bender, G.T. Herman,  
Journal of Theoretical Biology, Vol. 29, 1970, pp. 471-481.
- [GOR74a] *"Three-dimensional reconstruction from projections : a review of algorithms"*,  
R. Gordon, G.T. Herman,  
Int. Rev. Cytol., 1974, pp. 111-151.
- [GOR74b] *"A tutorial on ART (Algebraic Reconstruction Techniques)"*,  
R. Gordon,  
IEEE Trans. Nucl. Sci., NS-21, 1974, pp. 78-93.
- [GOR75] *"Image reconstruction from projections"*,  
R. Gordon, G.T. Herman, S.A. Johnson,  
Sci. Amer., Vol. 233, N°4, 1975, pp. 56-68.

- [GRE83] *"Computerized tomography with ultrasound"*,  
J.F. Greenleaf,  
IEEE Proceedings, Vol. 71, N°3, 1983, pp. 330-337.
  
- [HAM87] *"Introduction aux méthodes analytiques de reconstruction d'image"*,  
M.A. Hamelin,  
Examen probatoire en Informatique du CNAM, Grenoble, 1987.
  
- [HER73a] *"ART : mathematics and applications, a report on the mathematical foundations and on the applicability to real data of the algebraic reconstruction techniques"*,  
G. T. Herman, A. Lent, S.W. Rowland,  
J. Theor. Biol., Vol. 42, 1973, pp. 1-32.
  
- [HER73b] *"Three methods for reconstructing objects from X-rays : a comparative study"*,  
G. T. Herman, S.W. Rowland,  
Computer Graphics and Image Proc., Vol. 2, 1973, pp. 151-178.
  
- [HER76] *"Iterative reconstruction algorithms"*,  
G. T. Herman, A. Lent,  
Comput. Biol. Med., Vol. 6, 1976, pp. 273-294.
  
- [HER79] *"Image reconstruction from projections : implementation and applications"*,  
Editeur : G.T. Herman,  
Topics in Applied Physics, Vol. 32, Springer-Verlag, 1979.
  
- [HER80a] *"Image reconstruction from projections : the fundamentals of computerized tomography"*,  
Editeur : G.T. Herman,  
Computer Science and Applied Mathematics, Academic Press, 1980.

## Bibliographie

- [HER80b] *"Mathematical aspects of computerized tomography"*,  
Editeurs : G.T. Herman, F. Natterer,  
Lecture Notes in Medical Informatics, Springer Verlag, 1980.
- [HIN83] *"An introduction to NMR imaging: from the Bloch equation to the imaging equation"*,  
W.S. Hinshaw, A.H. Lent,  
IEEE Proceedings, Vol. 71, N°3, 1983, pp. 338-350.
- [HOA78] *"Communicating sequential processes"*,  
C.A.R. Hoare,  
Communication ACM, Vol. 21, N°8, 1978, pp. 666-677.
- [HOR78] *"Density reconstruction using arbitrary ray-sampling schemes"*,  
B.K.P. Horn,  
IEEE Proceedings, Vol. 66, N°5, 1978, pp. 551-562.
- [HOR79] *"Fan-beam reconstruction methods"*,  
B.K.P. Horn,  
IEEE Proceedings, Vol. 67, N°12, 1979, pp. 1616-1623.
- [HOU73] *"Computerized transverse axial scanning"*,  
G.N. Hounsfield,  
Brit. Journ. of Radiology, Vol. 46, 1973, pp. 1016-1022.
- [IBM84] *"Technical Reference Personal Computer AT"*,  
IBM, 1984.
- [INM84] *"OCCAM programming manual"*,  
INMOS limited, Prentice-Hall, 1984.
- [INM86a] *"Transputer Development System v. 2.0, user manual, Programming interface, OCCAM implementation"*,  
INMOS limited, Prentice-Hall, 1986.

- [INM86b] *"Transputer, reference manual",*  
INMOS limited, Prentice-Hall, 1986.
  
- [JAM85] *"Display of 3D anisotropic images from limited-view computed tomograms",*  
K.A. Jaman, R. Gordon, R.M. Rangayyan,  
Computer Vision, Graphics, and Image Proc., Vol. 30, pp. 345-361.
  
- [KAR89] *"Etude et réalisation d'un mécanisme d'acheminement de messages dans un réseau cellulaire",*  
S.M. Karabernou,  
Rapport de DEA Microélectronique, INPG, Grenoble, 1989.
  
- [KIN86a] *"Image reconstruction using the Transputer",*  
N. Kingswood, E.L. Dagless, R.M. Belchamber, D. Betteridge, T. Lilley, J.D.M. Roberts,  
IEEE Proceedings, Vol. 133, Pt. E, N°3, 1986, pp. 139-144.
  
- [KIN86b] *"Image reconstruction using parallel computation",*  
N. Kingswood,  
Second International Conference on Image Processing and its Applications, Londres, 1986, pp. 97-101.
  
- [KUN82] *"Why systolic architectures?",*  
H.T. Kung,  
IEEE Computer, Vol. 15, N°1, 1982, pp. 37-46.
  
- [LAT88a] *"Parallel image reconstruction by using a dedicated asynchronous cellular array",*  
D. Lattard, G. Mazaré,  
Parallel Processing for Computer Vision and Display, Leeds, 1988, Edition Addison Welsey, 1989, pp. 479-488.



## Bibliographie

- [LAT88b] *"An integrated highly parallel architecture to do image reconstruction"*,  
D. Lattard, G. Mazaré,  
Nice, septembre 1988.
- [LAT88c] *"Une nouvelle architecture cellulaire pour la reconstruction d'images"*,  
D. Lattard, G. Mazaré,  
Actes de la conférence PIXIM88, Paris, octobre 1988, pp 193-207.
- [LAT89] *"Image reconstruction using an original asynchronous cellular array"*,  
D. Lattard, G. Mazaré,  
Proceedings of the IEEE International Symposium on Circuits and  
Systems 89, Portland, mai 1989, pp 13-16.
- [LEW83] *"Reconstruction algorithms : transform methods"*,  
R. M. Lewitt,  
IEEE Proceedings, Vol. 71, N°3, 1983, pp. 98-116.
- [MAZ87] *"A VLSI asynchronous cellular array to accelerate logic simulation"*,  
R. Cornu-Emieux, G. Mazaré, P. Objois,  
Proceedings of the 30<sup>th</sup> Midwest International Symposium on Circuits  
and Systems, Syracuse, août 1987.
- [MAZ88] *"An integrated asynchronous cellular array to do parallel image  
reconstruction"*,  
D. Lattard, G. Mazaré,  
Proceedings of the International Association for Pattern Recognition  
workshop on Computer Vision, Special Hardware and Industrial  
Applications, Tokyo, octobre 1988, pp. 128-131.
- [MUL86] *"Arithmétique et fonctions mathématiques de base des calculateurs"*,  
J.M. Muller,  
Polycopié de cours, ENSIMAG 3ème année, INPG, Grenoble, 1986.

- [OBJ87a] *"Simulation fonctionnelle d'une architecture parallèle en OCCAM"*,  
R. Cornu-Emieux, B. Faure, D. Lattard, G. Mazaré, P. Objois  
7th OCCAM User Group Meeting & International Workshop on Parallel  
Programming of Transputer Based Machines, Grenoble, septembre  
1987.
  
- [OBJ87b] *"Highly parallel logic simulation accelerators based upon distributed  
discrete-event simulation"*,  
Y. Ansade, R. Cornu-Emieux, G. Mazaré, P. Objois,  
Proceedings of the International Workshop on Hardware Accelerators,  
Oxford, octobre 1987.
  
- [OBJ88] *"Réseau de cellules intégré : Mécanisme de communication inter-  
cellulaire et application à la simulation logique"*,  
P. Objois,  
Thèse de l'INPG, Grenoble, septembre 1988.
  
- [OFF85] *"VLSI image processing"*,  
Editeur : R.J. Offen,  
Collins Professional and Technical Books, 1985.
  
- [PAY88] *"Etude de la réalisation d'architecture séquencée par les données  
sous forme d'un réseau cellulaire"*,  
E. Payan,  
Rapport de DEA Informatique, INPG, Grenoble, juin 1988.
  
- [PAY89] *"A programmable highly parallel architecture for Digital Signal  
Processing"*,  
G. Mazaré, E. Payan,  
Proceedings of the IEEE International Symposium on Circuits and  
Systems 89, Portland, mai 1989, pp 1332-1335.

## Bibliographie

- [PER82] *"Méthodes de reconstruction et de visualisation d'images tridimensionnelles en tomographie numérique par rayons X"*,  
F. Peyrin,  
Thèse de 3ème cycle, INSA, Lyon, 1982.
- [RAD17] *"Über die bestimmung von funktionen durch ihre integralwerte langs gewisser mannigfaltigkeiten"*,  
J. Radon,  
Berl. Verh. Sachs. Akad. Wiss. Leipzig, Math-Nature Kl. 69, 1917.
- [ROB82] *"X-ray computed tomography : an engineering synthesis of multiscientific principles"*,  
R. A. Robb,  
CRC Crit. Reviews in Biomed. Eng., Vol. 7, 1982, pp. 265-333.
- [RUB89] *"Etude d'une architecture parallèle générale"*,  
P. Rubini,  
Rapport de DEA Informatique, INPG, Grenoble, 1989.
- [SCU78] *"Introduction to computer aided tomography"*,  
H.J. Scudder,  
IEEE Proceedings, Vol. 66, N°6, 1978, pp. 628-637.
- [SIL86] *"SL2000/Helix : User Documentation"*,  
Silvar-Lisco, 1986.
- [SMI73] *"Image reconstruction from finite numbers of projections"*,  
P.H. Smith, T.M. Peters, R.H.T. Bates,  
J. Phys. A : Math., Nucl. Gén., Vol. 6, N°6, 1973, pp. 361-382.
- [SZU86] *"Computed tomography for optical computing"*,  
H. Szu,  
Proceedings SPIE - Optical and Hybrid Computing, Vol. 634, 1986, pp. 475-489.

- [TRE86] *"La tomodensitométrie"*,  
J. Trémolières,  
Electronique Applications, N°11, pp. 35-47.



## ***Annexe 1***

---

# Aspect mathématique de la reconstruction d'images

## **1. Les méthodes analytiques.....235**

### **1.1. Les transformées de Fourier..... 235**

### **1.2. Le théorème Coupe-Projection ..... 236**

### **1.3. Les méthodes ..... 236**

#### **1.3.1. Inversion de la transformée de Fourier 2D ..... 236**

#### **1.3.2. Epannage des projections filtrées (FBP) ..... 237**

#### **1.3.3. Déconvolution de l'image épannée..... 238**

## **2. Les méthodes algébriques.....238**

### **2.1. Les méthodes ART..... 239**

#### **2.1.1. La méthode additive ..... 239**

#### **2.1.2. La méthode multiplicative..... 240**

#### **2.1.3. Les autres versions des méthodes ART..... 240**

##### **2.1.3.1. Les méthodes avec des contraintes..... 240**

##### **2.1.3.2. Les méthodes pondérées par un facteur..... 241**

##### **2.1.3.3. Les méthodes avec tolérance à l'erreur..... 241**

### **2.2. Les méthodes SIRT..... 242**

#### **2.2.1. La méthode additive ..... 242**

#### **2.2.2. La méthode multiplicative..... 243**

## 1. LES METHODES ANALYTIQUES

Ces méthodes se rapportent à la formulation continue du problème, elles essaient de résoudre l'équation de Radon :

$$P(r,\vartheta) = \int_{-\infty}^{+\infty} A(x,y) dl, \quad [1]$$

où  $P(r,\vartheta)$  sont les valeurs des points projetés,  $A(x,y)$  la fonction densité, et  $dl$  la longueur différentielle du chemin d'un rayon.

Avant de détailler ces méthodes, il faut rappeler les notions de transformée de Fourier et énoncer le théorème Coupe-Projection qui en est la base [HAM87].

### 1.1. Les transformées de Fourier

Une fonction périodique  $f$  à une variable  $x$  peut se décomposer en somme de sinus et cosinus, dont les pulsations sont des multiples entiers de la pulsation de la fonction d'origine. La fonction  $F$  ainsi générée est appelée transformée de Fourier de  $f$ , elle s'exprime par :

$$F(\Omega) = \int_{-\infty}^{+\infty} f(x)e^{-i\Omega x} dx, \quad \text{où } \Omega = 2\pi k.$$

$\mathcal{F}_1$  est défini comme l'opérateur qui fait correspondre à la fonction  $f$  sa transformée de Fourier  $F$ .

De même, nous pouvons généraliser avec une fonction  $g$  à deux variables  $x$  et  $y$ . La transformée de Fourier bidimensionnelle de  $g$  est la fonction  $G$  telle que :

$$G(\Omega,\Delta) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} g(x,y)e^{-i(\Omega x + \Delta y)} dx dy, \quad \text{où } \Omega = 2\pi k \text{ et } \Delta = 2\pi h.$$

$\mathcal{F}_2$  est défini comme l'opérateur qui fait correspondre à la fonction  $g$  sa transformée de Fourier  $G$ .



## 1.2. Le théorème Coupe-Projection

Comme ont été définis les opérateurs  $\mathcal{F}_1$  et  $\mathcal{F}_2$ , nous pouvons définir l'opérateur de Radon  $\mathcal{R}$  qui associe à l'ensemble des projections  $P(r, \vartheta)$  la fonction densité  $A(x, y)$ , et  $\mathcal{C}$  l'opérateur de conversion des coordonnées cartésiennes en coordonnées polaires.

Alors le théorème Coupe-Projection s'exprime par la relation :

$$\mathcal{F}_1 \circ \mathcal{R} = \mathcal{C} \circ \mathcal{F}_2 \quad [2]$$

Ce qui signifie que la transformée de Fourier d'une projection d'angle  $\vartheta$  d'une image est égale à la coupe sous le même angle  $\vartheta$ , de la transformée de Fourier bidimensionnelle de l'image.

Ce théorème est à la base de beaucoup de méthodes analytiques de reconstruction.

## 1.3. Les méthodes

Résoudre le problème de la reconstruction d'images revient à trouver l'opérateur inverse de l'opérateur de Radon  $\mathcal{R}$ , c'est à dire  $\mathcal{R}^{-1}$ .

Un corollaire du théorème [2] est :

$$\mathcal{R}^{-1} = (\mathcal{F}_2)^{-1} \circ \mathcal{C}^{-1} \circ \mathcal{F}_1$$

### 1.3.1. Inversion de la transformée de Fourier 2D

Cette méthode découle directement du théorème précédent, elle est composée de trois étapes : il faut premièrement réaliser la transformée de Fourier de chacune des projections, passer deuxièmement en coordonnées cartésiennes, et enfin troisièmement calculer la transformée de Fourier bidimensionnelle inverse de la fonction obtenue.

Cette relation peut s'exprimer mathématiquement en considérant l'équation intégrale de la formulation continue du problème [1], et en prenant la transformée de Fourier des deux termes, on obtient ainsi :

$$P(\Omega, \varnothing) = \int_{-\infty}^{+\infty} P(r, \varnothing) e^{-i\Omega r} dr = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} A(x, y) e^{-i\Omega(x\cos\varnothing + y\sin\varnothing)} dx dy, \quad [3]$$

où  $\Omega = 2\pi k$ , et  $(x\cos\varnothing + y\sin\varnothing)$  a été substitué à  $r$  dans le terme de droite.

Cette méthode nécessite dans la seconde étape une interpolation numérique (pour le changement de coordonnées), et une transformation de Fourier bidimensionnelle inverse. Les traitements bidimensionnels demandent un nombre important de calculs et l'interpolation implique des approximations, ces points forment l'inconvénient de cette méthode.

### 1.3.2. Epandage des projections filtrées (FBP)

Avec une transformation de coordonnées cette fois de cartésiennes en polaires, et si nous intégrons en faisant varier l'angle  $\varnothing$  de 0 à 180° la relation précédente [1] peut s'exprimer ainsi :

$$A(x, y) = \frac{1}{2\pi^2} \int_0^\pi \int_{-\infty}^{+\infty} P(\Omega, \varnothing) e^{i\Omega(x\cos\varnothing + y\sin\varnothing)} |\Omega| d\Omega d\varnothing \quad [4]$$

Cette équation est une autre forme de l'équation de Radon. Cette méthode se décompose en deux phases : premièrement il faut filtrer l'ensemble des projections, puis deuxièmement épandre les projections filtrées.

Des difficultés de discrétisation interviennent dans l'opération de filtrage, car les valeurs des projections ne sont pas bornées, et donc la réalisation de ce filtrage nécessite une infinité de calculs (la fonction de filtre est  $|\Omega|$ , c'est une rampe continue infinie).

L'approche la plus facile et utilisée dans les systèmes actuels consiste à réaliser le filtrage directement dans le domaine spatial comme une simple opération de convolution. Ce filtrage des projections avec une certaine fonction annulera l'effet de flou de

l'épandage (effet vu dans les méthodes par sommation au chapitre 1, paragraphe 1.2.2.2.). Les différentes fonctions de filtre classiquement utilisées ont engendré diverses méthodes.

### 1.3.3. Déconvolution de l'image épandue

Cette méthode est une autre interprétation de l'équation [4]. Elle consiste à épandre les projections puis à filtrer l'image obtenue par un filtre  $\frac{1}{|\Omega|}$ . Ceci est l'équivalent des techniques par sommation où l'image est améliorée afin de supprimer l'effet de flou.

Le filtrage d'une image est une opération bidimensionnelle, chaque point de l'image a une influence sur tout autre point, la mise en œuvre de ce traitement est coûteux.

## 2. LES METHODES ALGEBRIQUES

Ces méthodes se rapportent à la formulation discrète du problème, elles essaient d'en résoudre le système linéaire :

$$P_k = \sum_{i=1}^{i=n} w_{ik} a_i \quad \text{pour } k = 1, \dots, m, \quad [5]$$

où  $a_i$  est le coefficient d'atténuation du pixel  $i$ ,  $P_k$  la valeur du point projeté correspondant au rayon  $k$ , et  $w_{ik}$  un facteur pondérateur qui caractérise la contribution du rayon  $k$  au pixel  $i$ .

Les différentes méthodes algébriques résolvent le système par approximations successives afin d'obtenir une solution proche de la solution réelle. Elles peuvent se regrouper en deux groupes : les techniques algébriques de reconstruction (ART), et les techniques de reconstruction itératives et simultanées (SIRT).

## 2.1. Les méthodes ART

Le principe des méthodes ART vient de l'algorithme de Kaczmarz, très connu en analyse numérique [EGG81].

Nous partons d'une image initiale puis grâce à un traitement itératif nous allons l'affiner ; ainsi de proche en proche, la solution calculée tend vers la solution réelle du système linéaire. Pour cela, à chaque itération, un seul angle de projection est pris en compte, et les valeurs des pixels sont modifiées, de manière à ce que les valeurs mesurées de cette projection correspondent aux valeurs calculées à partir de l'image précédant la mise à jour [CEN83], [GOR74a], [GOR75].

Le facteur de correction intervenant à chaque pas de l'algorithme détermine de type de méthode ART utilisée.

Il est prouvé mathématiquement que la solution converge vers la solution réelle du système. Le procédé est effectué un nombre entier de fois le nombre de projections ( $M$ ), et le traitement est stoppé lorsque les variations de l'image deviennent négligeables, ou que l'ensemble des valeurs des projections calculées, sous tous les angles possibles, se rapproche beaucoup de l'ensemble des valeurs des projections mesurées. Nous sommes alors sûr d'avoir une image proche de la réalité et que l'ensemble des pixels vérifie avec une erreur très faible le système linéaire caractérisant le problème.

Pratiquement, le nombre d'itérations nécessaires pour obtenir une image de "bonne" qualité est de l'ordre de  $(5 \cdot M)$ .

### 2.1.1. La méthode additive

En prenant la convention de noter  $P_{\varnothing}(k)$  les valeurs des points projetés mesurés de la projection d'angle  $\varnothing$ ,  $R_{\varnothing}(k)$  les valeurs des points projetés calculés sous le même angle  $\varnothing$  ( $k$  pouvant varier de 1 à  $K$ ),  $a_i$  la valeur du pixel  $i$  ( $i$  pouvant varier de 1 à  $n$ ),  $N_{\varnothing}(k)$  le nombre de pixels traversés par le point projeté de valeur  $R_{\varnothing}(k)$  et  $w_{ik}$  la contribution du rayon  $k$  au pixel  $i$ , la correction effectuée au cours d'une

itération (q+1) de cette méthode peut se décrire en fonction des données de l'itération précédente q par la relation :

$$a_i^{(q+1)} = a_i^q + \sum_{k=1}^{k=K} (w_{ik} \frac{P_{\theta}(k) - R_{\theta}^q(k)}{N_{\theta}(k)}) \quad \text{pour } i = 1, \dots, n \quad [6]$$

Le facteur de correction est ici la différence entre la valeur mesurée du point projeté et sa valeur calculée pondérée par le nombre de pixels traversés par le rayon correspondant (pour que la distribution soit uniforme à tous les pixels).

### 2.1.2. La méthode multiplicative

Dans cette méthode, le facteur correcteur est tout simplement le rapport entre la valeur mesurée du point projeté et sa valeur calculée. Alors la relation récurrente s'écrit :

$$a_i^{(q+1)} = a_i^q \sum_{k=1}^{k=K} (w_{ik} \frac{P_{\theta}(k)}{R_{\theta}^q(k)}) \quad \text{pour } i = 1, \dots, n \quad [7]$$

### 2.1.3. Les autres versions des méthodes ART

Il existe d'autres versions des méthodes ART qui sont en fait des dérivations des deux précédemment présentées.

#### 2.1.3.1. Les méthodes avec des contraintes

Il y a deux versions pratiquement identiques à celles exposées auparavant ([6] et [7]), à la contrainte près que lorsque la mise à jour d'un pixel lui affecte une valeur négative, celle-ci est substituée par la valeur nulle. La relation récurrente de la méthode additive est alors :

$$a_i^{(q+1)} = \text{MAX} \left\{ 0, a_i^q + \sum_{k=1}^{k=K} \left( w_{ik} \frac{P_{\theta}(k) - R_{\theta}^q(k)}{N_{\theta}(k)} \right) \right\} \quad \text{pour } i = 1, \dots, n$$

Il est possible de rajouter une autre contrainte telle que la mise à jour d'un pixel ne lui affecte pas une valeur supérieure à une valeur limite maximale  $A$  (cette constante correspondant à la densité maximale de l'objet). Cette méthode est intéressante lorsque nous connaissons l'objet, car il est possible de fixer  $A$  assez précisément. La relation récurrente de la méthode additive est alors :

$$a_i^{(q+1)} = \text{MIN} \left\{ A, \text{MAX} \left\{ 0, a_i^q + \sum_{k=1}^{k=K} \left( w_{ik} \frac{P_{\theta}(k) - R_{\theta}^q(k)}{N_{\theta}(k)} \right) \right\} \right\}$$

pour  $i = 1, \dots, n$

### 2.1.3.2. Les méthodes pondérées par un facteur

Dans ces méthodes la mise à jour est pondérée par un facteur réel  $\lambda$  ( $\lambda > 0$ ), la relation récurrente de la méthode additive s'écrit :

$$a_i^{(q+1)} = a_i^q + \lambda \sum_{k=1}^{k=K} \left( w_{ik} \frac{P_{\theta}(k) - R_{\theta}^q(k)}{N_{\theta}(k)} \right) \quad \text{pour } i = 1, \dots, n$$

La reconstruction d'un objet de fort contraste peut être plus rapide, en appliquant un facteur  $\lambda$  de l'ordre de 2 au lieu de 1.

### 2.1.3.3. Les méthodes avec tolérance à l'erreur

En outre, une version consiste à résoudre le système [5] en supposant que les équations sont bruitées. A chaque mesure  $P_k$  est associé  $\Delta_k$  qui représente son erreur maximale absolue, ce qui signifie que la mesure réelle est comprise dans l'intervalle borné par les

valeurs :  $P_k \pm \Delta_k$ . Le problème traité est la résolution du système d'inéquations défini par :

$$P_k - \Delta_k \leq \sum_{i=1}^{i=n} w_{ik} a_i \leq P_k + \Delta_k \quad \text{pour } k = 1, \dots, m,$$

Cette solution où il y a une certaine tolérance à l'erreur correspond bien à la réalité, car toutes les mesures  $P_k$  sont évidemment bruitées.

## 2.2. Les méthodes SIRT

Les méthodes SIRT sont également des méthodes itératives, qui en partant d'une image initiale aboutissent par itérations successives à une image "satisfaisante", dont l'ensemble des pixels vérifie le système linéaire [5] caractérisant le problème de la reconstruction d'une manière discrète.

Contrairement aux méthodes ART où la modification des pixels de l'image à chaque itération ne tient compte que d'une projection, les méthodes SIRT effectuent à chaque itération une mise à jour des pixels en tenant compte de l'ensemble des projections de l'objet, d'où leur nom de techniques "simultanées". Ainsi à chaque pas de ces techniques, tous les points projetés de toutes les projections sont calculés afin de les comparer aux valeurs mesurées, puis en fonction de ces différences les valeurs des pixels sont modifiées.

De même, il existe les méthodes : additive et multiplicative, définies, en fonction du facteur de correction.

Les critères d'arrêt des algorithmes sont identiques à ceux des techniques ART. Le nombre d'itérations nécessaire constaté de façon expérimentale est de l'ordre de 15.

### 2.2.1. La méthode additive

Le facteur de correction est ici proportionnel à la différence entre les valeurs calculées et mesurées.

En reprenant les mêmes conventions d'écriture, la relation récurrente définissant le traitement de l'itération d'ordre (q+1) en fonction de celle d'ordre q s'écrit :

$$a_i^{(q+1)} = a_i^q + \sum_{\theta=0}^{\theta=\pi} \sum_{k=1}^{k=K} (w_{ik} \frac{P_{\theta}(k) - R_{\theta}^q(k)}{N_{\theta}(k)}) \text{ pour } i = 1, \dots, n \quad [8]$$

### 2.2.2. La méthode multiplicative

Le facteur correcteur correspond au rapport entre les valeurs mesurées et les valeurs calculées, la relation entre deux itérations successives est :

$$a_i^{(q+1)} = a_i^q + \sum_{\theta=0}^{\theta=\pi} ( \sum_{k=1}^{k=K} (w_{ik} \frac{P_{\theta}(k)}{R_{\theta}^q(k)}) ) \text{ pour } i = 1, \dots, n \quad [9]$$

De même que pour les méthodes ART, il existe d'autres versions des méthodes SIRT ([8] et [9]) : des méthodes avec contraintes, pondérées et tolérantes à l'erreur.





## ***Annexe 2***

---

### **Systèmes d'acquisition et de reconstruction**

|                                                                                      |            |
|--------------------------------------------------------------------------------------|------------|
| <b>1. Introduction .....</b>                                                         | <b>247</b> |
| <b>2. Les différentes générations de machines....</b>                                | <b>247</b> |
| 2.1. La première génération.....                                                     | 249        |
| 2.2. La seconde génération.....                                                      | 249        |
| 2.3. Les autres générations.....                                                     | 249        |
| <b>3. L'unité d'acquisition.....</b>                                                 | <b>250</b> |
| <b>4. L'unité de calcul.....</b>                                                     | <b>250</b> |
| <b>5. L'affichage des résultats.....</b>                                             | <b>251</b> |
| <b>6. La géométrie des machines<br/>    et les techniques de reconstruction.....</b> | <b>251</b> |

## **1. INTRODUCTION**

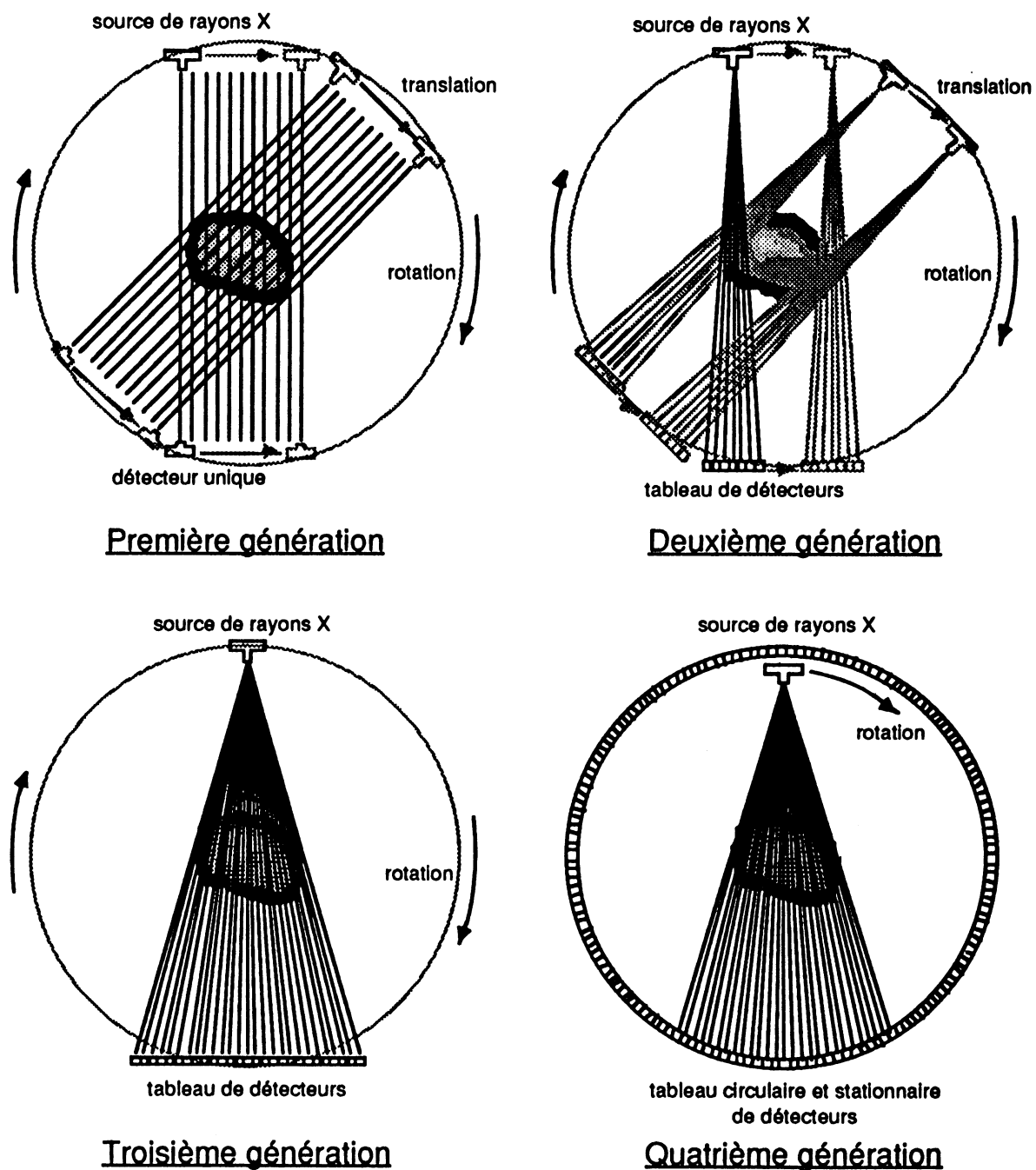
Nous donnons ici un aperçu des systèmes d'acquisition et de reconstruction d'images utilisées en tomographie par rayons X. De telles machines portent indifféremment les noms de tomographes, scanners ou tomodensitomètres.

La première machine a été mise au point en 1971, dans les laboratoires de l'Electrical Music Instruments (EMI) en Grande-Bretagne par G. N. Hounsfield. Ce prototype était utilisé uniquement pour reconstruire des coupes du cerveau [HOU73]. Le premier appareil de ce type fut mis en service en clinique en 1973. Et c'est en 1975 que la première machine capable d'explorer les corps entiers a été opérationnelle. Depuis lors, l'évolution a été extrêmement rapide à cause d'une part des progrès technologiques concernant les instruments de mesure (notamment les détecteurs), et d'autre part grâce à la fabuleuse amélioration des traitements des signaux par l'informatique moderne (avec l'apparition d'ordinateurs de plus en plus puissants). Tout ceci a fait qu'en une dizaine d'années, quatre générations de tomodensitomètres se sont succédées [TRE86].

Une machine tomographique comprend plusieurs unités : l'unité d'acquisition (pour l'obtention des projections), l'unité de calcul (pour la reconstruction de l'image à partir des projections) et un ensemble de périphériques : écran haute résolution (pour l'affichage) et disques (pour le stockage des images).

## **2. LES DIFFERENTES GENERATIONS DE MACHINES**

Les différences essentielles entre les quatre générations de machines sont liées à la manière, dont les projections sont générées, elles concernent la géométrie des parties mécaniques d'acquisition des machines [ROB82].



**Figure 1.8.** Les quatre générations de machines

Les diverses géométries se sont succédées en fonction des progrès réalisés dans les techniques d'instrumentation.

## **2.1. La première génération**

Le premier scanner développé par l'EMI est évidemment un appareil de cette génération, celle-ci est caractérisée par sa géométrie parallèle. Il y a une seule source et un seul détecteur, lesquels subissent un mouvement de translation pour acquérir une projection, puis grâce à une rotation de l'ensemble autour de l'objet, les autres mesures de projection sont obtenues.

Le temps nécessaire à l'acquisition sur  $180^\circ$  est de l'ordre de 5 minutes (pour le cas typique de 180 projections d'un millier de points chacune).

## **2.2. La seconde génération**

La deuxième génération reprend le principe de la première à la différence près que l'unique détecteur est remplacé par un tableau de détecteurs, ainsi la géométrie est en éventail. De même en conjuguant les mouvements de translation et de rotation, l'ensemble des mesures est obtenu, mais compte tenu que des mesures pour des angles différents de projection sont recueillies simultanément, le nombre de mouvements est considérablement diminué.

Ainsi, le temps d'acquisition d'une telle machine est de l'ordre de la minute (pour 10 rotations et un tableau de 30 détecteurs).

## **2.3. Les autres générations**

Les générations suivantes possèdent de pures géométries en éventail. La troisième génération utilise un tableau de détecteurs linéaire (voir figure 1.8.) ou circulaire d'une taille suffisante de manière à supprimer totalement les mouvements de translation. Pour la quatrième génération, le tableau est obligatoirement circulaire et il forme un cercle complet autour de l'objet. Ainsi, il n'y a plus besoin de le déplacer lors des mouvements de rotation, seule la source tourne.

Les rotations pour ces deux géométries peuvent être faites sur  $360^\circ$  au lieu de  $180^\circ$ , car les traces du faisceau en éventail après un

demi-tour ne coïncident pas automatiquement avec celles du demi-tour précédent.

Ces machines nécessitent un temps d'acquisition de quelques secondes seulement, plus ou moins de 5 secondes selon le nombre de mesures désiré.

### **3. L'UNITE D'ACQUISITION**

Les instruments principaux des tomographes sont les sources et les détecteurs de rayons X. C'est grâce à eux que l'acquisition des données est possible.

Les sources émettrices de rayons X des machines tomographiques sont des tubes à anode fixe pour la première génération et tournante pour les suivantes. Les faisceaux émis sont polychromatiques (une infinité de niveaux d'énergie et de longueurs d'onde). Grâce à une opération de collimation, les faisceaux sont orientés et divisés de manière très précise en fonction de la géométrie de la machine, afin qu'ils soient facilement collectés par les récepteurs.

Les détecteurs collectent les rayons après leur passage à travers le corps à étudier. Ils sont généralement constitués par une chambre à ionisation, où les ions formés sont recueillis sur des électrodes polarisées, ou par une chambre à scintillation avec photomultiplicateur, où un cristal absorbe l'énergie. L'information est récupérée sous forme de courant électronique. La précision de ces instruments est primordiale car elle détermine celle des mesures, et par conséquent elle conditionne la précision de l'image reconstruite.

### **4. L'UNITE DE CALCUL**

Les signaux électroniques provenant de l'unité d'acquisition sont digitalisés afin de subir le traitement informatique de la reconstruction.

Toutes les données manipulées lors d'une reconstruction sont en général codées sur 32 bits en virgule flottante ou fixe. Les pixels de l'image résultante nécessite un codage de 16 bits au moins en virgule fixe, ceci est la conséquence de l'échelle de densité des structures du corps humain (plusieurs milliers de niveaux différents).

Dans un tomographe standard, un mini-ordinateur gère l'instrumentation et réalise la reconstruction des images. En général, des processeurs vectoriels ou spécialisés lui sont associés, du fait du grand nombre d'opérations à réaliser. En effet, pour reconstruire une image  $256 \times 256$  à partir de 180 projections de 256 points chacune grâce à l'algorithme FBP, il faut environ 8 minutes en considérant des temps de calcul de l'unité centrale du mini-calculateur de l'ordre d'une microseconde pour une addition et de 4 microsecondes pour une multiplication. Tandis qu'avec un ensemble de processeurs spécialisés, le temps nécessaire pour la même reconstruction tombe à environ quelques secondes.

## **5. L'AFFICHAGE DES RESULTATS**

Les consoles d'affichage des tomographes modernes permettent de réaliser des opérations interactives sur les images reconstruites et des copies d'écran.

Les images sont affichées avec 512 niveaux de gris sur des écrans de  $512 \times 512$  ou  $1024 \times 1024$  pixels. Les pixels étant codés sur 16 bits, le système interactif permet d'explorer totalement l'image grâce à un système de choix de la valeur moyenne que nous voulons afficher et de réglage de l'échelle des niveaux de gris.

## **6. LA GEOMETRIE DES MACHINES ET LES TECHNIQUES DE RECONSTRUCTION**

Dans ce paragraphe, nous étudions l'influence de la géométrie sur les méthodes utilisées.



Il y a eu déjà quatre générations de machines, et nous pouvons bien imaginer des géométries possibles autres que celles exposées auparavant. Cependant toutes ces différentes solutions pour l'acquisition des projections peuvent se classer en deux grands groupes de géométrie : la géométrie parallèle (voir la première génération), et la géométrie éventail (voir les générations suivantes). L'évolution s'est faite vers des géométries en éventail car elles paraissent plus performantes du point de vue temps d'acquisition (elles demandent moins de déplacements de matériel). Cette constatation n'est pas forcément figée à tout jamais, car il est certainement possible de concevoir aujourd'hui des machines à géométrie parallèle aussi performantes.

Suivant le type de géométrie, les méthodes de reconstruction doivent être implémentées différemment, ceci du fait que l'ordre des mesures n'est pas le même et que les rayons dans le cas de la géométrie en éventail divergent au fur et à mesure qu'ils pénètrent dans l'objet. Ce travail était basé sur l'hypothèse que nous travaillions avec une architecture d'acquisition parallèle. Mais si nous voulons implémenter la méthode de reconstruction FBP sur une architecture d'acquisition en éventail, il est possible de le faire par deux voies différentes. La première solution consiste à réorganiser l'ensemble des mesures afin de les traiter comme pour une géométrie parallèle (cette opération s'appelle "re-binning"). La deuxième solution est de travailler directement avec les mesures provenant de l'acquisition en éventail, en incluant un pré-traitement sur les vecteurs de projections compensant l'effet produit par la divergence des rayons.





### ***Annexe 3***

---

## Les simulations du réseau

|                                                           |            |
|-----------------------------------------------------------|------------|
| <b>1. Les simulations fonctionnelles OCCAM.....</b>       | <b>257</b> |
| <b>Le processus CELLULE .....</b>                         | <b>257</b> |
| <br><b>2. Les simulations comportementales HELIX.....</b> | <b>261</b> |
| <b>2.1. Schémas de machine cellulaire simulée .....</b>   | <b>261</b> |
| <b>2.2. Modèle de la cellule.....</b>                     | <b>268</b> |

# 1. LES SIMULATIONS FONCTIONNELLES OCCAM

## Le processus CELLULE

```

PROC cellule (VAL INT nligne, ncolonne)
-- declarations du processus cellule
[4] CHAN to.sortie,from.sortie:      -- canaux vers tampons cote sortie
[4] INT adresse :                    -- adresses des tampons d'entree
BOOL arret,                          -- booleen pour l'arret
    flag :                          -- flag du tampon d'entree
BYTE car :                          -- caractere
INT ncell,                          -- numero cellule
    entree,                          -- indice de scrutation des buffers
    sortie,                          -- cote sortie du rayon
    num, type, info, s, tc, tg, z,   -- message d'entree
    nu, typ, inf, ss, tcs, zs,       -- message de sortie
    long.rayon,                      -- long du rayon dans la cellule
    entree.pixel,                    -- cote entree du rayon dans les pixels
    ad,ad.pixel,ad.fort,ad.faible :  -- calcul pixel dans cellule
[k*k]INT opacite :                  -- tableau resultats (valeurs pixels)
-- corps du processus cellule
SEQ
    -- initialisations
    adresse[0] := ((nligne * n) + ncolonne)      -- adresse du tampon Nord
    adresse[1] := ((nligne*(n+1)) + ncolonne)    -- adresse du tampon Ouest
    adresse[2] := (((nligne+1)*n) + ncolonne)    -- adresse du tampon Sud
    adresse[3] := (((nligne*(n+1))+ncolonne)+1)  -- adresse du tampon Est
    ncell := ((nligne * n) + ncolonne)           -- numero de la cellule
PAR
    -- lancement des tampons d'entree
    PAR i = 0 FOR 4
        buffer (adresse[i]+(i*nligne),to.sortie[i],from.sortie[i])
    -- traitement cellule
    SEQ
        -- initialisation
        arret := FALSE                        -- booleen pour l'arret du process
        entree := 0                          -- initialisation indice tampon entree
        SEQ j = 0 FOR (k * k)                -- initialisation tableau pixels
            opacite[j] := 0
        WHILE NOT arret
            SEQ
                PRI ALT
                    arret.cellule[ncell] ? car      -- arret du process cellule
                    arret := TRUE
                    reset.cellule[ncell] ? car      -- reset de la cellule
                    SEQ b = 0 FOR (k * k)
                        opacite[b] := 0
                TRUE&SKIP
                    -- traitement boucle cellule
                    SEQ
                        -- test drapeau tampon d'entree
                        to.sortie[entree] ! tester.flag
                        from.sortie[entree] ? flag
                    IF
                        flag = plein                -- tampon d'entree plein
                        SEQ

```

### Annexe 3 : Les simulations du réseau

```

-- lecture du message
to.sortie[entree] ! lire.message
from.sortie[entree] ? num;type;info;s;tc;tg;z
-- calcul adresse sortie
IF
  tc = 1
  -- tangente <= 1                                sortie Ouest ou Est
  SEQ
    zs      := (dim - 1) - z
    sortie := (entree + (s + 4)) \ 4
    tcs     := -tc
    ss      := -s
    entree.pixel := sortie
    s       := ss
    tc      := tcs
  tc = (-1)
  -- tangente > 1                                sortie Ouest ou Sud ou Est
  SEQ
    IF
      ((dim * tg) / fact.tg) >= z -- sortie Ouest ou Est
      SEQ
        zs      := ((dim * tg) / fact.tg) - z
        sortie := (entree + (s + 4)) \ 4
        tcs     := -tc
        ss      := -s
      ((dim * tg) / fact.tg) < z -- sortie Sud
      SEQ
        zs      := z - ((dim * tg) / fact.tg)
        sortie := (entree + 2) \ 4
        tcs     := tc
        ss      := s
        entree.pixel := entree
-- test drapeau tampon destinataire
to.entree[(((sortie+2)\4)*ligne)+adresse[sortie]] !
                                                    tester.flag
from.entree[(((sortie+2)\4)*ligne)+adresse[sortie]] ?
                                                    flag
IF
  flag = (NOT plein) -- tampon destinataire vide
  SEQ
    -- vider le tampon d'entree
    to.sortie[entree] ! vider.buffer
    -- traitement message
    IF
      type = transparence
      SKIP
    TRUE
    SEQ
      -- initialisation ad.fort, ad.faible
      IF
        s = 1
        ad := z / pdim
      TRUE
        ad := (k - 1) - (z / pdim)
      IF
        entree.pixel = 0 -- entree = Nord
        SEQ
          ad.fort := 0
          ad.faible := ad

```

### Annexe 3 : Les simulations du réseau

```

entree.pixel = 1                      -- entree = Ouest
SEQ
  ad.fort := (k - 1) - ad
  ad.faible := 0
entree.pixel = 2                      -- entree = Sud
SEQ
  ad.fort := (k - 1)
  ad.faible := (k - 1) - ad
entree.pixel = 3                      -- entree = Est
SEQ
  ad.fort := ad
  ad.faible := (k - 1)
TRUE
SKIP
z := z \ pdim
WHILE ((ad.fort>=0) AND (ad.faible>=0))
AND ((ad.fort<=(k-1)) AND (ad.faible<=(k-1)))
SEQ
  -- traitement d'un pixel
  IF
    tc = 1
    -- tangente <= 1      sortie Ouest ou Est
    SEQ
      long.rayon := (z * fact.z) / tg
      entree.pixel := (entree.pixel+(s+2))\4
      z := (pdim - 1) - z
      s := -s
      tc := -tc
    tc = (-1)
    -- tangente>1,      sortie Ouest ou Sud ou Est
    SEQ
      IF
        ((pdim*tg)/fact.tg)>=z -- Ouest ou Est
        SEQ
          long.rayon := (z * fact.z) / tg
          entree.pixel := (entree.pixel +
                          (s+2)) \4
          z := ((pdim * tg) / fact.tg) - z
          s := -s
          tc := -tc
        ((pdim*tg)/fact.tg) < z -- sortie Sud
        SEQ
          long.rayon := fact.z
          z := z - ((pdim * tg) / fact.tg)
      -- traitement suivant le type du message
      ad.pixel := (ad.fort * k) + ad.faible
    IF
      type = epandage
      opacite[ad.pixel] := opacite[ad.pixel] +
                          ((long.rayon * info) / fact.z)
      type = projection
      info := info + ((long.rayon *
                      opacite[ad.pixel]) / fact.z)
      type = sortie.image
      -- traitement sortie image
      SEQ
        inf := opacite[ad.pixel]
        typ := transparence
        -- test flag
        flag := plein
        WHILE flag = plein

```



```

SEQ
  to.entree[(((entree+2)\4)*ligne) +
    adresse[entree]] ! tester.flag
  from.entree[(((entree+2)\4)*ligne) +
    adresse[entree]] ? flag
-- passage du message valeur pixel
to.entree[(((entree+2)\4) * ligne) +
  adresse[entree]] ! donnees
to.entree[(((entree+2)\4)*ligne) +
  adresse[entree]]!num;typ;inf;s;tc;tg;z
-- remplir le buffer de sortie
to.entree[(((entree+2)\4)*ligne) +
  adresse[entree]] ! remplir.buffer
type = entree.image
-- traitement chargement image
SEQ
  -- vider le buffer d'entree
to.sortie[entree] ! vider.buffer
-- test flag buffer d'entree
flag := (NOT plein)
WHILE flag = (NOT plein)
  SEQ
    to.sortie[entree] ! tester.flag
    from.sortie[entree] ? flag
  -- lecture du message
to.sortie[entree] ! lire.message
from.sortie[entree] ? nu;typ;inf;s;tc;
  tg;z
  opacite[ad.pixel] := inf
TRUE
  SKIP
-- maj ad.fort, ad.faible
IF
  entree.pixel = 0          -- entree = Nord
    ad.fort := ad.fort + 1
  entree.pixel = 1          -- entree = Ouest
    ad.faible := ad.faible + 1
  entree.pixel = 2          -- entree = Sud
    ad.fort := ad.fort - 1
  entree.pixel = 3          -- entree = Est
    ad.faible := ad.faible - 1
TRUE
  SKIP
-- passage du message
to.entree[(((sortie+2)\4)*ligne)+adresse[sortie]]
  ! donnees
to.entree[(((sortie+2)\4)*ligne)+adresse[sortie]]
  ! num;type;info;ss;tcs;tg;zs
to.entree[(((sortie+2)\4)*ligne)+adresse[sortie]]
  ! remplir.buffer
TRUE
  SKIP
TRUE
  SKIP
  entree :=(entree+1)\4 -- incrementation indice tampon entree
-- arret des tampons d'entree
PAR j = 0 FOR 4
  to.sortie[j] ! arreter.buffer
:

```

## **2. LES SIMULATIONS COMPORTEMENTALES HELIX**

### **2.1. Schémas de machine cellulaire simulée**

Le schéma 1 présente la machine cellulaire complète, elle comprend trois parties :

- ⇒ le réseau (8\*8 cellules),
- ⇒ le maître qui simule le fonctionnement de l'ordinateur hôte,
- ⇒ et une partie contrôle du réseau, qui analyse son activité en permettant de connaître l'état des cellules : inactive ou active, et dans le second cas le message en cours de traitement. On peut ainsi avoir à tout instant la position des messages et leur cheminement.

Le schéma 2 montre la structure du réseau. Il inclut 8\*8 cellules, un mécanisme permettant un accès facile aux tampons périphériques à base de multiplexeurs et démultiplexeurs.

Le schéma 3 détaille la cellule de base du réseau et ses 8 tampons d'entrée et sortie.

Le schéma 4 illustre le maître. Il est composé d'un émetteur calculant et envoyant les messages au réseau, d'un récepteur recevant les messages sortants du réseau et un compteur permettant de connaître à tout moment l'occupation du réseau.

Le schéma 5 montre le résultat d'une simulation. Les chronogrammes visualisés caractérisent l'activité du réseau (nombre de messages émis, de messages reçus, de messages dans le réseau et de messages en cours de traitement).

Le schéma 6 montre l'évolution de l'état de quelques cellules. Elles ont été affectées d'un numéro de 1 à 64 pour les identifier, et de même pour les messages. On connaît donc tout ce qui se passe au niveau de chaque cellule.

Schéma 1. Machine cellulaire

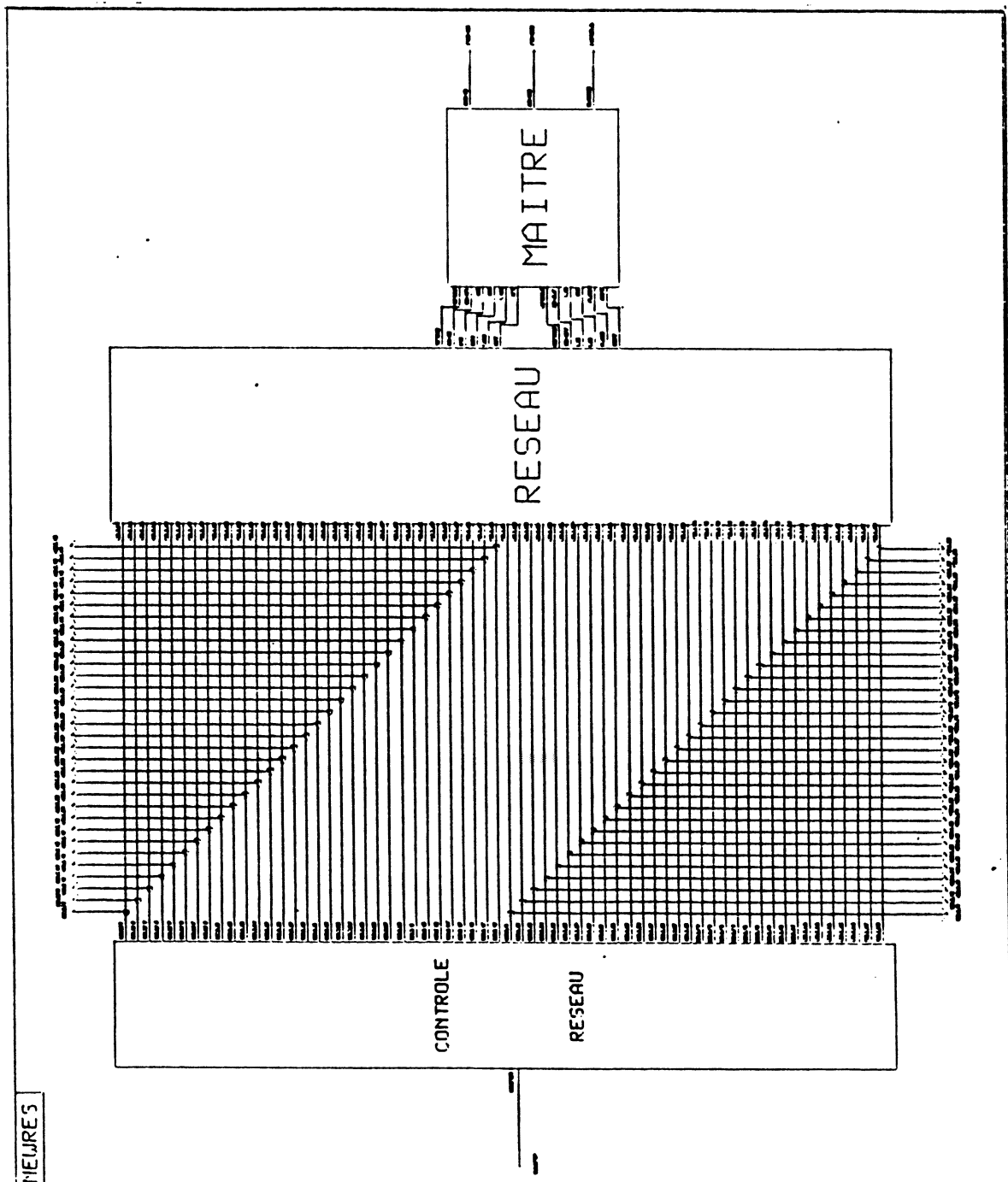


Schéma 2. Structure du réseau 8\*8

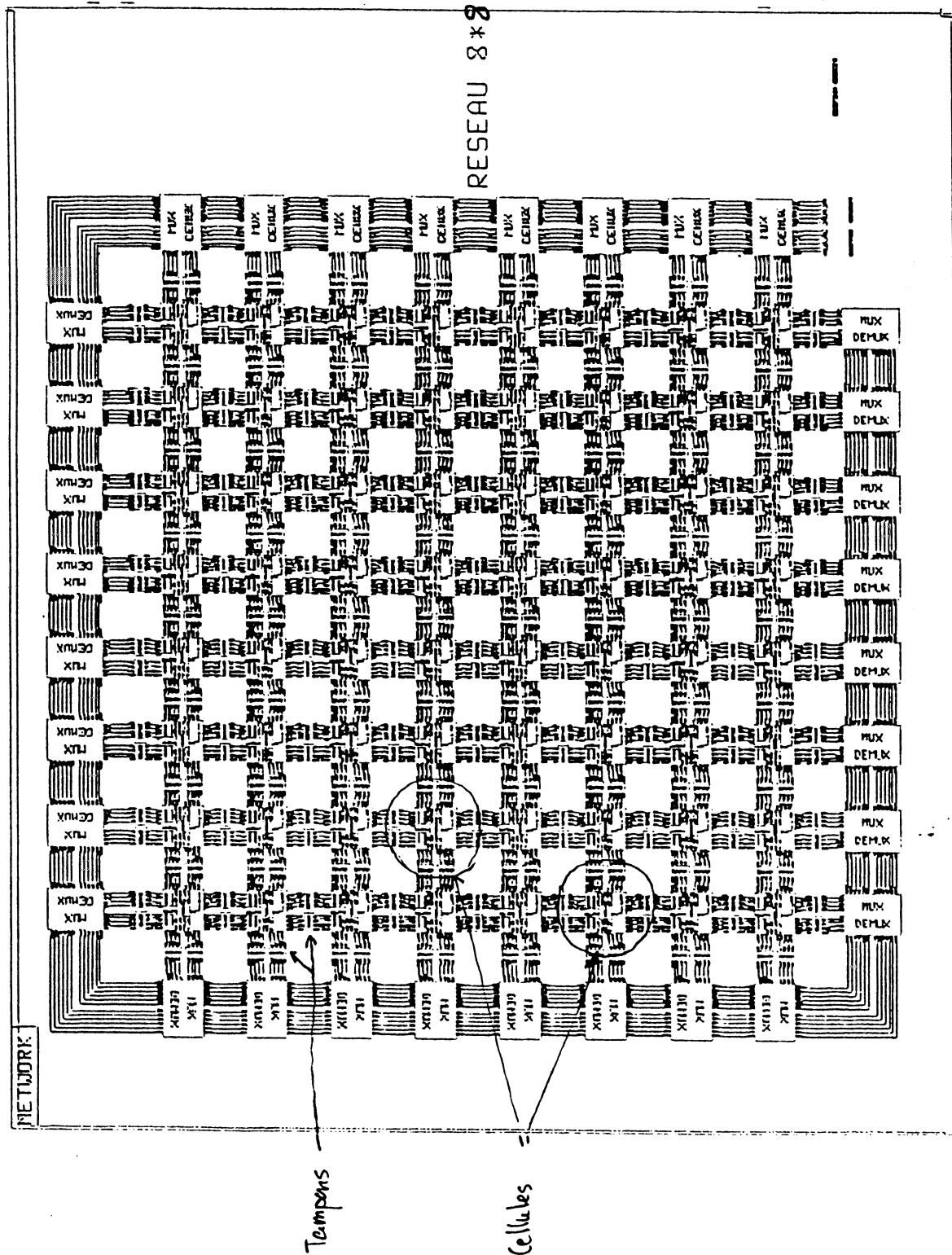


Schéma 3. Cellule de base

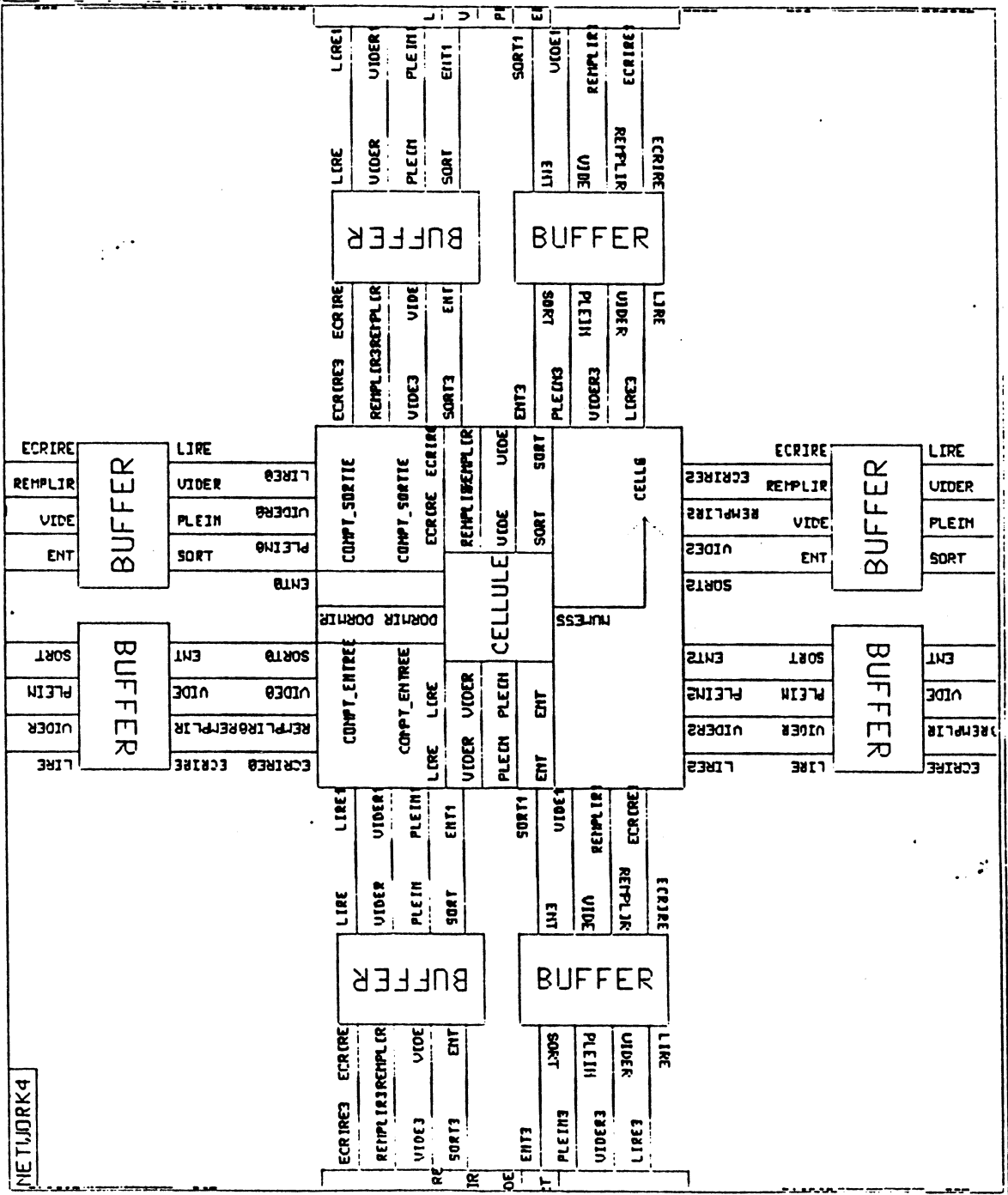


Schéma 4. Processus maître

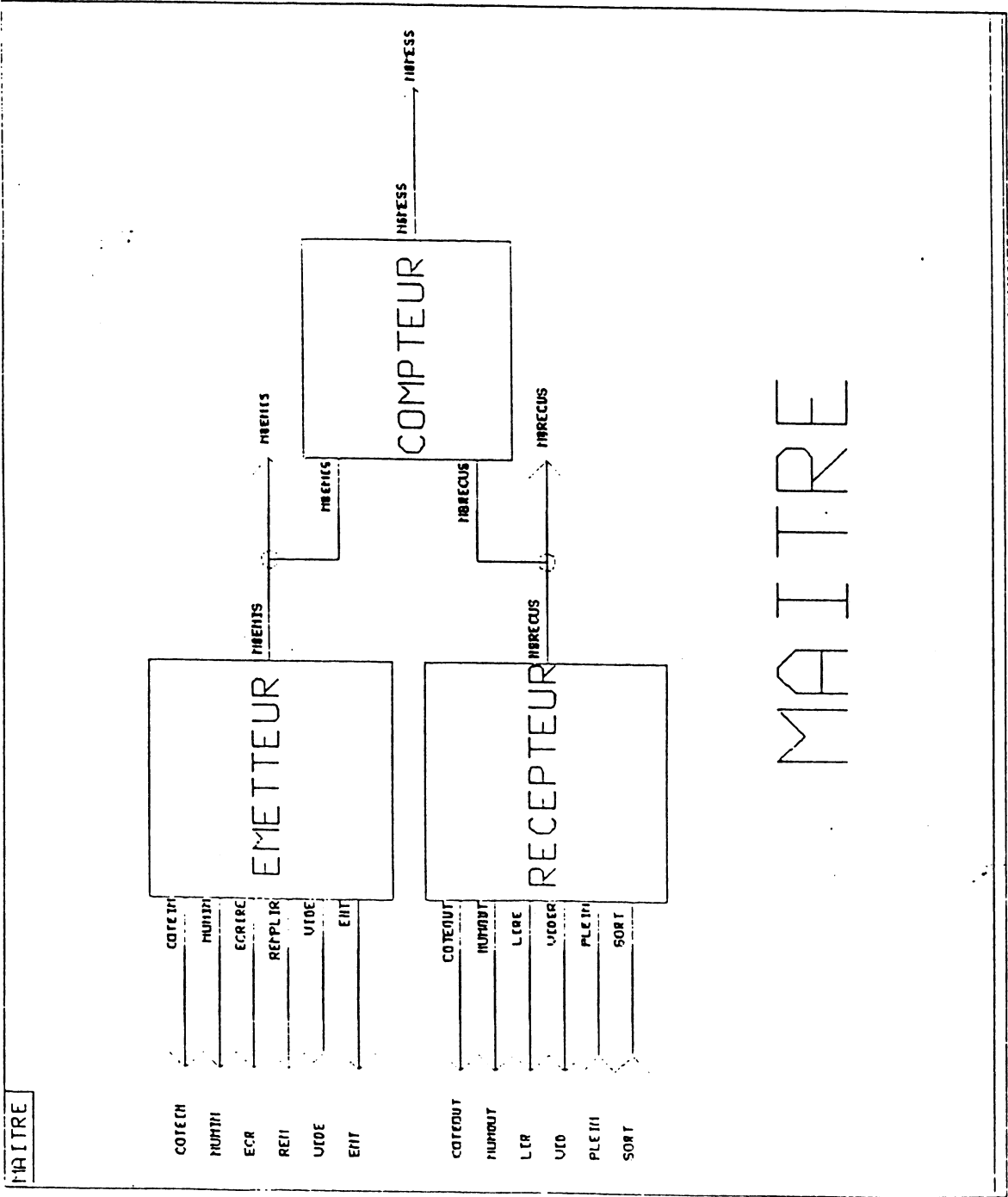
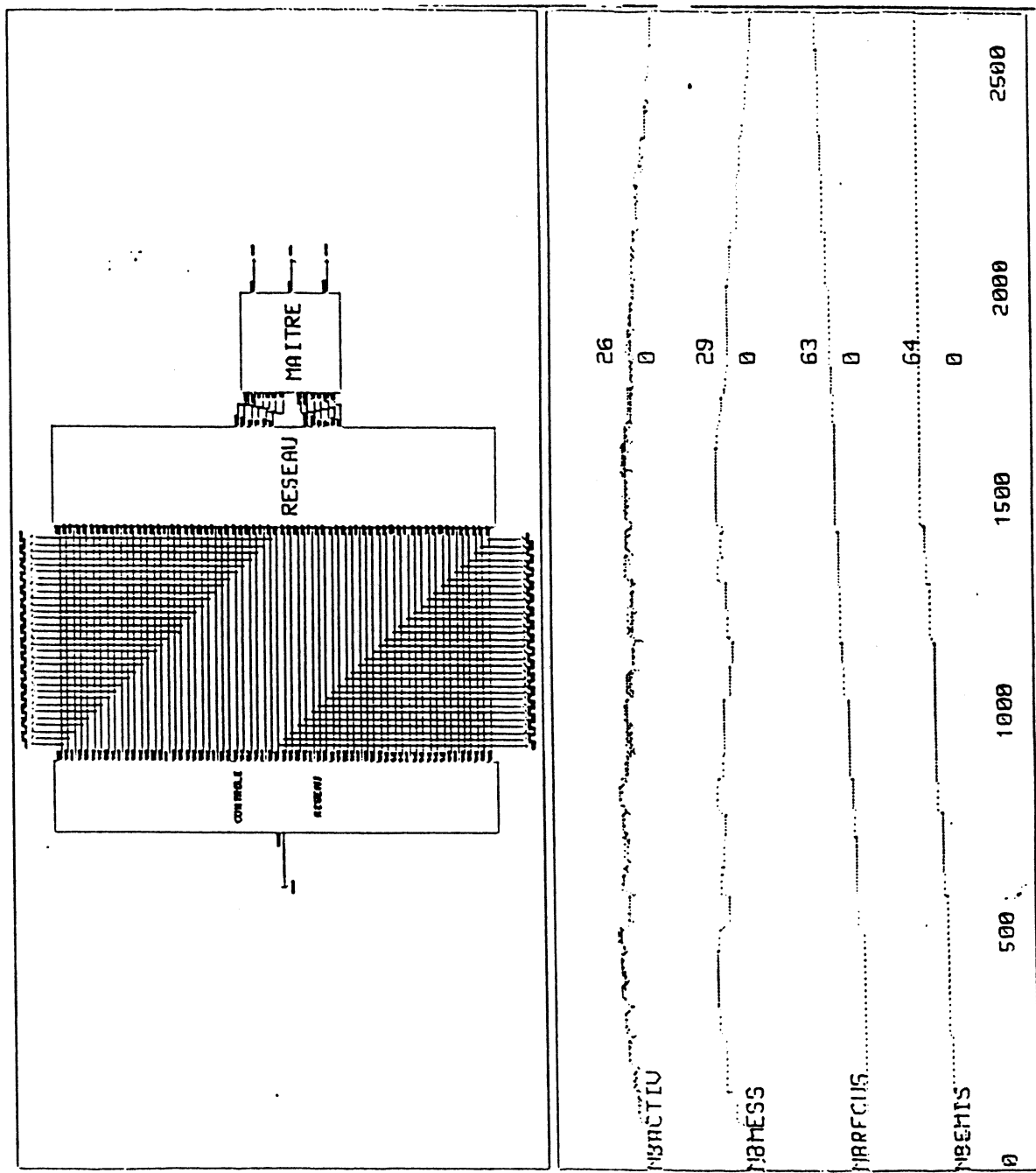


Schéma 5. Résultat : activité du réseau



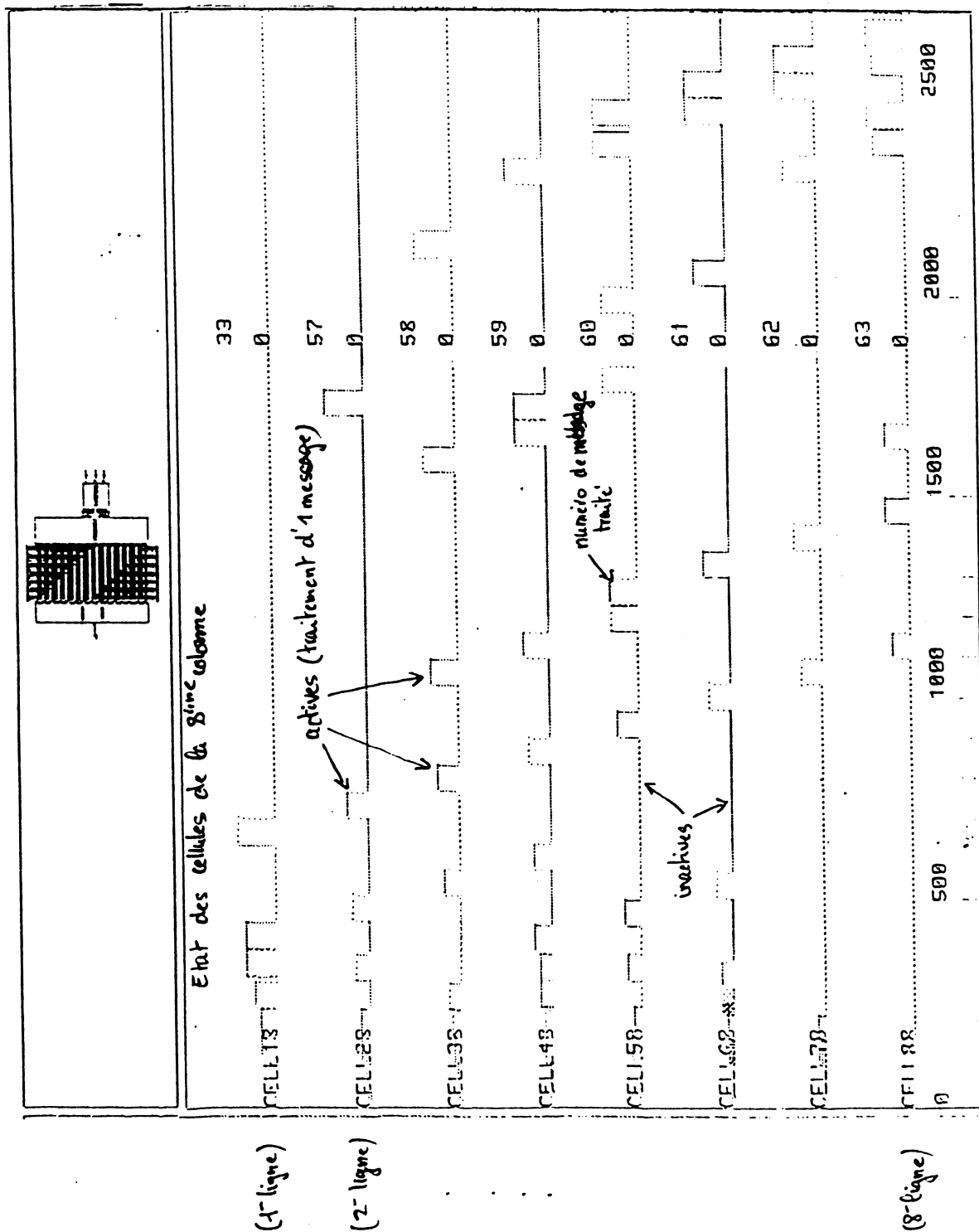
Nombre de  
cellules actives

Nombre de  
messages dans  
le réseau

Nombre de  
messages sortis  
du réseau

Nombre de  
messages envoyés  
au réseau

Schéma 6. Résultat : état de quelques cellules





## 2.2. Modèle de la cellule

```
COMPTYPE cellule;

INWARD dormir : boolnet;
INWARD plein : boolnet;
INWARD vide : boolnet;
INWARD ent : messnet;

OUTWARD compt_entree, compt_sortie : intnet;
OUTWARD lire, ecrire, remplir, vider : boolnet;
OUTWARD sort : messnet;
OUTWARD numess : intnet;

VAR message, mess_sortie : mess;
    esp : INTEGER;
    compt_ent, compt_sort : INTEGER;
    tg, tamp : INTEGER;
    flag : BOOLEAN;
    long : INTEGER;
    compt_mess, long_totale, compt_pixels : INTEGER;
    entree, ada, adl, ad2 : INTEGER;

BEGIN

    (* ----- initialisations des pins de sortie ----- *)

    WITH message DO
        BEGIN
            typ := 0;
            info := 0;
            z := 0;
            cos := 0;
            sin := 0;
            s := 0;
        END;
    ASSIGN message TO sort;
    ASSIGN 0 TO numess, compt_sortie ;
    ASSIGN FALSE TO lire, ecrire, remplir, vider;
    WAITFOR DELAY 10;
    compt_mess := 0;
    compt_pixels := 0;

    (* ----- traitement cellule ----- *)

    (* selection du premier tampon d'entree *)
    compt_ent := 0;

    WHILE TRUE DO
        BEGIN
            (* attente si tous les tampons d'entree sont vides *)
            IF dormir THEN WAITFOR TRUE CHECK dormir;

            (* lecture du flag du tampon d'entree *)
            ASSIGN compt_ent TO compt_entree;
            WAITFOR DELAY 5;
            flag := plein;
```

```

(* test du flag du tampon d'entree *)
IF flag THEN
  BEGIN (* traitement tampon d'entree plein *)

    (* lecture message *)
    ASSIGN TRUE TO lire;
    WAITFOR DELAY 3;
    message := ent;
    WAITFOR DELAY 2;
    ASSIGN FALSE TO lire;

    (* calcul adresse *)
    IF message.cos >= message.sin THEN

      BEGIN (* sortie est ou ouest *)
        tg := (message.sin*fact) DIV message.cos;
        long := (message.z*fact) DIV message.cos;
        mess_sortie.z := (dim-1) - ((message.z*tg) DIV fact);
        compt_sort := (compt_ent + (message.s+4)) MOD 4;
        mess_sortie.s := - message.s;
        mess_sortie.cos := message.sin;
        mess_sortie.sin := message.cos;
        mess_sortie.typ := message.typ;
        mess_sortie.info := message.info;
      END
    ELSE
      BEGIN
        tg := (message.cos*fact) DIV message.sin;
        IF (tg*k) > message.z THEN
          BEGIN
            long := (message.z*fact) DIV message.cos;
            mess_sortie.z := (dim-1) - ((message.z*fact) DIV tg);
            compt_sort := (compt_ent + (message.s+4)) MOD 4;
            mess_sortie.s := - message.s;
            mess_sortie.cos := message.sin;
            mess_sortie.sin := message.cos;
            mess_sortie.typ := message.typ;
            mess_sortie.info := message.info;
          END
        ELSE
          BEGIN
            mess_sortie.z := message.z - ((dim*tg) DIV fact);
            mess_sortie.s := message.s;
            mess_sortie.cos := message.cos;
            mess_sortie.sin := message.sin;
            mess_sortie.typ := message.typ;
            mess_sortie.info := message.info;
            compt_sort := (compt_ent+2) MOD 4;
            long := (dim*fact) DIV message.sin;
          END;
        END;
      END;

    (* lecture du flag du tampon de sortie *)
    ASSIGN compt_sort TO compt_sortie;
    WAITFOR DELAY 5;
    flag := vide;
  
```

### Annexe 3 : Les simulations du réseau

```
(* test du flag du tampon de sortie *)
IF flag THEN
  BEGIN (* traitement tampon destinataire vide *)

    (* vidage du tampon d'entree *)
    ASSIGN TRUE TO vider;
    WAITFOR DELAY 5;
    ASSIGN FALSE TO vider;

    (* traitement du tableau de pixels *)
    entree := compt_ent;
    ada := message.z DIV pdim;
    IF message.s=-1 THEN ada := (k-1)-ada;
    CASE entree OF
      0 : BEGIN (* entrée Nord *)
          ad1 := 0;
          ad2 := ada;
        END;
      1 : BEGIN (* entrée Ouest *)
          ad1 := (k-1)-ada;
          ad2 := 0;
        END;
      2 : BEGIN (* entrée Sud *)
          ad1 := k-1;
          ad2 := (k-1)-ada;
        END;
      3 : BEGIN (* entrée Est *)
          ad1 := ada;
          ad2 := k-1;
        END;
    END;
    message.z := message.z MOD pdim;
    WHILE ((ad1>=0) AND (ad2>=0)) AND
      ((ad1<=(k-1)) AND (ad2<=(k-1))) DO
      BEGIN
        IF message.cos >= message.sin THEN
          BEGIN (* sortie est ou ouest *)
            tg := (message.sin*fact) DIV message.cos;
            long := (message.z*fact) DIV message.cos;
            message.z := (pdim-1) - ((message.z*tg) DIV fact);
            entree := (entree + (message.s+2)) MOD 4;
            message.s := - message.s;
            tamp := message.cos;
            message.cos := message.sin;
            message.sin := tamp;
          END
        ELSE
          BEGIN
            tg := (message.cos*fact) DIV message.sin;
            IF tg > message.z THEN
              BEGIN
                long := (message.z*fact) DIV message.cos;
                message.z := (pdim-1)-((message.z*fact) DIV tg);
                entree := (entree+(message.s+2)) MOD 4;
                message.s := - message.s;
                tamp := message.cos;
                message.cos := message.sin;
                message.sin := tamp;
              END
            END
          END
        END
      END
```

```

ELSE
  BEGIN
    message.z := message.z - ((pdim*tg) DIV fact);
    long := (pdim*fact) DIV message.sin;
  END;
END;
CASE entree OF
  0 : ad1 := ad1 + 1;
  1 : ad2 := ad2 + 1;
  2 : ad1 := ad1 - 1;
  3 : ad2 := ad2 - 1;
END;
compt_pixels := compt_pixels + 1;
(* attente temps de traitement pixel *)
WAITFOR DELAY (trait_message);
END;

IF calcul_nb_mess = 3 THEN
  BEGIN
    ASSIGN compt_pixels TO numess;
  END
ELSE ASSIGN message.typ TO numess;

(* passage du message *)
ASSIGN mess_sortie TO sort;
ASSIGN TRUE TO ecrire;
WAITFOR DELAY 5;
ASSIGN FALSE TO ecrire;
WAITFOR DELAY 5;

(* remplissage du tampon de sortie *)
ASSIGN TRUE TO remplir;
WAITFOR DELAY 5;
ASSIGN FALSE TO remplir;
WAITFOR DELAY 5;

IF calcul_nb_mess = 0 THEN ASSIGN 0 TO numess;
END
END

(* passage au tampon d'entree suivant *)
compt_ent := (compt_ent + 1) MOD 4;

END;
END;

```



## ***Annexe 4***

---

### Description de l'architecture de la cellule de base

|                                                          |            |
|----------------------------------------------------------|------------|
| <b>1. Les simulations comportementales HELIX ....</b>    | <b>275</b> |
| 1.1. Schémas de l'architecture de la cellule.....        | 275        |
| 1.2. Modèles de quelques éléments.....                   | 281        |
| <b>2. Simulations électriques d'un additionneur ....</b> | <b>285</b> |
| 2.1. Additionneur séquentiel .....                       | 285        |
| 2.2. Additionneur à retenue bondissante .....            | 285        |
| <b>3. Simulations logiques.....</b>                      | <b>293</b> |
| 3.1. Schémas.....                                        | 293        |
| 3.2. Résultats .....                                     | 293        |
| <b>4. Dessins des masques.....</b>                       | <b>305</b> |
| 4.1. Bibliothèque SOLO1200 .....                         | 305        |
| 4.2. Modules du circuit.....                             | 307        |
| 4.3. Circuit .....                                       | 309        |
| <b>5. Plan de montage .....</b>                          | <b>310</b> |

## **1. LES SIMULATIONS COMPORTEMENTALES HELIX**

### **1.1. Schémas de l'architecture de la cellule**

Le schéma 1 présente l'architecture complète de la cellule : le séquenceur, la partie opérative et les 8 tampons d'entrée et sortie.

Le schéma 2 montre le séquenceur, le mécanisme de multiplexage permettant d'accéder aux tampons, un tampon d'entrée et un tampon de sortie.

Le schéma 3 détaille le module arithmétique.

Le schéma 4 illustre les registres Z et ZS, et le module combinatoire.

Le schéma 5 montre la structure du module combinatoire.



Schéma 1. La cellule

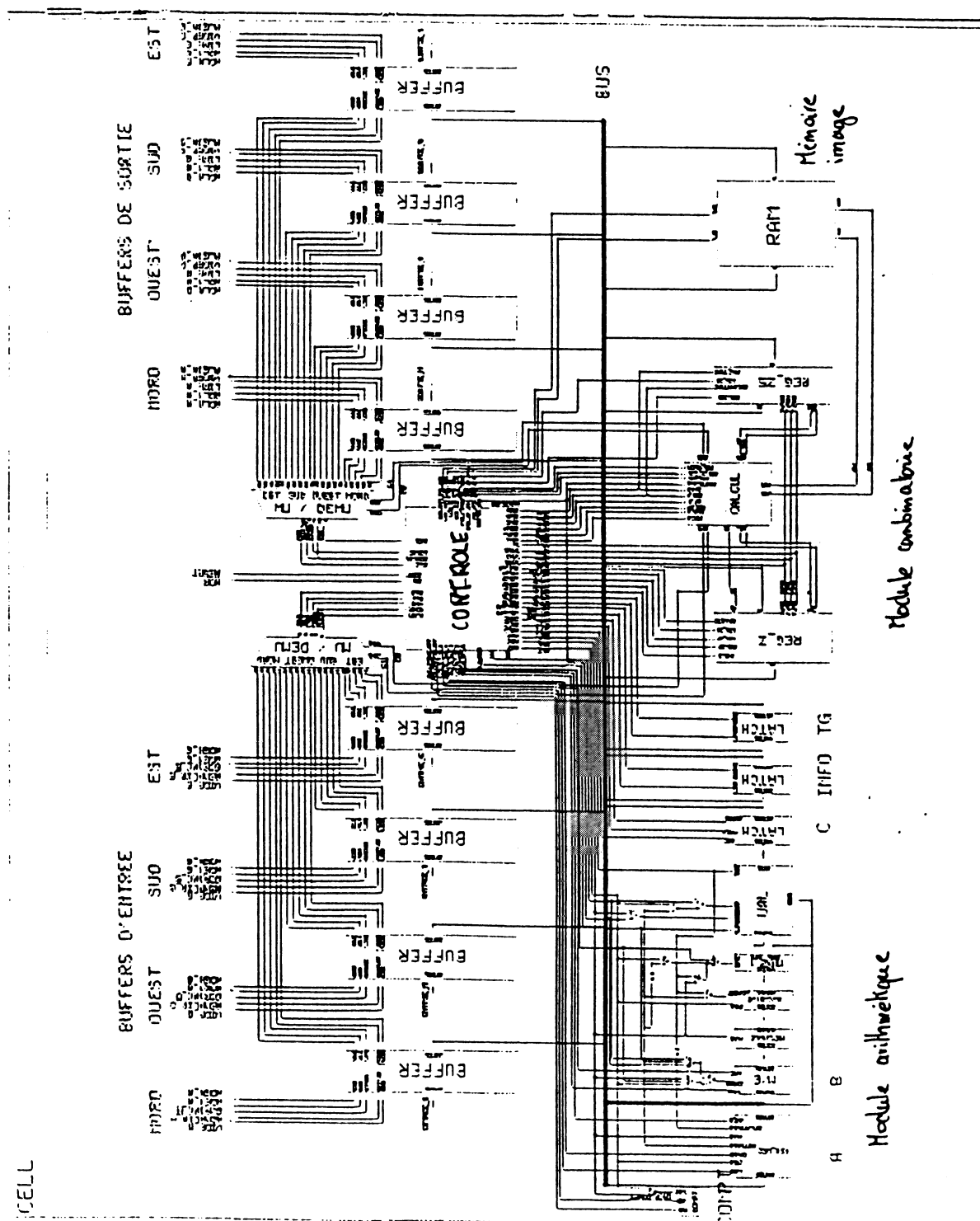


Schéma 2. Séquenceur et accès aux tampons

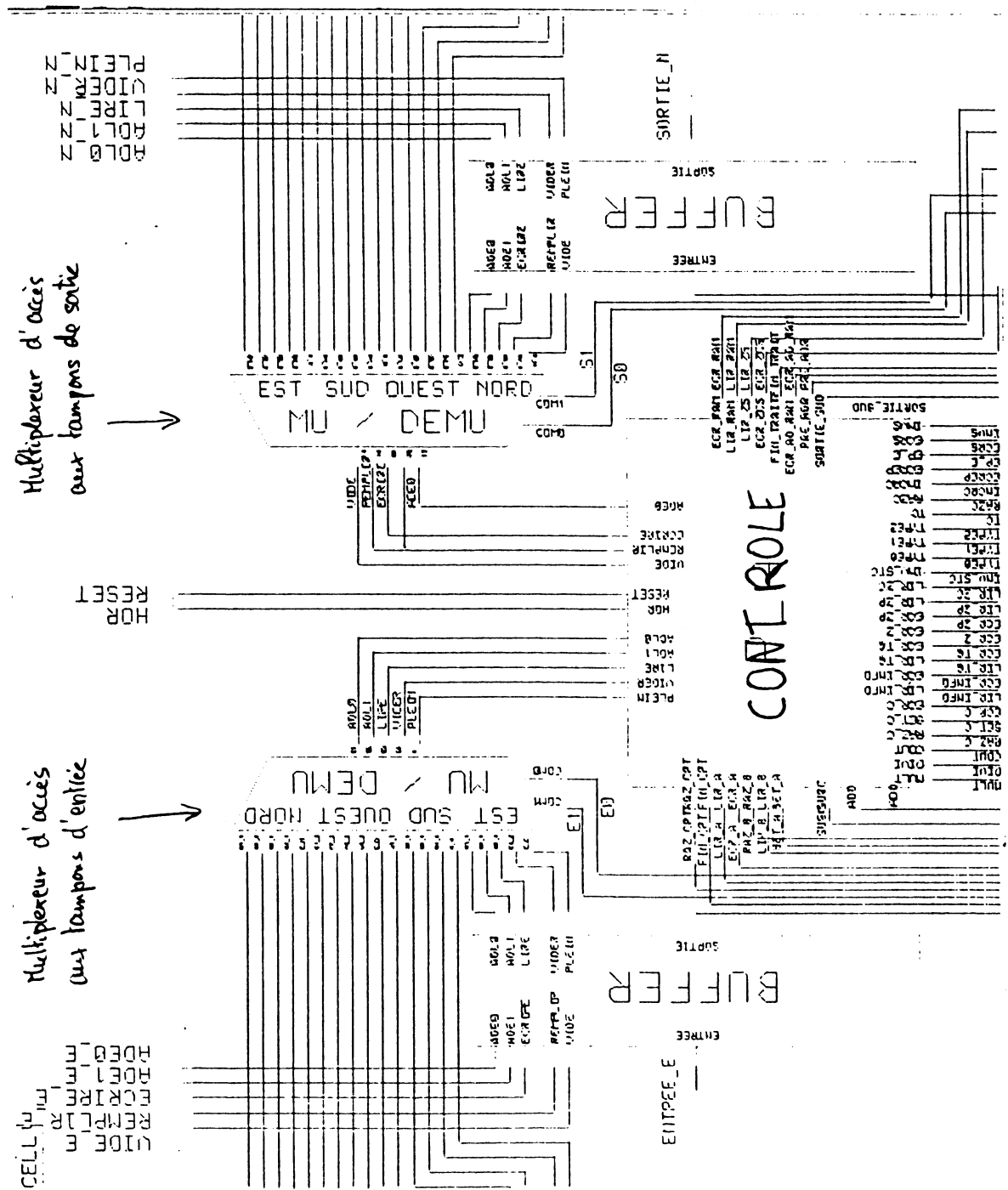
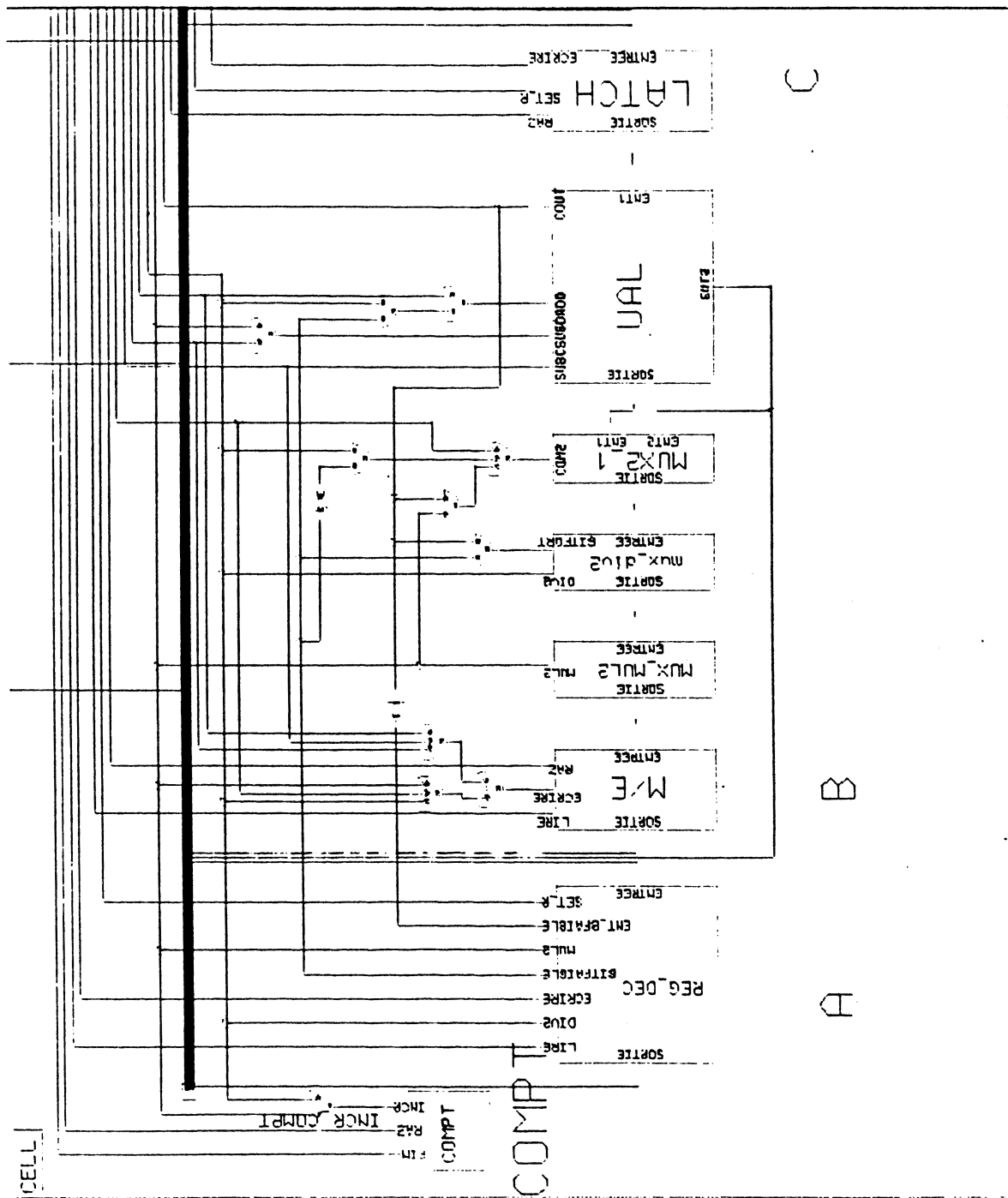
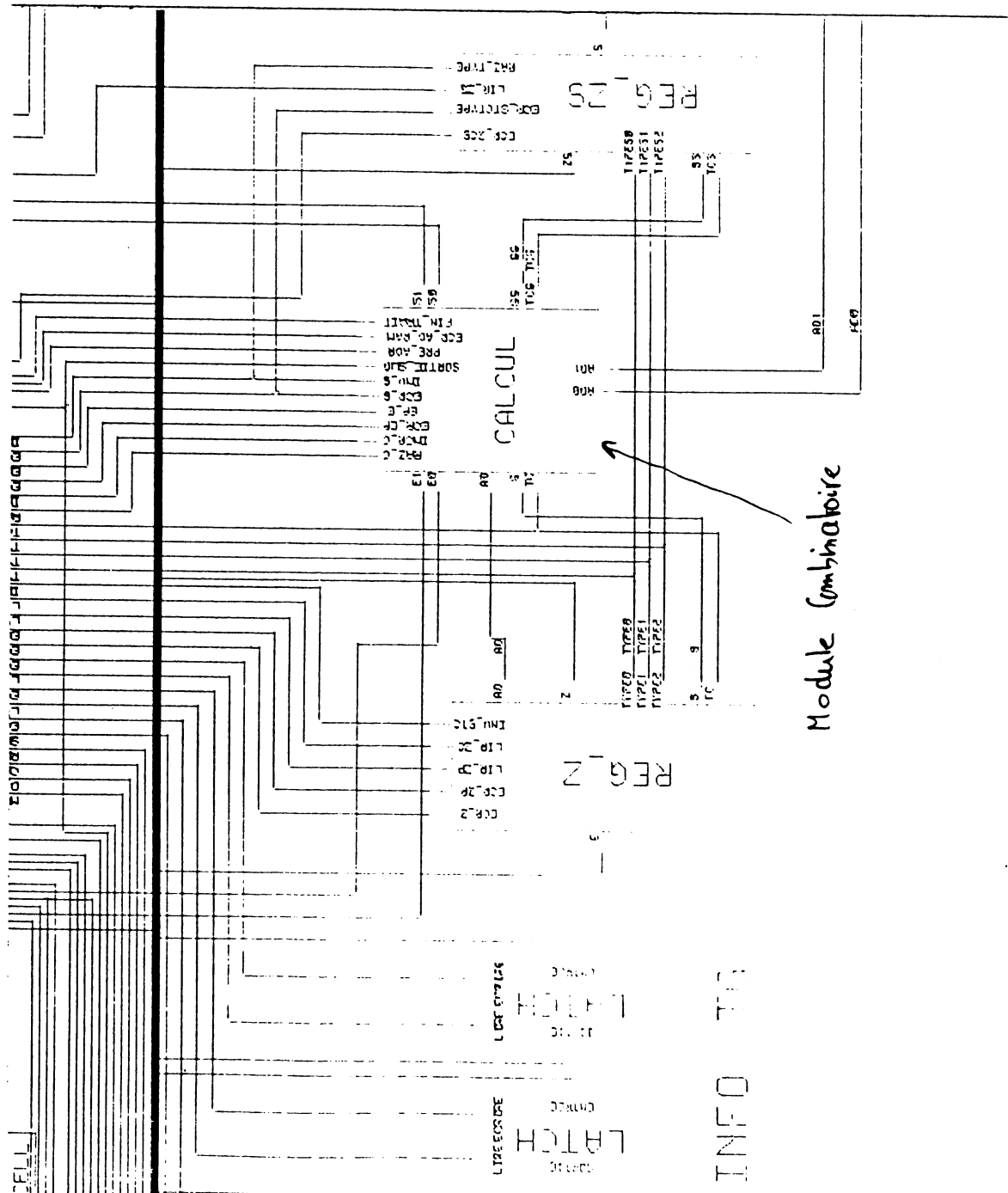
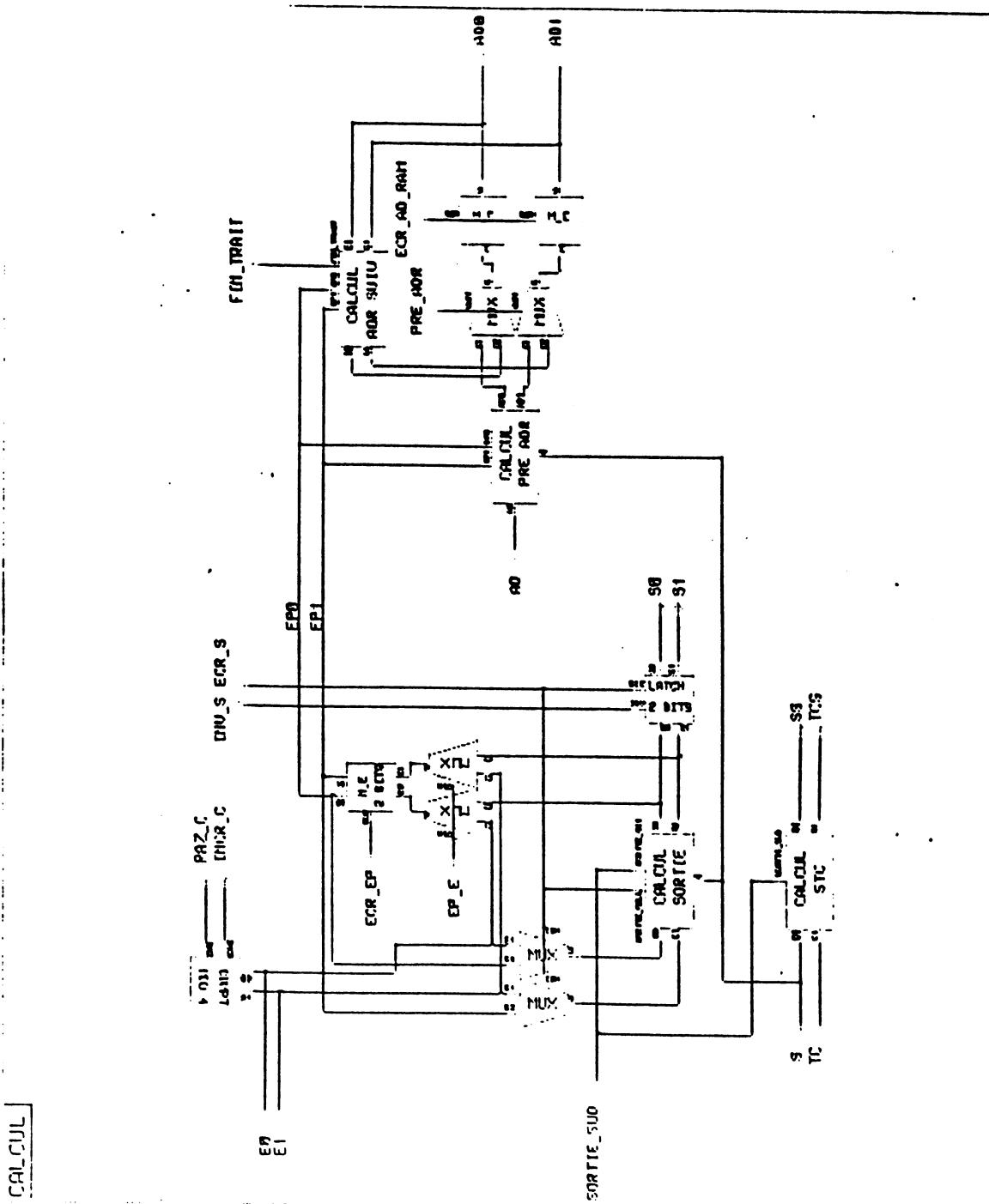


Schéma 3. Module arithmétique





### Schéma 5. Structure du module combinatoire



## 1.2. Modèles de quelques éléments

```
(*----- COMPOSANT TAMPON -----*)

COMPTYPE buffer;

INWARD lire, adl1, adl0, ecrire, adel, ade0, remplir, vider : boolnet;
INWARD entree : intnet;
OUTWARD plein, vide : boolnet;
OUTWARD sortie : intnet;

VAR z, tg, info : INTEGER;

SUBPROCESS
  remplissage :
    UPON NOT remplir CHECK remplir DO
      BEGIN
        ASSIGN FALSE TO vide;
        ASSIGN TRUE TO plein;
      END;

  vidage :
    UPON NOT vider CHECK vider DO
      BEGIN
        ASSIGN FALSE TO plein;
        ASSIGN TRUE TO vide;
      END;

  lecture :
    UPON lire CHECK lire DO
      BEGIN
        IF NOT adl1 THEN
          IF NOT adl0 THEN (* Z *)
            ASSIGN z TO sortie
          ELSE ASSIGN tg TO sortie (* TG *)
          ELSE ASSIGN info TO sortie; (* INFO *)
        END;

  ecriture :
    UPON NOT ecrire CHECK ecrire DO
      IF NOT RECALL(adel) THEN
        IF NOT RECALL(ade0) THEN (* Z *)
          z := entree
        ELSE tg := entree (* TG *)
        ELSE info := entree; (* INFO *)

BEGIN
  (* initialisations des pins de sortie *)
  ASSIGN FALSE TO plein;
  ASSIGN TRUE TO vide;
  ASSIGN 0 TO sortie;
  WAITFOR DELAY 10;
END;
```

## Annexe 4 : Description de l'architecture de la cellule de base

```

(*----- COMPOSANT COMPTEUR MODULO 4 -----*)

COMPTYPE compt_4;

INWARD raz, incr : boolnet;
OUTWARD s1, s0 : boolnet;

VAR valeur_compteur : INTEGER;

SUBPROCESS
  raz_compteur :
    UPON raz CHECK raz DO
      BEGIN
        valeur_compteur := 0;
        ASSIGN FALSE TO s1, s0;
      END;

  incr_compteur :
    UPON incr CHECK incr DO
      BEGIN
        valeur_compteur := (valeur_compteur + 1) MOD 4;
        ASSIGN ((valeur_compteur = 1) OR (valeur_compteur = 3)) TO s0;
        ASSIGN ((valeur_compteur = 2) OR (valeur_compteur = 3)) TO s1;
      END;

BEGIN
  (* initialisations des pins de sortie *)
  ASSIGN FALSE TO s1, s0;
  WAITFOR DELAY 10;
END;

(*----- COMPOSANT CALCUL STC -----*)

COMPTYPE cal_stc ;

INWARD sortie_sud : boolnet;
INWARD e0, e1 : boolnet;
OUTWARD s0, s1 : boolnet;

SUBPROCESS
  calcul :
    UPON TRUE CHECK sortie_sud, e0, e1 DO
      BEGIN
        IF sortie_sud THEN      (* ss = s et tcs = tc *)
          BEGIN
            ASSIGN e1 TO s1;
            ASSIGN e0 TO s0;
          END
        ELSE BEGIN              (* ss = NOT s et tcs = NOT tcs *)
          ASSIGN NOT e1 TO s1;
          ASSIGN NOT e0 TO s0;
        END
      END;
    END;

BEGIN
  (* initialisations des pins de sortie *)
  ASSIGN FALSE TO s0, s1;
  WAITFOR DELAY 10;
END;

```

## Annexe 4 : Description de l'architecture de la cellule de base

```

(*----- COMPOSANT LATCH -----*)

COMPTYPE latch ;

INWARD raz, ecrire, lire : boolnet;
INWARD entree : intnet;
BOTHWAYS sortie : intnet;

VAR valeur_registre : INTEGER;

SUBPROCESS
  raz_registre :
    UPON raz CHECK raz DO
      BEGIN
        valeur_registre := 0;
      END;
  ecr_registre :
    UPON NOT ecrire CHECK ecrire DO
      BEGIN
        valeur_registre := entree;
      END;
  lire_registre :
    UPON lire CHECK lire DO
      ASSIGN valeur_registre TO sortie;

BEGIN
  (* initialisations des pins de sortie *)
  valeur_registre := 0;
  ASSIGN valeur_registre TO sortie ;
  WAITFOR DELAY 10;
END;

(*----- COMPOSANT RAM -----*)

COMPTYPE ram ;

INWARD ecr, lir : boolnet;
INWARD e, ad0, ad1 : intnet;
OUTWARD s : intnet;

VAR tab : ARRAY[0..7,0..7] OF INTEGER;
    val_ad1, val_ad0,i ,j : INTEGER;

SUBPROCESS
  ecr_ram :
    UPON NOT ecr CHECK ecr DO
      BEGIN
        val_ad1 := ad1;
        val_ad0 := ad0;
        tab[val_ad1,val_ad0] := e;
      END;
  lir_ram :
    UPON lir CHECK lir DO
      BEGIN
        val_ad1 := ad1;
        val_ad0 := ad0;
        ASSIGN tab[val_ad1,val_ad0] TO s;
      END;

```



## Annexe 4 : Description de l'architecture de la cellule de base

```
BEGIN
  (* initialisations des pins de sortie *)
  FOR i := 0 TO 7 DO
    FOR j := 0 TO 7 DO
      tab[i,j] := 0;
    ASSIGN 0 TO s ;
    WAITFOR DELAY 10;
  END;

  (*----- COMPOSANT MUX 2-->1 ENTIER -----*)

  COMPTYPE mux2_1;

  INWARD com2 : boolnet;
  INWARD ent1, ent2 : intnet;
  OUTWARD sortie : intnet;

  SUBPROCESS
    multiplexage :
      UPON TRUE CHECK com2, ent1, ent2 DO
        BEGIN
          IF com2 THEN ASSIGN ent2 TO sortie DELAY 1
          ELSE ASSIGN ent1 TO sortie DELAY 1
        END;
  END;

  BEGIN
    (* initialisations des pins de sortie *)
    ASSIGN 0 TO sortie;
    WAITFOR DELAY 10;
  END;

  (*----- COMPOSANT MUX 2-->1 BOOLEEN -----*)

  COMPTYPE muxb2_1;

  INWARD com : boolnet;
  INWARD e1, e2 : boolnet;
  OUTWARD s : boolnet;

  SUBPROCESS
    multiplexage :
      UPON TRUE CHECK com, e1, e2 DO
        BEGIN
          IF com THEN ASSIGN e1 TO s
          ELSE ASSIGN e2 TO s
        END;
  END;

  BEGIN
    (* initialisations des pins de sortie *)
    ASSIGN TRUE TO s;
    WAITFOR DELAY 10;
  END;
```

## **2. SIMULATIONS ELECTRIQUES D'UN ADDITIONNEUR**

Ces simulations électriques ont été réalisées grâce au simulateur électrique ELDO développé au CNET. Les schémas ont été saisis à l'aide de l'éditeur SDS de Silvar-Lisco. Le but de ces simulations est de montrer que des additions sur 16 bits peuvent être réalisées en des temps très courts (10 nanosecondes et moins) si les cellules de base utilisées sont bien dimensionnées. Cette constatation prouve que le circuit aurait pu avoir de bien meilleures performances s'il avait été dessiné à la main.

### **2.1. Additionneur séquentiel**

Le schéma 1 présente la structure d'un additionneur 1 bit construit à partir de 2 portes ou-exclusif et 2 portes non-et à 2 entrées.

Le schéma 2 montre un additionneur séquentiel 4 bits construit à partir de l'additionneur 1 bit précédent et un additionneur séquentiel 16 bits.

Le schéma 3 détaille la simulation réalisée avec les capacités de sortie et les chronogrammes d'entrée. L'objectif est d'étudier le temps de propagation de la retenue dans l'additionneur séquentiel 16 bits.

Le schéma 4 illustre les résultats, le temps de propagation est de 10 nanosecondes.

### **2.2. Additionneur à retenue bondissante**

Le schéma 5 montre un additionneur à retenue bondissante de 16 bits. Les tranches de bits sont regroupées de manière irrégulière afin d'obtenir un meilleur résultat (1, 2, 3, 4, 3, 2, 1).

Le schéma 6 détaille la structure des blocs réalisant le rebond de la retenue pour 2, 3 et 4 bits.

Le schéma 7 illustre les résultats de simulation de l'additionneur 16 bits à retenue bondissante avec les mêmes capacités et chronogrammes. Le temps de propagation est de 7 nanosecondes.

Schéma 1. Additionneur 1 bit

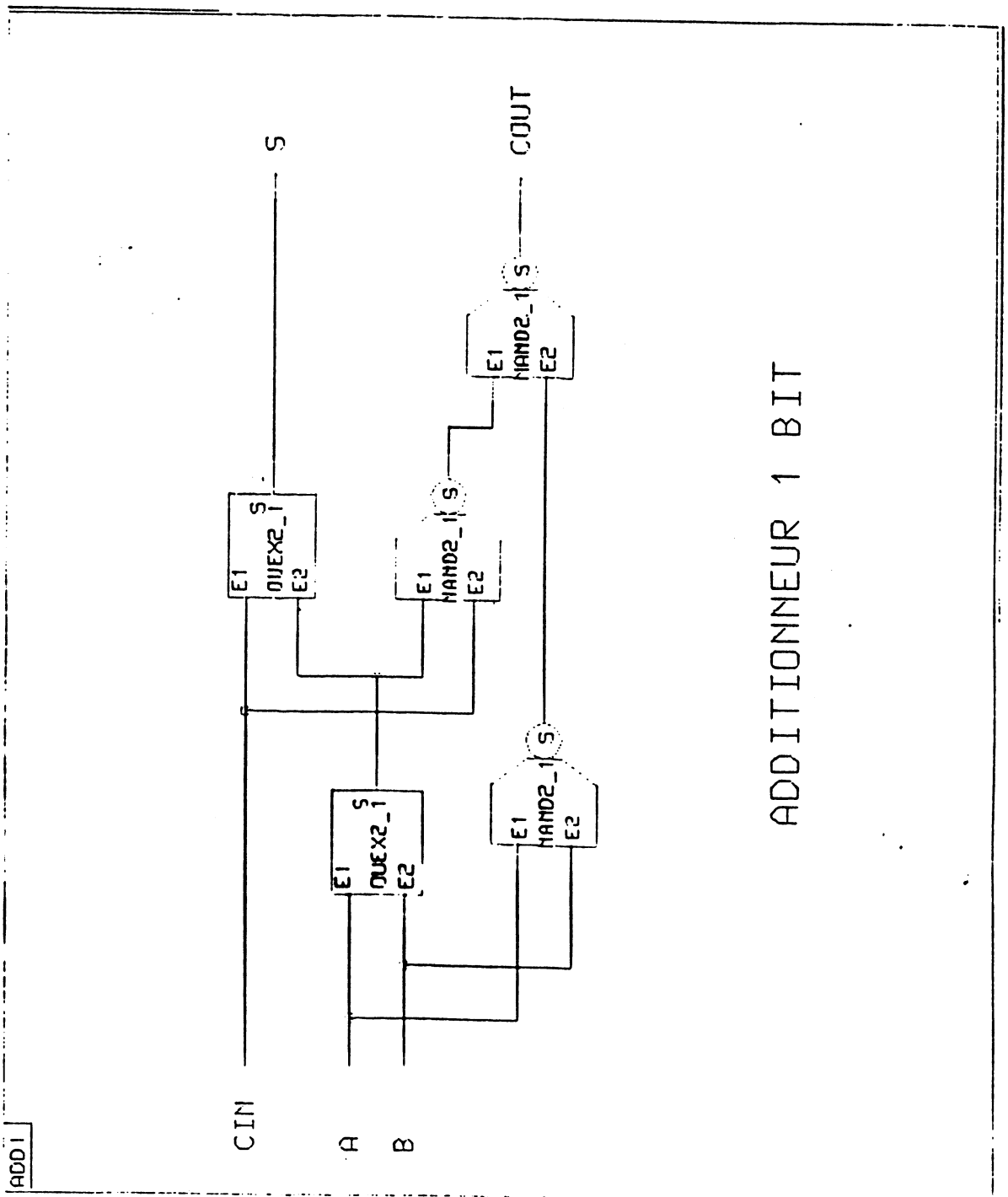
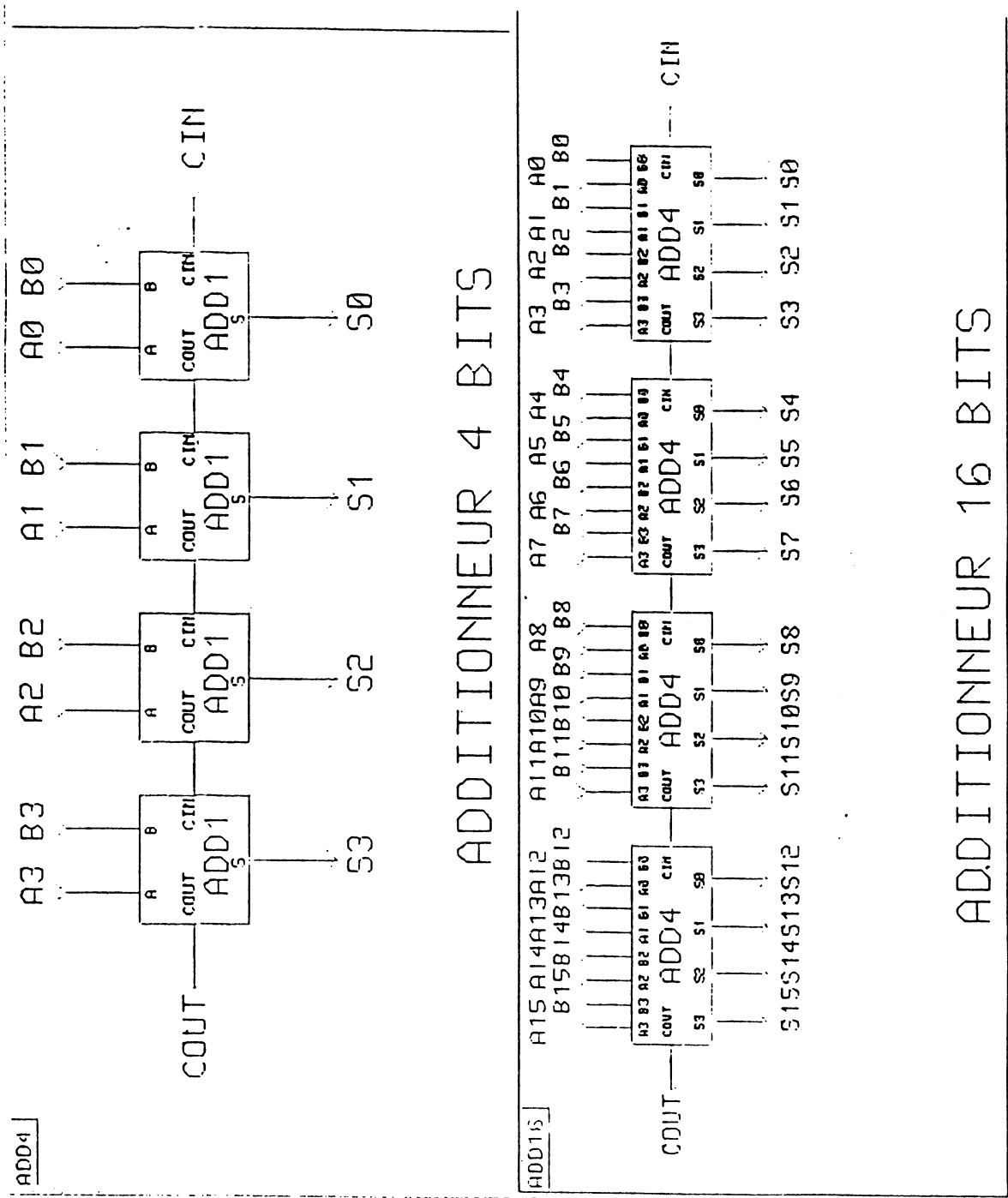
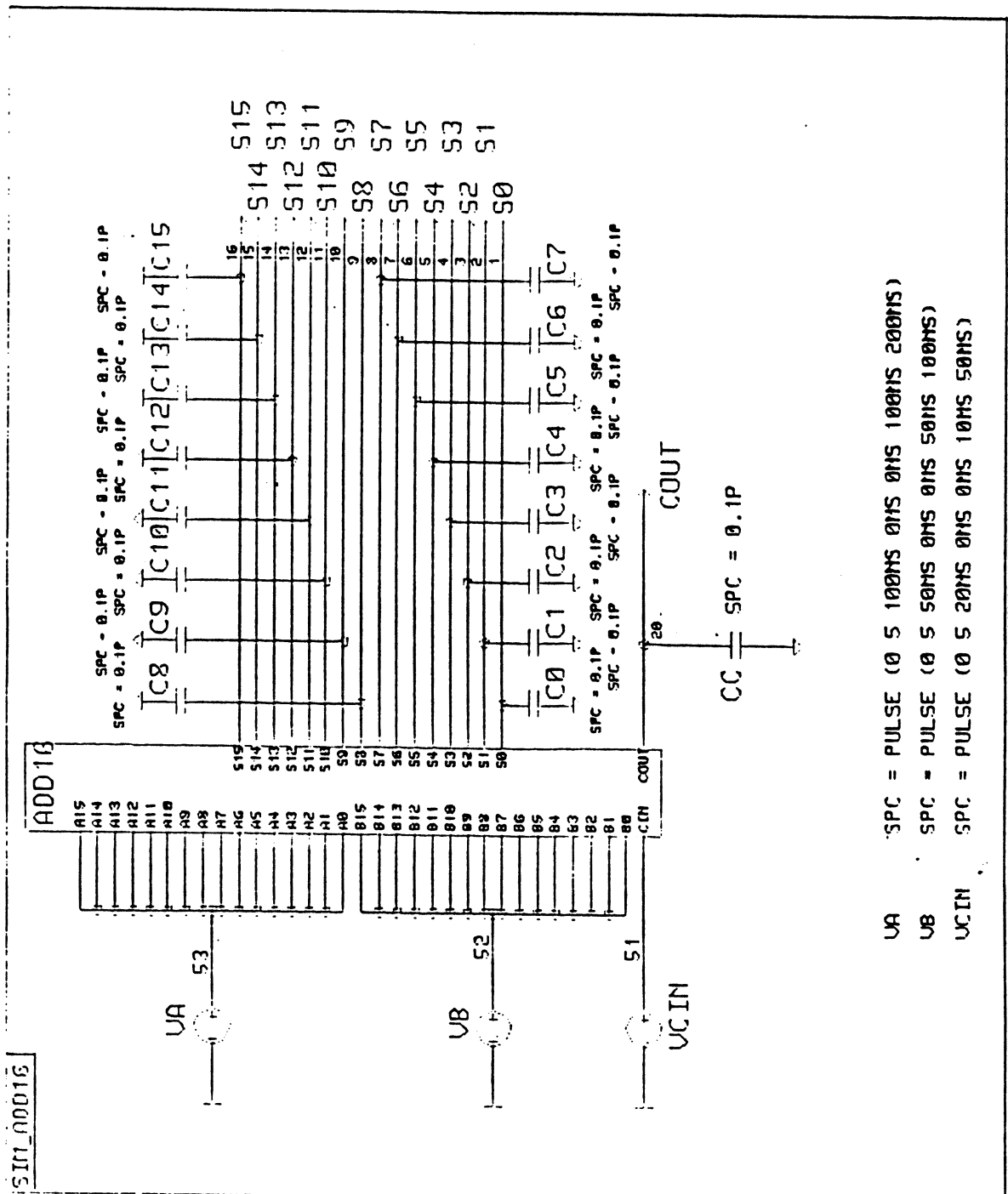


Schéma 2. Additionneurs séquentiels 4 et 16 bits



### Schéma 3. Schéma de simulation de l'additionneur séquentiel



**Schéma 4. Résultat de la simulation de l'additionneur séquentiel**

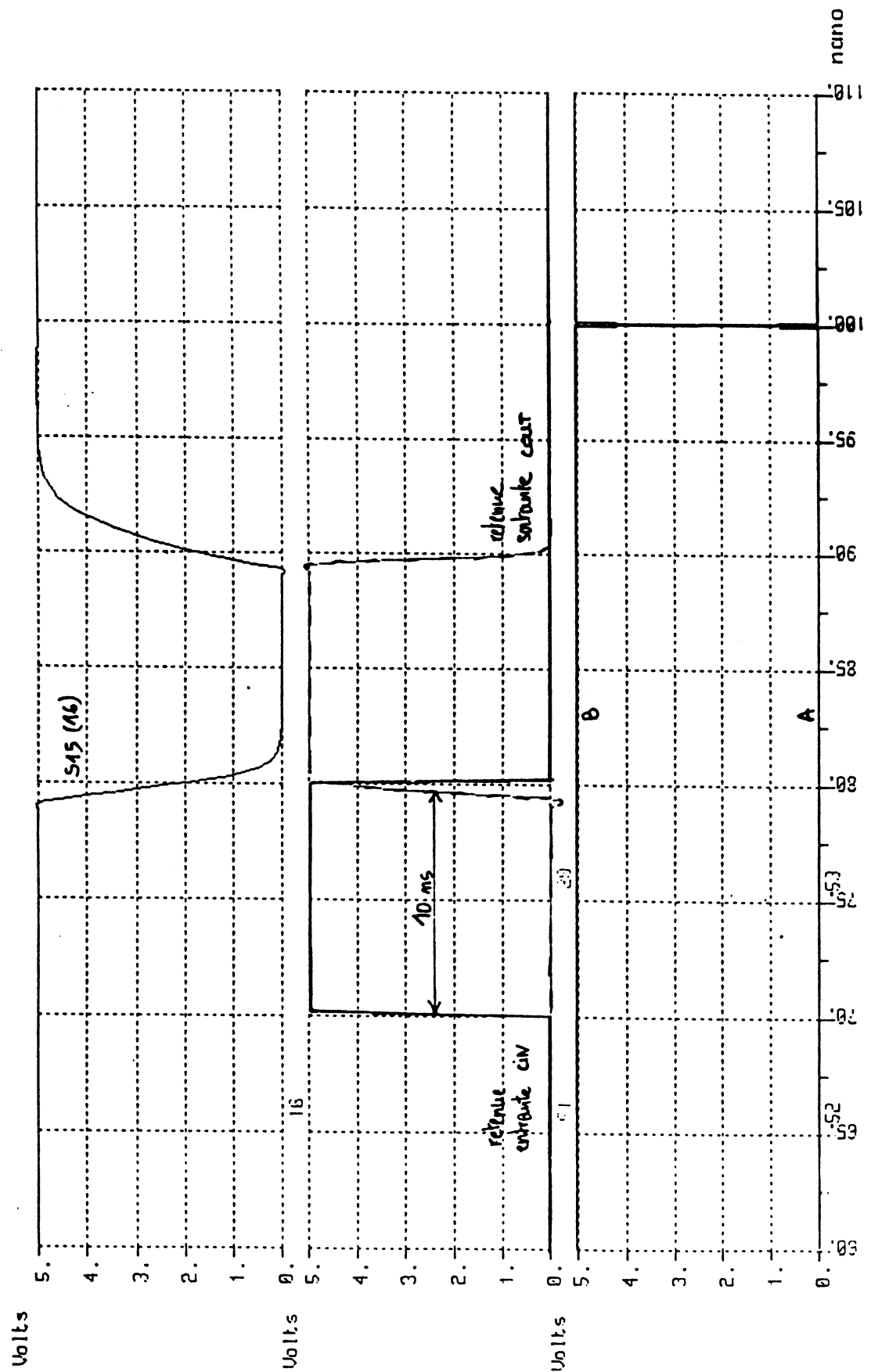


Schéma 5. Additionneur à retenue bondissante 16 bits

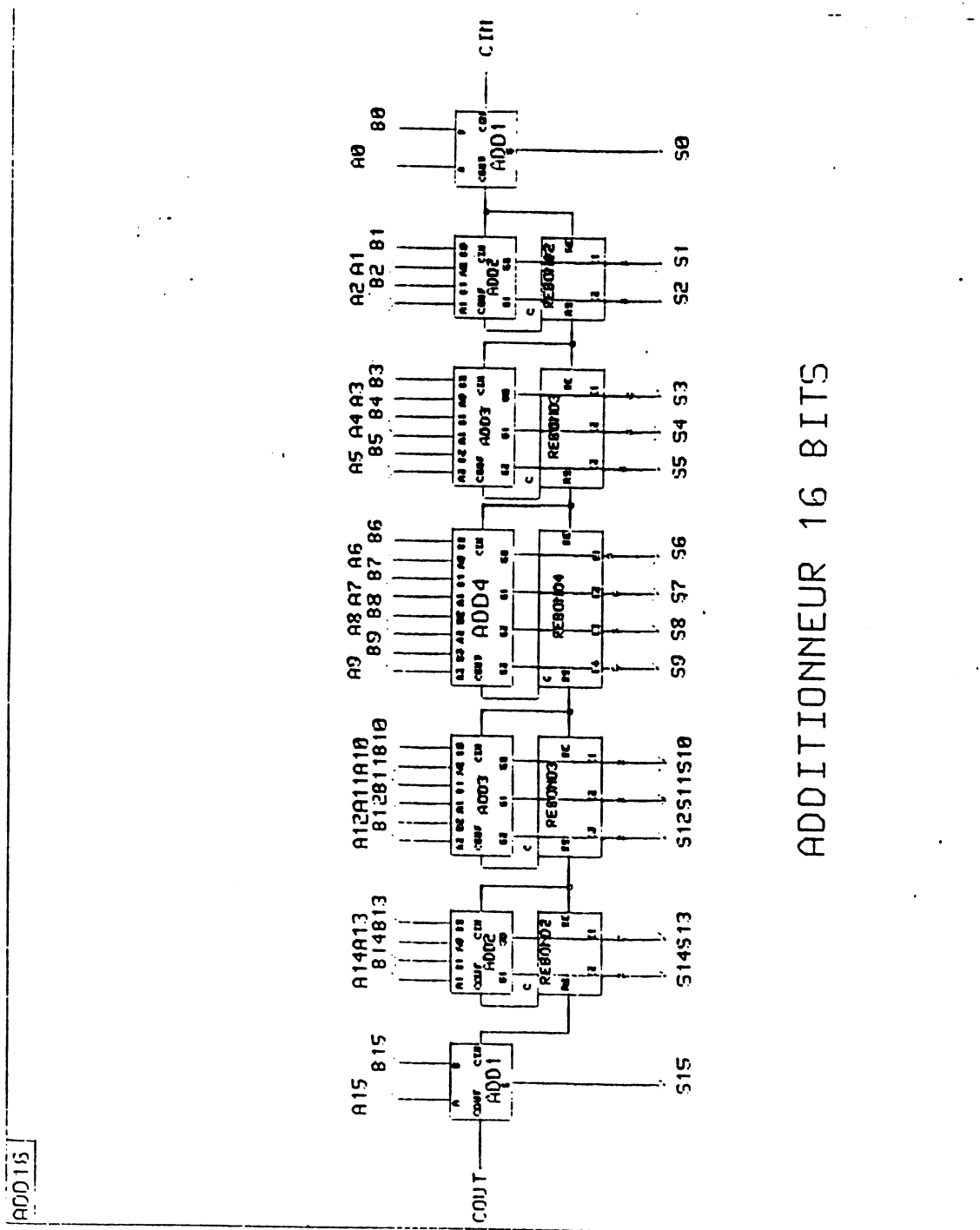
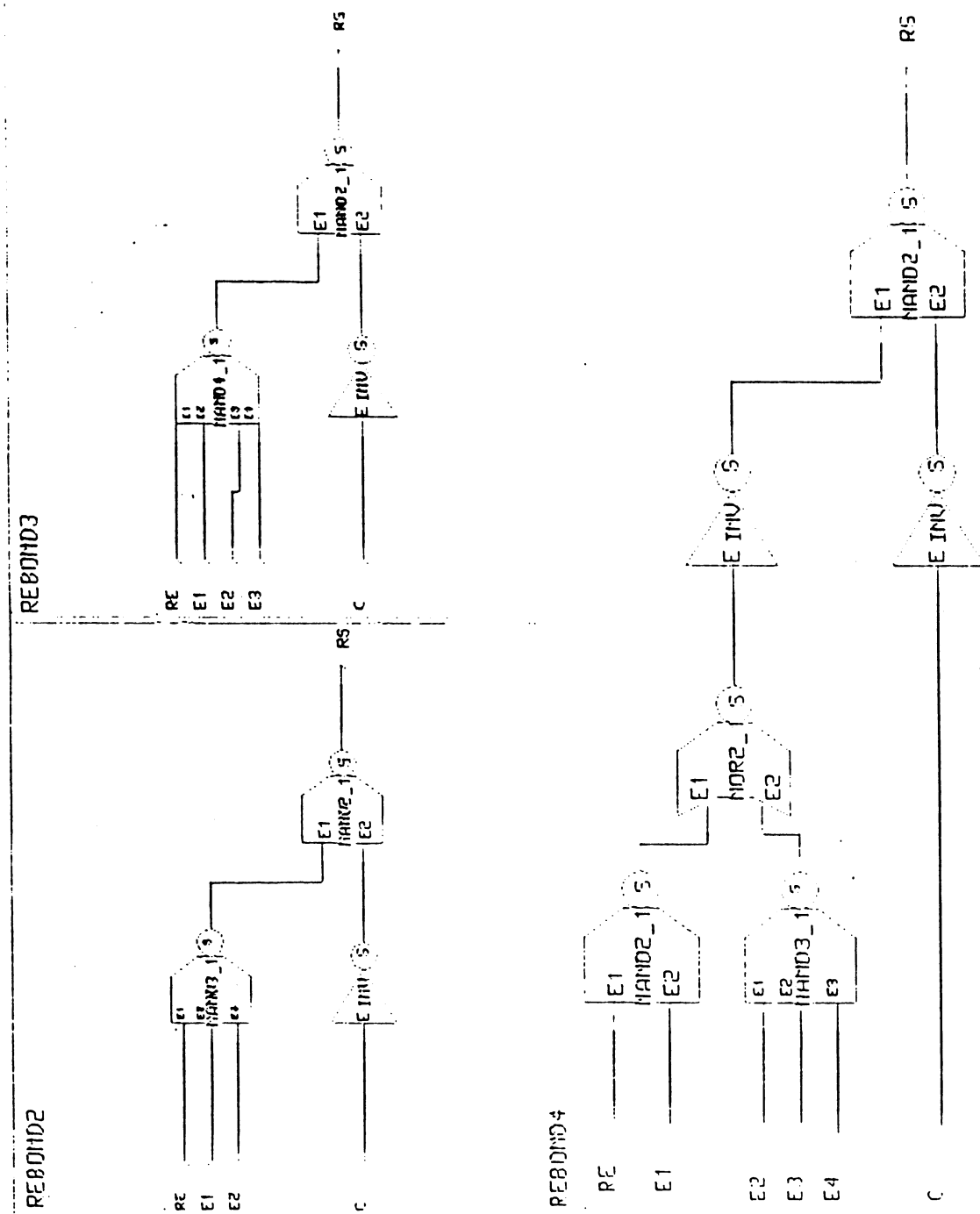
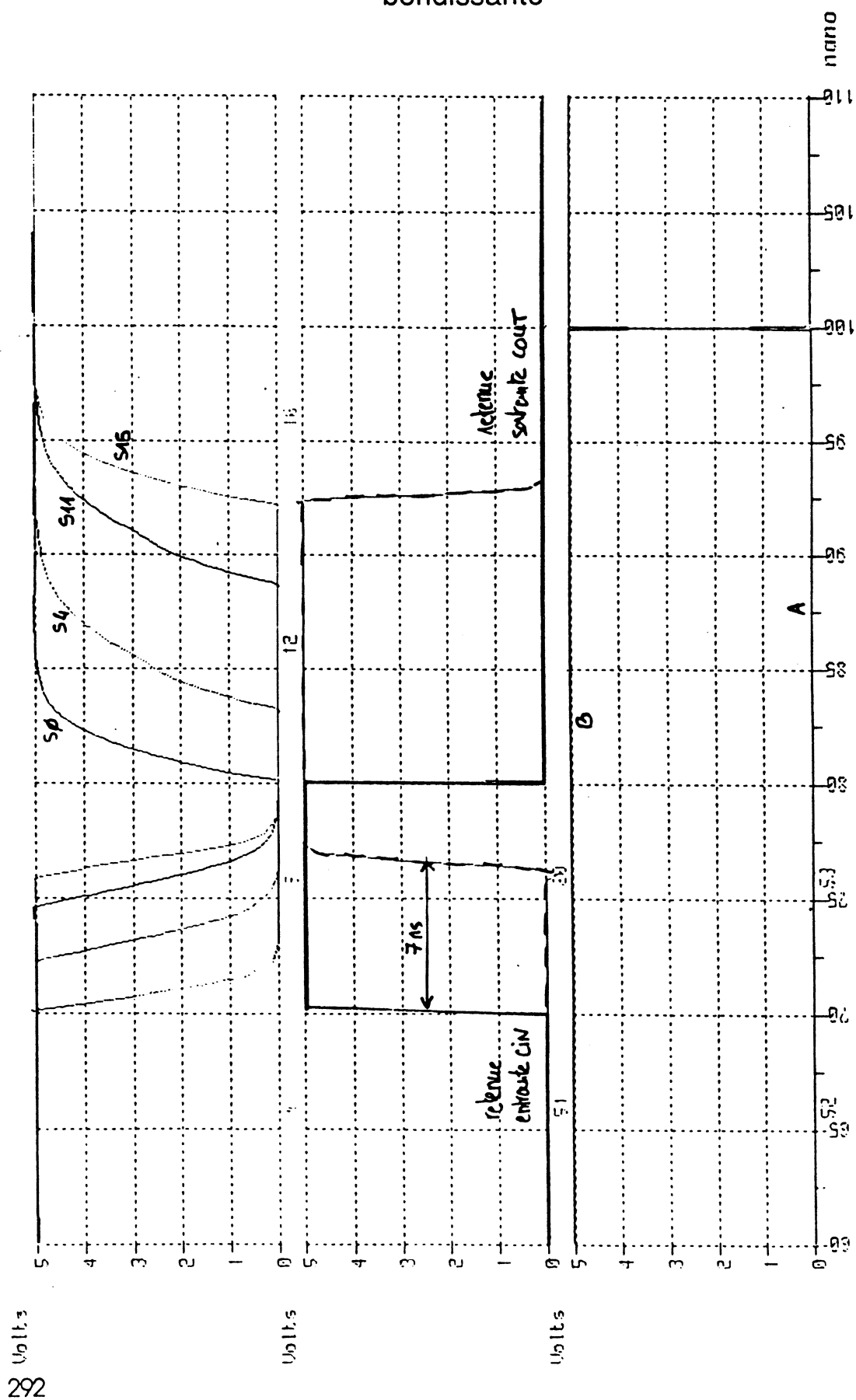


Schéma 6. Blocs de rebond





**Schéma 7. Résultat de la simulation de l'additionneur à retenue bondissante**



### 3. SIMULATIONS LOGIQUES

#### 3.1. Schémas

Les schémas logiques présentés dans ce paragraphe ont été réalisés grâce au logiciel SOLO 1200. Ils ont servi de base pour la génération automatique des dessins des masques du circuit.

Nous donnons le schéma complet du circuit (schéma 1), et les schémas de quelques parties, notamment :

- un tampon (schéma 2),
- le module de mémorisation (schéma 3),
- le registre B du module arithmétique (schéma 4),
- le multiplexeur réalisant les décalages à droite et à gauche utilisé pour les opérations de multiplication et division (schéma 5),
- et le bloc *Calcul Sortie* du module combinatoire utilisé pour déterminer le tampon de sortie destinataire (schéma 6).

#### 3.2. Résultats

Nous présentons ensuite les résultats de quelques simulations.

Le schéma 8 illustre la simulation de l'additionneur séquentiel, et le schéma 9 la simulation de l'additionneur à retenue bondissante. L'additionneur utilisé dans le circuit est l'additionneur séquentiel parce qu'il est plus rapide : le temps de propagation de la retenue est de 30 nanosecondes contre 35 pour l'additionneur à retenue bondissante.

Les schémas 10 et 11 illustrent des simulations du circuit complet faites à partir de messages tests.

Schéma 1. Le circuit

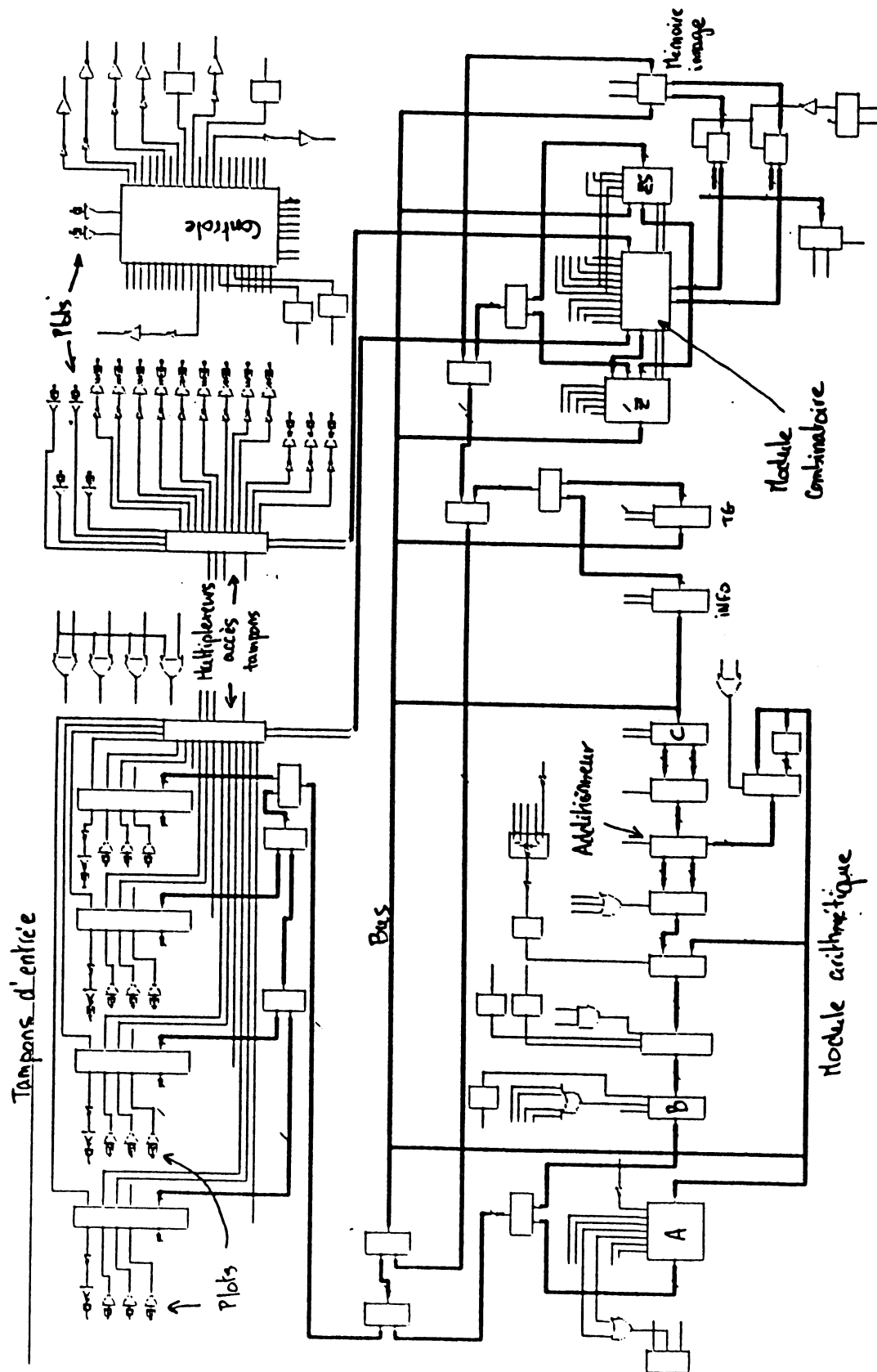


Schéma 2. Un tampon

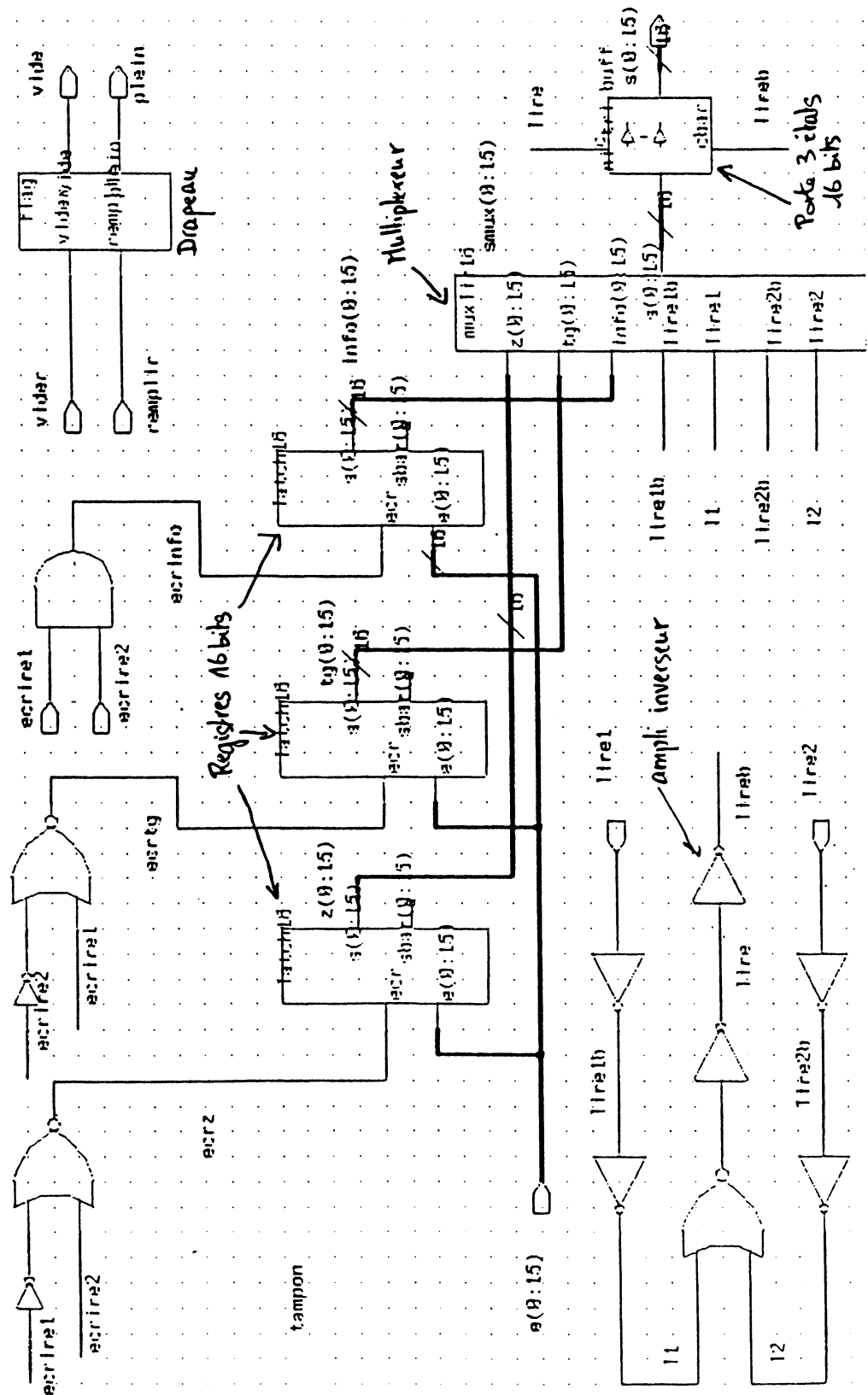


Schéma 3. Module de mémorisation de l'image

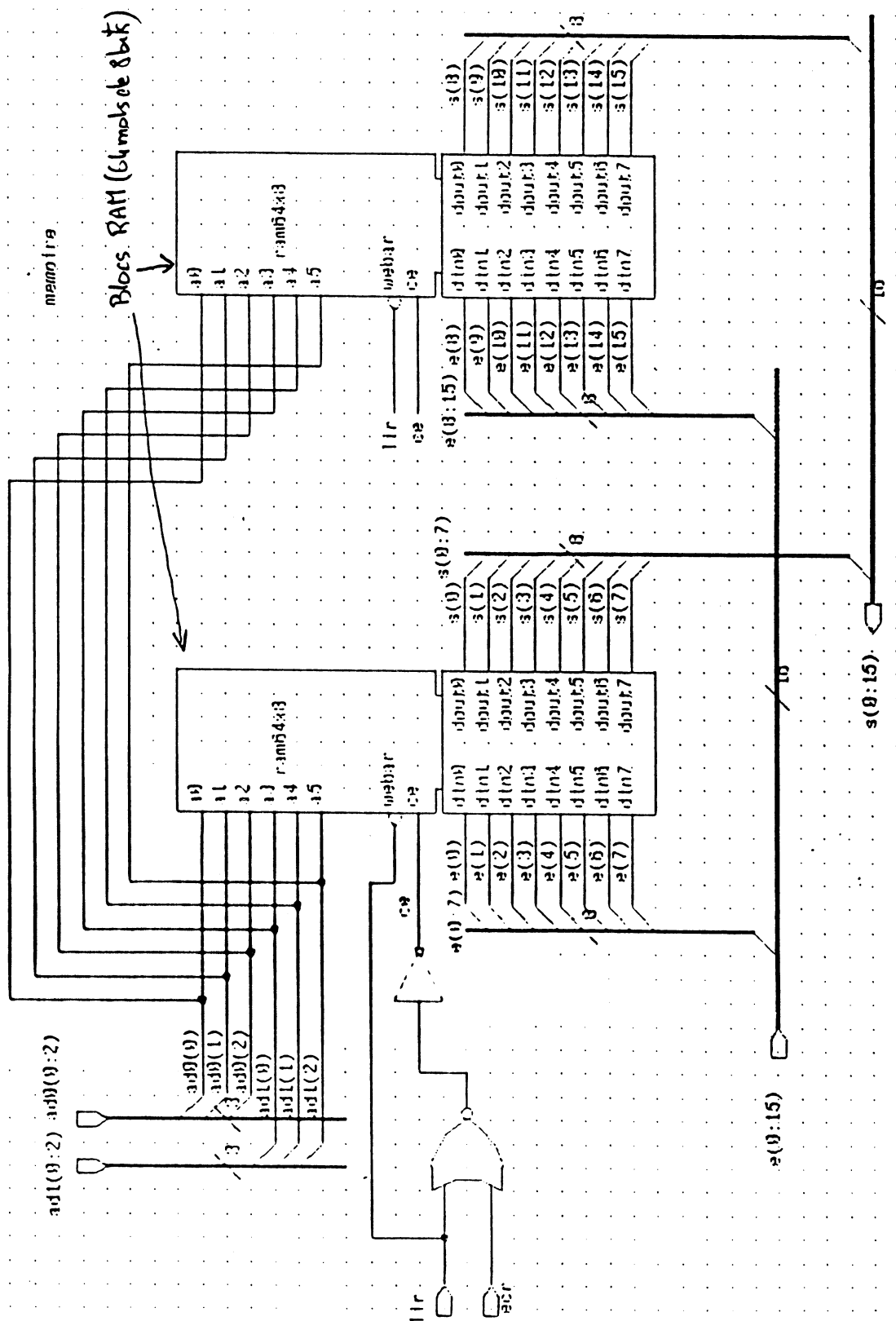


Schéma 4. Le registre B du module arithmétique

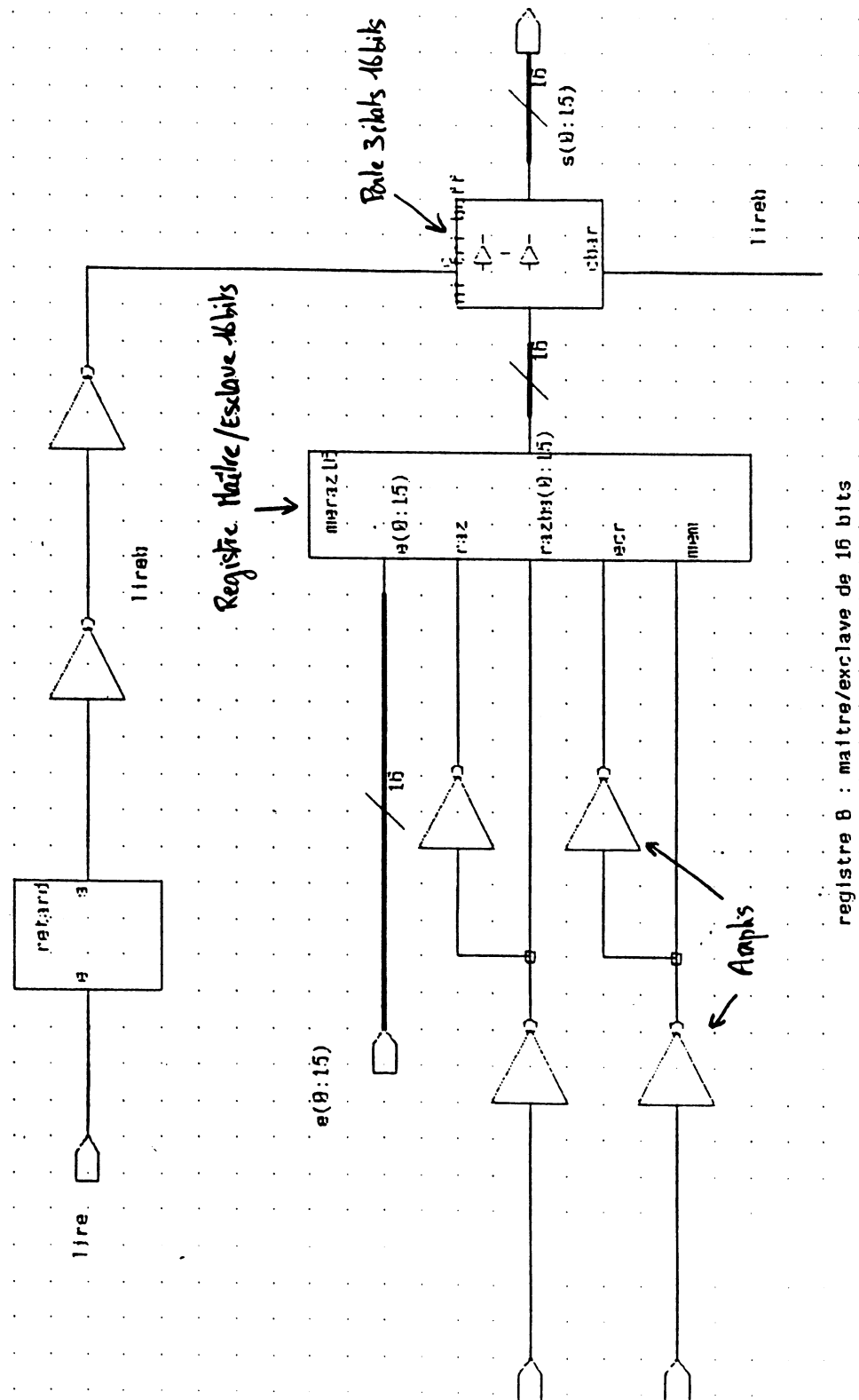


Schéma 5. Le multiplexeur réalisant les décalages à gauche et à droite du module arithmétique

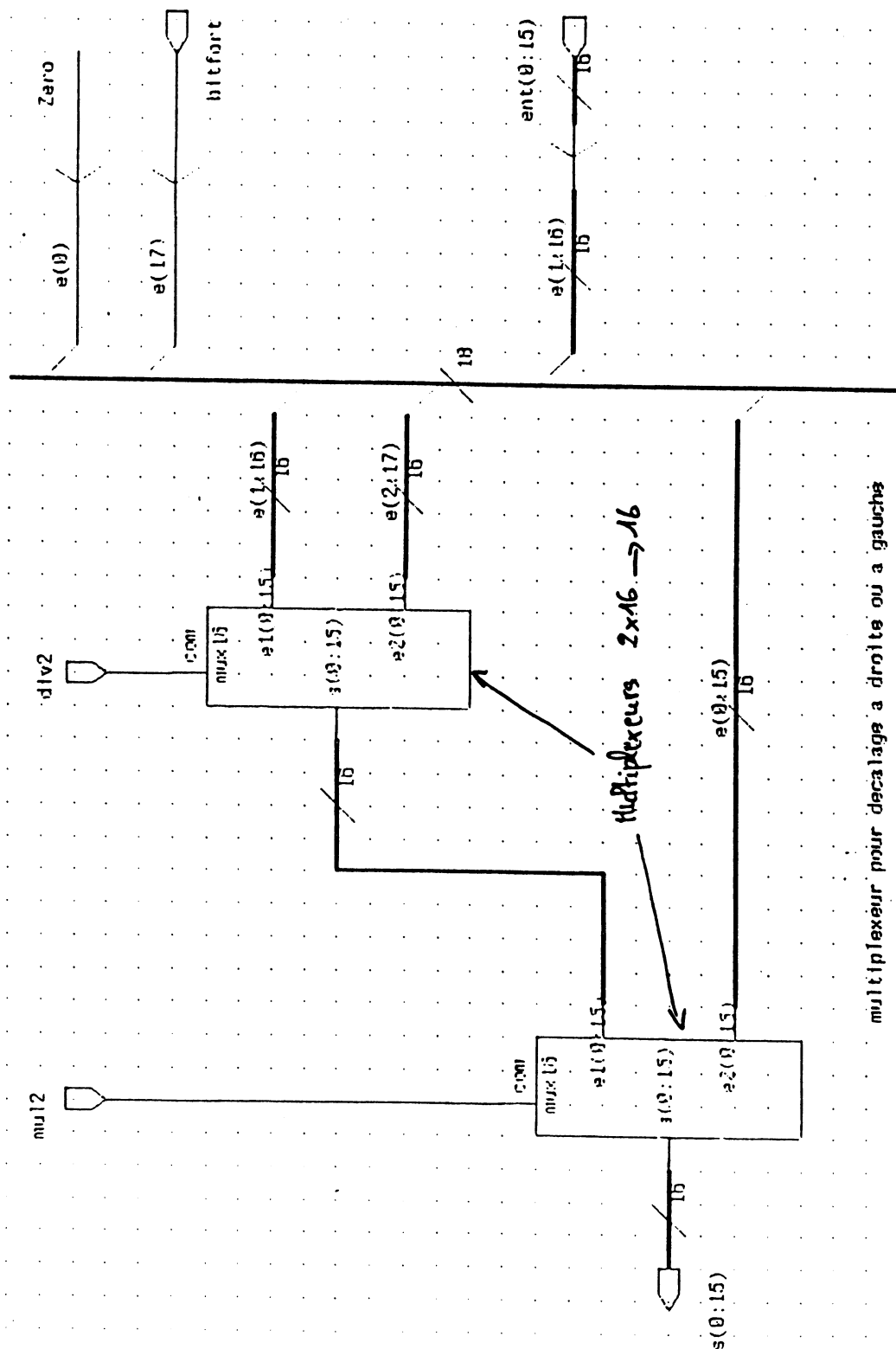
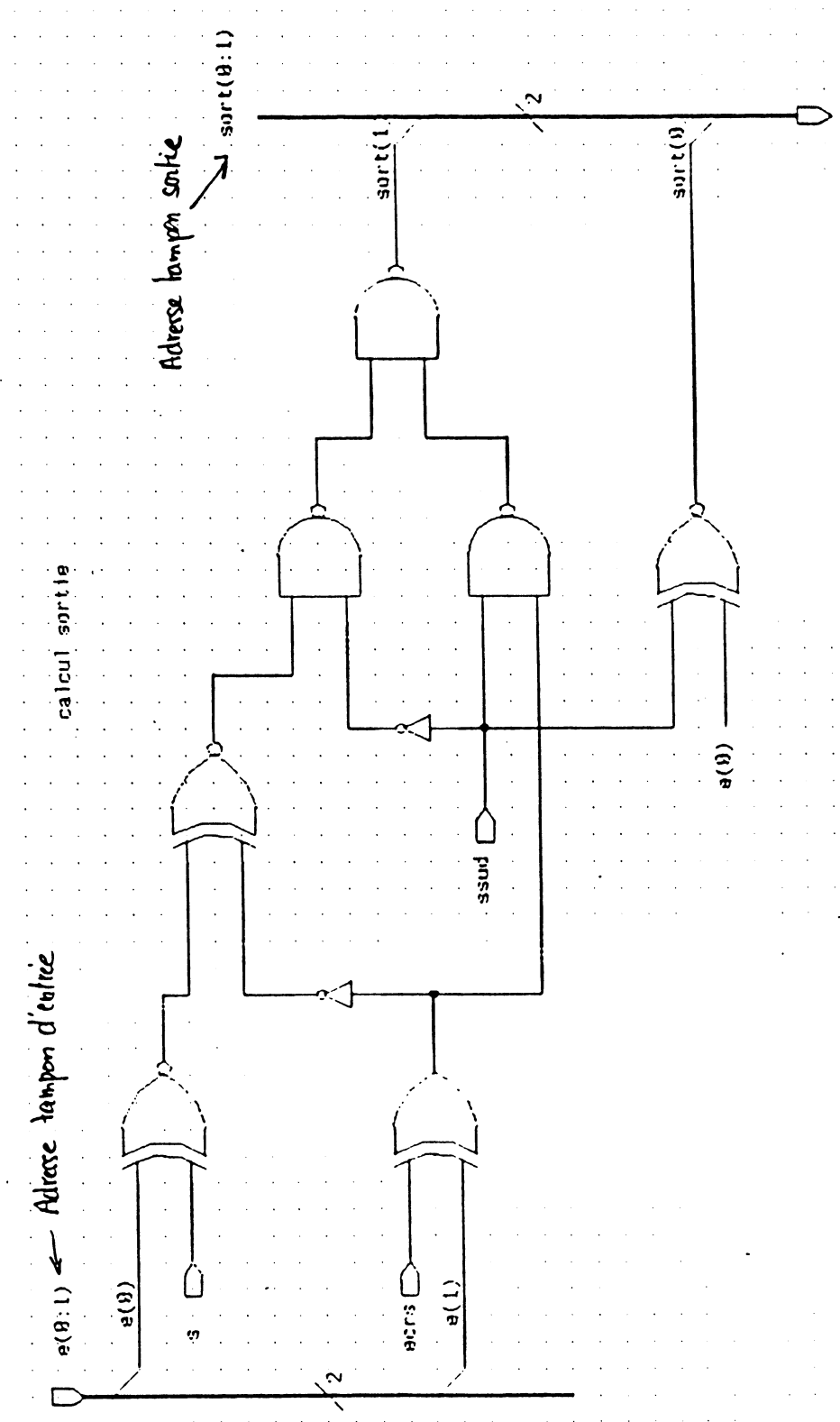
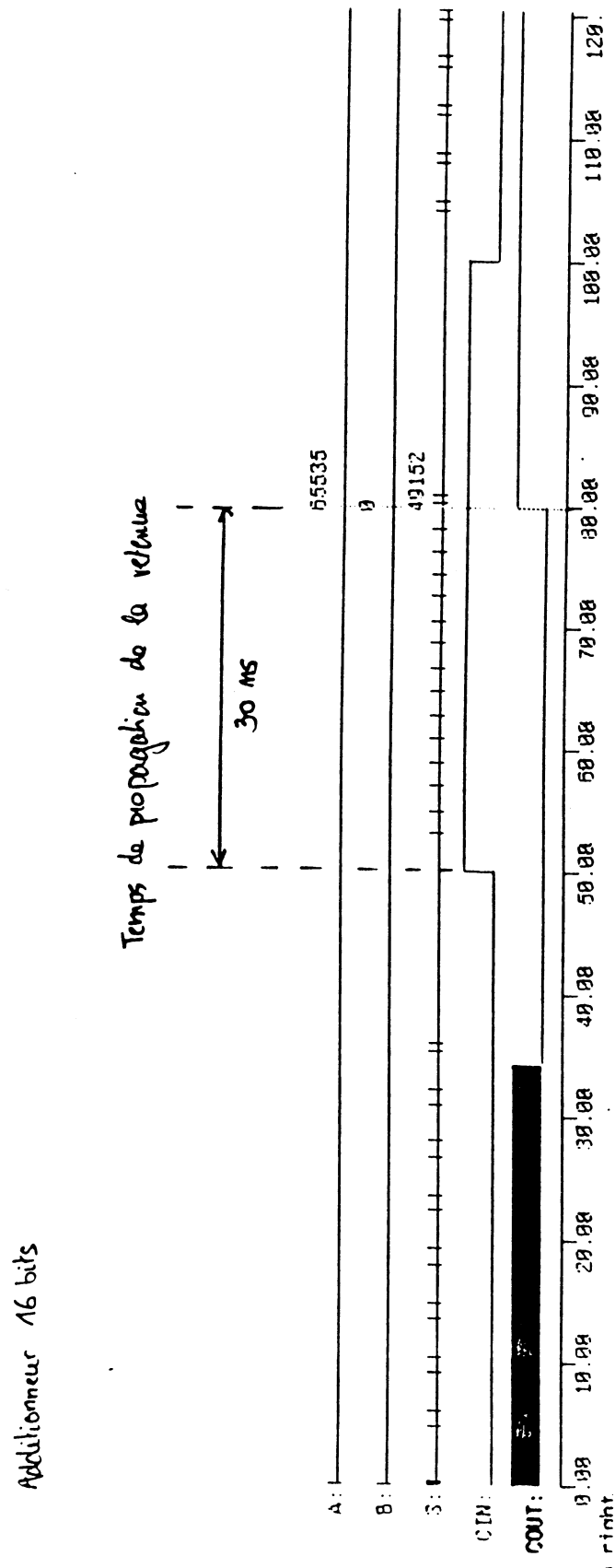


Schéma 6. Le bloc *Calcul Sortie* du module combinatoire





**Schéma 7. Résultat de la simulation de l' additionneur séquentiel**



**Schéma 8.** Résultat de la simulation de l'additionneur à retenue bondissante

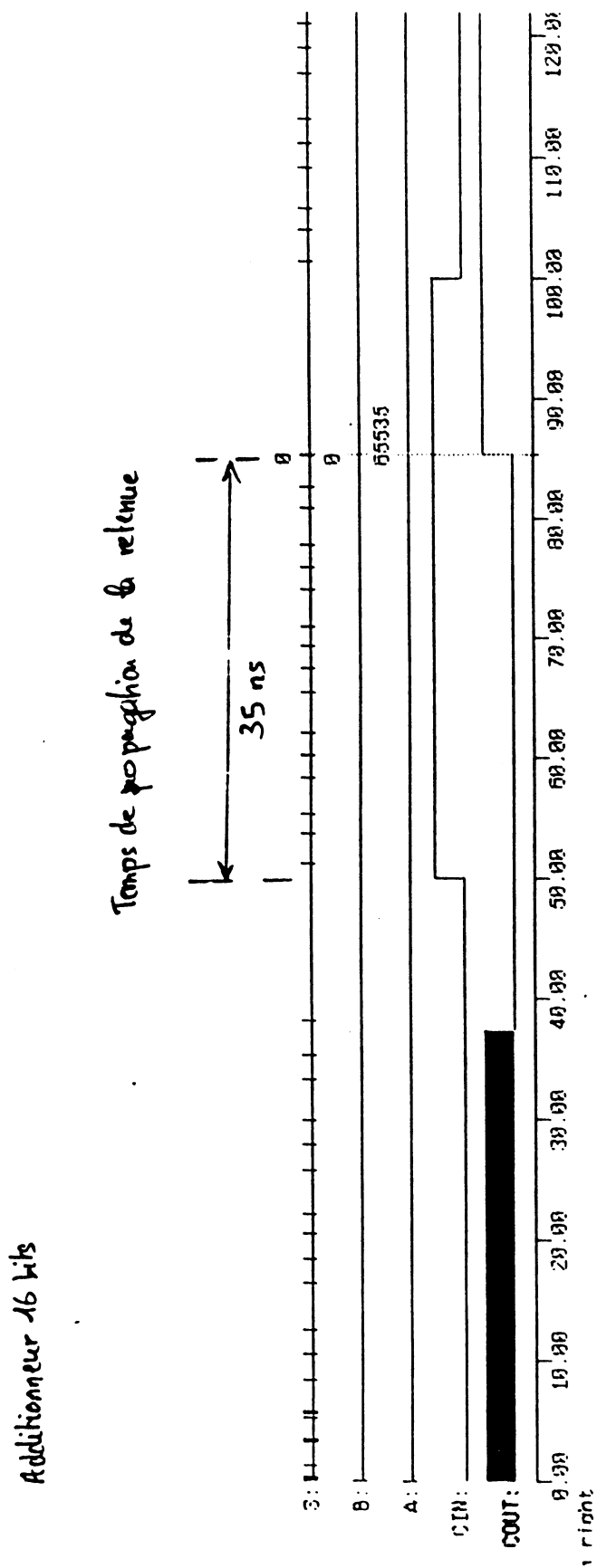


Schéma 9. Résultat de la simulation d'un tampon

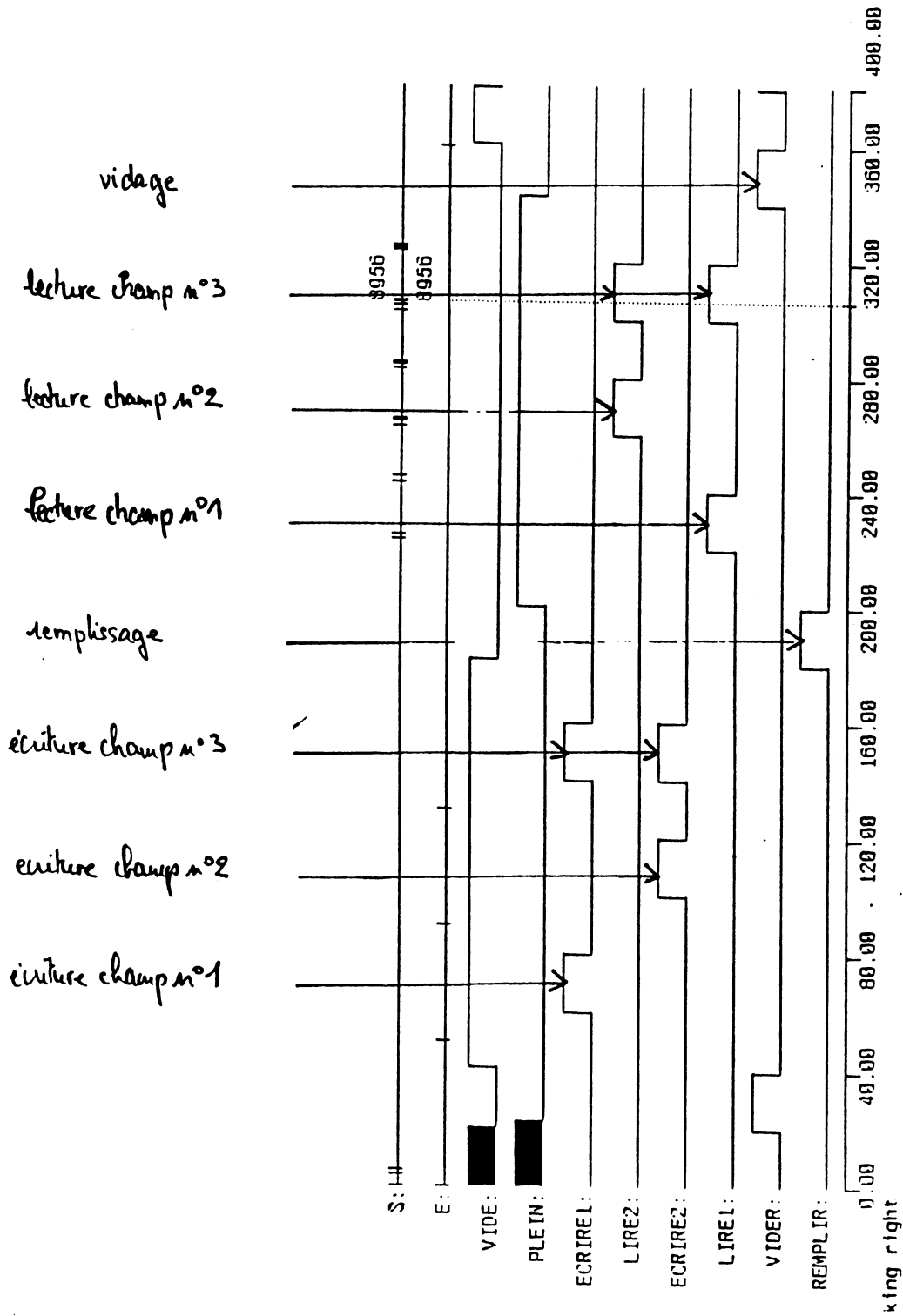


Schéma 10. Résultat de la simulation du circuit : traitement d'un message d'épandage

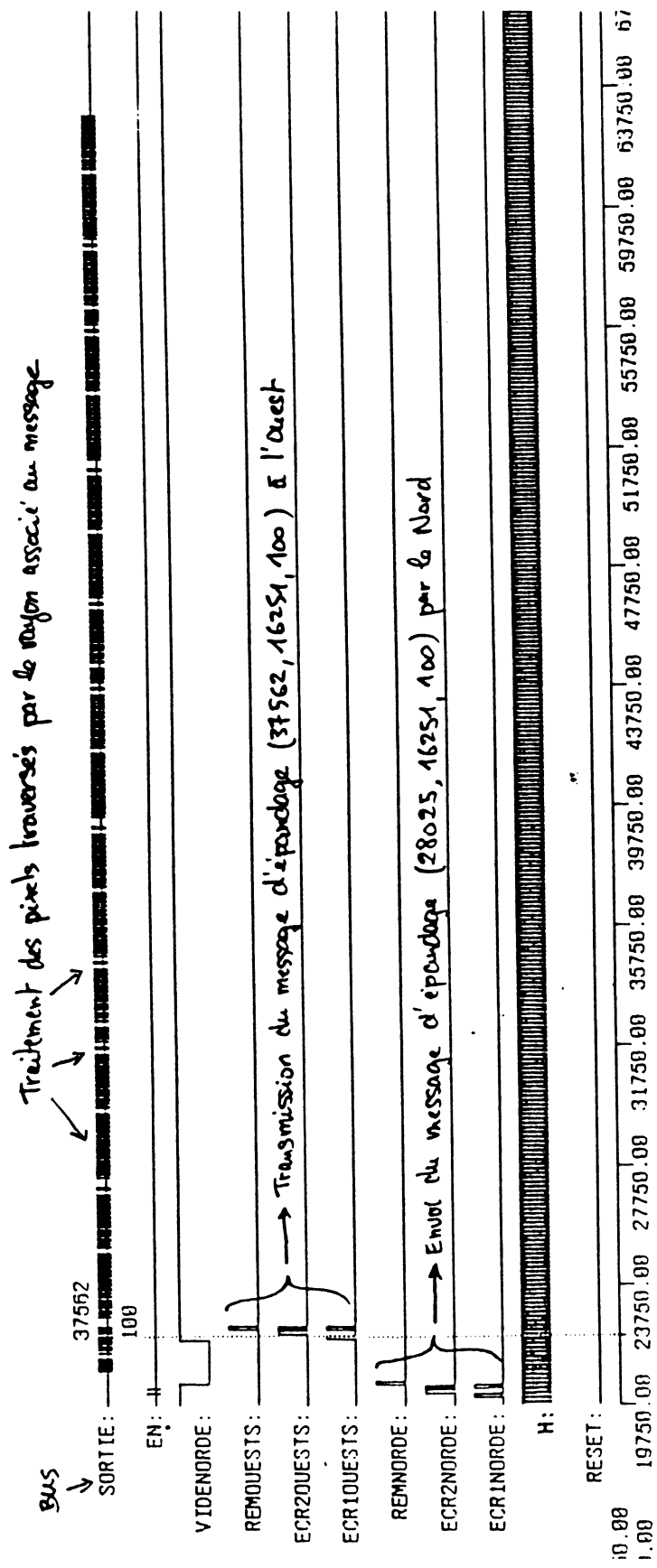
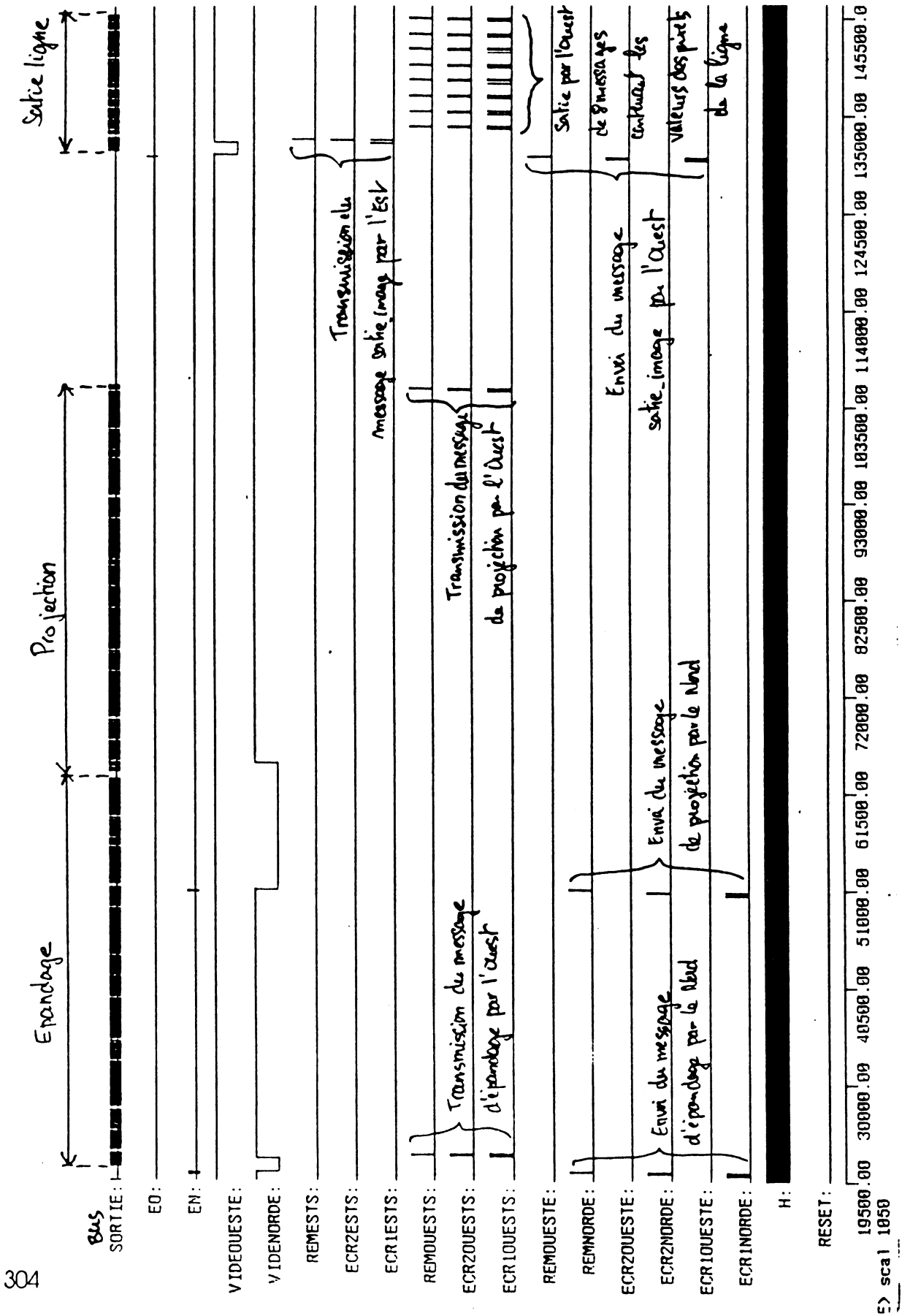


Schéma 11. Résultat de la simulation du circuit : traitement d'un message d'épandage, de projection, et sortie d'une ligne



## 4. DESSINS DES MASQUES

### 4.1. Bibliothèque SOLO1200

Schéma 1. Porte ou à 3 entrées

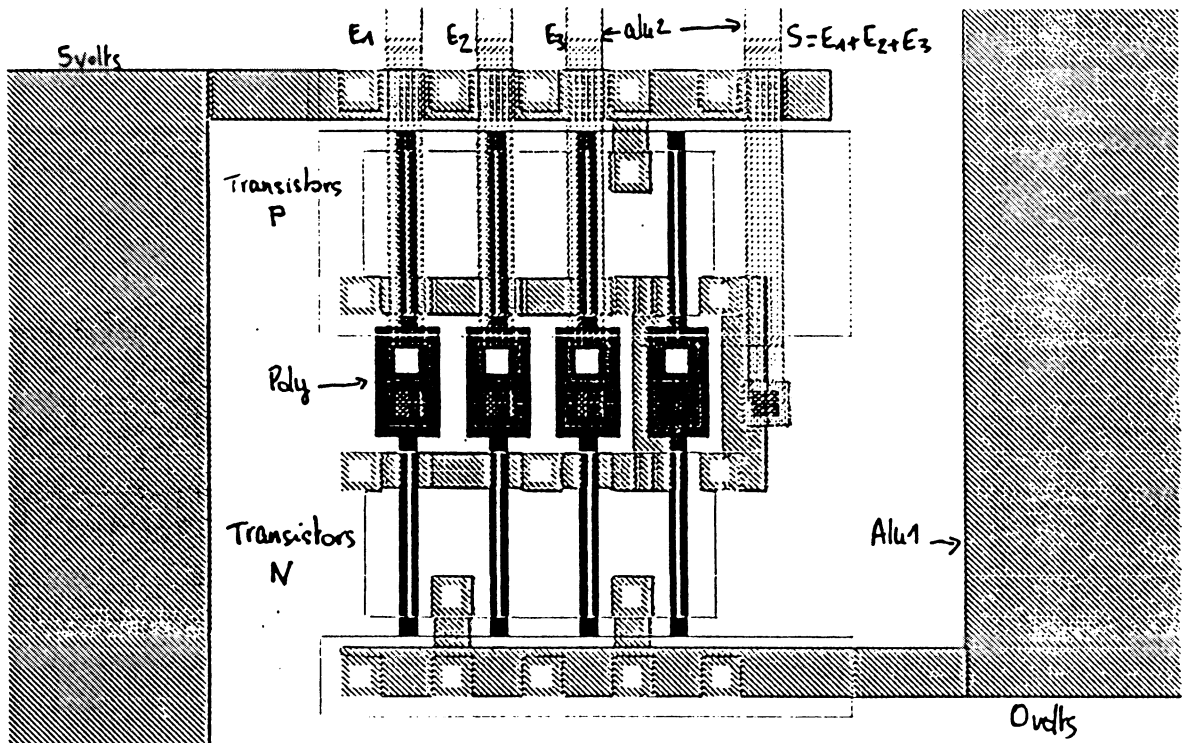


Schéma 2. Porte 3 états

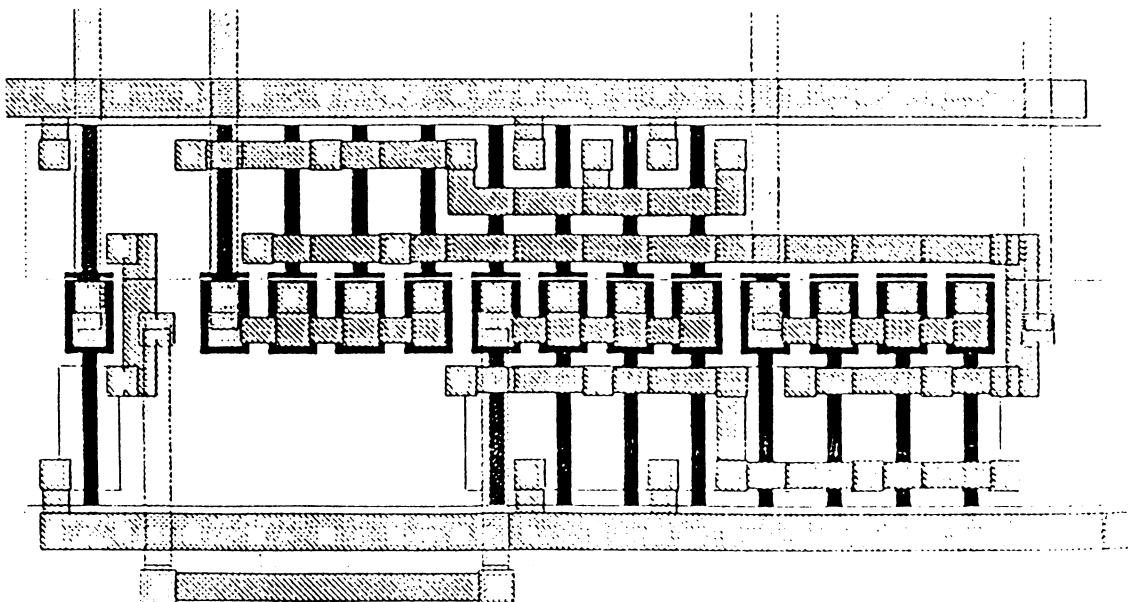


Schéma 3. Registre 1 bit

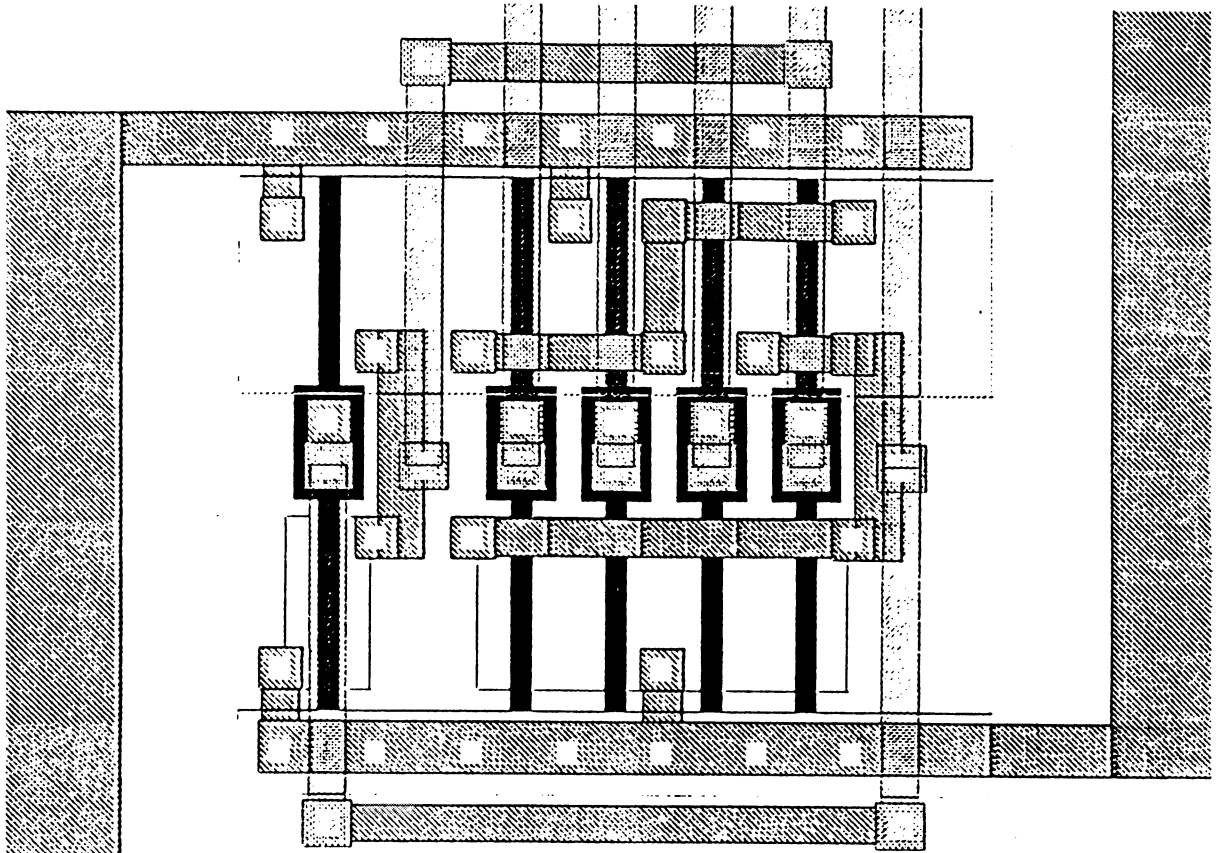
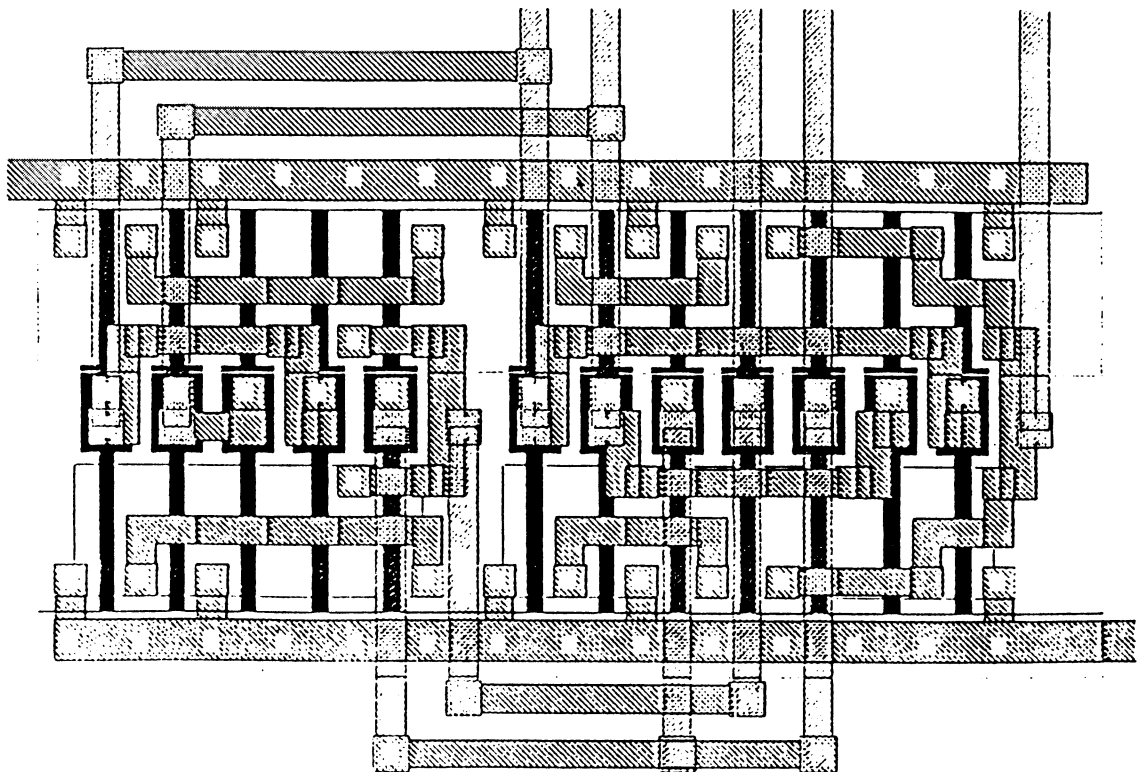


Schéma 4. Additionneur 1 bit



## 4.2. Modules du circuit

Schéma 5. Un tampon

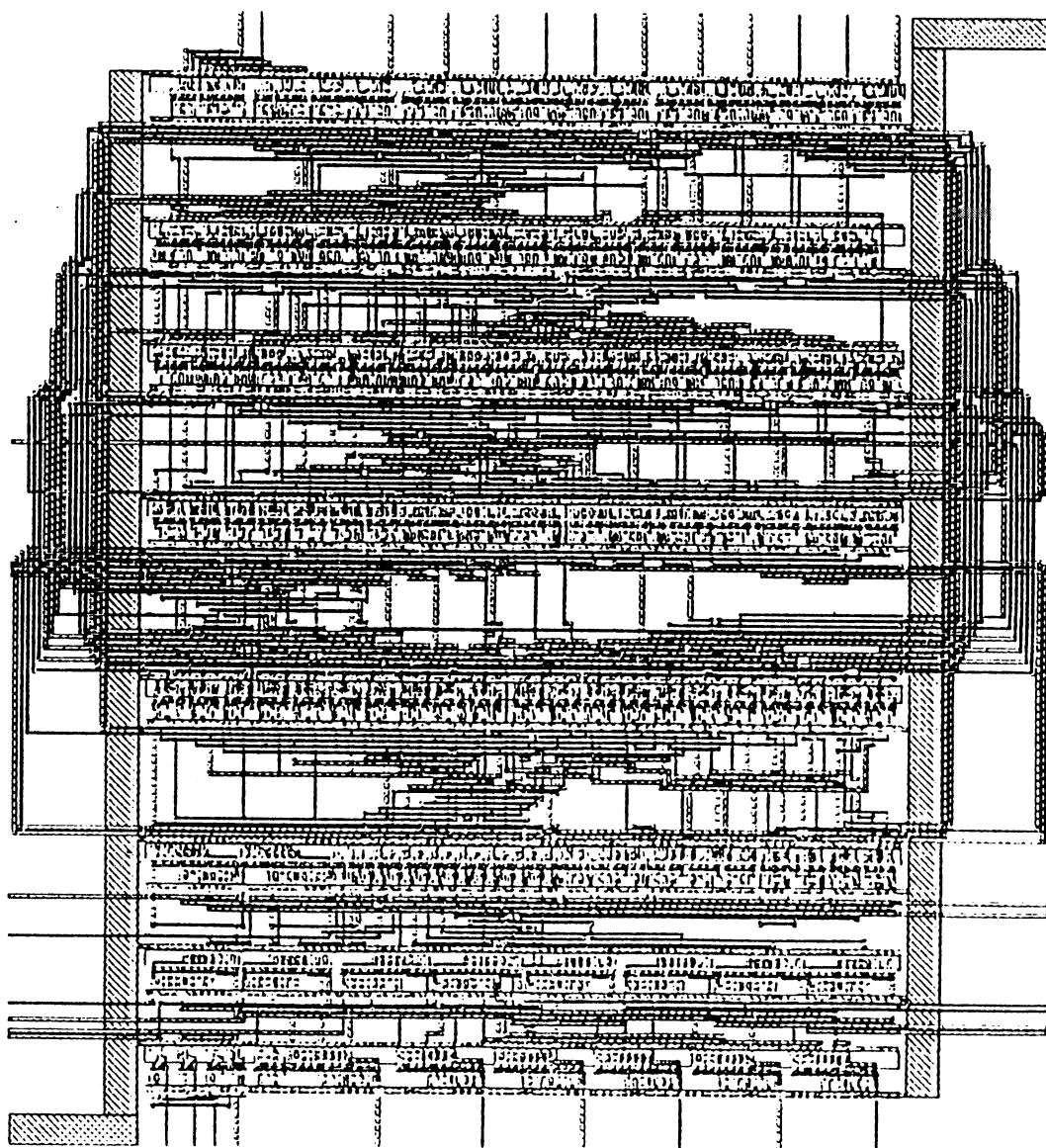
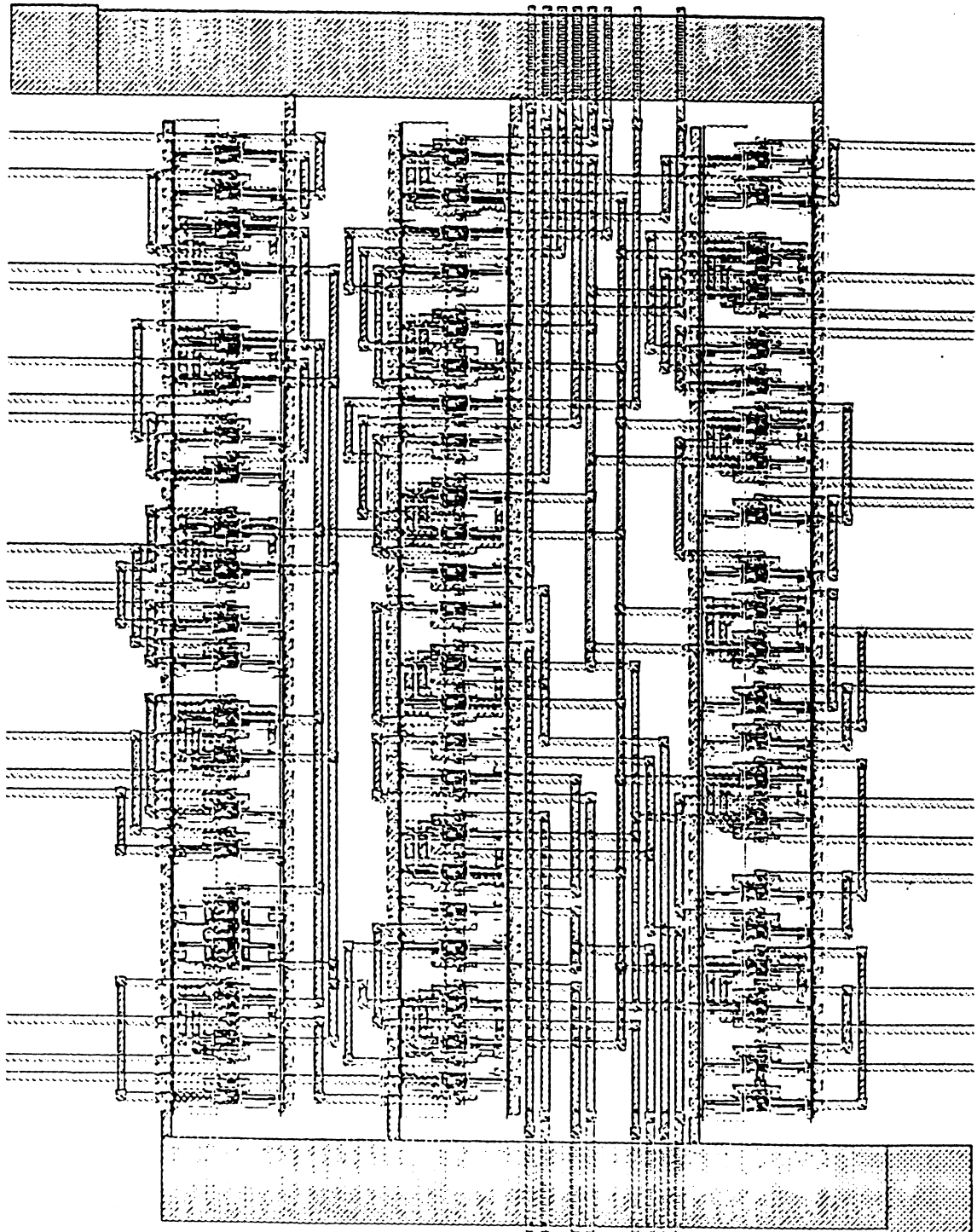
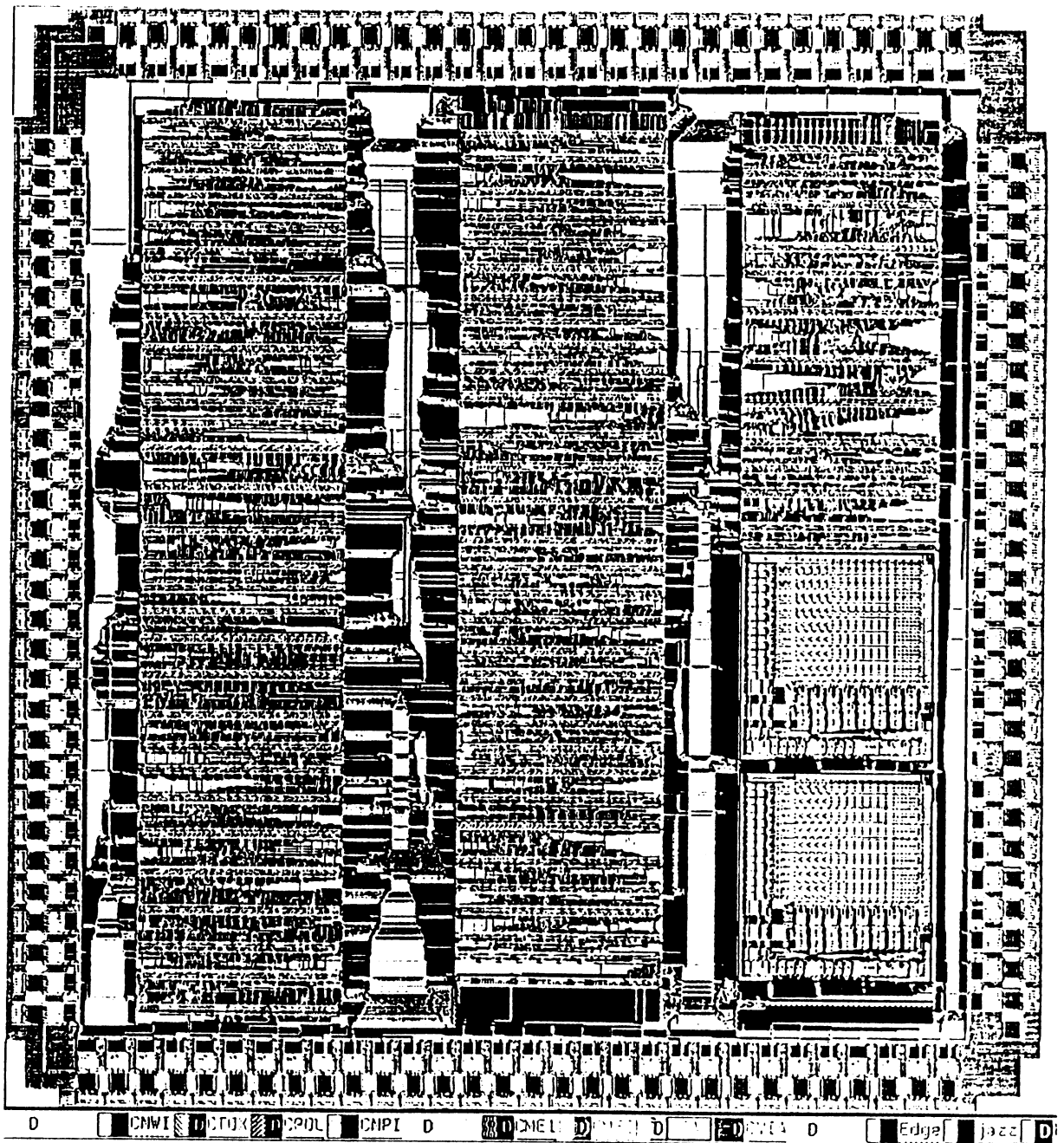




Schéma 6. Un registre de 16 bits

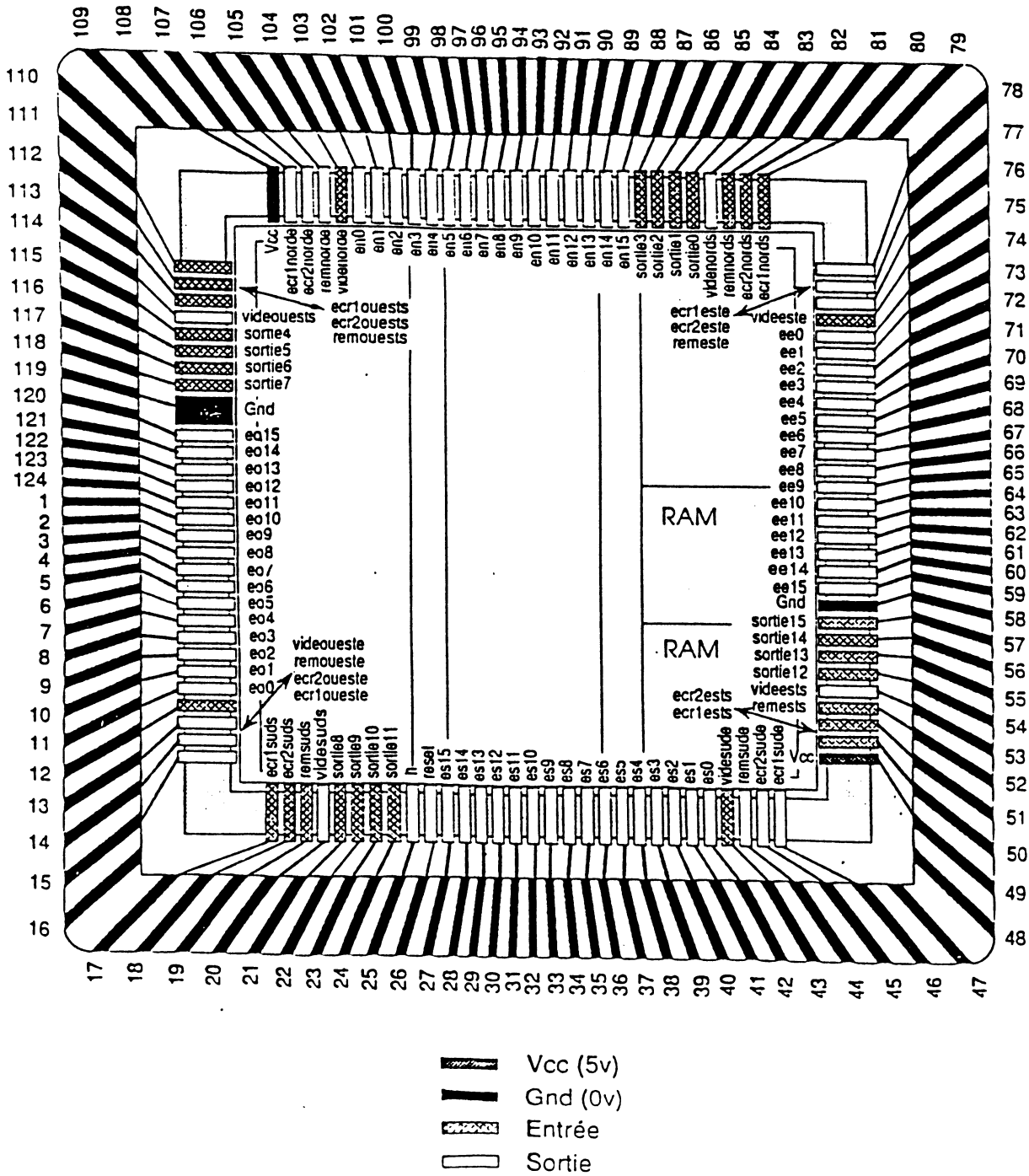


### 4.3. Circuit



5. PLAN DE MONTAGE

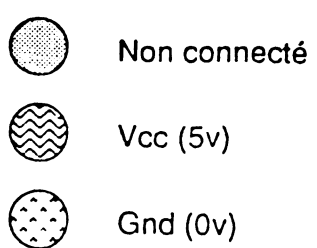
Schéma 1. Plan de montage



## Schéma 2. Liste des plots du circuit

| N° PAD | N° PIN | NOM       | N° PAD | N° PIN | NOM       |
|--------|--------|-----------|--------|--------|-----------|
| 1      | G-2    | eo10      | 64     | G-11   | ee10      |
| 2      | G-3    | eo9       | 65     | F-13   | ee9       |
| 3      | H-1    | eo8       | 66     | F-12   | ee8       |
| 4      | H-2    | eo7       | 67     | F-11   | ee7       |
| 5      | H-3    | eo6       | 68     | E-13   | ee6       |
| 6      | J-1    | eo5       | 69     | E-12   | ee5       |
| 7      | J-2    | eo4       | 70     | D-13   | ee4       |
| 8      | K-1    | eo3       | 71     | D-12   | ee3       |
| 9      | K-2    | eo2       | 72     | C-13   | ee2       |
| 10     | L-1    | eo1       | 73     | B-13   | ee1       |
| 11     | M-1    | eo0       | 74     | C-12   | ee0       |
| 12     | L-2    | videouest | 75     | A-13   | videeste  |
| 13     | N-1    | remouest  | 76     | B-12   | remeste   |
| 14     | M-2    | ecr2ouest | 77     | E-11   | ecr2este  |
| 15     | J-3    | ecr1ouest | 78     | D-11   | ecr1este  |
| 16     | K-3    | NC        | 79     | D-10   | NC        |
| 17     | K-4    | ecr1suds  | 80     | C-10   | ecr1nords |
| 18     | L-4    | ecr2suds  | 81     | C-11   | ecr2nords |
| 19     | L-3    | remsuds   | 82     | B-11   | remnords  |
| 20     | M-3    | videsuds  | 83     | A-12   | videnords |
| 21     | N-2    | sortie8   | 84     | A-11   | sortie0   |
| 22     | N-3    | sortie9   | 85     | B-10   | sortie1   |
| 23     | M-4    | sortie10  | 86     | A-10   | sortie2   |
| 24     | N-4    | sortie11  | 87     | B-9    | sortie3   |
| 25     | M-5    | h         | 88     | A-9    | en15      |
| 26     | N-5    | reset     | 89     | C-9    | en14      |
| 27     | L-5    | es15      | 90     | B-8    | en13      |
| 28     | M-6    | es14      | 91     | A-8    | en12      |
| 29     | N-6    | es13      | 92     | C-8    | en11      |
| 30     | L-6    | es12      | 93     | A-7    | en10      |
| 31     | N-7    | es11      | 94     | B-7    | en9       |
| 32     | M-7    | es10      | 95     | C-7    | en8       |
| 33     | L-7    | es9       | 96     | A-6    | en7       |
| 34     | N-8    | es8       | 97     | B-6    | en6       |
| 35     | M-8    | es7       | 98     | C-6    | en5       |
| 36     | L-8    | es6       | 99     | A-5    | en4       |
| 37     | N-9    | es5       | 100    | B-5    | en3       |
| 38     | M-9    | es4       | 101    | A-4    | en2       |
| 39     | N-10   | es3       | 102    | B-4    | en1       |
| 40     | M-10   | es2       | 103    | A-3    | en0       |
| 41     | N-11   | es1       | 104    | A-2    | videnorde |
| 42     | N-12   | es0       | 105    | B-3    | remnorde  |
| 43     | M-11   | videsude  | 106    | B-2    | ecr2norde |
| 44     | M-12   | remsude   | 107    | C-4    | ecr1norde |
| 45     | L-10   | ecr2sude  | 108    | C-5    | VCC       |
| 46     | L-9    | ecr1sude  | 109    | D-4    | NC        |
| 47     | K-10   | NC        | 110    | C-3    | NC        |
| 48     | L-11   | NC        | 111    | D-3    | ecr1ouest |
| 49     | K-11   | VCC       | 112    | A-1    | ecr2ouest |
| 50     | N-13   | ecr1ests  | 113    | C-2    | remouest  |
| 51     | L-12   | ecr2ests  | 114    | B-1    | videouest |
| 52     | M-13   | remests   | 115    | C-1    | sortie4   |
| 53     | L-13   | videests  | 116    | D-2    | sortie5   |
| 54     | K-12   | sortie12  | 117    | D-1    | sortie6   |
| 55     | K-13   | sortie13  | 118    | E-2    | sortie7   |
| 56     | J-12   | sortie14  | 119    | E-1    | GND       |
| 57     | J-13   | sortie15  | 120    | E-3    | eo15      |
| 58     | J-11   | GND       | 121    | F-2    | eo14      |
| 59     | H-12   | ee15      | 122    | F-1    | eo13      |
| 60     | H-13   | ee14      | 123    | F-3    | eo12      |
| 61     | H-11   | ee13      | 124    | G-1    | eo11      |
| 62     | G-13   | ee12      |        |        |           |
| 63     | G-12   | ee11      |        |        |           |

### Schéma 3. Boîtier







A U T O R I S A T I O N de S O U T E N A N C E

VU les dispositions de l'Arrêté du 23 novembre 1988 relatif aux Etudes doctorales

VU les rapports de présentation de

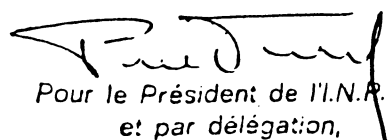
Monsieur BASILLE Jean-Luc

Monsieur SANSONNET Jean-Paul

Monsieur LATTARD Didier

est autorisé(e) à présenter une thèse en soutenance en vue de l'obtention du diplôme  
de DOCTEUR de L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE, spécialité  
"Microélectronique"

Fait à Grenoble, le 18 Octobre 1989

  
Pour le Président de l'I.N.P.-G.  
et par délégation,  
le Vice-Président  
P. VENNEREAU







## **Résumé**

Depuis quelques années, l'intérêt pour des problèmes de grande complexité tels que le traitement du signal et de l'image, et la limitation de puissance due à la séquentialité des opérations des machines basées sur le modèle de Von Neumann, ont poussé les informaticiens à étudier un nouveau modèle de traitement caractérisé par l'exécution parallèle des opérations. Nous proposons dans cette thèse une architecture massivement parallèle basée sur un réseau régulier de cellules, qui ont la particularité d'être totalement asynchrones et de pouvoir communiquer entre elles grâce à un mécanisme d'acheminement de messages. Chaque cellule comprend une partie de traitement élémentaire réalisant les fonctions nécessitées par l'application et une partie routage permettant d'acheminer les messages. Nous avons validé cette architecture en développant une machine complète dédiée à la reconstruction d'images.

Pour cette application particulière, chaque cellule traite une sous-image, et le réseau est utilisé pour réaliser des opérations d'épandage et de projection, qui sont essentielles dans les principales méthodes de reconstruction. Les différents paramètres de la cellule sont définis en fonction des contraintes temporelles, de manière à obtenir d'excellentes performances et une bonne activité globale du réseau. L'interface dans un environnement hôte d'une telle machine a été étudiée. Nous avons réalisé un circuit intégré comprenant une cellule, afin de développer une maquette de démonstration.

## **Mots-clés**

Architecture parallèle, Réseau cellulaire, Communication, Circuit intégré, Reconstruction d'images, Epandage, Projection.