REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'EDUCATION NATIONALE

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT D'ELECTRONIQUE

PROJET DE FIN D'ETUDES

SUJET

RESEAUX SYSTOLIQUES : PROGRAMMATION EN LANGAGE OCCAM

Proposé par : Mr A.FARAH

Etudié par : Melle A.LAROUI

Dirigé par : Mr A.FARAH

Promotion : Juin 93

E.N.P. 10,Avenue Hacen Badi El Harrach - ALGER


DEDICACES

A LA MEMOIRE DE MA GRAND-MERE AICHA

A MES PARENTS ADORES

A TOUS CEUX QUI ME SONT CHERS

ABLA


REMERCIEMENTS :

Je tiens à éxprimer mes sincères remerciements à :

-Mr A.FARAH Maitre de conférences à L'ENP pour avoir bien voulu me proposer le sujet, me guider et me diriger par ses conseils éclairés et ses heureuses suggestions.

-Mr H.GOUGAM Enseignant à L'ENELEC pour avoir bien su me guider et orienter mes réflexions.

-Mr N.NADIR Ingénieur en Informatique pour l'extreme disponibilité dont il a fait preuve à mon égard tout le long de mon travail.

-Mme K.DIB Docteur en Informatique pour toute son aide bibliographique apportée pour la réalisation de ce projet.

SOMMAIRE

ملخص :

تمثل الشبكات الإنقباضية هدف هذه الدراسة التي عرضت فيها خصائصها ومبادئها. ومن أجل إعطاء نموذج لهذه الشبكات تمت دراسة هيكل أسلوب اوكام.
وأخيرا تم إدخال الترنسبيوتر الذى لا يتجزء من اوكام مصحوب ببعض التطبيقات.

RESUME :

Les réseaux systoliques font l'objet de cette étude.Leur caractéristique et leurs principes y sont exposés.

La structure du langage Occam, a été étudiée dans le but de simuler ces réseaux.

Le transputer indissociable de l'Occam y a été introduit, accompagné de quelques applications.

ABSTRACT :

The Systolic networks are the subject of this study.Their caracteristics and principles have been exposed.

The structure of Occam language has been elaborated in order to simulate these networks.

The transputer, infinitely related to the Occam has been introduced in this study,togheter with some applications.

INTRODUCTION

INTRODUCTION

L'évolution des citcuits VLSI (Very Large Scale Integration) cette dernière decennie à conduit à la réalisation de plusieurs types de composants électroniques permettant de diverses applications aussi intéressantes les unes que les autres où l'on distingue les réseaux systoliques qui, grâce à leur architecture paralléle peuvent traiter un trés grand nombre de données et ceci pendants un laps de temps trés appréciable, de plus, ces réseaux détiennent un rôle trés important dans le domaine des implémentations des algorithmes numériques : algorithmes des opérations matricielles, ou encore dans le domaine de traitement d'image et autres.

La premiére partie du projet est consacrée aux réseaux systoliques. Elle se compose du : Chapitre I, il nous introduit dans le domaine des architectures paralléles, où on donne une vision sur ces architectures et leur classification, dans le Chapitre II, on parlera des caractéristiques essentielles des réseaux systoliques, le Chapitre III décrit l'implémentation Hardware des réseaux systoliques ou plusieurs exemples d'algorithmes d'opérations matricielle sont donnés.

La deuxiéme partie est consacrée : à la programmation concurrente (Chapitre I), une étude bien détaillée du langage concurrent Occam (Chapitre II) et au Transputer et ses applications (Chapitre III), dans le chapitreIV, on verra comment programmer des réseaux systoliques en utilisant Occam pour l'implémentation Software des algorithmes,on dennera aussi des exemples de programmation en Occam.

Nous donnerons enfin une conclusion génerale sur l'étude présentée.

A. PREMIERE PARTIE : Reseau Systolique

CHAP I : Architectures Paralléles

CHAP II : Réseaux Systoliques

CHAP III : Implementation Hardware des réseaux systoliques

Conclusion

CHAPITRE I :

ARCHITECTURES PARALLELES

I.1 INTRODUCTION :

Le concept de parallélisme ne va plus avec l'approche qui consiste à gagner de la vitesse en effectuant plus rapidement chaque opération. En calcul paralléle , le gain de vitese provient de la réalisation simultanée de plusieurs opérations. Le recours au parallélisme pour accélérer des calculs n'est guère une idée nouvelle, mais ce n'est que récemment que la technologie a permis de construire des architectures multiprocesseurs et de les utiliser efficacement.

Comme le nom l'indique, les multiprocesseurs sont des ordinateurs possédant plusieurs processeurs (de 2 à plusieurs dizaines pour des systèmes généraux ou beaucoup plus pour des machines spécialisées). L'architecture se complique , ainsi les problémes d'accès à la mémoire deviennent cruciaux pour pouvoir acheminer des données au rythmes de traitement des processeurs; de même les problèmes de communication et de synchronisation entre processeurs sont importants.

I.2 CLASSIFICATION DES ARCHITECTURES PARALLELES :

Plusieurs modèles d'architectures parallèles ont été proposés. Pour les classifier , le principe de classification le plus repandu est la classification selon la multiplicite des flux d`instructions et de données disponibles matériellement , cette classification comporte (FIG 1):

- Le mode SISD (single instruction , single data streams): c`est le mode de fonctionnement de la plus part des systemes conventionnels utilisant des microprocesseurs classiques. Les instructions sont éxecutées séquentiellement. Cette architecture gére un flux d`instruction et un flux de données a la fois.

- Le mode SIMD (single instruction , multiple data streams): c`est le mode de fonctionnement priviligie des matrices de processeurs . (FIG 1a) Plusieurs processeurs élémentaires identiques éxecutent la même instruction sur des données différentes , sous la supervision d`une unite de contrôle. Chaque opérateur opére sur son propre

flux de données. Dans ce modèle, plusieurs unités de traitement reçoivent la même instruction diffusée par l'unite de contrôle,mais opére sur des ensemble de données distincts,provenant de flux de données distincts. Chaque processeur éxecutant la même instruction au même instant , on obtient un fonctionnement synchrone.La mémoire partagée peut être subdivisée en plusieurs modules.Dans ce cas,l'accés des unités de traitement aux differents modules se fait par un réseau d'interconnexion.

- Le mode MIMD (multiple instruction, multiple data streams): C'est un mode reservé aux grands ordinateurs ils ont de multiples flux d'instructions agissant sur de multiples flux de données (FIG 1b). Dans ce cas les flux de données sont issus d'une mémoire partagée entre les processeurs du systèmes, chaque processeur éxecutant son propre programme.Il peut y avoir divers types d'interactions entre differents flux d'instruction et de données.

Ci : Contrôle
Pi : Processeur
Mi : Mémoire

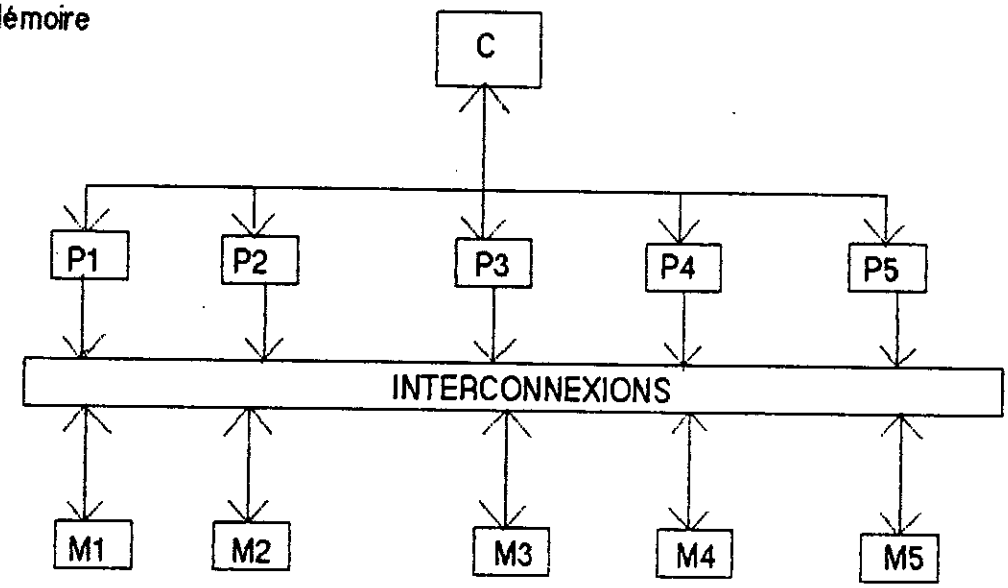


FIG (1a) Architecture SIMD

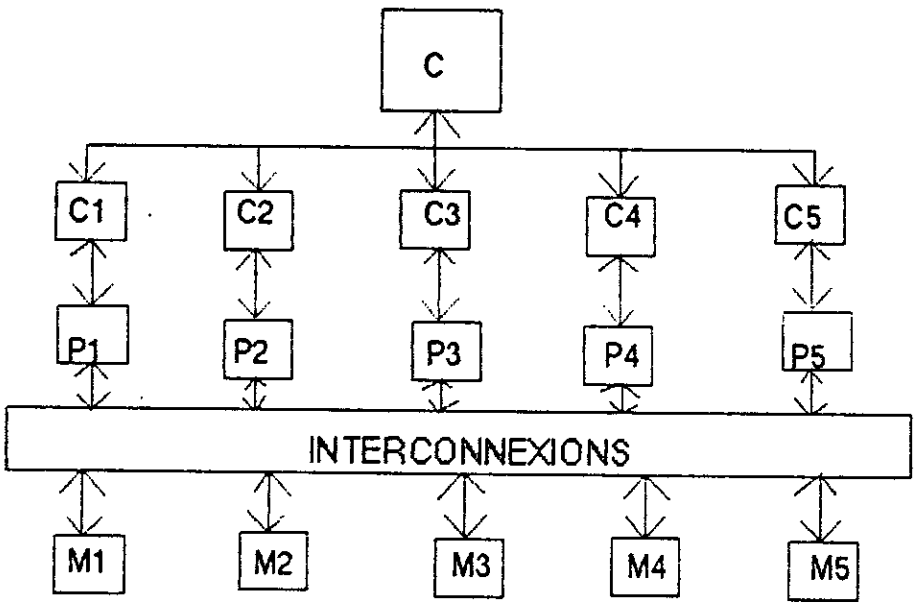


FIG (1b) Architecture MIMD

FIG (1) Architectures parallèles

Il existe d`autres classifications d`architectures paralléle fondé : sur le nombre de processeurs d`instructions et de données présent dans le système ,leur relation, le modèle mémoire des instructions et celui des données, ainsi que la manière dont les différents processeurs de données interagissent. La figure (2) donne une classification détaillée des architectures paralléles.

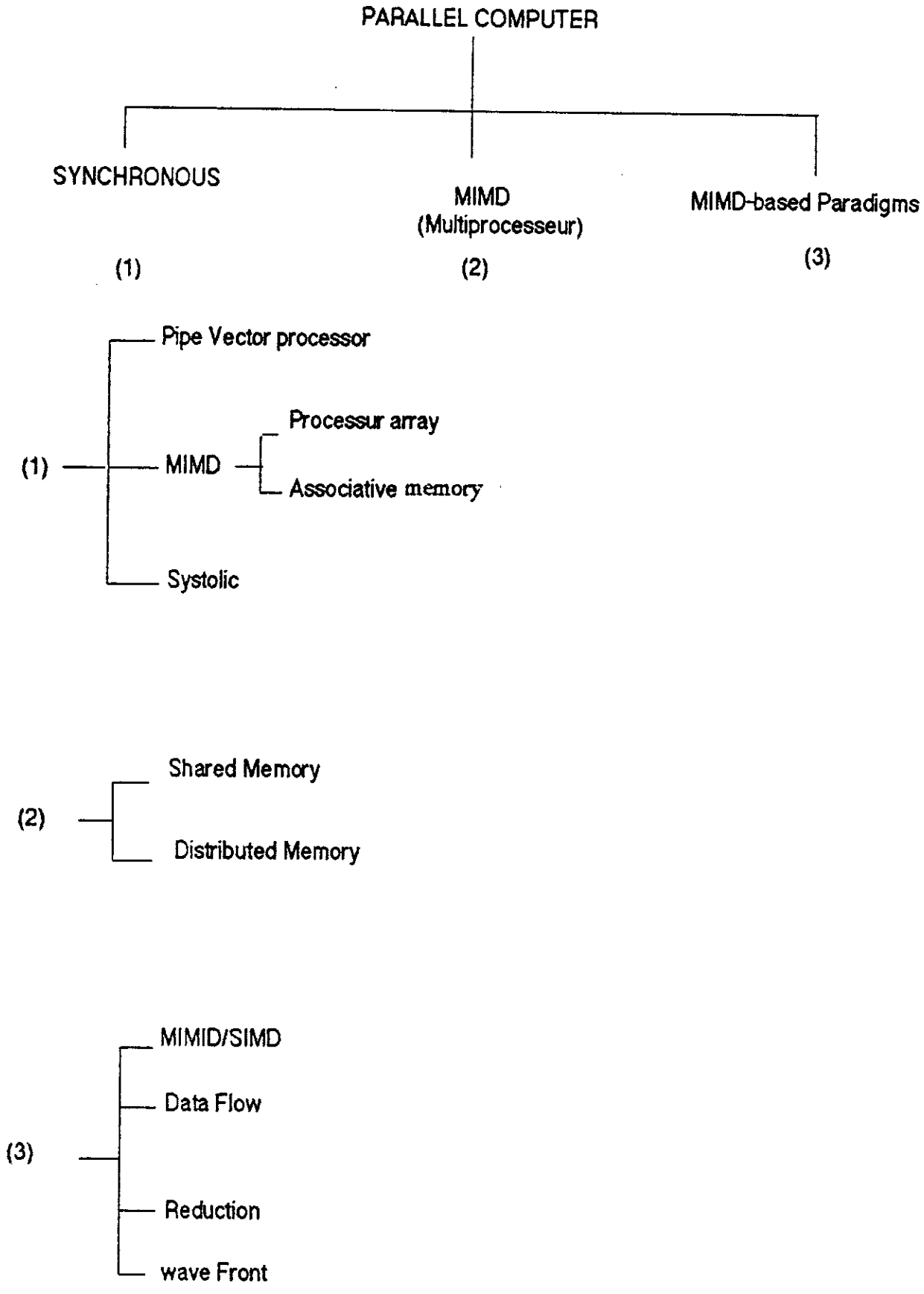


FIG (2) Classification des Architectures Paralléles

CHAPITRE II :

RESEAUX SYSTOLIQUES :

CARACTERISTIQUES

II.1 INTRODUCTION :

La complexite des circuits disponibles a l`heure actuelle rend possible la réalisation a un faible coût de systèmes massivement paralléles, tel que le système systolique introduit en 1978 qui s`est revelé être un outil puissant pour la conception de processeurs intégrés spécialisés.

Une architecture systolique est agencée en forme de réseau. Ces réseaux se composent d`un grand nombre de cellules élémentaires identiques et localement interconnectées. Chaque cellule reçoit des données en provenance des cellules voisines, éffectue un calcul simple, puis transmet les resultats toujours aux cellules voisines, un temps de cycle plus tard, Seules les cellules situées à la frontière du réseau communiquent avec le monde exterieur.les cellules évoluent en paralléle,sous le contrôle d`une horloge globale (synchronisme total) plusieurs calculs sont éffectués simultanement sur le réseau et on peut pipeliner la resolution de plusieurs instances du même problème sur le réseau. [2,8]

II.2 DEFINITION :

Le réseau systolique introduit par Kung et Leirson en 1978 est une machine paralélle spécialisée faite de processeurs ou de cellules connectés de façon régulière et locale capable d'exécuter des opérations simples (FIG3). Dans un réseau systolique,les calculs effectués utilisent à la fois la notion de pipline et de parallélisme vrai et sont exécuter de façon synchrone.

II.3 PRINCIPE DES RESEAUX SYSTOLIQUES :

Une architecture systolique est un réseau regulier de processeurs relativement simples, réalisés sous formes de circuits intégrés, ces données circulent à travers la structure de façon synchrone et selon un cheminement fixe.

Dans l'architecture systolique plusieurs calculs sont effectués sur une même données à l'intérieur du réseau une fois qu'une donnée en provenance de la mémoire externe est lue par le réseau elle passe de cellule en cellule et peut donc être utilisées de nombreuse fois.

Ce mode de calcul est à l'opposé du fonctionnement basé sur un accés mémoire à chaque utilisation d'une donnée en conséquence,un réseau systolique peut être étendu pour traiter un problème couteux en nombre d'opérations sans qu'il faille imposer pour autant une augmentation correspondante de la bande passante de la mémoire externe . Cette propriété est bien illustrée par la figure (4) , elle confère aux réseaux systoliques un avantage majeur sur les architectures traditionnelles, qui sont limités par le "goulot d'étranglment de Von Neumann".[3]

Une restriction cependant bien que seules les cellules frontières d'un réseau systolique communiquent avec le monde exterieur,il reste une limite incontournable en matières d' E/S,à savoir le nombre de plots disponibles sur un même boitier.Cette limitation est particulièrement importante pour les réseaux bidimensionnels faisant appel à l'arithmétique à virgule flottante,même si divers artifices comme l'utilisation des ports bidirectionnels ou de multiplixeurs peuvent en réduire la sévérité.

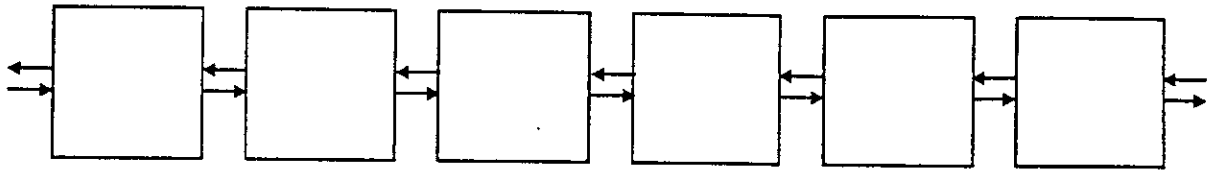


FIG. (3a) Réseau linéaire

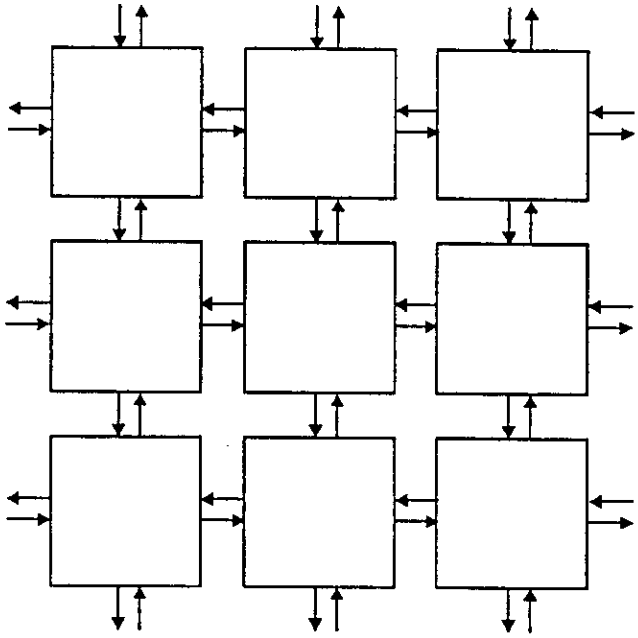


FIG. (3b) Réseau orthogonal

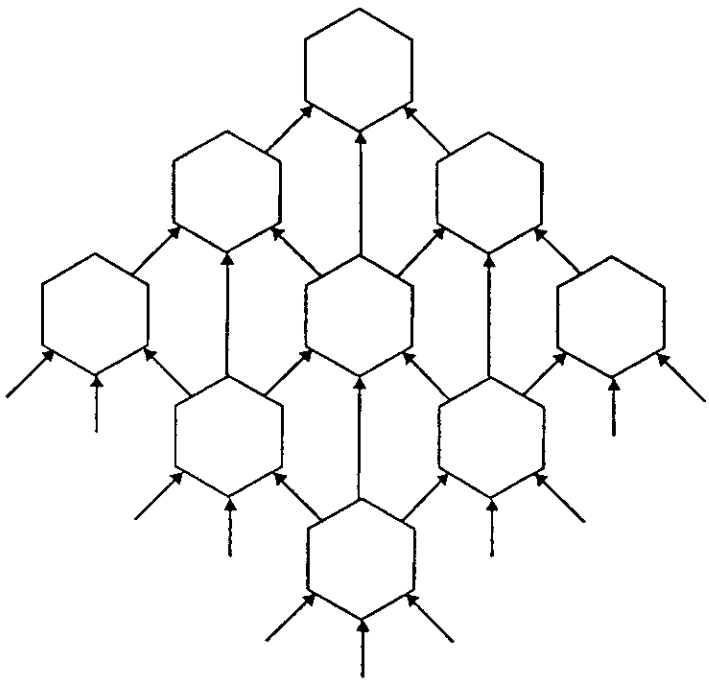
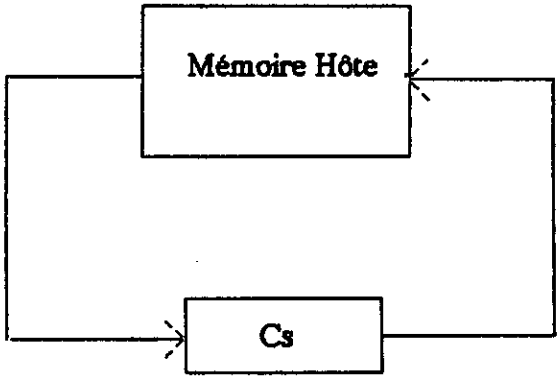


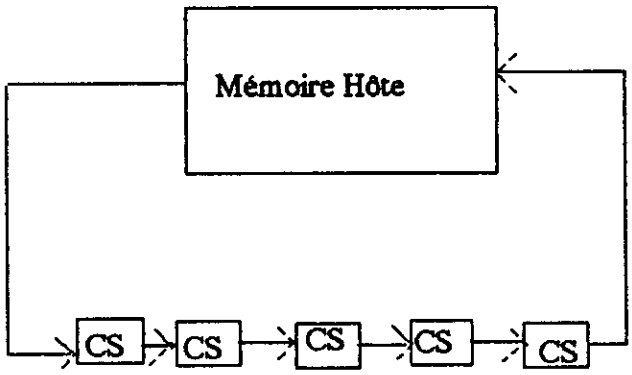
FIG. (3c) Réseau hexagonal

FIG (3) Exemples de topologie



CS :Cellule Systolique

FIG (4 a) Echange de données entre
une cellule systolique et une mémoire



CS : Cellule Systolique

FIG (4b-) Principe des réseaux systoliques

FIG (4)

On appélle "Systole" le mouvement de contraction du coeur, le terme évoque donc l'idée d'une pulsation rythmique comme on l'a déja vu une machine systolique se presente comme un réseau de processeurs, un ordinateur hôte injecte les données dans le réseau,l'ensemble du systéme fonctionne d'une manière totalement synchrone comme par "battement" au cour d'un battement chaque processeur reçoit de données nécessaires, les traites puis les envoies à ses voisins immédiats.ainsi l'information traitée progresse à travers le réseau subissant à chaque battement un calcul qui l'a combine avec les resultats produits par les processeurs immédiatement voisins .D'ou la nomination systolique qui provient de cette analogie entre la circulation des flux de données dans le réseau et celle du sang dans le corps humain,l'horloge qui assure la synchronisation globale constituant le coeur du système. [6]

II.4 SIMPLICITE ET REGULARITE :

La réalisation des circuits VLSI spécialisés à un côut résonable est une préoccupation majeur pour les concepteurs, le coût de conception d'un tel circuit doit

être assez bas pour pouvoir être amorti sur un faible volume de production.

Les architectures systoliques sont composées à partir d'un petit nombre de cellules de base, premier avantage sur une architecture composée d'une grande variété de cellules complexes . Deuxiéme avantage,l'interconnection locale et régulière des cellules facilite grandement l'implantation topologique. Dans le cas le plus simple celui des réseaux linéaires on peut même dire que tout algorithme systolique conduit directement à un schéma de réalisation sur silicium.

II.5 APPLICATION DES RESEAUX SYSTOLIQUES :

On distingue deux domaines d'application des réseaux systoliques et qui sont:

* Applications numériques et traitement du signal
  - Traitement du signal:
    . convolution unidirectionnelle
    . convolution bidirectionnelle et corrélation
    . lissage par médianne
    . transformée de Fourier discréte
    . projections géométriques

. systèmes adaptatifs

. filtres de Kalman

- Arithmétiques matricielles:

. multiplication matrice-vecteur

. multiplication matrice-matrice

. triangularisation de matrice

. résolution des systèmes triangulaires

. décomposition QR

. problèmes aux valeurs propres

* Applications non numériques:

- Structures de données:

. pile,file d'attente

. tri

. bases de données relationnelles

- Graphes et algorithme géométriques:

. fermeture transitive, problèmes de chemin dans un graphe

.arbre de degré minimum,composantes connexes

.enveloppes convexes

- Manipulation de chaines de caractères:

. occurences d'un mot dans une chaine

. plus longue sous-suite

. reconnaissance de langages

. programmation dynamique

- Algèbre des polynomes et arithmétique ;
  - multiplication, division euclidienne, PGCD de polynomes
  - arithmétique entière et dans les corps finis multiplication,division,PGCD

Notre étude se limitera a une application numérique plus précisement aux opérations matricilles ou nous aurons à étudier plus loin plusieurs algorithmes systoliques appliqués à ces opérations.

II.6 INTEGRATION D'UN CIRCUIT SYSTOLIQUE DANS UN SYSTEME :

La communication entre un circuit systolique et l'ordinateur hôte peut être représentée par la figure ci-dessous (FIG 5):

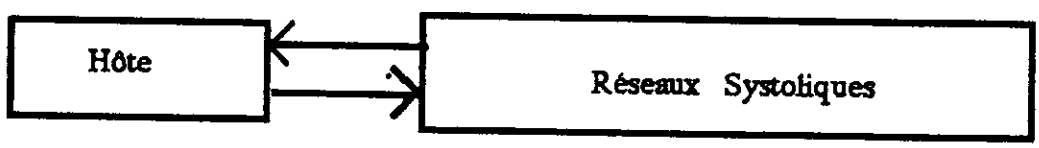


FIG (5) Structure d'un système Systolique

Un système systolique se compose :

- d'un ordinateur hôte: son rôle est de fournir la mémoire principale pour le stockage et la gestion des données.Il doit contrôler le système d'interface et le réseau d'interconnection, et génére le code objet qui sera chargé dans un processeur élémentaire.

- d'un système d'interface: celui-ci connecté à l'hôte par un bus, a pour fonction d'assurer le traitement intermédiaire des données depuis la mémoire centrale jusqu'en entrée du réseau systolique. Il comprend une unité de contrôle qui gère des registres temporaires trés rapides: ceux-ci doivent compenser le débit trés lent de la structures hôte,afin d'alimenter le réseau,dont la bande passante est beaucoup plus large,à plein régime.Ce système d'interface doit également être capable de réaliser certaines opérations élémentaires trés usuelles sur les données(permutations,transposition de matrices,...).

- d'un réseau d'interconnection:il s'agit d'assurer une gamme de communications entre les points de mémorisation et les ports d'entrées des différents processeurs : la reconfigurabilité des liaisons apporte la souplesse indispensable à une programation éfficace.

- d'un ou plusieur réseaux systoliques, les cellules des réseaux peuvent être dotées d'une mémoire locale.

II.7 CARACTERISTIQUES DES RESEAUX SYSTOLOIQUES :

A partir de leur structure, on peut en déduire que les caractéristiques essentielles et principales des réseaux systoliques sont:

* Un parallelisme massif et décentralisé.
* Des communications locales.
* Un mode opératoire synchrone.

Ce qui montre l'importance apportée par ses réseaux dans la résolution des problèmes ou les calculs sont trés importants que ça soit dans les applications matricielles ou autres domaines tel que le traitment de signal ou autres applications déja citées auparavant.

Les systèmes VLSI sont adaptés à l'implémentation d'algorithmes bornés par les calculs plutôt qu'à la résolution de problèmes bornés par les entrées/sorties. Dans un algorithme borné par les calculs, le nombre de calculs élémentaires est plus grand que le nombre de données en entrées/sorties.Sinon,le problème est borné par les entrées/sorties, et ne se prète pas à un traitement VLSI, ou le nombre de ports d'entrées/sorties est limité. Par exemple,la multiplication de deux matrices de taille n necessite $0(n^3)$ multiplications et $0(n^2)$ additions pour $0(n^2)$ données, c'est un problème borné par les calculs. Alors que l'addition de deux matrices éxige n^2 additions pour $3n^2$ opérations d'entrées. sorties , et donc borné par les entrées/sorrties. Les caratéristiques des architectures systoliques conduisent pour la plus part des problèmes bornés par les calculs,à des calculs en temps réel, ou les sorties sont délivrées au même rythme que les entrées. De fait, de telles architectures se sont avérées trés performantes pour la résolution de nombreux problèmes ou le volume de calcul est trés important,et le traitement local est régulier. [3,13]

CHAPITRE III :

IMPLEMENTATION HARDWARE DES RESEAUX SYSTOLIQUES

III.1 CONVOLUTION NON-RECURSIVE :

Soit une suite $x_1, x_2, x_3, \ldots$ de données on va calculer pour tout entier $i>=k$ la valeur de yi tel que:

$$yi = a_1x_i + a_2x_{i-1} + a_3x_{i-2} + \ldots + a_kx_{i-k+1}$$

ou les $a_1, a_2, \ldots a_k$ sont des coefficients constants. La solution la plus simple bien connue en théorie de filtrage numérique est illustrée par la figure (6) (pour k=4). [3]

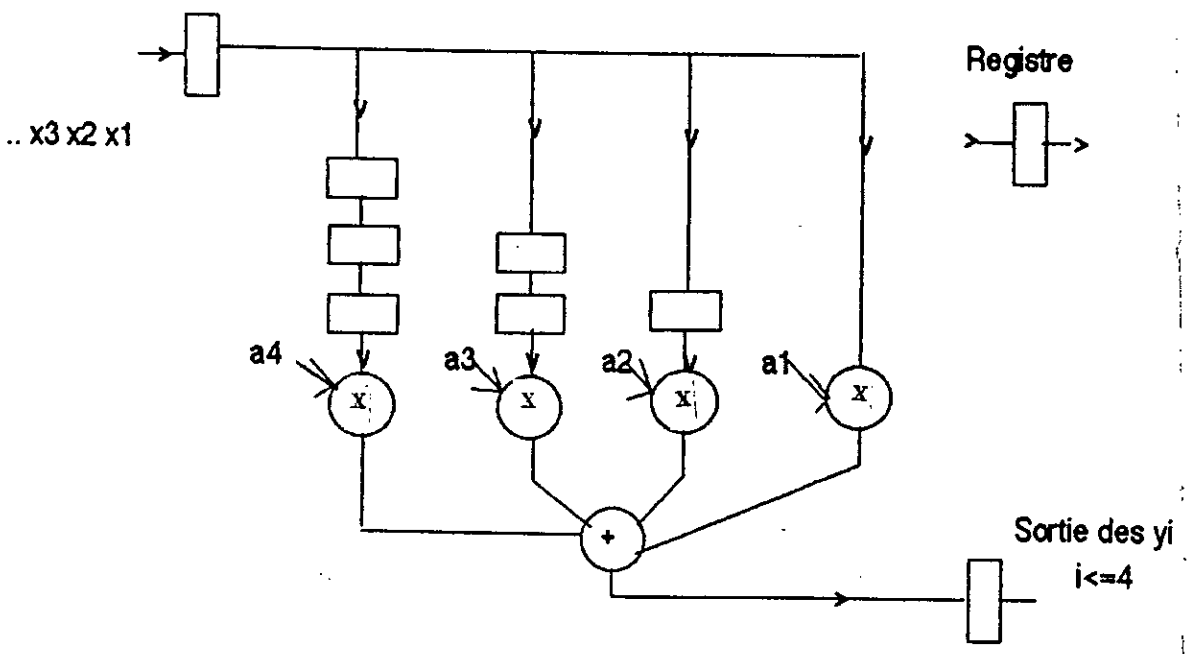


FIG (6) Circuit classique de convolution

-Principe de fonctionnement:

à chaque top d'horloge un nouvel x_i entre dans le circuit, les k multiplications ont lieu en parallèle puis on additionne les k resultats à l'aide d'un additionneurs à k entrées . Notons qu'un nouvel yi est calculé tout les temps de cycle P_k,défini comme la plus petite période d'horloge permettant la réalisation d'une multiplication et de k-1 additions.

La deuxième solution est la solution systolique illustrée par la figure (7) (pour k=4).

-principe de fonctionnement:

Le réseau, composé de k cellules identiques, fonctionne de manière totalement synchrone au rythme d'une horloge globale qui délivre des tops à l'ensemble des cellules.Chaque cellule est capable de réaliser une multiplication suivie d'une addition, d'ou le nom de la cellule MAC (Multiplication-Accumlation).

Il y'a deux flots de données dans le réseau : les x_i circlulentde gauche à droite, gagnant une nouvelle cellule à chaque top, et sans être modifiés durant

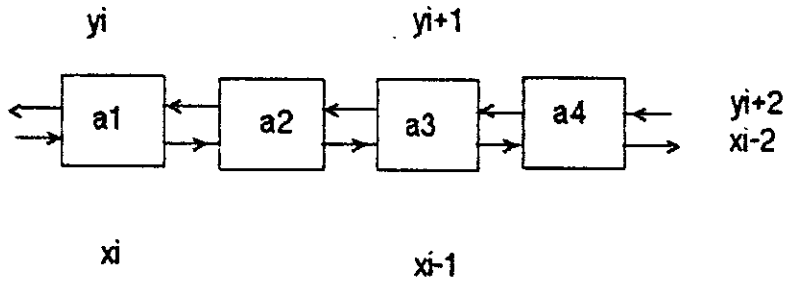
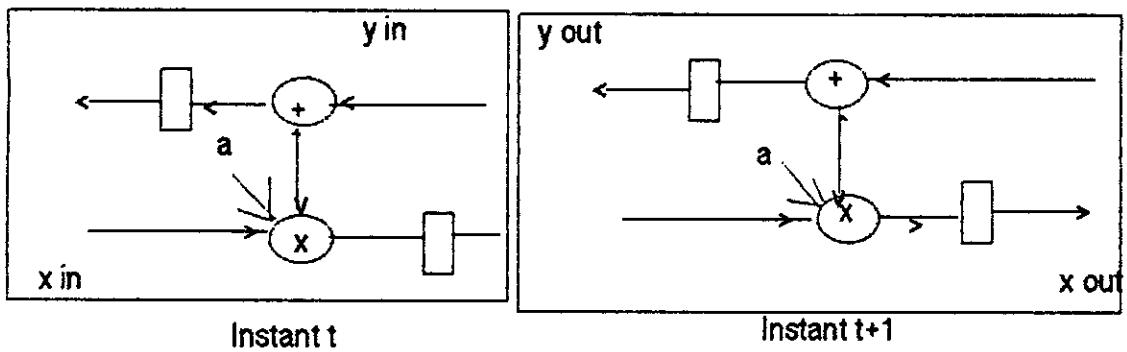


FIG (7) Réseau de convolution non récursive



x out = x in
y out = y in + a * x in

FIG (8) Une cellule du réseau

leur parcours. L'ordinateur hôte délivre un nouvel x_i tous les deux tops. Les yi circulent dans le sens inverse,à la même vitesse et se calculent par accumulations succéssive.Initialement leur valeur est 0 ,et quand ils sont délivrés en sortie du réseau, ils ont accumulé leur valeur définitive.

III.2 MULTIPLICATION MATRICE-VECTEUR :

Pour calculer le produit de matrice-vecteur Y= AX ou A=(aij) est une matrice carrée d'ordre n et Y,X sont des vecteurs à n composantes, on peut utiliser les formules récurrentes suivantes :

$$Y_i(0) = 0$$

$$Y_i(k) = Y_i(k-1) + A_{ik} X_k$$

$$Y_i = Y_i(n)$$

On voit bien que les calculs effectués par cette récurrence sont de simples multiplications suivies d'addition. pour cela on utilise un résau de cellules MAC,dont le fonctionnement est décrit par la figure(9).

La seule différence avec la convolution non récurssive est que le registre interne de chaque cellule MAC doit maintenant être changé dynamiquement au cours du temps d'ou la nécessité d'adjoindre un port d'entrée/sortie vertical pour recevoir les coefficients de la matrices A Pour n=4, La figure (10) illutre comment est éffectué ce produit.

- Les coefficients de la matrice A entrent dans le réseau par diagonales,un nouvel élément tous les deux tops.

- Les composantes du vecteur X circulent inchangées de droite à gauche.

- Les composantes du vecteur Y circulent en sens inverse des Xi leur valeur initiale est 0 et passant à travers le réseau, chaque Yi accumule ses produits partiels

$$a_{i1}X_1, a_{i2}X_2 \ldots a_{in}X_n.$$

Le réseau de Kung et Leirson comprend 2n-1 cellules.Aprés un delai de 2n-1 tops, les yi commencent à émerger à la droite du réseau.Puis 2n-1 pulsations plutard, la derniéres composantes Yn quitte le réseau. Il faut donc un total de 4n-3 cycles pour réaliser le produit matrice-vecteur.

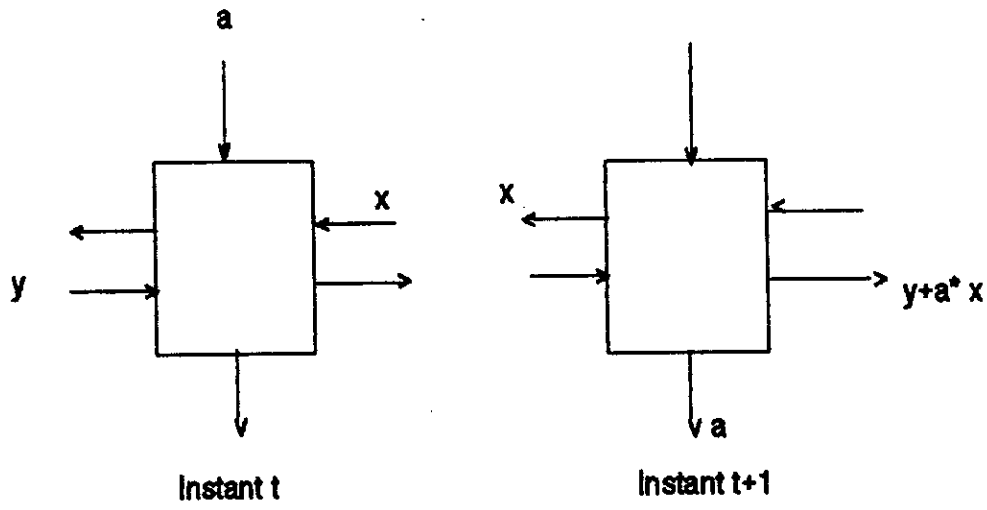


FIG (9) La cellule MAC pour le produit matrice-vecteur

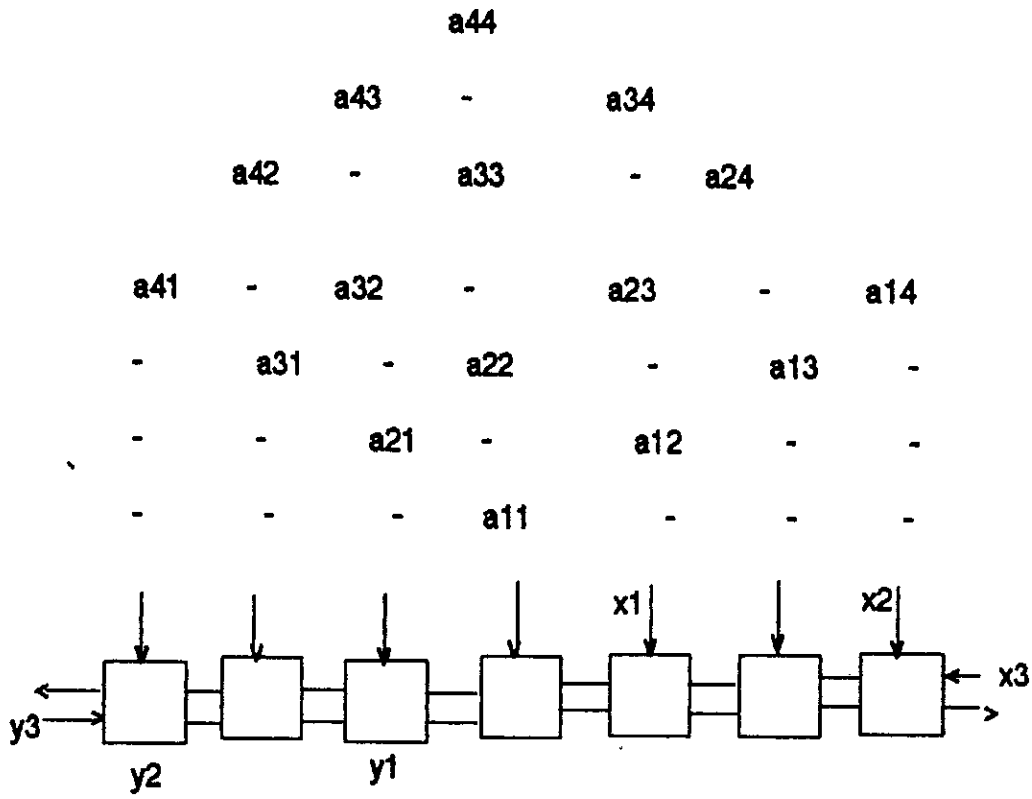


FIG (10) Réseau de Multiplication matrice Vecteur

La figure (11) décrit les pulsations de 2 à 7 du produit pour n=3. Aucun calcul n'est effectué pendant les deux premiers et les deux derniers tops , seuls les contraintes d'entrée/sortie rendent ceci nécessaire.[3] Il est à noter qu'en régime permanent,chaque cellule du réseau n'est activée qu'une pulsation sur deux , le fait que la cellule travaille une fois sur deux n'implique pas que l'efficacité est e=1/2 .le nombre de pas en séquentiel est n et nous obtenons :

$$e = \frac{T_{seq}}{pT_p} = \frac{n^2}{(2n-1)(4n-3)}$$

Ou :

T_{seq} : temps nécessaire pour effectuer l'opération en séquentiel.

T_p : temps nécessaire pour effetuer l'opération en paralléle.

P : nombre de cellules nécessaire pour éffectuer l'opération.

Soit e = 1/8 pour n grand.Ceci est dû à l'initialisation du réseau ou aucun calcul n'est effectué pendant les n-1 premiers pas,pendant que X_1 et

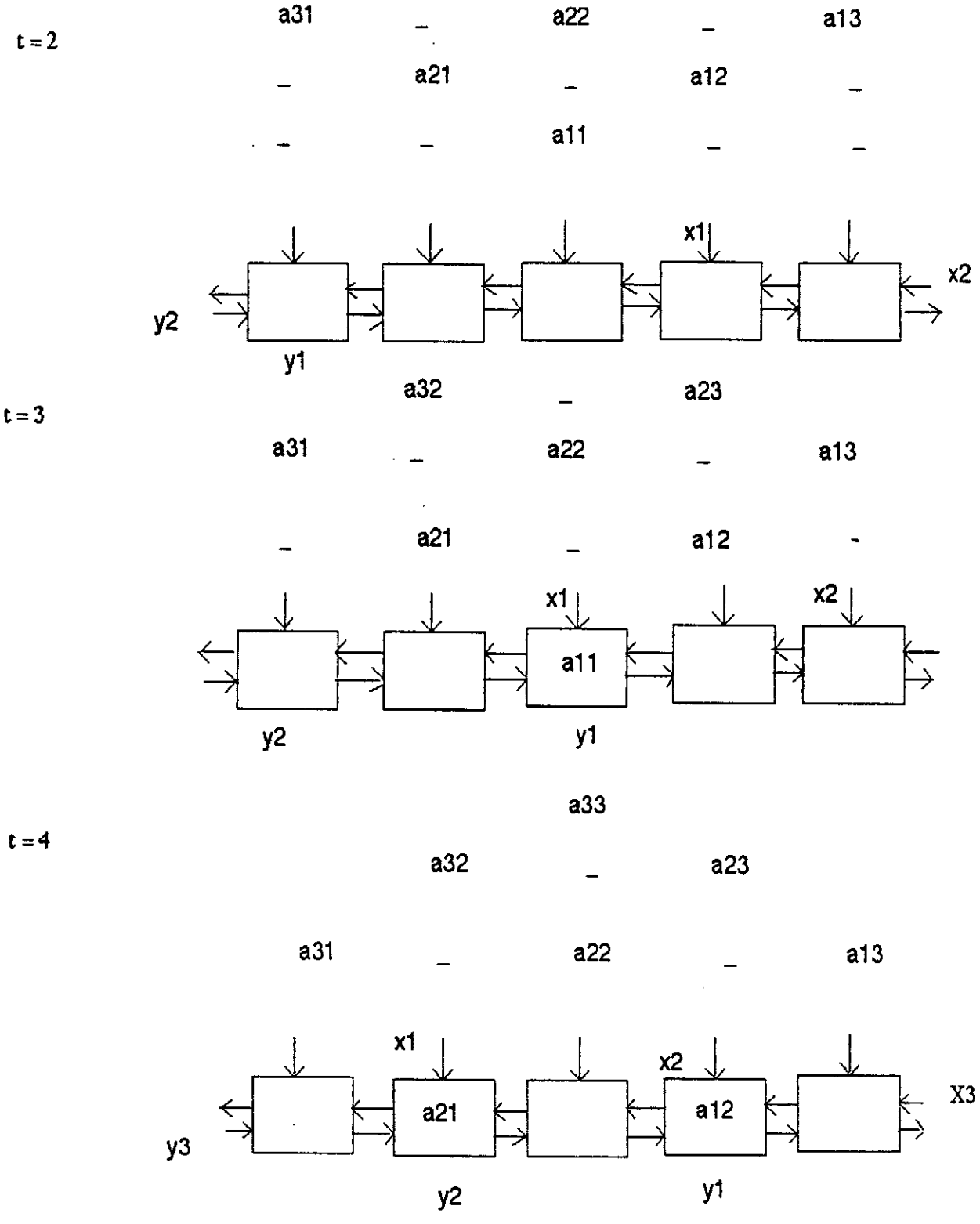
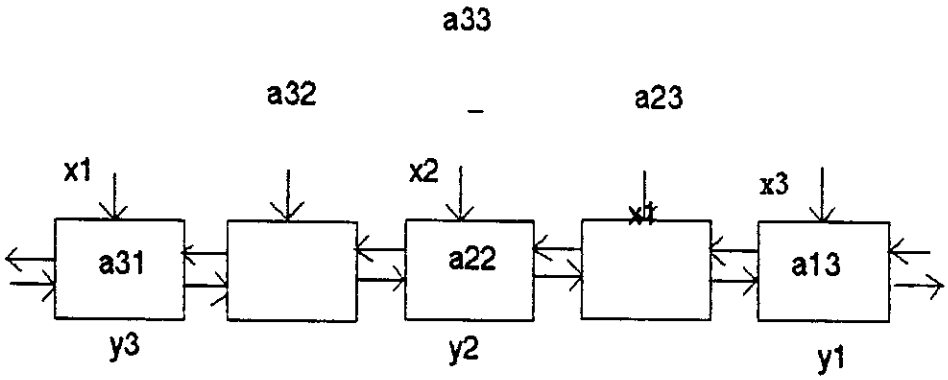
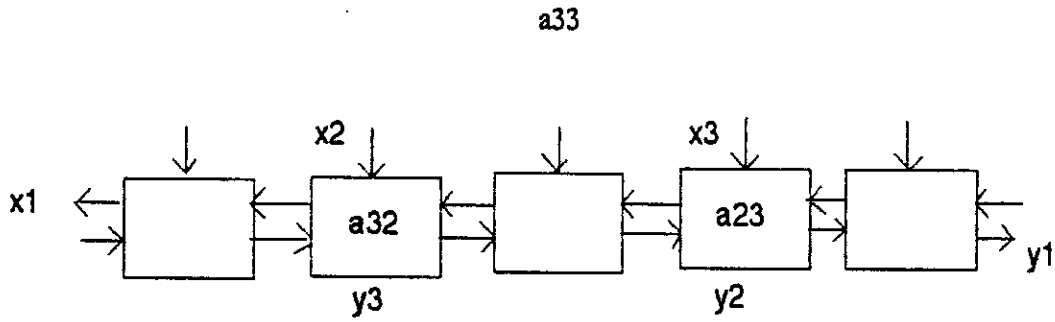


FIG (11 a) Multiplication matrice-vecteur : Y = AX

t = 5



t = 6



t = 7

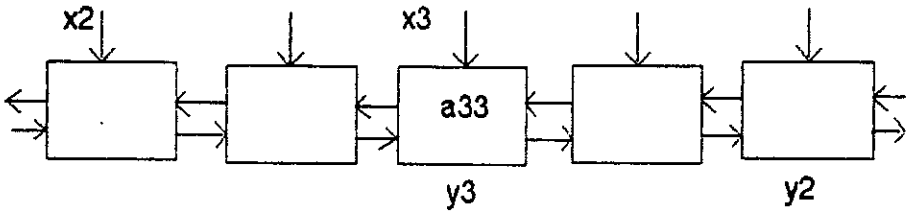


FIG (11 b) Multiplication matrice-vecteur : Y = AX

Y_1 circulent en sens inverse pour se rencontrer dans la cellule centrale.

Il est important de souligner l'impossibilité de lire directement X_1 depuis la structre hôte dans la cellule centrale , afin d'économiser le temps d'initialisation.On suppose que seules les celulues frontières communiquent avec l'hôte grâce à leur ports d'entrée / sortie horizontaux.Les cellules internes communiquent avec l'hôte que via leur port vertical.Elles reçoivent les X_i et les Y_i de leurs voisines c'est pourquoi la cellule centrale doit attendre que les cellules extérieures lui transmettent X_1 et Y_1 avant de pouvoir commencer son calcul.

III.3 SYSTEME LINEAIRE TRIANGULAIRE :

C'est le problème inverse du précedent: soit une matrice A triangulaire inférieure d'ordre n et un vecteur B à n composante on veut trouver la solution X du système AX = B. Le vecteur solution $X=(X_1,X_2,.....X_n)$ de ce système peut être calculer à l'aide des formules récurrentes suivantes :

$$X_i^{(0)} = 0$$

$$X_i^{(k)} = X_i^{(k-1)} + A_{ik} X_k$$

$$X_i = (B_i - X_i^{(i-1)} / A_{ii}$$

Avec : $1<=K<=i-1$

La similarité entre les deux formes récurrentes nous méne à envisager un réseau linéaire identique à celui utilisé dans le produit matrice-vecteur. Soit à résoudre pour n = 4, le sytème AX = B .Le fonctionnement de la cellule ronde est donné par la figure (12),elle est utilsée dans le réseau qui permet la résolution du système.

La solution de ce systèmes est illustrée par la figure(13) On remarque que dans le réseau il y' a deux types de cellules : les cellules carrées sont des processeurs MAC et les cellules rondes celles qui calculent la valeur finale des composantes du vecteur solution X. Quand un élément X initialisé à 0 circule vers la droite dans le réseau,il accumule ses produits partiels

$$a_{i1} X_1 + a_{i2} X_2 + .. + a_{i,i-1} X_{i-1} \quad .$$

Dans la cellule ronde,sa valeur finale est calculée par:

$$X_i := (b_i - X_i)/a_{ii}$$

ensuite X_i est renvoyé dans le réseau et circule vers la gauche sans être modifié,afin de permettre aux composantes suivantes $X_{i,j>=i}$ d'accumuler leur produit partiel $a_{ji}x_i$.

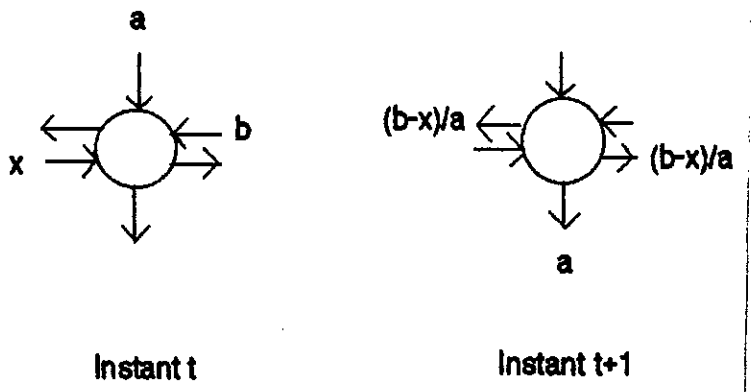


FIG (12) Fonctionnement de la cellule ronde

			a44
		a43	-
	a42	-	a33
a41	-	a32	-
-	a31	-	a22
-	-	a21	-
-	-	-	a11

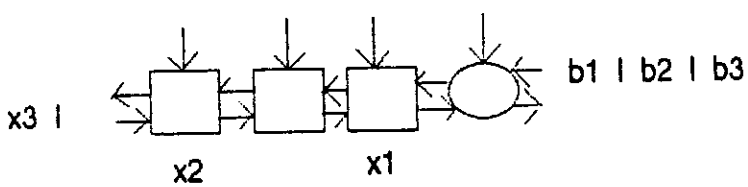


FIG (13) Réseau de solution d'un systmèmetriangulaire:
Ax = b

soit à un instant donné à t=1 Quand la cellule ronde calcule le quotient $X_1 = b_1/a_1$.Il n'y a pas ici à attendre que X_1 soit transmis depuis l'extrémité gauche du réseau,puisque sa valeur initiale est 0.

Un nouvel X_i est délivré par la cellule ronde tous les temps de cycle,ce qui conduit à un temps de calcul égal à 2n-1 tops.[9] [10]

On a :

$$T = n^2/2+0(n)$$

l'efficacité obtenue est $e = \frac{T_{seq}}{n.2n}$

e = 1/4 pour n grand.

III.4 MULTIPLICATION DE DEUX MATRICES DENSES :

a) réseau rectangulaire :

Soient A,B deux matrices carrées denses d'ordre n. on veut calculer le produit C de ces deux matrices . Le produit de deux matrices d'ordre n n'est autre que le résultat de n^2 produits scalaires à n termes,donc pour calculer un élément de C soit C11,on doit réaliser le

produit scalaire de la première ligne de A et la première colonne de B. soit à calculer pour n = 3

$$C = A.B$$

Pour réaliser ce produit on prend un réseau de trois cellules MAC (figures 14), juxtaposons deux autres réseaux identiques à la droite du premier.On commence par calculer le produit dela première ligne de A par les deuxième et troisème colonnes de B on obtient ainsi les éléments c_{12} et c_{13} . En suite on envoyant les autres lignes de A à la suite de la première, on peut calculer les éléments c_{22} ,c_{23} et c_{33} . Pour obtenir la partie triangulaire inférieur de C, il faut donc cette fois-ci juxtaposer à gauche du réseau précedent,deux copies supplémentaire. Ainsi on obtiendra le réseau complé pour le produit de deux matrices,illustré par la figure (15).Il est à noter que comme dans le produit matrice-vecteur,un espace doit être inséré entre deux élément consécutifs des matrices A et B et que les éléments de la matrices C sont tous initialisés à 0. Dans le cas général d'un réseau de (2n-1).n cellues permet de calculer le produit de deux matrices de taille n en temps 4n-3. Le temps d'exécution

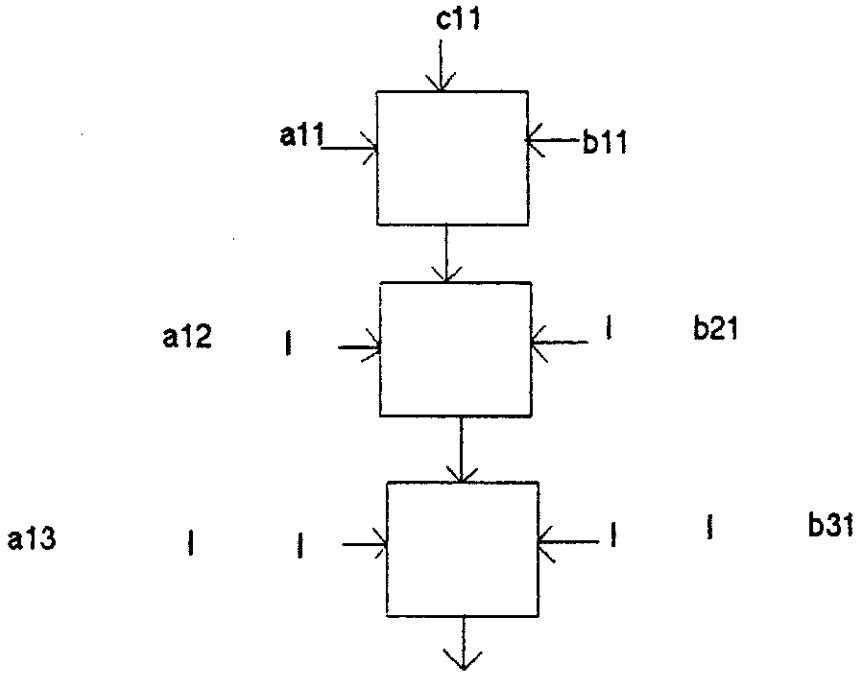
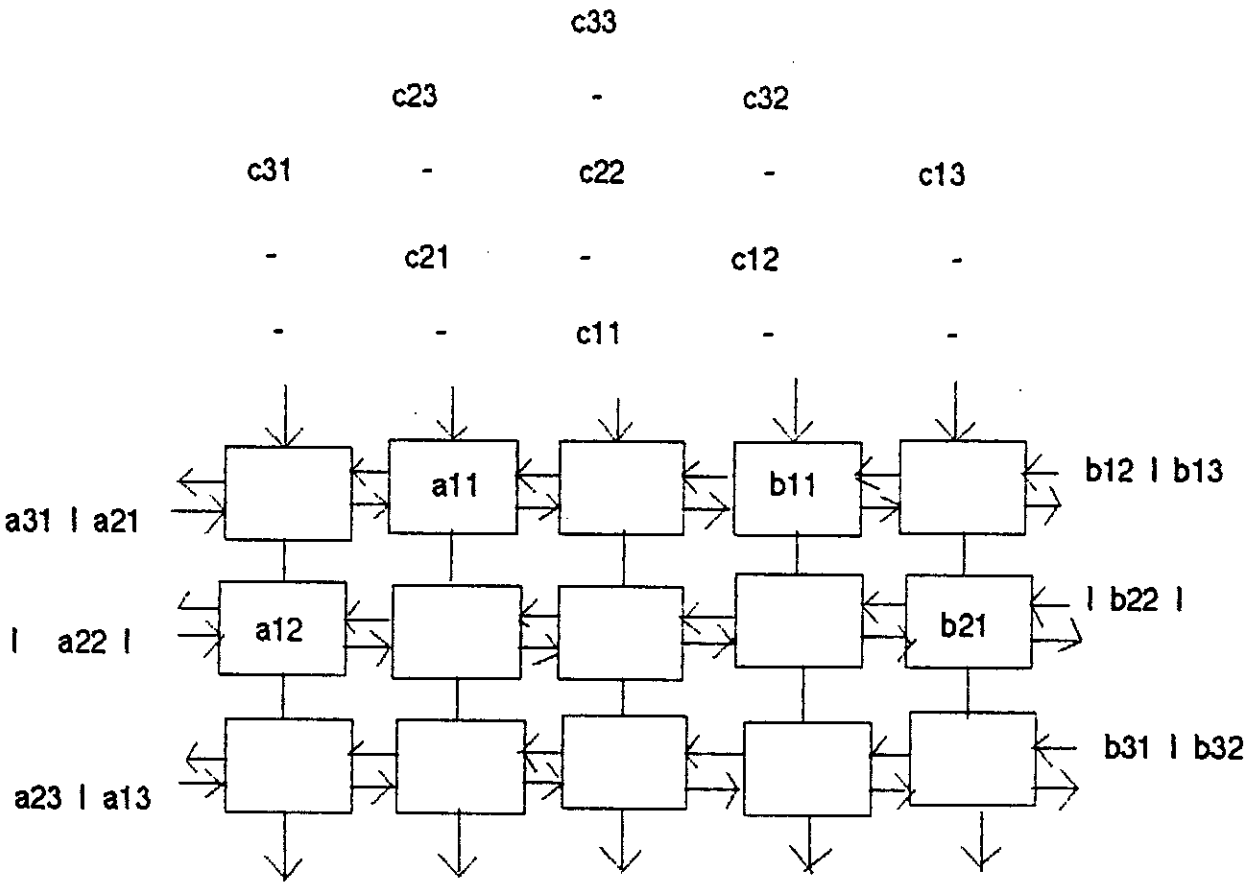


FIG (14) Calcul d'un produit scalaire



(FIG 15) Produit de deux matrices denses (3*3)

est calculé comme suit : le premier élément c_{11} entre dans le réseau au top n ,le dernier élément cnn arrive 2(n-1) tops plus tard,et n-1 tops sont encore nécéssaire pour compléter son calcul.A noter qu'au top 4n-3,certains coefficients des matrices A et B circulent encore dans le réseau,mais n'effectuent plus de rencontres et que rien ne se passe durant les n premiers tops ou les a et b circulent à travers le réseau pour se rencontrer dans la cellule centrale.[9]

L'efficacité est dans ce cas donnée par :

$$e = \frac{n^3}{2\,n^2.4\,n} \quad ; \quad e = 1/8 \text{ pour n grand.}$$

b) réseau carré :

On peut effectuer le produit de deux matrices A et B de taille n mais cette fois-ci en utilisant un réseau carré dans ce cas on a une cellule (figure (16)) dont le registre interne est modifié par accumulations successives jusqu'à acquérir la valeur finale du produit scalaire.

Pour calculer les c ,l'algorithme est :

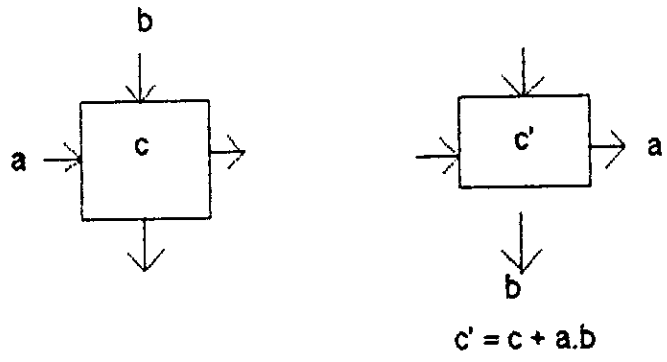


FIG (16) Cellule MAC (deuxiéme version)

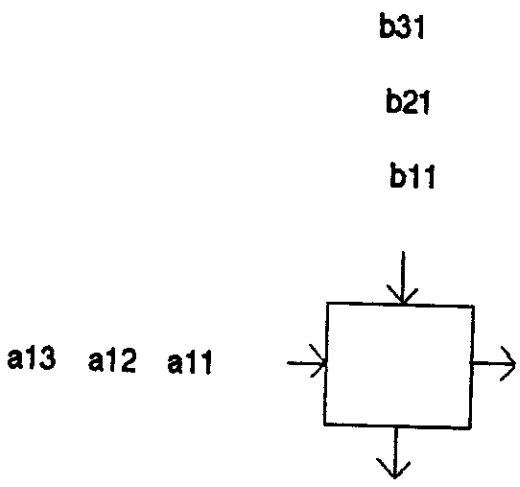


FIG (17) Calcul d'un produit scalaire (deuxième version)

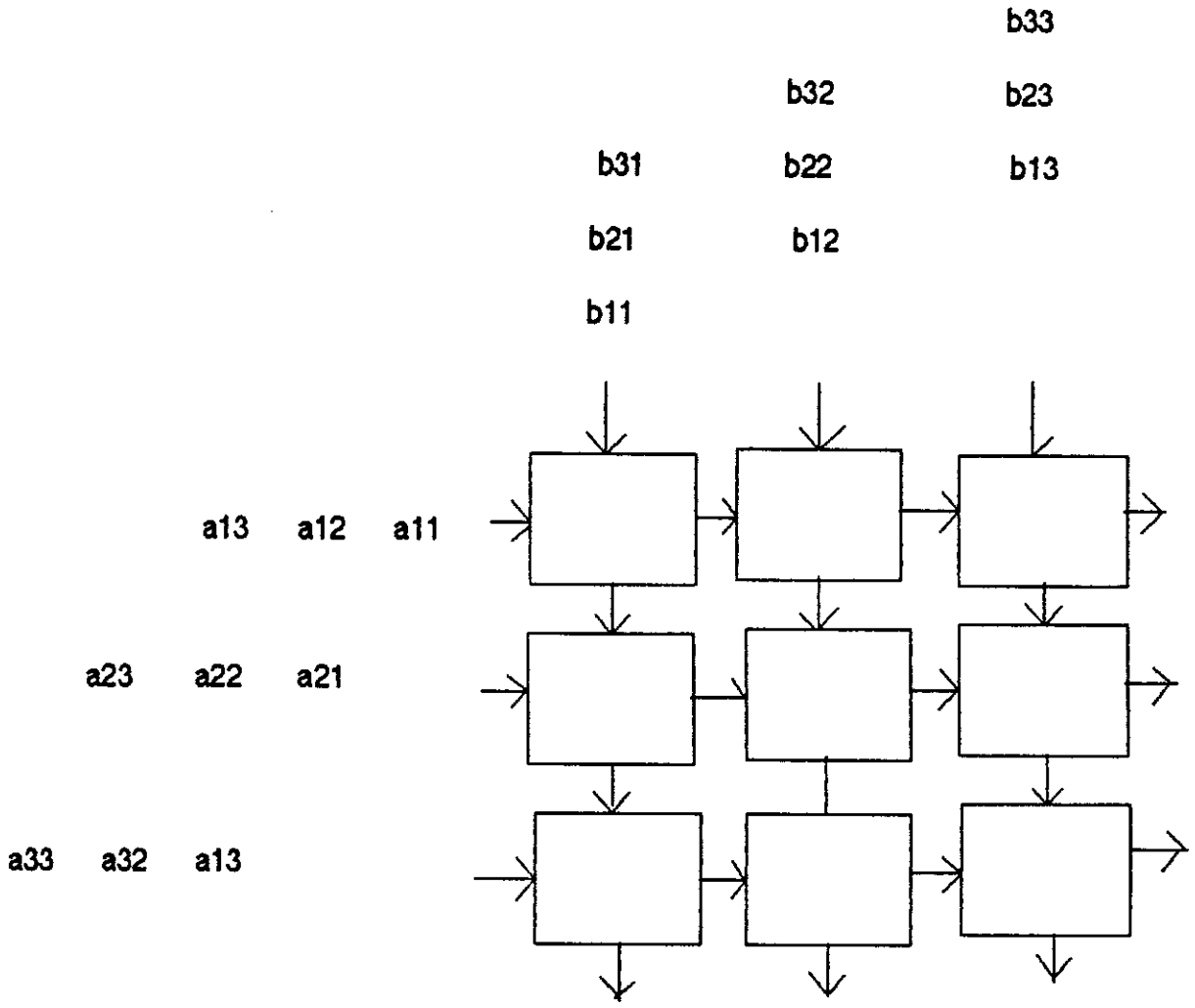


FIG (18) Réseau de produit de deux matrices denses de taille 3*3

Le principe est le suivant :si la dernière accumulation dans une cellule C a lieu à un instant t à l'instant t+1 celle-ci envoie sur la droite le contenu c de son registre interne.Au temps t+1,sa voisine de la droite C' laisse passer inchangée la variable c en la transmettant à nouveau vers la cellule de droite C".Au temps t+2,C' envoie à son tour le contenu de son registre interne à C". Le booléen contrôlant le vidage se propage donc à la vitesse 1/2, gagnant une nouvelle cellule tous les deux tops. On peut donc donner
la description détaillée du fonctionnement d'une cellule lors d'une implémentation concréte. Tout d'abord le contrôle des cellules doit coder les états suivants:

* début:commencer un nouveau calcul

* accum:accumuler un pas de produit scalaire dans le registre

interne

* trans:transmettre inchangé un coéfficient vers la droite

* vider:transmettre le contenu du registre interne vers la droite.

CONCLUSION

L'intéret essentiel des réseaux systoliques demeure dans l'implémentation ceci dit arriver à concevoir des architectures à partir de la description des algorithmes qu'on désire éffectuer ,on a vu que ceci était bien réalisable dans le cas des algorithmes des opérations matriciellestelle que le produit de deux matrices ou on a pu concevoir un réseau effectuant cette opération ou le temps d'éxecution est nettement performent vis-à-vis du réseau séquentiel.

B. DEUXIEME PARTIE : Langage Occam

CHAP I : Programmation Concurrente

CHAP II : Occam

CHAP III : Les Transputers

CHAP IV : Implémentation software des réseaux systoliques

Conclusion

CHAPITRE I :

PROGRAMMATION CONCURRENTE

I.1 INTRODUCTION :

L'idée de la programmation concurrente vient du fait que notre environnement est essentiellement concurrent : des actions particulières se déroulent simultanément en divers endroits avec ou sans interactions entre elles.Ce qui a pousser les chercheurs à intégrer cette idée surtout basée sur la notion de simulatneïté,de synchronisation et la competition entre tâches. La programmation concurrente est trés difficile à réaliser sur des systèmes conventionnels par nature de type séquentiel.Ce qui a alors conduit à la conception des systèmes spécialisés permettant de programmer concurremment.

Cette technique de programmation entraine des gains en performences inéstimables.Le traitement paralléle avec des processeurs convotionnels conduit automatiquement à un accroissement des performances proportionnel au nombre de processeurs utilisés.

Les programmes classiques ne sont pas du tout adaptés pour la programmation concurrente Pour cela,certains langages ont été modifiés et d'autre crés spécialement dans ce but l' Occam est l'un d'eux.

I.2 LE PROGRAMME CONCURRENT :

Un programme concurrent est par définition un programme non séquentiel;donc,pour l'exécuter, des opérations conduisant à sa réalisation sont effectuées en paralléle, simultanément plutôt que séquentiellement une opération aprés l'autre. L'avantage que présentel'exécution concurrente est bien évident:si l'on dispose d'autant de prcesseurs qu'il y a d'opération à éffectuer.Le temps total nécessaire à l'exécution est diminué.Comme le même programme peut être exécuté avec un seul processeur,dans ce cas,une fraction du temps processeur est périodiquement allouée à chaque opération.

CHAPITRE II :

OCCAM

II.1 LE LANGAGE OCCAM :

L'occam est un langage paralléle,il permet avec la plus grande simplicité la communication entre différents processus.La philosophie de ce langage est dans la fameuse devise : "il ne faut pas multiplier les entités au delà de ce qui est nécessaire" énoncée par Guillaume d'Ockham (1270-1349).Ce philosophe est aujourd'hui reconnu comme un précurseur de la méthode scientifique moderne.[1]

De la même façon le langage Occam,qui tire son nom de celui du philosophe,est fondé sur un petit nombre de prémitives de base,parmi lesquelles se trouve le traitement paralléle de tâches indépendantes et la communication entre ces tâches. L'occam autant que langage de programmation concurrente a été conçu en 1982 par May pour programmer à haut niveau des applications communiquant et s'executant en paralléle,le succés d'occam est indissociable du transputer le circuit d'Inmos dont il est le langage de programation.Prévu pour des applications synchrones,occam peut être utilisé pour décrire et simuler des architectures systoliques.

II.2 LA STRUCTURE DE L'OCCAM :

En occam on trouve deux principes essentiels et qui sont:

-Les processus

-Les canaux

1°) Les processus :

Les trois processus primitifs sont :

entrée,sortie,assignation.

-Entrée:

EST ! message signifie que la valeur de la variable

message est emise sur le canal EST.

-Sortie:

ouest ? message

signifie qu'une valeur est reçu par le cannal ouest et stockée

sous le nom de message.

-Assignation :classique de la forme

variable := expression.

A partir de ces trois processus on peut construire tous les autres,ils sont combinés à l'aide des constructeurs du langage au nombre de six:

SEQ (sequentiel)

PAR (parallel)

ALT (alternative)

IF

WHILE

FOR

On va voir des exemples qui illustrent le fonctionnement des différent constructeurs.

Exemple (1):

```
SEQ
  message := TRUE
  OUEST ! message
```

l'exemple (1) montre que le fonctionnement du constructeur SEQ,qui combine des processus pour en former

un nouveau qui excute ses processus premitifs dans l'ordre ou ils sont listés et se termine quand son dernier processus premitif a compléter son exécution.

Exemple (2):

```
PAR
  OUEST ! any
  EST ? message
```

Avec le constructeur PAR l'execution de chacun des processus primitifs se fait de manière sumiltanée.Le mot reservé any est employé quand la valeur réelle envoyée sur le canal n'est pas utilisée afin d'assurer une synchronisation d'état.

Exempe (3):

```
ALT
EST ? message
OUEST ! message
NORD? message
SUD! message
```

Le constructeur ALT choisit de façon non deterministe d'executer l'un des processus primitifs parmi le premier qui sont prés.Plus genéralment les alternatives sont des commandes gardées contenant des expressions booléennes,des processus d'entrée et des conditions temporelles.

Exemple (4):

```
IF
 trouve
      OUEST ! 'ok'
TRUE
      OUEST ! 'erreur'
```

Le processus teste si la variable trouve est vraie si oui, il délivre le message 'ok' sinon le message 'erreur'

Exemple (5):

```
WHILE TRUE
  j:=j+1
```

La variable j est incrémentée pour l'éternité puisque le mot reservé TRUE est toujour évalué comme vrai.

2°) Les canaux :

Dans le langage occam les cannaux sont utilisés pour communiquer des valeurs entre deux processeurs, et pour synchroniser leur activité .Quand un processus émet une valeur sur un canal,il doit attendre que le processus d'entrée soit prêt, de même un processus d'entrée doit attendre que le processus de sortie soit prêt quand il accepte une valeur sur un canal en d'autres termes les communications se font par rendez-vous.[5] Un canal Occam est similaire à une variable, mais plutôt que d'y stocker une valeur par affectation, on y écrit et y lit en communiquant, et pour réaliser une communication bidirectionelle il faut utilises deux canaux.[5]

3°) La syntaxe de l'Occam :

La syntaxe du langage Occam est assez spéciale par rapportà celle des langages convontionnels,on ne va pas trôp s'étaller ici sur l'aspet syntaxique d'un programme Occam, mais on citera quelques unes de ces particlarités. Dans un programme occam les déclarations des variables et des canaux doivent toujours se faire en tête

du programme. Tous les mot clés doivent être écris en magiscule sinon ils ne seront pas reconnus par le logiciel. Aprés chaque mot clés on doit laisser deux blanc pour écrire la nouvelle instruction c'estla conception du logiciel qui veut cela.

Il est à noter que occam permet la notion de procedure et fonction.

4°)La communication entre processus :

Un processus débute et exécute un nombre donné d'opérations puis termine ou s'arrête.Chaque opération peut être une affectation,une entrée ou une sortie.Cette définition, correspond bien a celle d'un programme séquentiel,mais il ne faut pas oublier qu'en occam plusieurs processus peuvent s'exécuter simultanément et qu'ils peuvent échanger des messages a tout instant.Entre son début et sa fin,un processus peut être prêt à communiquer au moyen d'un ou plusieurs canaux.Chaque canal constitue une connection unidirectionnelle entre deux processus concurrents,un des processus pouvant uniquement lire et l'autre seulement écrire.Donc occam permêt la communication entre

processus c'est là ou demeure la caractérestique de ce langage qui ne demande aucune instruction particulière quel que s oit le type de système de communication matériel utilisé.
De même pour Occam il n'est pas important de savoir si les processus en communication s'exécutent sur le même processeur ou sur deux processurs distincts .

La communication entre processus est la cractérestique principale du langage Occam, pour réaliser une communication il nous faut :

-Deux processus s'exécutant en paralléle

- un canal joignant ces deux processus

cette communication peut être ilustrée par un simple programme tel que :

```
CHAN comm:
VAR x,y:
PAR
  comm ! y          (a)
  comm ? x          (b)
```

Le programme communique la valeur y dans le processus (a)à la variable x du processus (b) à travers le canal comm reliant ces deux processus,la communication entre les deux processus est faite grâce à la construction "PAR" et ne doit se faire que par l'intermédiaire de canaux, car Occam n'autorise pas le passage de valeurs entre processus concurrent au moyen desvariables partagées.De plus, si un composant d'une structure "PAR" réalise une affectation ou une entrée avec une variable donnée,alors celle-ci ne doit pas être utilisée du tout dans un autre composant dela structure,les variables doivent restées locales aux composants; ceux-ci échangeant des valeurs par des canaux.Cette restriction est dûe du fait de la concurrence entre processus.Les processus concurrents sont en effet exécutés simultanément et d'une manière asynchrone les uns par rapport aux autres.Ils ne se synchronisent que lorsqu'une communication doit se faire par canal à partir toutes ces données on peut résumer les différentes régles principales d'Occam et qui sont :

-Les variables sont utilisées pour stocker des valeurs à l'intérieur d'un processus.

-Seulement deux processus d'une construction "PAR" doivent utilisés un canal donné l'un comme émetteur,l'autre comme récepteur.

-Le processus émetteur exécute uniquement des opérations de sortie vers le canal commun,tandis que le processus récepteur ne fait que lire ce même canal.

Remarque:

L'ordre des entrées et des sorties dans chaque processus impliqué dans une communication est primordial, un mauvais séquencement conduit automatiquement au blocage de l'application (deadlock),donc les échanges entre processus doivent être soigneusement analysés et implémentés.

Exemple:

```
CHAN C1,C2:
VAR x,y:
PAR
SEQ
  C1 ? x    (1)
  C2 ! 1    (2)
SEQ
  C1 ? y
  C2 ! 2
```

Dans cet exemple la construction "PAR" ne se temine jamais,chaque compsante attent indéfiniment une information du processus concorrent par un canal ce qui conduit au blocage de cette application,on peut tout de même éviter le blocage dans ce cas en permuttant par exemple les instructions (1) et (2) du processus (a).
Donc pour éviter ce genre de problème il faut toujours séquencer son programme de manière à s'assurer que deux processus distincts n'attendent jamais chacun une sotie de l'autre,séquentiellement prévue plus tard dans chaque processus. Il est à noter que ces problèmes de blocage ne sont pas indiqués par le logiciel lors de la

compilation,mais ils apparaissent à l'exécution du programme.

II.3 LE LOGICIEL UTILISE :

Le logiciel dont on dispose au niveau du laboratoire est Occam1 exécutable sur PC ,c'est la première version du logiciel Occam il présente des manques concernant sa librairie interne pour exécuter une opération quelconque telle que la somme de deux nombres soit à sommer: 5 plus 5 alors le logiciel n'affiche pas 10 mais le caractére du code ascii du 10,en plus le logiciel ne sait pas lire une chaine decaractéres,il ne pouvait lire qu'un seul caractére à partir du clavier. pour cela on a dû dévlopper ce logiciel en écrivant des programmes en Occam (voir progrmme(2) en annexe) qui réglent tous ces problèmes.

Il est à noter qu'il existe l'autre version de L'Occam qui est Occam2 beucoup plus dévlopp ée que celle d'Occam1 qui est exécutable sur transputer, mais pas encore sur un PC (ou PC ayant des cartes transputers).

CHAPITRE III :

LES TRANSPUTERS

III.1 INTRODUCTION :

Historiquement,la réalisation des systémes partait du principe que le coût du traitement est plus élevé que celui des mémoires.Ce qui a conduit à réaliser des systèmes monoprocesseurs connectés à d'énormes mémoires,aboutissant au goulet d'étranglement intrinséque à la nature séquentielle des processeurs utilisés.Les progrés de la technologie sont tels qu'à l'heure actuelle cette tendance peut être inversée.Cette potentialité a été exploitée par INMOS et a conduit à la mise sur marché du TRANSPUTER.Ce microprocesseur peut directement être utilisé pour concevoir des systèmes hautement concurent.il a par ailleurs été conçu de manière à être l'élément de base de systèmes complexes,à l'analogue du transistor dans les réalisations électroniques analogique.Ce que traduit sa dénomination.C'est un comPUTER (ordinateur) monoboitier et un composant silicium comme un TRANSistor.[6]

III.2 DEFINITION :

Le transputer est un microprocesseur à 16 ou 32 bits interconnectable à quatre autres transputers au

moyen de canaux de communication série unidirectionnels intégrés appelés liens.Il peut par ailleurs accéder à des ressources propres ou partagées à l'aide d'un bus parallèle configurable conventionnel.

Le transputer peut être utilisé comme un microprocesseur conventionnel de haute performance,mais il est particulièrement destiné à la réalisation de réseaux de processeurs de topologies variées.La conception du transputer est le résultat d'une collaboration étroite entre les concepteurs du circuit et les créateurs du langage occam. Le transputer intégre ainmatériellement des caractérestiques importantes d'occam,en particulier le transputer est une réalisation matérielle de l'objet de base de ce langage, à savoir le processus.

Un processus est une tâche élémentaire et autonome, avec ses propres données et sont propre code, qui peut communiquer avec d'autres processus s'exécutant en même temps. L'avantage de ces nouveaux composants ou des réseaux de processeurs est d'autoriser l'obtention de performances notablement accrues et permettre la réalisation d'applications jusqu'ici difficilement imaginable.

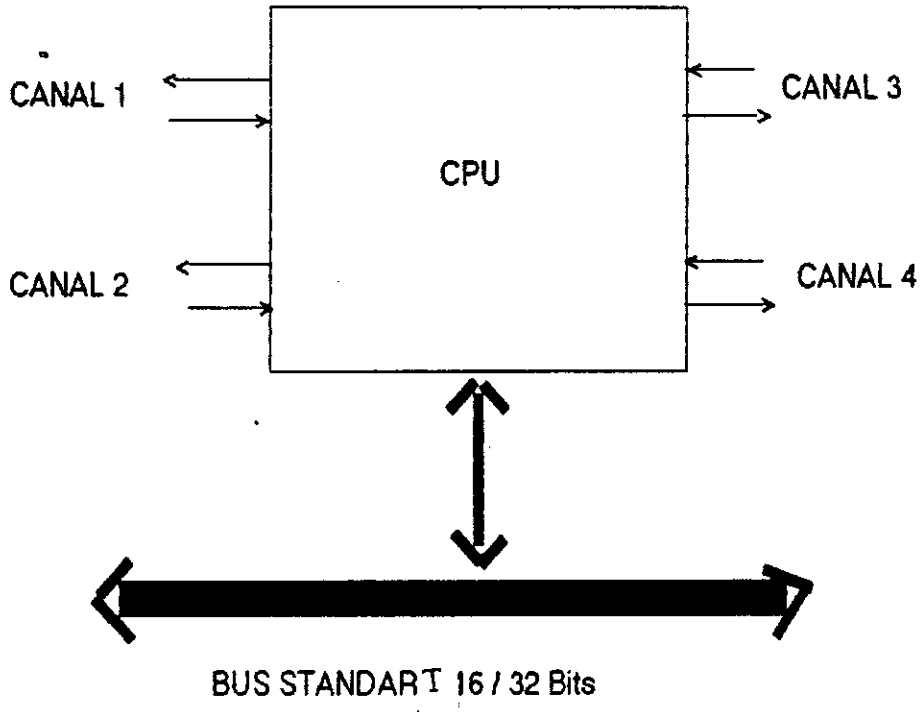


FIG (19) Représentation Fonctionnelle du Transputer

Le transputers est un composant VLSI programmable, il permet d'avoir des performances élevées en plus une adaptabilité extrême aux besion des utilisateurs, son architecture interne révolutionnaire autorise l'implémentation de la concurrence dans un système.

III.3 STRUCTURE INTERNE DU TRANSPUTER :

Le transputer est un microprocesseur disposant d'une mémoire locale et de liens pour connecter un circuit à un autre L'architecture "transputer" définit en une famille de composant VLSI programmables.

Un produit transputer est un circuit qui contient :

* Un processeur
* Une mémoire
* Des liens de communication pour les liaisons point à point entre transputer
* Une interface mémoire programmable permettant de l'adapter à un usage particulier Il est pourvu en plus d'une unité arithmétique travaillant en virgule

flottante. A noter qu'un transputer peut être utilisé seul ou en réseau pour construie des systèmes concurrents de haut niveau.(FIG 20a),(20b)..

Le réseau est alors obtenu aisément en utilisant les liens de communication point à point.Chaque transputer disposant de plusieurs de ces liens,peut être relié à un autre lien ou à un autre composant aprés adaptation,ce qui permet de construire des réseaux quelconques de processeurs de taille est de topologie arbitraitre.[6]

Avantage d'utilisation des liens point à point :

l'avantage des liens point à point par rapport aux bus multiprocesseurs est multiple :

-L'utilisation n'a pas à se préocuper du mécanisme de communication, celui-ci étant intégré au niveau du composant.

-La charge capacitive de chaque lien ne devient pas pénalisante lorsqu'on fait croître la taille d'un réseau.

-La bande passante des transferts n'est pas saturée lorsque le nombre de transputers augumente : au contraire plus ce nombre est grand,plus la capacité de transfert globale est grande.

L'implémentation physique des liens est telle que leur utilisation devient évidente.la communication proprement dite est du type série avec une vitesse standard de 10 Mbits/s.Cette vitesse est cependant programmable est peut aller de 5 Mbits/s à 20 Mbits/s selon le type de circuit.Elle ne dépend pas par contre des performences du processeurs; La communication doit cependant respecter le protocole décrit ci-dessous (Fig 21) Toute donnée échangée suppose donc un dialogue entre les deux éléments concernés,ce qui contribue à la fiabilité de la communication.Ce dialogue permet aussi d'assurer la synchroninsation entre les deux circuits. La synchronisation est réalisée grâce à l'accusé de réception du message,envoyé par le destinataire vers l'émetteur de l'information.Un lien doit donc être constitué par une liaison électrique dans chaque direction,ce lien peut être établi matériellement en reliant un interface lien d'un transputer à un interface du même type d'un autre circuit par l'intermédiaire de deux lignes unidirctionnelles utilisées en mode série. Dans ce protocole chaque ligne transmet des données et des informations de contrôle.Chaque message est transmis comme une séquence de transfert d'octets,ce qui nécessite seulement un tampon de un octet dans le transputer

récepteur pour assurer qu'aucune information n'est perdue. Aprés émission complète d'un octet,l'émetteur attend la réception d'un accusé de réception avant de continuer sa tâche, donc l'accusé de réception a une double signifiation :

-Le récepteur a été capable de recevoir l'octet.

-Le lien récepteur est prêt à accepter un nouvel octet.

Les données et les accusés sont multiplexés sur chaque lien de transmission.l'accusé de réception peut donc être envoyé dés que la réception d'un octet est engagée.En conséquence, la transmission peut être continue,sans délai entre les octets transférés.

Autre avantage de La communication par lien c'est qu'elle n'est pas sensible à la phase de l'horloge du système; ce qui permet d'assurer un échange d'information par lien entre composant utilisant des horloges distinctes, à condition toutefois que la fréquence de communication soit la même. D'ou les deux possibilités de cadencement des transputers [6] :(FIG 22a,22b).

L'intérêt de ces liens demeure dans le sens ou il permettent l'échange des informations entre des programmes différents s'exécutant en paralléle sur des transputers distincts.C'est à la base d'Occam que sont faites ses échanges pour l'implémentation.

III.4 APPLICATION DES TRANSPUTERS :

Les produits réalisés à partir de transputer sont multiples ils vont de la simple carte d'extension aux machines complètes à base de transputers. La carte d'extension est destinée à être insérées dans un microordinteur classique et va se comporter comme un co-processeur spécialisé vis-à-vis de ce dernier ,elle permet de s'initier au multiprocessing et au langage occam et ceci à partir d' un environnement de programmation classique. Ces extension peuvent être utilisées dans le but d'augmenter les performances du microordinateur. Pour ce qui est des machines à base de transputers il existe plusieurs types pour diverses applications parmi elles des dispositifs destinés aux traitement d'image et la synthèse d'image, d'autre à la simulation pour la conception des circuits intégrés,comme on trouve des applications dans le traitement de signal et récemment on

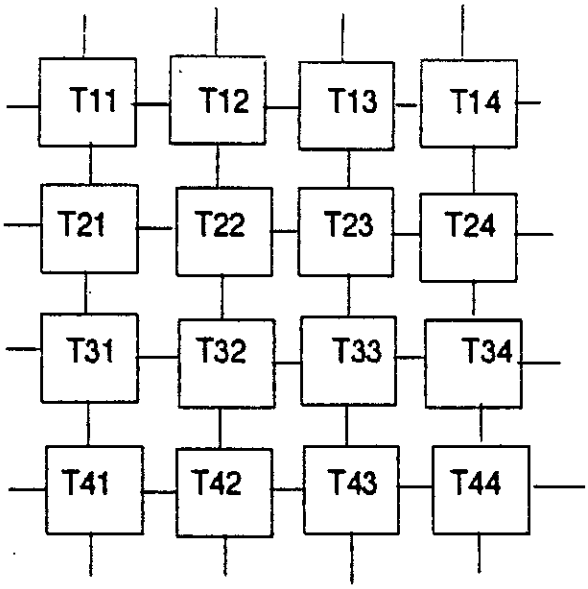


FIG (20a) Systèmeconcurrent à base
16 Transputers

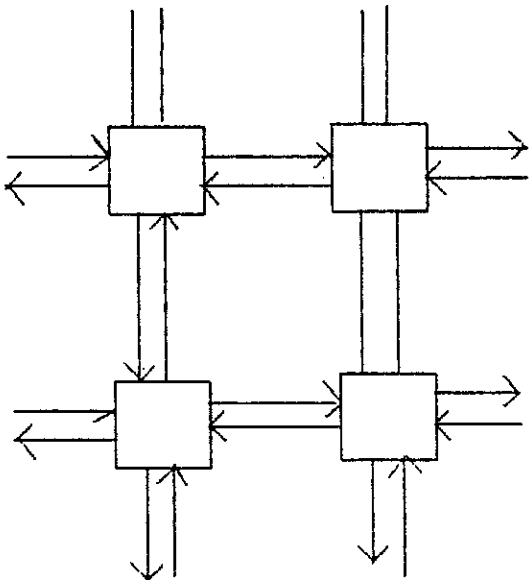


FIG (20b) Un réseau de 4 transputers
constituant un noeud

a vu l'entrée de l'intelligence artificielle dans les appilcations transputers.

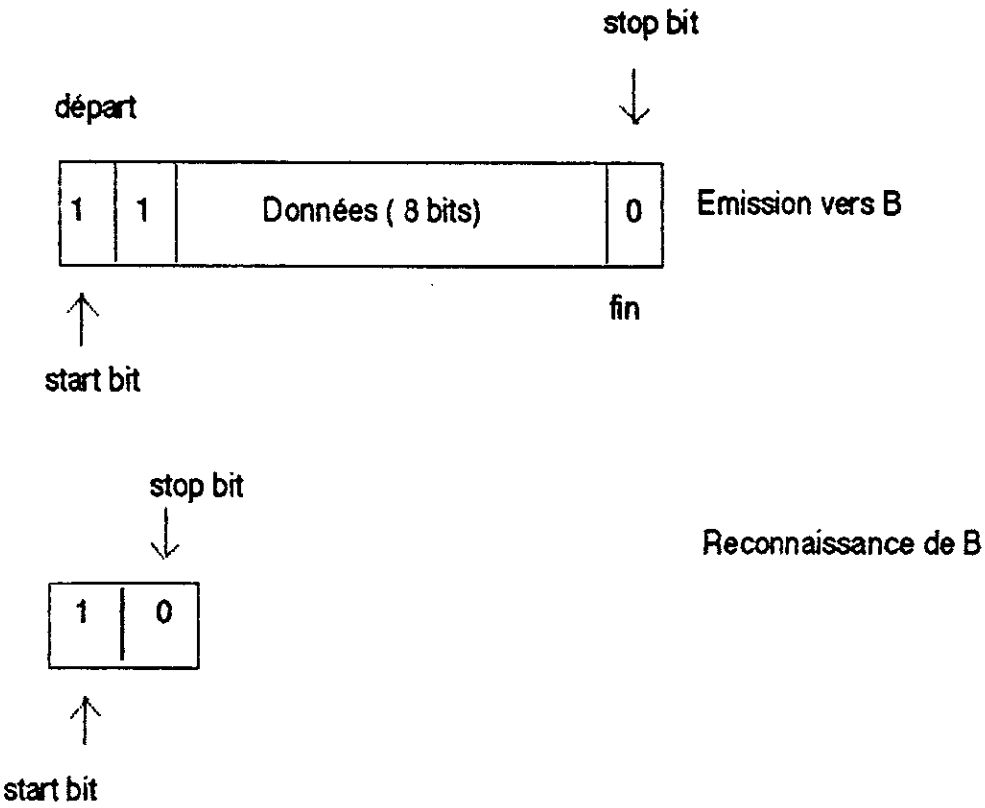

stop bit
départ
1
1
Données (8 bits)
0
Emission vers B
fin
start bit
stop bit
Reconnaissance de B
1
0
start bit

FIG (21)

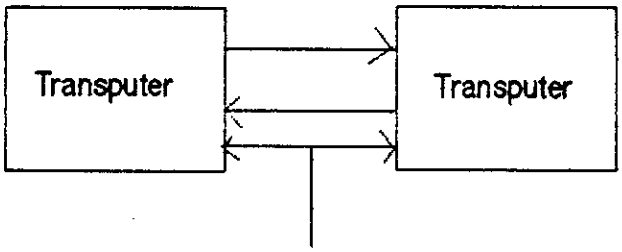
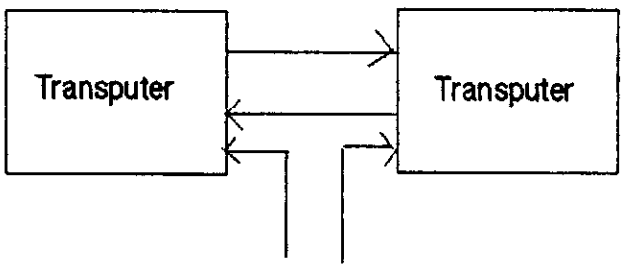


FIG (22a)



FIG(22b)

FIG (22) cadencement des transputers

CHAPITRE IV :

IMPLEMENTATION SOFTWARE DES RESEAUX SYSTOLIQUES

VI.1 INTRODUCTION :

Il existe plusieurs moyens pour simuler les réseaux systoliques tels que le langage SYSIC, qui est spécifiquement conçu pour les réseaux systoliques on trouve aussi le langage ADA qui est un langage concurrent trés performent et enfin le langage Occam qu'on a bien étudier auparavant et qu'on utilise ici pour simuler les réseaux systoliques.

Ce langage qui présente l'intéret d'être largement diffusé, simule les architectures systoliques grâce à sa structure et sa caractérestique de programme concurrent (paralléle).Il permet effectivement de décrire une architecture systolique ,et donc on pourra résoudre grâce au logiciel les problèmes soulevés par la réalisation matériel.[11].

IV.2 PROGRAMMATION DES RESEAUX SYSTOLIQUES :

On a vu dans la remiére partie quelques algorithmes systoliques pour les opérations matricielles.Tous ces algorithmes peuvent étre programmer à partir d'Occam et donc on passe du réseau systolique

exécutant une opération donnée à un programme Occam simulant ce réseau,mais nous nous contentera de donner que quelques exemples de programmation des réseaux systoliques en Occam.

IV.3 EXEMPLES DE PROGRAMME OCCAM :

Différents programmes réalisés sont donnés en annexe.

-Pogramme(1) "producteur consommateur" :

Ce programme est un programme Occam élémentaire,il montre comment se fait la communication entre processus.Ici on a deux processus : le processus producteur,et le processus consommateur sous forme de deux procedures.La communication entre ces deux processus se fait à travers le canal comms. L'activité des deux processus se déroule schématiquement suivant le cycle suivant :

producteur	consommateur
PROD : Produire un message	CONS : Prelever un message dans le tampon
Deposer un message dans le tampon	Consomer le message
Aller à PROD	Aller à CONS

Le programme principal comprend les appels des deux procedures.On vérifie à partir du clavier et l'écran que les deux pocessus arrivent éffectivement à communiquer entre eux, en appuyant sur une touche,c'est l'action de produire on a comme echos une écriture sur l'écran qui représente le processus de consommation.

-Le programme(2) "programme de développement" :

Ce programme comporte 4 procedures :

-La procedure "proc read.s" c'est la procedure de lecture de chaine de caractéres grâce à son appel ,on arrive à lire à partir du clavier une chaine de caractéres.

-La procedure "proc write.s",c'est la procedure d'écriture de chaine de caracrtéres, elle permet d'écrire correctement sur l'écran une chaine de caractéres.

-La procedure "proc read.n", c'est la procedure de lecture d'un nombre,elle permet donc de lire à partir du clavier un nombre donné.

-La procedure "proc write.n",c'est la procedure d'écriture des nombres sur l'écran.

Dans le programme principal on trouve les appels de ces procedures et des tests permettant de vérifier le fonctionnement de ces procedures.

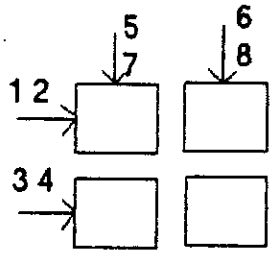
-Le programme(3) "produit de deux matrices" :

Dans ce programme les deux matrices sont mises en mémoire sous forme de vecteurs car le logiciel n'admet pas la notion de matrice,la communication entre processus se fait à patir des canaux up,down,left et right,à l'aide de l'instruction PAR on éffectue le calcul des différents produits et additions en paralléle.Le resultat final sera affiché sur l'écran.

Soit à réaliser le produit de deux matrices A et B :

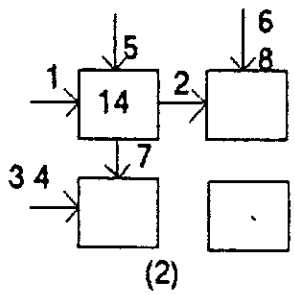
$$A = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Pour éffectuer ce produit en utilisant Occam, il nous faut alors 4 transputers tel que :



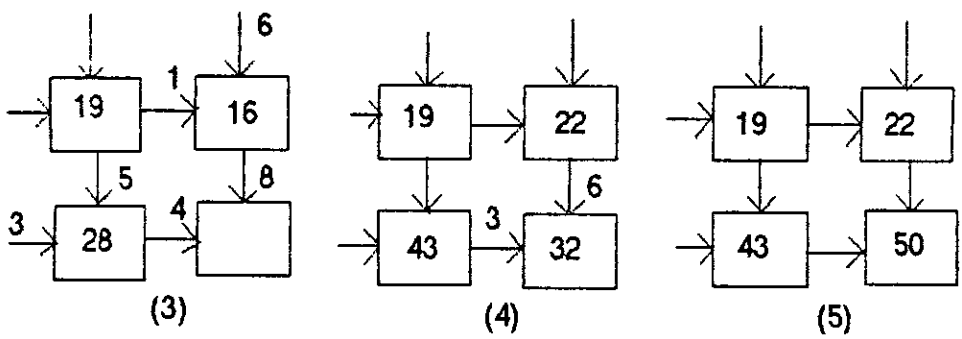
(1)

Les données rentrent dans les transputers à travers les canaux nord et ouest tel que :



(2)

Le transputer calcule le produit de ces donnés,il stocke ce produit et transfert les données à ses voisins immédiat à traves les canaux est et sud .



(3) (4) (5)

Ainsi les transputers évoluent en paralléle,on le voit bien en (3) ou les trois transputers éffectuent leur produits et leurs additions simultanément,l'opération continue jusqu'à obtenir le resultat final en (4).

Le programme (3) représente lea programmation de ce produit en Occam,ainsi à partir du fonctionnement des traunsputers pour éffectuer le produit des deux matrices A et B, on peut voir comment évolu chaque transputer comme une cellule systolique,ce qui nous a permet de simuler ces réseaux par Occam qui est le langage de programmation des transputers.

Le programme (4) est une version plus simplifiée du programme(3) ou les différents produits sont mis en procedure dont l'appel est fait dans le programme principale pour les différents i et j.

CONCLUSION

A partir de son étude ci-dessus occam peut être vu comme une solution au problème de la programmation des systèmes concurrents d'architecture quelconque.Il représente un outil trés puissant pour la description et l'implémentation d'algorithmes concurrents. Comme occam présente une relation privilégiée avec le transputer d'Inmos ,il peut être vu dans ce sens là comme le langage "assembleur" du transputer. De même on peut aussi concevoir des systèmes parallèles à partir d'occam et ensuite les réaliser en utilisant des transputers en tant que processus occam matérialisés. Ce lien intime entre le matériel et le logiciel constitue la grande nouveauté introduite par les microprocesseurs du type transputer. les transputers peuvent être programmés en utilisant la plus part des langages de haut niveau comme le langagr C,pascal Fortran.Le maximum de performance sera toutefois obtenu en programmant l'ensemble du système en occam.Avec les avantages d'un langage de haut niveau, l'efficacité propre à occam et la possibilité d'utiliser pleinement les caractéristiques particulières des transputers.Il faut préciser que le transputer et occam ont été conçus presque simultanément.
Le transputer inclut par suite des instructions et du matérel spécifique en vue d'une exécution optimisée des

modéles occam de la concurrence et de la communication entre processus.Occam peut ainsi, par exemple , être utilisé pour concevoir des systèmes concurrents à base de transputers, comme l'on utilise l'algèbre de Boole pour concevoir des applications en logique cablée.La relation étroite occam-transputer (logociel-matériel) facilite de surcroit cette conception.Un programme exécuté sur un transputer est formellement équivalent à un processus logiciel occam et par suite un réseau de transputer peut directement être décrit comme un programme occam.

CONCLUSION GENERALE

A partir de notre étude, on peut conclure que l'application des réseaux systoliques est trés bénifique,gâce à leur architecture paralléle, ceci d'une part et d'autre part ils permettent grâce à leur mode de synchronisation globale de suprimer tous les modules de commandes destinés à synchroniser les traitement ainsi, a partir d'une architecture systolique on régle les problèmes les plus difficiles de synchronisation posés par l'architecture paralléle simple (non systolique).

On a vu que l'application essentielle des réseaux systoliques est l'implémentation, on est arriver à implémenter des algorithmes sous formes de réseaux systoliques tel que les algorithmes d'opérations matricielles,cette caractéristique pourrait être expoiteé aussi pour implémenter d'autres algorithmes comme les algorithmes de traitement d'image ou ceux de la reconnaissance de la parole qui pourraient eux aussi adaptés la structures systoliques,ou les performances sont systèmatiquement les plus accrus.

Grâce au développement du logiciel on a pu verifier que

le langage qui s'adapte bien avec les réseaux systoliques est l'Occam .on a simulé les réseaux systoliques pour remplacer une structure matérielle par une structure logiciel. Il serait important de noter qu'à partir du système de programmation mit à notre disposition durant notre étude (un PC) on a pu éffectivement implémenter des programmes Occam qui simulent les réseaux systoliques,mais la notion du parallélisme n'est jamais réellement réalisée du fait de la structure séquentielle de la machine utilisée, donc on a pas réaliser un vrai parallélisme mais une simulation de celui-ci qui est bien le but de notre étude; et donc nous ne disposant pas actuellement d'une machine concurrente à multiprocesseurs on peut toujours implémenter des programmes concurrents et donc pratiquer la concurrence sur des machine séquentielles une fois que ses programmes tournent sur une machine séquentielle ils tourneront bien évidement plus tard sur une machine concurrente.

BIBLIOGRAPHIE

[1] A.Carling,

Parallel Processing (The teransputer and Occam)

Edition Sigma 1988

[2] S Chen,F Fuller,L Loomis, M Magleby ,S Whitney

Introduction to Computer Architectures

[3] P.Quinton Y.Robert,

Algorithmes et Architectures systoliques

Edition Masson 1989

[4] C.MEAD-L.Conway

Introduction aux Systèmes VLSI

Inter Edition 1980

[5] E.Hirch

Les transputers (application à la programmation concurrente)

Edition Eyrolles 1990

[6] Herscvic,

Introduction aux grands Ordinateurs Scientifiques

[7] B.Rayan,

La vague des multiprocesseurs

Micro système .septembre 91. PP 163-167

[8] M.Cornu,

Supper-Ordinateurs :Les chemins du futur

Dossier Micro système. septembre 87. PP 159-171

[9] R.Dettmer

Occam and the Transputer

Electronics & power .April 1985.PP 283-287

[10] D A.Fensome

The transputer a ptotoyping tool for systems

Computing & Control Engineering journal

january 90 PP. 41-45

[11] R.G.Melhem et W.C.Reinbold

A language for the simulation of systolic

Computer Architecture 1985 .PP 331-337

[12] P.Quinton

Les hyperordinateurs

La recherche Juin 1985

Volume16. N°= 167. PP 740-74

[13] J.Forte ,B.Wah

Systolic Arryas from concept to implmentation

IEEE Computer Architecture. 1987 PP.12-17

ANNEXES

UTILISATION DU LOGOCIEL :

ALT 1 : Check

ALT 2 : Complile

ALT 3 : Make a program

ALT 4 : Make SC

ALT 5 : Descriptor info

ALT 9 : Replace:

ALT 0 : List

F1 : Start of line

F2 : End of line

F3 : Undelete line

Alt F9 : Open file

Alt F10 : Close file

Alt F6 : Finish

Les Cannaux occam : il y'en a 25:

CHAN parametrs	AT	0	:
CHAN screen	AT	1	:
CHAN Keyboard	AT	2	:
CHAN Returncode	AT	3	:
CHAN Fileout 0	AT	4	:
CHAN Fileout 1	AT	5	:

```
CHAN Fileout 2      AT      6       :
CHAN Fileout 3      AT      7       :
CHAN Fileout 4      AT      8       :
CHAN Fileout 5      AT      9       :
CHAN Fileout 6      AT      10      :
CHAN Fileout 7      AT      11      :
CHAN Filein 0       AT      12      :
CHAN Filein 1       AT      13      :
CHAN Filein 2       AT      14      :
CHAN Filein 3       AT      15      :
CHAN Filein 4       AT      16      :
CHAN Filein 5       AT      17      :
CHAN Filein 6       AT      18      :
CHAN Filein 7       AT      19      :
CHAN Printer        AT      20      :
CHAN Modemout       AT      21      :
CHAN Modemin        AT      22      :
CHAN Setaddress     AT      23      :
CHAN Poke           AT      24      :
```

PROGRAMME (1)

```
--
PROC producer ( CHAN s.comms ) =
  CHAN input AT 2 :
  WHILE TRUE
    VAR x :
    SEQ
      input ? x
      s.comms ! x :
PROC consumer ( CHAN d.comms ) =
  CHAN output AT 1 :
  WHILE TRUE
    VAR y :
    SEQ
      d.comms ? y
      output ! y ; -2 :

CHAN comms :
PAR
  producer ( comms )
  consumer ( comms )
--  code

--  descriptor

--  code

--  descriptor

--  code

--  descriptor
```

PROGRAMME (2)

```
--
CHAN screen AT 1, keyboard AT 2 :
VAR digit [BYTE 5],k,string[BYTE 10]:
--****************PROCEDURE DE LECTURE DES CARACTERES**********************
PROC read.s (VAR string[ ]) =
  VAR string [5]:
  DEF DELETE = 8, EOL = 13 :
  --
  -- read a string from the keyboard into string[]
  -- the string will begin at start+1 in string[]
  -- string[BYTE 0] will contain the length (max 255) of the string
  --
  -- string terminated by 'RETURN' which is NOT echoed back to the screen
  --
  VAR i, ch:
  SEQ
    i := 1
    keyboard ? ch
    WHILE ch <> EOL
      IF
        ch = DELETE
          SEQ
            IF
              i > 1
                SEQ
                  i := i-1
                  screen ! ch ; ' ' ; ch; -2
              TRUE
                SKIP
            keyboard ? ch
        TRUE
          SEQ
            screen ! ch ; -2
            string[BYTE i] := ch
            i := i+1
            keyboard ? ch
    string[BYTE 0] := (i-1)
    screen ! string[BYTE 3]
    screen ! string[BYTE 1]
    screen ! string[BYTE 0]:
--******************PROCEDURE D'ECTRITURE DES CARACTERES********************
PROC write.s(VALUE string[]) =
  SEQ
    k:= 50
    WHILE k < 61
      SEQ
        --string[BYTE k]:=8
        k:=k+1
    SEQ
      --string[BYTE 0]:=60
      SEQ i=[ 1 FOR string[BYTE 0]]
        screen ! string[BYTE i] :
      --screen ! 80:
--********************PROCEDURE DE LECTURE DES NOMBRES**********************
PROC read.n (VAR nbr) =
  VAR n,v,i,valid,ok,digit.string[BYTE 7],negative:
  SEQ
    valid:=FALSE
    WHILE NOT valid
      SEQ
        read.s(digit.string)
        n:=0
        i:=1
        IF
```

```
          digit.string[BYTE 1] ='-'
            SEQ
              negative:= TRUE
              i:= 2
          TRUE
            negative:= FALSE
        ok := TRUE
        WHILE (i<= digit.string[BYTE 0] ) AND ok
          SEQ
            v := digit.string[BYTE i] - '0'
            IF
              ( v >= 0) AND (v <=9)
                SEQ
                  n := (n *10) + v
                  i:= i+1
              TRUE
                SEQ
                  ok := FALSE
                  screen ! 13
                  screen !10
                  write.s("  invalid number, try again ")
        IF
          ok
            valid := TRUE
          NOT ok
            SKIP
    IF
      NOT negative
        nbr := n
      negative
        nbr := -(n)
    screen ! nbr:
--**********************PROCEDURE D'ECRITURE DES NOMBRES*********************
PROC write.n(VAR nbre) =
  VAR digits[BYTE 12],i,negative,n,z :
  SEQ
    SEQ i = [1 FOR 10 ]
      digits[BYTE i]:= ' '
    digits[BYTE 1]:= '0'
    digits[BYTE 0]:= 11
    IF
      nbre >= 0
        SEQ
          negative := FALSE
          n := nbre
      nbre < 0
        SEQ
          negative := TRUE
          n:= -(nbre)
    i := 11
    WHILE n > 0
      SEQ
        digits[BYTE i] := (n\10) + '0'
        n := n/10
        i := i-1
    IF
      negative
        digits[BYTE i] := '-'
      TRUE
        SKIP
    write.s(digits):
--**********************PROGRAMME PRINCIPAL**********************************
VAR x,y[BYTE 70],i,nbr,digits[BYTE 12],nbre:
SEQ
```

```
    read.n(nbr)
    write.s(y)
    --read.s (y)
    --i := 0
    --WHILE i < 50
      --SEQ
        --y[BYTE i ] := 82
        --i := i +1
    --y[BYTE 49]:= 84
    --screen ! y[BYTE 5]
    --screen ! 13
    --screen ! 10
    --write.s (y)
    write.n(nbre)
    screen ! 20
    screen ! -5
    screen ! 70

--  code

--  descriptor
```

PROGRAMME (3)

```
--  program multiply
--  configuration constants
DEF n=2:
CHAN screen AT 1, keyboard AT 2:
VAR sinkH[n],sinkV[n]:
DEF vecA=TABLE[7,8,5,6]:
DEF vecB=TABLE[2,4,1,3]:
VAR result0,result1,result2,result3:

--  proc multiply
PROC mult(VAR T,CHAN
up,down,right,left)=
  VAR A,B:
  SEQ
    PAR
      up ? A
      left ? B
    T:=T+(A*B)
    PAR
      down ! A
      right ! B:


--  proc sinkV
PROC sinkV(CHAN south)=
  VAR V:
  SEQ
    south ? V:

--  proc sinkH
PROC sinkH(CHAN east)=
  VAR H:
  SEQ
    east ? H:

--  main program
VAR result[n*n]:
SEQ
  --  mat result=0
  result0:=0
  result1:=0
  result2:=0
  result3:=0

  --  program instance
  CHAN vert[n*(n+1)]:
  CHAN horiz[n*(n+1)]:
  SEQ i=[0 FOR n]
    PAR
      --  produce matA,matB
      PAR j=[0 FOR n]
        vert[j] ! vecA[(n*i)+j]
      PAR j=[0 FOR n]
        horiz[j] ! vecB[(n*i)+j]
      --  sink matA,matB
      PAR j=[0 FOR n]
        sinkV(vert[(n*n)+j])
      PAR j=[0 FOR n]
        sinkH(horiz[(n*n)+j])

      --  inst0
      VAR A,B:
      SEQ
        PAR
          vert[0] ? A
          horiz[0] ? B
        result0:=result0+(A*B)
        PAR
          vert[2] ! A
          horiz[2] ! B


      --  inst1
      VAR A,B:
      SEQ
        PAR
          vert[1] ? A
          horiz[2] ? B
        result1:=result1+(A*B)
        PAR
          vert[3] ! A
          horiz[4] ! B

      --  inst2
      VAR A,B:
      SEQ
        PAR
          vert[2] ? A
          horiz[1] ? B
        result2:=result2+(A*B)
        PAR
          vert[4] ! A
          horiz[3] ! B

      --  inst3
      VAR A,B:
      SEQ
        PAR
          vert[3] ? A
          horiz[3] ? B
        result3:=result3+(A*B)
        PAR
          vert[5] ! A
          horiz[5] ! B

--  print mat result
move.cursor(26,20)
SEQ
  write.number(result0)
  write.number(result1)
  write.number(result2)
  write.number(result3)
```

PROGRAMME (4)

```
--  PROGRAM
--
--  utility

--  read a number from keyboard
into a variable
PROC read.number ( VAR number ) =
  -- input a decimal number from
keyboard and echo it back to
screen

--  write a number to the screen
PROC write.number (  VALUE number
)=
  -- output a number to the
screen, right justified

--  move cursor
PROC move.cursor ( VALUE row,
column ) =



--  program multiply
--  configuration constants
DEF n=1:
CHAN screen AT 1, keyboard AT 2:
VAR sinkH[n],sinkV[n]:
DEF vecA=TABLE[7,8,5,6]:
DEF vecB=TABLE[2,4,1,3]:

--  proc multiply
PROC mult(VAR T,CHAN
up,down,right,left)=
  VAR A,B:
  SEQ
    PAR
      up ? A
      left ? B
    T:=T+(A*B)
    PAR
      down ! A
      right ! B:


--  proc sinkV
PROC sinkV(CHAN south)=
  VAR V:
  SEQ
    south ? V:

--  proc sinkH
PROC sinkH(CHAN east)=
  VAR H:
  SEQ
    east ? H:
```

```
--  main program
VAR result[n*n]:
SEQ
  --  mat result=0
  SEQ i=[0 FOR n*n]
    result[i]:=0

  --  program instance
  CHAN vert[n*(n+1)]:
  CHAN horiz[n*(n+1)]:
  SEQ i=[0 FOR n]
    PAR
      --  produce matA,matB
      PAR j=[0 FOR n]
        vert[j] ! vecA[(n*i)+j]
      PAR j=[0 FOR n]
        horiz[j] ! vecB[(n*i)+j]

      --  sink matA,matB
      PAR j=[0 FOR n]
        sinkV(vert[(n*n)+j])
      PAR j=[0 FOR n]
        sinkH(horiz[(n*n)+j])

      --  call mult
      PAR i=[0 FOR n]
        PAR j=[0 FOR n]
          mult(result[(n*i)+j],
               vert[(n*i)+j],
vert[(n*(i+1))+j],

horiz[(n*(j+1))+i],
horiz[(n*j)+i])

  --  print mat result

  SEQ i=[0 FOR n]
    SEQ j=[0 FOR n]

write.number(result[(n*i)+j])
```

GLOSSAIRE

Concurrence : cas ou des opérations sont éffectuées simultanément par des processeurs différents.

Pipline : traitement à chaine des données, une série de données subissent en circulant d'un processeur à un autre un même traitement.

Simulation : remplacement d'un systéme par un modéle plus simple ayant un comportement semblable.

Synchronisation : mise en concordance des processus. la synchronisation entre processus c'est le fait de permettre à un processus actif de changer d'état ou de faire changer d'état à un autre processus.