Algorithmes systoliques de la theorie a la pratique

Françoise André, Patrice Frison, Patrice Quinton

Algorithmes systoliques de la theorie a la pratique

Françoise André, Patrice Frison, Patrice Quinton

IRIA

CENTRE DE RENNES

IRISA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél.: 954 90 20

Rapports de Recherche

N° 214

ALGORITHMES SYSTOLIQUES : DE LA THÉORIE À LA PRATIQUE

Françoise ANDRÉ
Patrice FRISON
Patrice QUINTON

Mai 1983

ALGORITHMES SYSTOLIQUES: DE LA THEORIE A LA PRATIQUE

Françoise ANDRE
Patrice FRISON
Patrice QUINTON
IRISA, Campus de Beaulieu,
35042 RENNES-CEDEX

Publication Interne n° 196
19 pages
Mars 1983

RESUME: Cet article tente de faire le point sur les architectures systoliques, domaine de recherche en pleine évolution actuellement. Dans une première partie, on définit ce qu'est une architecture systolique et on indique les algorithmes pour lesquels des solutions systoliques ont été trouvées. Dans une seconde partie, on s'intéresse aux techniques qui permettent de trouver une structure systolique à partir des équations mathématiques d'un problème. On montre en particulier sur l'exemple très classique du produit de convolution comment l'on peut trouver de façon systématique de nombreuses versions systoliques de cet algorithme. Dans la troisième partie, on décrit une machine systolique pour la détection de mots dans la parole continue dont la réalisation est actuellement en cours à l'IRISA. Cette machine comporte une centaine de cellules de complexité moyenne, et permet de rechercher en temps réel (c'est-à-dire au fur et à mesure qu'une phrase est prononcée) un vocabulaire de près de 2000 mots. On décrit en particulier un circuit intégré expérimental réalisant la fonctionnalité d'une cellule du réseau, comportant 12000 transistors et qui est actuellement en cours de fabrication dans le cadre du projet universitaire CMP (Circuit multi-projets).

SUMMARY: The design of systolic arrays is a research area which motivates currently a large interest. This paper is a survey of recent developments in this field. In the first part, the systolic array concept is defined and problems that received a systolic implementation are listed. Second part deals with a systematic method for the design of systolic arrays, starting from the equations of the problem. Finally, part three describes a practical implementation of a systolic array for speech recognition. This machine is made out of a hundred of identical cells, and allows the real-time detection of vocabularies of 2000 words. We describe a 12000 transistors chip which implements the basic cell. This chip is currently being fabricated by the French MPC project.

calcul importante sans pour autant faire appel à des technologies ultra-rapides qui sont coûteuses;
- en donnant à la machine une structure régulière, on divise l'effort de conception par le facteur de régularité, puisqu'il suffit en principe de réaliser et mettre au point chaque type de cellule que l'on réplique ensuite autant qu'il est nécessaire;
- en supposant que le fonctionnement est synchrone et les communications sont locales, on simplifie considérablement les problèmes de synchronisation;
- enfin, en faisant un usage multiple de la même donnée par le jeu du pipeline, on minimise les échanges avec le calculateur hôte qui sont coûteux en temps.

S'il peut apparaître à priori que la définition d'une architecture systolique est extrêmement contraignante et limite son emploi à quelques algorithmes, le nombre de publications parues sur ce sujet semble prouver que bien peu de problèmes résistent à la "systolisation", tout au moins en théorie. Citons entre autre:
- en analyse numérique, le produit de matrice, le produit matrice-vecteur, la triangulation de matrice, l'inversion etc...
- en traitement de signal et d'image, la convolution à une ou deux dimensions, la transformée de Fourier discrète, l'interpolation etc...
- dans le domaine des algorithmes non numériques, la gestion de certaines structures de données (piles, files d'attente, arbres), certains algorithmes sur les graphes (fermeture transitive, recherche de composantes connexes), l'analyse syntaxique de langages (régulier et hors-contexte), la programmation dynamique etc...

Le lecteur intéressé trouvera une liste non exhaustive de références bibliographiques sur ce sujet dans [KUNG 82].

Cette énumération ne doit cependant pas masquer les problèmes non résolus concernant l'utilisation en pratique de telles architectures, aussi séduisantes soient-elles. Citons en deux:
- une application concrète se présente très rarement sous la forme d'un algorithme connu, répertorié dans la littérature. Il importe donc de se doter de moyens permettant la synthèse de machines systoliques. Sur ce sujet, les résultats sont peu nombreux, et en tout cas insuffisants. C'est ce problème que nous abordons dans la seconde partie.
- Par ailleurs, l'adéquation d'une machine à une application donnée dépend d'un très grand nombre de facteurs. La rigidité d'une architecture systolique est souvent peu compatible avec la souplesse d'utilisation requise. Ceci explique que ce type d'architecture n'ait pas encore donné lieu (à notre connaissance) à des applications industrielles. Dans la troisième partie de cet article, nous nous intéressons à l'utilisation de telles structures pour la reconnaissance de la parole à partir d'un exemple (partiel) de réalisation. Nous nous attachons en particulier à montrer quels problèmes pratiques se sont posés et comment ils ont été résolus.

En conclusion, nous indiquons les perspectives offertes par ce type d'architectures qui restent encore largement du domaine de la recherche.

3-ILLUSTRATION D'UNE APPROCHE SYSTEMATIQUE POUR LA CONSTRUCTION D'ARCHITECTURES SYSTOLIQUES

L'objectif de ce paragraphe est de montrer à l'aide de l'exemple du produit de convolution introduit précédemment, l'approche que nous proposons pour apprèhender de manière systèmatique l'étude des algorithmes systoliques et des architectures adaptées.

La démarche comporte trois phases:
- réécriture des équations du problème sous forme d'équations récurrentes uniformes;
- définition de fonctions temporelles spécifiant le cadencement des calculs en fonction des vitesses de propagation des données;
- définition d'architectures systoliques par application de fonctions d'allocations des calculs aux processeurs.

3.1-ALGORITHMES SYSTOLIQUES POUR LE PRODUIT DE CONVOLUTION

Cette première étape consiste à transformer les équations de manière à mettre en évidence d'une part, les calculs élémentaires effectués par chaque cellule de traitement; les flux de données entre les cellules, d'autre part. L'équation (1) du paragraphe 2 peut être mise sous la forme

$$
(2)\quad
\begin{aligned}
&y(i,k)=y(i,k-1)+w(k)x(i-k) && i\geq 0,\ K\geq k\geq 0\\
&y(i,-1)=0 && i\geq 0\\
&x(i-k)=0 && i<k
\end{aligned}
$$

Pour tout point (i,k) du domaine $D=\{0\leq i;0\leq k\leq K\}$, le calcul élémentaire est donc:

$$y(s)=y(e)+w(e)x(e)$$

moyennant l'affectation aux entrées y(e),w(e),x(e) des valeurs correctes y(i,k-1),w(k),x(i-k).

Il devient alors évident que le calcul au point (i,k) dépend du calcul au point (i,k-1). Le système d'équations (2) ne possède cependant pas les propriétés d'un algorithme systolique car les données x(i-k) et w(k) sont obtenues individuellement pour chacun des calculs (i,k) de D à partir de la mémoire. Les transferts de données ne sont donc pas réalisés par des communications locales et régulières.

Une alternative au fonctionnement décrit ci-dessus consiste à faire circuler les données w(k) et x(i-k) de calcul en calcul. Par exemple, w(k) qui est utilisé au point (i,k), l'est aussi au point (i-1,k); de même x(i-k), utilisé en (i,k), l'est aussi en (i-1,k-1). Dans ce cas, on peut écrire (2) sous la forme:

$$
(3)\quad
\begin{aligned}
&y(i,k)=y(i,k-1)+w(i-1,k)x(i-1,k-1)\\
&w(i,k)=w(i-1,k) && 0<i\\
&x(i,k)=x(i-1,k-1) && 0\leq k\leq K
\end{aligned}
$$

Ce système d'équations est dit uniforme car le calcul au point (i,k) ne dépend que de valeurs provenant de points obtenus par des translations indépendantes de (i,k). Les liens entre les points de D

sont donc réguliers comme le montre la figure 2. Ce graphe met en évidence un ensemble de vecteurs de dépendances, $H=\{\Theta(w),\Theta(y),\Theta(x)\}$, qui définit en chaque point (i,k) de D les points à partir desquels (i,k) obtient ses données. Pour le graphe de la figure 2, nous avons:

$$\Theta(w)=(-1,\emptyset)$$
$$\Theta(y)=(\emptyset,-1)$$
$$\Theta(x)=(-1,-1)$$

Bien évidemment, les dépendances que nous avons créées ne sont pas les seules possibles. Des solutions différentes sont proposées dans [QUINTON 83].

La première étape vers une réalisation systolique consiste donc à définir les systèmes d'équations récurrentes uniformes qui satisfont les conditions du problème et à en dégager les vecteurs de dépendances.

3.2-CADENCEMENT DES CALCULS

L'étape suivante de la méthode consiste à plannifier les instants d'éxécution des différents calculs. Une plannification est spécifiée au moyen d'une fonction temporelle t, associant à chaque point d de D, l'instant t(d) de N où le calcul en d est effectué. La fonction t doit respecter la contrainte de dépendance entre les calculs créée par les flots de données, ce qui s'exprime par la règle suivante: si le calcul au point a dépend du calcul au point b alors

$$t(a)>t(b)$$

Par exemple, pour l'algorithme du produit de convolution défini par les équations (3), la fonction

$$t(i,k)=i+k$$

respecte cette condition et l'on obtient le cadencement décrit par la figure 3.

Dans le cas général, étant donné D et un ensemble de vecteurs de dépendances $\Theta(i)$, la fonction t peut ne pas exister. Dans [QUINTON 83], Quinton donne une condition nécessaire et suffisante sur Θ et D pour qu'une fonction t existe. Dans le cas particulier où D est un domaine convexe, les conditions d'obtention d'une solution optimale sont précisées.

3.3-CHOIX D'UNE ARCHITECTURE SYSTOLIQUE

La dernière étape de la méthode consiste à définir une architecture sur laquelle l'algorithme va être réalisé. L'architecture se compose d'un nombre fini de processeurs, ce qui impose de choisir leur mode d'allocation aux calculs définis en chaque point de D.

Pour l'exemple du produit de convolution, une solution à ce problème d'allocation peut être spécifiée à l'aide de la fonction linéaire de D dans un sous ensemble fini de N suivante:

$$a(i,k)=k \quad \emptyset\leq k\leq K$$

Les calculs effectués par chaque processeur et les communications entre eux sont déduits de l'algorithme et de la fonction de temps t(i,k), ce qui conduit à l'architecture présentée par la figure 1 pour K=2.

La répartition des calculs sur les processeurs met en évidence une condition que doit satisfaire la fonction d'allocation: deux calculs en des points différents x et y de D, qui ont même fonction de temps, ne peuvent être alloués au même processeur. Autrement dit

$$\forall\ x,y \text{ appartenant à } D,\ a(x)=a(y) \Rightarrow t(x)\neq t(y)$$

Des méthodes pour déterminer des fonctions d'allocation dans des cas plus généraux sont définies dans [QUINTON 83].

4-UNE MACHINE SYSTOLIQUE POUR LA DETECTION DE MOTS

4.1-LE PROBLEME

La reconnaissance de la parole comprend en général plusieurs étapes. La première consiste à digitaliser le signal de parole et à en extraire ses paramètres acoustiques. Dans une seconde étape, appelée analyse phonétique, les sons élémentaires du langage sont extraits et identifiés. A l'issue de cette analyse, la phrase prononcée est transformée en une suite de N segments phonétiques. Notons X=(X(1),...,X(N)) cette séquence. Chaque segment X(i) est lui-même constitué d'un ensemble de symboles phonétiques, qui représentent les phonèmes les plus probables pour ce segment. Dans une troisième étape, tous les mots du vocabulaire du langage choisi sont comparés à la phrase codée, chaque mot étant comparé à chaque sous-suite de la phrase codée. Cette étape, appelée détection de mots, est en général très longue.

La détection de mot consiste à rechercher dans la phrase codée, toutes les occurences d'un mot donné sous sa forme phonétique. Soit Y=(y(1),...,y(M)) la suite de phonèmes représentant ce mot. Le problème se résume à comparer la suite Y avec toute sous-suite (X(k+1),...,X(k+P)) extraite de X, et à évaluer la qualité de cette ressemblance.

Pour évaluer le degré de ressemblance entre un mot et une partie de la phrase codée, il doit être tenu compte des erreurs éventuellement commises par l'analyseur phonétique. Les erreurs considérées ici sont de trois types: des confusions, des insertions et des omissions. Une confusion apparaît lorsqu'un phonème prononcé z est traduit par un segment contenant plusieurs phonèmes et dans lequel z peut ne pas se trouver. Une insertion se produit lorsque l'analyseur phonétique détermine des segments en surnombre. Enfin un phonème peut être omis lorsque l'analyseur ne détecte pas le phonème et ne construit donc pas le segment correspondant.

La figure 4 donne un exemple de transcription phonétique d'une phrase prononcée, en l'occurrence: "liste des connecteurs". A la suite de l'analyse phonétique, la phrase a été scindée en 13 segments. On note que le segment X(1) à été inséré en début de la phrase. Parmi les omissions, on note le phonème t du mot "liste", le second phonème k

du mot "connecteur", et enfin le phonème r de ce même mot en fin de phrase.

Dans cette phrase codée, le détecteur de mots trouvera par exemple, le mot "liste" entre les segments X(1) et X(5), le mot "des" entre X(6) et X(7), et le mot "connecteur" entre X(8) et X(13). Mais il trouvera également, par exemple, les mots "piste" entre X(1) et X(5), "vecteur" entre X(10) et X(13).

4.2-L'ALGORITHME

Le principe de l'algorithme de détection de mot est basé sur un modèle probabiliste de l'analyseur phonétique. Supposons que le dernier phonème y(M) soit un symbol spécial] appelé marqueur de fin de mot. Au mot Y est associé un automate d'état fini probabiliste (noté AEFP dans la suite du texte) qui modélise le comportement de l'analyseur phonétique lorsqu'il traite Y. La figure 5 décrit l'AEFP associé à un mot de 2 phonèmes (y(1),y(2),]). Cet AEFP contient 4 états. L'état S(i) décrit le comportement de l'analyseur phonétique lorsqu'il traite le phonème y(i+1). Trois possibilités s'offrent à l'analyseur:

- rester dans l'état S(i) avec une probabilité Pi(y(i+1)) et produire un phonème x avec une probabilité conditionnelle qi(x/y(i+1)): ce cas modélise l'insertion;
- entrer dans l'état S(i+1) avec la probabilité Pc(y(i+1)) et produire un phonème x avec un probabilité conditionnelle qc(x/y(i+1)): c'est le cas de la confusion;
- entrer dans l'état S(i+1) avec la probabilité Po(y(i+1)), sans produire de phonème: c'est le cas de l'omission.

Il est clair que, dans le cas du marqueur de fin de mot, Pi(])=Pc(])=Ø, afin d'éviter que l'AEFP ne produise le marqueur. Supposons que chaque segment X(j) ne contiennne qu'un seul phonème x(j), il a été montré [BAHL 76] que la probabilité que Y ait été prononcé est égale à la probabilité que l'AEFP associé à Y produise la suite de phonèmes x(1),...,x(P) en transitant de S(Ø) à S(M+1). Soit L(i,j) la probabilité pour l'AEFP d'entrer dans l'état S(i) après avoir produit x(1),...,x(j), alors L est caractérisé par les formules suivantes:

(4) $L(i,j) = Lc(i,j)+Li(i,j)+Lo(i,j)$

avec

(5) $Lc(i,j)= L(i-1,j-1)*Pc(y(i))*qc(x(j)/y(i))$

(6) $Li(i,j)= L(i,j-1)*Pi(y(i+1))*qi(x(j)/y(i+1))$

(7) $Lo(i,j)= L(i-1,j)*Po(y(i))$

Lc(i,j), Li(i,j) et Lo(i,j) dénotent respectivement les probabilités pour l'AEFP d'entrer dans l'état S(i) après confusion de x(j) et y(i), insertion de x(j) ou omission de y(i). Les équations (4) à (7) sont valides pour $1 \leq i \leq M$ et $1 \leq j \leq P$, avec les conditions initiales suivantes:

(5.1) $(\forall i)(1\leq i\leq M)$ $Lc(i,\emptyset)=\emptyset$,
$(\forall j)(1\leq j\leq P)$ $Lc(\emptyset,j)=\emptyset$,
$Lc(\emptyset,\emptyset)=1$

(6.1) $(\forall i)(\emptyset<i\leq M)$ $Li(i,\emptyset)=\emptyset$

(7.1) $(\forall j)(\emptyset\leq j\leq P)$ $Lo(\emptyset,j)=\emptyset$

et finalement, pour compléter le schéma de récurrence:

(8) $(\forall j)(\emptyset\leq j\leq P)$ $L(M+1,j)=Lo(M+1,j)=L(M,j)$

qui dérive du fait que $Pc(])=\emptyset$, $Po(])=1$ et que $S(M+1)$ n'a pas de transition d'insertion.

Avec cet algorithme, la détection de mots dans une phrase est alors réalisée de la manière suivante. Soit $V=\{Y(1),\ldots,Y(d)\}$ un vocabulaire de d mots à reconnaître et soit $X=(X(1),\ldots,X(N))$ la transcription phonétique d'une phrase prononcée. Le problème est de comparer chaque mot Y de V à chaque sous-suite $(X(k+1),\ldots,X(k+p))$ de X $(\emptyset\leq k\leq k+p\leq N)$. Chacune de ces comparaisons fournit une vraisemblance L. Si L est une valeur suffisamment bonne (par exemple, si L dépasse un certain seuil), on dit que Y est détecté entre les segments X(k+1) et X(k+p).

4.3-DESCRIPTION DU RESEAU DE PROCESSEURS

L'idée de base consiste à attacher un processeur à chaque point (i,j). Notons <i,j> le processeur associé à (i,j). <i,j> reçoit les valeurs Lc(i,j), Lo(i,j) et Li(i,j). Il calcule d'abord L(i,j) en appliquant l'équation (4) puis il calcule Lc(i+1,j+1), Li(i,j+1) et Lo(i+1,j) en appliquant respectivement les équations (5),(6) et (7). Pour effectuer ces calculs, <i,j> doit recevoir également les valeurs X(j+1) et y(i+1). En conséquence, <i,j> doit être connecté à <i-1,j-1>, <i-1,j> et <i,j-1> pour pouvoir calculer L(i,j). La stucture du réseau est représentée sur la figure 6.

Cette structure montre que chaque processeur possède 6 ports d'entrée/sortie (3 ports d'entrée et 3 ports de sortie). Les lignes de connexion entre processeurs sont unidirectionnelles (leur direction est indiquée par les flèches). Pour chaque processeur les ports d'entrées/sorties sont notés comme suit: Ed (entrée diagonale), Ev (entrée verticale), Eh (entrée horizontale) pour les entrées et Sd (sortie diagonale), Sv (sortie verticale), Sh (sortie horizontale) pour les sorties.

A chaque cycle le processeur <i,j> reçoit les données suivantes:
- sur Ev, successivement X(j+1) et Lo(i,j)
- sur Ed, Lc(i,j)
- sur Eh, successivement y(i+1) et Li(i,j)

Par ailleurs, il mémorise d'une façon permanente les distributions de probabilité qc(x/y), Pi(y) et Po(y). Enfin, le processeur <i,j> émet ensuite les valeurs suivantes:
- y(i+1) et L(i,j+1) sur Sh;
- Lc(i+1,j+1) sur Sd;
- X(j+1) et Lo(i+1,j) sur Sv.

Ce traitement est valide pour les processeurs intérieurs i.e. les processeurs <i,j> satisfaisant 1≤i≤M et 1≤j≤P. Les remarques suivantes montrent que le traitement décrit précédemment est également valide pour les processeurs des bords.

- les équations (5.1) et (6.1) définissent les valeurs initiales à envoyer aux processeurs <i,j> avec i=Ø et 1≤j≤P-1 ou j=Ø et 1≤i≤M.
- les processeurs <M,j> avec 1≤j≤P-1, produisent L(M+1,j) sur leur port Sv, puisque Pc(])=Ø.
- enfin, considérons les processeurs <i,P> où Ø≤i≤M. Afin que les processeurs se comportent exactement comme les autres, il est pratique d'introduire un segment fictif de fin de phrase X(P+1) contenant uniquement le marqueur de fin de mot]. Si l'on pose:

$$(\forall y)\ qc(]/y)=qi(]/y)=Ø$$

on montre que <i,P> délivre Lo(i+1,P) sur son port Sv et donc que <M,P> délivre L(M+1,P)=Lo(M+1,P) sur son port Sv.

4.4-DETECTION D'UN MOT

Supposons que le réseau fonctionne de façon synchrone. Aux instants successifs Ø,1,...,t,.. chaque processeur exécute le calcul décrit au paragraphe précédant. On veut utiliser le réseau pour détecter le mot Y=(y(1),...,y(M)) dans la séquence X=(X(1)...X(P)). La figure 7 illustre le fontionnement du réseau.

Au temps t=Ø, le processeur <Ø,Ø> reçoit X(1) et Ø sur Ev, 1 sur Ed et y(1) et Ø sur Eh. <Ø,Ø> exécute son programme puis émet X(1) et Lo(1,Ø) sur Sv, Lc(1,1) sur Sd et y(1) et Li(Ø,1) sur Sh.

Au temps t=1, <1,Ø> reçoit les valeurs envoyées par <Ø,Ø> sur Ev, Ø sur Ed, y(2) et Ø sur Eh. Il envoie ensuite X(1) et Lo(2,Ø) sur Sv, Lc(2,1) sur Sd et y(2) et Li(1,1) sur Sh. Au même moment , <Ø,1> reçoit X(2) et Ø sur Ev, et Ø sur Ed et y(1) et Li(Ø,1) sur Eh. Il produit ensuite X(2) et Lo(1,1) sur Sv, Lc(1,2) sur Sd, Y(1) et Li(Ø,2) sur Sh.

Plus généralement, au temps t, X(t+1) et Ø sont disponibles sur l'entrée Ev du processeur <Ø,t> et Ø sur l'entrée de <Ø,t>. De même , Y(t+1) et Ø sont envoyés sur Eh de <t,Ø> et Ø sur Ed de <t,Ø>. La diagonale de processeurs <i,j> avec i+j=t, délivre ensuite X(i+1) et Lo(i+1,j) sur Sv, Lc(i+1,j+1) sur Sd et enfin y(i+1) et Li(i,j+1) sur Sh.

Au temps t=M+P, on voit que la probabilité L(M+1,P) de correspondance entre X et Y apparaît sur le processeur <M,P>. On peut noter que le réseau fonctionne également pour la détection de mots de longueur l inférieure à M, à condition que ces mots soient complétés par M-l marqueurs de fin de mot. De même , si la phrase a une longueur l inférieure à P, la transcription phonétique de cette phrase doit être complétée par P-l marqueurs de fin de phrase.

4.5-FONCTIONNEMENT DU RESEAU EN MODE PIPELINE

Supposons tout d'abord que les mots du vocabulaire aient une longueur inférieure ou égale à M. Nous avons vu qu'à l'instant t,

seuls les processeurs <i,j> avec i+j=t sont actifs. Ceci suggère une meilleure méthode d'introduction des données dans le réseau afin d'effectuer des traitements à la chaine (ou pipeline).

Le pipeline sur les entrées y(i), appelé aussi pipeline sur les mots, est obtenu en fixant sur les entrées verticales du réseau les valeurs X(1),...,X(P) et en présentant successivement les mots Y(Ø),...,Y(d) sur les entrées horizontales. Il permet de comparer consécutivement chaque mot du vocabulaire V, avec la même séquence X(1),...,X(P). Ce mode de fonctionnement est illustré sur la figure 8. A chaque cycle, le premier segment Y(1,1) d'un nouveau mot Y(1) est introduit dans le réseau vers le processeur <Ø,Ø>. Aux cycles suivants, les segments Y(1,2),Y(1,3),...,Y(1,M+1) sont envoyés respectivement sur <1,Ø>,<2,Ø>,...,<M,Ø> à raison de 1 par cycle processeur.

L'utilisation la plus naturelle de ce mode de fonctionnement consiste à comparer tous les mots du vocabulaire aux sous-suites commencant à X(1), puis aux sous-suites commencant à X(2) et ainsi de suite. En effet, si l'on désire reconnaître de la parole en temps réel, ceci permet le lancement du processus de détection dès que les premiers segments de X sont produits par l'analyseur phonétique, le traitement s'opérant au fur et à mesure de leur production. C'est ce mode de fonctionnement que l'on considère dans la suite du texte.

4.6-MISE EN OEUVRE DU RESEAU

Le but de ce paragraphe est de décrire l'architecture des processeurs qui réalisent la structure multiprocesseur présentée dans le paragraphe précédant. Trois points seront successivement abordés: le nombre de processeurs, le contrôle du système et les connexions entre processeurs.

4.6.1-TAILLE DU RESEAU

Si l'on suppose que toutes les correspondances entre un mot de longueur M et une phrase de longueur N sont vraisemblables, le nombre de processeurs est (M+1)(N+1). Dans la pratique, on constate que seules les valeurs L(i,j) "proches" de la diagonale i=j sont significatives. Il est habituel de ne considérer que les valeurs (i,j) pour lesquelles:

$$abs(i-j) \leq delta$$

où delta est une constante. Cela conduit à donner au réseau une forme telle que celle de la figure 9.

Les valeurs X(j) et Y(i) peuvent être entrées directement sur les processeurs des bords de la diagonale à condition de leur appliquer un retard approprié. Les formules (6.1) et (7.1) fournissant les valeurs initiales des termes Li et Lo doivent être modifiées de la manière suivante:

(6.2) $(\forall i),(\forall j),\ (i-j \geq delta)\ Li(i,j)=\emptyset$

(7.2) $(\forall i),(\forall j),\ (j-i \geq delta)\ Lo(i,j)=\emptyset$

Un calcul simple montre alors que le nombre de processeurs du réseau est:

$$Np=(M+1)(2\ delta+1)-delta(delta+1)/2$$

A titre indicatif, M peut être fixé à 10 et delta à 4 ce qui donne Np=89 processeurs.

4.6.2-CONTROLE DU SYSTEME

Comme il a été vu précédemment, le cycle de base d'un processeur se décompose en 3 parties: réception des données sur les ports d'entrée, calcul des valeurs Lc, Lo, Li et envoie des données sur les ports de sortie. Les processeurs peuvent fonctionner indépendemment, en supposant par exemple que les communications interprocesseurs se font suivant un mécanisme de producteur-consommateur. Ceci entraine une structure de machine très générale de type MIMD.

Or, il a été montré que lors du traitement en mode pipeline sur les mots, les processeurs d'une même diagonale traitent un même mot et que, de plus, ils effectuent le même programme (cycle de base). Cette remarque suggère le contrôle des processeurs d'une même diagonale par un contrôleur unique. Celui-ci envoie un seul flot d'instructions exécutées simultanément par tous les processeurs d'une même diagonale (mode de fonctionnement SIMD).

Par ailleurs, il est clair que les processeurs de deux diagonales successives exécutent le même programme avec un certain décalage. En fixant ce décalage à un cycle processeur exactement, il est alors possible de contrôler les deux diagonales avec le même contrôleur. Par conséquent, on en déduit qu'un seul contrôleur (appelé contrôleur général et noté CG) est utilisable pour tout le réseau, tous les processeurs exécutant la même instruction simultanément avec leur données propres. Cependant, il est nécessaire de pouvoir inhiber certaines diagonales afin que les processeurs qui les composent soient inactifs, par exemple, lors du début ou de la fin du pipeline sur les mots.

Dans ce but, à chaque diagonale est associée un contrôleur de diagonale (noté CD). Ce contrôleur peut se trouver dans deux états. Dans l'état actif, il transmet aux processeurs de sa diagonale, les instructions qui lui sont envoyées par le contrôleur général. Dans l'état inactif, il inhibe l'instruction du contrôleur, de telle sorte que les processeurs de la diagonale n'effectuent aucune opération. La figure 10 montre l'organisation du réseau de processeurs et de ces contrôleurs.

Deux bus sont contrôlés par le contrôleur général: le bus d'instructions, sur lequel est codée l'instruction à exécuter et le bus de validation qui sert à activer ou désactiver un contrôleur de diagonale.

4.6.3-CONNEXIONS INTERPROCESSEURS

La struture du réseau nécessite théoriquement que chaque processeur possède 6 ports d'entrée-sortie, (3 d'entrée et 3 de sortie). En réalité, il est suffisant de ne mettre en oeuvre que 3 connecteurs d'entrée-sortie par processeur, deux connecteurs d'entrée et un connecteur de sortie. La figure 11, montre comment les processeurs sont interconnectés physiquement. Lorsqu'un processeur lit une donnée de type t, sur son port d'entrée verticale (resp horizontale), le fonctionnement synchrone du réseau entraine que tous les processeurs lisent simultanément sur leur port V (resp port H) une donnée de type t et que, par conséquent, tous les processeurs ont affiché sur leur port de sortie la valeur de type t qu'ils viennent d'utiliser ou de calculer.

Dans le cas d'un transfert logique diagonal, on constate que l'information reçue par un processeur doit être mémorisée pendant un cycle processeur avant de pouvoir être utilisée. En effet, si un processeur émetteur appartient à la diagonale d, le processeur receveur appartient à la diagonale d+2 (voir figure 6). Un transfert logique diagonal sera remplacé par deux transferts physiques: un transfert vertical, puis un transfert horizontal.

4.7-STRUCTURE INTERNE DU PROCESSEUR

Le processeur est constitué de 4 modules principaux:
- un module mémoire où sont stockées les valeurs des probabilités de confusion Qc(x/y);
- une unité arithmétique et logique capable d'effectuer des opérations élémentaires telles que soustraction, addition, incrémentation;
- un tableau de registres généraux servant de mémoire de travail;
- un module, noté Z, réalisé à l'aide d'un PLA, effectuant une fonction de tabulation.

Ces modules sont organisés autour d'un bus unique comme l'illustre la figure 12. Egalement connectés au bus, se trouvent 3 registres spécialisés, notés RS (registre sortie), RH (registre entrée horizontale), RV (registre entrée verticale). Ils permettent au processeur de communiquer avec l'extérieur.

Un module de contrôle fait également partie du processeur. Son rôle est de recevoir les instructions envoyées par le contrôleur CG, de les décoder et d'envoyer les commandes ainsi engendrées vers les différents modules. La description et mise en place des modules sur la plaquette de silicium sont détaillées dans [FRISON 83].

5-CONCLUSION

Nous avons montré dans cet article ce qu'est une architecture systolique, comment sous certaines conditions on peut en synthétiser à partir des équations d'un problème et enfin, une réalisation de machine systolique pour la reconnaissance de la parole. Ainsi que nous l'avons déja signalé, un important effort tant théorique que pratique reste à faire pour que les architectures systoliques entrent dans l'arsenal des techniques classiques employées au stade industriel. Cependant, les caractéristiques des prototypes existant sont plus qu'encourageantes et

deviendront d'autant plus intéressantes à mesure que la densité des circuits augmente. Des problèmes de fond restent à résoudre: sur le plan théorique, déterminer la classe des algorithmes qui peuvent être systolisés, élaborer des méthodes de synthèse suffisamment sophistiquées pour prendre en compte des problèmes concrêts, pour ne citer que ceux là. Sur un plan plus pratique, il faut pouvoir déterminer à priori quelles sont les applications qui bénificient réellement d'une approche systolique par rapport à des approches plus classiques.

BIBLIOGRAPHIE

BAHL 76
Bahl L.R., Jelinek F., Decoding for Channels with Insertions, Deletions, and Substitutions with Application to Speech Recognition, IEEE Trans. on Information Theory, 21, no 4, Juillet 1976, pp 404-411.

FRISON 83
Frison P., Un Processeur Intègrè pour la Reconnaissance de la Parole, Publication Interne IRISA, no 192, Mars 1983.

KUNG 82
Kung H.T., Why Systolic Architectures?, Computer vol 15, no 1, Janvier 1982, pp 37-46.

QUINTON 83
Quinton P., The Systematic Design of Systolic Arrays, Publication Interne IRISA, no 193, Mars 1983.

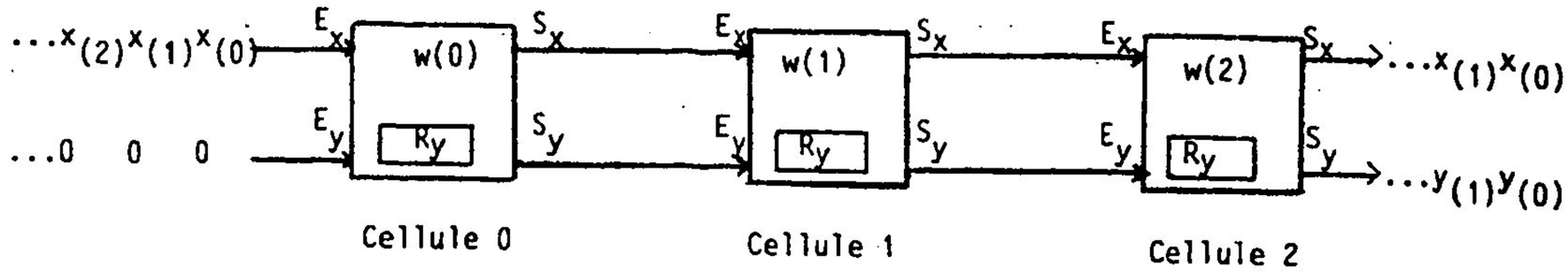
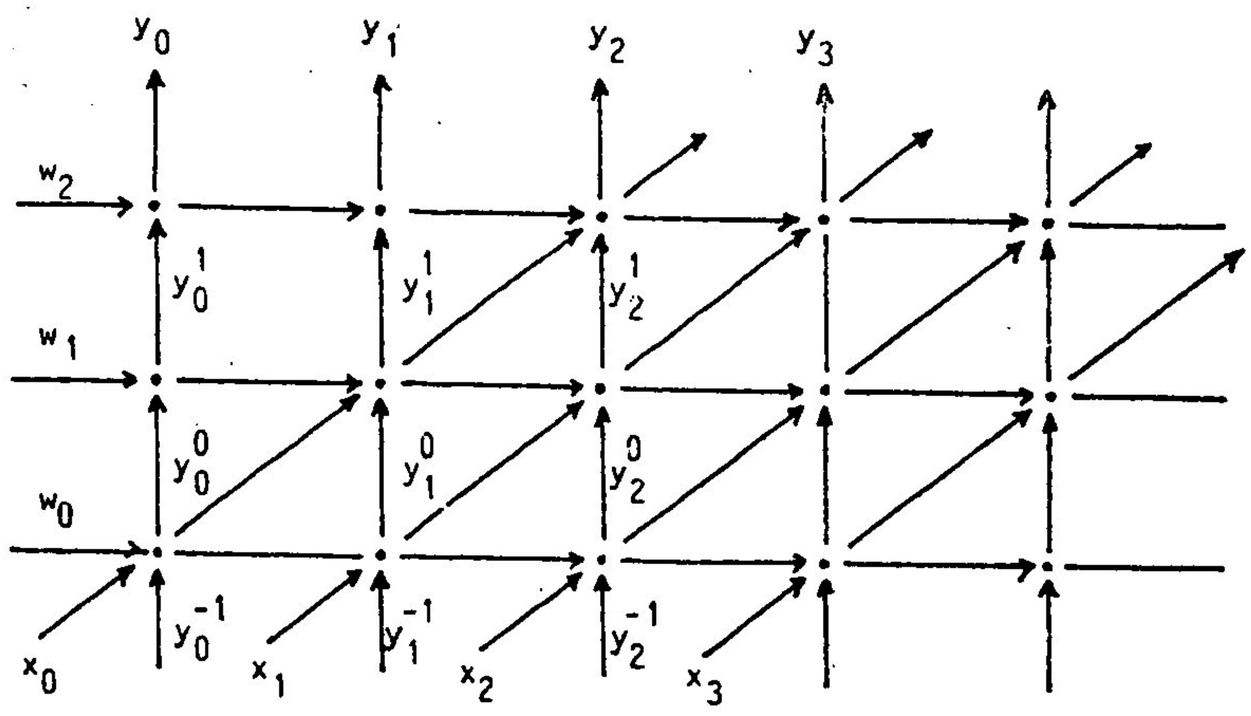


Figure 1 : Architecture systolique pour le produit de convolution (K=2)



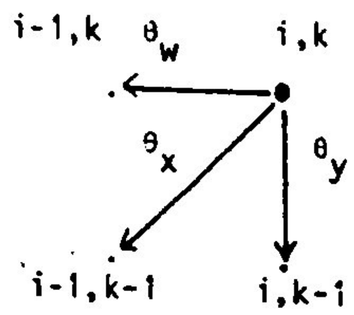


Figure 2 : Graphe de dépendance pour le produit de convolution décrit par les équations (3)

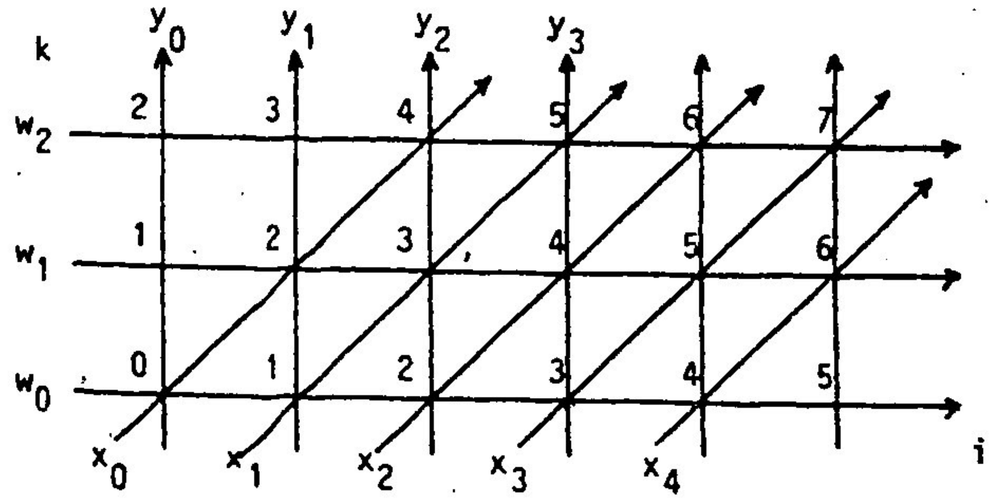


Figure 3 : Cadencement spécifié par la fonction t(i,k)=i+k

Comportement:	I	C	C	C	O	C	C	C	C	C	C	C	O	C	C	O
Phrase prononcee:		l	i	s	t	œ	d	e	k	o	n	ɛ	k	t	œ	r
X(i):	1	2	3	4		5	6	7	8	9	10	11		12	13	
Transcription phonetique :	p t k	l j	i e y	s f ʃ t k		o œ ɛ̃	b d g	e y ɛ	p t k	o œ ɛ̃	n v ʃ g m d z b w j	ɛ ɛ̃ e		p t k	o œ	

Figure 4 : Exemple de transcription phonétique

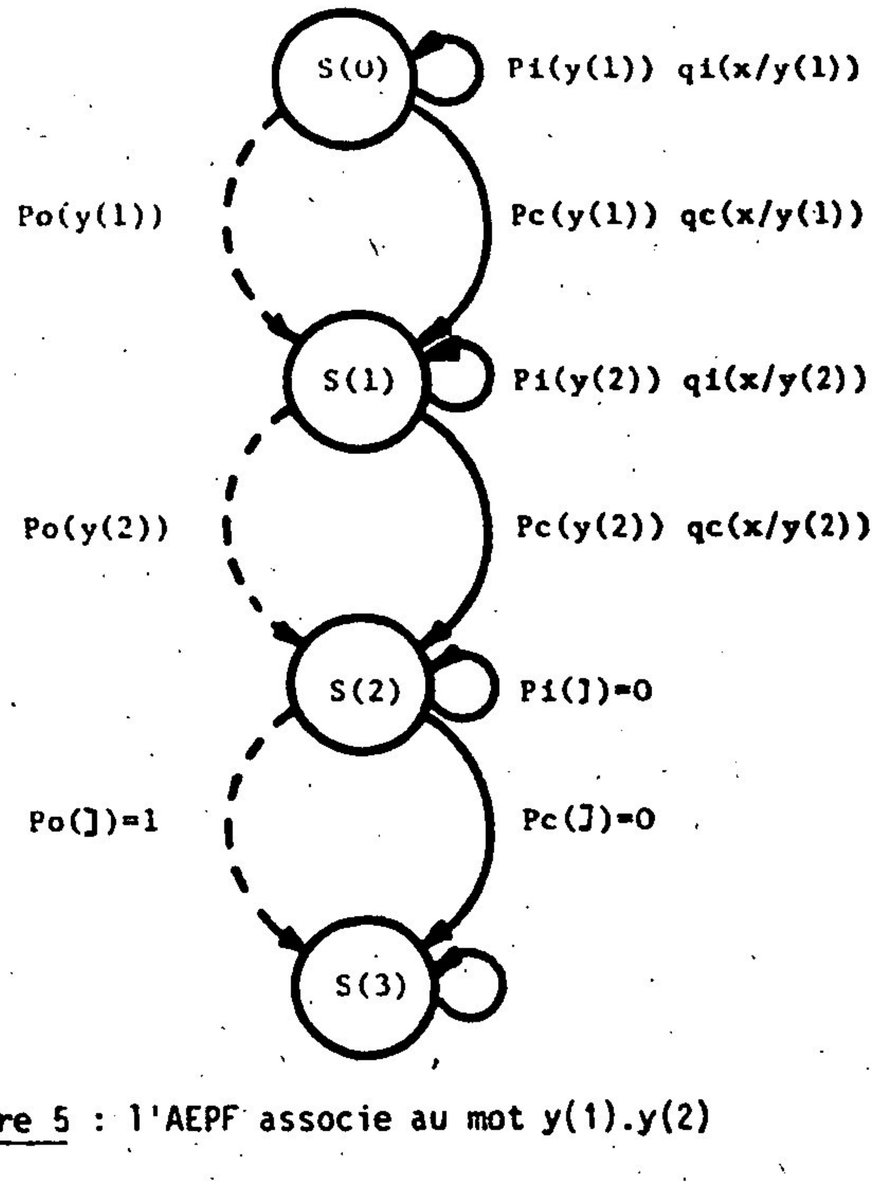


Figure 5 : l'AEPF associe au mot y(1).y(2)

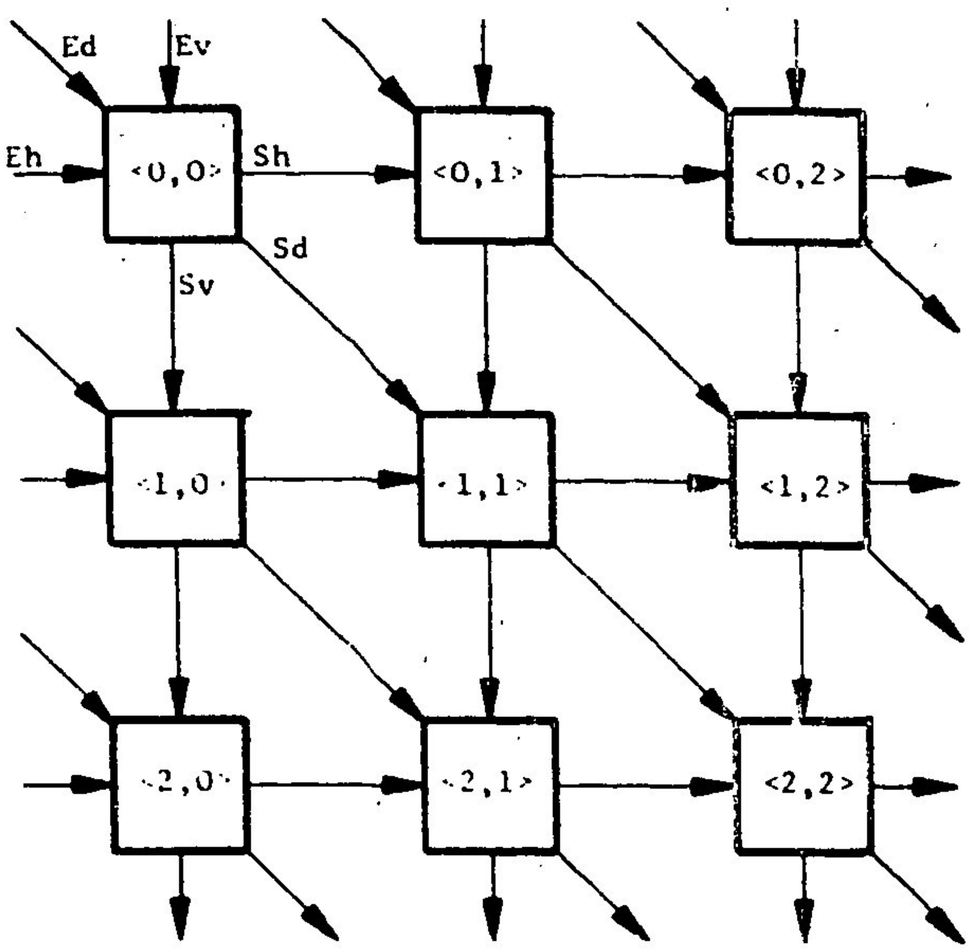
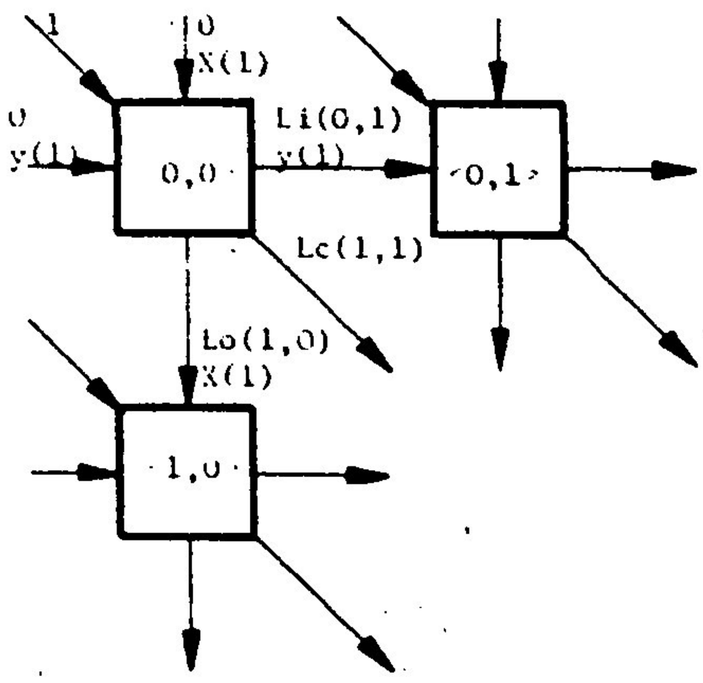
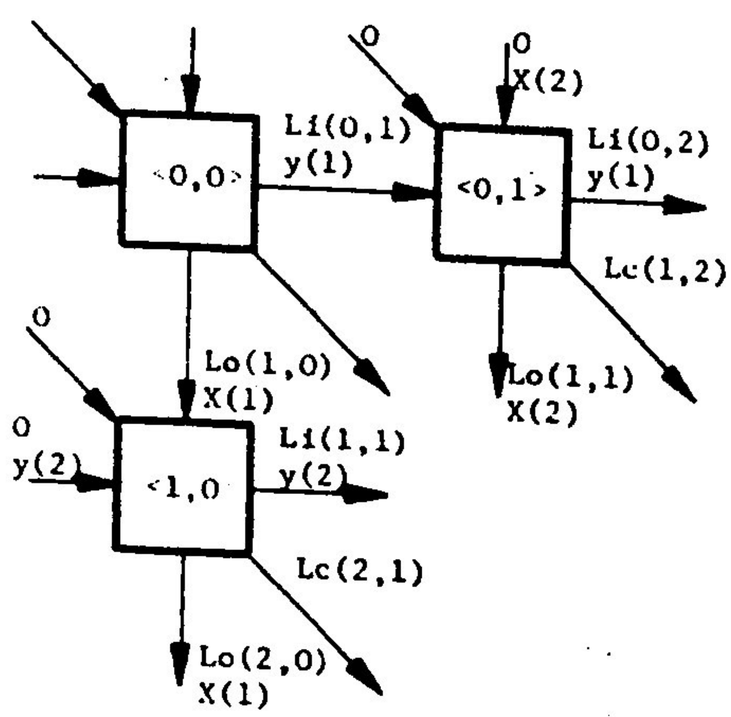


Figure 6 : structure du réseau



ETAPE 0



ETAPE 1

Figure 7 : Fonctionnement du réseau

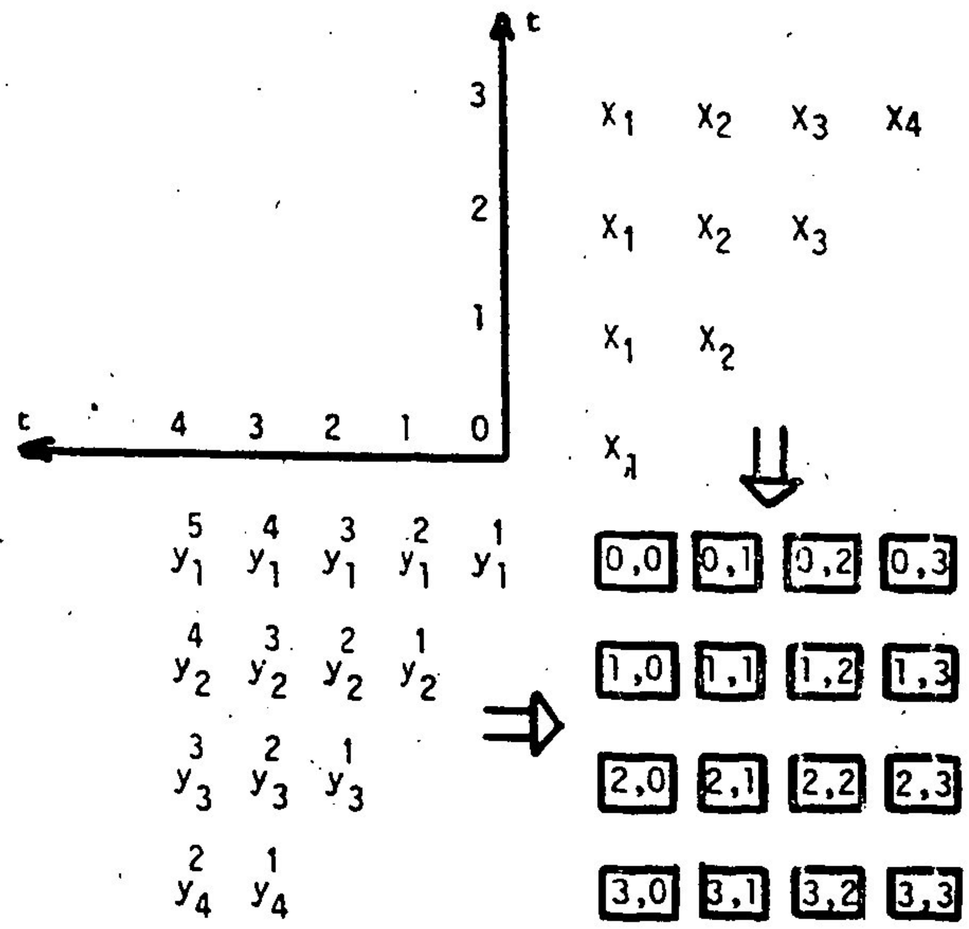


Figure 8 : fonctionnement du réseau en mode pipeline sur les mots

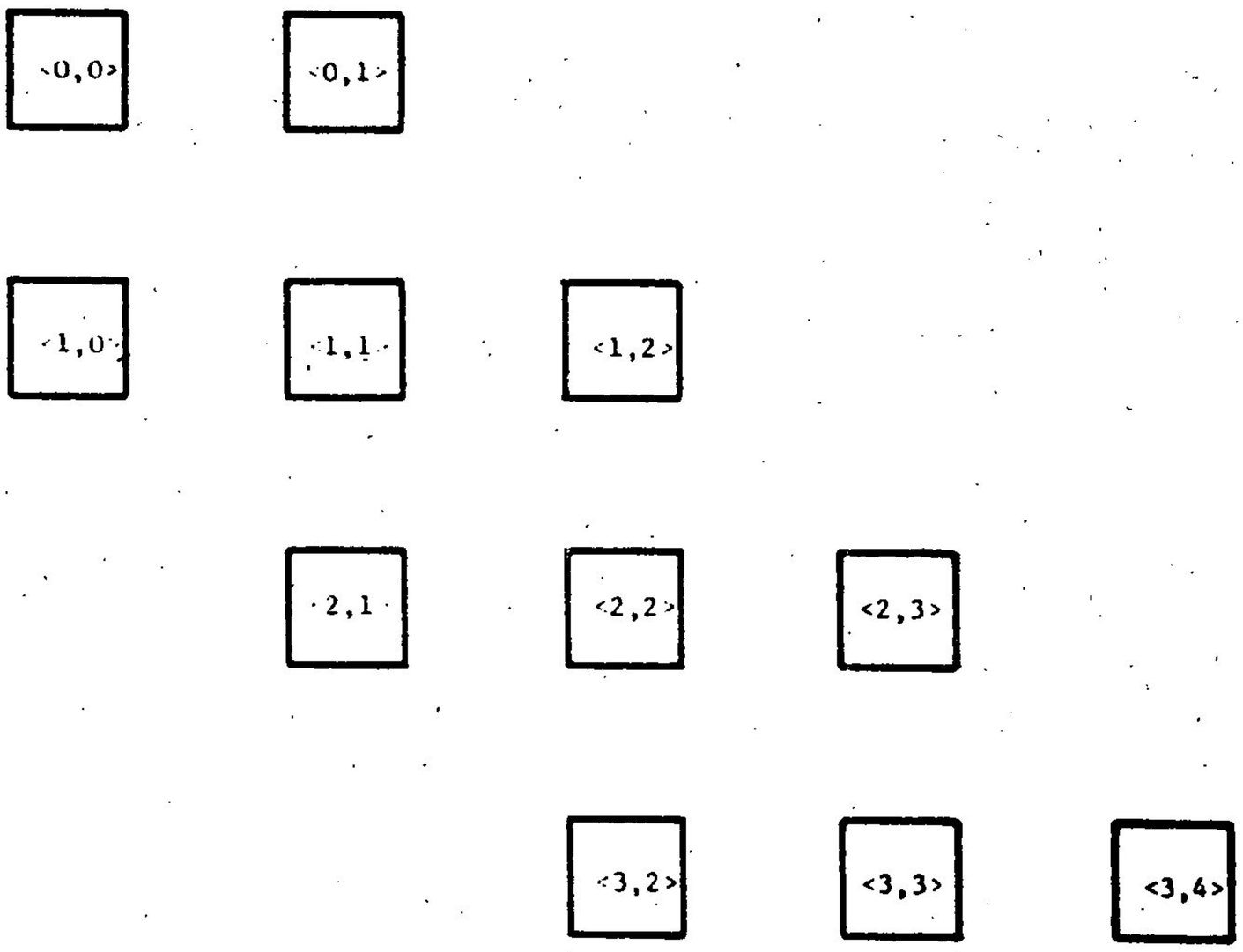


Figure 9 : Forme du réseau pour M=3 et delta=1

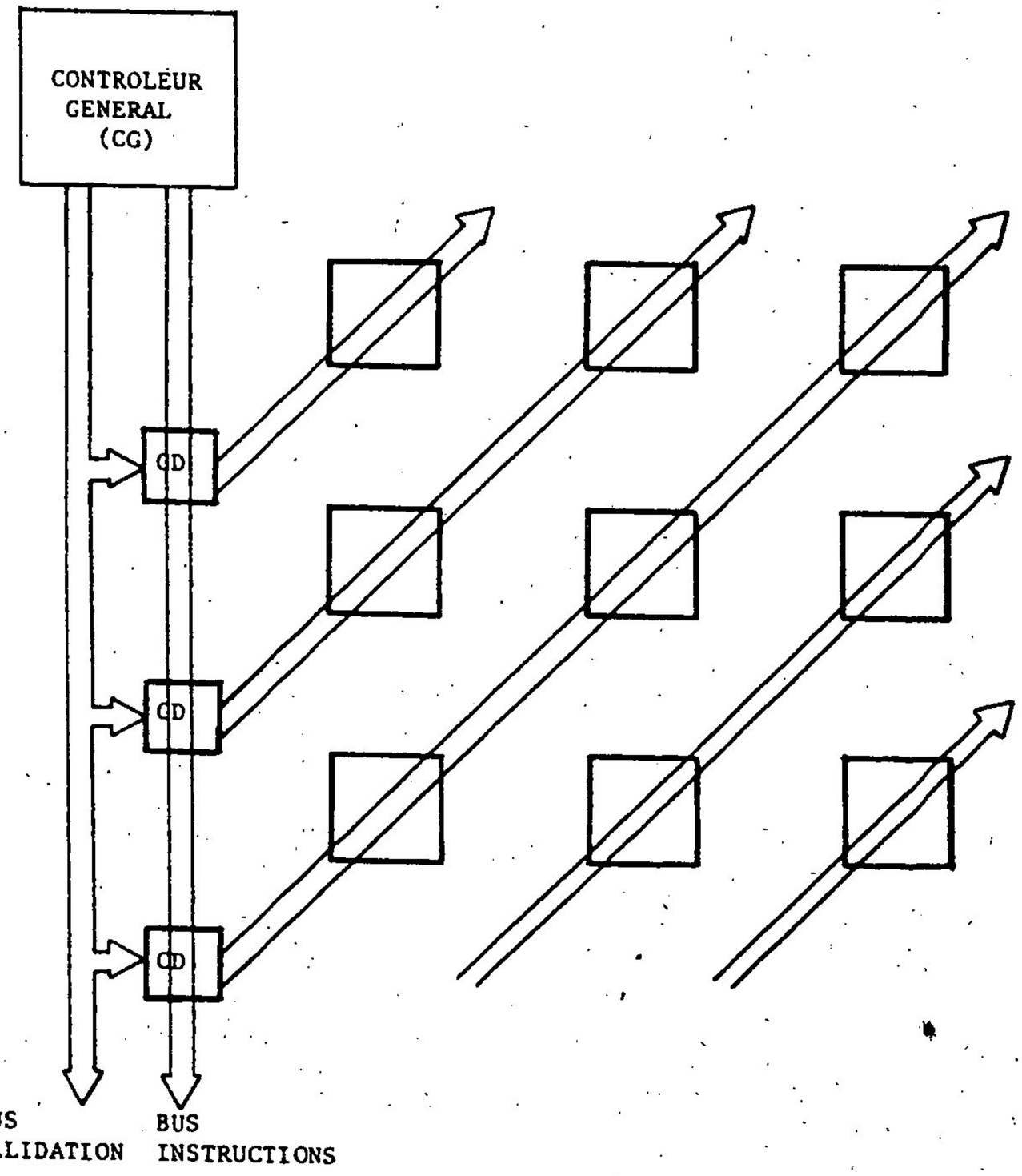


Figure 10 : Organisation du contrôle de la machine

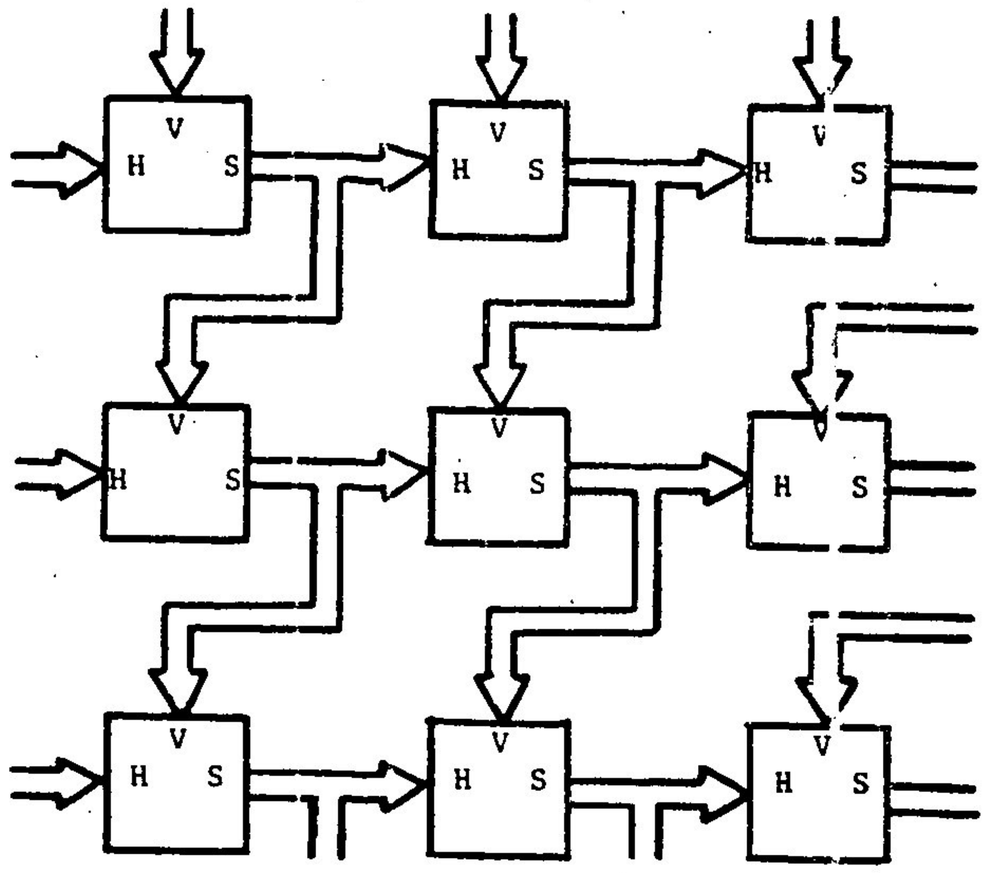


Figure 11 : Connexions interprocesseurs

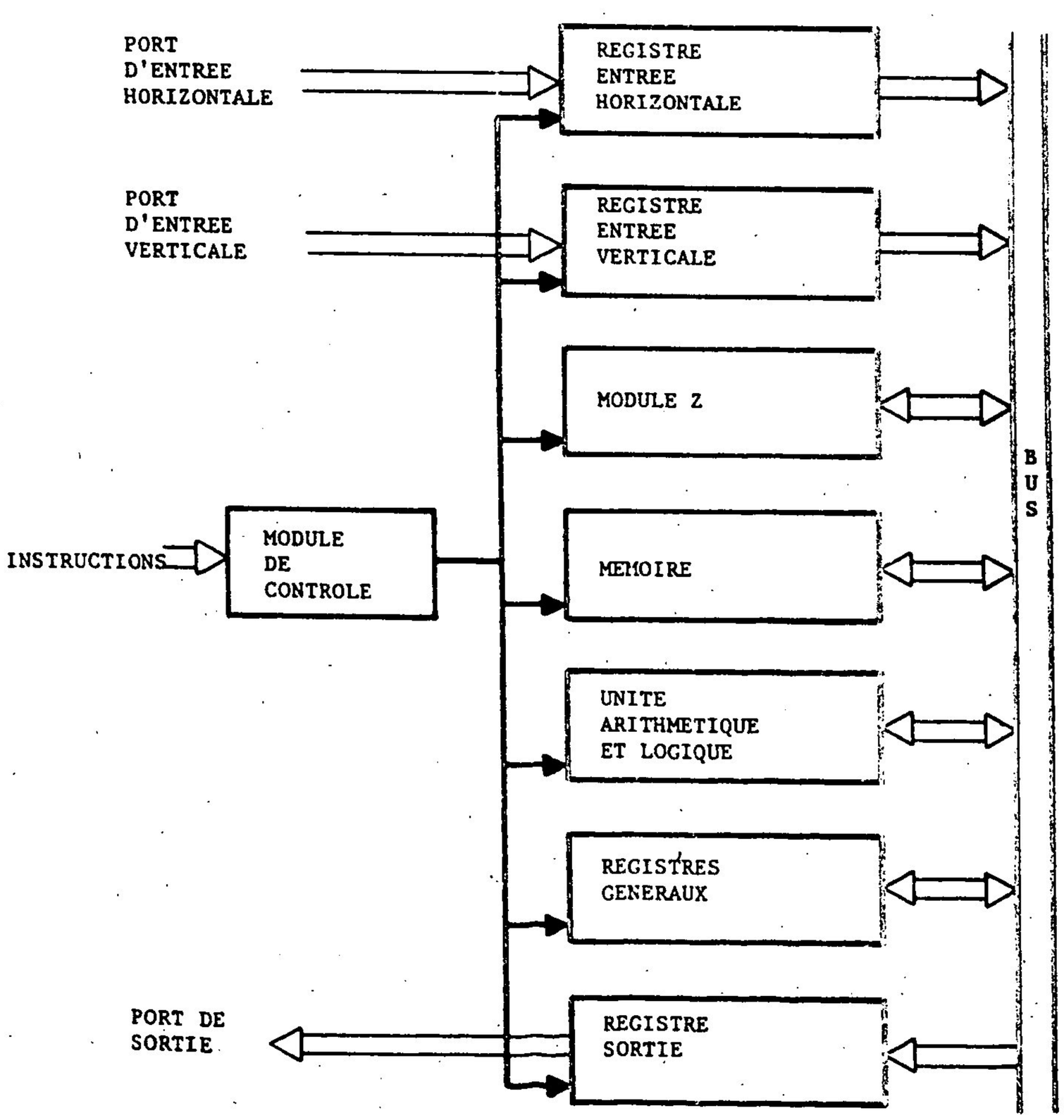


Figure 12 : Architecture d'un processeur

Liste des Publications Internes IRISA

PI 150 **Construction automatique et évaluation d'un graphe d'«implication» issu de données binaires, dans le cadre de la didactique des mathématiques**
H. Rostam , 112 pages ; Juin 1981

PI 151 **Réalisation d'un outil d'évaluation de mécanismes de détection de pannes]-]Projet Pilote SURF**
B. Decouty, G. Michel, C. Wagner, Y. Crouzet , 59 pages ; Juillet 1981

PI 152 **Règle maximale**
J. Pellaumail , 18 pages ; Septembre 1981

PI 153 **Corrélation partielle dans le cas « qualitatif »**
I.C. Lerman , 125 pages ; Octobre 1981

PI 154 **Stability analysis of adt ptively controlled not-necessarily minimum phase systems with disturbances**
Cl. Samson , 40 pages ; Octobre 1981

PI 155 **Analyses d'opinions d'instituteurs à l'égard de l'appropriation des nombres naturels par les élèves de cycle préparatoire**
R. Gras , 37 pages ; Octobre 1981

PI 156 **Récursion induction principle revisited**
G. Boudol, L. Kott , 49 pages ; Décembre 1981

PI 157 **Loi d'une variable aléatoire à valeur R^+ réalisant le minimum des moments d'ordre supérieur à deux lorsque les deux premiers sont fixés**
M.Kowalowka, R. Marie , 8 pages ; Décembre 1981

PI 158 **Réalisations stochastiques de signaux non stationnaires, et identification sur un seul échantillon**
A. Benveniste J.J. Fuchs , 33 pages ; Mars 1982

PI 159 **Méthode d'interprétation d'une classification hiérarchique d'attributs-modalites pour l'«explication» d'une variable ; application à la recherche de seuil critique de la tension artérielle systolique et des indicateurs de risque cardiovasculaire**
B. Tallur , 34 pages ; Janvier 1982

PI 160 **Probabilite stationnaire d'un réseau de files d'attente multiclasse à serveur central et à routages dépendant de l'état**
L.M. Le Ny , 18 pages ; Janvier 1982

PI 161 **Détection séqentielle de changements brusques des caractéristiques spectrales d'un signal numérique**
M. Basseville, A. Benveniste , pages ; Mars 1982

PI 162 **Actes regroupés des journées de Classification de Toulouse (Mai 1980), et de Nancy (Juin 1981)**
I.C. Lerman , 304 pages ;

PI 163 **Modélisation et Identification des caractéristiques d'une structure vibratoire : un problème de réalisation stochastique d'un grand système non stationnaire**
M. Prévosto, A. Benveniste, B. Barnouin , 46 pages ; Mars 182

PI 164 **An enlarged definition and complete axiomatization of observational congruence of finite processes**
Ph. Darondeau , 45 pages ; Avril 1982

PI 165 **Accès vidéotex à une banque de données médicales**
A. Chauffaut, M. Dragone, R. Rivoire, J.M. Roger ; 25 pages ; Mai 1982

PI 166 **Comparaison de groupes de variables définies sur le même ensemble d'individus**
B. Escofier, J. Pages , 115 pages ; Mai 1982

PI 167 **Transport en circuits virtuels internes sur réseau local et connexion Transpac**
M. Tournois, R. Trépos , 90 pages ; Mai 1982

PI 168 **Impact de l'intégration sur le traitement automatique de la parole**
P. Quinton , 14 pages ; Mai 1982

PI 169 **A systolic algorithm for connected word recognition**
J.P. Banâtre, P. Frison, P. Quinton , 13 pages ; Mai 1982

PI 170 **A network for the detection of words in continuous speech**
J.P. Banâtre, P. Frison, P. Quinton , 24 pages ; Mai 1982

PI 171 **Le langage ADA : Etude bibliographique**
J. André, Y. Jégou, M. Raynal , 12 pages ; Juin 1982

PI 172 **Comparaison de groupes de variables : 2ème partie : un exemple d'application**
B. Escofier, J. Pajès , 37 pages ; Juillet 1982

PI 173 **Unfold-fold program transformations**
L. Kott , 29pages ; Juillet 1982

PI 174 **Remarques sur les langages de parenthèses**
J.M. Autebert, J. Beauquier, L. Boasson, G. Senizergues , 20 pages ; Juillet 1982

PI 175 **Langages de parenthèses, langages N.T.S. et homomorphismes inverses**
J.M. Autebert, L. Boasson, G. Senizergues , 26 pages ; Juillet 1982

PI 176 **Tris pour machines synchrones ou Baudet Stevenson revisited**
R. Rannou , 26 pages ; Juillet 1982

PI 177 **Un nouvel algorithme de classification hiérarchique des éléments constitutifs de tableau de contingence basé sur la corrélation**
B. Tallur , Juillet 1982 ;

PI 178 **Programmes d'analyse des re'sultats d'une classification automatique**
I.C. Lerman et collaborateurs , 79 pages ; Septembre 1982

PI 179 **Attitude à l'égard des mathématiques des élèves de sixième**
J.Degouys, R. Gras, M. Postic , 29 pages ; Septembre 1982

PI 180 **Traitements de textes et manipulations de documents : bibliographie analytique**
J. André , 20 pages ; Septembre 1982

PI 181 **Algorithme assurant l'insertion dynamique d'un processeur autour d'un réseau à diffusion et garantissant la cohérence d'un système de numérotation des paquets global et réparti**
Annick Le Coz, Hervé Le Goff, Michel Ollivier , 31 pages ; Octobre 1982

PI 182 **Interprétation non linéaire d'un coefficient d'association entre modalités d'une juxtaposition de tables de contingence**
Israël César Lerman , 34 pages ; Novembre 1982

PI 183 **L'IRISA vu à travers les stages effectués par ses étudiants de DEA (1è année de thèse)**
Daniel Herman , 41 pages ; Novembre 1982

PI 184 **Commande non linéaire robuste des robots manipulateurs**
Claude Samson , 52 pages ; Janvier 1983

PI 185 **Dialogue et représentation des informations dans un système de messagerie intelligent**
Philippe Besnard, René Quiniou, Patrice Quinton, Patrick Saint-Dizier, Jacques Siroux, Laurent Trilling , 45 pages ; Janvier 1983

PI 186 **Analyse classificatoire d'une correspondance multiple ; typologie et régression**
I.C. Lerman , 54 pages ; Janvier 1983

PI 187 **Estimation de mouvement dans une sequence d'images de télévision en vue d'un codage avec compensation de mouvement**
Claude Labit , 132 pages ; Janvier 1983

PI 188 **Conception et réalisation d'un logiciel de saisie et restitution de cartes élémentaires**
Eric Sécher , 45 pages ; Janvier 1983

PI 189 **Etude comparative d'algorithmes pour l'amélioration de dessins au trait sur surfaces point par point**
M.A. ROY , 96 pages ; Janvier 1983

PI 190 **Généralisation de l'analyse des correspondances à la comparaison de tableaux de fréquence**
Brigitte Escofier , 35 pages ; Mars 1983

PI 191 **Association entre variables qualitatives ordinales «nettes» ou «floues»**
Israël-César Lerman , 42 pages ; Mars 1983

PI 192 **Un processeur intégré pour la reconnaissance de la parole**
Patrice Frison , 80 pages ; Mars 1983

PI 193

PI 194 **Régime stationnaire pour une file M/H/1 avec impatience**
Raymond Marie et Jean Pellaumail , 8 pages ; Mars 1983

PI 195 **SIGNAL : un langage pour le traitement du signal**
Paul Le Guernic, Albert Benveniste, Thierry Gautier , 49 pages ; Mars 1983

PI 196 **Algorithmes systoliques : de la théorie à la pratique**
Françoise André, Patrice Frison, Patrice Quinton , 19 pages ; Mars 1983

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique