Étude d'une machine cellulaire pour la simulation logique de circuits intégrés

Jean-Pierre Bernard

THESE

présentée à

l' Université Scientifique et Médicale de Grenoble

pour obtenir le grade de
DOCTEUR INGENIEUR
«Microelectronique»

par

Jean-Pierre BERNARD

ETUDE D'UNE MACHINE CELLULAIRE POUR LA SIMULATION LOGIQUE DE CIRCUITS INTEGRES.

Thèse soutenue le 3 juillet 1985 devant la commission d'examen.

A. DENEUVILLE	Président
C. LANDRAULT	
J. LECOURVOISIER	Examinateurs
G. MAZARÉ	

UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

Année universitaire 1982-1983

Président de l'Université : M. TANCHE

MEMBRES DU CORPS ENSEIGNANT DE L'U.S.M.G.

(RANG A)

SAUF ENSEIGNANTS EN MEDECINE ET PHARMACIE

PROFESSEURS DE 1ère CLASSE

ARNAUD Paul	Chimie organique
ARVIEU Robert	Physique nucléaire I.S.N.
AUBERT Guy	Physique C.N.R.S.
AYANT Yves	Physique approfondie
BARBIER Marie-Jeanne	Electrochimie
BARBIER Jean-Claude	Physique expérimentale C.N.R.S. (labo de magnétisme)
BARJON Robert	Physique nucléaire I.S.N.
BARNOUD Fernand	Biosynthèse de la cellulose-Biologie
BARRA Jean-René	Statistiques - Mathématiques appliquées
BELORISKY Elie	Physique
BENZAKEN Claude (M.)	Mathématiques pures
BERNARD Alain	Mathématiques pures
BERTRANDIAS Françoise	Mathématiques pures
BERTRANDIAS Jean-Paul	Mathématiques pures
BILLET Jean	Géographie
BONNIER Jean-Marie	Chimie générale
BOUCHEZ Robert	Physique nucléaire I.S.N.
BRAVARD Yves	Géographie
CARLIER Georges	Biologie végétale
CAUQUIS Georges	Chimie organique
CHIBON Pierre	Biologie animale
COLIN DE VERDIERE Yves	Mathématiques pures
CRABBE Pierre (détaché)	C.E.R.M.O.
CYROT Michel	Physique du solide
DAUMAS Max	Géographie
DEBELMAS Jacques	Géologie générale
DEGRANGE Charles	Zoologie
DELOBEL Claude (M.)	M.I.A.G. Mathématiques appliquées
DEPORTES Charles	Chimie minérale
DESRE Pierre	Electrochimie
DOLIQUE Jean-Michel	Physique des plasmas
DUCROS Pierre	Cristallographie
FONTAINE Jean-Marc	Mathématiques pures
GAGNAIRE Didier	Chimie physique

.../...

GASTINEL Noël	Analyse numérique - Mathématiques appliquées
GERBER Robert	Mathématiques pures
GERMAIN Jean-Pierre	Mécanique
GIRAUD Pierre	Géologie
IDELMAN Simon	Physiologie animale
JANIN Bernard	Géographie
JOLY Jean-René	Mathématiques pures
JULLIEN Pierre	Mathématiques appliquées
KAHANE André (détaché DAFCO)	Physique
KAHANE Josette	Physique
KOSZUL Jean-Louis	Mathématiques pures
KRAKOWIAK Sacha	Mathématiques appliquées
KUPTA Yvon	Mathématiques pures
LACAZE Albert	Thermodynamique
LAJZEROWICZ Jeannine	Physique
LAJZEROWICZ Joseph	Physique
LAURENT Pierre	Mathématiques appliquées
DE LEIRIS Joël	Biologie
LLIBOUTRY Louis	Géophysique
LOISEAUX Jean-Marie	Sciences nucléaires I.S.N.
LOUP Jean	Géographie
MACHE Régis	Physiologie végétale
MAYNARD Roger	Physique du solide
MICHEL Robert	Minéralogie et pétrographie (géologie)
MOZIERES Philippe	Spectrométrie - Physique
OMONT Alain	Astrophysique
OZENDA Paul	Botanique (biologie végétale)
PAYAN Jean-Jacques (détaché)	Mathématiques pures
PEBAY PEYROULA Jean-Claude	Physique
PERRIAUX Jacques	Géologie
PERRIER Guy	Géophysique
PIERRARD Jean-Marie	Mécanique
RASSAT André	Chimie systématique
RENARD Michel	Thermodynamique
RICHARD Lucien	Biologie végétale
RINAUDO Marguerite	Chimie CERMAV
SENGEL Philippe	Biologie animale
SERGERAERT Francis	Mathématiques pures
SOUTIF Michel	Physique
VAILLANT François	Zoologie
VALENTIN Jacques	Physique nucléaire I.S.N.
VAN CUTSEN Bernard	Mathématiques appliquées
VAUQUOIS Bernard	Mathématiques appliquées
VIALON Pierre	Géologie

PROFESSEURS DE 2ème CLASSE

ADIBA Michel	Mathématiques pures
ARMAND Gilbert	Géographie

.../...

AURIAULT Jean-Louis	Mécanique
BEGUIN Claude (M.)	Chimie organique
BOEHLER Jean-Paul	Mécanique
BOITET Christian	Mathématiques appliquées
BORNAREL Jean	Physique
BRUN Gilbert	Biologie
CASTAING Bernard	Physique
CHARDON Michel	Géographie
COHENADDAD Jean-Pierre	Physique
DENEUVILLE Alain	Physique
DEPASSEL Roger	Mécanique des fluides
DOUCE Roland	Physiologie végétale
DUFRESNOY Alain	Mathématiques pures
GASPARD François	Physique
GAUTRON René	Chimie
GIDON Maurice	Géologie
GIGNOUX Claude (M.)	Sciences nucléaires I.S.N.
GUITTON Jacques	Chimie
HACQUES Gérard	Mathématiques appliquées
HERBIN Jacky	Géographie
HICTER Pierre	Chimie
JOSELEAU Jean-Paul	Biochimie
KERCKOVE Claude (M.)	Géologie
LE BRETON Alain	Mathématiques appliquées
LONGEQUEUE Nicole	Sciences nucléaires I.S.N.
LUCAS Robert	Physiques
LUNA Domingo	Mathématiques pures
MASCLE Georges	Géologie
NEMOZ Alain	Thermodynamique (CNRS - CRTBT)
OUDET Bruno	Mathématiques appliquées
PELMONT Jean	Biochimie
PERRIN Claude (M.)	Sciences nucléaires I.S.N.
PFISTER Jean-Claude (détaché)	Physique du solide
PIBOULE Michel	Géologie
PIERRE Jean-Louis	Chimie organique
RAYNAUD Hervé	Mathématiques appliquées
ROBERT Gilles	Mathématiques pures
ROBERT Jean-Bernard	Chimie physique
ROSSI André	Physiologie végétale
SAKAROVITCH Michel	Mathématiques appliquées
SARROT REYNAUD Jean	Géologie
SAXOD Raymond	Biologie animale
SOUTIF Jeanne	Physique
SCHOOL Pierre-Claude	Mathématiques appliquées
STUTZ Pierre	Mécanique
SUBRA Robert	Chimie
VIDAL Michel	Chimie organique
VIVIAN Robert	Géographie

A MA MERE

Je tiens à remercier :

-Monsieur Alain DENEUVILLE, professeur à l'USMG, de l'honneur qu'il m'a fait en acceptant de présider le jury de cette thèse.

-Messieurs Christian LANDRAULT, chargé de recherche au CNRS, et Jacques LECOURVOISIER, chef de département au CNET CNS, d'avoir accepté de participer au jury de cette thèse.

-Monsieur Guy MAZARE, professeur à l'ENSIMAG, de m'avoir accueilli dans son équipe et d'avoir assuré l'encadrement de cette thèse.

INTRODUCTION

INTRODUCTION

Le développement important des possibilités d'intégration dans le monde de l'électronique rend indispensable le recours à des outils mathématiques et informatiques dans la phase de conception des circuits intégrés.

Vu la complexité de cette phase de conception et les moyens mis en oeuvre pour la fabrication de ces circuits intégrés, il est nécessaire que la conception soit aussi sûre que possible avant de passer à l'étape de fabrication. Il est donc apparu des outils de vérification, les simulateurs logiques, permettant une aide appréciable pour la conception de gros circuits logiques.

Ces outils ont d'abord été développés sous forme logiciel. Mais la complexité des circuits à haute densité d'intégration progressant sans cesse, les simulateurs logiques traditionnels ne s'avèrent quelquefois plus assez performants. des machines spécialisées de simulation logique ont alors été développées et permettent actuellement des vitesses bien supérieures à celles proposées par les simulateurs logiciels.

Notre idée est de mettre à profit le développement des circuits intégrés V.L.S.I. pour étudier la conception d'un circuit spécialisé d'aide à la simulation logique.

La première partie de cette thèse présente la simulation logique traditionnelle, les machines spécialisées existantes pour la simulation logique, et une nouvelle classe de machines à base de circuits V.L.S.I., les machines cellulaires.

La deuxième partie expose notre proposition d'un réseau cellulaire adapté à la simulation logique et détaille la conception d'une cellule élémentaire.

La partie III étudie les possibilités d'utilisation de notre réseau en tant que machine spécialisée de simulation logique. Une estimation des performances attendues est exposée dans chaque architecture possible.

Enfin, la conclusion indique quelles sont les solutions facilement réalisables, et donne des exemples d'autres utilisations possibles d'un réseau cellulaire basé sur le même principe.

P R E M I E R E P A R T I E

##########################

SIMULATION LOGIQUE

****

ET

**

MACHINES SPECIALISEES

****

CHAPITRE I

LA SIMULATION LOGIQUE

I. 1. INTRODUCTION

Durant la phase de conception d'un circuit électronique, la conception doit déterminer les caractéristiques des signaux électriques se propageant dans celui-ci, afin d'en vérifier la cohérence et l'exactitude. Pour des circuits contenant un petit nombre d'éléments, le concepteur construit généralement une maquette. Il analyse ensuite expéritalement le circuit et ajuste les différents composants.

Cette approche traditionnelle devient difficile, voire impossible pour des circuits complexes. De plus, une maquette de composants discrets peut très mal modèliser un circuit L.S.I. ou V.L.S.I., les caractéristiques électriques et physiques pouvant être très différentes entre le circuit intégré et la maquette.

Les concepteurs se sont alors tournés vers des outils mathématiques et informatiques permettant des vérifications plus faciles et donc une conception plus fiable et plus rapide. Ces outils, les simulateurs permettent de simuler de manière précise le comportement des circuits.

Un simulateur essaie de prédire le comportement d'un sys-

tème en fonction d'un ensemble de signaux d'entrée. Il a besoin pour cela d'une représentation appelée modèle du système. La modèlisation est un problème important, car elle conditionne la précision et la vitesse d'obtention des résultats.
Un modèle doit être suffisamment précis pour garantir la validité des résultats, mais plus la précision est grande plus le temps d'obtention des résultats sera long. Un compromis doit donc être trouvé .

Les modèles peuvent être des ensembles d'équations différentielles comme des définitions empiriques de phénomènes physiques, tout dépend du niveau de détail désiré par l'utilisateur et de sa connaissance du phénomène physique.

La conception des circuits électroniques se fait généralement de manière hiérarchique. Un circuit est décrit au niveau le plus haut par un ou plusieurs blocs fonctionnels. Chaque bloc est ensuite défini en sous-blocs plus complexes jusqu'à ce que le niveau de détail désiré soit atteint.
Il apparait donc par cette hierarchie dans la description, des niveaux de détail différents, nécessitant un ou plusieurs modèles propres à chaque niveau.
Selon le ou les modèles possédés par le simulateur, on parlera de simulateur à tel niveau ou de simulateur multiniveau.

Le domaine qui nous intéresse, la simulation logique, nécessite la précision de certains termes.

La simulation logique permet de simuler des circuits logiques, intégrés ou non. Les signaux véhiculés dans ces circuits sont pour l'utilisateur des signaux binaires codant la tension des signaux électriques (puis par la suite les impédances des éléments électriques, voir plus loin).

Les niveaux hiérarchiques de la description des circuits logiques introduisent autant de niveaux de simulation.

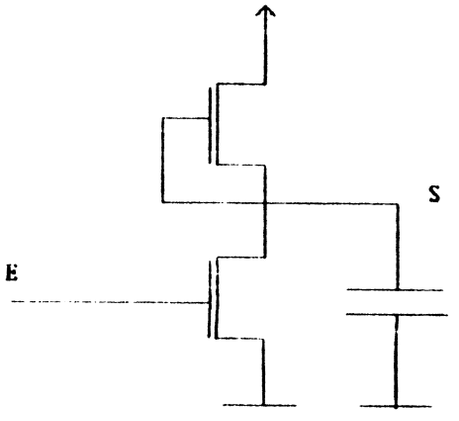
De manière simplifiée, les niveaux de description sont les suivants :

- niveau fonctionnel
- niveau registre
- niveau porte logique
- niveau électrique

Au niveau électrique, la description se fait en terme d'éléments électriques actifs ou passifs, connectés entre eux par des noeuds.
Les signaux sont les courants dans les éléments et les tensions aux noeuds du circuit. Ils sont donc sous forme analogique.

Les modèles sont les représentations des éléments électriques sous forme d'équations en tension et en courant.

Exemple :

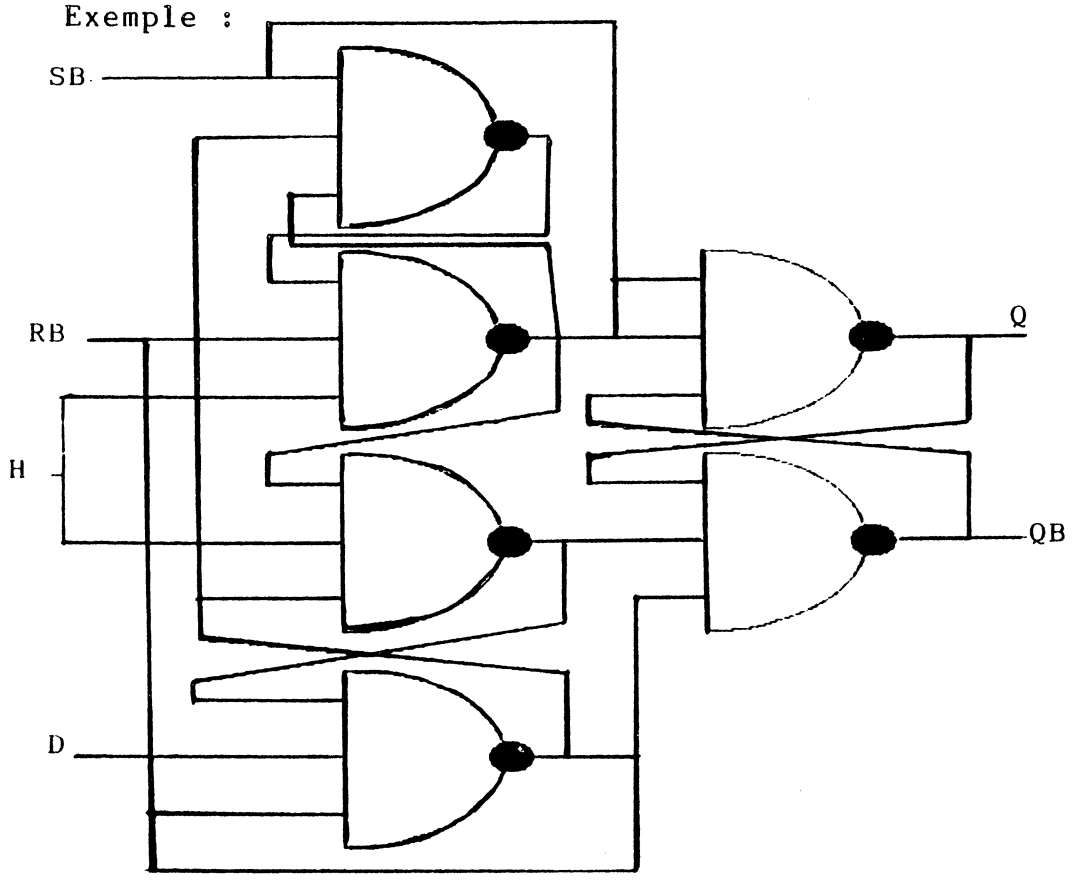


inverseur NMOS, 2 transistors, 1 capacité, 4 noeuds.

Au niveau porte logique, la description se fait en termes de portes logiques reliées entre elles par des équipotentielles. Les signaux logiques sont le codage des tensions électriques de ces équipotentielles.

Les modèles sont les tables de vérité des fonctions logiques utilisées dans la description du circuit.

Exemple :

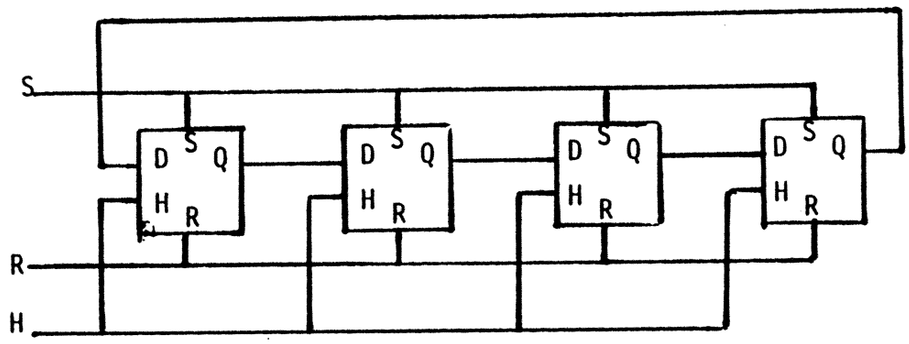


bascule D, 6 portes NAND à 3 entrées.

Au niveau registre, la description se fait en terme de blocs (registres) composés de plusieurs portes logiques, effectuant une fonction logique précise.

Les modèles sont les équations booléennes des registres.

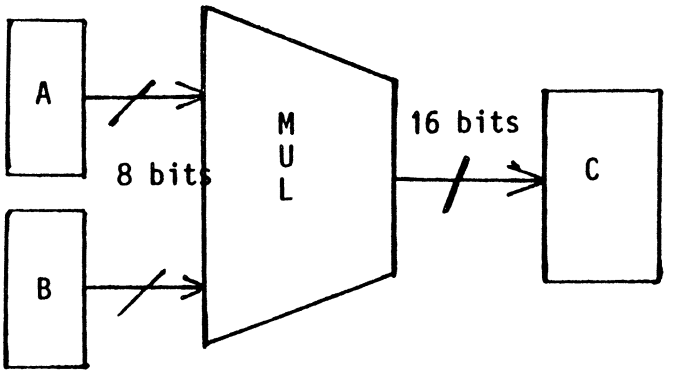
Exemple :



compteur en anneau, 4 bascules D.

Au niveau fonctionnel, la description se fait en terme de blocs fonctionnels s'échangeant des données qui sont des groupes de signaux et effectuant des opérations logiques complexes ou arithmétiques de blocs comme des compteurs des mémoires des unités arithmétiques et logiques .....

Exemple :



multiplieur.

Le domaine des outils informatiques dans la conception de circuit étant très récent, le vocabulaire utilisé n'est pas normalisé, et une certaine confusion règne sur la précision des termes employés.

Certains auteurs considèrent qu'un simulateur logique traite les niveaux (porte) logique et registre (ensemble de portes), et qu'un simulateur fonctionnel traite le niveau fonctionnel [WER 84].

D'autres considèrent que la simulation logique porte sur les 3 premiers niveaux, les signaux étant dans ces trois

cas sous forme logique . On parle alors de simulation de circuit logique [ACK 79] [BRE 72] [THO 75].

Le niveau électrique est traité par un simulateur électrique ou analogique .Les signaux analogiques sont les tensions et courants électriques du circuit.
Le terme analogique crèe une confusion, car un simulateur électrique ou analogique peut simuler des circuits logiques ou analogiques au niveau électrique.

Dans la conception de circuit logique, on peut distinguer 3 types de simulation logique permettant 3 types de vérification différente intervenant à des étapes différentes de la conception [BRE 72]:

- Une vérification de la synthèse logique permettant la définition complète de l'architecture au niveau de fonctions logiques employées ...
Cette simulation s'effectue avec des considérations temporelles très grossières.

- Une vérification temporelle s'effectuant à un niveau de détail plus fin (registre, porte).
Cette simulation doit permettre d'établir un fonctionnement temporel précis du circuit, de manière à détecter

les aléas de fonctionnement, les oscillations possibles et à y remédier .Elle permet de détailler complètement les blocs fonctionnels et logiques (dimensionnement des portes, ajustage des retards....).
Si la précision n'est pas suffisante, on utilise en plus des simulations électriques permettant une vérification à un niveau de détail plus fin (taille des transistors..).

- Une simulation de pannes s'effectuant une fois la conception figée. On étudie le comportement du circuit en présence de défauts.
Cette simulation sert à évaluer les vecteurs de tests permettant de détecter un certain nombre de pannes dans les circuits après fabrication .Ce type de simulation ne s'intègre pas dans la phase de conception, mais après, dans une phase de test.
La simulation de pannes est citée ici, car c'est généralement le même outil qui peut effectuer les vérifications de synthèse temporelles et évaluer les vecteurs de tests (EPISODE [EFC 82], TEGAS [THO 75]).

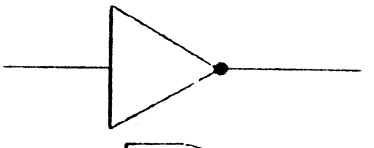
I. 2. MODELISATION

I. 2-1. Modèlisation des éléments logiques

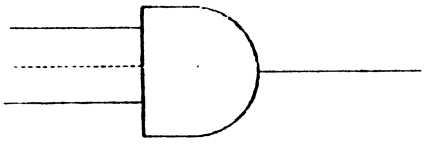
Un circuit logique est modèlisé comme un réseau d'élé-léments logiques interconnectés entre eux et véhiculant des signaux discrets. La description en portes logiques est la plus détaillée.

Les portes logiques effectuent les fonctions booléennes classiques. Les interconnexions véhiculent des états logiques codant la tension électrique des équipotentielles du circuit. Les portes logiques de base sont :

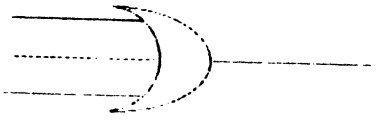
- L'inverseur



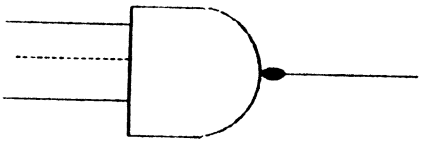
- Le ET



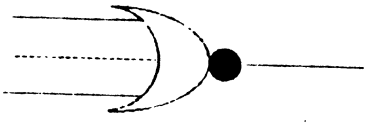
- Le OU



- Le NON-ET



- Le NON-OU



L'inverseur possède une entrée et une sortie, les autres portes plusieurs entrées et une seule sortie.
D'autres éléments logiques sont construits à partir de ces portes de base, et effectuent des fonctions logiques plus complexes.

Le nombre d'entrées d'une porte est son entrance.
Le nombre de portes qu'elle alimente en sortie est sa sortance.

La description externe des portes comprend donc sa fonction, son entrance, sa sortance.
Si l'on veut étudier le fonctionnement temporel d'un circuit, il est nécessaire d'avoir des informations sur la propagation des signaux à l'intérieur des portes.

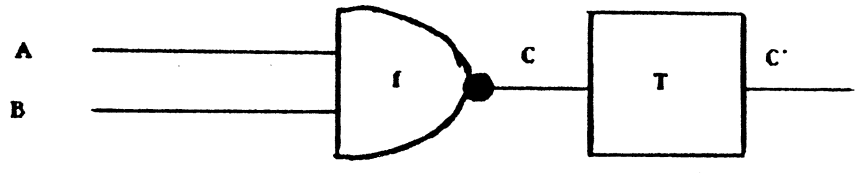
I. 2-2. Modèlisation des temps de propagation

Différents temps de propagation peuvent être utilisés.

- Retard nul : On néglige le temps de traversée. de la porte, la propagation est instantanée. On ne peut alors vérifier que la fonction logique de sortie du circuit après stabilisation.

- Retard unitaire : Toutes les portes ont le même temps de traversée, un pas d'horloge du simulateur. On peut avoir ainsi une idée du temps de stabilisation du circuit.

- Retard variable propre à chaque porte : Le temps de propagation est un multiple entier du pas d'horloge du simulateur.



$C(t) = f[a(t), b(t)]$

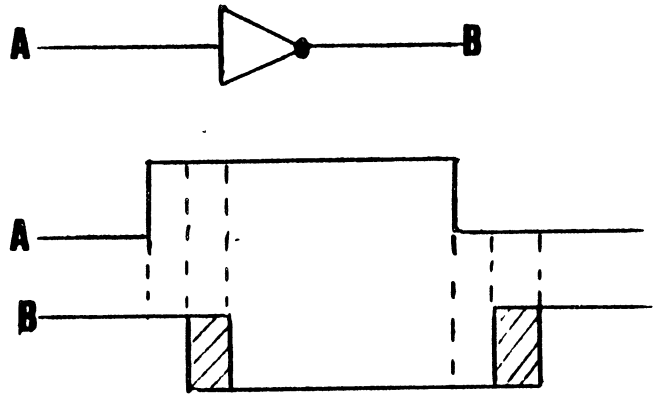
$C'(t+dt) = C(t)$

Dans certains simulateurs, ce retard est décrit comme un bloc logique à part entière, l'élément retard (EPISODE [EFC 82]).Au lieu de N portes, la description en contient alors 2 N.

- Retard ambigu : Lorsque le temps de propagation n'est pas exactement connu, on utilise une zone d'incertitude définie par un retard maximum et un retard minimum.

Exemple : - Retard maximum = 2

- Retard minimum = 1

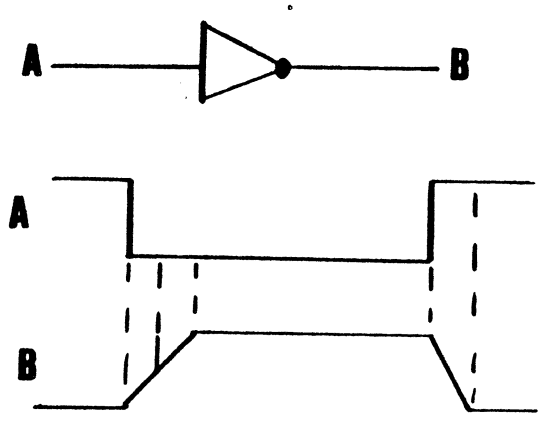


L'inverseur bascule en au minimum DT et au maximum 2 DT.

Ces zones d'incertitude servent à détecter des aléas de fonctionnement. Par exemple, 2 zones d'incertitude se recouvrant sur 2 entrées d'une porte indique un risque d'aléa.

- Temps de montée et de descente : Ils sont nécessaires pour observer les transitions lorsque la montée et la descente se font à des vitesses différentes.

Exemple :



- Retard inertiel D I : C'est le temps minimum pendant lequel le signal d'entrée doit être maintenu pour obtenir un changement de valeur en sortie. Son utilisation a un effet de filtre haute-fréquence , puisqu'il sert à éliminer tous les signaux de durée inférieure à DI.

Selon la précision de l'analyse temporelle et les états logiques disponibles sur les simulateurs, un ou plusieurs types de ces retards sont employés.

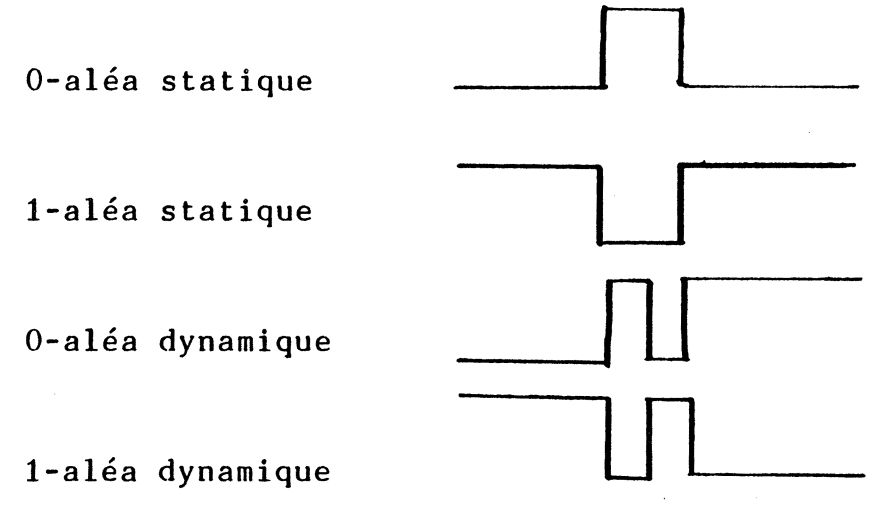
I. 2-3. Modèlisation des aléas

Un aléa statique est un changement bref de la valeur d'un signal qui devrait rester stable.

On distingue le 0-aléa, bref passage à 1 d'un signal qui devrait être et rester à 0, vice-versa, le 1-aléa.

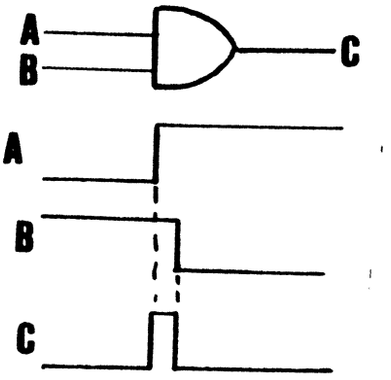
Un aléa dynamique est un bref retour à sa valeur initiale d'un signal changeant de valeur.
On distingue le 1-aléa dynamique (Séquence 0101 au lieu de 01) et le 0-aléa dynamique (Séquence 1010 au lieu de 10).

Chronogrammes des aléas :



Un exemple de production d'aléas est le changement quasi-simultané des différents signaux d'entrée d'une porte logique.

Exemple :



Si dans le fonctionnement prévu, A et B doivent changer de valeur en même temps et que A change légèrement avant B, un 0-aléa statique se produit sur C et peut induire un mauvais fonctionnement en aval du circuit.

I. 2-4. Modèlisation des signaux

Les signaux sont représentés par états logiques codant la tension ou la variation de tension d'une équipotentielle. Ces informations logiques peuvent prendre plusieurs valeurs.

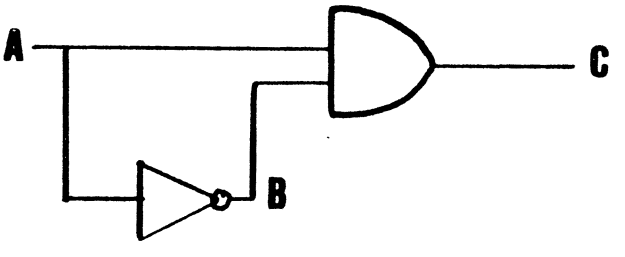
A l'origine on utilisait 2 états, haut (0) et bas (1). Ces deux états ne permettent d'étudier qu'un fonctionne-

ment statique du circuit (vérification de la fonction logique seulement, sans aucune considération temporelle). Pour obtenir une plus grande précision, de nouveaux états ont été ajoutés.

X est un état non-initialisé ou inconnu .Au début de la simulation, les portes peuvent être initialisées à X par l'utilisateur, on ne sait pas si elles sont à 0 ou 1.
La simulation avec 0, 1, X, comme états logiques est la simulation classique trois états qui est la plus employée, mais la moins précise (aucune indication temporelle).

L'emploi du X a un inconvénient majeur, c'est qu'il n'a pas d'inverse. Il représente donc un cas très pessimiste de fonctionnement du circuit.

Exemples :



Si A = X alors B = X et C = X
or quels que soient A et B, C est toujours à 0 mais le simulateur détecte X.
Un flip-flop aura le même état X sur ses deux sorties Q et $\overline{Q}$ à l'initialisation. Si les deux

sorties sont utilisées dans le reste du circuit, on aura en aval une initialisation pessimiste.

L'intérêt du X est de déterminer la stabilisation ou non du circuit .Toutefois ,une non stabilisation ou une stabilisation longue peuvent être détectées dans un circuit, être dûes à l'emploi intempestif du X et ne pas exister dans le circuit réel.
Avec l'emploi d'éléments spéciaux comme les portes de transferts ou les interrupteurs, le X sur la commande d'un tel élément pose d'énormes problèmes.
On peut même arriver à des blocages en X [BRY 80].

Certains simulateurs introduisent plusieurs états Xi et leur inverse $\overline{Xi}$, permettant d'étudier avec précision la stabilisation à l'initialisation (LOGMOS [DEM 82]).

Des états dynamiques indiquent les transitions entre les états statiques. Les principaux sont :

- La Transition montante, 0 vers 1: U (up)
- La Transition descendante, 1 vers 0: D(down)
- La Transition indéterminée : ϕ

BREUER a introduit des états supplémentaires codant les aléas [BRE 72]:

00* code le 0-aléa statique

11* code le 1-aléa statique

01* code le 0-aléa dynamique

10* code le 1-aléa dynamique

I. 2-5. Exemples de logiques utilisées

Logique binaire: Les signaux sont codés par les 2 états 0 et 1. Le retard employé est généralement nul. Cette logique ne permet que la vérification de la fonction logique du circuit sans aucune possibilité d'analyse temporelle.
La vérification est du type table de vérité. On ne peut pas parler de chronogramme de sortie, le temps n'intervenant pas dans l'analyse.

Logique ternaire: Les signaux sont codés par 3 états 0, 1, X. Par l'emploi du X à l'initialisation du circuit et des retards unitaire ou variable, on peut vérifier et chiffrer la stabilisation du circuit qui intervient lorsque l'état X a disparu des sorties.

Le temps écoulé en nombre de pas de simulation entre l'initialisation et la disparition des X indique le temps de stabilisation.

Logique 4 états : On rajoute l'état transitoire ϕ à la logique 3 états (EPISODE [EFC 82]).On peut alors détecter les aléas statiques.

Exemple : changement simultané de 2 valeurs d'entrée d'une porte ET.

Comportement en logique binaire :

	A	B	C
1ère Séquence	0	1	0
2ème Séquence	0	1	0
3ème Séquence	1	0	0
4ème Séquence	1	0	0

C reste toujours à 0.

Comportement avec l'état transitoire ϕ :

	A	B	C
1ère Séquence	0	1	0
2ème Séquence	0	1	0
3ème Séquence	ϕ	ϕ	ϕ
4ème Séquence	1	0	0

Le simulateur détecte un état ϕ à la 3ème séquence qui perturbe l'état stable à 0 de C avec la logique binaire. La modèlisation permet d'indiquer ici une possibilité de 0-aléa statique.

Logique 5 états : Les transitions montantes et descendantes sont alors spécifiées et remplacent l'état ϕ . On peut utiliser alors les temps de montée et temps de descente qui peuvent être effectivement différents (par exemple inverseur NMOS).

D'autres logiques utilisant de nombreux états existants permettent des simulations plus précises [BRE 72],[ACK79].

Toutefois, l'apparition de technologie utilisant les portes de transferts ou les interrupteurs (MOS par exemple) a conduit à développer des logiques utilisant des états haute-impédance. Il existe par exemple des logiques 4 états 0, 1, X, Z. Z étant l'état haute-impédance de sortie d'une porte de transfert lorsque la commande est à 0 (TEGAS [THO 75]). Le simulateur SALOGS [ACK 79] utilise 8 états comprenant la haute-impédance et la transition vers la haute-impédance.

Cet état de haute-impédance a d'abord été considéré comme un état logique à part entière. Or il code la conductance de sortie d'un élément alors que les autres états codent la tension ou ses variations sur une équipotentielle. Son emploi et le développement des technologies MOS ont conduit à une nécessaire extension de la modèlisation exposée ci-dessus.

I. 3. EXTENSION DE LA MODELISATION

La modèlisation présentée ci-dessus est adaptée aux problèmes de circuits à logique câblée. Elle devient insuffisante pour les circuits intégrés, car ceux-ci font appel à des fonctions spéciales telles que portes de transfert, bus ....

Les insuffisances sont les suivantes :

- Une connexion (ou noeud) ne peut être commandée que par une seule porte logique. Dans le cas où deux portes, l'une à 1, l'autre à 0 commandent un même noeud, il y a conflit. Or, par exemple en NMOS, cette configuration a un sens, et réalise un ET-câblé. Le problème peut être tourné en représentant dans la description du circuit une porte factice ET.

- Les éléments spéciaux comme interrupteurs, bus, porte trois états, points capacitifs, sont difficiles à modèliser avec les seules informations sur les tensions électriques des noeuds.
Une extension de la modèlisation est donc apparue sous la forme d'états supplémentaires codant la force des si-

gnaux. Cette force est dûe à l'impédance de sortie de l'élément commandant le noeud, et à la propre capacité du noeud en cas de stockage.

Différentes forces sont employées selon les simulateurs. Ces forces sont ordonnées en ce sens, que en cas de conflit, le signal ayant la force la plus élevée écrase les autres.
Les principales sont par ordre décroissant :

- E (externe): elle code un signal externe, alimentation, masse, horloge .

- D (fort): elle code un signal régénéré, par exemple un signal sortant du transistor signal d'une porte MOS.

- R (résistif) ou W (faible): elles codent un signal dégénéré par son passage dans un élément résistif (transistor de charge par exemple).

- Z (très haute impédance): elle code un signal stocké sur une grosse capacité (bus par exemple).

- z (haute impédance): elle code un signal stocké sur une petite capacité (point capacitif).

Certains simulateurs emploient des forces suplémentaires, subdivisions de celles décrites ci-dessus, et également une ou plusieurs forces inconnues (U).

Un signal est décrit par un couple de valeurs (état, force) à chaque instant de simulation.
Les simulateurs annonçant l'emploi de 12 états comme le Daisy Logic Simulator [DAI 84] codent les signaux par trois niveaux de tension 0, 1, X,et quatre niveaux de force D, R,Z , U, soit douze couples possibles.

Les noeuds se voient attribuer une taille selon leur capacitance;
Un point capacitif peut mémoriser (1,z), (0,z), (X,z) ;
Un bus peut mémoriser (1,Z),(0,Z),(X,Z).
Les interconnexions ne sont donc plus considérées comme passives.

Les éléments se voient attribuer des forces proportionnelles à leurs conductances de sortie. Par exemple, un inverseur NMOS fournit en sortie un 'Zéro fort' et un 'Un faible'. Dans le cas du Zéro, l'impédance de sortie de l'inverseur est celle du transistor signal passant, donc très faible, la conductance est donc élevée, et la force également. Dans le cas du Un, l'impédance de sortie est celle du transistor de charge qui fonctionne en résistance donc moyenne ou forte, la force associée est donc résistive ou faible.

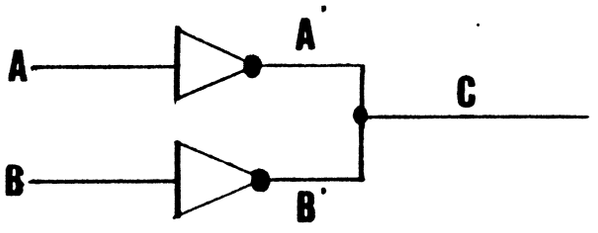
Un signal indéterminé X à l'entrée de l'inverseur NMOS fournit une force inconnue en sortie.

Un inverseur CMOS fournit un 'Zéro fort' et un 'Un fort' puisqu'il y a deux transistors Signaux PMOS et NMOS.
Un signal indéterminé fournit un signal fort en sortie.
Un interrupteur fermé transforme un signal fort en signal résistif puisqu'il agit comme une résistance entre ses drain et source.

Avec cette modèlisation, un noeud peut être commandé par plusieurs portes logiques, et il se comporte alors comme un élément ET-câblé. Les règles de fonctionnement de ce ET-câblé sont alors les suivantes :
Plusieurs signaux tentent de commander un noeud,

- Si les forces de ces signaux sont différentes, alors le signal prend au noeud considéré l'état et la force du signal de plus grande force.
- Si les plus grandes forces sont identiques et de même état, alors le signal résultant prend ce couple de valeurs.
- Si les plus grandes forces sont identiques et les états différents, alors le signal prend l'état inconnu et cette force.

Exemple: Noeud commandé par 2 inverseurs NMOS



A= (0,D) => A' = (1,R)
B= (1,D) => B' = (0,D)
=> C = (0,D)

I. 4. MODELE DE L'INTERRUPTEUR M.O.S. (SWITCH-LEVEL)

L'utilisation des forces décrites ci-dessus s'est fait dans un premier temps avec des éléments logiques c'est-à-dire en conservant le principe d'unidirectionnalité. Or les interrupteurs MOS sont des éléments bi-directionnels.

Afin d'étudier de manière plus précise les effets de cette bi-directionnalité, un nouveau niveau de simulation a été introduit, le niveau interrupteur (Switch-Level), développé par BRYANT, avec le simulateur MOSSIM [BRY 80].

Le circuit à simuler est décrit par un ensemble de noeuds et de transistors.

Le transistor est défini comme un interrupteur commandé en tension par sa grille. La grille est un noeud d'entrée, les drain et source sont tous deux noeuds d'entrée-sortie. On peut assigner ou non un retard d'ouverture et de fermeture à l'interrupteur.

L'état du transistor en fonction de son type et de sa tension grille est indiqué par le tableau ci-dessous :

GRILLE	TYPE N	TYPE P	TYPE D
0	ouvert	fermé	fermé
1	fermé	ouvert	fermé
X	inconnu	inconnu	fermé

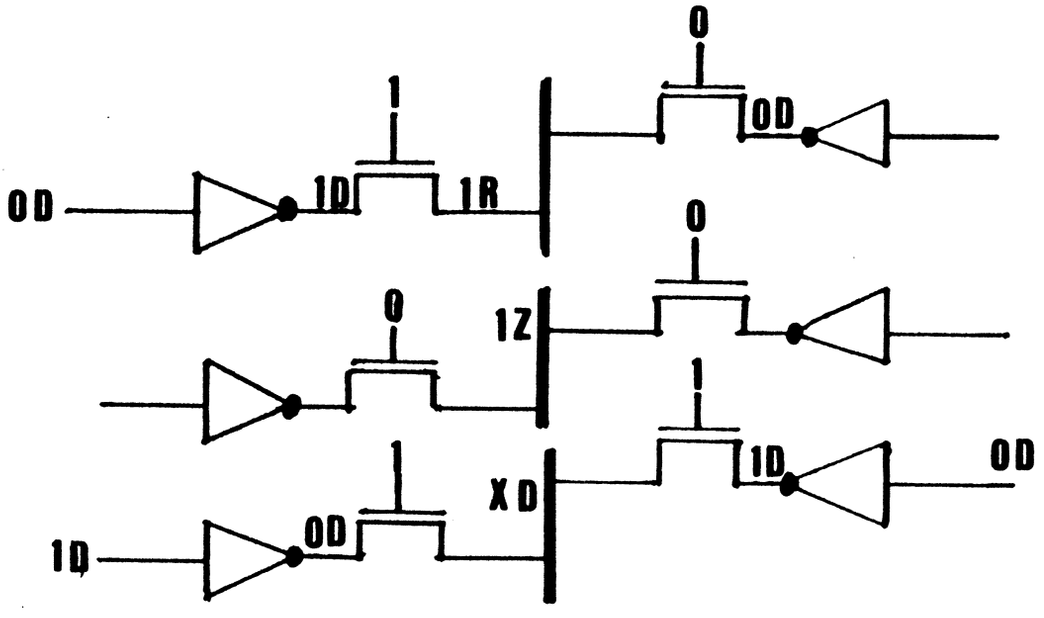
On assigne au transistor une force selon son type et sa taille géométrique. Cette force code la conductance de sortie du transistor.

Les valeurs des signaux logiques (force et état) sont calculés à chaque noeud du circuit suivant la règle du ET-cablé définie plus haut, en considérant seulement les interrupteurs fermés conduisant au noeud. On ne tient pas compte d'un sens de propagation des signaux.

Des problèmes se posent avec la valeur X sur la grille de l'interrupteur, donnant un état inconnu de l'interrupteur pour les types P et N.
Suivant les simulateurs, on considère que l'interrupteur est passant, non-passant, ou alors le simulateur étudie les deux solutions possibles dans une certaine profondeur du circuit [BRY 80],[BRY 81],[STE 83].

L'étude des interrupteurs en état indéterminé est le gros problème des simulateurs utilisant cette modèlisation..
Le comportement du circuit peut être complètement différent suivant l'état d'un interrupteur puisque celui-ci peut conditionner la propagation des signaux dans une partie du circuit.

Exemple : Bus



I. 5. FONCTIONNEMENT DE LA SIMULATION A EVENEMENTS

La simulation d'un circuit nécessite des informations sur la description du circuit, les séquences d'entrée et leur instant d'apparition, les valeurs initiales des éléments du circuit, les éléments du circuit, les éléments à observer en sortie du simulateur. Une fois ces données définies par l'utilisateur et saisies par le simulateur, la simulation peut débuter.

Elle se décompose en trois phases :

- Prétraitement
- Calcul
- Mise en forme et édition des résultats

Prétraitement : le simulateur vérifie la cohérence des informations fournies par le concepteur et les transforme en une structure de données directement utilisable.
La structure la plus employée est dite en tableau.
Elle comprend de manière simplifiée trois tableaux, [SZY 72].

- Un tableau de description des éléments logiques.

- Un tableau d'interconnexion décrivant l'interconnexion des éléments entre eux. Une interconnecxion correspond à une équipotentielle; il lui sera associé une variable au cours de la simulation.

- Un tableau de description des types logiques contenant les différentes fonctions logiques réalisées par les éléments (sous forme de tables de vérité par exemple).

Calcul : une fois l'initialisation des éléments effectuée et la première séquence d'entrée chargée, le calcul peut commencer. Le dispositif de base est une liste appelée échéancier, dans laquelle sont ordonnés, dans un ordre temporel correct les évènements à survenir.

L'échéancier est le coeur des simulateurs logiciels; de nombreux travaux ont porté sur son optimisation et son utilisation [SZY 72],[PHI 78].

Dans le cas de la simulation à retard unitaire, chaque élément introduit un retard égal à une unité de temps du simulateur. La simulation procède alors en recalculant les valeurs de toutes les variables (interconnexions) à t + 1 en fonction de ce qu'elles étaient à t Il suffit donc de deux exemplaires de l'ensemble de ces variables.

Par contre, dans le cas d'une simulation à retard variable, il n'est bien sûr pas possible de conserver N + 1 exemplaires de l'ensemble de ces variables (si N est le délai le plus long). C'est pour éviter cela,qu'a été introduite une simulation par évènement guidée par un échéancier, dans lequel on range dans leur ordre d'occurence les évènements à survenir.

Les définitions de base sont :

- Une évaluation est le calcul de la valeur du signal de sortie d'un élément logique en fonction des valeurs de signaux d'entrée.

- Un évènement est un changement de valeur d'un signal d'entrée. Il est donc produit par une évaluation changeant la valeur de sortie de l'élément.

- Un pas de simulation correspond au calcul de tous les évèvements se produisant au même instant de simulation.

- Un cycle de simulation correspond aux pas de simulation compris entre deux séquences d'entrée.

- Activité d'un circuit - un élément est actif s'il survient un changement de valeur sur un de ses signaux d'entrée. On considère que dans un circuit 10 à 40% (suivant les auteurs) des éléments sont actifs à chaque pas de simulation. L'activité d'un circuit dépend du type de retard employé pour la simulation.

L'algorithme de simulation est le suivant :

```
Pour chaque séquence d'entrée
    Pour chaque évènement de l'instant T évaluer
    | tous les éléments concernés par l'évènement.
    |   Si il y a changement d'état alors
    |   |   Calculer le retard
    |   |   Pour chaque successeur, prévoir le
    |   |   nouvel évènement et l'insérer dans
    |   |   l'échéancier.
    |   Sinon rien..
    |   effacer l'évènement traité de l'échéancier.
    | incrémenter le temps de simulation.
```

Lorsque l'échéancier ne comporte plus d'évènements à traiter, le calcul est terminé. Cependant il peut y avoir oscillation du circuit simulé et donc indéfiniment des évènements à traiter, d'où un bouclage possible du simulateur.

Complexité de l'algorithme : BREUER a proposé une estimation de la complexité de l'algorithme de simulation [BRE72].

Soit N le nombre de portes à simuler.

Soit t microsecondes, le temps CPU nécessaire à l'évaluation d'une porte.

Pour traiter N portes, il faut t.N. microsec.

Soit A le coefficient d'activité du circuit.

La longueur du pas de simulation est alors

A. t. N. microsec.

Le nombre de pas de simulation nécessaire à la longueur du cycle est quelquefois proportionnelle au nombre de portes : L = K.N.

donc $T = A.\ K.\ t.\ N^2$

L'algorithme serait donc en $O\ (N^2)$.

En fait l'hypothèse de BREUER, valable pour des petits circuits combinatoires, ne l'est probablement plus pour des circuits complexes comme les circuits intégrés actuels. L'estimation de BREUER est la seule que nous ayons trouvée sur les algorithmes de simulation logique.

L'opération principale du calcul est donc l'insertion d'évènements dans l'échéancier.
Dans le cas d'un circuit à sortance moyenne élevée, le nombre d'insertions peut devenir très grand. De plus, si

on traite de très gros circuits, la taille de l'échéancier peut devenir gigantesque.
Ceci est la grosse limitation des simulateurs logiciels.

Edition des résultats: le simulateur communique à l'utilisateur les valeurs des signaux demandés, aux instants demandés. Deux formes d'édition sont généralement employées, une forme tabulaire et une forme graphique.

Dans la forme tabulaire, les valeurs des signaux sont rangées en fonction du temps de simulation dans un tableau.

Dans la forme graphique, les signaux apparaissent sous forme de chronogramme, la forme codant l'état de tension, la couleur de l'onde codant la force du signal (Daisy Logic Simulator).

I. 6. CARACTERISTIQUES DES SIMULATEURS

Ces caractéristiques sont : la vitesse d'exécution, la précision, la capacité, la facilité d'utilisation.

Vitesse d'exécution et capacité : la vitesse s'exprime selon les fabricants en nombre d'évaluations par seconde, nombre d'évènements par seconde, nombre de portes simulées par seconde. On peut également indiquer la durée du cycle de simulation pour un nombre de portes déterminées.

Il faut noter que :

1) Un ou plusieurs évènements simultanés induisent une évaluation. Par exemple, pour une porte à trois entrées, le changement simultané des trois signaux d'entrée (trois évènements) n'induisent qu'une évaluation.
Cet effet est important dans une simulation retard unitaire, car de nombreux évènements simultanés peuvent se produire; par contre, avec une simulation à retard variable, la probabilité d'évènements simultanés à l'entrée d'une porte est beaucoup plus faible et cet effet est amoindri.

2) Une évaluation n'induit pas d'évènement si la valeur de sortie ne change pas.

3) Par contre, si elle change, le nombre d'évènements induits dépend de la sortance de la porte.

Pour comparer les différentes unités utilisées dans la mesure de l'efficacité des simulateurs, [WER 84] propose un rapport nombre d'évaluation par seconde sur nombre d'évènements par seconde égal à la sortance moyenne du circuit [WER 83].

$$\frac{\text{Evaluation}}{\text{Evènements}} = \text{Sortance moyenne}$$

Le nombre de portes traitées par seconde peut être ramené au nombre d'évaluations par seconde en considérant l'activité du circuit :

$$\frac{\text{N Evaluations}}{\text{N Portes}} = \text{Activité}$$

La vitesse de simulation va de la centaine d'évaluations par seconde, à la dizaine de milliers.
La capacité et la vitesse sont souvent liées. Elles dépendent toutes deux de la machine hôte :

- Micrologic (Spectrum) effectue 300 évaluations par seconde sur APPLE II, et traite des circuits de plusieurs centaines de portes.[MIL 84].

- Daisy Logic Simulator effectue 1000 évaluations par seconde sans accélérateur [DAI 84].

-TEGAS et LOGIS atteignent la dizaine de milliers d'évaluations par seconde sur Tegas Station et VAX 780 respectivement.

Un même circuit est simulé sur SCALD (Valid) en 11 secondes et sur C 2000 (Chancellor) en 45mn (C 2000 tourne sur IBM PC) [WER 83].

Ces différents simulateurs sont chacun adaptés à des tailles et des types de circuits différents pour répondre à des besoins différents. La limite en vitesse des simulateurs logiciels se situe dans la dizaine de milliers d'évaluations par seconde.
Pour aller au-delà et traiter des circuits de taille encore plus grande, des machines spécialisées sont nécessaires.

Précision :c'est généralement le nombre d'états employés qui détermine la précision de la simulation.
Le nombre d'états va de 2 (premiers simulateurs) à 99 (ICAP de Phoenix). Les simulateurs classiques offrent 4 états : Zéro, Un, Indéterminé, Haute-Impédance comme TEGAS et LOGCAP. Actuellement on voit apparaître des simulateurs 9 états, 3 forces et 3 tensions (CAE, Chancellor

Mentor...), des simulateurs 12 états, 4 forces et 3 tensions (Daisy, Cadat).. ICAP avec ses 99 états emploie 6 tensions et 16 forces plus 3 états indéterminés. Le nombre d'états employés dépend également du type de simulation désirée.

Pour effectuer des simulations temporelles, on rajoute généralement des états transitoires supplémentaires. Par exemple SALOGS passe de 4 états en statique à 8 états en dynamique [ACK 79] et Daisy Logic Simulator passe de 12 à 24 états [DAI 84].

La précision et la vitesse sont bien sûr liées. Plus un simulateur est précis, plus il est lent.

Facilité d'utilisation : d'après [WER 84], le confort d'utilisation se juge sur les trois points suivants :

- Description d'entrée du circuit fourni par l'utilisateur au simulateur.
- Temps de réponse du simulateur.
- Formats des résultats.

1) Description d'entrée : la plupart des simulateurs offrent des possibilités de saisie graphique.

Si ce n'est pas le cas, l'utilisateur doit spécifier ses circuits sous forme de liste de connectivité, ou dans un langage de description particulier (HDL).

2) Temps de réponse : il est indépendant des temps de simulation et d'édition des résultats. De façon à permettre une conception rapide, ce temps de réponse doit être à l'échelle humaine. Tous les efforts de fabricants tendent vers l'obtention de systèmes permettant des simulations interactives.

3) Formats des résultats : le format le plus confortable est l'édition de chronogrammes sur écran graphique.
Là, plusieurs degrés sont possibles : possibilité pour l'utilisateur de définir ses axes, d'utiliser ses chronogrammes de sortie comme séquence d'entrée d'autres circuits (CAE2000), de zoomer les chronogrammes (DAISY), affichage des valeurs de bus en hexadécimal (SCALD) [WER 83] .....

D'une façon générale, le simulateur logique, outil de base, tend à s'intégrer dans un système de CAO comportant un grand nombre d'outils.

CHAPITRE II

MACHINES SPECIALISEES

POUR LA SIMULATION LOGIQUE

Présentation

Les améliorations apportées aux simulateurs logiciels ont porté sur l'affinage des algorithmes de simulation [BRE 72],sur les structures de données employées [THO 75] et sur le coeur du système, l'échéancier et sa gestion [PHI 78] [NEW].

Ces modifications ont amélioré les performances des simulateurs, mais n'ont pas pu changer l'ordre de grandeur de la vitesse de simulation. Or, la taille des circuits augmentant de façon considérable, il était nécessaire pour conserver des temps de simulation permettant des conceptions rapides, d'étudier des architectures spécialisées capables de vitesse de simulation très grande.

Le simulateur logiciel, bien qu'ayant plusieurs évènements à traiter au même instant de simulation, les traite les uns après les autres, de manière séquentielle. Les études ont donc porté sur des architectures capables de traiter des évènements ou des tâches en parallèle. Elles ont débouché sur des machines à architecture pipeline, parallèle , distribuée ou à des combinaisons de ces architectures.

Le parallélisme permet le traitement simultané de plusieurs évènements de manière à réduire le temps global de calcul. Deux types de parallélisme peuvent être exploités dans la simulation logique :

- le premier dans l'algorithme de simulation.
- le second dans le circuit à simuler.

Le premier type s'appelle parallélisme algorithmique.
Un certain nombre d'opérations sont à effectuer dans un pas de simulation: détermination de l'évènement courant, remise à jour des valeurs d'entrée, évaluation, prévision des évènements futurs.
Ces opérations sont effectuées les unes après les autres dans un simulateur classique pour un seul évènement et les évènements traités séquentiellement. Or, si plusieurs de ces opérations algorithmiques concernant un même évènement sont effectuées en même temps, la vitesse de simulation peut être augmentée.

Par exemple, après l'évaluation d'un élément, on peut en évaluer un autre pendant que l'on prévoit les évènements induits par le premier. La principale recette de cette approche est le partitionnement des tâches, leur répartition entre plusieurs processeurs, et l'exécution ' pipeline ' des travaux.

Le second type de parallélisme peut s'appeler parallélisme matériel. Des évènements simultanés se produisent dans le même pas de simulation parce que des signaux électriques se propagent simultanément le long de différentes connexions dans le circuit. Si les éléments devenant actifs au même moment sont simulés simultanément par des processeurs différents, alors le temps global de simulation sera réduit.

Ces deux types de parallélisme induisent deux architectures de base :

- l'une dans laquelle des processeurs se partagent les tâches de l'algorithme, l'architecture pipeline est un exemple.
- l'autre, dans laquelle des processeurs se partagent le circuit à simuler. On parle alors d'architecture distribuée.

Des machines spécialisées ont été développées, utilisant l'une et l'autre architecture de base ou une combinaison de ces deux types. Neuf machines ont été recensées par nos soins dans la littérature.
Leur architecture et leurs principes sont exposés ci-après.

II. 1."HARDWARE SIMULATOR" de BARTO et SZYGENDA [BAR 84]

BARTO et SYGENDA ont présenté une machine pipeline basée sur le fait que les deux principales phases de la simulation sont la phase de remise à jour des signaux et celle d'évaluation des éléments logiques.
L'architecture est construite avec cinq blocs mémoires et trois processeurs.
3 blocs mémoires servent à la description du réseau :

- SDM (State Data Memory) construit la description des éléments logiques.
- FIM (Fanin Memory) contient les entrées des éléments et les valeurs de leurs signaux.
- FOM (Fanout Memory) contient les sorties et leurs valeurs.

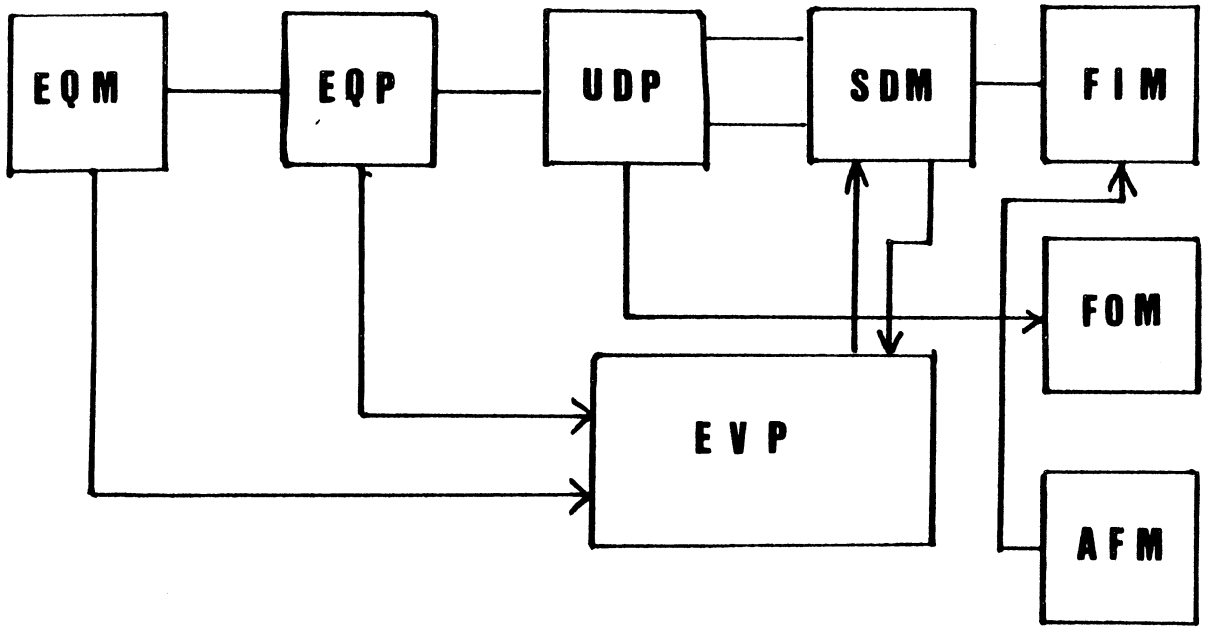
Les informations sur l'activité du circuit sont stockées dans le bloc mémoire AFM (Activity Memory Flag).

Le rôle de l'échéancier est tenu par un bloc mémoire EQM (Event Queue Memory).

Les évènements sont générés par le processeur d'évaluation EVP (Event Queue Processor).

Le processeur UDP (Update Processor) effectue la remise à jour des signaux.

Le synoptique ci-dessous indique l'architecture globale de cette machine :



UDP est un processeur pipeline à sept étages en logique câclée TTL.
EQP et EVP sont des microprocesseurs microprogrammés ayant un temps de cycle de 100 ns par instruction.
Les performances attendues sont de 200.000 à 1 Million d'évaluations par seconde, selon l'entrance moyenne du circuit. Les performances se dégradent si cette entrance augmente.

II.2. MACHINE DE SIMULATION LOGIQUE DE BELL [ABR 82]

Dans cette étude, plusieurs processeurs sont affectés à des tâches spécifiques de l'algorithme. L'exécution de celui-ci est donc le résultat de la coopération de tous les processeurs. L'effet d'accélération produit peut être atténué ou annulé, par le fait que des processeurs demandent l'accès d'une mémoire commune.
De manière à éviter ce problème, le principe de l'architecture proposée ici est la répartition des informations dans des mémoires séparées, celles-ci devenant mémoires locales des processeurs.

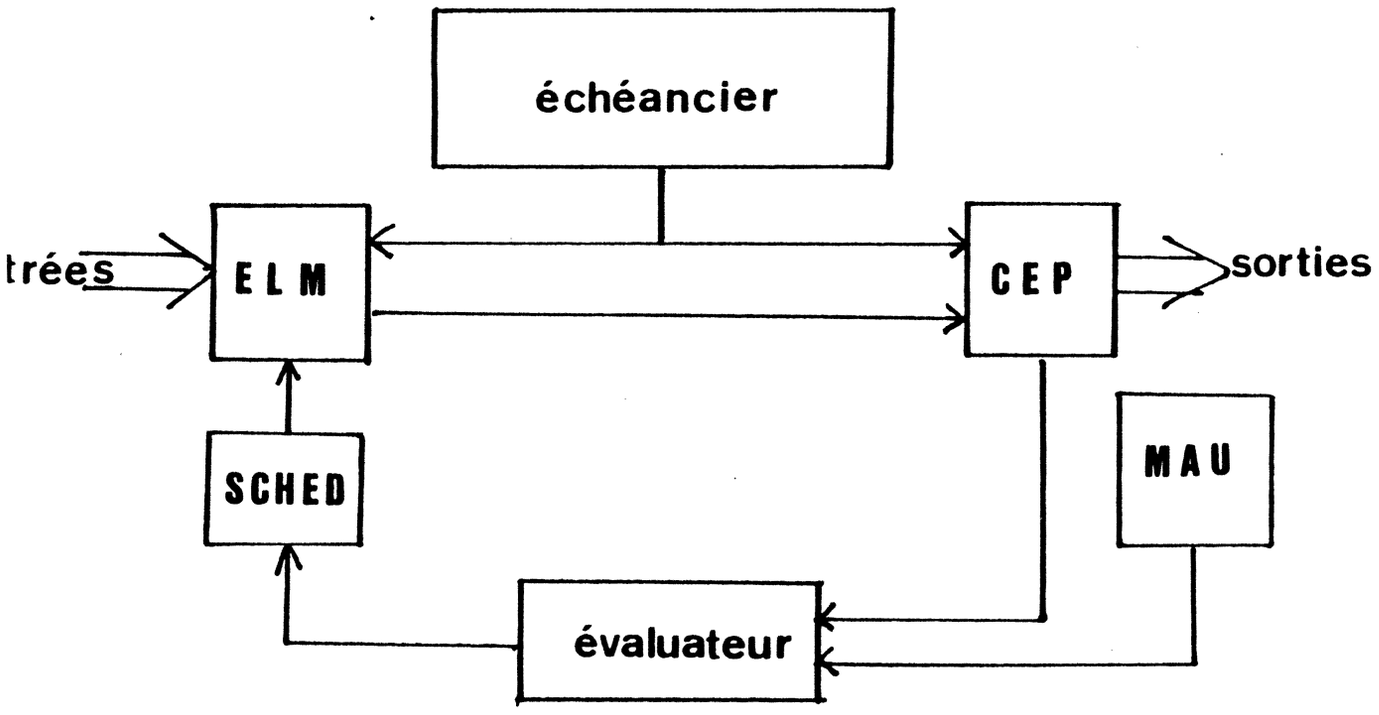
La répartition des tâches entre processeurs et mémoires est indiquée ci-dessous :

Tâche	Processeur	Information
Retrouver l'évènement courant	C E P (current event processor)	Echéancier
Déterminer les successeurs	M A U (model access unit)	Modèle du circuit
Prévision des évènements	S C H E D (Schedule)	Retard des éléments
Insertion dans l'échéancier	E L M (event list manager)	Echéancier

L'échéancier est nécessaire pour deux tâches de l'algorithme.

Ce sera donc une mémoire commune aux deux processeurs CEP et ECM effectuant ces deux tâches.

D'où l'architecture de la machine :



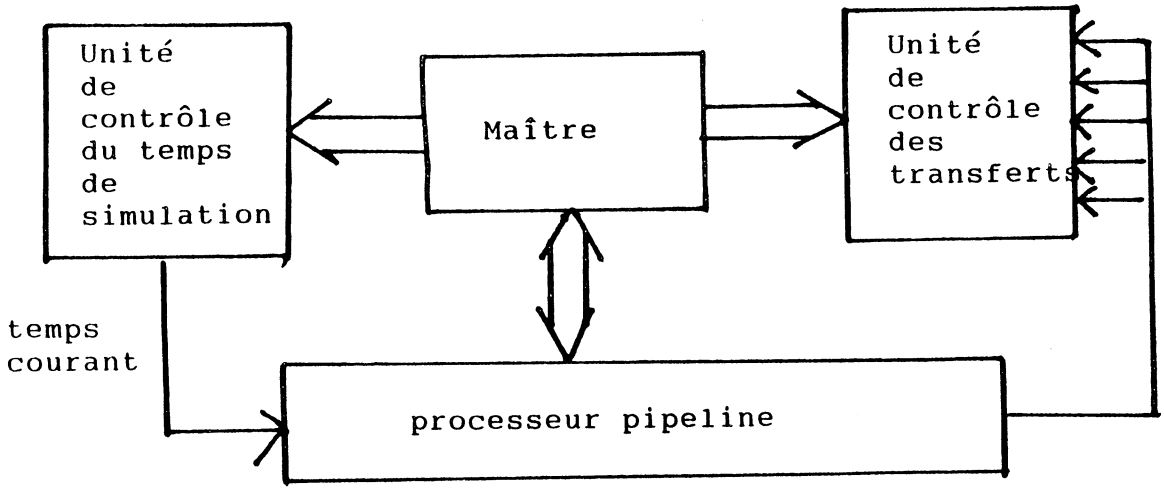
Les performances annoncées par les auteurs vont de 500.000 à 1 Million d'évaluations par seconde.

II. 3. ULTIMATE [GLA 84]

Ultimate est une architecture pipeline synchronisée sur une horloge maître. Les transferts de données entre étages

sont commandés par un processeur maître. La durée des tâches effectuées dans chaque étage pouvant varier, chacun délivre au maître un signal libre/occupé. Le maître autorise le transfert ou non, suivant le contenu du message.

Schéma de l'architecture :



Le processeur pipeline contient 11 étages avec mémoires locales. Le maître contient des procédures pour tâches utilisées de manière non-systematique dans la simulation (détection d'oscillation....). lorsque ces procédures sont demandées par le processeur pipeline, le maître génère des signaux d'interruption pour celui-ci.

Les performances annoncées sont de l'ordre de 2 Millions d'évaluations par seconde pour un circuit à entrance moyenne de 3. Les auteurs prévoient une dégradation des performances, si cette entrance augmente.

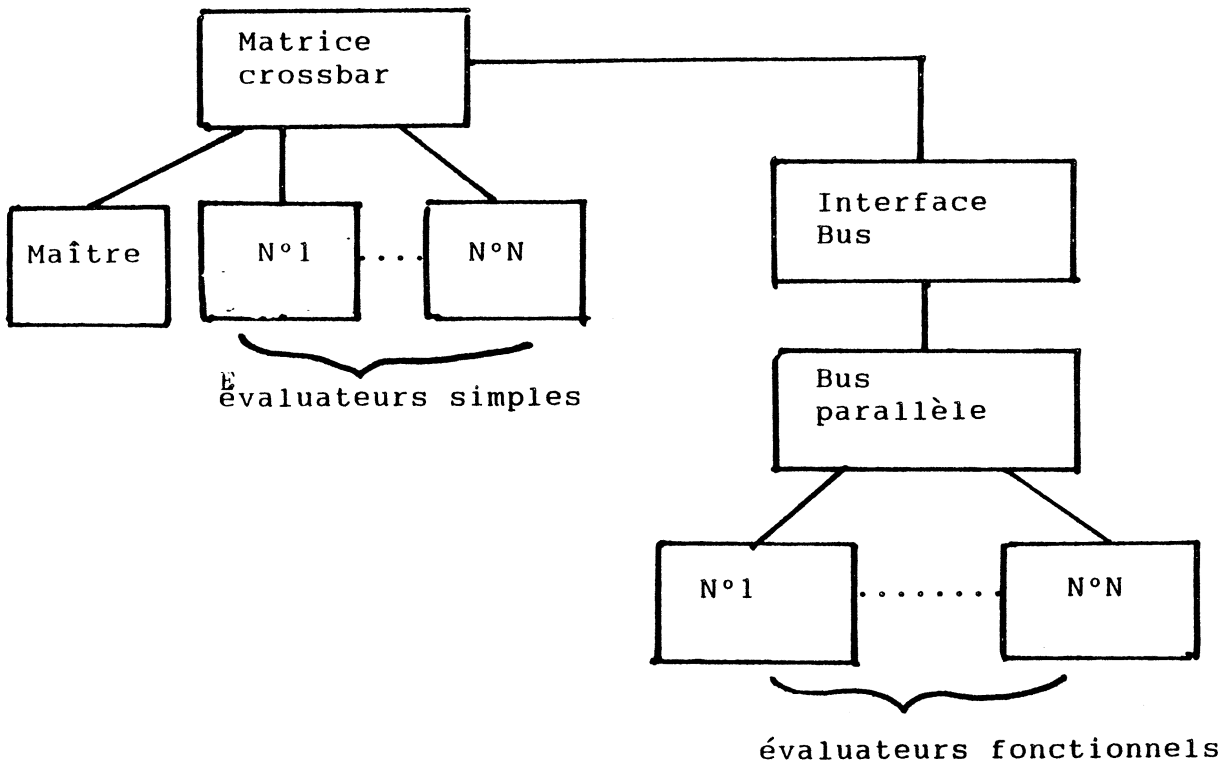
II. 4.MACHINE MULTIPROCESSEUR DE BELL [LEV 82]

Cette machine utilise le partitionnement du circuit à simuler en plusieurs blocs.
Chaque bloc est affecté à un processeur.

La simulation est possible aux niveaux porte et fonctionnel.
La machine comprend (n+1) microprocesseurs interconnectés par une structure de communication.

L'étude a été menée avec des microprocesseurs
16 bits Intel 80 86 et Am 29116.

Schéma de l'architecture :



La machine comprend un maître et 2 N esclaves (évaluateurs) Les N évaluateurs simples simulent des blocs au niveau porte. Les N évaluateurs fonctionnels simulent des blocs au niveau fonctionnel. Tous les microprocesseurs sont identiques, y compris le maître.

Au début de chaque cycle de simulation, le maître envoie les signaux d'entrée aux évaluateurs concernés par l'intermédiaire de la structure de communication Puis il gé-

nère un signal START autorisant les esclaves à travailler.

Les évaluateurs génèrent des données pour eux-mêmes ou pour leurs voisins, pour les instants de simulation suivants. L'heure de simulation globale n'est pas envoyée aux évaluateurs. Chaque esclave informe le maître lorsqu'il a fini le traitement, en envoyant un signal DONE. Lorsque tous les signaux DONE sont arrivés au maître, celui-ci peut envoyer la séquence et le signal START suivant.

Le point clé de la simulation est le partitionnement du circuit. Il est effectué par la machine hôte qui effectue également le pré-traitement et post-traitement de la simulation.
Le circuit est décomposé en blocs, chacun étant associé à un évaluateur. Les connexions entre blocs induisent des communications interévaluateurs. Le partitionnement doit être effectué de façon à minimiser ces communications.
Les auteurs indiquent un partitionnement élémentaire, basé sur la profondeur du circuit représenté entièrement au niveau porte (donc suppression de la hiérarchie)
Ce partitionnement est assez intuitif pour des petits circuits combinatoires. Par contre, pour des circuits rebouclés, il semble insuffisant.

Chaque évaluateur possède en mémoire la description du bloc, une liste d'activité (éléments à évaluer à chaque instant), et un échéancier interne sous forme de liste circulaire (timing wheel).
Le même logiciel de simulation est implanté sur tous les microprocesseurs. Les performances annoncées sont résumées dans le tableau ci-dessous en fonction du nombre N de proces_eurs pour évaluation simple (niveau porte).

N	90	256	512
nombre d'évaluations par seconde	3 Millions	9 Millions	18 Millions

II.5.MACHINE DE YORKTOWN [DEN 82] [PFI 82] [KRO 82]

YSE est une machine spécialisée pour la simulation logique utilisant 256 processeurs en parallèle. Les processeurs communiquent entre eux par l'intermédiaire d'un processeur de communication. Le circuit à simuler est décomposé en blocs, chacun étant affecté à un des 256 processeurs.

Chaque processeur peut simuler jusqu'à 4.096 fonctions logiques à 4 entrées, 1 sortie, la sortance n'étant pas limitée, dans une simulation 4 états à retard nul ou unitaire. YSE peut simuler des circuits de 1 Million de portes, et selon les auteurs à la vitesse de 3 Milliards d'évaluations par seconde.
On distingue deux types de processeurs :

- Les processeurs logiques sont des processeurs pipeline à 8 étages, permettant l'évaluation de 16 fonctions logiques différentes. L'évaluation d'une porte s'effectue en 80 ns.

- Les processeurs ARRAY permettent de simuler des RAM et des ROM. Ils évitent de saturer les processeurs logiques qui pourraient accomplir cette tâche.

YSE permet des vérifications logiques et pseudo-fonctionnelles. Le niveau fonctionnel apparaît juste au niveau de la description d'entrée. Cette description est éclatée au prétraitement et au réseau de portes logiques.

Le niveau interrupteur (SWITCH) a récemment été introduit dans la machine.

La phase de prétraitement éclate ou regroupe les blocs logiques en fonctions logiques compatibles avec la machine. Le réseau global est ensuite partitionné en plusieurs sous-réseaux, chacun étant associé à un processeur logique. Plusieurs types de partitionnement sont utilisés devant réduire les communications inter-processeurs :

- remplissage des processeurs selon la liste de connectivité.
- même remplissage que précédemment, avec en plus les prédécesseurs de chaque porte placés dans le même processeur.
- placement aléatoire et amélioration par relaxation.

Ces types de partitionnement sont assez rudimentaires, (surtout les deux premiers) et selon les auteurs, ils donnent d'excellents résultats.

Au niveau matériel, un processeur logique comprend 600 modules logiques TTL et 130 modules mémoires MOS.
Au total 20.000 composants MSI en technologie TTL, ECL, MOS composent la machine YSE.

Au niveau des performances, les auteurs annoncent des simulations 1000 fois plus rapides qu'avec leur simulateur

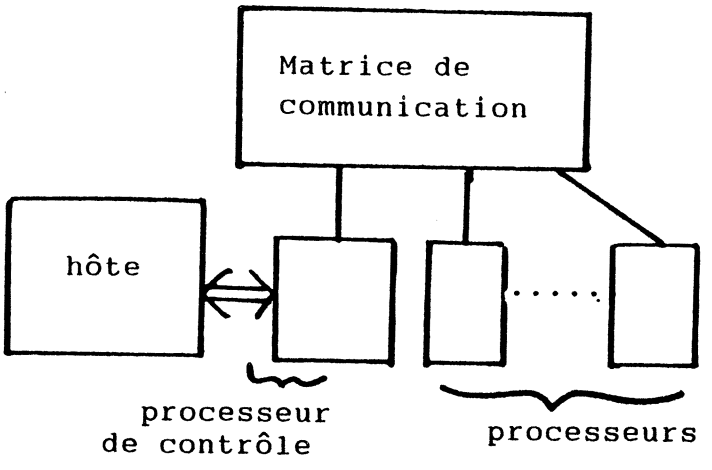
logiciel tournant sur IBM 370. Toutefois le chiffre le de 3 Milliards d'évaluations par seconde paraît assez douteux: un processeur pipeline effectue une évaluation en 80 ns. 256 processeurs sont en parallèle , on a donc 256 évaluations en 80ns, soit 3,2 Milliards d'évaluations par seconde.

Pour faire ce calcul, on considère donc qu'il n'y a aucune communication de processeur à processeur, ce qui est tout à fait irréaliste.

II. 6. HAL [SAS 83]

HAL est une machine spécialisée pour la simulation de gros systèmes logiques 32 processeurs sont en parallèle, interconnectés entre eux par une matrice de communication.

Schéma de l'architecture :



On distingue 29 processeurs logiques, 2 processeurs mémoires et 1 processeur de contrôle relié à la machine hôte.

Le système logique à simuler est décomposé en blocs logiques et mémoires. Le bloc est le point essentiel de la machine HAL. Il peut représenter un réseau logique de quelques portes à quelques centaines de portes, une mémoire RAM ou plusieurs dizaines de RAM.
Le bloc logique est de préférence une entité fonctionnelle du circuit, un circuit entier ou un morceau de circuit.
Le bloc mémoire représente une mémoire ou un ensemble de mémoires RAM.

Le partitionnement du circuit est donc effectué en respectant au maximum la découpe en unités fonctionnelles (hiérarchie du concepteur). Certains processeurs n'utiliseront pas leur capacité de traitement au maximum, par contre les connexions entre processeurs devraient être minimisées et se limiter aux connexions de bloc à bloc.
Chaque processeur logique possède un simulateur câblé effectuant l'algorithme de simulation de manière pipeline.
Les fonctions logiques nécessaires à la simulation sont stockées dans chaque processeur logique sous forme de mémoires associatives décodées suivant le type et les valeurs d'entrée de la porte.

La machine peut effectuer des évaluations aux niveaux porte, registre et fonctionnel..
Chaque processeur est composé de 600 puces MSI. La matrice de communication permet de relier les 32 processeurs entre eux. Elle est composée de 80 puces de routage, chacune comprenant 1000 portes logiques, 2000 composants sont donc nécessaires à la réalisation de la machine.

Les programmes de commande effectuent le partitionnement en blocs, prévoient les communications entre blocs et entre processeurs.

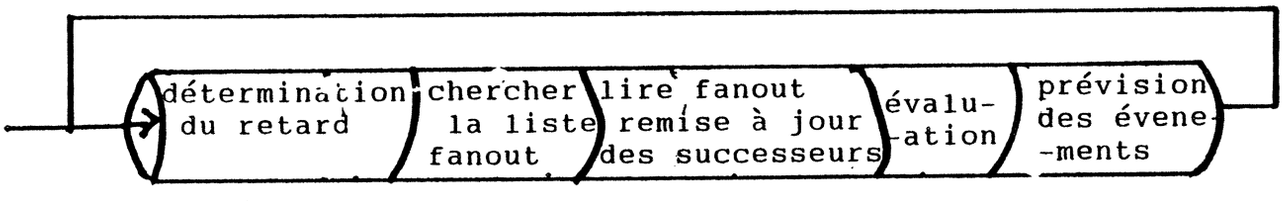
L'hôte est un miniordinateur ACOS de NEC (16 MIPS) .

Les performances annoncées sont les suivantes :

- Un cycle de simulation de 5 msec. pour un circuit de 500.000 portes, ce qui correspond à 50 Millions d'évaluations par seconde avec une activité de 20 %.
- Un prétraitement de 2 heures CPU sur ACOS pour le même circuit.

II.7. SERIE LE 1000 DE ZYCAD [ZYC 84]

ZYCAD commercialise une machine spécialisée pour la simulation logique. Cette machine utilise une architecture basée sur le pipeline et le parallélisme.
L'algorithme de simulation est entièrement câblé dans un module pipeline :



Un module peut traiter 65.000 éléments logiques à une vitesse de 1 Million d'évaluations par seconde. On peut accoler jusqu'à 16 modules en parallèle ; la communication entre modules s'effectuant sur un bus rapide.

Suivant le nombre de modules, on peut traiter des circuits de 65.000 portes à la vitesse de 1 Million d'évaluations par seconde (LE 1002) jusqu'à des circuits de 1,6 Million de portes à la vitesse de 16 Millions d'évaluations par seconde (LE 1032).
Toutefois, le calcul sommaire des auteurs est valable dans l'hypothèse peu réaliste d'échanges entre modules ne ralentissant pas la simulation.

La machine peut effectuer des simulations logiques 12 états (3 tensions et 4 forces) avec retard variable, des simulations temporelles avec détection d'oscillations et des simulations de pannes.
Le fabricant annonce dans un test comparatif, que sa machine est 710 fois plus rapide que TEGAS sur VAX 780 pour la simulation d'un circuit de 1.636 portes unidirectionnelles (0,2 sec. au lieu de 2,4 mn).

II. 8. REALFAST DE VALID [HAR 84]

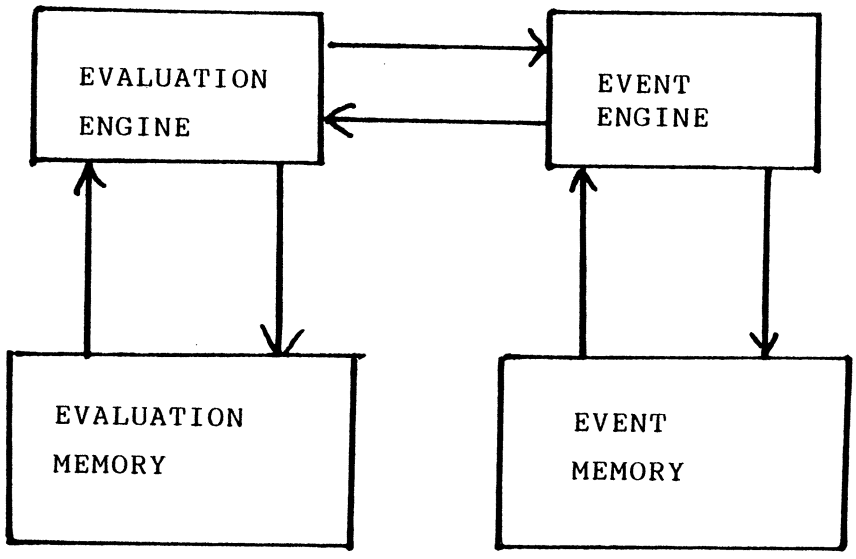
Realfast est un accélérateur de simulation utilisé avec les stations SCALD. Il permet de monter la vitesse de simulation de 1000 évaluations par seconde à 500.000 et de traiter des circuits de 2,5 Millions de portes.

L'architecture est basée sur deux processeurs spécialisés.

- 'Event Engine'- construit en TTL qui gère le temps de simulation, prévoit les évènements.
- 'Evaluation Engine'- qui traite les évaluations est construit avec des microprocesseurs 'bit Slice' Am 2900.

Les processeurs ont leur mémoire spécifique : 32 Méga-octets de RAM. Il exécutent des instructions de 64 bits. l'accélérateur est relié par un interface au S 32 (basé sur 68010) de la station SCALD.

Schéma de l'architecture :



II. 9. MEGALOGICIAN DE DAISY [GIN 83]

La station de travail contient un accélérateur de simulation consistant en trois processeurs spécialisés, connectés en anneau. Chaque processeur est relié à l'hôte

80286/80287, coeur de la station de travail. Ils utilisent 24 bits de donnée et 24 bits d'adressage.

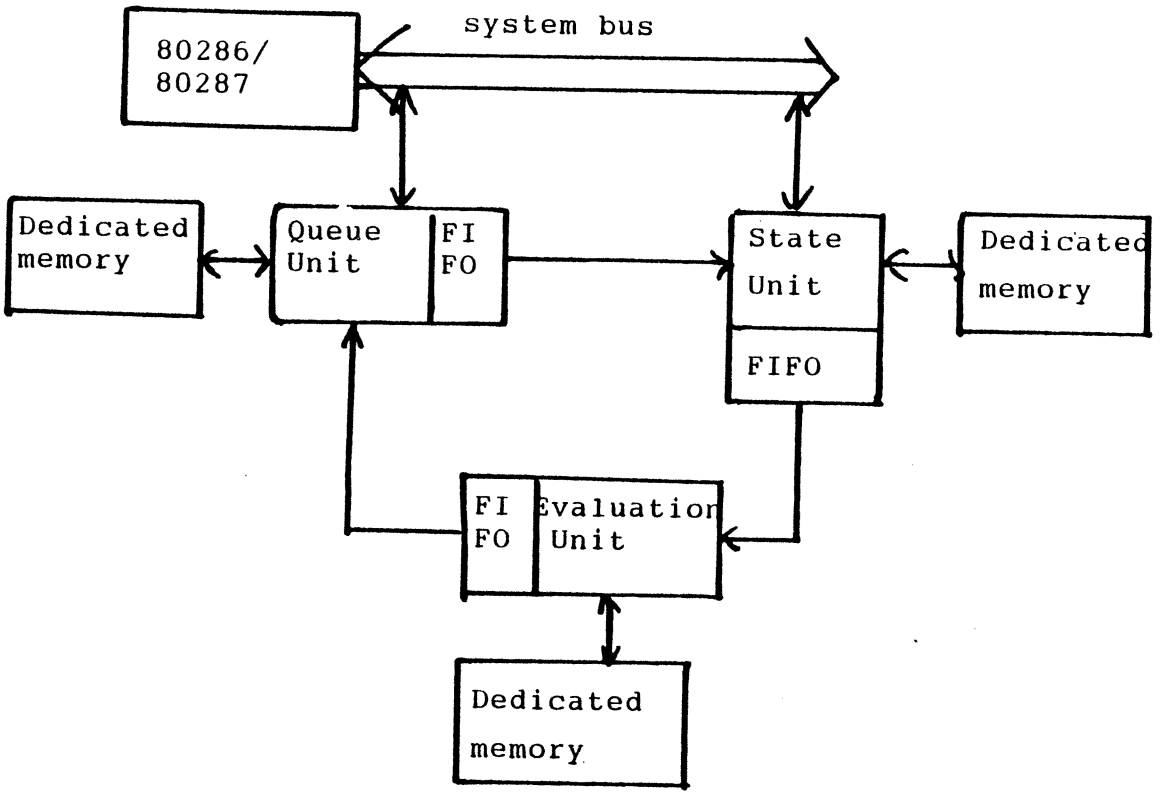
Les trois processeurs communiquent entre eux par des FIFO rapides, insérés dans l'anneau. Ces processeurs sont:

- 'Queue Unit', jouant le rôle d'échéancier et effectuant les manipulations de celui-ci.
- 'State Unit', stockant les états et forces des signaux aux noeuds. Il possède en mémoire les informations sur la connectivité du réseau. Il effectue les remises à jour des noeuds selon les évènements transmis par 'Queue Unit' Il indique à 'Evaluation Unit' les fanout des éléments remis à jour.
- 'Evaluation Unit', effectue les évaluations et s'il y a changement par rapport à la valeur antérieure, transmet à 'Queue Unit' qui détermine les évènements à propager selon les retards adéquats.

Les trois processeurs possèdent leur mémoire locale.

Il n'y a pas de mémoire commune.

Schéma de l'architecture :



L'accélérateur permet de porter la vitesse de simulation de 1000 à 100.000 évaluations par seconde sur la station de travail, de traiter des réseaux de 1 Million de portes logiques dans des simulations temporelles, fonctionnelles et simulations de pannes.

II. 10. REFLEXIONS SUR LES MACHINES EXPOSEES

Les quatre premières machines sont des études d'architecture.
YSE et HAL sont des machines existantes destinées à des besoins internes de grosses sociétés (IBM et NEC), et conçues par elles.

LE 1000 de ZYCAD est une machine commercialisée, destinée à de gros utilisateurs.

Les deux dernières machines, VALID et DAISY sont des accélérateurs plus modestes, mais directement utilisables sur les stations de travail des ces deux sociétés.

Les machines existant réellement sont donc chacune adaptées à des besoins et des utilisateurs différents.

Les notions de parallélisme algorithme et matériel débouchent sur trois architectures : une architecture pipeline, une architecture multiprocesseur distribuée et une architecture combinant les deux premières.

Se classent dans l'architecture purement pipeline, les

trois premières machines et les deux accélérateurs VALID et DAISY.

Se classent dans l'architecture purement multiprocesseur, la machine multiprocesseur de BELL, tous les processeurs possédant le même programme complet de simulation.

Se classent dans l'architecture mixte, la machine de YORKTOWN, HAL, LE 1000.

Les architectures des accélérateurs des stations DAISY et VALID semblent être des réalisations matérielles des études des deux premières machines . Les performances des trois premières machines pipeline semblent être assez optimistes, et devraient être revues en baisse pour concorder avec celles annoncées pour les deux accélerateurs. Les machines purement pipeline ne semblent pas alors dépasser plusieurs centaines de milliers d'évaluations par seconde.
Pour dépasser le Million d'évaluations par seconde, une architecture mixte semble nécessaire (ZYCAD, YSE, HAL).

Une seule étude porte sur une architecture purement multiprocesseur : BELL. Les auteurs en déduisent un nombre de processeurs nécessaire pour atteindre le Million d'évaluations par seconde. L'avantage de cette machine

est qu'elle est réalisable à partir de microprocesseurs standards.

Le point critique des machines utilisant le parallélisme matériel est le partitionnement du circuit et la répartition entre les différents processeurs. De ce point de vue, les auteurs sont peu loquaces; seuls ceux de la machine multiprocesseur de BELL [LEV 82] s'intéressent à ce problème.

Les performances annoncées par certains, notamment ZYCAD et YSE sont sujettes à caution; ZYCAD considère qu'en disposant seize modules pipelines en parallèle la vitesse de simulation est multipliée par seize, pour YSE le même type de raisonnement est utilisé. Seuls les auteurs de [LEV 82] tentent d'estimer les dégradations en vitesse dûes aux communications entre processeurs.

Enfin, dans les performances annoncées, seule YSE atteint le Milliard d'évaluations par seconde, loin devant les autres machines.

CHAPITRE III

MACHINES CELLULAIRES

III. 1. HISTORIQUE ET DEFINITION

Une machine cellulaire est une machine comportant un grand nombre de processeurs ayant leur mémoire locale, connectés selon une certaine architecture.

Ce type de machine permet de traiter des problèmes de manière parallèle dans la mesure où :

- Le système à traiter peut être représenté par un réseau de noeuds connectés entre eux .
- Chaque noeud est affecté à une cellule (processeur).
- Les opérations à effectuer sur chaque cellule ne dépendent que d'informations locales.

Historiquement, les premières machines cellulaires ont été étudiées pour résoudre des problèmes de traitement d'image. Une image étant décrite comme une grille (réseau bi-dimensionnel) de points, un mot de m bits sert à décrire l'information sur chaque point. Une image de N x M points conduit donc à l'étude de machines à N x M processeurs traitant chacun des mots de m bits.

L'architecture de la machine cellulaire est alors calquée sur la topologie du système physique à traiter. Le nombre de processeurs doit alors être égal ou supérieur au nombre d'éléments à traiter, et le facteur d'accélération du traitement est donc ce nombre de processeurs.
Si le nombre d'éléments est supérieur au nombre de cellules, la machine ne peut traiter qu'une fenêtre, sous-ensemble d'éléments, de dimension égale à la grille de processeurs.

Les études sur de telles architectures sont déjà anciennes. UNGER a présenté en 1958, une machine N x N points [UNG 58] .Chaque cellule est connectée à ces huit plus proches voisins.

Actuellement NTT construit une machine 256 x 256 processeurs, destinée à des problèmes de reconnaissance de caractères. Ce réseau est construit avec 1024 puces de 848 processeurs [SU 82]. La puce 848 comprend 81.000 transistors NMOS pour une surface de 94 mm2.

D'autres machines importantes sont en cours de réalisation, comme le 'Cyto-Computer' [LOU 80] , le ' Massively Parallel Processor' construit pour la

NASA pour des problèmes de traitement d'images par satellite [BA 80].

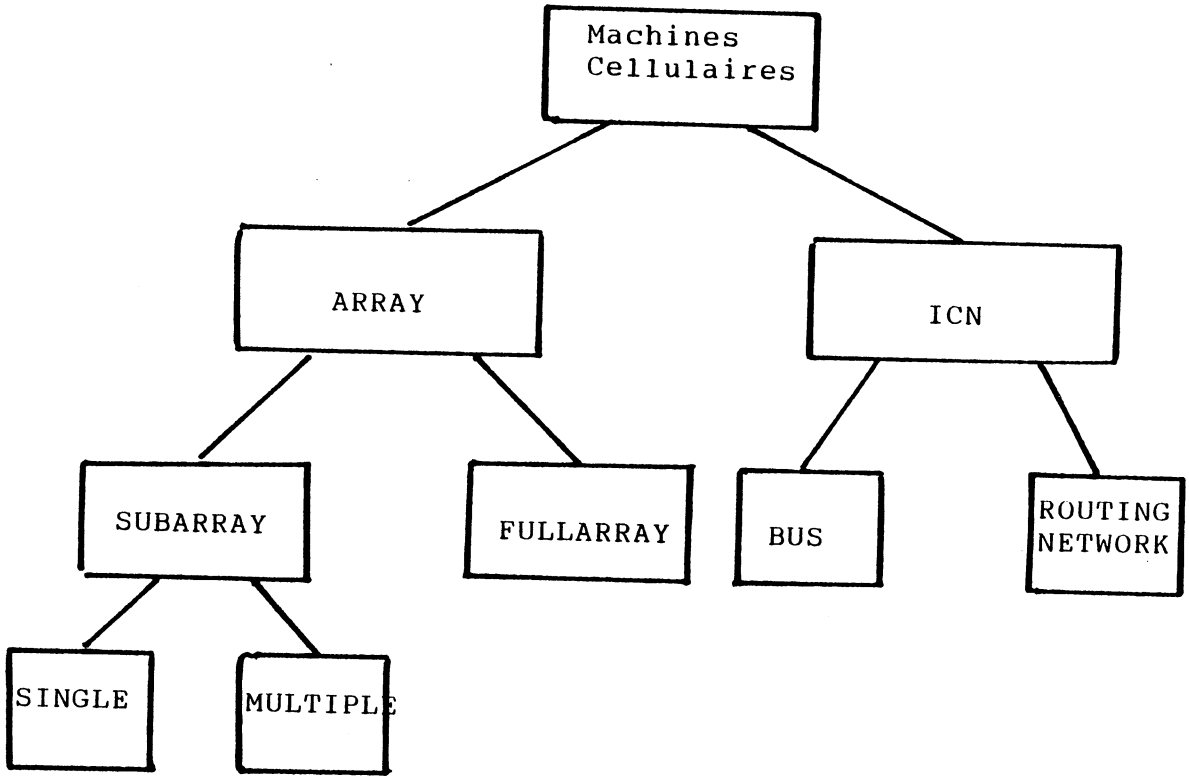
Il existe de nombreuses autres machines cellulaires dans le domaine de traitement d'image et de reconnaissance des formes.
Nous présentons un essai de classification ci-dessous, et on trouvera une classification plus détaillée dans [PRE 83].

III. 2. CLASSIFICATION DES MACHINES CELLULAIRES

Cette classification proposée par RUTENBAR & Co [RUT 84] est établie sur les trois points suivants:

- Comment sont réparties entre les cellules les informations sur les éléments du réseau à traiter?
- Comment le traitement est effectué par les cellules individuellement, ou par des groupes de cellules ?
- Comment les éléments traitement et les éléments mémoire sont interconnectés ?

Le Schéma suivant montre la classification sous forme arborescente :



Au premier niveau, on distingue deux types d'architecture, l'une avec une structure de communication entre les processeurs appelés ICN ('Interconnection Network'), l'autre avec une architecture réseau appelée ARRAY, les

communications se faisant à l'intérieur du réseau par l'intermédiaire des cellules elles-mêmes.

Les machines ICN peuvent avoir deux types de structures de communication :
Soit un bus commun à tous les processeurs,
Soit un réseau de routage (HAL par exemple [SAS83]).

Par machine ARRAY, on entend un ensemble de cellules (processeur + mémoire) ayant une certaine topologie et une machinerie de communication à travers les cellules.

Cette classe se décompose en subarray et fullarray.

Le fullarray est un réseau traditionnel de cellules, chacune connectée à ses proches voisines. Des machines de ce type sont généralement de grosses matrices de processeurs 1 bit, capables de traiter la grille complète, donc de dimension égale à celle-ci, ou alors de dimension plus petite avec des possibilités de repliage de la grille sur le réseau [BLA 82].Le fullarray est envisageable dans la mesure où la grille n'est pas trop importante.

Le subarray est un réseau de taille plus petite que la grille à traiter, n'en traitant qu'une fenêtre. Il se décompose en simple et multiple.

III.3. LES MACHINES CELLULAIRES DANS LA C.A.O.

Dans les domaines de la CAO, les machines traditionnelles sont souvent insuffisamment puissantes pour résoudre rapidement les problèmes posés. Les études menées sur les machines cellulaires ont donc inspiré les ingénieurs en CAO car, pour certains problèmes de conception électronique, une représentation sur grille est utilisée (implantation, extraction, placement, routage) et les informations aux points ne dépendent que des proches voisins.

- Breuer et Shamsa [BRE 81] ont proposé une puce de 256 x 256 processeurs destinée à une machine 1024 x 1024 cellules pour routeur de Lee..

- Blank propose deux architectures ARRAY pour des problèmes de CAO: un 'Bit Mat Processor', réseau 1024 x 1024 de cellules élémentaires avec mémoire locale, un 'Virtual Bit Mat Processor' plus petit 32 x 32, avec mémoire plus grande de manière à replier plusieurs fois une grille plus grande sur le réseau, [BLA 82]
Il utilise ensuite ses architectures pour des problèmes de vérification de règles d'implantation (DRC) et de routage (routeur de Lee).

- Une machine à router, basée sur un réseau 8 x 8 de microprocesseurs a été proposée par HONG & Co [HON 81]. Pour les problèmes de placement, deux machines utilisant des échanges par paires de cellules voisines ont été proposées [CHY 83] et [VED 83], toujours d'après une représentation par grille.

Citons encore un DRC à base de réseau systolique [KAN 83].

D E U X I E M E P A R T I E

##########################

PROPOSITION D'ARCHITECTURE PARALLELE

****

POUR

****

SIMULATEUR LOGIQUE

****

CHAPITRE I

ORGANISATION GENERALE

I. 1. PRESENTATION

Nous proposons une architecture cellulaire de N x N processeurs de type ARRAY (dans la classification du 1-3-2).

Le réseau sert à la simulation logique de la manière suivante: le circuit à simuler, ensemble d'éléments logiques interconnectés, est positionné sur le réseau cellulaire, une porte étant associée à une cellule. La connectivité du circuit est représentée par des messages que s'échangent les cellules, dont les portes associées sont interconnectées.
Les messages sont constitués de l'adresse du destinataire, représentant l'équipotentielle entre une porte et son successeur, et de la valeur du signal logique envoyé.

De manière à simplifier l'étude, une seule porte est affectée à une cellule du réseau. Ceci implique d'avoir un réseau N x N dont le nombre de cellules (N^2) est supérieur ou égal au nombre de portes du circuit à simuler.

Pour simuler de gros réseaux, plusieurs extensions de la machine N x N sont possibles (qui ne seront pas étudiées

en détail ici) : la réalisation de très gros tableaux en WSI, la juxtaposition de nombreux circuits N x N sur une carte , avec partitionnement des réseaux à simuler, ou encore le repliage d'une matrice virtuelle très grande sur la matrice physique.

Nous avons donc à résoudre dans une étape préalable, un problème d'affectation des portes du circuit aux cellules du réseau.
Ce problème crèe une étape de compilation en supplément du prétraitement de la description d'entrée dans les simulateurs classiques.

Chaque cellule doit savoir quel type de porte elle représente, quelles sont les adresses de ses successeurs et tout autre renseignement nécessaire à la simulation. Un autre problème à résoudre, est donc l'initialisation du réseau La phase de compilation doit en plus de l'affectation répartir les renseignements nécessaires entre les cellules, et rendre ainsi possible l'initialisation.

La simulation du circuit sera le résultat du traitement de chaque cellule et de l'acheminement de tous les messages à l'intérieur du réseau. Les cellules ont donc deux tâches à accomplir, le traitement de la porte représentée

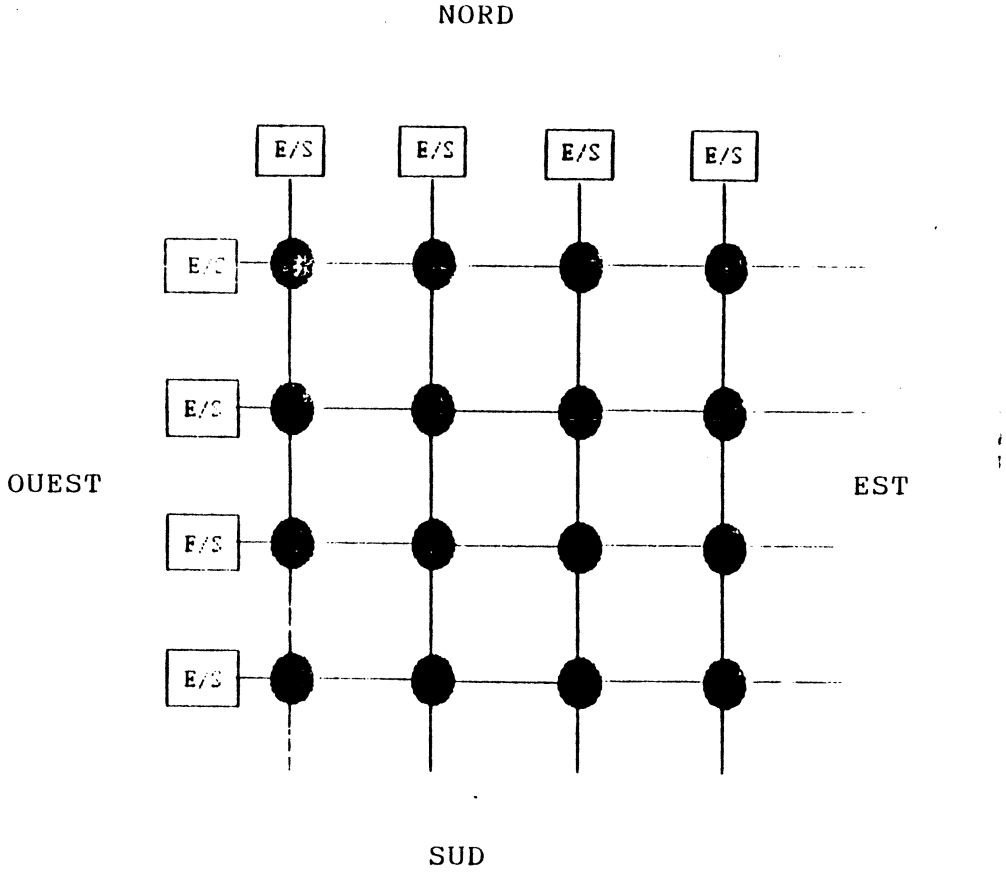
en fonction des messages reçus en entrée, et l'acheminement des messages jusqu'aux cellules destinataires. Cet acheminement conditionne la vitesse de simulation : sa complexité dépend de la connectivité du circuit à simuler, de la qualité de l'affectation du schéma logique au résau réalisé durant la phase de compilation, de la topologie des connexions du réseau et de l'algorithme d'acheminement implanté sur celui-ci.

La topologie retenue est la plus simple possible :
un réseau carré de type matriciel, chaque cellule communiquant avec ses quatre plus proches voisins.
Cette structure n'autorise que les communications sur deux directions orthogonales, ce qui simplifie le problème. Du point de vue de l'implantation, l'absence de lieu physique autres qu'horizontaux et verticaux rend la conception de la cellule et du réseau extrêmement simple.
Par la suite, il sera intéressant d'étudier d'autres topologies et de comparer entre elles, les différentes performances attendues pour la simulation.

L'accès au réseau depuis l'extérieur se fait par les cellules périphériques, puis de celles-ci aux cellules destinataires par le réseau lui-même.

La récupération des résultats se fait par le chemin inverse.

Une description topologique simple du réseau est la suivante :



Chaque cellule périphérique est reliée à un bloc d'entrée-sortie. Chaque cellule non-periphérique est reliée à ses quatre plus proches voisins.

Quatre directions d'acheminement sont possibles :
Nord, Ouest, Sud, Est.

Les déplacements sont répertoriés en X et en Y, positifs ou négatifs :

déplacement positif en X : Ouest vers Est
déplacement négatif en X : Est vers Ouest
déplacement positif en Y : Nord vers sud
déplacement négatif en Y : Sud vers Nord

Chaque cellule possède en mémoire les déplacements relatifs entre elle et ses successeurs. Ce déplacement est un couple de valeurs, déplacement en X, déplacement en Y (respectivement dX et dY).
Ce couple de valeurs est placé en tête du message envoyé dans la direction adéquate, déterminée par l'algorithme de routage implanté dans le réseau. Arrivant à une cellule, le couple de valeur (adresse du destinataire) est incrémenté suivant la direction de provenance.
Lorsque ce couple est nul, le message est arrivé à destination et traité par la cellule . Sinon il est réémis dans la direction adéquate, de manière à diminuer la distance de Manhattan, $|dX| + |dY|$, la séparant de son destinataire.

La procédure est la suivante :

Le message [dX, dY,a] est reçu par une cellule ;

S'il arrive par l'Ouest alors décrémenter dX ;

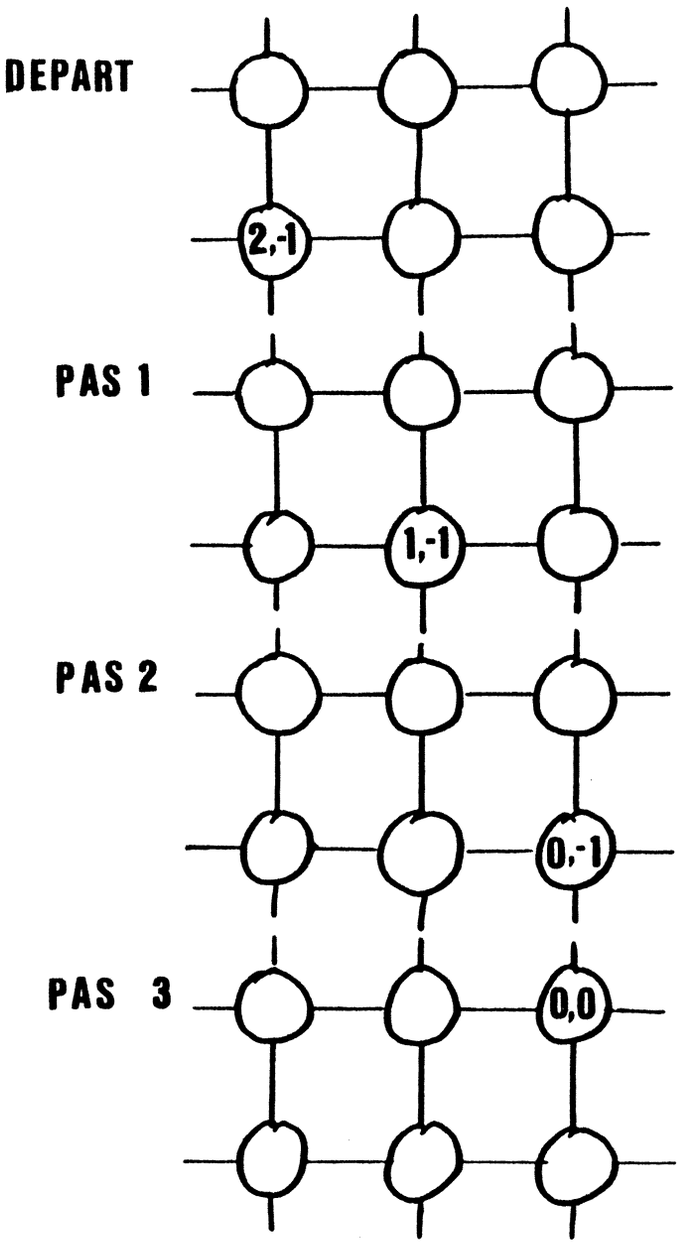
S'il arrive par l'Est alors incrémenter dX ;

S'il arrive par le Nord alors décrémenter dY ;

S'il arrive par le Sud alors incrémenter dY ;

Si dX et dY sont tous deux nuls, alors traiter le message.

Exemple d'acheminement :



La distance de Manhattan | dX | + |dY | détermine le nombre de pas de transmission (de cellule à cellule voisine) nécessaire à l'acheminement.
Sur cet exemple, trois pas sont nécessaires(|2| +|-1|) .
L'adressage relatif évite de donner un nom (ou coordonnées) à chaque cellule, de faire les comparaisons entre l'adresse du destinataire et le nom de la cellule traversée.
En contre-partie, il faut prévoir quatre dispositifs d'incrémentation-décrémentation par cellule. Mais la complexité de tels dispositifs est sûrement moins grande que celle d'un dispositif mémoire plus comparateur.

Les deux tâches à accomplir, acheminement et calcul induisent donc deux parties dans la cellule, une partie transmission et une partie calcul.
La partie transmission s'occupe du traitement de l'adresse du message reçu, de son décodage et de sa réemission éventuelle selon un dispositif d'aiguillage.
La partie calcul comprend un opérateur général, capable d'effectuer les fonctions logiques choisies, ainsi que la mémoire nécessaire au stockage des informations.

I. 2. FONCTIONNEMENT DE LA SIMULATION

Comme pour une simulation classique, on retrouve les différentes étapes :

- Compilation
- Initialisation
- Calcul
- Edition des résultats

I. 2-1. Compilation

La compilation comprend le prétraitement de la description d'entrée du circuit, la phase d'affectation dont nous avons parlé plus haut, la préparation des séquences d'entrée pour l'initialisation.

Le prétraitement consiste à adapter la description du circuit fournie par l'utilisateur à la structure de donnée du simulateur. Il convient de définir le type de simulation employée, de manière à préciser l'étude du réseau dans la simulation.
Le but étant l'étude et la définition de l'architecture du réseau, nous choisissons la simulation la plus simple possible, en logique 3 états (0, 1, X) pour tester et

affiner cette architecture. L'implantation de logiques plus précises avec un plus grand nombre d'états est tout à fait envisageable, et est prévue dans le cadre de l'extension des possibilités du réseau.

Nous avons choisi de limiter la connectivité des circuits simulables sur le réseau à une entrance maximale de quatre et une sortance maximale de quatre. Cette restriction doit permettre d'éviter un grand nombre d'embouteillages, les liens physiques conduisant à une cellule, étant au nombre de quatre.
De plus, il est apparu que dans les circuits de la bibliothèque utilisée, l'entrance moyenne est inférieure à trois la sortance moyenne voisine de trois. Les portes à plus de trois entrées sont donc assez rares.
Or le réseau comprenant N^2 cellules identiques prévoir des cellules suffisamment complexes pour traiter des portes à entrance et sortance élevées, c'est augmenter de façon considérable la taille du réseau tout entier en diminuant son pourcentage d'utilisation.

Il apparaît donc plus intéressant d'éclater au prétraitement ces grosses portes en plus petites, de manière à obtenir une utilisation optimale du réseau et à essayer d'uniformiser la densité de l'acheminement.

Il faut noter que ce type de limitation existe aussi sur les gros simulateurs, l'entrance étant limitée à quatre dans la machine de YORKTOWN et à trois, dans la machine de ZYCAD. .

Les fonctions logiques de base utilisées sont les huit suivantes, codées sur trois bits, un bit supplémentaire indique si la cellule représente ou non une porte logique:

Ramification, Inverseur, ET, OU, NON-ET,
NON-OU, NON-OUX, OUX.

La ramification n'effectue aucune fonction logique, mais permet de représenter un retard seul (ligne de transmission par exemple), ou de représenter avec une autre cellule une porte à sortance supérieure à quatre. .

Les retards utilisés peuvent être nul, unitaire, de deux ou trois unités de temps du simulateur. Ces quatre retards sont codés sur deux bits.
Le retard d'une porte peut-être traité par la même cellule associée à la porte, dans la mesure où il ne dépasse pas trois unités de temps.
Pour des retards importants, on peut utiliser plusieurs cellules en combinant un type ramification, avec un retard cellulaire.

La machine hôte traite la description du circuit à simuler pour le rendre compatible avec les considérations ci-dessus. Une fois les éléments bien caractérisés, la phase d'affectation est résolue par un programme de placement.

Le problème du placement est assez critique, puisqu'il conditionne la longueur des chemins à parcourir, et donc le temps de simulation.

Les essais et choix retenus sont décrits dans le chapitre suivant. Une fois le placement effectué, les informations nécessaires à la programmation de chaque cellule devront être traitées par les séquences d'initialisation.

I. 2-2. Initialisation

Chaque cellule doit posséder en mémoire des informations sur la porte représentée et ses successeurs.

Ces informations sont :

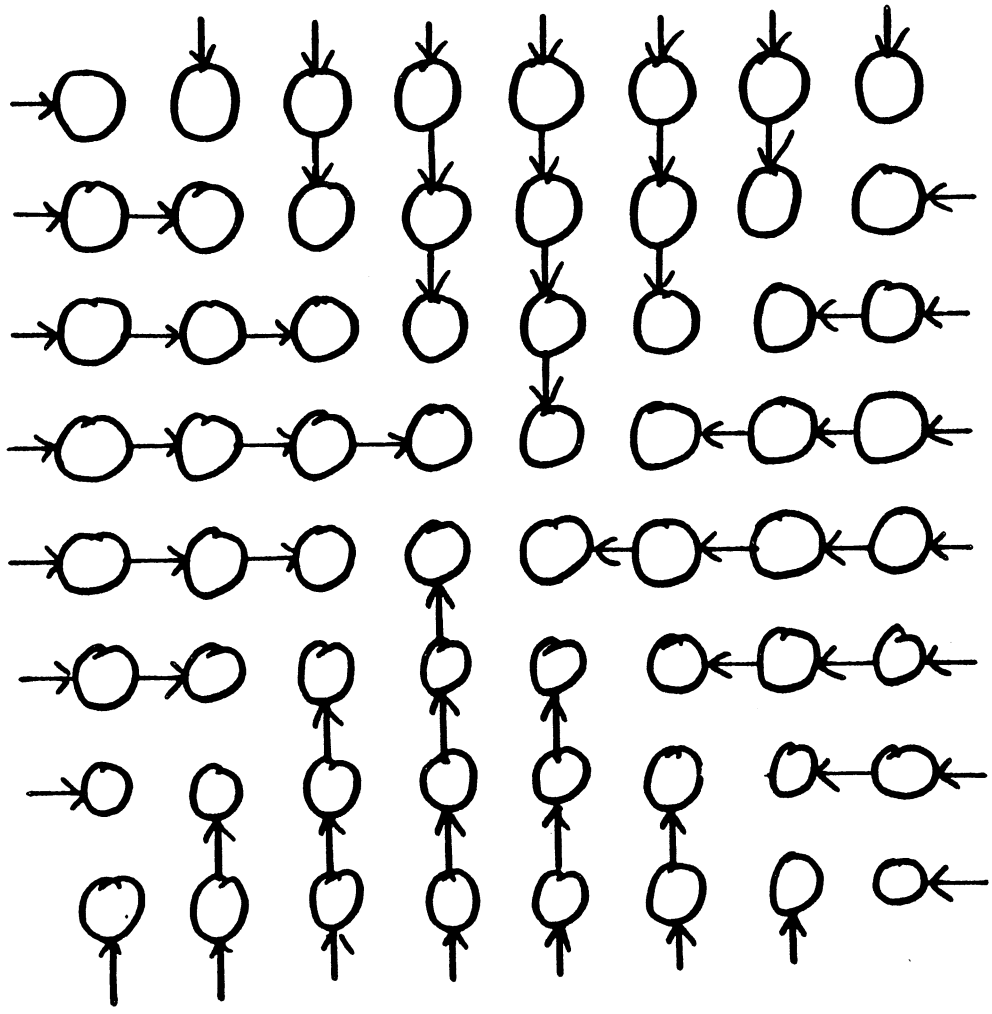
- Le type logique
- Le retard associé
- Le nombre d'entrée, c'est-à-dire le nombre de messages à recevoir.

- Le nombre de successeurs, c'est-à-dire le nombre de messages à envoyer.
- Les adresses des destinataires, c'est-à-dire au maximum quatre fois dX et dY.

La machine hôte indiquera au réseau s'il s'agit de l'initialisation ou de la simulation. Ceci permet d'utiliser le réseau d'acheminement lui-même pour l'initialisation.

Les messages comprenant deux bits d'information, la programmation d'une cellule nécessite le changement d'un certain nombre de bits répartis en plusieurs séquences d'initialisation. Le changement s'effectue sur toutes les cellules périphériques depuis l'extérieur. Une séquence atteint toutes les cellules en un temps de transmission interne de $\left(\frac{N}{2} - 1\right)$ pas.
Le changement s'effectue de manière extrêment rapide. Il n'y a aucun conflit dans le routage, car les flots de communication ne se joignent, ni ne se coupent.

Le principe du routage est exposé ci-dessous :



routage de l'initialisation d'un réseau 8 x 8.

Un message est routé à chaque cellule en trois pas de transmission interne au maximum. 64 messages sont chargés dans le réseau en trois pas de transmission interne.

I. 2-3. Calcul

La simulation est gérée par la machine hôte. Elle seule contrôle l'heure de simulation.
L'hôte envoie les séquences d'entrée sous forme de messages aux cellules périphériques. Une fois dans le réseau, les messages sont véhiculés par un routage normal semblable à celui de l'initialisation. L'hôte délivre au réseau un signal GO autorisant l'envoi des messages à travers le réseau.
Lorsque les messages sont arrivés à destination, le calcul se fait dans chaque cellule concernée, puis les nouvelles valeurs de sortie sont placées dans les messages à envoyer.
Lorsque tous les messages sont arrivés et traités de cette façon, le réseau répond à l'hôte un signal END indiquant qu'il est libre pour la suite de la simulation. Le réseau n'a pas conscience de l'heure de simulation globale.

L'intervalle entre un signal GO et le signal GO suivant constitue un pas de simulation. Durant ce pas, toutes les portes ont reçu leurs valeurs d'entrée et toutes ont été remises à jour. Le réseau effectue donc de

manière autonome un pas de simulation du circuit tout entier. C'est le rôle de la machine hôte de récupérer les résultats et de constituer l'ensemble de la simulation.

Les différences avec une simulation classique sont les suivantes :
Durant un pas de simulation tous les évènements sont traités. Il n'y a pas besoin d'échéancier pour ordonner les évènements.
Les valeurs de sortie sont automatiquement envoyées à chaque pas de simulation, même si elles sont égales aux valeurs précédentes. Ceci fait que le routage est le même à chaque pas de simulation.
Dans un simulateur classique, seuls les changements des valeurs de sortie induisent des évènements de manière à réduire le nombre d'évaluations et de manipulation d'échéancier. Cette réduction est intéressante lorsque les opérations s'effectuent séquentiellement.
Dans le cas de notre réseau, chaque cellule attent un nombre de messages déterminé, indépendamment du contenu de l'information. Il faudrait donc envoyer un message indiquant que rien n'a changé. Le routage ne serait donc pas simplifié.

Le fait d'envoyer quand même les valeurs d'entrée identiques aux précédentes, fait que le calcul va se refaire dans un certain nombre de cellules, alors que la valeur de sortie est déjà stockée. Ceci n'entraîne pas d'augmentation du temps de calcul global car les calculs dans les cellules s'effectuent de manière indépendante les uns des autres, et n'ont pas d'influence sur les temps d'acheminement.

La durée d'un pas de simulation dépend principalement du chemin le plus long à parcourir dans le réseau. Seule une simplification du routage peut diminuer le temps global de calcul.
Dans le cas où on envisage l'envoi des seules valeurs ayant changé, le nombre de messages acheminés serait réduit, mais la complexité et la taille des cellules seraient accrues par le fait qu'il faudrait un dispositif mémoire supplémentaire (stockage des valeurs précédentes) et un système de réception plus complexe.
Le temps de traversée d'une cellule peut alors être augmenté de façon considérable.

I. 2-4. Exploitation des résultats

Les valeurs de sortie du circuit doivent être récupérées par la machine hôte. En plus du routage représentant la connectivité du circuit à simuler, l'affectation doit tenir compte des portes de sortie du circuit, ainsi que des connexions à visualiser et prévoir la connexion de ces portes aux cellules périphériques les plus proches. Ceci crèe donc pour chaque sortie du circuit ou pour toute autre porte intéressant l'utilisateur des successeurs supplémentaires n'ayant aucune signification physique.Il n'y a pas lieu de différencier les successeurs supplémentaires de ceux prévus dans le schéma logique, la machine hôte faisant la différence.

Ces valeurs, stockées par la machine hôte en fonction des instants de simulation, permettent une édition des résultats tout à fait classique.
Le désir par l'utilisateur d'observer différentes portes à l'intérieur du circuit, fait qu'il faudra prévoir toutes les portes intéressantes dès la phase de compilation, ou alors remodifier celle-ci et relancer une simulation globale.

Une autre possibilité qui pourrait être offerte par le réseau, serait de permettre de stopper la simulation à un instant donné, et de sortir l'état du circuit complet à cet instant. Il suffit pour cela, à l'instant de simulation considéré de récupérer le END du réseau, de ne pas envoyer le GO suivant, et d'ordonner le rapatriement complet des valeurs de sortie de toutes les cellules selon le processus inverse de l'initialisation. Ceci permettrait d'avoir des renseignements sur l'état complet du circuit, à tout instant intéressant de la simulation.

CHAPITRE II

AFFECTATION PLACEMENT

II.1. INTRODUCTION

Le problème de l'affectation est primordial dans cette étude, car de celle-ci dépend la qualité de l'acheminement et les performances du réseau dans la simulation.

Le but à atteindre dans la conception de ce réseau, est une rapidité importante de la simulation, ainsi qu'une implantation facilitée par l'utilisation d'une cellule de base simple.Il ne doit pas y avoir une disproportion trop grande entre les phases préparatoires (initialisation et compilation) et la simulation proprement dite. Or les algorithmes de placement peuvent être très complexes et ils constituent d'ailleurs un domaine d'activité à part entière dans la CAO. Un compromis doit être trouvé; nous essaierons de nous tenir à des algorithmes simples.

La difficulté dans notre cas est la définition d'un 'bon' placement, et de trouver les critères de décision permettant le choix de celui-ci.
L'étude initiale est de définir un placement suffisam-

ment rapide et simple, de manière à pouvoir étudier ensuite les problèmes d'acheminement sur des exemples de circuit , ce qui permet alors de définir et concevoir une première architecture pour le réseau. Il sera ensuite intéressant de revenir sur le problème de placement pour étudier son influence sur les performances de la machine.

Dans les études traditionnelles faites sur les problèmes de placement, l'objectif est généralement la minimisation de la longueur totale des interconnexions, ou de leurs densités (nombre d'interconnexions franchissant les lignes placées entre les composants) [BRE 77], [DUP 84].

Le placement comporte généralement deux phases:

placement initial, et amélioration.

Il existe des placements initiaux à croissance épitaxiale [HAN 73], à relaxation [HAN 72] pour la minimisation de la longueur totale des interconnexions; des placements initiaux à bi-partitionnement [BRE 77], [PRE 78], à linéarisation [KAN 83], pour la minimisation des densités d'interconnexion.
Des algorithmes et heuristiques d'améliorations locales itératives d'échanges par paires existent également [GOT 79],[LAU 79].

Pour notre placement,le circuit est représenté par un graphe non-orienté de N élements interconnectés.
La grille d'affectation comporte H x L sites Le problème est donc d'associer les N éléments du graphe à N sites, avec N inférieur ou égal à H x L.
Le principe de base retenu est de placer N éléments sur un cadre rectangulaire, de dimension H x L, de forme proche d'un carré, de manière à minimiser le nombre de cellules inutilisées.

Avec ce principe de base, deux approches sont possibles concernant le placement d'un gros circuit :

- Soit placer le circuit éclaté tout entier sur la grille.

- Soit placer le circuit suivant la découpe hiérarchisée, en blocs fonctionnels, le placement des blocs élémentaires étant effectué suivant le principe de base.

L'étude de ces deux types de placement a été effectué en prenant comme exemples des circuits décrits dans le système de CAO CASSIOPEE [CNE 84].

II. 2. PLACEMENT DU CIRCUIT ECLATE

II.2-1. Dimensionnement de la grille d'accueil

Le schéma logique du circuit comprend N éléments .
La grille d'accueil a les dimensions H x L.

On doit donc placer N éléments sur la grille H x L.
Afin d'éviter d'avoir trop de cellules inutilisées, nous avons choisi d'utiliser un réseau H x L de taille immédiatementsupérieure ou égale à N, de forme rectangulaire proche d'un carré.
La procédure est la suivante :

Si $\sqrt{N}$ entier alors H = L = $\sqrt{N}$

Si $\sqrt{N}$ n'est pas entier alor faire

début

Si $(E[\sqrt{N}])x(E[\sqrt{N}]+1)>,N$ alors $H=E[\sqrt{N}]$

$L= H+1$

sinon $H=L=E[\sqrt{N}]+1$

fin

Donc pour placer un circuit de 2499 portes ou de 2500, on utilise une grille 50x50. pour un circuit de 2501 portes, on utilise une grille 50x51.

II. 2-2. Placement

Dans la description d'un circuit en CASSIOPEE, on distingue trois types de portes :

Point d'entrée : Type = 2

Point de sortie : Type = 3

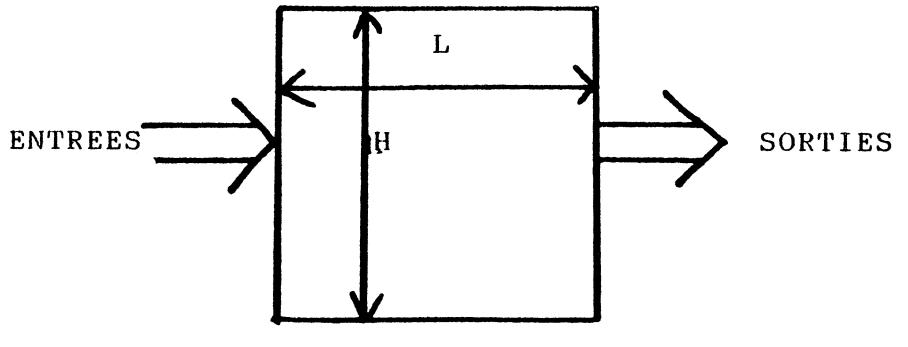
Porte logique : Type > 3

Le placement comporte deux phases :

a) Placement initial des premiers objets choisis arbitrairement.

b) Placement constructif des objets suivants selon les liens de connexions entre ceux-ci et les précédents.

a) Pour le placement initial, les objets choisis arbitrairement sont les points d'entrée et de sortie. Les points d'entrée sont positionnés sur le côté Ouest de la grille, les points de sortie sur le côte Est.
On définit ainsi un sens global de propagation dans le réseau de l'Ouest vers l'Est :



L'algorithme est le suivant :

Placement initial

Lire et compter les points d'entrée.

Si leur nombre Nb est inférieur à H, alors faire

```
début
    I = 0
    tant que I ≤ Nb faire
        début
            J = 1
            Placer le Point I sur le site [ 1,J]
            J = J + 1
        fin
fin
```

Si leur nombre Nb est supérieur à H, alors faire

```
début
    n = E [ Nb / H ]
    Pour k de ( I-1 )x H à I x H faire
        début
            Site [ I,J ] = Kième point d'entrée
            J = J + 1
        fin
fin
```

Les points d'entrée sont donc placés séquentiellement sur la première colonne de haut en bas, puis sur la deuxième... Le placement des points de sortie s'effectue systématiquement à partir de la dernière colonne, de haut en bas, puis de l'avant-dernière ....

b) Placement constructif

On veut placer les portes connectées à la porte I précédemment placée. La procédure est la suivante :

```
Lire la porte I
Lire ses interconnexions
Pour toutes les portes connectées à I faire :
    début
        Si la porte J (reliée à I) n'est pas
        placée  alors la placer
        Décrémenter le nombre de portes Non-pla-
        cées
    fin
Si toutes les portes sont placées alors STOP.
```

On lira successivement la porte du site [1, 1] (première ligne, première colonne), puis du site [L,1] (dernière colonne, première ligne), puis du site [1,2], puis du site [L,2].....

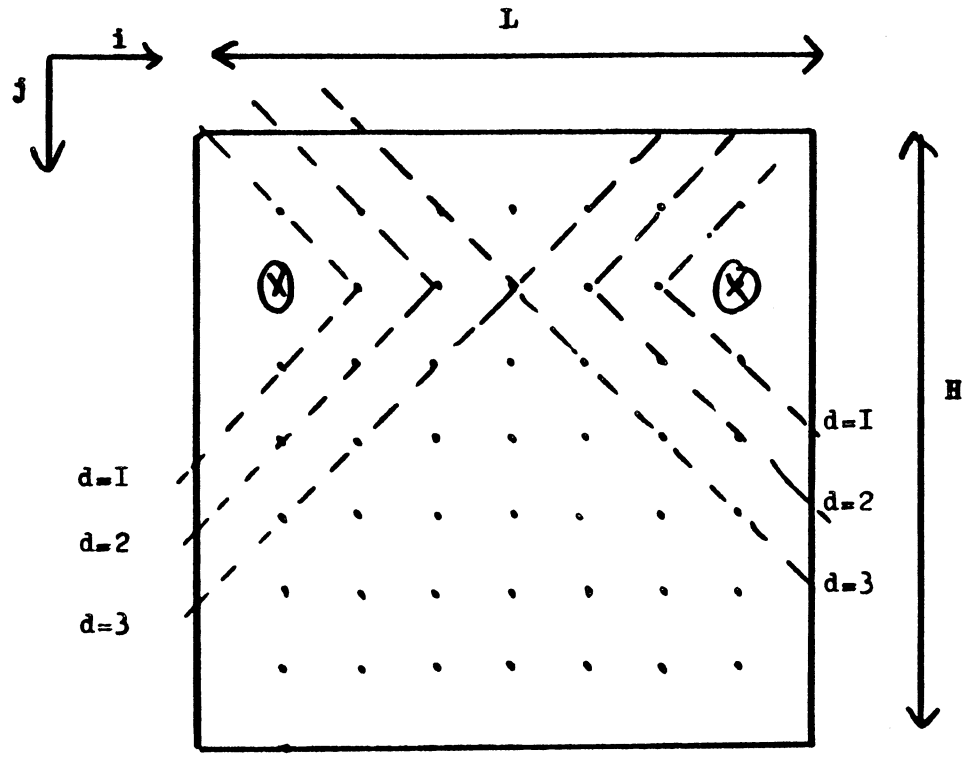
La recherche d'emplacement vide se fait donc alternative-

ment à partir de l'Ouest et à partir de l'Est de la grille. La procédure de recherche d'emplacement vide doit rechercher le site vide de coordonnées [I', J'] situé à la distance de Manhattan la plus faible du site [I, J] (associé à la porte n) afin d'y affecter la porte n' connectée à n.

La procédure est la suivante :

```
d est la distance de Manhattan
( d= | I- I' | + | J- J' | )
d = o
tant que (d inférieur ou égal à d max) et (stop
égal zéro) faire
  début
    d = d + 1
    dJ =  - 1
    dI = d + 1
    tant que dI  # 0 faire
      début
        dI = dI - 1
        dJ = dJ + 1
        I' = I ± dI
        J' = J ± dJ
        Si le site [I',J'] est vide alors
              début
                Site [I',J' ]=n'
                Stop = 1
              fin
      fin
  fin
```

Schéma de remplissage de la grille :



Une amélioration itérarive [SCH 76] peut permettre d'améliorer la qualité du placement. Cependant, la complexité de ces algorithmes est très supérieure à celle du placement constructif et le temps de compilation peut alors devenir disproportionné au temps de caclul.
Le travail de l'amélioration du placement n'a pas été entrepris ici. Nous nous sommes restreints à un placement élémentaire permettant d'étudier les influences du placement sur la qualité de la simulation.

Exemples de placements et de routages obtenus

Bascule D : la description de la bascule comprend 10 élé-

ments avec les points d'entrée-sortie. Elle est placée sur une grille 4 x 3.

Les entrées sont sur le côté Ouest, les sorties sur le côté Est, assurant une direction globale de communication Ouest vers Est.

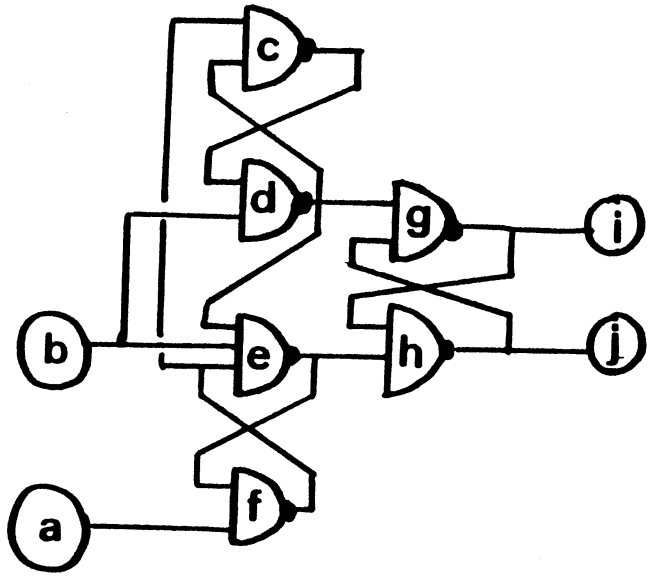
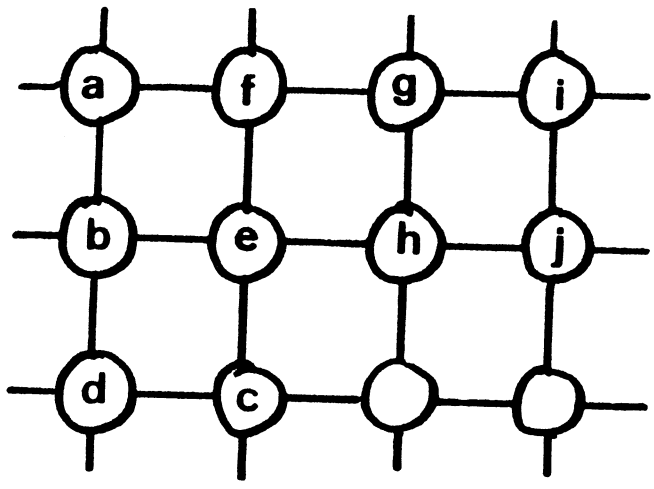
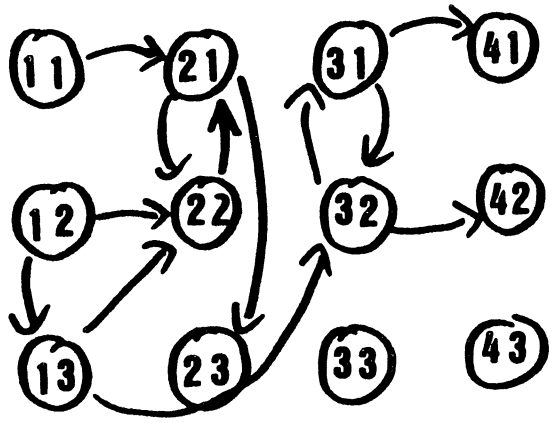


Schéma logique de la Bascule D



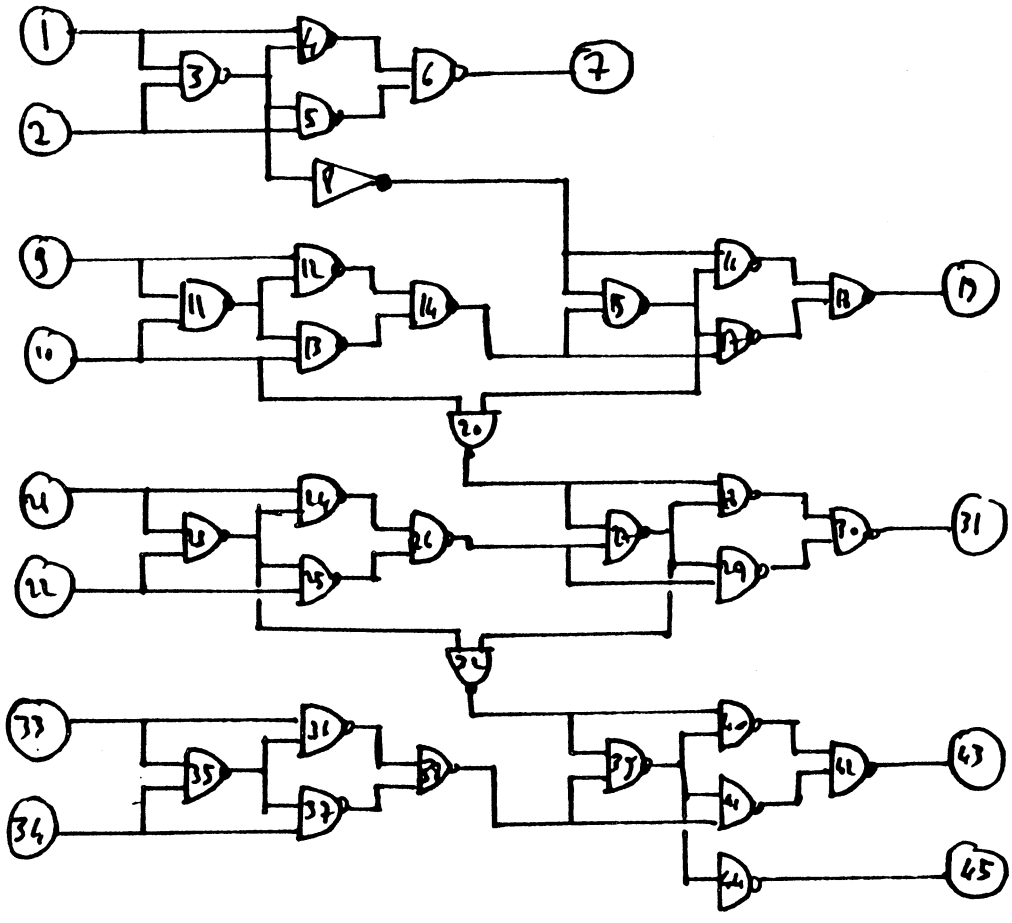
Placement associé : les points d'entrées-sortie a,b,i,j, sont la semence du placement constructif.

Le flot de communication est :

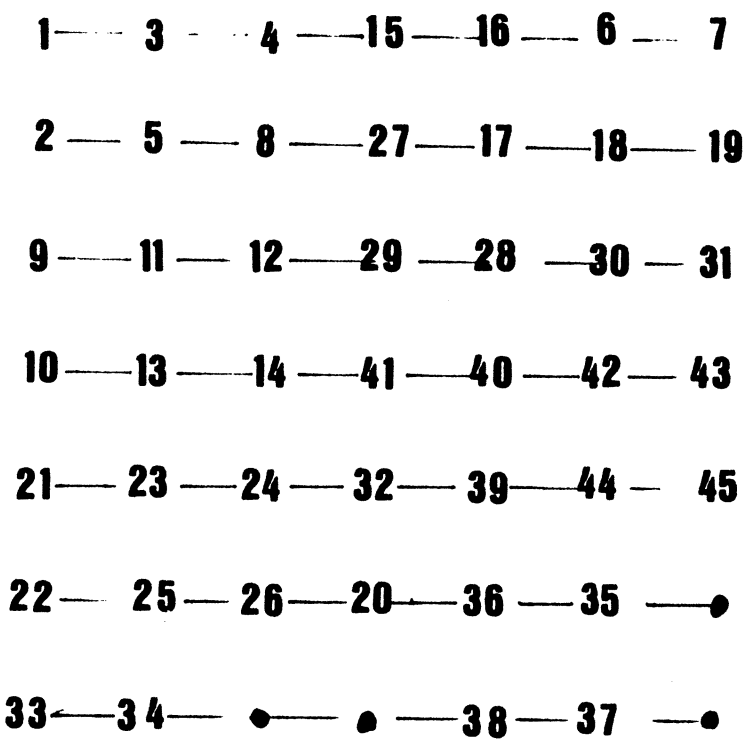


Additionneur 4 bits

Le Schéma logique contient 13 points d'entrée-sortie et 45 éléments au total.



Le placement associé sur une grille 7 x 7 est le suivant:



c) Performances du placement constructif:

Le placement décrit ci-dessus est du type à croissance épitaxiale. Les points d'entrée-sortie servent de semence au programme. Ce placeur a été implémenté en Pascal sur HB - 68 Multics.

Le programme a été testé sur différents exemples de circuit. Le temps de placement obtenu dépend du nombre de portes du circuit ainsi que de la complexité de ses interconnexions.

Plus le circuit est grand, plus la grille d'affectation l'est, et plus la distance de Manhattan pour la recherche des sites vides peut s'incrémenter.

Nous avons pu mesurer le temps de placement en fonction du nombre de portes et de la connectivité moyenne du circuit. La connectivité moyenne du circuit a été calculée en divisant la somme des entrances des portes par la mesure des portes :

$$E\,[\,\text{Comex}\,] = \frac{\sum_{i=1}^{n} E_i}{N}$$

Le tableau suivant indique les mesures relevées :

Nombre de Portes	Connectivité Moyenne	Temps CPU sur Multics
29	1,5	25 msec.
60	1,6	70 msec.
163	2,7	1,2 sec.
169	1,4	0,5 sec.
900	1,3	9 sec.
900	2,8	17 sec.
2.500	1,3	74 sec.
2.500	3,2	133 sec.

II.3. PLACEMENT DU CIRCUIT PAR BLOCS HIERARCHISES

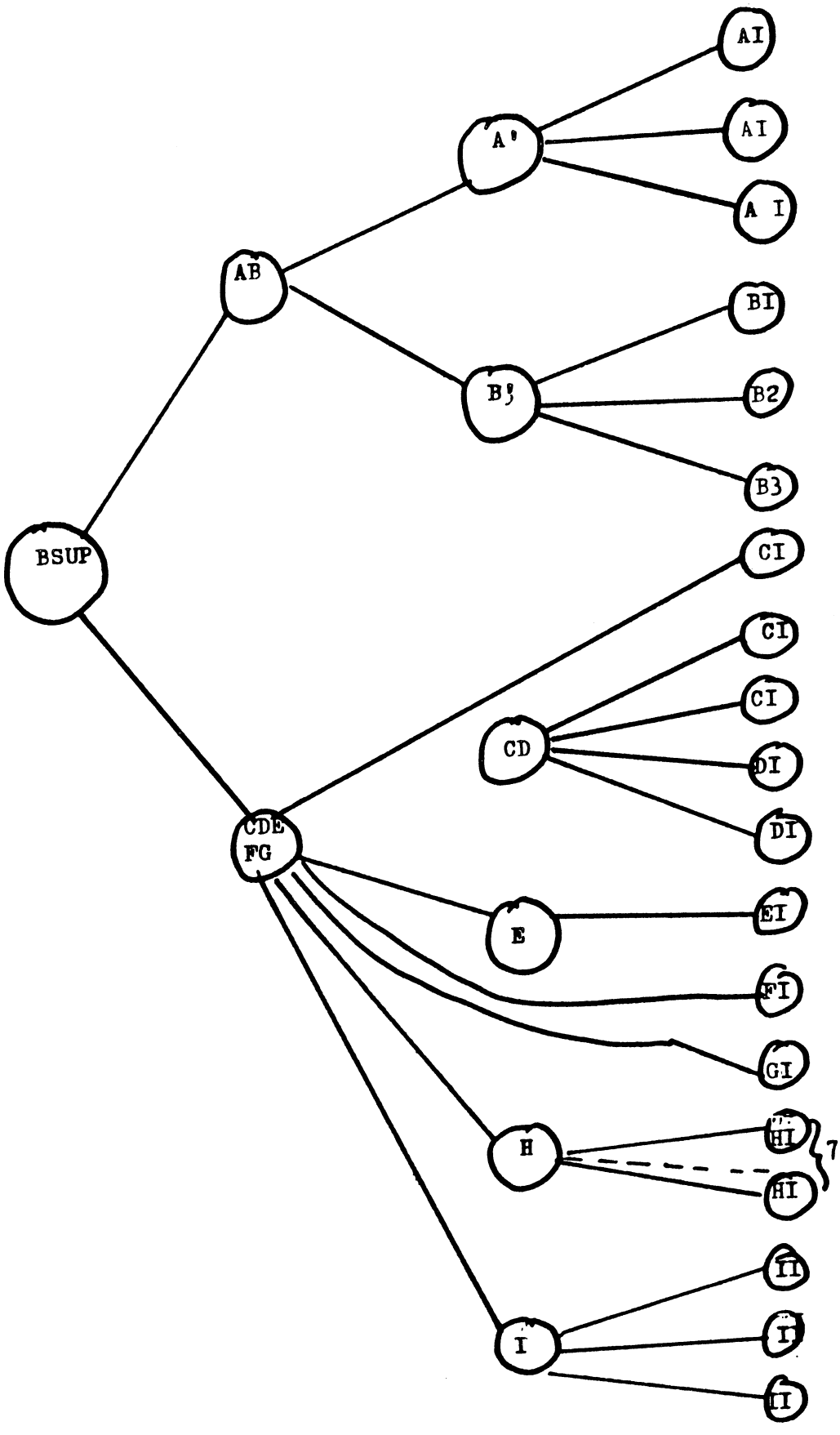
Un circuit logique est décrit sous forme hiérarchisée par différents blocs.

Un circuit élémentaire est composé de portes logiques de base et de portes fantômes d'entrée et de sortie.

Un circuit plus complexe contiendra des points d'entrée et de sortie, des portes logiques élémentaires du type ci-dessus, et même d'autres circuits complexes.

La description d'un circuit complet peut donc se mettre sous forme arborescente.

Exemple : circuit B S U P (CNET) de 1255 portes décrit dans CASSIOPEE comportant 4 niveaux hiérarchiques.



Les feuilles de l'arbre représentent les circuits élémentaires décrits en terme de portes logiques.
Les noeuds sont des éléments logiques construits avec des points d'entrée-sortie, d'autres circuits noeuds et des circuits feuilles.

Une méthode intéressante peut être de suivre cette description hiérarchisée pour le placement, c'est-à-dire :

- de placer les circuits feuilles dans des blocs rectangulaires avec les algorithmes décrits dans le placement éclaté.
- de placer les circuits noeuds en assemblant les blocs feuilles et noeuds.

Le problème est que les blocs peuvent être de taille très différente. Il en résulte beaucoup de place perdue au niveau du réseau (beaucoup de cellules inutilisées).
De plus, l'assemblage de blocs de taille et forme différentes est extrêmement complexe.
Il existe des algorithmes d'assemblage de blocs, mais cela ne concerne que des blocs pas trop différents [LAU 79].

Un inconvénient de la description est que l'on peut avoir dans un même bloc des portes logiques, donc de dimension

Un sur la grille d'affectation, et des circuits blocs, donc de dimension Deux; l'assemblage des deux types étant assez incertain.

Le circuit décrit ci-dessus, 1255 portes a été assemblé à la main, suivant sa découpe en blocs.

On assemble d'abord les blocs élémentaires (circuits feuilles):

Nom du circuit	Nombre de portes	Dimension du bloc
A 1	10	4x3
B 1	24	5x5
B 2	26	5x4
B 3	28	6x5
C 1	25	5x5
D 1	21	5x5
E 1	8	3x3
F 1	60	8x8
G 1	81	9x9
H 1	43	7x7
I 1	21	5x5

On assemble ensuite les blocs intermédiaires :

A'	31	11x5
B'	128	28x6
CD	112	24x6
E	25	6x5
H	360	69x8
I	69	18x5
AB	159	28x12
CDEFG	1096	74x27
BSUP	1245	74x40

Le circuit BSUP est alors placé sur une grille 74x40. Seulement 51% des cellules sont utilisées.
Le dernier assemblage consiste à réunir en un seul bloc. les circuits AB et CDEFG. Or AB et 6 fois plus petit .
Cette étude manuelle nous a donc donné un premier aperçu des résultats que l'on peut attendre d'un placement hiérarchique. Si l'idée paraît séduisante, son application (au moins dans le cadre du circuit étudié) s'est avéré très décevante au niveau de la place utilisée.
De plus cette méthode paraît difficilement automatisable avec des blocs trop différents.
La méthode de placement constructif du circuit éclaté paraît donc plus simple, malgré l'éclatement de la description des circuits.

CHAPITRE III

LE RESEAU CELLULAIRE

III. 1. INTRODUCTION

Nous avons déjà vu plus haut que le réseau effectue deux tâches distinctes, l'acheminement des messages depuis l'expédition jusqu'au destinataire, et le calcul à l'intérieur de la cellule destinataire.

Ceci implique deux parties distinctes dans la cellule de base, une partie transmission et une partie calcul.

Les problèmes d'acheminement et les solutions appliquées induisent les différents blocs fonctionnels de la partie transmission, ceux et celles du calcul induisent les blocs de la partie calcul.
Nous allons passer en revue les différents problèmes et indiquer les choix retenus pour l'architecture de la cellule.

III.2. ACHEMINEMENT ET PARTIE TRANSMISSION

Dès l'autorisation de l'hôte par l'intermédiaire du signal G O , les cellules concernées envoient leurs

messages dans le réseau.

Ceux-ci vont se propager et charger certaines voies de communications.
Différents dispositifs chargés de l'accueil des messages, de leur décodage, de leur réemission éventuelle sont donc à prévoir. Ils doivent respecter les contraintes suivantes:

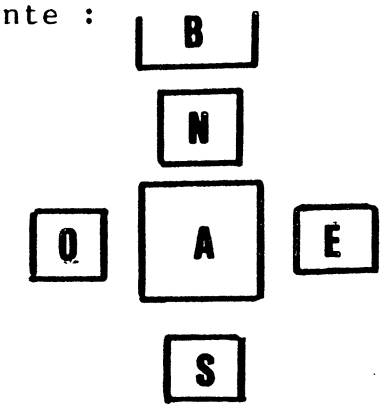
- Avoir une complexité et une taille aussi faible que possible.
- Permettre un acheminement fiable et rapide.

III.2-1.Registres Tampons et Traitement des adresses

Le premier problème à résoudre est le stockage des messages dans l'attente de leur décodage et de leur transfert. Nous avons pensé entourer les cellules de registres tampons permettant aux messages, la traversée des cellules.

Une transmission élémentaire consiste donc au transfert d'un message, d'un registre tampon entrée d'une cellule dans le registre tampon, sortie de la même cellule, dans la bonne direction.
La solution la plus simple consiste à associer quatre registres tampons, un par face, à la cellule.

Ces registres Nord, Ouest, Sud, Est, sont alors disposés de la manière suivante :

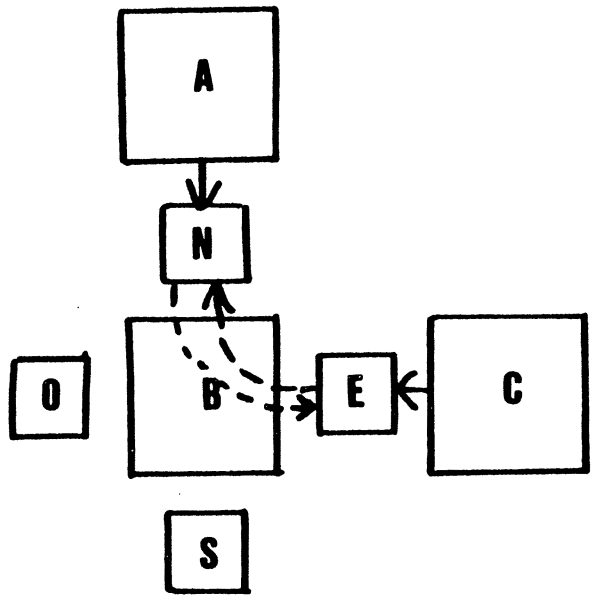


Le registre tampon N est un registre Nord pour la cellule A et registre Sud pour la cellule B.

Un pas de transmission peut consister au transfert d'un message du registre N au registre O à travers la cellule A.

Mais cette solution à un seul tampon par face, bien qu'économique au niveau de la taille de la cellule, conduit facilement à des blocages.

Un exemple de blocage élémentaire est le suivant :

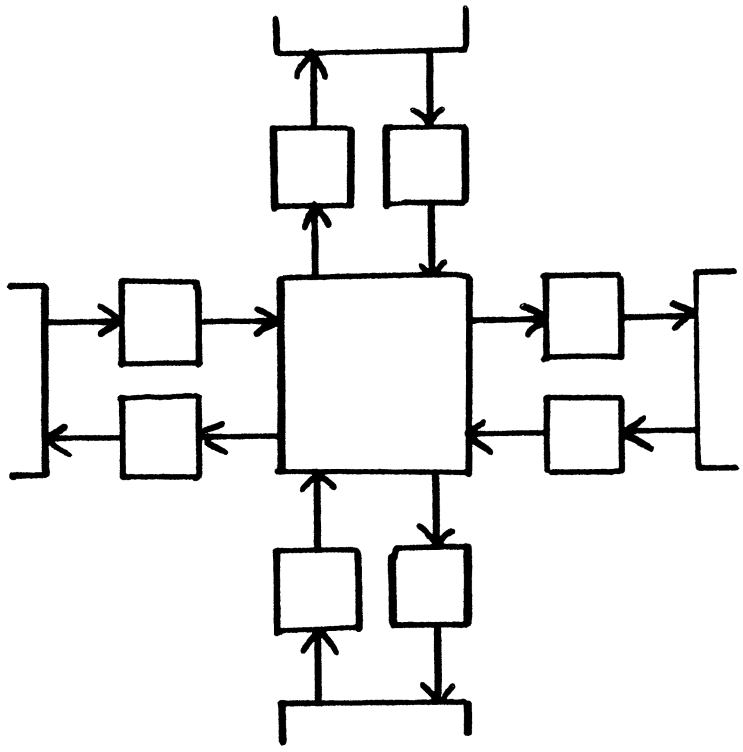


Si au même instant la cellule A charge le registre N par un message destiné ensuite à C, et C charge le registre O par un message destiné à A, les deux registres sont donc occupés.
Or la cellule B va orienter le message contenu dans N vers O et le message contenu dans O vers N. Sans l'adjonction d'un registre intermédiaire supplémentaire interne à B, cela produit un blocage.
Une solution pourrait être d'envisager un routage adaptatif, c'est-à-dire des détournements, en cas de voie de passage occupé Cette solution implique un dispositif de transmission assez complexe, donc une taille de cellule assez grande. De plus, un détournement allongera la longueur du chemin à parcourir, et donc le temps nécessaire à l'acheminement.
Cette idée présente donc des inconvénients que nous avons jugés trop importants et qui nous ont fait abandonner la solution de routage adaptatif.

Nous nous en tiendrons à des acheminements qui décrémentent la distance de Manhattan à chaque transmission élémentaire.

Nous avons choisi de dédoubler le nombre de registre

tampons en associant sur chaque face deux registres, un pour chaque sens de communication.



Cette configuration permet d'éviter les problèmes de blocage et de diminuer la probabilité de goulot d'étranglement.

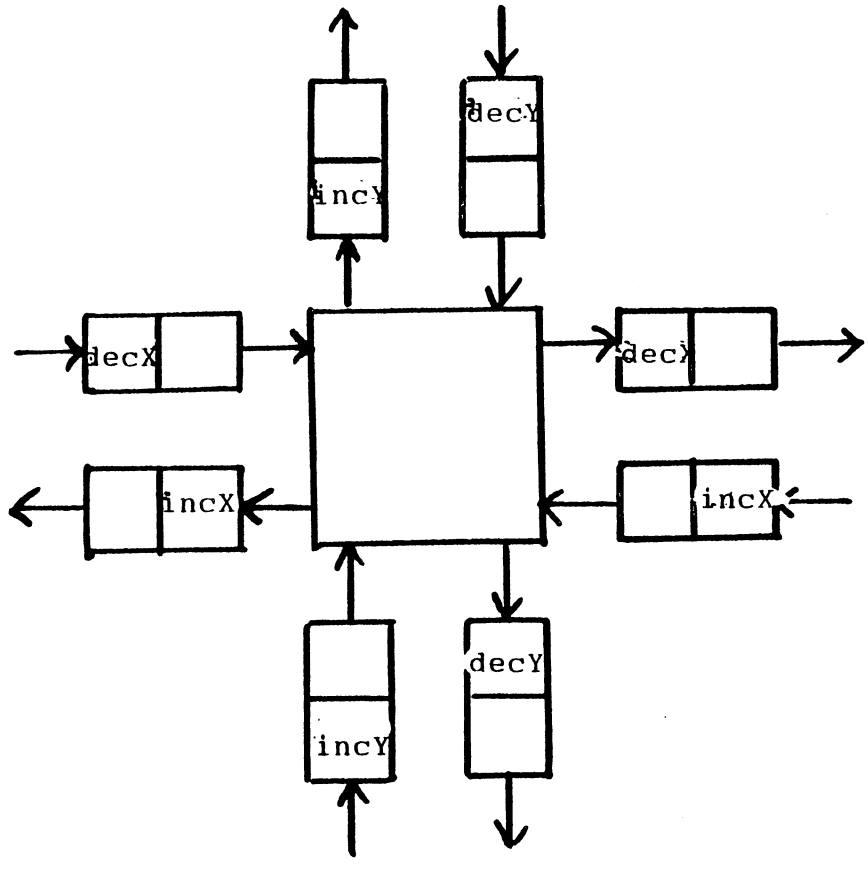
Au niveau de la taille, les dimensions de la cellule (tampons exclus) devraient être suffisamment grandes pour permettre de positionner deux registres tampons le long d'une face. La taille de la cellule serait alors supérieure en nombre de transistors à la solution à un seul tampon par face, mais la surface globale du réseau ne serait pas modifiée à cause des contraintes topologiques d'accolage des cellules.

Les différentes étapes d'une transmission élémentaire sont les suivantes :

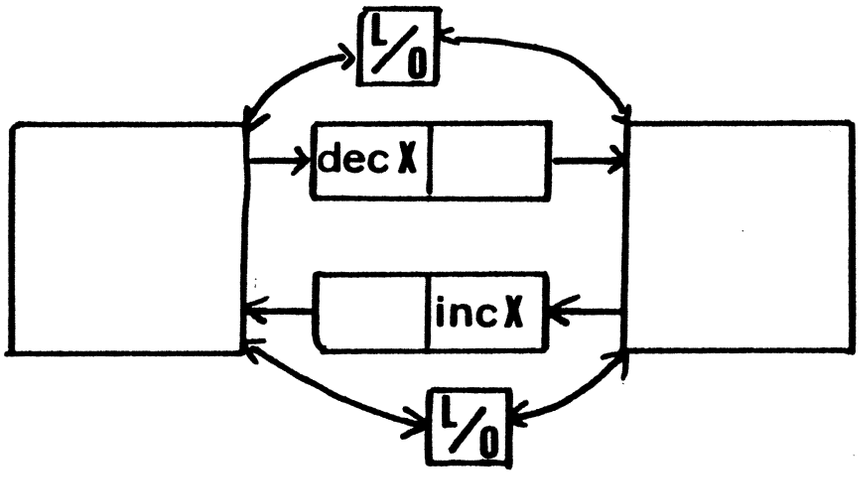
- Arrivée d'un message dans un registre d'entrée et stockage.
- Traitement de l'adresse de destinataire (incrémation ou décrémation).
- Choix du registre à lire, si plusieurs demandent l'accès simultanément à la cellule.
- Décodage du message.
- Aiguillage et écriture dans le registre de sortie adéquat.
- Décision, si la voie de sortie est occupée.

Le stockage du message est effectué par les registres tampons.
Huit registres sont nécessaires à une cellule, deux sur chaque face. Chaque registre est commun à deux cellules: il sert d'entrée pour l'une, de sortie pour l'autre.
Le dispositif du traitement de l'adresse peut être directement inclus dans chaque registre puisque le traitement ne dépend que de la dernière direction de provenance:

- Au registre de sortie Est, est associé un décrémenteur en d X , D E C X.
- Au registre de sortie Sud, est associé un décrémenteur en d Y , D E C Y.
- Au registre de sortie Nord, est associé un incrémenteur en d Y , I N C Y.
- Au registre de sortie Ouest, est associé un incrémenteur en d X , I N C X.



A chaque registre est associé un dispositif Drapeau indiquant 'libre ou occupé'. Ce Drapeau est testé en permanence par la cellule qui sait alors si elle peut lire ou non le registre d'entrée, et écrire ou non dans le registre de sortie.



Si le Drapeau indique 'libre', la cellule A peut écrire dans le registre tampon. Une fois le message chargé, le Drapeau est positionné à 'occupé'. Le signal 'occupé' déclenche dans la cellule B le processus de lecture. Une fois la lecture effectuée, la cellule B remet le Drapeau 'libre'.

III.2-2. Règles de priorités

Une cellule peut recevoir des messages de quatre directions différentes. Plusieurs registres d'entrée peuvent demander l'accès à la cellule en même temps.

Afin d'éviter les conflits, il est nécessaire d'établir des règles de priorités . Ces règles sont implantées dans un dispositif interne à la cellule, un codeur de priorités.

Les règles de priorités concernant les quatre directions géographiques sont purement arbitraires. Par contre, au début du pas de simulation, les cellules peuvent avoir un ou plusieurs messages (quatre au maximum) à envoyer sur le réseau.
Ces messages proviennent de la partie calcul. Il est souhaitable qu'ils soient le plus rapidement possible sur les voies de communication du réseau. Ces messages seront donc prioritaires sur ceux en transit.

D'après ces considérations, les règles de priorités sont alors les suivantes :

- Si plusieurs messages sont en attente, alors la priorité est :
  Registre Calcul > Nord > Ouest > Sud > Est.
- Si tous les registres sont vides, alors le premier arrivé est introduit dans la cellule et inhibe les autres. Lorsque ce message est transis, l'inhibition est retirée.

III.2-3. Aiguillage

L'aiguillage consiste à décoder un message entrant, l'orienter dans la bonne direction selon l'algorithme d'acheminement, et le charger dans le registre de sortie. Il est effectué par un dispositif aiguilleur.

Les possibilités de 'routage adaptatif' ayant été abandonnées, les algorithmes testés décrémentent à chaque pas de transmission la distance de Manhattan. En cas de voie occupée, la demande du message entrant est annulée et un autre message est introduit selon les règles de priorité établies plus haut.

Une mesure de l'étranglement possible dans une cellule peut être donnée par le nombre de messages devant traverser la cellule (densité) en un pas de simulation. Cette mesure pour tous les noeuds du réseau peut être faite dès le prétraitement, car elle dépend du placement et de l'algorithme de routage.
Avec un algorithme de routage figé, cette mesure peut donner des critères de qualité sur le placement effectué.

Deux algorithmes simples ont été testés sur de nombreux exemples.

Le premier ne privilège aucune direction, et à | dX | et | dY | égaux, décrémente alternativement | dX | et | dY |

Le second privilège arbitrairement une direction .

Algorithme 1 :

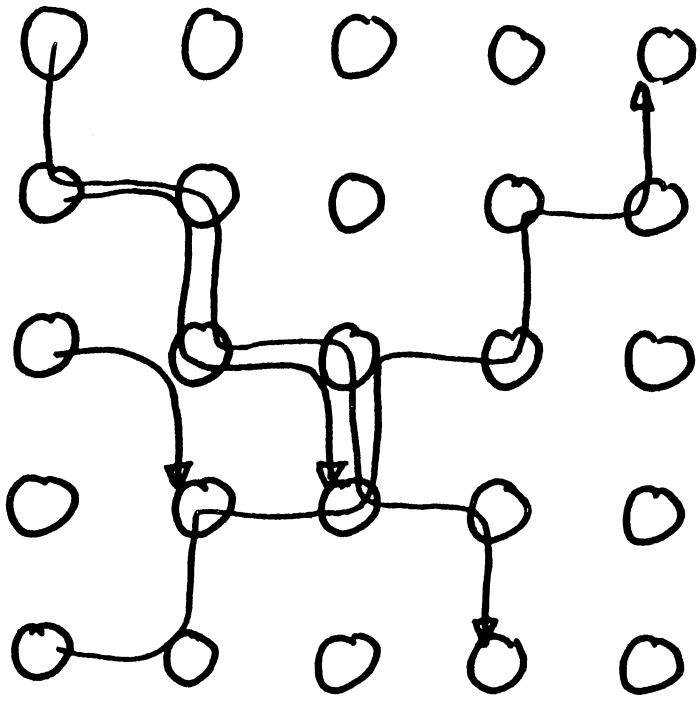
```
Si dX et dY ne sont pas tous deux nuls alors,
début
    Si | dX | ≥ | dY | alors
        début
            Si dX >o alors envoi vers l'Est
            si dX <o alors envoi vers l'Ouest
        fin
    sinon
        début
            Si dY >o alors envoi vers le Sud
            Si dY <o alors envoi vers le nord
        fin
fin
```

Algoritme 2 :

```
Si dX est différent de 0 alors
  début
    Si dX > 0 alors envoi vers l'Est
    Si dX < 0 alors envoi vers L'Ouest
  fin   Sinon    SI dY différent de 0 alors
          début
            Si dY > 0 alors envoi vers le Sud
            Si dY < 0 alors envoi vers le Nord
          fin
```

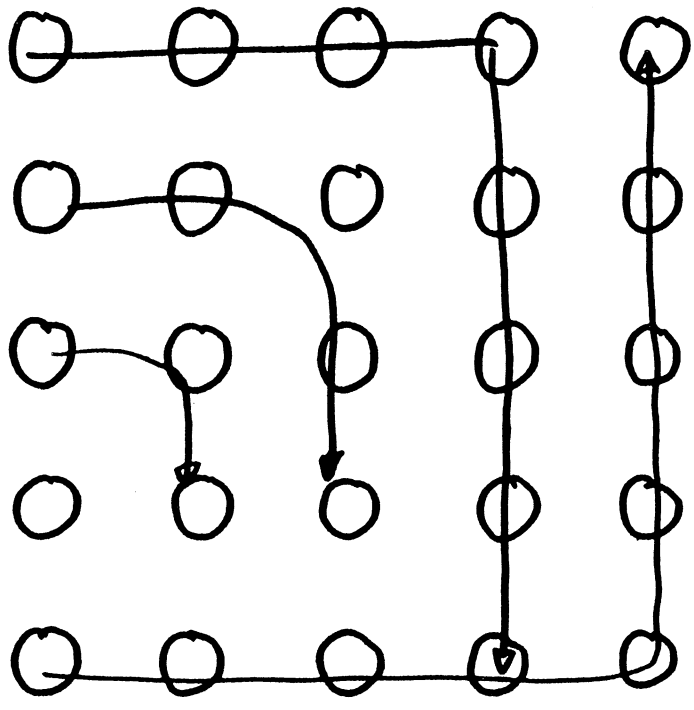
L'algorithme 1 a tendance à faire converger tous les messages vers le centre du réseau et donc à en dégarnir les parties périphériques. Sur plusieurs exemples simulés, des attentes ont été obtenues sur des cellules centrales alors que la circulation était possible par des cellules voisines. Des attentes existantes avec l'algo--rithme 1 ont disparu avec l'algorithme 2, qui a tendance à uniformiser la densité de la cir--culation sur tout le réseau.

Exemple : Algorithme 1



La densité des noeuds centraux atteint 3.

Algorithme 2



La densité des noeuds ne dépasse pas 1.

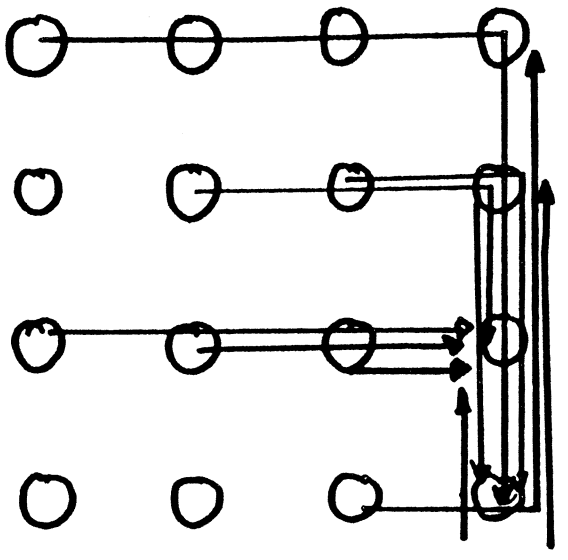
Au niveau de l'implantation, l'algorithme 2 est beaucoup plus simple, car il suffit de tester un seul bit (de signe) du message alors qu'avec l'algorithme 1, il est nécessaire de faire la comparaison entre | dX | et | dY |.

L'algoritnme 2 a donc été retenu pour l'implémentation.

Des simulations sur les problèmes d'acheminement ont été effectuées en langage PASCAL..
Une dizaine de circuits logiques une fois placés avec le programme décrit plus haut,ont fourni des acheminements types avec les deux algorithmes d'acheminement.
Des acheminements fictifs ont été également simulés de manière à voir le comportement du réseau avec une forte densité de messages sur certains noeuds..

Les résultats que nous avons pu tirer de ces simulations sont les suivantes :

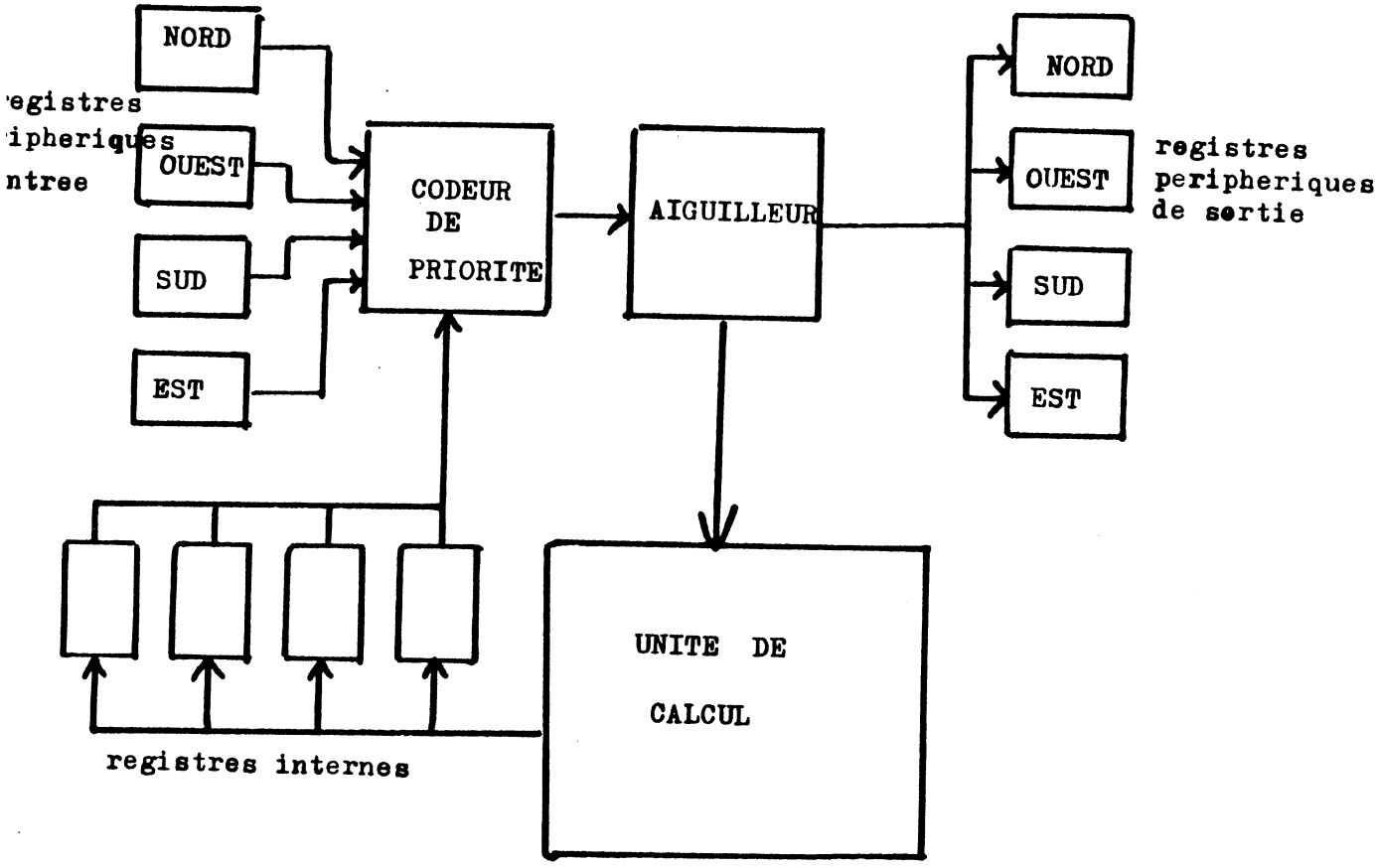
- Aucun blocage n'a pu être trouvé. Intuitivement le blocage paraît impossible, cependant il reste à en faire la démonstration.
- Des attentes ont été obtenues dans le cas ou une cellule est un point de passage important:



La densité du noeud [4,3] atteint 10.

Ce cas de figure est un cas extrême. Il correspond à un placement aberrant qui de toute façon ne peut pas se produire avec les restrictions imposées au prétraitement sur la description des circuits à simuler.
Il faut noter que même avec ce type de cas extrême, il y a plusieurs attentes, mais qu'il n'ya pas blocage, toutes les communications arrivent à destination.
Il sera cependant intéressant d'approfondir l'étude de la saturation du réseau en vue d'autres applications.

D'après toutes ces considérations, le schéma fonctionnel de cellule est :



III.2-4. Synchronisme ou Asynchronisme

Une question se pose au niveau des transmissions dans le réseau; faut-il les synchroniser sur une horloge globale, ou envisager un fonctionnement asynchrone ?

La synchronisation des transmissions a plusieurs inconvénients.
Il est difficilement envisageable au niveau de l'implantation du réseau sur une seule puce de faire parvenir à toutes les cellules une horloge externe non-dégradée sans occuper une surface importante sur la puce. La synchronisation ralentit les communications dans le réseau, puisqu'il faut attendre l'autorisation de l'horloge pour communiquer.
Par contre dans le cas asynchrone, les communications s'effectuent dès qu'un registre d'entrée est plein, et que le registre correspondant est vide.

La difficulté de la solution asynchrone tient à la réalisation de la transmission du message d'une cellule à sa voisine : la première doit écrire le message dans le registre de sortie adéquat (mais seulement si ce

registre est libre, c'est-à-dire si la valeur précédente a été lue) ; la seconde doit le lire (mais seulement après qu'il ait été écrit).
Pour assurer le respect de ces conditions, nous avons rajouté une bascule au registre, signifiant son état ' occupé ' ou ' libre '. Les deux cellules, asynchrones communiquent alors comme des processus parallèles.

L'implémentation et le fonctionnement détaillés de ce protocole seront décrits plus loin; on verra qu'ils peuvent être simples du point de vue de la complexité des circuits.

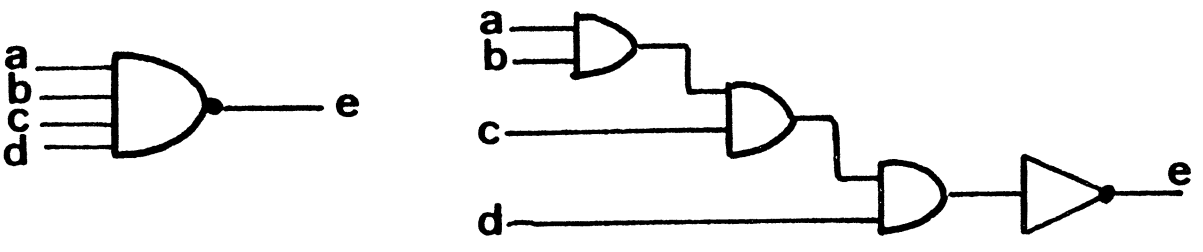
La solution asynchrone nous apparaissant la plus simple, a donc été retenue.

III.3. TRAITEMENT ET PARTIE CALCUL

La partie calcul peut recevoir de une à quatre valeurs d'entrée. Elle doit pouvoir effectuer le calcul logique de huit fonctions différentes, de portes ayant jusqu 'à quatre entrées.

Afin d'économiser sur la taille des dispositifs de calcul, une fonction logique à quatre entrées est décomposée en une suite d'opérations logiques à deux entrées maximum; de même pour une fonction à trois entrées.

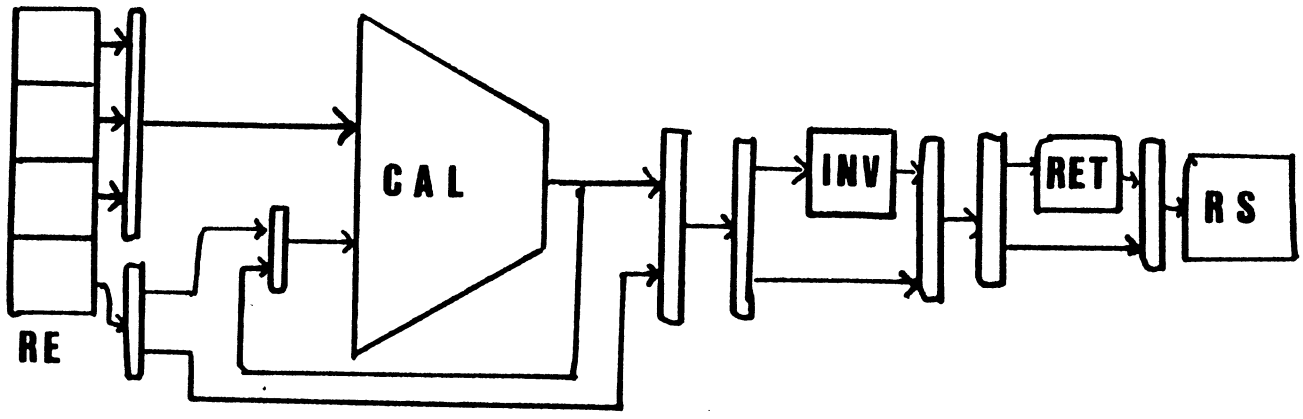
Exemple de décomposition :



La fonction ET est accomplie trois fois par le même dispositif, un séquenceur gérant la succession des opérations.

La partie calcul comprend donc une partie opérative, une partie contrôle programmable donnant le séquencement et une partie mémoire stockant les informations reçues pendant l'initialisation.

Le schéma fonctionnel de la partie opérative découle de la décomposition exposée ci-dessus :



Les valeurs d'entrée, quatre au maximum, arrivent dans le registre d'entrée RE à quatre étages.
Le séquenceur prévoit d'envoyer les deux premières valeurs sur le dispositif de calcul, puis d'envoyer la troisième avec le résultat des deux premiers ....

Si la cellule ne représente qu'une porte à une entrée, le séquenceur règle les différentes commandes pour court-circuiter le dispositif de calcul, et utiliser ou non la partie inversion.

Le dispositif de calcul contient une bibliothèque des trois fonctions logiques de base non-inverseuses ET, OU, OU exclusif.Ce dispositif peut être réalisé soit sous forme de mémoire associative, soit en logique combinatoire.
Cette dernière solution est bien plus économique au point de vue taille que la première.

Le dispositif INV réalise l'inversion . Il est du même type que le dispositif précédent.

Le retard est réalisé par un registre à décalage à trois étages, RET, synchronisé sur le signal externe GO.

Si le retard associé est nul, le dispositif RET est court-circuité.

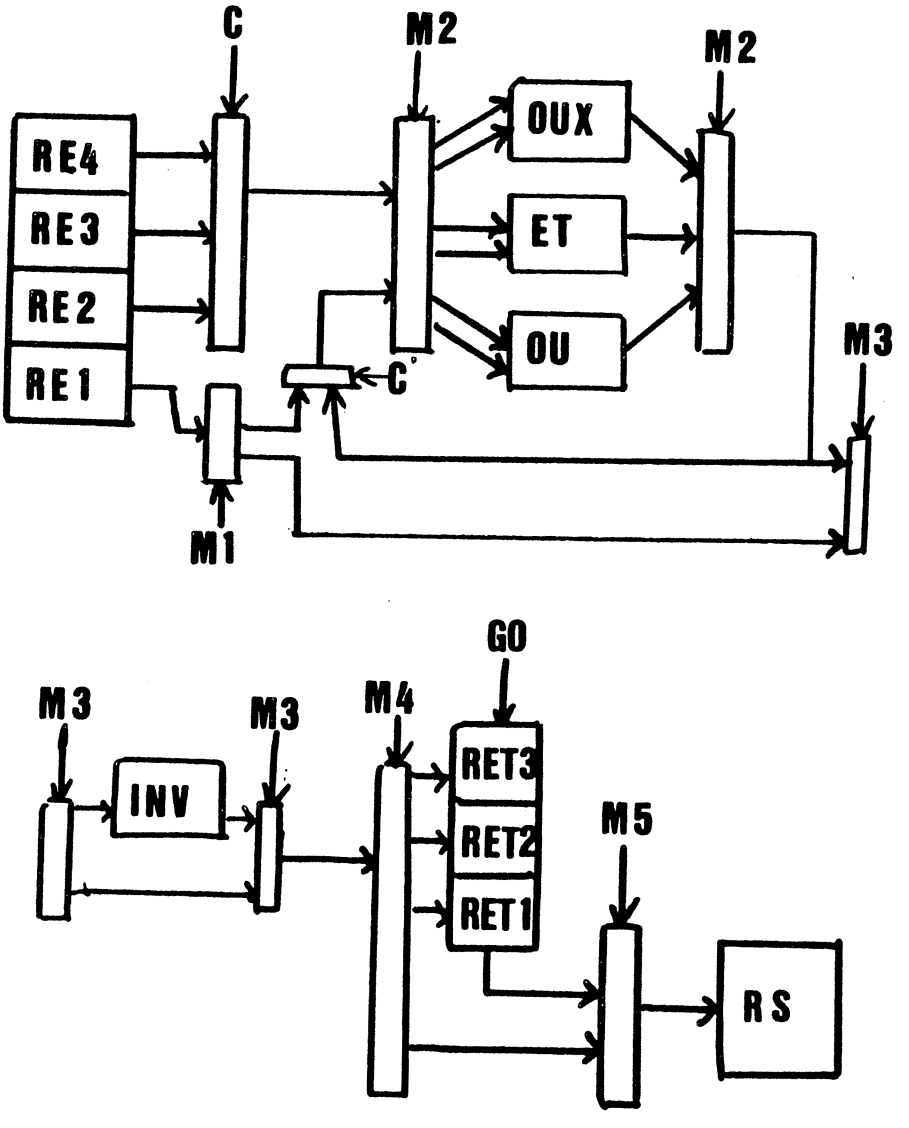
Si le retard associé est de une à trois unités de temps, la valeur logique est chargée à l'étage correspondant du registre RET.

Le résultat est chargé dans le registre RS, registre de sortie.

Les messages y sont assemblés avec les valeurs calculées et les adresses des destinataires.

Des multiplexeurs dont les commandes sont générées par la partie contrôle d'après le contenu des mémoires, servent à aiguiller le cheminement du calcul.

Le schéma fin de la partie opérative est indiqué ci-dessous. Les différents blocs et la génération des commandes sont détaillés dans la partie suivante.



T R O I S I E M E P A R T I E

##############################

DESCRIPTION DETAILLEE

****

DE

**

L'ARCHITECTURE

****

CHAPITRE I

CHOIX RETENUS

I. 1. SIMULATION CHOISIE

Le type de simulation retenue est celle d'une simulation trois états, Zéro, Un, indéterminé pour les raisons indiquées dans la deuxième partie. Ce choix exclut les éléments logiques du type interrupteur, ce qui fait que le réseau décrit ici ne peut simuler que des fonctions logiques traditionnelles. Cette restriction n'est pas gênante dans la mesure où il s'agit d'une première ébauche.

Il sera intéressant d'étudier le cas de la simulation avec haute-impédance ou de la simulation au niveau 'Switch' par la suite de manière à étendre les possibilités du réseau dans le domaine de la simulation.

I. 2. DIMENSIONS RETENUES

Les spécifications de la partie transmission de la cellule de base dépendent de la taille globale du réseau. De manière à bien comprendre les problèmes soulevés par une telle architecture, il est indispensable de spécifier les détails afin qu'une simulation, puis un prototype puisse être envisagés.

Nous avons choisi de fixer la taille du réseau à 8 x 8 cellules.

Ce choix est basé sur les deux points suivants :

- Dans l'optique d'une intégration sur une seule puce, nous tablons à priori sur une cellule d'une complexité inférieure à 2000 transistors. Un réseau 8 x 8 apparaît alors réalisable actuellement, alors qu'un réseau plus grand 16 x 16, l'est plus difficilement.
- Il existe un réseau de cellules 8 x 8 ayant la la même topologie [SU 82], 'The LSI adaptative array processor' La cellule est capable d'effectuer 16 opérations logiques et comprend 1200 transistors.

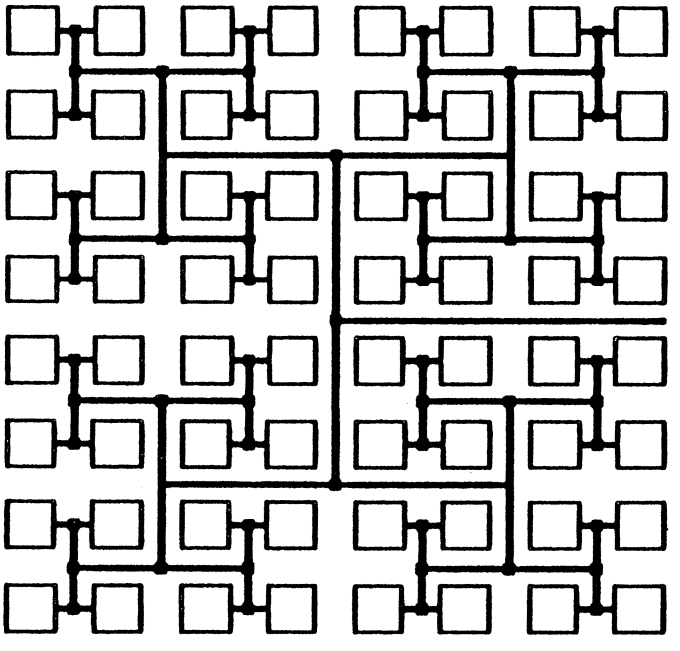
I. 3. PROBLEME DES SIGNAUX DE CONTROLE ET DE COMMANDE

Le réseau lui-même est utilisé pour véhiculer presque toutes les informations qui lui sont nécessaires, y compris pendant l'utilisation. Seuls les signaux GO, END, et de mode ne peuvent utiliser le réseau pour se propager. Il est donc nécessaire de prévoir un réseau physique

spécial pour irriguer toutes les cellules depuis l'extérieur.

Cette irrigation se pose en problème de surface pour l'intégration sur une puce. La solution généralement adoptée est une irrigation accomplie par une topologie en H décrite sur la figure ci-dessous :

Topologie en H :



Les noeuds peuvent être des points de régénération.

Pour le signal END, le ET des signaux arrivant à chaque noeud est effectué.

CHAPITRE II

DESCRIPTION DETAILLEE DE LA CELLULE

II. 1. MESSAGES ET INFORMATIONS

Il est nécessaire de définir le codage des différentes informations.
Avec un réseau 8 x 8, les dX et dY sont codés chacun sur 4 bits, 3 bits pour 8 valeurs plus 1 bit de signe.

Les valeurs logiques de simulation trois états nécessite deux bits de codage :

0 est codé 01

1 est codé 10

X est codé 11

Les messages sont de longueur fixe : 10 bits.

d X	d Y	info.

Les informations nécessaires à la programmation de la cellule sont codés de la manière suivante :

- Type logique : 8 fonctions possibles codées sur 3 bits, plus la non-utilisation de la partie calcul codée sur 1 bit; la cellule est alors utilisée juste en transit.

T4	T3	T2	T1	fonction
0	0	0	0	ramification
0	0	0	1	inverseur
0	0	1	0	E T
0	0	1	1	O U
0	1	0	0	N O N - E T
0	1	0	1	N O N - O U
0	1	1	0	N O N - O U X
0	1	1	1	O U X
1	ϕ	ϕ	ϕ	Rien (Transit)

- Retard : 4 retards possibles codés sur 2 bits.

R 2	R 1	fonction
0	0	Retard nul
0	1	Retard unitaire
1	0	Retard 2 unités
1	1	Retard 3 unités

- Entrance: 4 possibilités codées sur 2 bits.

E 2	E 1	Entrance
0	0	1
0	1	2
1	1	3
1	0	4

- Sortance : 4 possibilités codées sur 2 bits .

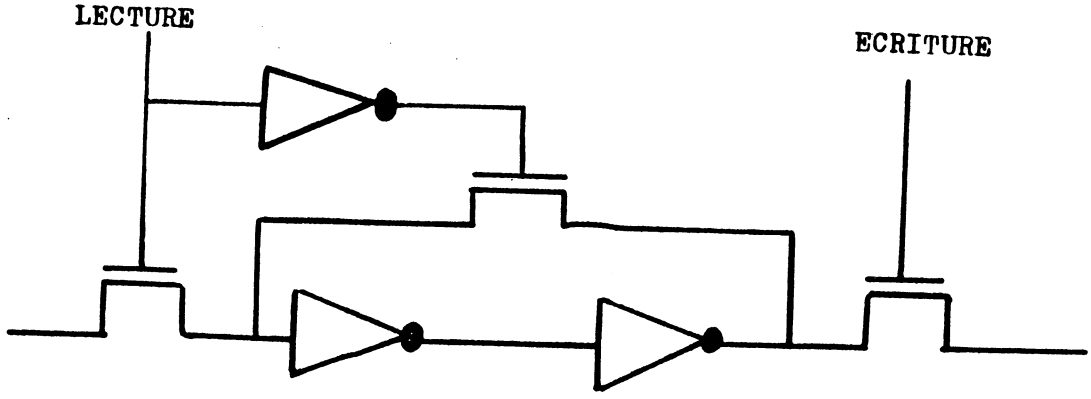
S 2	S 1	Sortance
0	0	1
0	1	2
1	0	3
1	1	4

II. 2. PARTIE TRANSMISSION

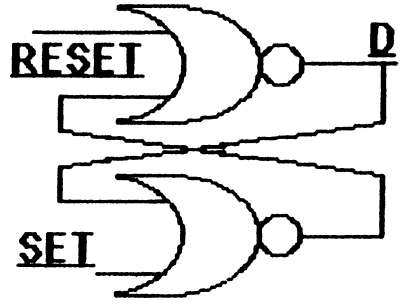
II. 2-1. Registres et Drapeaux

Les registres doivent être capables de stocker chacun un message de 10 bits.Les communications des 10 bits s'effectuent en parallèle de manière à augmenter la vitesse de communication. De plus, les dispositions sont plus simples que dans le cas de la Transmission série. La Transmission parallèle impose par contre, des messages de longueur fixe.

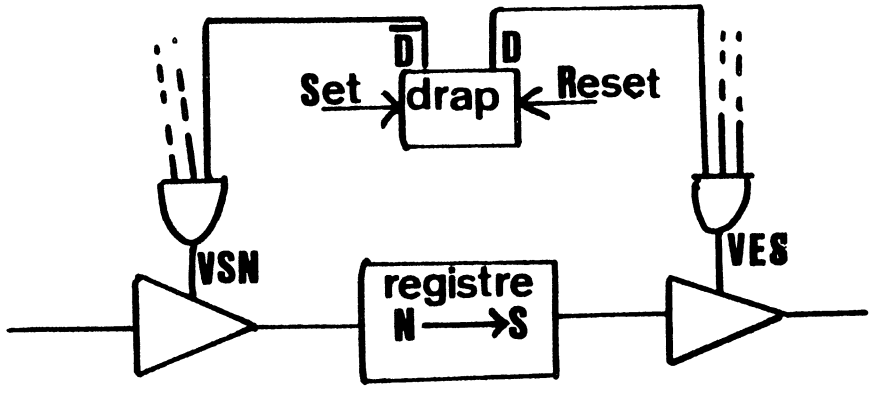
Un registre est donc constitué de 10 points mémoire en étage :



Un drapeau est associé à chaque registre indiquant s'il est occupé ou libre, si l'on doit lire ou écrire.
Ce drapeau est une bascule R S :



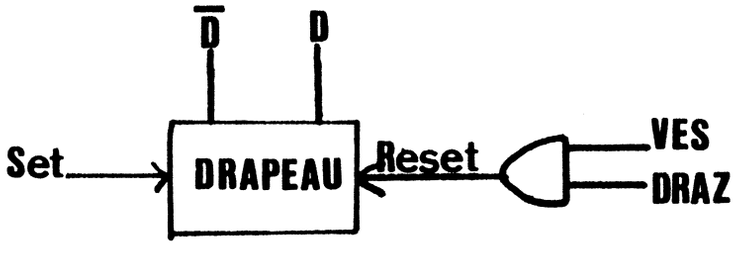
Le registre est relié au décodeur par une porte de transfert pour chaque étage à l'aiguilleur de la cellule par une porte de transfert :



VES est le signal de lecture Sud.

VSN est le signal d'écriture Nord.

Lorsque toutes les conditions sont réunies (Porte ET), le signal d'entrée VES passe à 1 et autorise la communication du registre vers la cellule. Une fois le message chargé dans le registre suivant, la cellule agit sur la commande Reset et remet le drapeau à 0 (libre), par l'intermédiaire d'une commande de remise à Zéro générale DRAZ. L'action sur le Reset se fait par la porte ET sur le signal de lecture et le signal DRAZ.



Pour charger un registre en sortie d'une cellule, le fonctionnement est identique. Le registre doit être libre, drapeau à 0, c'est donc D̄ qui agit sur la porte ET. Lorsque toutes les conditions sont réunies, le ET génère le signal VSN à 1 et autorise ainsi la communication.

II. 2-2. Incrémenteur-Décrémenteur

L'adresse du destinataire est constituée de 4 bits pour d X, et 4 bits pour d Y, 3 bits de déplacement et 1 bit de signe L'incrémentation s'effectue sur un déplacement négatif, la décrémentation sur un mot positif.
L'incrémentation d'un mot négatif consiste à retrancher 1 à sa valeur absolue, c'est-à-dire au nombre constitué de ses 3 premiers bits, le bit signe étant enlevé . Les deux opérations sont donc identiques et ne portent que sur les 3 premiers bits.

Les 4 dispositifs effectuant ces opérations sont donc identiques. Si les 4 bits de d X sont a3, a2, a1, a0 et que a3 est le bit de signe, la décrémentation est indiquée dans le tableau suivant :

a2	a1	a0	s2	s1	s0
0	0	0	ϕ	ϕ	ϕ
0	0	1	0	0	0
0	1	0	0	0	1
0	1	1	0	1	0
1	0	0	0	1	1
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	1	0

a3,s2,s1,s0 sont les 4 bits du dX décrémenté.

Les équations logiques sont: $s0 = \overline{a0}.(a1 + a2)$

$s1 = a1.a0 + a2.\overline{a1}.\overline{a0}$

$s2 = a2.(a0 + a1)$

II. 2-3. Codeur de priorité

Nous avons vu plus haut que la priorité est au premier message arrivé ou alors dans l'ordre :

Registre calcul Nord Ouest Sud Est

Or le registre de sortie de la partie calcul contient quatre étages, RS1, RS2, RS3, RS4, correspondant aux quatre possibles destinataires. La règle de priorité devient :

RS1 > RS2 > RS3 > RS4 > Nord > Ouest > Sud > ESt

Lorsqu'un message est en cours de décodage, les autres registres de priorités supérieures doivent être inhibés. Si après un décodage, il s'avère que le transfert est impossible à cause de l'occupation du registre de sortie, l'inhibition sur les autres registres doit être levée afin que les règles de priorité puissent jouer, et le drapeau initial doit être maintenu à 'occupé' pour demander la lecture au tour suivant.

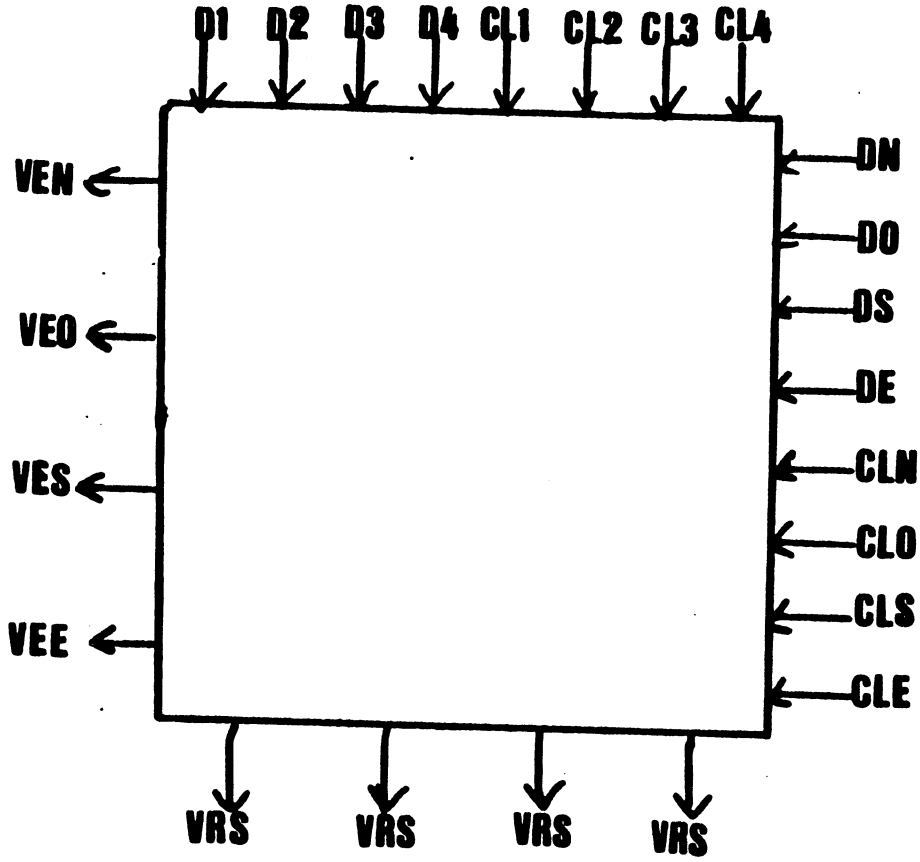
Les signaux D1, D2, D3, D4, DN, DO, DS, DE sont les sorties de drapeaux associés aux registres.

Les signaux CL1, CL2, CL3, CL4, CLN, CLO, CLS, CLE sont les signaux d'annulation de l'inhibition.

Les signaux VEN, VEO, VES, VEE sont les signaux de lecture Nord-Ouest, Sud, Est, validant le transfert des registres d'entrée dans l'aiguilleur.

Les signaux VRS1, VRS2, VRS3, VRS4 sont les signaux de lecture des registres de sortie de la partie calcul.

Le codeur de priorité évalue les signaux de lecture des huit registres en fonction des signaux drapeaux, d'annulation et d'inhibition.



Les équations du codeur sont les suivantes :

$VRS1 = D1.CL1.\overline{VRS2}.\overline{VRS3}.\overline{VRS4}.\overline{VEN}.\overline{VEO}.\overline{VES}.\overline{VEE}$

$A = \overline{D1}+VRS2$

$VRS2 = A.D2.CL2.\overline{VRS3}.\overline{VRS4}.\overline{VEN}.\overline{VEO}.\overline{VES}.\overline{VEE}$

$B = \overline{D2}.A + VRS3$

$VRS3 = B.D3.CL3\ \overline{VRS4}.\overline{VEN}\ \overline{VEO}.\overline{VES}.\overline{VEE}$

$C = \overline{D3}.B + VRS4$

$VRS4 = C.D4.CL4.\overline{VEN}.\overline{VEO}.\overline{VES}.\overline{VEE}$

$D = \overline{D4}.C + VEN$

$VEN = D.DN.CLN.\overline{VEO}.\overline{VES}.\overline{VEE}$

$E = \overline{DN}.D + VEO$

$VEO = E.DO.CLO.\overline{VES}.\overline{VEE}$

$F = \overline{DO}.E + VES$

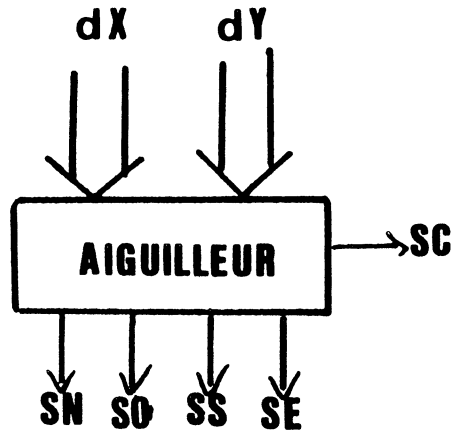
$VES = F.DS.CLS.\overline{VEE}$

$VEE = (F.\overline{DS}.+\ \overline{DE}).DE.CLE$

II. 2-4 Aiguilleur

L'aiguilleur possède l'algorithme d'acheminement.

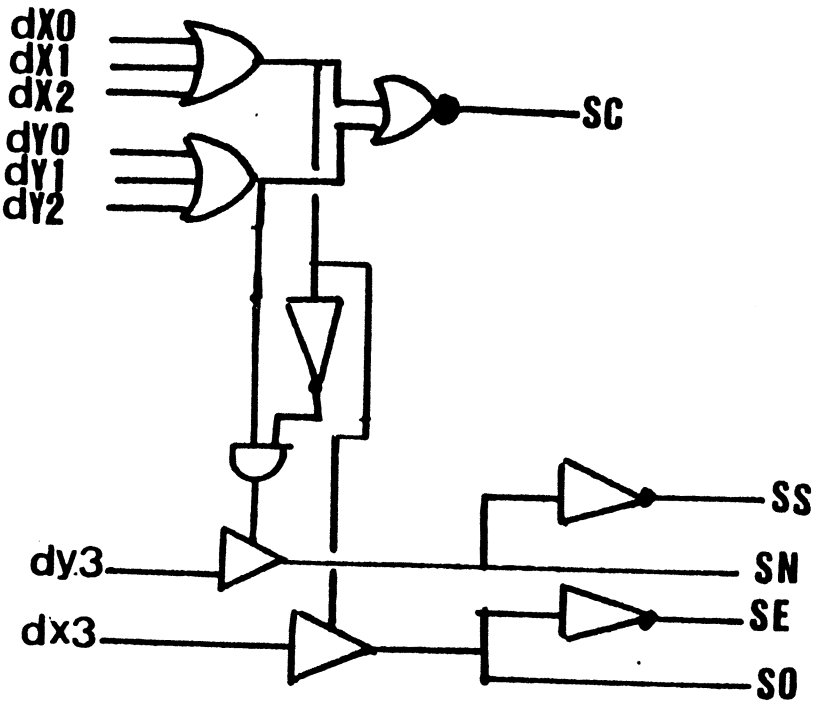
Son schéma fonctionnel est ;



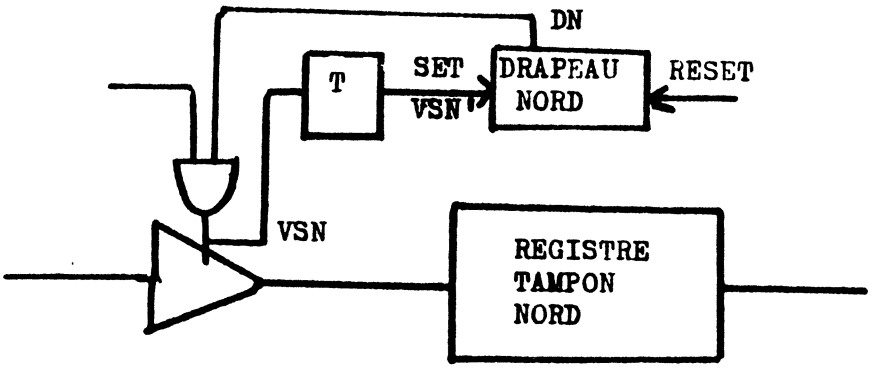
SC, SN, SO, SS, SE sont les signaux de demande d'accès respectifs à la partie calcul, à la sortie Nord, à la sortie Ouest, à la sortie Sud, à la sortie Est.

L'implémentation de l'algorithme est très simple et immédiate.
Le mot dX est constitué de dX3, bit de signe et des bits dX2, dX1, dX0 (idem pour dY).
Si dX est positif alors dX3 est nul, si dX est négatif alors dX3 est égal à un.

Schéma de l'aiguilleur :



Le chargement des registres de sortie s'effectue par le dispositif suivant :

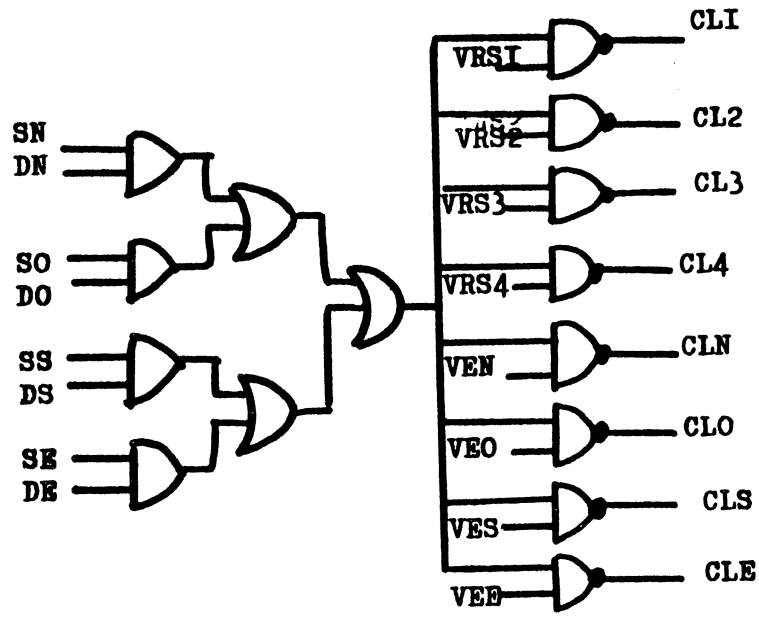


Si l'aiguilleur envoie un message vers le Nord, le signal SN vaut 1. Si le registre est libre, le signal DN vaut 0. Le signal VSN passe donc à 1 et autorise la communication. Après un temps T suffisamment long pour que le chargement soit bien terminé, le drapeau est positionné à 1 ('occupé') et le signal DN coupe la communication.

II. 2-5 Génération des signaux CL de remise à zéro

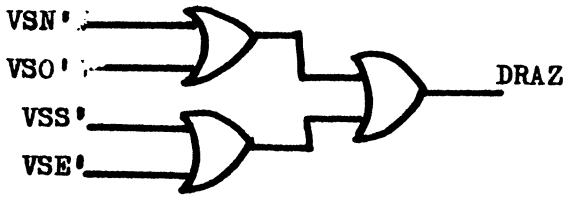
L'annulation de la demande de lecture d'un registre d'entrée est nécessaire lorsque le registre de sortie adéquat est occupé. Les signaux CL sont générés à partir des signaux de demande d'accès venant de l'aiguilleur et des signaux ' libre / occupé ' des drapeaux de sortie.

Schéma du dispositif :



II. 2-6. Génération du signal de remise à zéro des drapeaux

Le signal DRAZ est obtenu en détectant l'écriture d'un message dans un registre de sortie ou dans le registre d'entrée de la partie calcul.



II. 3. PARTIE CALCUL

II. 3-1. Unités de calcul

II.3-1-1. Logique combinatoire

Le bloc Calcul comporte trois unités effectuant les fonctions logiques ET, OU, OUX en logique 3 états 0,1,X.

- Fonction ET, a et b sont les deux entrées, s est la sortie.

La table de vérité de cette fonction est :

a \ b	0	X	1
0	0	0	0
X	0	X	X
1	0	X	1

Avec le codage utilisé, la table de Karnaugh est :

a1a0 \ b1b0	0 0	0 1	1 1	1 0
0 0	ϕ ϕ	ϕ ϕ	ϕ ϕ	ϕ ϕ
0 1	ϕ ϕ	0 1	0 1	1 1
1 1	ϕ ϕ	0 1	1 1	1 1
1 0	ϕ ϕ	0 1	1 1	1 0

D'où : s0 = b0 + a0

s1 = a1.b1

-Fonction OU, a et b sont les entrées, s est la sortie.

table de vérité :

a \ b	0	X	1
0	0	X	1
X	X	X	X
1	1	1	1

table de karnaugh :

	0 0	0 1	1 1	1 0
0 0	ϕ ϕ	ϕ ϕ	ϕ ϕ	ϕ ϕ
0 1	ϕ ϕ	0 1	1 1	1 0
1 1	ϕ ϕ	1 1	1 1	1 0
1 0	ϕ ϕ	1 0	1 0	1 0

D'ou s0 = a0.b0

s1 = a1 + b1

-Fonction inversion, a est l'entrée, s est la sortie.

table de vérité :

a	0	X	1
s	1	X	0

table de karnaugh :

a	0 0	0 1	1 1	1 0
s	ϕ ϕ	1 0	1 1	0 1

D'ou s0 = a1

s1 = a0

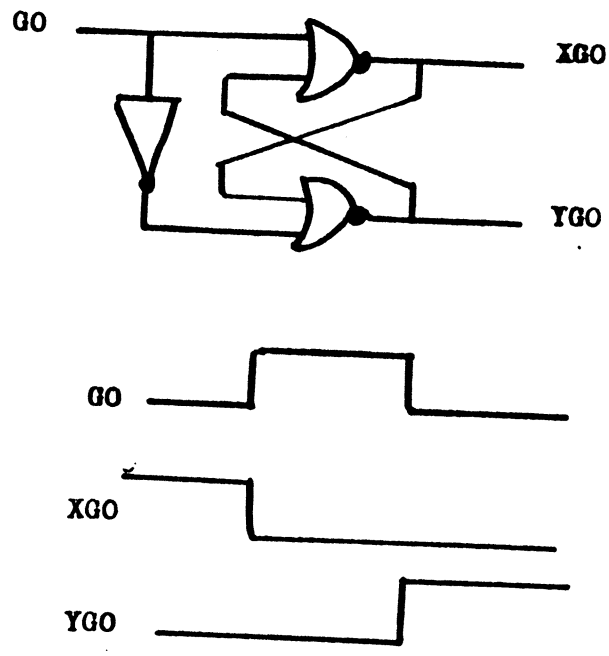
II.3-1-2. Mémoire associative

L'utilisation de mémoires associatives pour effectuer les fonctions logiques de calcul est possible à condition de stocker les tables de vérité de ces fonctions.
Une fonction à deux entrées nécessite une mémoire de neuf mots de deux bits soit environ une centaine de transistors plus le mécanisme de sélection (en NMOS). Cette solution donne une complexité de dispositif complètement disproportionnée avec la précédente et a donc été abandonnée.

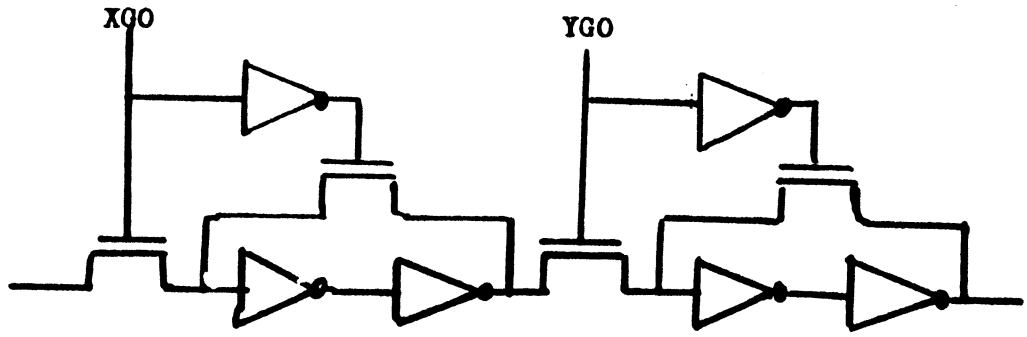
II. 3-2. Dispositif retard

Le retard est réalisé par un registre à décalage à trois étages. Les valeurs véhiculées dans la partie calcul étant des mots de deux bits, chaque étage doit pouvoir stocker deux bits.

Chaque étage doit avoir un maître et un esclave de manière à éviter le décalage complet sur un seul signal de décalage GO. Le signal GO génère donc deux signaux inversés et non-recouvrants de manière à effectuer le décalage de façon correcte.



Dispositif 1 étage 1 bit :



II. 3-3. Registres

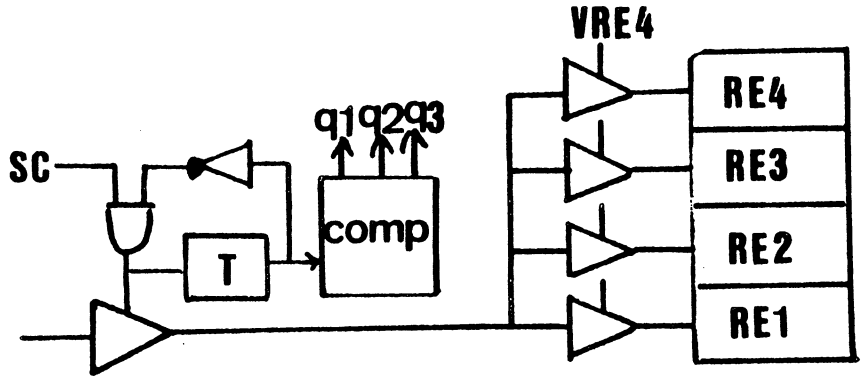
II.3-3-1. Registre d'entrée R E

RE est un registre deux bits à quatre étages.

A chaque arrivée de valeurs, un compteur s'incrémente.

Selon la valeur indiquée par le compteur l'accès à l'étage adéquat est validé.

Schéma du dispositif :



Si aucune valeur n'est arrivée alors Q = 0 (Q3Q2Q1= 000) et l'étage 1 est accessible (VRE1 = 1).
Si la valeur (i-1) est arrivée ,alors Q = i-1, l'étage i est accessible (VREi = 1). Lorsque la valeur i arrive, elle se range au ième étage . Q passe à i + 1 au bout du temps T, ferme l'accès à l'étage i (VREi = 0), et ouvre l'accès à l'étage i + 1. (VREi + 1= 1).

D'où les commandes d'accès au registre d'entrée :

$$VRE1 = \overline{Q3}.\ \overline{Q2}.\ \overline{Q1}.$$

$$VRE2 = \overline{Q3}.\ \overline{Q2}.\ Q1.$$

$$VRE3 = \overline{Q3}.\ Q2.\ Q1.$$

$$VRE4 = Q3.\ Q2.\ Q1.$$

II.3-3-2. Registre de sortie RS

RS contient quatre étages de 10 bits chacun puisque quatre messages au plus peuvent être envoyés.
Chaque étage contient l'adresse des quatre destinataires possibles de la cellule. La valeur calculée est chargée dans les quatre étages de RS en queue de message. Les quatre drapeaux associés sont positionnés à 1 ou à 0 selon le nombre de messages à envoyer.
Les signaux Set des drapeaux sont donc obtenus à partir du signal GO et des informations TYPE et SORTANCE.

$$Set\ 1 = \overline{T4}\ .\ GO$$

$$Set\ 2 = \overline{T4}\ .\ (S1 + S2)\ .\ GO$$

$$Set\ 3 = \overline{T4}\ .\ S2\ .\ GO$$

$$Set\ 4 = \overline{T4}\ .\ S1\ .\ S2\ .\ GO$$

II. 3-4. Séquencement du calcul

De manière à pouvoir effectuer le calcul correctement, des commandes sélectionnent les différents dispositifs. Ces commandes peuvent être fixes pour toute la simulation ou mobile, elles dépendent alors du nombre de valeurs d'entrée arrivées.

Les commandes fixes sont : M1, M21, M22, M23, M3, M41, M42, M43, M5 . Elles dépendent des informations Type et Retard.

M1, M2, M3 dépendent de Type

M1 permet l'utilisation de la cellule en inverseur et ramification.

M2 permet l'utilisation de la cellule en autres fonctions logiques.

M3 permet l'inversion.

D'où $M1 = \overline{T3} \cdot \overline{T2}$

$M21 = T3 \cdot T2$

$M22 = T1 \cdot (T2.\overline{T3}+\overline{T2}.T3)$

$M23 = \overline{T1} \cdot (T2.\overline{T3}+\overline{T2}.T3)$

M4 et M5 dépendent de Retard.

$$M41 = R1 \cdot \overline{R2}$$

$$M42 = \overline{R1} \cdot R2$$

$$M43 = R1 \cdot R2$$

$$M\ 5 = R1 + R2$$

Les commandes mobiles sont C1, C2, C3, C4.
Elles servent à effectuer la succession des opérations logiques. Elles sont générées par le compteur d'entrée.

Lorsque les deux premières valeurs d'entrée sont arrivées, la première opération peut être effectuée.
Lorsque la troisième opération et (quatrième) valeur d'entrée est arrivée, la deuxième (dernière) opération peut être effectuée.
Les équations de commandes sont donc :

$$C1 = VRE3$$

$$C2 = VRE4$$

$$C3 = Q3 \cdot Q2 \cdot \overline{Q1}$$

$$C4 = C3 \cdot C2$$

II. 3-5. Génération du signal END

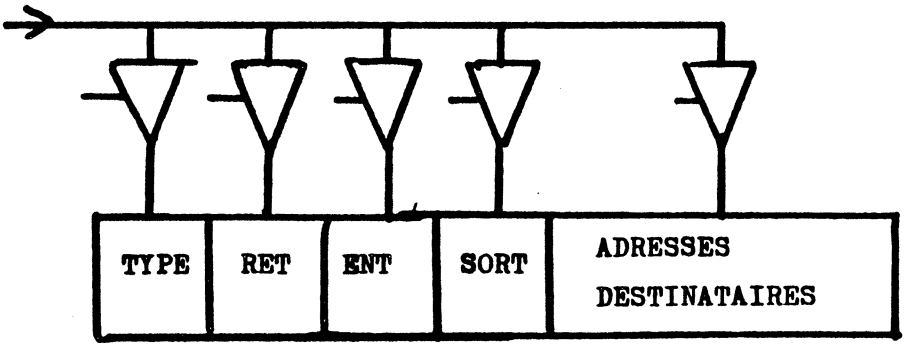
Lorsque les quatre drapeaux des quatre étages du registre RS sont à zéro, le signal END propre à la cellule peut être généré.

$$END = \overline{D1 \cdot D2 \cdot D3 \cdot D4}$$

II. 3-6. Mémoire de la cellule

Les informations arrivant pendant l'initialisation sont rangées en mémoire au fur et à mesure de leur arrivée. L'ordre d'arrivée doit être établi. Un compteur s'incrémente à chaque arrivée et valide successivement les différentes parties de la mémoire en générant les commandes VM.

Schéma du dispositif :



21 messages de deux bits doivent rentrer en mémoire.

CHAPITRE III

EVALUATIONS DES TAILLE ET PERFORMANCE DE LA CELLULE

III. 1. PERFORMANCES

Le temps de traversée d'une cellule conditionne le temps global de simulation pouvant être obtenu à l'aide d'un tel réseau. Afin d'obtenir une évaluation aussi précise que possible des temps de traversée de la cellule, nous avons étudié deux exemples particuliers, une Traversée Nord-Sud, et une Traversée Ouest-Sud avortée. Nous avons spécifié le dispositif de routage au complet d'une cellule NMOS pour la Traversée Nord-Sud (voir schéma en annexe).

III. 1-1. Exemple de Traversée Nord-Sud

Le schéma détaillé de la partie routage donné en annexe nous permet de déduire les chronogrammes de la traversée. La partie critique du dispositif de routage est le retard T qui doit être supérieur au temps d'écriture dans le registre de sortie. Ce retard peut être réalisé par une succession de portes à grand temps de propagation.

Des simulations électriques ont permis de constater un temps d'écriture de 2.Tel, Tel étant le temps de propagation d'une porte élémentaire (2 entrées).

Afin d'établir une certaine marge de sécurité, nous avons choisi T = 5.Tel.
Les chronogrammes de la Traversée sont indiqués en annexe.
Le signal Reset Nord va agir pendant le temps T. Il est nécessaire d'éviter l'action du signal Set Nord (demande d'écriture de la part de la cellule adjacente) tant que le Reset Nord est à 1. Ceci est effectué par le signal $\overline{\text{Reset}}$ commandant un interrupteur sur le signal Set. Avec ce dispositif, une demande d'écriture en entrée Nord restera en attente ou passera la main tant que le signal Reset ne redescendra pas à 0.

Le chronogramme nous donne un temps de traversée de 31.Tel.

Dans le cas d'une technologie NMOS à 3 microns, le temps de traversée élémentaire est de 2ns.

D'où T traversée = 62 ns.

III. 1-2. Exemple de Traversée avortée Ouest-Sud

Les chronogrammes correspondants sont indiqués en annexe.
L'entrée Ouest se remplit et demande l'accès à l'aiguilleur au temps t1. Le message s'oriente par décodage vers le Sud.

Mais la sortie Sud est occupée (DS à 1). CLO remet à zéro au temps t2 la commande d'entrée Ouest et pendant T' (jusqu'à t3), l'accès à l'aiguilleur est possible pour les directions à priorité supérieure à celle de Ouest.Au temps t'1 (inférieur à t2), Nord reçoit un message qui est donc en attente. Lorsque l'entrée Ouest est remise à zéro, au temps t2, Nord peut alors demander l'accès à l'aiguilleur et la commande d'entrée Nord passe à Un au temps t'2, puis bloque l'entrée Ouest. Ouest a perdu l'accès vers l'aiguilleur au profit de Nord.

Le point délicat du dispositif est l'ajustage du retard T' de manière suffisamment grande pour qu'une demande d'accès de priorité supérieure ait le temps de s'imposer.

Le temps d'annulation correspond d'après les chronogrammes à $t'2 - t1 = 16.Tel$.

Soit t annulation = 32 ns.

III. 1-3. Temps nécessaire au calcul

Le calcul se fait en trois étapes au maximum (dans le cas de deux valeurs d'entrée). Lorsque la dernière valeur attendue arrive dans le registre d'entrée de la

partie calcul, le compteur s'incrémente et autorise ensuite le calcul.
D'après le schéma détaillé de la partie calcul (voir en annexe), une étape de calcul pour deux valeurs (ouverture de l'étage adéquat du registre d'entrée jusqu'au chargement dans RET ou RS) s'effectue en 20 Tel.

Soit t cal = 40 ns.

Au niveau du temps de calcul global de la simulation, interviennent le temps de transmission jusqu'à la cellule destinataire et le temps de la dernière étape de calcul. Les étapes de calcul précédentes n'influent pas sur le temps global puisqu'elles se produisent pendant l'acheminement de la dernière valeur.

III. 2. TAILLE DE LA CELLULE

Les schémas des différents dispositifs ont été détaillés en vue d'une implantation en technologie NMOS. Ils permettent de chiffrer en nombre de transistors la complexité de la cellule et d'en évaluer une taille possible dans cette technologie.

III. 2-1. Partie transmission

-registre 10 bits : 5 transistors par bit, soit 50 transistors. 8registres sont nécessaires à une cellule, mais chacun est commun à deux cellules. On ne comptabilise donc que 4 registres par cellules. Soit 120 transistors.

-décrémenteur : 23 transistors. 8 décrémenteurs sont nécessaires à une cellule, et chacun est commun à deux cellules. On ne comptabilise que 4 décrémenteurs. Soit 92 transistors.

-codeur de priorité : 40 portes à 2 entrées, 8 portes à 3 entrées, 7 inverseurs. Soit 159 transistors.

-aiguilleur : 1 seul par cellule, soit 26 transistors.

-drapeau : 9 transistors, 1 drapeau par registre, 4 drapeaux par cellule. Soit 36 transistors.

-dispositif DRAZ : 7 portes à 2entrées. Soit 21 transistors.

-dispositif CL : 15 portes à 2 entrées, 1 retard. Soit 51 transistors.

-divers dispositifs de commandes : environ 50 transistors.

La partie transmission nécessite environ 500 transistors.

III. 2-2. Partie calcul

-registre d'entrée et commande : 26 inverseurs, 1 porte à 2 entrées, 4 portes à 3 entrées,24 interrupteurs. Soit95 transistors.
-compteur 3 bits : 12 portes à 2 entrées, 1 inverseur. Soit 38 transistors.
-dispositif de calcul : 22 inverseurs, 28 interrupteurs, 10 portes à 2 entrées. Soit 102 transistors.
-dispositif retard : 30 interrupteurs, 30 inverseurs, 4 portes à 2 entrées. Soit 102 transistors.
-registre de sortie : 16 interrupteurs, 30 inverseurs. Soit 76 transistors.
-mémoire de 42 bits et commandes : 260 transistors.

La partie calcul nécessite environ 673 transistors.

III. 2-3. Complexité et taille d'une cellule complète

L'ensemble partie transmission et partie calcul (avec mémoire) nécessite environ 1230 transistors NMOS. Dans une technologie NMOS 3 micron actuelle, et d'après les travaux effectués dans des domaines similaires, [SUD 82], une densité de 2100 transistors par mm2 semble réaliste. La taille d'une cellule peut donc être estimée à 0,6mm2 dans une telle technologie.

Q U A T R I E M E P A R T I E

##

PERFORMANCES GLOBALES

****

DE

**

SIMULATION

****

CHAPITRE I

PERFORMANCES D'UN RESEAU 8 X 8

I. 1. ESTIMATION DE LA VITESSE DE SIMULATION

Un réseau 8x8 peut simuler 64 portes logiques en un pas de simulation. L'évaluation de la durée de ce pas permet de déduire la vitesse de simulation.
La durée du pas de simulation est entièrement dépendante de l'acheminement dans le réseau. S'il n'y a pas d'attente dans celui-ci, la durée du pas de simulation dépend du plus long chemin à parcourir par les messages. Ce plus long chemin est déterminé durant la phase de placement, qui a donc une influence sur la vitesse de simulation. En cas d'encombrement, des attentes peuvent allonger la durée du pas si elles se produisent sur le plus long chemin, ou si elles se produisent sur un autre chemin et lui font dépasser le temps d'acheminement du chemin le plus long.
Nous avons choisi de borner supérieurement la durée du pas de simulation par le temps de trajet du chemin le plus long et de considérer des encombrements sur celui-ci. Afin d'obtenir des informations plus précises sur les phénomènes d'encombrement et d'attente, nous avons effectué des simulations sur les problèmes d'acheminement dans des réseaux N x M. Des circuits logiques réels (additionneur, compteur...), ont été positionnés sur une

grille par le placeur décrit dans la deuxième partie. De ce placement et de l'algorithme de routage choisi, il en résulte un acheminement particulier pour chaque circuit que nous avons simulé. Le dépouillement des simulations a permis de constater des rares attentes sur des cellules ètant des points de passage pour plusieurs messages. La durée du pas de simulation a toujours été proportionnelle au chemin le plus long obtenu par placement.

En fait ces résultats sont assez compréhensibles dans la mesure où les circuits logiques simulés ont une connectivité moyenne assez faible(inférieure à 3). Pour obtenir des simulations avec attentes importantes conduisant à des retards sur le trajet 1 plus long, il nous a fallu simuler des circuits fictifs à forte connectivité(jusqu'à 6) et effectuer des placements aberrants (deux portes connectées entre elles placées aux antipodes du réseau par exemple)qui ont une faible probabilité de se produire.

Pour un réseau N X N, le plus long chemin est de longueur 2N-2. Pour N égal à 8, 14 cellules sont à traverser par le message sur ce trajet le plus long.

Nous avons choisi un trajet avec 7 traversées de cellules sans problèmes et 7 traversées avec une attente de 1. Une fois le dernier message arrivé, la cellule le recevant effectue le calcul de deux valeurs, celle en attente dans la partie calcul et celle se trouvant dans le dernier message arrivant. D'où la durée maximale du pas de simulation :

$$Tmax = 7.Ttrav + 7.(Ttrav + Tanul) + Tcal$$

$$Tmax = 1200ns$$

(Ttrav, Tanul, Tcal ètant les temps élémentaires de la cellule définis en III.I)

La durée du pas de simulation serait donc de l'ordre de la microseconde.

64 portes sont simulées en 1,2 microseconde au maximum. La vitesse de simulation théorique peut doc être supérieure à 50 millions de portes par seconde. En considérant une activité moyenne de 20% dans un circuit logique, la vitesse de simulation serait donc supérieure à 10 million d'évenements par seconde.

Si le nombre obtenu paraît impressionnant, des réserves sont à apporter, dûes à la taille du réseau(64 portes simulables seulement) et aux problèmes d'entrée/sortie. Cependant le nombre obtenu, même dégradé, montre l'intérêt de cette architecture particulière par rapport aux machines spécialisées existantes.

I. 2. ESTIMATION DE LA COMPLEXITE

La taille estimée pour une cellule élémentaire en technologie NMOS en vue d'un réseau 8x8 est de 0,6 mm2 pour 1230 transistors. Ce calcul est basé sur une forte densité de 2100 transistors par mm2, [SUD 82]. Pour l'implantation du réseau tout entier, la densité sera forcément dégradée par l'accolement des cellules, le passage d'alimentations et signaux de contrôle, et les dispositifs d'entrée/sortie. Le 'LSI adaptative array' a une densité de 2100 trans/mm2 pour la cellule et de 860 trans/mm2 pour le circuit tout entier. Nous pouvons espérer une densité plus élevée pour notre architecture dans la mesure où notre topologie est plus simple (lien avec les 4 plus proches voisins au lieu des 8) et les problèmes d'accolement assez réduit. De plus, l'asynchronisme du réseau diminue le nombre de signaux de contrôle à véhiculer à l'intérieur du réseau. Mais le problème le plus important est celui des communications à l'extérieur.

De manière à garantir une certaine vitesse de communication, le nombre de plots d'entrée/sortie par cellule périphérique doit être assez élevé. Or le nombre de plots

total d'un circuit ne dépasse généralement pas 180 (141 pour le 'LSI adaptative array, 3plots par cellule périphérique). Dans notre cas, 4 plots par cellule périphérique donnent 128 plots pour les données, 5 plots donnent 160 plots , plus ceux nécessaires aux alimentations, signaux de commande et contrôle. De plus, le nombre de plots joue un rôle non négligeable dans la taille du circuit. 5 plots par cellule périphérique permettraient de sortir 5 bits à la fois avec un mécanisme de conversion parallèle-série.

Il est difficile de chiffrer la taille du réseau sans une étude plus approfondie de l'implantation dans une technologie bien précise. La complexité peut être évaluée directement avec le nombre de transistors en NMOS, soit 80000 environ. A titre indicatif, avec une densité de 1000trans/mm2, légerement supérieure à [SUL 82] dans une technologie NMOS 3micron, la taille serait d'environ 80mm2.

CHAPITRE II

ARCHITECTURES UTILISANT LE RESEAU CELLULAIRE

II. 1. PERFORMANCES D'UN CIRCUIT N X N

Le calcul pour N supérieure à 8 est le même que précedemment. La taille de la partie transmission d'une cellule doit être étendue : 2 bits de plus sont nécessaires pour un résaeu 15 X 16, 4 bits pour un résaeu 32 X 32. Les registres de le partie transmission doivent être agrandis pour recevoir les messages. L'aiguilleur et le décrémenteur doivent également être modifiés.
Les temps de traversée et decalcul d'une cellule élémentaire ne seront que peu modifiés, les communications s'effectuant en parallèle; seule la taille de la cellule de base sera augmentée.
Le tableau ci-dessous donne des estimations de la compexité et des performances pour différents N en prenant en compte les augmentations de taille de la cellule de base.

N X N	taille en transistors	nombre de portes simulées	vitesse
8 x 8	80.000	84	10 Mev/sec
16 x16	350.000	256	20 Mev/sec
32 x 32	1.600.000	1.024	40 Mev/sec

Un circuit 16 x 16 semble le maximum intégrable actuellement. Pour obtenir une taille encore plus grande, il convient d'utiliser un certain nombre de circuit de base 8 x 8 ou 16 x 16.

II. 2. ARCHITECTURES POSSIBLES

L'intérêt d'une machine spécialisée de simulation est de pouvoir traiter de très gros circuits. Par exemple YSE simule des circuits de l'ordre du million de portes. Un circuit 8 x 8 ou 16 x 16 peut donc servir de base à une machine spécialisée; il ressemble notamment au processeur logique de la machine YSE. La solution consiste 0 utiliser un grand nombre de circuits de base pour arriver à traiter un grand nombre de portes. Par exemple 65.000 portes sont simulables avec une machine composée de 32 x 32 circuits 8 x 8 ou 16 x 16 cicuits 16 x 16. Un simulateur materiel peut être réalisé avec une machine hôte et un grand nombre de circuits de base, chacun correspondant au processeur logique de la machine de Yorktown ou au processeur 'block' de la machine HAL (I.2). La machine hôte aura à effectuer un partitionnement du circuit et une affectation de chaque partition d'une manière analogue aux deux machines citées. Les performances pour-

raient alors être du mêm ordre de grandeur que ces deux machines. UNe technologie particulièrement interessante pour ce type de machine peut être la 'WSI', McD 84 . Une tranche ('wafer') de silicium de quatre pouce carré peut contenir environ une centaine de circuits élémentaires, disposer de plus de 2000 plots d'entrée/sortie et constituer alors un réseau N x N de très grande taille. L'avantage de cette technologie est la réduction des temps de communication entre circuits élémentaires, du fait de leur proximité. Elle est particulièrement adaptée aux mémoires, aux machines cellulaires et parallèle. Dans notre cas, elle pourrait diminuer les problèmes de communication entre circuits.

Par exemple, le 'restructurable VLSI', développé au MIT contient 192 cellules de 4 compteurs 10 bits chacun; Texas Instruments a conçu une mémoire de 32 Kbits sur une tranche de 4 pouce carré McD 84 . L'inconvénient de cette technologie est actuellement le faible rendement de fabrication obtenu. Dans les deux machines citées ci-dessus, à peine 50% des cellules fabriquées sont en état de marche. Cependant on peut s'attendre à une augmentation du rendement dans les années à venir. De plus, le fait de concevoir des cellules extrèmement simples devrait per-

mettre des rendements plus élevés. Après fabrication, le test des réseaux de base permettrait de détecter les bons et mauvais; l'initialisation pourrait avertir chaque cellule de l'état de ces voisines, et l'algorithme d'acheminement en tiendrait compte de manière à faire contourner les zones en pannes par les messages. De tels algorithmes sont actuellement à l'étude.

C O N C L U S I O N

****

****

******

Le but de cette thèse est d'étudier une architecture cellulaire ayant une topologie précise pour résoudre les problèmes de la simulation logique de gros circuits. Ceci nous a conduit à étudier les problèmes associés à la simulation logique, modèlisation, algorithme, machines spécialisées, exposés dans la première partie. Nous avons proposé une topologie cellulaire simple permettant des communications asynchrones à travers celle-ci. Cette architecture nous a amené devant des problèmes de placement, d'acheminement qui sont des activités à part entière dans la CAO. Différentes solutions de placement et d'acheminement ont été envisagées, puis simulées. Les comparaisons des résultats de simulation ont permis de sélectionner un type de placement et un type de routage adaptés à notre architecture. Les algorithmes choisis ont permis de détailler l'architecture d'une cellule de base permettant un acheminement performant et un traitement de simulation élémentaire. La cellule de base a pu être simulé de manière logique et électrique et est actuellement en cours de dessin.

L'originalité de cette étude est dans la démarche effectuée pour la conception; l'implantation avec des restrictions de taille, de topologie est sans

cesse présente à l'esprit, et c'est elle qui a guidé l'étude sur cette machine.L'inconvénient en est le caractère limité, spécialisé de l'architecture. L'avantage en est la facilité d'implantation puisqu'elle se réduit quasiment à l'étude détaillée d'une cellule de base.

Au niveau de l'intérêt de cette architecture pour la simulation, les performances annoncées semblent très interessantes. Elles sont comparables ou supérieures à celles annoncées pour les machines spécialisées existantes, et de toute façon bien supérieure à celles des simulateurs logiciels. Il est cependant nécessaire d'envisager de grands réseaus N x N pour traiter des circuits de grande taille. Une limitation est dans le choix de la simulation 3 états; il est intéressant de pouvoir effectuer des simulations temporelles et donc d'enrichir les possibilités de la machine.

Des extensions possibles pour la simulation peuvent être de rajouter les états haute-impédance puis transitoires. La partie information des messages doit alors être enrichi pour véhiculer les états supplémentaires. Les cellules doivent pouvoir repérer les entrées des commandes (interrupteurs par exemple). La partie calcul doit contenir le fonctionnement des portes de

base avec ces états... On peut noter que dans notre proposition, les messages de 10 bits ne véhiculent que 2 bits d'information. La partie traitement semble donc disproportionnée par rapport à la partie acheminement. Cependant, les deux parties ètantrelativement indépendantes, il est facile d'étendre les possibilités de la partie calcul.

Nous pensons que cette architecture peut être adaptée facilement à d'autres domaines que la simulation.

Si l'on regarde le fonctionnement de l'acheminement dans le réseau, on s'aperçoit que celui-ci est en fait un circuit d'acheminement asynchrone à dimension N x N. Chaque cellule peut envoyer un message à n'importe qelle autre dans le réseau. Ceci en fait une machine plus générale qu'une machine de simulation.

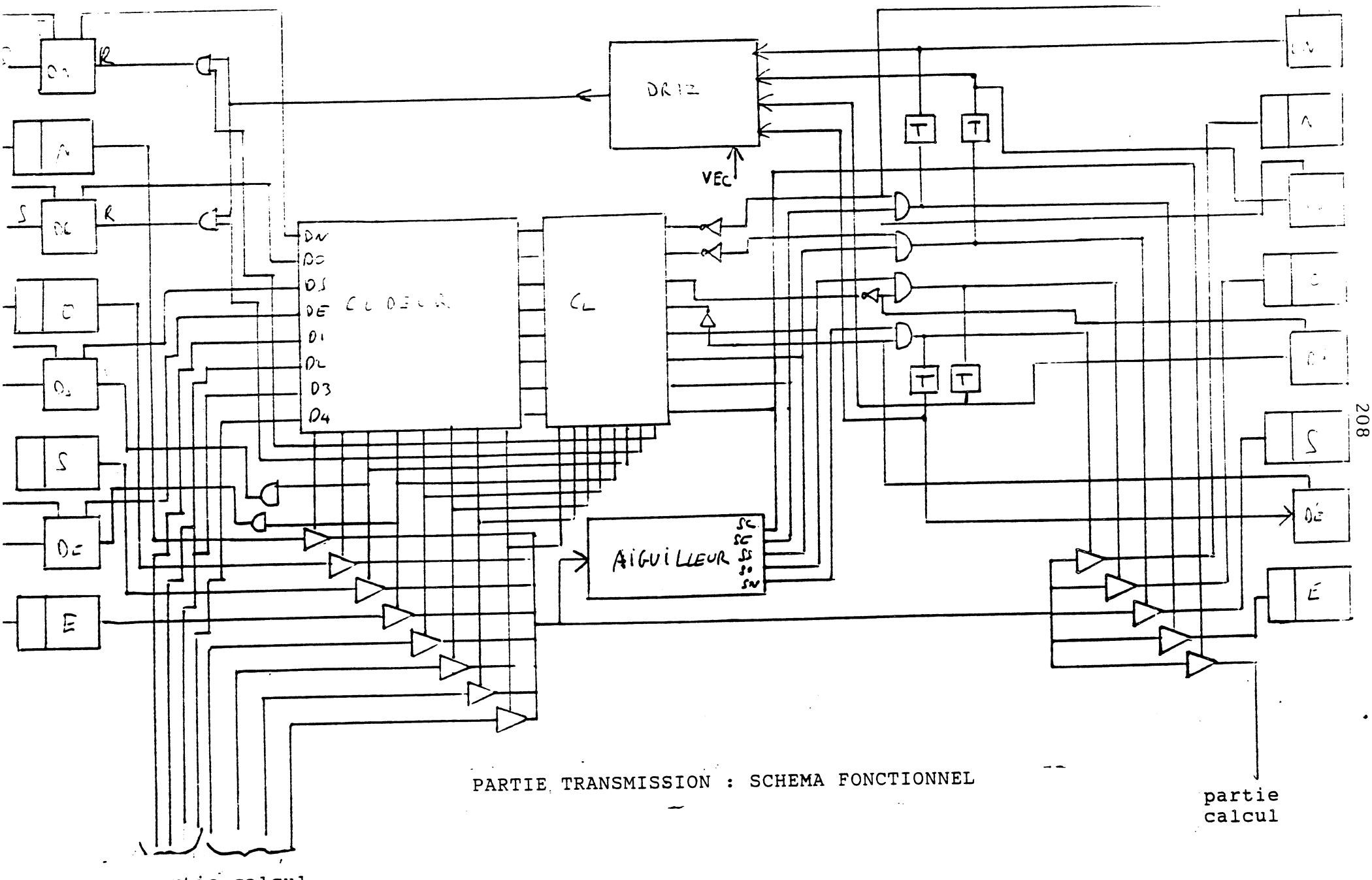
Il suffit pour celà d'associer une partie calcul adaptée au problème envisagé et de dimensionner de manière adéquate la partie acheminement. Par exemple des travaux sont en cours sur l'emploi d'un tel réseau d'acheminement dans la résolution de problèmes neuroniques [ANS 85].

Du point de vue de la conception de circuits intégrés, des architectures basées sur une topologie simple comme des réseaux cellulaires sont facilement et ra-

pidement concevables et implantables; ils peuvent permettre de résoudre par des solutions materielles des problèmesalgorithmiques, la 'quantité' de matériel n'ètant plus un obstacle de réalisation.

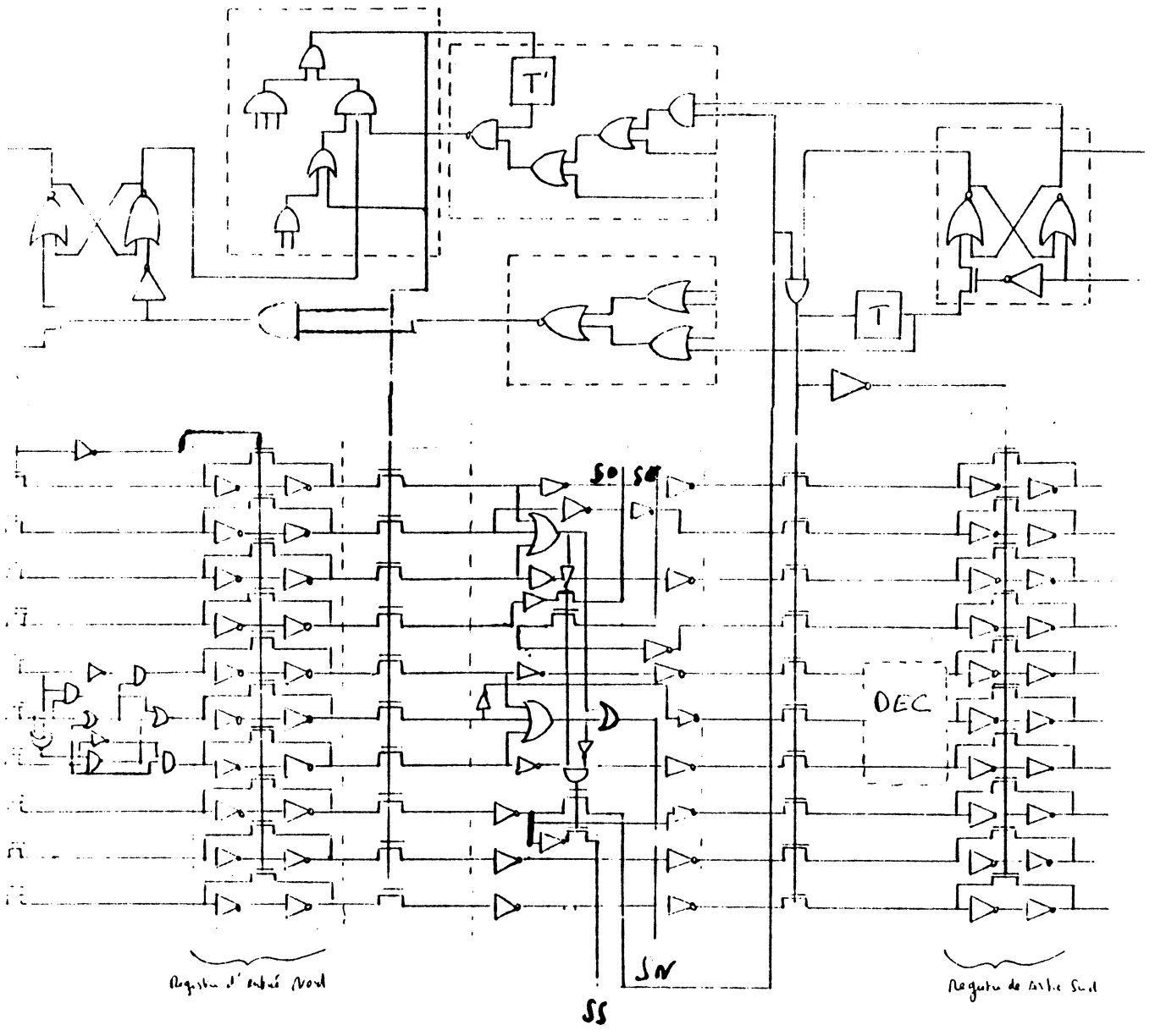
A N N E X E

##########

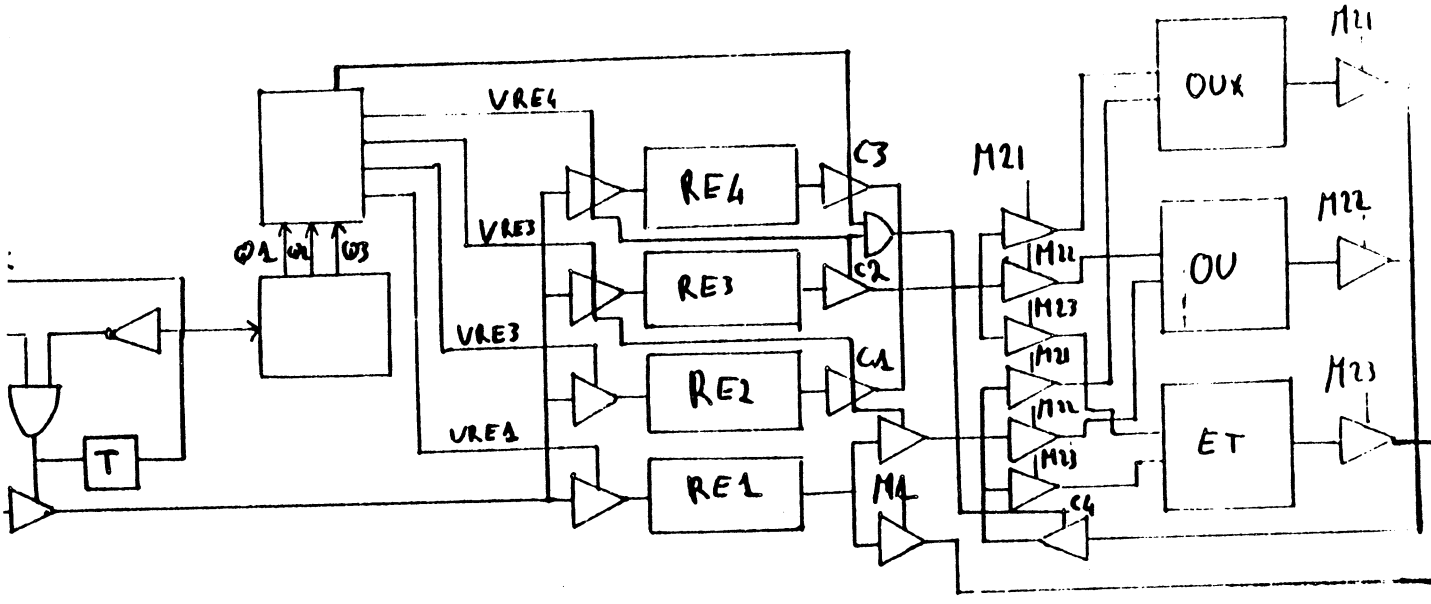


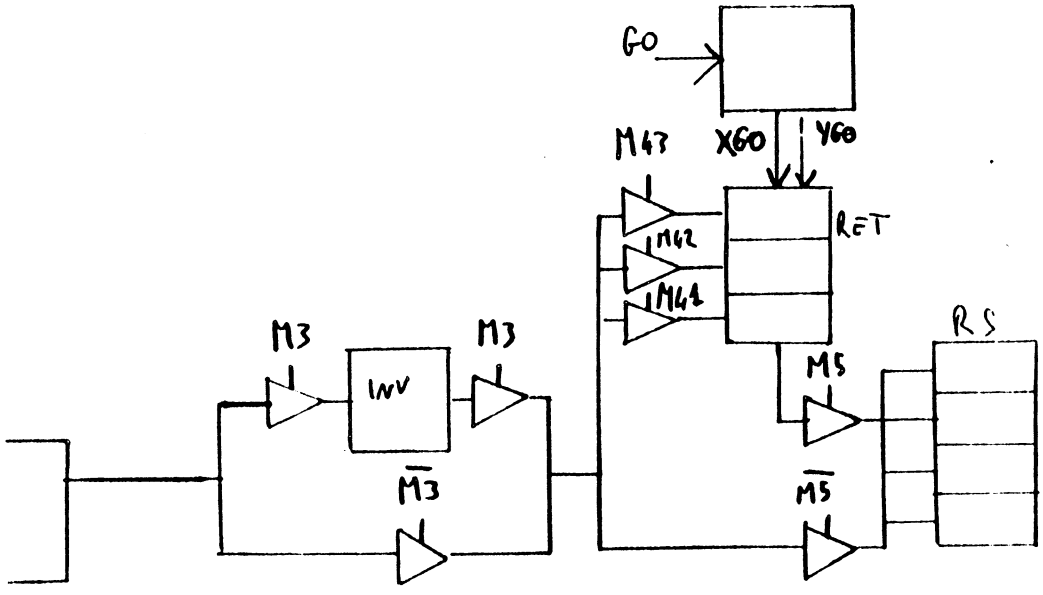
PARTIE TRANSMISSION : SCHEMA FONCTIONNEL

PARTIE TRANSMISSION : schéma détaillé(Nord-Sud)

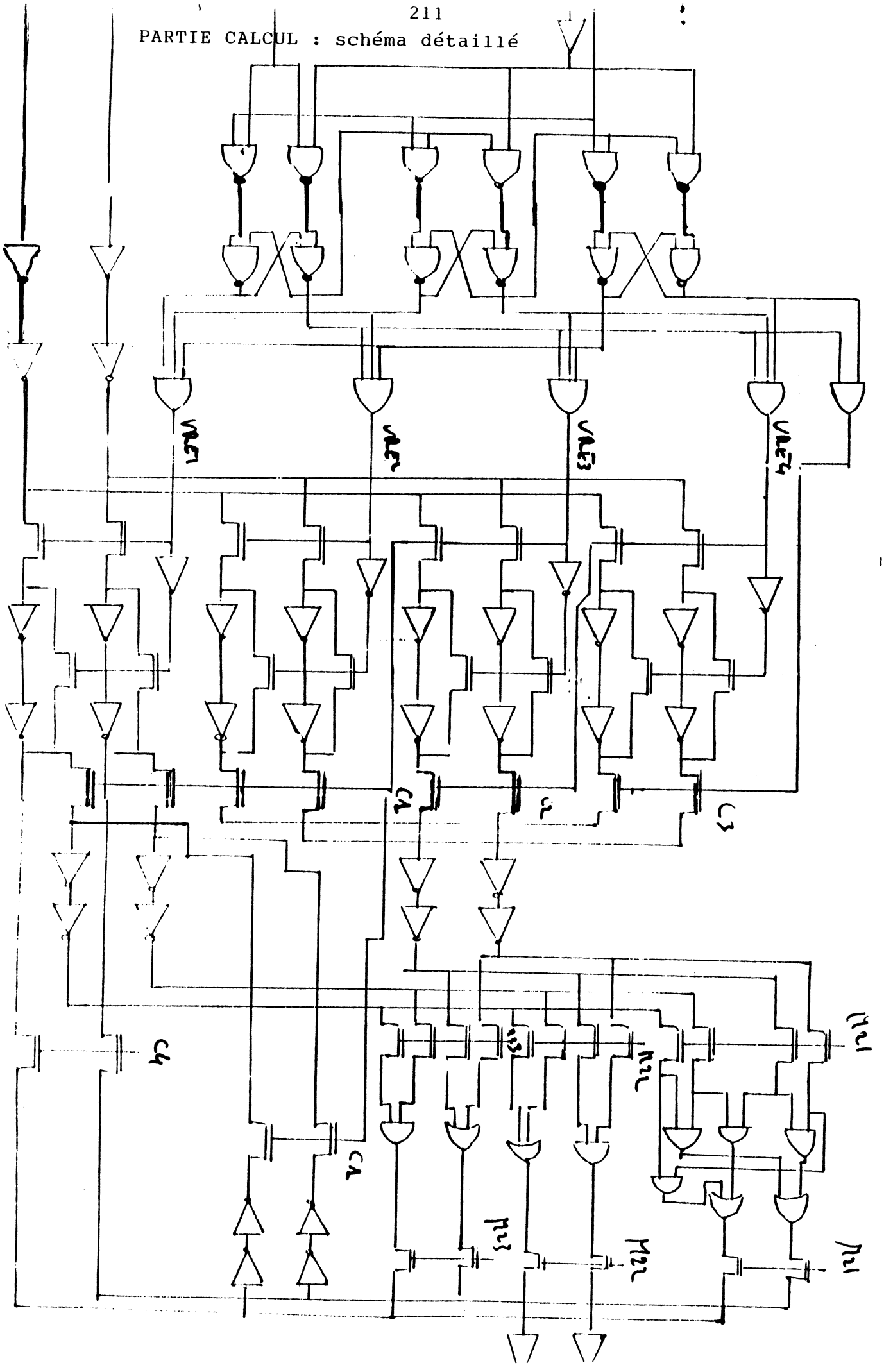


PARTIE CALCUL:schéma fonctionnel

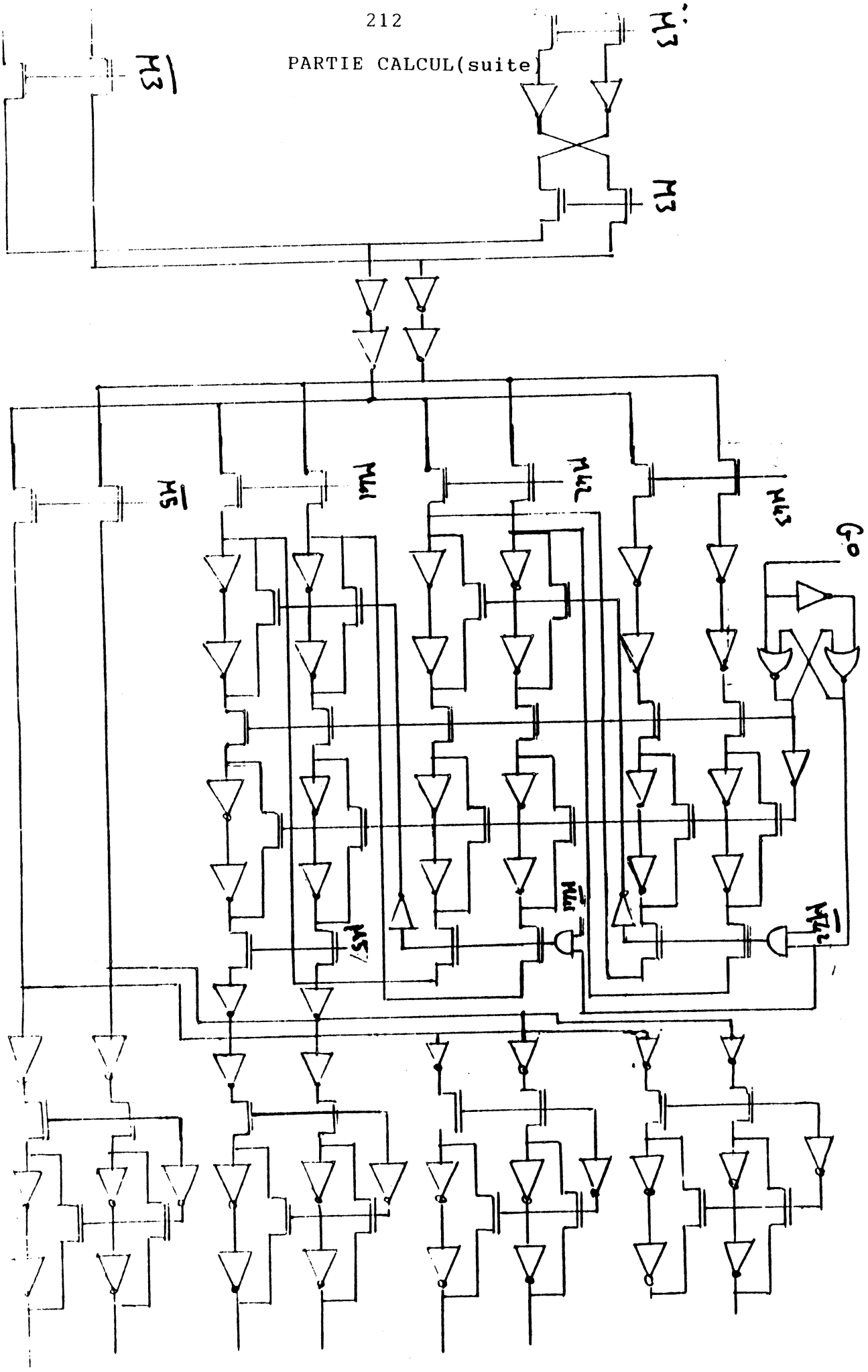




PARTIE CALCUL : schéma détaillé

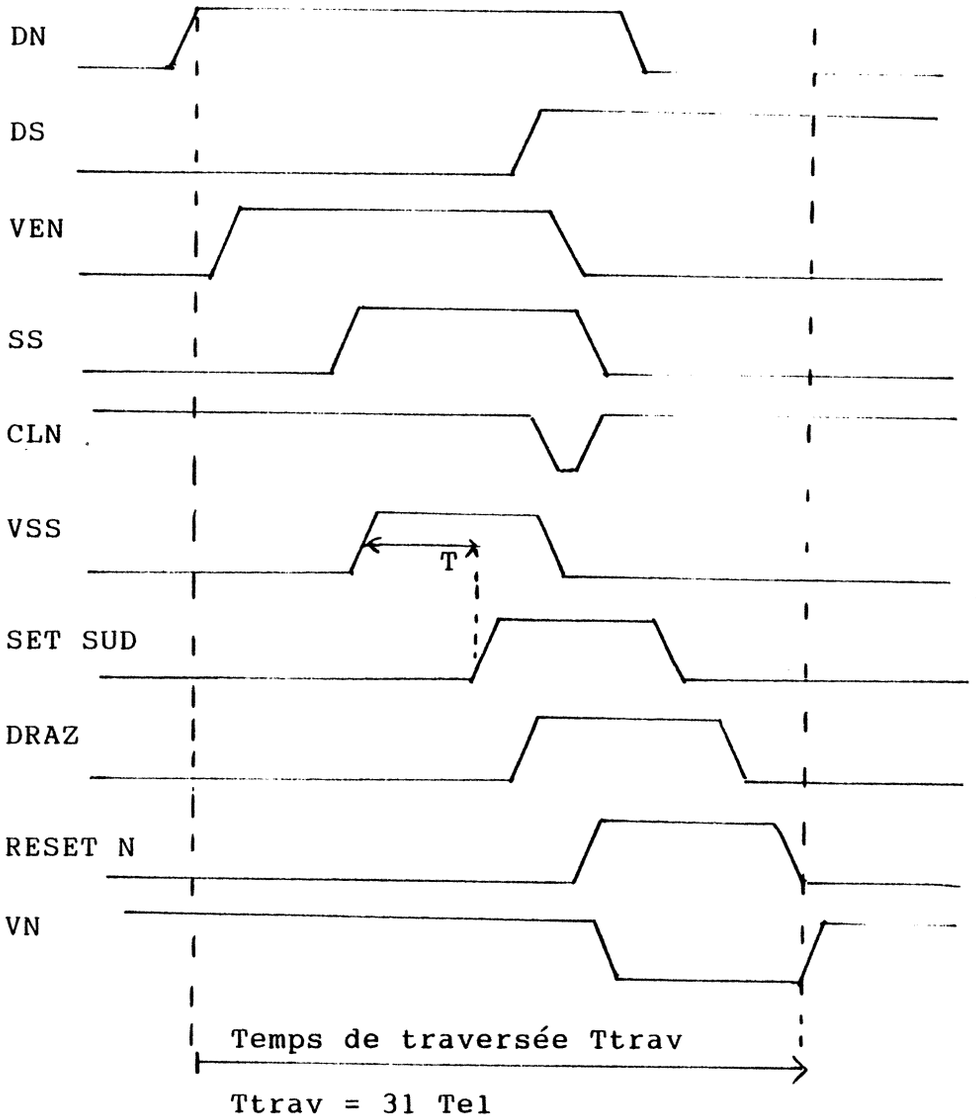


PARTIE CALCUL(suite)



Exemple de traversée de cellule

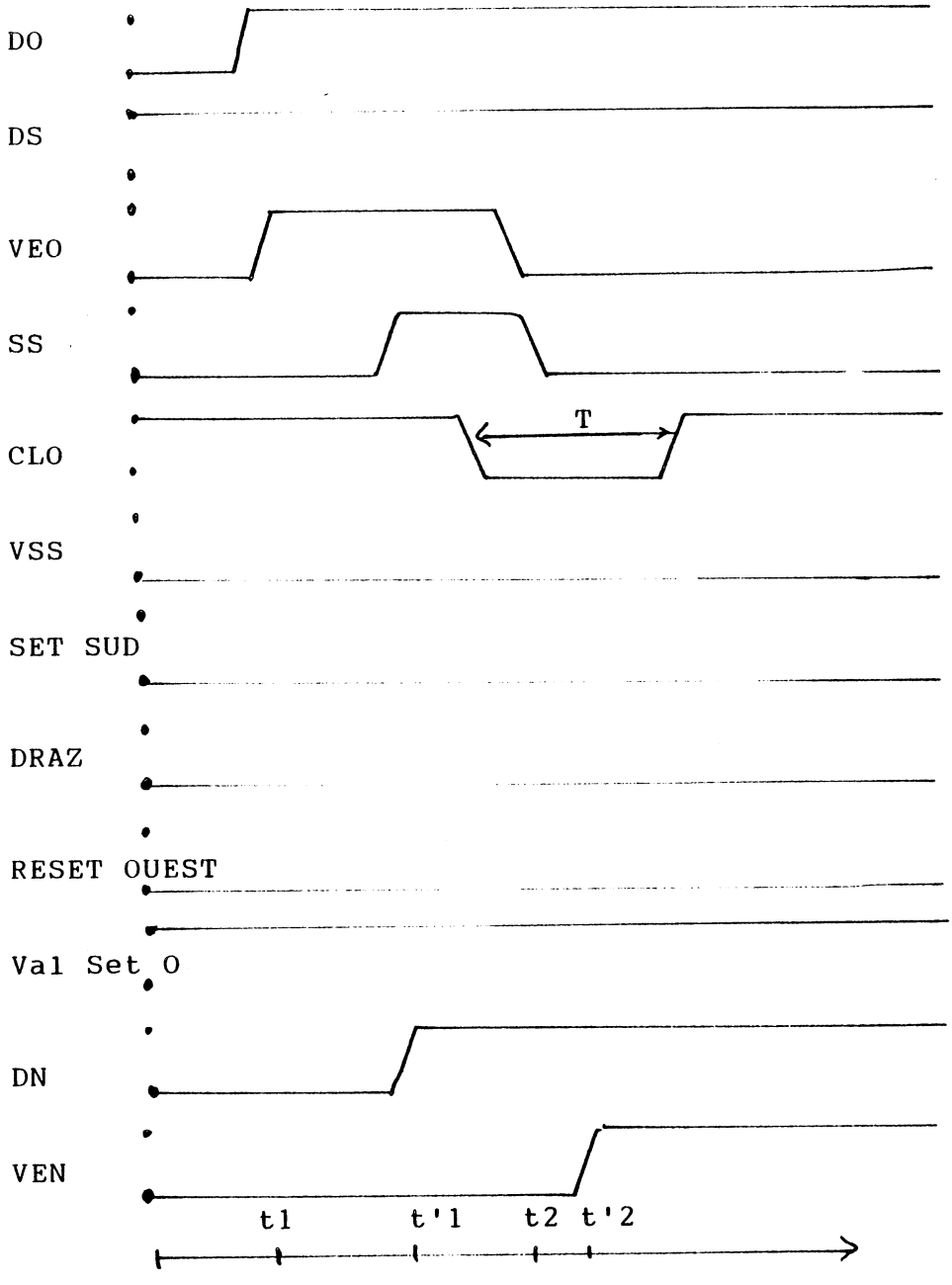
Chronogramme de traversée. Nord-Sud



Ttrav = 31 Tel

T = temps d'écriture en sortie

Exemple de traversée avortée nord-sud



Tanul = t'2 - t'1 = 16.Tel

BIBLIOGRAPHIE

****

BIBLIOGRAPHIE

[ABR82] M. ABRAMOVICI, Y. LEVENDEL, P. MENON
"A logic simulation machine"
Proc. 19th Design Automation Conference, pp.65-73, June 1982.

[ANS85] Y. ANSADE, J-P. BERNARD, G. MAZARE
"A highly parallel architecture dedicated to logical simulation"
2nd International Symposium on VLSI technologies, TAIWAN, May 1985.

[BAR80] R. BARTO, S. SZYGARDA, E. THOMPSON
"A computer architecture for digital logic simulation"
Electronic Engineering, pp.35-68, September 1980.

[BAT80] K. BATCHER
"Design of a massively parallel processor"
IEEE trans. on computers C-29, pp.836-840, 1980.

[BLA81] T. BLANK, W. VAN CLEEMPUT, Y. STOFIK
"A parallel bit map processor for DA algorithms"
Proc. 18th Design Automation Conference, June 1981.

[BLA82] T. BLANK
"A bit map architecture and algorithms for design automation"
PhD thesis, University of Stanford, September 1982.

[BRE72] M. BREUER
"Design Automation of digital systems"
Prenctice Hall, 1972.

[BRE77] M. BREUER
"Mui-cut placement"
Journal of Design Automation and fault tolerant computing, Vol.1, n°4, pp.343-362, October 1977.

[BRE81] M. BREUER, SHAMSA
"A hardware router"
Journal of digital system, Vol.4, p.393, 1981.

[BRY80] R. BRYANT
"An algorithm for MOS logic simulation"
Lambda magazine, fourth quarter 1980, pp.46-53.

[BRY81] R. BRYANT
"A switch-level model of MOS logic circuits"
MIT memo, n° 81-52, June 1981.

[CHY83] D. CHYAN, M. BREUER
"A Placement algorithm for array processors"
Proc. 20th Design Automation Conference, p.182,
June 1983.

[CNE84] M. ALBAREIL, J. LECOURVOISIER, A. PUISSOCHET
"CASSIOPEE, présentation générale"
CNET, note technique, Mai 1984.

[DAI84] Daisy
"Manuel de Présentation, MEGALOGICIAN"
1984.

[DEM81] H. De MAN and al
"LOGMOS, a transistor oriented logic simulator
with assignable delays"
Proc. ICCC September 1981, pp.42-45.

[DEN82] M. DENNEAU
"The Yorktown Simulation Engine"
Proc. 19th Design Automation Conference, June 1982,
pp.55-60.

[DUP84] G. DUPENLOUP
"Tracé automatique de canaux d'interconnexion"
Thèse 3° cycle, Grenoble, Juin 1984.

[EFC82] EFCIS
"Manuel de référence EPISODE"
1982.

[GIN83] L. GINDRAUX, G. CATLIN
"CAE Station's simulators tackle 1 Million gates"
Electronic Design, 10 Nov. 1983, p.127.

[GLA84] M. GLAZER, A. AMBLER
"Ultimate : A hardware logic simulation engine"
Proc.21st Design Automation Conference, June 1984,
pp.336-342.

[GOT79] S. GOTO
"A two-dimensional placement algorithm for the master Slice layout problem"
Proc. 16th Design Automation Conference, June 1979, pp.11-17.

[HAN72] M. HANAN, J-M. KURTZBERG
"Placements techniques"
Design Automation of digital systems, M. BREUER Ed., Prenctice Hall, Vol.1, pp.213-282, 1972.

[HAN73] M. HANAN, P. WOLFF, B. ANGULI
"Some experimental results ou placements techniques"
Proc.13th Design Automation Workshop, June 1973, pp.214-224.

[HAR84] W. HARDING, M. Mc WILLIAMS, J. RUBINS
"Twin processors speed CAE workstation's complex simulations"
Electronic Design, May 31. 1984, p.79.

[HEY83] M. HEYDEMANN
"A survey of MOS logic simulation tools"
Proc. ESSIRC 83, September 1983, pp.19-24.

[HIL81] D. HILLIS
"The connection Machine"
MIT report n° 646, September 1981.

[HON81] S. HONG, R. NAIR, E. SHAPIRO
"A physical design machine"
VLSI 81, Gray Ed. London, p.346.

[IOS80] A. IOSUPOVICZ
"Design of an iterative array maze router"
Proc. ICCC, pp.908-911, October 1980.

[KAN83] R. KANE, S. SAHNI
"A systolic design rule checker"
TR 83-13, Computer Science dpt., University of Minnesota, July 1983.

[KRO82] E. KRONSTADT, G. PFISTER
"Software support for the Yorktown simulation engine"
Proc. 19th Design Automation Conference, June 1982, pp.60-64.

[KUN80] H.T. KUNG, E. LEISERSON
"Systolic Arrays"
in introduction to VLSI Systems by Mead and Conway, Addison - Wesley, 1980, chap. 8.3.

[LAU79] U. LAUTHER
"A min-cut placement algorithm for general cell assemblies based on a graph representation"
Proc. 16th Design Automation Conference, pp.1-10, June 1979.

[LEV82] Y. LEVENDEL, P. MENON, S. PATEL
"Special-purpose computer for logic simulation"
The BELL System technical Journal, pp.2873-2909, December 1982.

[LOU80] R. LOUGHEED, D. Mc CUBBREY
"The Cytocomputer : A practical image processor"
Proc. 7th annual international symposium on Computer architecture, pp.271-277, May 1980.

[MCD84] Mac DONALD, ROGERS
"The trials of WSI"
IEEE Spectrum, October 1984.

[NAI82] R. NAIR
"Global wiring on a wire routing machine"
Proc. 19th Design Automation Conference, p.224, June 1982.

[NEW80] A. NEWTON
"Timing, Logic and mixed-mode simulation for large MOS integrated circuits"
NATO summer school, SOGESTA Urbino, Aug. 1980, pp.175-239.

[PFI82] G. PFISTER
"The Yorktown Simulation Engine : Introduction"
Proc. 19 Design Automation Conference, pp.51-54, June 1982.

[PHI78] N. PHILIPP, J. TELLIER
"Efficient event manipulation, the key to large scale simulation"
Proc. semi conductor test Conference, October 1978.

[PRE78] B. PREAS, C. GWYN
"Methods for hierarchical automatic layout of Custom LSI circuit masks"
Proc. 15th Design Automation Conference, pp.206-212, June 1978.

[PRE79] B. PREAS, W. Van CLEEMPUT
"Placement algorithms for arbitrarily shaped blocks"
Proc. 16th Design Automation Conference, pp.474-480, June 1979.

[PRE83] K. PRESTON
"Cellular logic computers for pattern recognition"
Computer, January 1983, p.36.

[RUT84] R. RUTENBAR, T. MUDGE, D. ATKINS
"A class of cellular architectures to support physical design automation"
IEEE trans.on CAD, Vol.3, n°4, October 1984.

[SAS83] T. SASAKI and al.
"HAL : a block level hardware logic simulator"
Proc. 20th Design Automation Conference, pp.150-155, June 1983.

[SCH76] D. SCHWEIKERT
"PLAC:A 2 dimensional placement algorithm for the layout of electrical circuits"
Proc. 13th Design Automation Conference, pp.408-416, June 1976.

[SOU81] J. SOUKUP
"Circuit layout"
Proc. IEEE, Vol.69, pp.1281-1304, October 1981.

[STE83] P. STEVENS, G. ARNOUT
"BIMOS, a MOS oriented multi-level logic simulator"
Proc. 20th Design Automation Conference, June 1983, pp.100-106.

[SUD82] T. SUDO and al.
"An LSI adaptative array"
Digest of ISSC conference, February 1982.

[SZY72] S. SZYGENDA
"TEGAS2 - Anatomy of a general purpose test generation and simulation system for digital logic"
Proc. of Design Automation Workshop, n°9, 1972.

[THO75] E. THOMPSON, S. SZYGENDA
"Digital logic simulation in a time-based table-driven environment"
IEEE computer, n°3, March 1975, pp.24-37.

[UED83] K. UEDA and al.
"A parallel processing approach for logic module placement"
IEEE trans.on CAD, vol.2, pp.39-47, January 1983.

[UNG58] S. UNGER
"A computer oriented toward spatial problems"
Proc. of the IRE, October 1958, pp.1744-1750.

[WER83] J. WERNER
"Comparing the CAE systems in action"
VLSI design, November 1983.

[WER84] J. WERNER, R. BERESFORD
"A system engineer's guide to simulators"
VLSI design, February 1984.

[ZYC84] Zycad
"LE 1000 reference manual"
1984.

°°°

TABLE DES MATIERES

AUTORISATION DE SOUTENANCE

DOCTORAT 3ème CYCLE, DOCTORAT-INGENIEUR, DOCTORAT USMG

*Vu les dispositions de l'arrêté du 16 avril 1974,

Vu les dispositions de l'arrêté du 5 juillet 1984,

Vu les rapports de Mr. LANDRAULT.....

Mr. MAZARÉ.....

Mr. Jean-Pierre BERNARD..... est autorisé

à présenter une thèse en vue de l'obtention du grade de DOCTEUR-INGÉNIEUR

en Microélectronique.....

Grenoble, le 20 JUIN 1985

Le Président de l'Université Scientifique

et Médicale

UNIVERSITÉ SCIENTIFIQUE ET MÉDICALE DE GRENOBLE

M. TANCHE

RESUME

Cette thèse propose une architecture cellulaire pour simulation logique.

Une première partie présente la simulation logique. Les modèlisations, algorithmes, structures des Simulateurs classiques sont décrits. Un recensement et une étude sommaire des machines spécialisées existantes sont proposés ainsi qu'une classification des machines cellulaires.
Une deuxième partie présente les spécifications d'une architecture cellulaire et propose des solutions aux problèmes d'affectation et d'acheminement soulevés.
Une troisième partie décrit la cellule de base d'une manière détaillée dans un réseau 8x8 et évalue la complexité et les performances attendues.
Une dernière partie expose les performances globales de simulation du réseau de base et cite quelques extensions dans et hors de la simulation logique.

MOTS-CLES

Simulation logique- architecture cellulaire- circuit intégré- réseau asynchrone- accélérateur de simulation.