Réseaux systoliques pour la résolution de problèmes linéaires

Lamine Melkemi

THESE

présentée à

l'UNIVERSITE SCIENTIFIQUE, TECHNOLOGIQUE ET MEDICALE DE GRENOBLE

pour obtenir le grade de
DOCTEUR DE 3ème CYCLE
«Mathématiques appliquées»

par

MELKEMI Lamine

RESEAUX SYSTOLIQUES POUR LA RESOLUTION DE PROBLEMES LINEAIRES.

Thèse soutenue le 28 avril 1986 devant la commission d'examen.

F. ROBERT	Président
J. DELLA-DORA	Examinateurs
J.M. LABORDE	
P. QUINTON	
Y. ROBERT	
M. TCHUENTE	

UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

Année universitaire 1982-1983

Président de l'Université : M. TANCHE

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

(RANG A)

SAUF ENSEIGNANTS EN MEDECINE ET PHARMACIE

PROFESSEURS DE 1ère CLASSE

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

.../...

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

PROFESSEURS DE 2ème CLASSE

ADIBA Michel	Mathématiques pures
ARMAND Gilbert	Géographie

.../...

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

A mes parents,

A mes frères et soeurs,

et à tous mes amis.

Je tiens à exprimer mes remerciements les plus sincères à Monsieur le Professeur Maurice TCHUENTE pour m'avoir guidé tout au long de ce travail. Je suis trés heureux de lui témoigner toute ma reconnaissance pour son aide constante. Ses grandes qualités scientifiques et humaines m'ont permis de mener à bien ce travail dans les meilleures conditions.

Mes vifs remerciements vont également à:

Monsieur le Professeur François ROBERT pour l'honneur qu'il me fait en présidant le jury de cette thèse.

Monsieur Jean DELLA DORA, Directeur du laboratoire TIM3 pour avoir accepté de faire partie du jury.

Monsieur Jean Marie LABORDE, Directeur de recherches du laboratoire LSD pour avoir accepté de siéger dans le jury.

Monsieur Patrice QUINTON, Directeur de recherches de l'IRISA Rennes pour avoir accepté de se déplacer pour participer à ce jury.

Monsieur Yves ROBERT, Chargé de recherches du laboratoire TIM3 pour les nombreuses discussions que nous avons pues avoir ensemble et qui m'ont été trés utiles, et pour avoir accepté de faire partie du jury.

De même, Je suis particulièrement heureux de pouvoir témoigner mes amitiés à tous les membres de l'équipe algorithmique mathématique, et notamment au groupe algorithmique parallèle. J'ai pu apprécier les qualités de sympathie et de bonne humeur qui y règnent.

Merci également à tous mes amis avec qui j'ai partagé tant de joies pendant mon séjour en France.

Je remercie toute l'équipe du service de reprographie pour l'exellente qualité de leur travail.

SOMMAIRE

INTRODUCTION GENERALE

1) Introduction

La dernière décennie a vu naître et se développer de nombreuses recherches sur le parallélisme. C'est ainsi que sont nées les machines multi-processeurs de type S.I.M.D et M.I.M.D (Flynn[11]) destinées à apporter des solutions à la saturation quasi-totale des ordinateurs séquentiels pour la résolution des problèmes du calcul scientifique (hydrodynamique, météorologie,...).

Aujourd'hui, avec le developpement des circuits VLSI (Very Large Scale Integration), on entrevoit dans un futur proche, l'intégration de plusieurs centaines de milliers de transistors sur une même puce de silicium. Néanmoins, la conception des architectures parallèles se heurte à des problèmes importants en ce qui concerne l'interconnexion entre les processeurs et l'interface avec l'ordinateur hôte (Mead et Conway[30]).

Les réseaux systoliques, qui constituent le thème de cette thèse, sont une réponse au problème de la conception des circuits VLSI destinés à être utilisés comme périphériques d'une structure hôte.

Les réseaux systoliques ont été introduits en 1978 par Kung et Leiserson[24] et sont des architectures parallèles spécialisées, composées de cellules élémentaires interconnectées de manière locale et régulière. Dans un tel réseau, les cellules opèrent de manière synchrone; à l'instant t, chaque cellule reçoit les données des cellules qui lui sont voisines en entrée et, après avoir effectué des transformations élémentaires, elle les envoie aux cellules qui lui sont voisines en sortie.

Cette approche a un double intérêt:

- D'abord, la conception d'un réseau systolique est beaucoup plus aisée que la conception d'un circuit VLSI général. En effet, un tel réseau est généralement composé de cellules élémentaires en un grand nombre mais n'appartenant qu'à un, deux ou trois types. Le concepteur peut donc, dans un premier temps, concentrer ses efforts à l'optimisation des cellules types; la deuxième phase qui consiste à dupliquer et à interconnecter les copies des cellules types est facilitée par le caractère planaire et modulaire de l'interconnexion.

- Ensuite comme Kung[21] l'a fait remarquer, les performances réelles qu'une architecture de type S.I.M.D peut atteindre, peuvent être très décevantes par rapport à la capacité de calcul de la machine, à cause du grand nombre d'accès mémoires éventuellement nécessaires. Le modèle

systolique résoud ce problème; en effet, dans un tel réseau, seules les cellules frontières peuvent communiquer avec la mémoire centrale.

Les besoins en matériel requis par les réseaux systoliques sont donc modestes; par ailleurs ils se sont révélés très adaptés à la théorie du calcul.

En effet, au cours de ces dernières années, plusieurs algorithmes systoliques, dont les solutions sont optimales (en un sens que nous précisons plus loin), ont été proposés dans la littérature pour la résolution de différents types de problèmes: le calcul matriciel ([6], [13], [15], [24], [32]), le traitement du signal et de l'image[1], [20], la reconnaissance de languages ([2], [12], [42],[44]), le traitement de graphes[14], [32], [45] etc...

De fait, les fondéments théoriques du modèle se trouvent déjà dans les travaux sur les réseaux d'automates menés dans les années 60 [16]; en particulier, il a été établi que ce modèle est adapté au calcul en temps réel[3],[7]...

Enfin, il est à noter que les réseaux systoliques ont fait naître de nombreux thèmes de recherches. Citons notamment les deux sujets suivants:

A) Méthodologies

Souvent, il n'est pas aisé de construire pour un problème, sans outil autre que l'intuition ou l'expérience, un réseau solution. Ceci a amené plusieurs auteurs à proposer, pour ce but, des méthodologies permettant d'aboutir, automatiquement, au réseau systolique désiré: Lam et Mostow[26], Leiserson et Saxe[28], Moldovan[39], Quinton[41]...

B) Problèmes de pannes

L'expérience acquise dans les réseaux systoliques a montré qu'on ne pourrait éviter la présence de cellules défaillantes sur une puce; pour remédier à ces défauts, l'étude des réseaux robustes s'avère nécessaire.
Différentes approches générales et/ou spécialisées ont été suggérées afin de rendre faisable la production de tels réseaux: Kung et Lam[23], Leighton et Leiserson[27], Varman et Fussel[46]...

D'autres sujets méritent aussi d'être cités: la validation des réseaux systoliques [31], la complexité ([29], [33])...

Exemple d'un réseau systolique

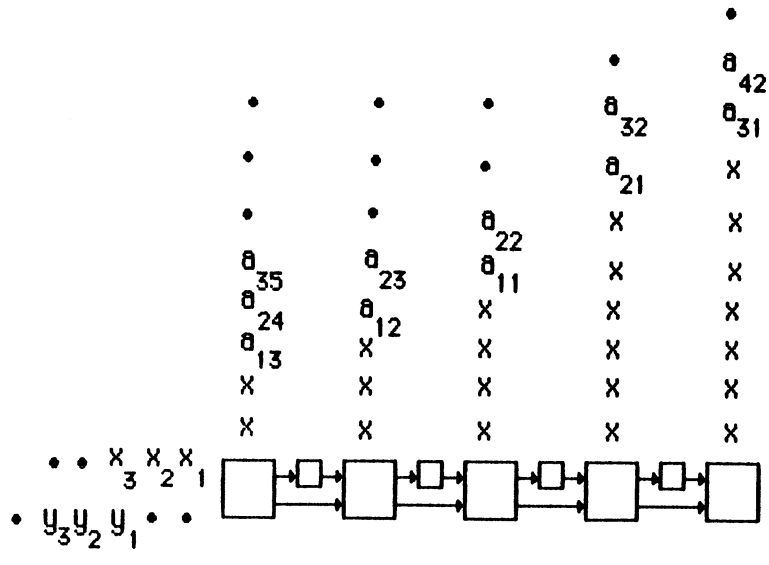
La figure ci dessous illustre un réseau systolique pour le calcul du produit y=A.x d'une matrice carrée bande de largeur-bande w=5 par un vecteur x. On remarque aisément que:

-les coefficients y_i et x_i circulent dans une même direction.

-les coefficients a_{ij} sont présentés dans la mémoire de manière à ce que la taille du réseau ne dépende que de la largeur-bande de la matrice A.

-sur le flux des x_i, il y a un registre de retard entre deux cellules consécutives; en conséquense, chaque variable x_i met deux temps de cycle pour aller d'une cellule à sa voisine. Par contre, chaque coefficient y_i se trouvant dans une cellule est directement transmis, le temps de cycle d'après, à la cellule voisine.

On vérifie aisément que, sous ces trois propriétés, le réseau réalise le produit y = A.x.



2) Plan de la thèse:

Dans cette thèse, nous commençons par étudier le calcul du produit matriciel sur la classe des architectures systoliques (chapitres 1 et 2). Ensuite, nous présentons des réseaux systoliques pour la détection en temps linéaire des répétitions dans une chaîne de caractères (chapitre 3).

Le premier chapitre comprend deux articles, et est consacré à l'étude de la complexité en temps du produit matriciel sur la classe des réseaux systoliques rectangulaires.

Nous montrons que la conception d'un algorithme pour le calcul du produit C de deux matrices carrées d'ordre n sur un réseau rectangulaire est équivalente à la formulation combinatoire qui consiste à affecter les coefficients c_{ij} dans le plan de manière à ce que:

(i) si c_{ij} et c_{hq} appartiennent à un même segment horizontal alors i = h
(ii) si c_{ij} et c_{hq} appartiennent à un même segment horizontal alors j = q

Dans le premier article, nous démontrons que le temps minimum nécessaire pour le calcul de C sur la classe des réseaux rectangulaires est égal à 3n - 2. Le deuxième article étend ce résultat au cas du produit de deux matrices rectangulaires. Si T(n,m,p) désigne le temps minimum nécessaire pour le calcul du produit C = A.B de deux matrices rectangulaires A et B d'ordres respectivement nxm et mxq alors:

$$T(n,m,p) = \begin{cases} m+2p+2q-2 & \text{si } n-p = 3q \leq 3(p-1)/2 \\ m+2p+2q-2 & \text{si } n-p = 3q-1 \leq 3(p-1)/2 \\ m+2p+2q-1 & \text{si } n-p = 3q+1 \leq 3(p-1)/2 \\ m+n+\lceil (p-1)/2 \rceil & \text{sinon} \end{cases}$$

Le deuxième chapitre se compose également de deux articles: le premier se base sur la formulation combinatoire ainsi citée. En se donnant un réseau rectangulaire de taille nxm, m≤n, nous proposons des approches variées permettant de programmer, sur ce réseau, des algorithmes performants.

Dans le deuxième article, nous étudions le compromis entre d'une part le temps et d'autre part le nombre d'accès-mémoires sur la classe des

réseaux programmables. Nous montrons que le nombre d'accès-mémoires est l'une des causes de croissance du temps de calcul.

Au troisième chapitre, nous proposons un réseau bidimensionnel permettant de détecter en temps linéaire tous les carrés dans un mot. Puis nous montrons un lemme sur le chevauchement des carrés sur une chaine de caractères; ceci va nous permettre de construire un réseau unidimensionnel pour la reconnaissance en temps linéaire du language des mots sans carrés.

CHAPITRE 1

Complexité en temps du calcul du produit matriciel sur la classe des réseaux rectangulaires

title: **COMPLEXITY OF MATRIX PRODUCT ON A CLASS OF ORTHOGONALLY CONNECTED SYSTOLIC ARRAYS**

authors: Lamine MELKEMI and Maurice TCHUENTE
CNRS/INPG Laboratoire TIM3
BP 68 38402 Saint Martin d'hères cédex France

soumis à IEEE Transactions on computers

ABSTRACT

This paper studies the time complexity of the parallel computation of the product C = A.B of two dense square matrices A,B of order n, on a class of rectangular orthogonally connected systolic arrays which are the two-dimensional extensions of the classical pipeline scheme. Such arrays are composed of multiply-add cells without local memory, and, as C is computed, the coefficients c_{ij} move vertically, whereas a_{ik} and b_{kj} move horizontally in opposite directions. We first introduce a combinatorial formulation of the problem. Then we show that, if the cycle-time of a multiply-add cell is taken as time unit, and if T(p,m) denotes the running time of an optimal algorithm associated with an array of size pxm, then $\text{Min}_{p,m}\ T(p,m) = 3n - 2$, and the minimum value of p.m for which this bound is tight is n.n (resp. n(n+1)) if n is odd (resp. even). When compared with the algorithms previously proposed for the class of arrays based on multiply-add cells without local memory, the solutions proposed here appear to be the best because they are the only ones which run in time $T \leq 3n - 2$ on a network of area $S \leq n(n+1)$.

Index terms: time-complexity, parallel computation, matrix multiplication, systolic array, multiply-add cell, optimal algorithm, combinatorial formulation.

1 - INTRODUCTION

In the 1970's, LSI technology allowed tens of thousands of devices to fit on a single chip. Today, with VLSI technology, the complexity and number of components that can be integrated on a small silicon area is rapidly increasing, and will be at least one or two orders of magnitude bigger in the next decade. As a consequence, many traditional computer design concepts are no longer justified technologically, and it is now possible to formulate a great deal of new challenging problems.

A distinct characteristic of VLSI chips is that data communication dominates the cost and performance of computing devices, whereas in traditional processes, it is assumed that memories and processors are the dominate factors. As a consequence, when trying to take advantage of the high degree of parallelism inherent to VLSI devices, one must pay attention to I/O requirements.

Systolic arrays, as defined by Kung and Leiserson [5], are a useful tool for special purpose VLSI system design. Typically a systolic network of processors is implemented with only a few types of elementary cells interconnected in a simple, local and regular way. This makes them economical to manufacture and repair. Moreover, in a context where the network consists of single chip microprocessors used in groups of tens or hundreds [1], the system can be enlarged to accomodate more variables by simply adding more chips. The high performance achieved by systolic arrays, is obtained by extensive use of multiprocessing and pipelining. It is important to note that, contrary to the classical pipeline scheme where pipelining is applied only to result-variables, (see the variables c_{ij} in fig. 1), the two-dimensional systolic algorithms also pipeline data-variables (see the variables a_{ik} , b_{kj} in fig. 2). This pipelining of data-variables enables multiple computations to be performed for every I/O access. As a consequence, as explained in [3], systolic designs lead to structures which can solve compute-bound problems without increasing unreasonably I/O requirements.

2 - PREVIOUS RESULTS

In this paper, we are interested in the computation of the product C = A.B of two square matrices A,B of order n >= 2 on two-dimensional

systolic arrays which generalize the classical pipeline scheme. Such arrays are composed of multiply-add cells without local memory (see fig. 1). During the computation of the product C = A.B, the coefficients c_{ij} move vertically, whereas the coefficients a_{ik} and b_{kj} move horizontally in opposite directions; c_{ij} is initialized to zero and as it travels through the array it performs the accumulations $c_{ij} := c_{ij} + a_{ik}.b_{kj}$, for k = 1,..., n.

Hereafter, the cycle-time of a multiply-add cell is taken as time unit. The running time of a systolic algorithm is defined as the delay between the first input and the last output. On the other hand, the area of an array is defined as the number of cells.

In the case where A and B are band matrices of band-width w and w' respectively, Kung and Leiserson [5] have proposed an hexagonally connected array of area w.w' which works in time (3n-1) + min(w,w'). Subsequently, Weiser and Davis [16] have given an alternate solution of area w.w' and time n-1 + max(w,w').

Let us now consider the case of dense matrices, i.e. square matrices A ,B of order n where all the entries may be non zero. Since a dense matrix of order n is a band matrix of band-width 2n-1, the solution of [5] is of area $n^2 - n + 1$ and time 5n - 2, whereas that of [16] is of area $3n^2 - 3n + 1$ and time 3n - 2. On the other hand, on orthogonally connected networks, the solution proposed by Kung and Lehman [4] requires an area n^2 and runs in time 4n - 2. More recently, Robert and Tchuente [15] have shown that a divide-and-conquer approach can be used to derive from [4], a solution of area n^2 and time $T \leq 7n/2$.

In the case of orthogonally connected networks based on multiply-add cells with local memory, it is possible to define algorithms which proceed as follows: the computation of any coefficient c_{ij} starts with a first phase during which the accumulations $c_{ij} := c_{ij} + a_{ik}.b_{kj}$, k = 1, ... ,n, are performed in a cell of the array (see fig. 3); in the second phase, the coeficient c_{ij} is output from the array. These arrays have been used by Guibas, Kung and Thompson [2], Preparata and Vuillemin [13] to define algorithms of area n^2 and time 4n - 2; the coefficients c_{ij} are computed after 3n - 2 steps, but one needs n more steps to output them from the array. Recently, the authors have established I/O-time trade-offs for matrix product on such structures [10].

Let us note that the methodologies proposed by several authors ([6] , [7], [11] , [12] , [14]), for the automatic synthesis of systolic arrays, can be used to derive a great variety of solutions for matrix product.

In this paper, we restrict ourselves to arrays composed of multiply-add cells without local memory. As a consequence, we do not consider orthogonally connected designs where one of the matrices may be permanently stored in the array. We introduce a combinatorial formulation which enables us to derive complexity results as well as optimal algorithms.

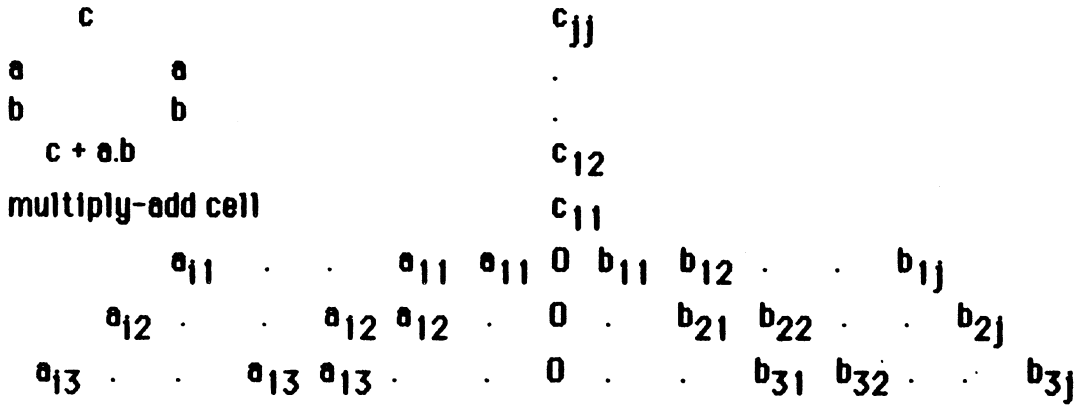


Fig. 1: n = 3, m = 1, **(a partial view of the algorithm)**

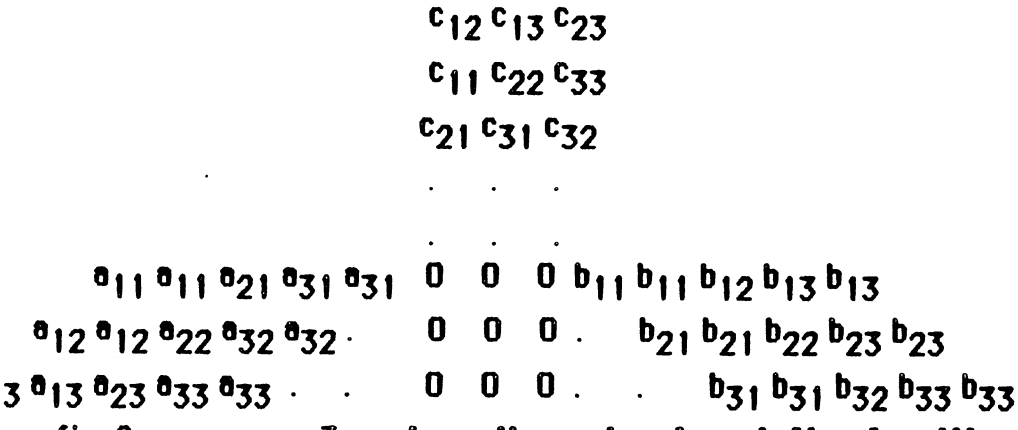


fig. 2: n = m = p = 3; **a two-dimensional systolic algorithm**

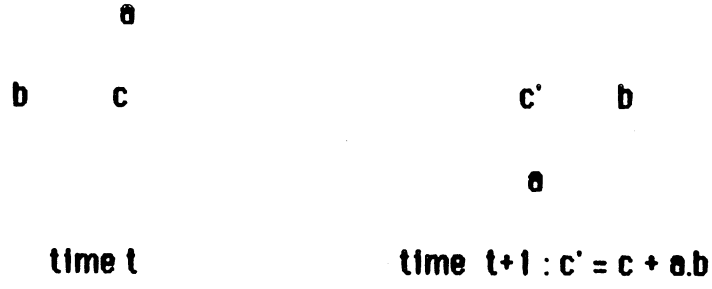


Fig. 3 multiply-add cell with local memory

3 - TIME-COMPLEXITY RESULTS

Let us consider a systolic algorithm which computes C = A.B on a rectangular orthogonally connected systolic array with p rows and m columns, and where c_{ij} moves vertically, a_{ik} moves from right to left and b_{kj} moves from left to right. If we apply this algorithm to the product $C^t = B^t.A^t$, then any coefficient c_{ij} moves vertically, a_{ik} moves from left to right and b_{kj} moves from right to left. Since the computation of C^t is

equivalent to the computation of C, we have obtained an a new algorithm with the same running time, and where the directions of a_{ik} and b_{kj} have been interchanged. As a consequence, when studying time-complexity results for the computation of the product C = A.B on orthogonally connected arrays where c_{ij} flows vertically and a_{ik} , b_{kj} flow horizontally in opposite directions, we can assume without any loss of generality, that a_{ik} moves from left to right and b_{kj} moves from right to left.

Hereafter, any array of size pxm is identified with a rectangle of order pxm contained in ZxZ and with edges parallel to the horizontal and vertical axis. On the other hand, any variable which moves according to a direction d belonging to { (1,0), (-1,0), (0,-1)), and which is input at time t to a cell at position (x,y), is represented by assiging that variable to the point (x,y) - d(t+1). For instance, in figure 2, c_{21}, c_{11} and c_{12} are input to the cell at the upper left corner respectively at times t = 0,1,2.

3.1 CLASSICAL PIPELINE SCHEMES (m = 1)

For m = 1, the array is linear, and this corresponds to the classical pipeline scheme.

If p >= n, then any c_{ij} can perform the n accumulations $c_{ij} := c_{ij} + a_{ik}.b_{kj}$, k = 1 , ... , n by going once through the array. It is easily verified that the design of an algorithm which runs in time T = n + H can be obtained as follows (see fig. 1):

<u>step 1</u> : assign the variables c_{ij} , $1 \leq i,j \leq n$ to the H lowest points above the array.

<u>step 2</u> : assign the variables a_{i1} , ... , a_{in} in this order from top to bottom, to the segment which is at the intersection between a line of slope +1 containing c_{ij} and the horizontal band containing the array.

<u>step 3</u> : assign the variables b_{1j} , ... , b_{nj} in this order from top to bottom, to the segment which is at the intersection between a line of slope -1 containing c_{ij} and the horizontal band containing the array.

If p < n (see fig. 4), we denote u the smallest integer greater than or equal to n/p. Clearly, any variable c_{ij} must go at least u times through the

array in order to perform the n accumulations $c_{ij} := c_{ij} + a_{ik} \cdot b_{kj}$, $k = 1, \ldots, n$. Let c_{ij}^{r} , $0 \leq r < u$, represent the variable c_{ij} when it has already gone r times through the array. Since c_{ij}^{r} must be output from the array before giving birth to c_{ij}^{r+1}, it follows that, as they flow downwards, during the execution of the algorithm, c_{ij}^{r} and c_{ij}^{r+1} cannot be simultaneously in the array. As a consequence, if at time t = 0 they are assigned to two point P = (x,y) and Q = (x',y'), then $y' - y >= p$. An algorithm which runs in time n + H can therefore be obtained as follows:

step 1 : assign the variables c_{ij}^{r} , $0 \leq r < u$, to the H lowest points above the array in such a way that if c_{ij}^{r} and c_{ij}^{r+1} are assigned respectively to P = (x,y) and Q = (x',y') then $y' - y >= p$.

step 2 : for $0 \leq r < u-1$ (resp. r = u), assign the variables $a_{rp+1}, \ldots, a_{r(p+1)}$ (resp. $a_{r(p+1)+1}, \ldots, a_{n}$) in this order to the segment which is at the intersection between the line of slope +1 containing c_{ij}^{r} and the horizontal band containing the array.

step 3 : for $0 \leq r < u-1$ (resp. r = u), assign the variables $b_{rp+1}, \ldots, b_{r(p+1)-1}$ (resp. $b_{r(p+1)+1}, \ldots, b_{n}$) in this order to the segment which is at the intersection between the line of slope -1 containing c_{ij}^{r} and the horizontal band containing the array.

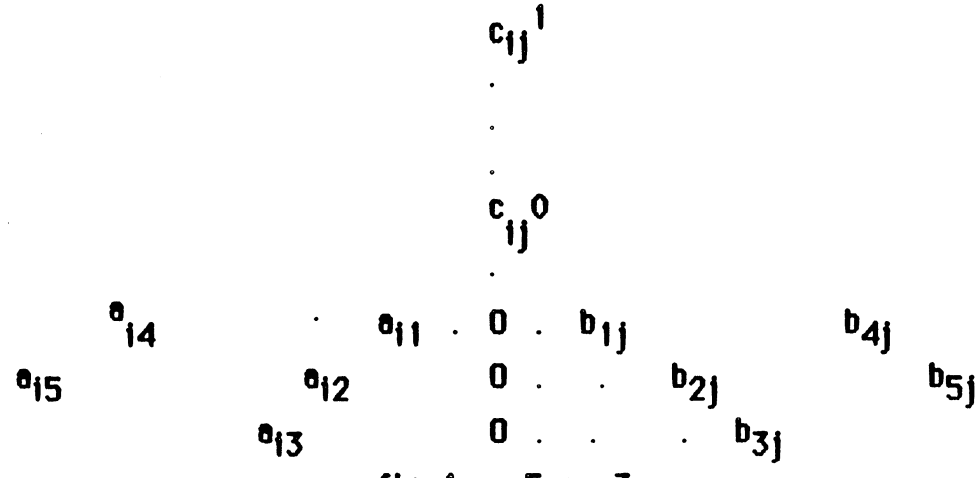


fig. 4 n = 5, p = 3

3 - TIME-COMPLEXITY RESULTS

Let us consider a systolic algorithm which computes C = A.B on a rectangular orthogonally connected systolic array with p rows and m columns, and where c_{ij} moves vertically, a_{ik} moves from right to left and b_{kj} moves from left to right. If we apply this algorithm to the product $C^t = B^t.A^t$, then any coefficient c_{ij} moves vertically, a_{ik} moves from left to right and b_{kj} moves from right to left. Since the computation of C^t is equivalent to the computation of C, we have obtained a new algorithm with the same running time, and where the directions of a_{ik} and b_{kj} have been interchanged. As a consequence, when studying time-complexity results for the computation of the product C = A.B on orthogonally connected arrays where c_{ij} flows vertically and a_{ik} , b_{kj} flow horizontally in opposite directions, we can assume without any loss of generality, that a_{ik} moves from left to right and b_{kj} moves from right to left.

Hereafter, any array of size pxm is identified with a rectangle of order pxm contained in $\mathbb{Z}x\mathbb{Z}$ and with edges parallel to the horizontal and vertical axis.

For any k, $1 \leq k \leq p$, we denote (see fig. 4,5)

- $\mathcal{H}$ the horizontal band of width p which contains the array
- $\mathcal{L}_k$ the subset of $\mathcal{H}$ consisting of the segments of slope +1 and length greater than or equal to k, which are at the left of the array
- $\mathcal{R}_k$ the subset of $\mathcal{H}$ consisting of the segments of slope -1 and length greater than or equal to k, which are at the right of the array
- $\mathcal{A}_k$ the set of intersection points between lines of slope +1 which contain a segment of $\mathcal{L}_k$, and lines of slope -1 which contain a segment of $\mathcal{R}_k$; it is easily seen that all points of $\mathcal{A}_k$ are above the array.

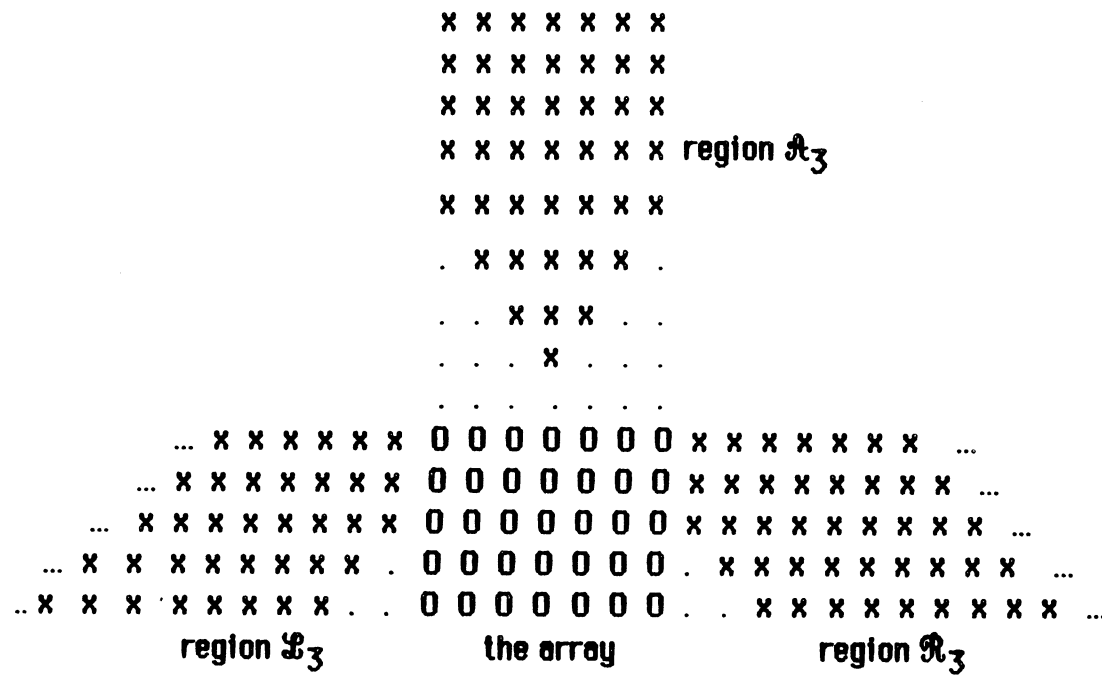


Fig. 4 $p = 5$, $m = 7$

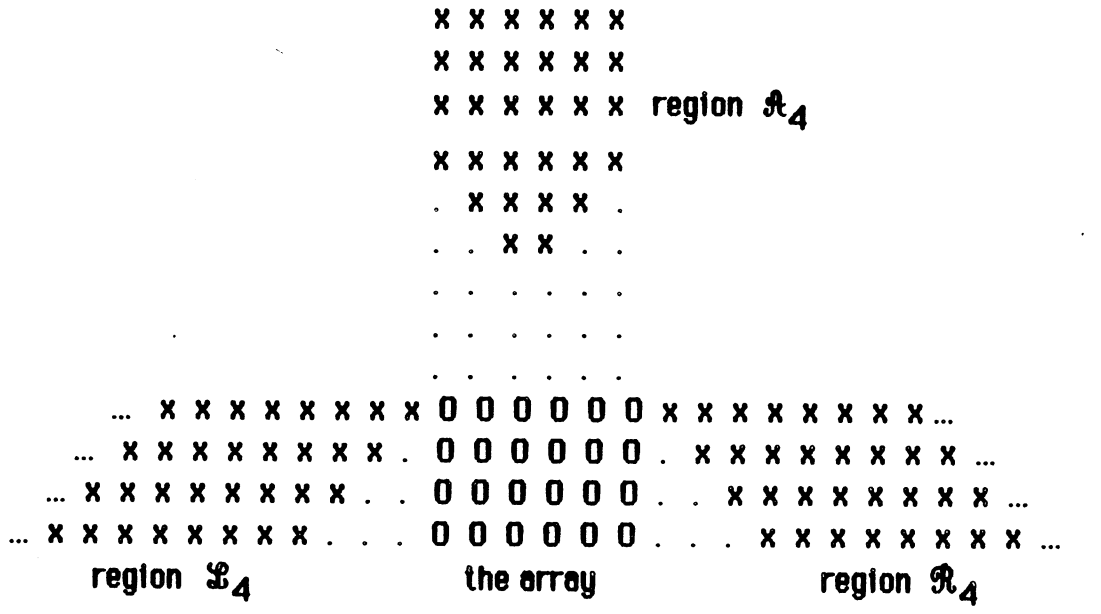


Fig. 5 $p = 4$, $m = 6$

On the other hand, in the presentation of a systolic algorithm, any variable which moves according to a direction Δ belonging to { (1,0), (-1,0), (0,-1), (0,1)}, and which is processed at time t, by the cell at position (x,y), is represented by assigning that variable to the point (x,y) - (t+1) Δ. For instance, in figure 2, the variables c_{21}, c_{11} and c_{12} are processed by the cell at the upper left corner respectively at times t = 2,3,4.

Proposition 1 (combinatorial formulation for p = n).

The designing of a systolic algorithm which computes the product C = A.B of two dense square matrices A,B of order n, on an array of size nxm, and where any c_{ij} is computed by going once through the array is equivalent to the combinatorial problem of assigning the variables c_{ij} , $1 \leq i,j \leq n$ to region $\mathcal{A}_n$ in such a way that :

(i) if c_{ij} and c_{hq} belong to a line of slope +1 then i = h
(ii) if c_{ij} and c_{hq} belong to a line of slope -1 then j = q

proof :

case 1 : m = 1 (see fig. 1)
For m = 1, the array is linear, and this corresponds to the classical pipeline scheme.

Since the array contains n rows, any c_{ij} can perform the n accumulations $c_{ij} := c_{ij} + a_{ik}.b_{kj}$, k = 1 , ... , n by going once through the array. It is easily verified that the design of an algorithm which runs in time $T = n + n^2$, can be obtained as follows (see fig. 1):

<u>step 1</u> : assign the variables c_{ij} , $1 \leq i,j \leq n$ to the n^2 lowest points above the array; since m = 1, it is easily seen that these points belong to region $\mathcal{A}_n$.

step 2 : assign the variables $a_{i1}, \ldots, a_{in}$ in this order from top to bottom, to the segment which is at the intersection between $\mathfrak{L}_n$ and a line of slope +1 containing c_{ij}.

step 3 : assign the variables $b_{1j}, \ldots, b_{nj}$ in this order from top to bottom, to the segment which is at the intersection between $\mathfrak{R}_n$ and a line of slope -1 containing c_{ij}.

case 2: $m > 1$

For $m > 1$, the variables c_{ij} which are above a column of the array, are computed in pipeline in this column. As a consequence, a systolic algorithm defined on the array is the union of pipeline schemes associated with the columns of the array. Since initially, all the variables are out of the array, it follows from case 1 that:

- any row $(a_{i1}, \ldots, a_{in})$ of A must be assigned to a segment of slope +1 contained in $\mathfrak{L}_n$
- any column $(b_{1j}, \ldots, b_{nj})$ of B must be assigned to a segment of slope -1 contained in $\mathfrak{R}_n$
- any variable c_{ij} must be assigned to a point of $\mathfrak{A}_n$ which is at the intersection between a line of slope +1 containing the ith row of A and a line of slope -1 containing the jth column of B .

(i): Let us assume that c_{ij} and c_{hq} belong to a line L of slope +1, and let $(a_{r1}, \ldots, a_{rn})$ be the row of A which has been assigned to L. It is easily seen that both c_{ij} and c_{hq} perform accumulations corresponding to $(a_{r1}, \ldots, a_{rn})$, hence $i = h = r$.
(ii): similar to (i) □

Hereafter, we shall make an extensive use of the following result:

Proposition 2

We can associate with any systolic algorithm defined on a network of size pxm such that $p \geq 1$, $n = up + w$, $1 \leq w \leq n$, an assignment of variables c^r_{ij}, $0 \leq r \leq u$, $1 \leq i,j \leq n$ to region $\mathcal{A}_w$ in such a way that:

(i) if c^r_{ij} and c^s_{ij}, $r < s$, are assigned respectively to $P = (x,y)$ and $P' = (x',y')$ then $y' - y \geq p$.

(ii) if c^r_{ij} and c^s_{hq} belong to a line of slope +1 then $i = h$.

(iii) if c^r_{ij} and c^s_{hq} belong to a line of slope -1 then $j = q$.

proof

Clearly, any variable c_{ij} must go at least u+1 times through the array in order to perform the n accumulations

$$c_{ij} := c_{ij} + a_{ik}.b_{kj}\ ,\ k = 1\ ,\ \dots\ ,\ n\ .$$

Let c^r_{ij}, $0 \leq r < u$, represent the variable c_{ij} when it has already gone r times through the array (see fig. 6).

(i): Since c^r_{ij} must be output from the array before giving birth to c^s_{ij}, $r<s$, it follows that, as they flow downwards, during the execution of the algorithm, c^r_{ij} and c^s_{ij} cannot be simultaneously in the array. As a consequence, if they are assigned respectively to $P = (x,y)$ and $P' = (x',y')$, then $y' - y \geq p$.

Clearly, c^0_{ij} must perform at least one accumulation, otherwise it is useless and can be eliminated; as a consequence, from lemma 1, it belongs to $\mathcal{A}_1$, and it follows from property (i) above that:

(*) c^r_{ij} belongs to $\mathcal{A}_w$ for any $r \geq 1$.

Let us assume that c_{ij} is computed by going more than u+1 times through

Lemma 1 :

If a variable c_{ij} performs the r accumulations $c_{ij} := a_{ik_1}.b_{k_1j} + a_{ik_2}.b_{k_2j} + \ldots a_{ik_r}.b_{k_rj}$ as it goes through the array, then:

- c_{ij} must be assigned to $\mathfrak{R}_r$,
- the coefficients $a_{ik_1}, \ldots, a_{ik_r}$ must be assigned to the intersection between $\mathfrak{L}_r$ and the line of slope +1 containing c_{ij}
- the coefficients $b_{k_1j}, \ldots, b_{k_rj}$ must be assigned to the intersection between $\mathfrak{R}_r$ and the line of slope -1 containing c_{ij}.

proof:

Since c_{ij} performs r accumulations, its lowest position corresponds to the case where these accumulations are performed in the r lowest rows of the array, and the result follows from the proof of proposition 1 □

We are now ready to study the general case.

For any array of size pxm, we decompose n into the form $n = u.p + w$, where $u \geq 0$ and $1 \leq w \leq n$. Clearly, any coefficient c_{ij} must go at least u+1 times through the array in order to perform the n accumulations

$$c_{ij} := c_{ij} + a_{ik}.b_{kj}, \quad k = 1, \ldots, n .$$

Moreover, if c_{ij} is computed by going u+1 times through the array, then as it goes through the array for the first time, it must perform at least w accumulations. For instance:

- if $p = n-1$ then $u = 1$ and $w = 1$
- if $p \geq n$, then $u = 0$ and $w = n$.

the array. Since from property (*) above, c^s_{ij} belongs to $\mathcal{A}_w$ for $s = 1, \ldots, u+1$, we just have to rename the variables c^s_{ij} $1 \le s \le u+1$, as c^r_{ij}, $0 \le r \le u$, in order to match with the formulation of the proposition.

If c_{ij} is computed by going $u+1$ times through the array, then c^0_{ij} performs at least w accumulations, and from lemma 1, it must be assigned to a point $P = (x,y)$ of $\mathcal{A}_w$. Since from (i), any variable c^r_{ij}, $1 \le r \le u$ must be assigned to a point $P' = (x',y')$ such that $y' - y \ge p$, it follows that all the variables c^r_{ij}, $0 \le r \le u$ are assigned to $\mathcal{A}_w$.

(ii): similar to property (i) of proposition 1
(iii): similar to property (ii) of proposition 1 □

Comment:

As shown in figure 6, two variables c^r_{ij} and c^r_{iq}, such that $j \neq q$, or two variables c^r_{ij} and c^s_{iq} with $r \neq s$, may belong to a line of slope +1.

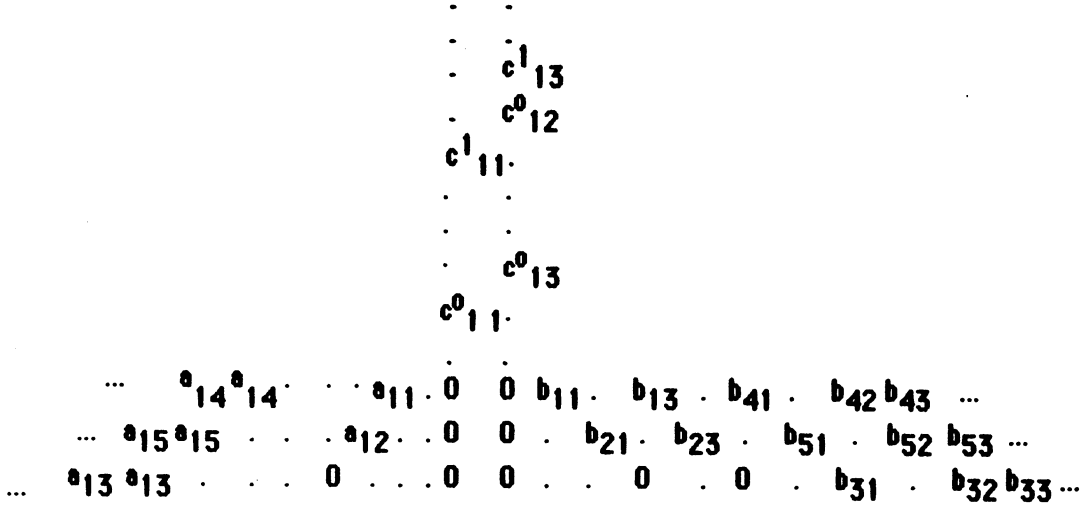


Fig. 6 **n = 5, m = 2, p = 3 (a partial view of the algorithm)**

Remark

Proposition 2 does not characterize the systolic algorithms associated with arbitrary orthogonally connected networks. It just gives some necessary conditions.

Proposition 3:

For any algorithm defined on an array of size pxm, 1 ≤p,m, and which runs in time T, there exists a new algorithm associated with an array of size nxm, which runs in time T' ≤ T, and where any c_{ij} is computed by going once through the array.

proof

We denote n = u.p + w, where u ≥ 0 and 1 ≤ w ≤ n.

Let us consider an optimal algorithm defined on a network of size pxm and which runs in time T. Let us denote by T' the running time of a new algorithm associated with a network of size nxm, and constructed as follows:

- eliminate the variables c^r_{ij} such that r ≥ 1; at this stage, only the variables c^0_{ij} 1 ≤ i,j ≤ n, are maintained.
- eliminate the p-w upper rows of the array; at this stage, the new array is of order wxm, and the variables c^0_{ij} are assigned to $\mathcal{A}_w$
- add n-w new rows at the bottom the array; now, the array is of order nxm, and the variables c^0_{ij} are assigned to $\mathcal{A}_n$

Clearly, in this new algorithm, any variable c_{ij} is represented by c^0_{ij} and is computed by going once through the array (see proposition 1).

Let y* be the y-coordinate of the heighest point (x,y) where a variable c^0_{ij} has been assigned in the original algorithm i.e.
y*= max{y: a variable c^0_{ij} has been assigned to a point of the form (x,y)}.
Let c^0_{hq} be a variable assigned in the original algorithm, to a point of the form (x,y*). From proposition 2, c^u_{hq} is assigned in the original algorithm to a point (x',y') such that y' ≥ y* + u.p (we recall that n = u.p + w, u ≥ 0, 1≤w ≤ n).

The situation can therefore be summarized as follows:

- the region where the variables c^0_{ij} corresponding to the new algorithm are assigned, is obtained from the region where the variables c^r_{ij}, $0 \le r \le u$, $1 \le i,j \le n$, of the original algorithm are assigned, by deleting at least u.p upper rows.

- the array corresponding to the new algorithm is obtained from the array associated with the original algorithm by adding n-w lower rows.

It follows that:

$$T' \le T + (n-w) - u.p$$
$$\le T \qquad \text{(since } n = u.p + w\text{)}$$

This ends the proof of the proposition □

Remarks

1- Since we are interested in optimal algorithms, we shall hereafter restrict our attention to algorithms defined on arrays of size nxm and where any c_{ij} is computed by going once through the array.

2- Let D be the subset of region $\mathcal{R}_n$ consisting of the couples (x,y) such that $y \le y^*$, where y^* is the y-coordinate of the heighest point where a variable c_{ij} has been assigned. In the usual presentation of a matrix C, the rows are horizontal and the columns are vertical. Proposition 1 says that the designing of the class of systolic algorithms studied here, corresponds to a presentation where any row (resp. column) of C is assigned to one or more segments of D with slope +1 (resp. -1). Hereafter the segments of slope +1 (resp. -1) are called L-segments (resp. R-segments).

3- Since region D admits a vertical symmetry axis, it follows that the number of L-segments is equal to the number of R-segments.

We shall denote

- S(D) the number of L-segments (resp. R-segments) of D
- H(D) the number of horizontal segments of D.
- $\lfloor x \rfloor$ the greatest integer lower than or equal to x
- $\lceil x \rceil$ the smallest integer greater than or equal to x

Lemma 2

Let D be the region associated with an algorithm defined on an array of size nxm.

(i) If the algorithm runs in time T, then:

$$T=n+\lfloor m/2 \rfloor + H(D) - 1 = \begin{cases} n+\lfloor m/2 \rfloor + \lceil S(D)/2 \rceil - 1 & \text{if } S(D) \le m \\ n+S(D)-1 & \text{if } S(D) \ge m \end{cases}$$

(ii) If all the segments of D are of length less than n, then $S(D) \ge 2n$.

(iii) If we can assign C to a region D containing exactly n^2 points then the algorithm associated with D is optimal.

proof

(i) Clearly the width of the horizontal band which lies between the network and the lowest point of D is $\lfloor m/2 \rfloor$. Since the network contains n rows, and T is the distance between the heighest point D and the lowest point of the network, it follows that (see fig.7a, 7b, 7c, 7d)

$$(*) \quad T = n + \lfloor m/2 \rfloor + H(D) - 1$$

case 1: S(D) ≤ m (see Fig. 7c, 7d)

It is easily seen that, the upper horizontal boundary of region D is of length S(D).

- If m is even, then the sequence of lengths of the horizontal segments of region D, is of the form:

2, 4, 6, ..., 2q-2, 2q=S(D)

hence H(D)= S(D)/2

- If m is odd, then the sequence of lengths of the horizontal segments of region D, is of the form:

1, 3, 5, ..., 2q-1, 2q+1 = S(D)

hence H(D) = (S(D)+1)/2.

This shows that, in any case, $H(D)= \lceil S(D)/2 \rceil$, hence from (*),

$$T= n +\lfloor m/2 \rfloor + \lceil S(D)/2 \rceil - 1$$

case 2: S(D) ≥ m (see fig. 7a,7b)

By considering the set of intersection points between the vertical line containing the left boundary of the array, and the lines of slope +1 containing the L-segments of D, it is easily seen that (see fig. 7a, 7b):

$S(D) = \lfloor m/2 \rfloor + H(D)$

hence, from (*)

$$T = n + \lfloor m/2 \rfloor + H(D) - 1 = n + S(D) - 1 \quad \square$$

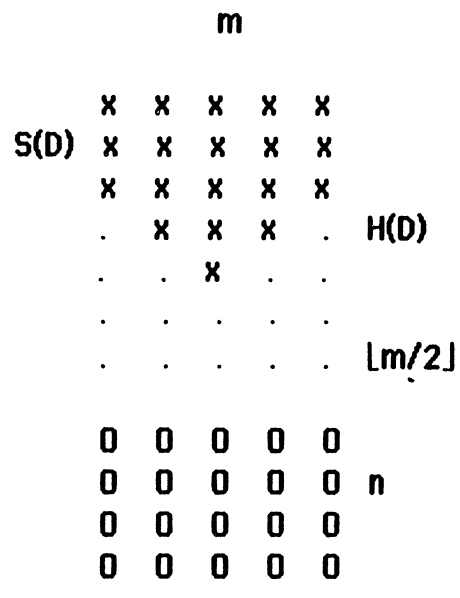


Fig. 7a : m=5 , n=4 , S(D)=7

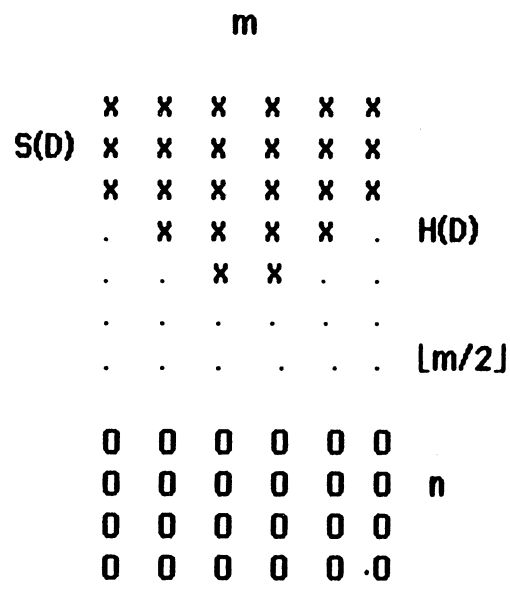


Fig. 7b : m=6 , n=4 , S(D)=8

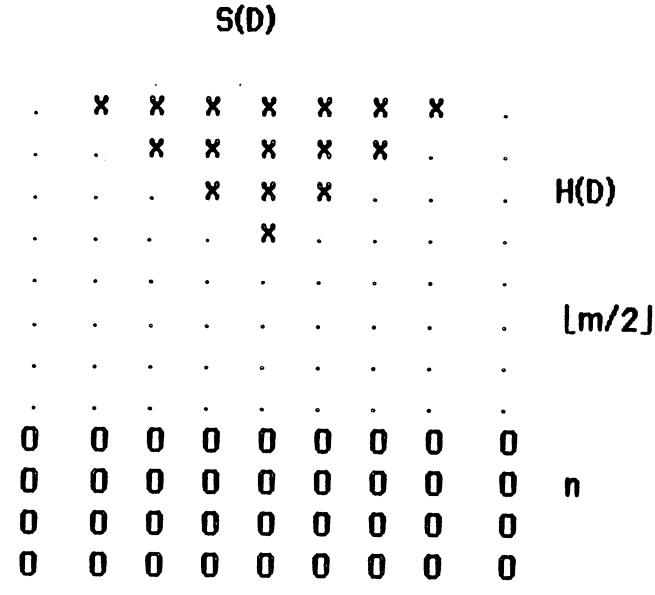


Fig. 7c : m=9 , n=4 , S(D)=7 , H(D)=4

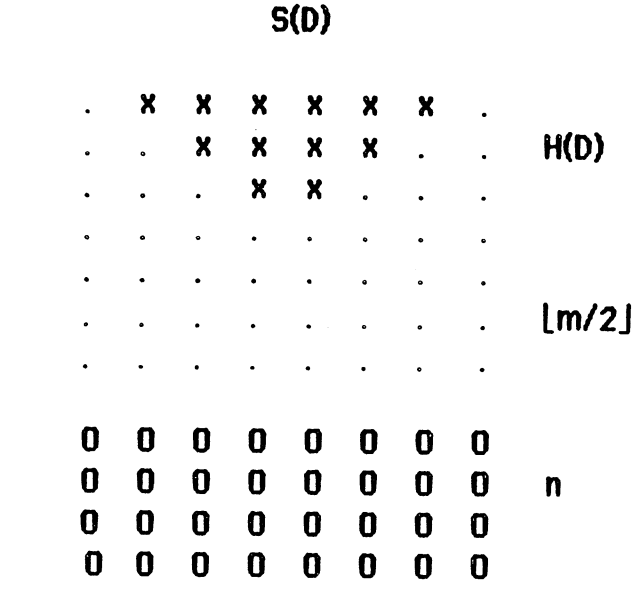


Fig. 7d : m=8 , n=4 , S(D)=6 , H(D)=3

We are now ready to prove the main theorem.

Theorem

(i) If $m < n$ then $T(n,m) \geq 3n-1$
(ii) If $m \geq n$ then

$$T(n,m) = \begin{cases} 3n-2 & \text{if } n \leq m \leq 2n-1 \text{ , } m \text{ odd} \\ 3n-1 & \text{if } n \leq m \leq 2n \text{ , } m \text{ even} \\ 2n-1+\lfloor m/2 \rfloor & \text{if } 2n-1 \leq m \end{cases}$$

proof

(i) Let us assume that $m < n$, and let D be the region associated with an optimal algorithm.
Since all the L-segments of D are of length less than n, it follows from lemma 2 (ii), that $S(D) \geq 2n > m$; thus, from lemma 2 (i),

$$T(n,m) = n + S(D) - 1 \geq 3n - 1.$$

(ii) Let us now study the case where $m \geq n$.

case 1: $m = 2q - 1 \geq 2n - 1$.

The region D* such that $H(D^*) = n$, is composed of two triangular grids of orders nxn and (n-1)x(n-1) ; it can therefore store the matrix C according to the constraints of proposition 1 (see fig. 8a for an illustration). Since D* contains n^2 points, it follows from lemma 2 (iii) that the algorithm associated with D* is optimal, hence

$$T(n,m) = H(D^*) + \lfloor m/2 \rfloor + n - 1 = 2n - 1 + \lfloor m/2 \rfloor.$$

case 2: $n \leq m = 2q - 1 < 2n - 1$.

The sequence of lengths (from bottom to top) of the L-segments (resp. R-segments) of the grid D* exhibited in case 1, is

$$n, n-1, n-1, \ldots, q, q, \ldots, 2, 2, 1, 1.$$

If we assign to these L-segments (resp. R-segments) the numbers 1, 2, ..., 2n - 1 from bottom to top, and if we apply to them the permutation

$$\sigma = (1 , n)(2 , n-1)(3 , n-2) \ldots (q-1 , q+1)$$

(see fig. 8c for an illustration), then we obtain a new grid associated with an array of size nxm whose L-segments from bottom to top, are of lengths

q, q, q+1, q+1, ..., n-1, n-1, n, q-1 , q-1 , ..., 2, 2, 1 , 1.

This grid is contained in a region D such that

S(D) = S(D*)= 2n-1.

Since S(D) ≥ m, it follows from lemma 2 (i) that this new algorithm runs in time

T = n + S(D) - 1 = 3n - 2

Now, if we consider the region D' obtained from D by deleting the heighest horizontal segment, then it is easily seen that:

- S(D') = 2n - 2
- all the L-segments of D' are of length less than n.

Thus, from lemma 2 (ii), it is not possible to define an algorithm associated with D'.

This shows that the algorithm associated with D is optimal, i.e.

T(n,m) = 3n - 2.

case 3: 2n ≤ m = 2q

Let D denote the region associated with an optimal algorithm. If H(D) < n (see fig. 7d for an illustration), then the lengths of the horizontal segments of D from bottom to top are 2, 4, 6, ..., 2H(D). As a concequence, the number of points of region D is H(D)(H(D)+1) < n^2, and this is a contradiction because D cannot store the n^2 coefficients of matrix C. This shows that H(D) ≥ n and this bound is tight because any region associated with an array of size nxm and such that H(D) = n, contains the grid D* of case 1 (see fig. 8b).

case 4: n < m = 2q < 2n

We first exhibit an algorithm which runs in time T = 3n - 1 in a way similar to the method of case 2 as follows:

- take the grid D' corresponding to the optimal solution exhibited in case 3, and number its L-segments (resp. R-segments) 1 , 2 , ... , 2n from bottom to top.

- apply to the L-segments (resp. R-segments) of D' , the permutation σ = (1 , n+1) (2 , n) (3 , n - 1) ... (q , q+2). This permutation yields a solution which runs in time T = 3n - 1 on an array of size nxm (see fig. 8b).

We are now going to show by contradiction, that T(n,m) > 3n - 2.

Let D" be the region obtained from the region associated with the preceding algorithm, by deleting the heighest horizontal segment, and let us assume the existence of an algorithm associated with D". It is easily verified that:

- $S(D^*) = 2n - 1$;
- D^* contains a single L-segment L^* (resp. R-segment R^*) of length n
- D^* contains $2(n-1)$ L-segments (resp. R-segments) of lengths less than n.

As a consequence, L^* (resp. R^*) must store a whole row, say row i^* of C (resp. a whole column, say column j^* of C), while the remaining $2(n-1)$ L-segments (resp. R-segments) of D^* must be grouped into n-1 pairs in order to store the remaining n-1 rows (resp. columns) of C; this leads to a contradiction because R^* and L^* do not intersect, and $c_{i^*j^*}$ will not be assigned (see fig. 9).
This ends the proof of the theorem □

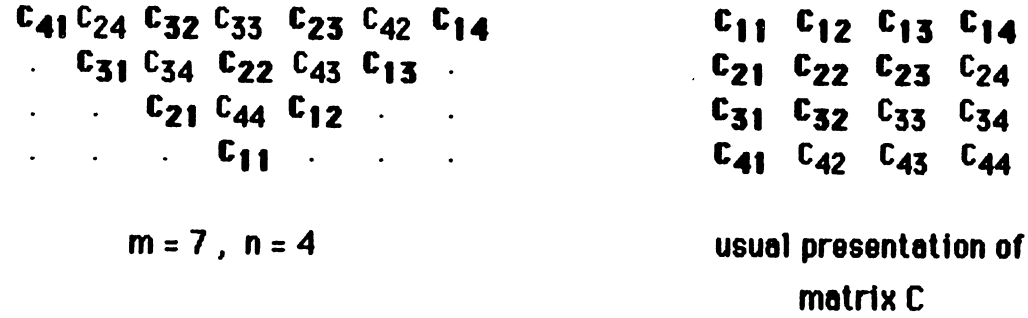


fig. 8a : **The triangular grid of order4x4 is darkened**

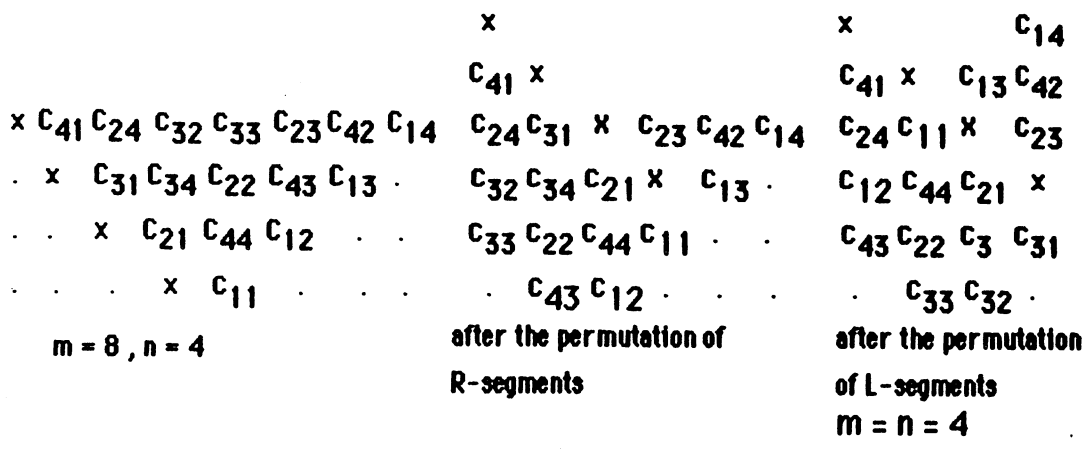


Fig. 8b $\sigma = (1,5)\,(2,4)$

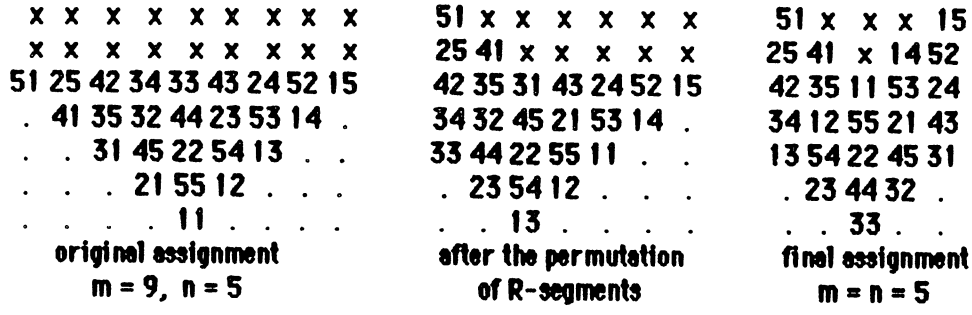
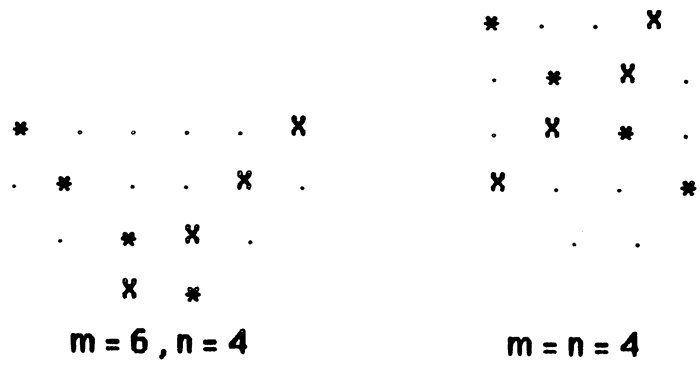


Fig. 8c $\sigma = (1,5)(2,4)$; c_{ij} is represented by ij .



points of L are represented by x
points of R are represented by *

Fig. 9

4-CONCLUSION

In this paper, we have studied the time complexity of the product of dense square matrices on the class of orthogonally connected systolic arrays composed of multiply-add cells without local memory, and where, for i,j,k=1,...,n, c_{ij} moves vertically and a_{ik}, b_{kj} move horizontally in opposite directions. The results show that optimal arrays, i.e. those which can support an algorithm of running time $Min_{p,m}\ T(p,m)$ are of size at least n.n (resp. n(n+1)) if n is odd (resp. even).

Since the complexity results are based on the assignment of the coefficients of the matrix C, to region $\mathcal{R}_n$, the method can trivially be extended to the computation of the product C=A.B where A is of order nxq and B is of order qxn.

On the other hand , the combinatorial approach introduced here, can be adapted to the designing of systolic algorithms for matrix product on any class of two-dimensional arrays; the only modifications concern the directions of the segments where the variables must be assigned. For instance, if one considers the class of algorithms defined on orthogonally connected networks, and where a_{ik} flows vertically whereas c_{ij} and b_{kj} flow horizontally in opposite directions, then, as shown in figure 10, it is possible to design, for any $n \geq 1$, an algorithm which runs in time 3n - 2 on an array of size nxn. However, this class of algorithm is not suited to situations where the array is much smaller than the matrices. On the other hand, as shown in figure 11, The combinatorial approach may also be applied to hexagonal arrays, in order to obtain algorithms with run in time 3n - 2, on an array of area n^2 .

In practical situations, especially when n is large, it is not realistic to assume the existence of arrays of size nxn. Therefore, the study of optimal algorithms associated with networks of size pxm, where p and m are smaller than n, is a very important and interesting problem. In [11], we have applied the combinatorial approach introduced here to derive optimal algorithms for the product of dense square matrices of order n, on arrays of order nx2 and nx3.

In a forthcoming paper, we shall show how the combinatorial approach can be used to derive optimal algorithms for matrix product on various classes of regular arrays.

Recently, Li and Wah [8] have proposed a general approach for the designing of systolic algorithms. Their method can be applied to a wide class of problems; for instance, it yields a very efficient algorithm for triangular matrix inversion. When applied to matrix product, their method gives the algorithm of Weiser and Davis [15] which runs in time 3n-2 but which needs an array of size $3n^2 - 3n + 1$.

In all the algorithms exhibited here, and which compute the product of two square matrices of order n, in time T = 3n - 2 on a network of size n^2, some of the data variables a_{ik} and b_{kj} have to be input more than once to the array. We conjecture that this multiple reading of some data variables is a necessary condition for the designing of algorithms which run in time $T \leq 3n - 2$ on arrays of size nxn. Since, in all the methodologies proposed in the literature for the automatic synthesis of systolic arrays,([6], [7], [8], [11], [12], [14], [16]), any input of a problem is represented by a single variable, this conjecture implies that these methods cannot lead to algorithms which compute in time T = 3n - 2, the product of two dense matrices of order n, on a network of size nxn.

	a					.	.	a_{32}						
c		c + a.b				.	a_{31}	a_{31}						
b		b				a_{33}	a_{33}	a_{21}						
	a					a_{32}	a_{23}	a_{13}						
						a_{22}	a_{12}	a_{13}						
						a_{11}	a_{12}	.						
						a_{11}	.	.						
c_{33}	c_{31}	c_{23}	.	.	0	0	0	b_{23}	b_{21}	b_{33}	b_{31}	b_{11}	b_{23}	
c_{32}	c_{22}	c_{12}	.	.	0	0	0	b_{12}	b_{22}	b_{22}	b_{32}	b_{12}	b_{12}	
c_{21}	c_{13}	c_{11}	.	.	0	0	0	b_{11}	b_{13}	b_{21}	b_{23}	b_{31}	b_{11}	

Fig. 10 n = m = 3

c	a				c_{12}	c_{11}	c_{13}	.	.	a_{12}	a_{13}	a_{11}			
b		b			.	c_{21}	c_{23}	c_{22}	.	a_{23}	a_{21}	a_{22}			
	a	c + a.b			.	.	c_{33}	c_{32}	c_{31}	a_{31}	a_{32}	a_{33}			
					.	.	b_{23}	b_{32}	b_{11}	0	.	.			
						b_{21}	b_{33}	b_{12}	b_{21}	0	0	.			
					b_{22}	b_{31}	b_{13}	b_{22}	b_{31}	0	0	0			
					b_{32}	b_{11}	b_{23}	b_{32}	.	.	0	0			
					b_{12}	b_{21}	b_{33}	.	.	.	.	0			

Fig. 11 n = 3.

5-REFERENCES

[1] **A. L. Fisher, H. T. Kung, L. M. Monier, H. Walker and Y. Dohi**
Design of the PSC: a programmable systolic chip.
in Proc. Third Caltech Conf. on VLSI, R. Bryant ed., Computer Science Press Inc. March 1983

[2] **L. J. Guibas, H.T. Kung and C.D. Thompson**
Direct VLSI implementation of combinatorial algorithms,
in Proc. Caltech Conf. on VLSI: Architecture, Design, Fabrication, Caltech, jan. 1979, 509-521

[3] **H. T. Kung**
Why systolic architectures?
Computer Magazine, 15 (1), jan. 1982, 37-46

[4] **H.T. Kung and P.L. Lehman**
Systolic (VLSI) arrays for relational data base operations,
in Proc. of Int. Conf. on Management of data,
ACM-SIGMOD, 1980, 105-116

[5] **H. T. Kung and C. E. Leiserson**
Systolic arrays for VLSI,
in C. A. Mead and L. A. Conway ed.
Introduction to VLSI systems, Addison Wesly, 1980 section 8.3, 37-46

[6] **H.T Kung and W.T. Lin**
An algebra for VLSI algorithm design,
Technical Report, Carnegie Mellon University, April 1983

[7] **C.E. Leiserson and J. B. Saxe**
Optimising synchronous systems,
Journal of VLSI and computer Systems Vol.1 N°1 41-47

[8] **G.J. Li and B. W. Wah**
The design of optimal systolic algorithms,
IEEE. Transaction on computers, Vol. c-34, N° 1, Jan. 1985, 66-77

[9] **L. Melkemi and M. Tchuente**
Programmation du produit matriciel sur un réseau systolique rectangulaire
TSI Vol. 4 , N° 5 Sept.-Oct. 1985, 459-470

[10] **L. Melkemi and M. Tchuente**
I/O-time trade-offs for matrix product on programmable arrays,
to appear in the Proceedings of COMPAR 85

[11] **W. L. Miranker and A. Winkler**
Space-time representations of computational structures.
I.B.M Research Report RC 9775, 1982

[12] **D. I. Moldovan**
On the analysis and synthesis of VLSI algorithms,
IEEE. Transactions on computers Vol.c-31 N°11, Nov. 82, 1121-1126

[13] **F.P. Preparata and J.Vuillemin**
Area-time optimal VLSI networks for multiplying matrices,
Information Processing Letters, 11, (2), 1980, 77-80

[14] **P. Quinton**
The systematic design of systolic arrays,
INRIA Research Report N°216, 1983

[15] **Y. Robert and M. Tchuente**
Une approche divide-and-conquer pour des algorithmes systoliques.
Rapport de Recherche I.M.A.G, N° 393, Sept. 1983

[16] **U. Weiser and A. Davis**
Mathematical representation for VLSI arrays.
Report UUCS-80-111, University of Utah, Sept. 1980

SYSTOLIC ALGORITHMS FOR RECTANGULAR MATRIX PRODUCT: A COMPLEXITY RESULT †

Lamine MELKEMI and Maurice TCHUENTE
CNRS/INPG Laboratoire TIM3 BP 68
38402 Saint Martin d'Hères cédex France

Abstract. - In this paper, we provide a time-complexity result for the product C = A.B of two rectangular matrices A and B of orders nxm and mxp , p ≤ n, on a class of two-dimensionnal systolic arrays with orthogonal interconnection pattern and composed of multiply-add cells. Taking the cycle-time of a multiply-add cell as time unit, we show that the minimum time necessary to solve the problem on this class of networks is

$$T(n,m,p) = \begin{cases} m + 2p + 2q - 2 & \text{if } n - p = 3q \leq 3(p-1)/2 \\ m + 2p + 2q - 2 & \text{if } n - p = 3q-1 \leq 3(p-1)/2 \\ m + 2p + 2q - 1 & \text{if } n - p = 3q+1 \leq 3(p-1)/2 \\ m + n - 1 + \lceil (p-1)/2 \rceil & \text{otherwise} \end{cases}$$

and an algorithm with time-performance T(n,m,p) can be implemented on an array of size mx(2p-1). This generalizes a result previously established by the authors for the product of square matrices

Résumé. - Dans cet article, nous étudions la complexité en temps, du calcul du produit C = A.B de deux matrices rectangulaires A et B d'ordres nxm et mxp, sur une classe de réseaux systoliques bi-dimensionnels à connexions orthogonales et composées de cellules d'accumulation. Nous montrons que le temps minimum nécessaire pour résoudre le problème sur ce type de réseau est

$$T(n,m,p) = \begin{cases} m + 2p + 2q - 2 & \text{si } n - p = 3q \leq 3(p-1)/2 \\ m + 2p + 2q - 2 & \text{si } n - p = 3q-1 \leq 3(p-1)/2 \\ m + 2p + 2q - 1 & \text{si } n - p = 3q+1 \leq 3(p-1)/2 \\ m + n - + \lceil (p-1)/2 \rceil & \text{sinon} \end{cases}$$

et qu'un algorithme s'exécutant en temps T(n,m,p) peut s'implémenter sur un réseau de taille mx(2p-1). Ceci généralize un résultat obtenu précédemment par les auteurs pour le produit de matrices carrées.

† Le Calcul Demain: à la mémoire de N. Gastinel

1 - INTRODUCTION

Systolic arrays, as defined by Kung and Leiserson (4), are a very nice framework for the VLSI implementation of specialized algorithms. First of all, they are easy to manufacture because they are composed of a few types of processing elements interconnected in a local and regular way. On the other hand, they can exploit the great potentiality of pipelining and multiprocessing: once a data item is read by a boundary element, it can be used successively by several cells of the array as it circulates from a cell to its neighbour. As a consequence, systolic arrays are very efficient parallel architectures which can fasten the solution of compute-bound problems while maintaining only modest bandwidth, Kung (2).

A typical problem for which systolic arrays can be used efficiently, is the computation of the product $C = A.B$ of two rectangular matrices A and B of orders nxm and mxp. Indeed, as the coefficients $c_{ij} = a_{i1}.b_{1j} + \ldots + a_{im}.b_{mj}$, $1 \le i \le n$, $1 \le j \le p$ are computed, any data item a_{ik} must be used p times and any data item b_{kj} must be used n times.

Many ad-hoc systolic designs have already been proposed in the literature for this problem, see for instance Guibas et al (1), Kung and Leiserson (4), Kung and Lehman (3), Preparata and Vuillemin (6), Robert and Tchuente (7), Weiser and Davis (8).

However, the first time-complexity result is due to the authors. In Melkemi and Tchuente (5), they have shown that if $T_{r,s}(n)$ denotes the time-performance of an optimal algorithm which computes the product of two square matrices of order n, on an array of size rxs, then:

- if s is fixed then $T_{r,s}(n)$ is minimum for r=n
- if $s < n$ then $T_{n,s}(n) \ge 3n-1$
- if $s \ge n$ then

$$T_{n,s}(n) = \begin{cases} 3n-2 & \text{if } n \le s \le 2n-1 \quad s \text{ odd} \\ 3n-1 & \text{if } n \le s \le 2n \quad s \text{ even} \\ 2n-1+\lfloor s/2 \rfloor & \text{if } 2n-1 \le s \end{cases}$$

The objective of this paper, is to extend this result to the case of rectangular matrices.

2 - NOTATIONS AND BASIC DEFINITIONS

Clearly, any coefficient c_{ij} can be computed according to the following sequential program:

$c_{ij} := 0;$

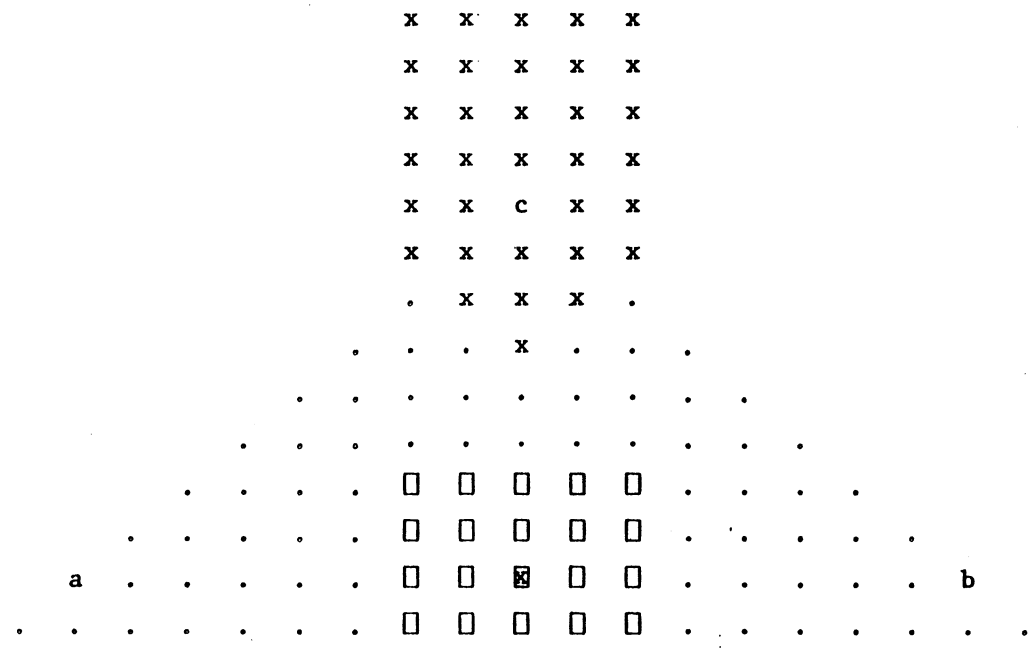
for k := 1 to m do

$c_{ij} := c_{ij} + a_{ik} \cdot b_{kj}$

It is therefore natural to assume that the basic processing element is a multiply-add cell: when such a cell receives three variables a,b,c at time t, the data a and b are output without any change at time t+1 whereas c becomes c+a.b. The cells of the array are assumed to be orthogonally connected: the cell at position (x,y) can directly communicate with the cells at positions (x-1,y), (x+1,y), (x,y-1), (x,y+1). The class of computation schemes studied here generalizes the classical pipeline scheme and has been introduced for the first time by Kung and Lehman (3). As c_{ij} performs the n accumulations $c_{ij} := c_{ij} + a_{ik} \cdot b_{kj}$, it flows vertically, whereas a_{ik} and b_{kj} flow horizontally in opposite directions.

Let us identify an array of size rxs with a rectangle of size rxs contained in ZxZ and whose edges are horizontal and vertical. As illustrated in figure 1 below, if three variables c,a,b have to meet in a cell at position (x,y) in order to perform the accumulation c := c + a.b, then c must belong to the vertical axis containing (x,y) whereas a (resp. b) must be at the intersection point between the horizontal line containing (x,y) and the line of slope +1 (resp. -1) containing c.

Let E denote the set of intersection points between the lines of slope which pass above the array without crossing it and the lines of slope -1 which pass above the array without crossing it. (see figure 1 for an illustration).



⊠ represents the cell at position (x,y); x stands for a point of E

fig. 1

step 1

x	x	33	x	x
x	32	x	23	x
31	x	22	x	13
	21	x	12	
		11		

step 2

42	43	x	x	x
41	x	x	x	x
x	x	x	x	x
	x	x	x	
		x		

step 3

x	x	x	52	x
x	x	53	x	62
x	51	x	63	x
	x	61	x	
		x		

n = 6 and p = 3

62	45	53	x	44	63	35	72	x
x	52	55	43	61	34	73	25	82
51	54	42	65	33	71	24	83	15
	41	64	32	75	23	81	14	
		31	74	22	85	13		
			21	84	12			
				11				

ij : step 1

ij : step 2

ij : step 3

n = 8 and p = 5

fig. 2

The complexity results of this paper, are based on the following proposition whose proof may be found in Melkemi and Tchuente (5).

PROPOSITION 1 - Let $T_{r,s}(n,m,p)$ denote the minimum time necessary to compute the product of two square matrices A and B of orders nxm and mxp, on an array of order rxs.

(i) if $r < n$ then $T_{r,s}(n,m,p) \geq T_{m,s}(n,m,p)$

(ii) the designing of a systolic algorithm associated with an array of size mxs is equivalent to the combinatorial problem of assigning the coefficients c_{ij} to region E in such a way that:
- if c_{ij} and c_{hq} belong to a segment of slope +1 then i=h
- if c_{ij} and c_{hq} belong to a segment of slope -1 then j=q

COMMENT : In the usual presentation of a matrix, the rows are horizontal and the columns are vertical. Proposition 1 says that the designing of a systolic algorithm for matrix product on the class of arrays studied here, consists of presenting the matrix C in such a way that any row (resp. column) of C is assigned to one or several segments of E with slope +1 (resp. -1). Proposition 1 also shows that, as far as optimal algorithms are concerned, we can restrict ourselves to arrays with m rows; in such an array, any c_{ij} can be computed by going exactly once through the network.

Hereafter, for any systolic algorithm associated with an array of size mxs, D denotes the set of points (x,y) of E such that $y \leq \bar{y}$, where $\bar{y}$ is the maximum y-coordinate of a point where a variable c_{ij} has been assigned. The segments of D with slope +1 are called L-segments and S(D) denotes their number; $S^{-}(D)$ denotes the number of L-segments of length less than p.

It is easily verified that the time-performance of an algorithm associated with a region D verifies $T \geq S(D) + m - 1$.

3 - COMPLEXITY RESULTS

The main theorem of this paper is the following:

THEOREM 1 - If $p \leq n$ then

$$\min_{r,s} T_{r,s}(n,m,p) = T(n,m,p) = \begin{cases} m+2p+2q-2 & \text{if } n-p = 3q \leq 3(p-1)/2 \\ m+2p+2q-2 & \text{if } n-p = 3q-1 \leq 3(p-1)/2 \\ m+2p+2q-1 & \text{if } n-p = 3q+1 \leq 3(p-1)/2 \\ m+n-1+\lceil (p-1)/2 \rceil & \text{otherwise} \end{cases}$$

We are not going to give a complete proof of the theorem. We shall rather present some lemmas which will give a flavour of the method to follow in order to carry out a detailed proof.

If we consider a set of L-segments of length less than p, then we need at least two of them in order to store a single row of C hence $S(D) \geq n + \bar{S}(D)/2$; in particular, if all the L-segments are of length less than p then $S(D) \geq 2n$, and consequently, $T \geq m + 2n - 1$. This shows that, in order to prove the main theorem, we only need to consider the case where the region D contains at least one L-segment of length p.

On the other hand, let N(D) denote the maximum number of rows of length p which can be stored in a region D; it is easily verified that:

(a) $N(D) \leq S(D) - \bar{S}(D)/2$

(b) If D and D' are associated with the same network then

$S(D') = S(D) + 1 \qquad N(D') \leq N(D) + 2$

$S(D') = S(D) + 2 \qquad N(D') \leq N(D) + 3$

LEMMA 1 - For any n,m,p, $T(n,m,p) \geq n + m - 1 + \lceil (p-1)/2 \rceil$

Proof: As noted previously, we can assume that the region D associated with an optimal algorithm, contains an L-segment of length p. Therefore, the lengths of L-segments from top to bottom, is a sequence $1 \leq a_1, a_2, \ldots, p, a_{i+1}, \ldots$ such that $|a_{j+1} - a_j| \leq 1$ for any j , hence $\bar{S}(D) \geq p-1$, and

$T(n,m,p) \geq m + S(D) - 1 \geq m + n - 1 + \lceil (p-1)/2 \rceil$ □

LEMMA 2 - If $0 \quad n-p \leq 3(p-1)/2$ then

$$T(n,m,p) \geq \begin{cases} m + 2p - 2 + 2q & \text{if } n-p \in \{ 3q, 3q-1 \} \\ m + 2p - 2 + 2q + 1 & \text{if } n-p = 3q+1 \end{cases}$$

Proof: We assume that $n-p = 3q$ and we restrict ourselves to arrays with an odd number of columns; the other cases may be carried out in a similar way.

We are going to prove by induction on q that any region with an odd number of columns and which can store C verifies $N(D) \geq p + 3q$, hence $S(D) \geq 2p-1+2q$.

For $q = 0$, the result follows from the proof of the main theorem of Melkemi and Tchuente (5). Now let D be a region with an odd number of columns and such that $N(D) = p+3q$, $q>1$ and let D' be the region obtained from D by deleting the two uppest horizontal rows. Clearly, $S(D') = S(D)-2$. Since, from property (b) above, $N(D') \geq p+3(q-1)$, it follows from the induction hypothesis that $S(D') \geq 2p-1+2(q-1)$, hence $S(D) = S(D') + 2 \geq 2p-1+2q$ and $T = m-1+S(D) \geq m+2p-2+2q$ □

OPTIMAL ALGORITHMS - In order to prove the main theorem, we just have to show that the lower bounds of lemma 1 and lemma 2 are tight. In order to do that, we consider the regions of width $2p-1$ which correspond to these bounds and we proceed as follows: first partition D into D' and D'' where D' is the set of couples (x,y) such that $x+y = 0$ (modulo 2).

step 1 - assign the rows of C to the L-segments of D' with length p, from bottom to top.

step 2 - for any L-segment of D' of length k less than p, consider an L-segment of D'' of length $p-k$ and assign a row of C to these two L-segments

step 3 - assign the remaining rows of C to the remaining L-segments of D'' with length p, from top to bottom.

An illustration is given in figure 2.

REFERENCES

1 - L.J. Guibas, H.T. Kung and C.D. Thompson, Direct VLSI implementation of combinatorial algorithms, in Proc. Caltech Conference on VLSI, 1879, pp. 509-521

2 - H.T. Kung, Why systolic architectures, IEEE Computer, jan. 1982, pp. 37-46

3 - H.T. Kung and P.L. Lehman, Systolic (VLSI) arrays for relational data base operations, in Proc. International Conf. on Management of Data, ACM-SIGMOD, 1980 pp. 105-116

4 - H.T. Kung and C.E. Leiserson, Systolic arrays for VLSI, in Introduction to VLSI, C.A. Mead and L.A. Conway ed. 1980, section 8.3

5 - L. Melkemi and M. Tchuente, Complexity of matrix product on orthogonally connected systolic arrays, Rapport de Recherche N°441, IMAG Grenoble, 1984 (submitted to IEEE-TC)

6 - F.P. Preparata and J. Vuillemin, Area-time optimal VLSI networks for multiplying matrices, Information Processing Letters, Vol. 11 (1980) N° 2, pp. 77-80

7 - Y. Robert and M. Tchuente, Une approche "divide-and-conquer" pour des algorithmes systoliques, Rapport de Recherche N° 393, IMAG Grenoble, Sept. 1983

8 - U. Weiser and A. Davis, Mathematical representation for VLSI arrays, Report UUCS-80-111, University of Utah, Sept. 1980

CHAPITRE 2

Algorithmes systoliques pour la multiplication de deux matrices

ALGORITHMIQUE

réseau systolique, programmation, produit matriciel, formulation combinatoire, algorithmes optimaux

ALGORITHMS

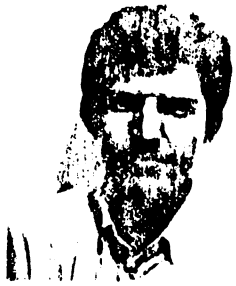
systolic network, programming, matrix product, combinatorial formulation, optimal algorithms

Programmation du produit matriciel sur un réseau systolique rectangulaire

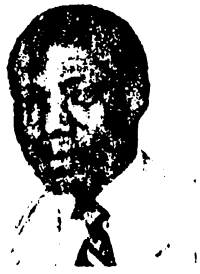
Matrix Product Programming on a Rectangular Systolic Network

Lamine MELKEMI et Maurice TCHUENTE

CNRS-Imag, Laboratoire TIM3, BP 68 38402 Saint Martin d'Hères Cedex



Lamine Melkemi a obtenu sa maîtrise en Mathématiques à l'université de Batna (Algérie) en juin 1982 et son DEA d'Analyse Numérique à l'université scientifique et médicale de Grenoble en juin 1983. Il prépare, actuellement, au sein du laboratoire TIM3 de l'Institut Imag, une thèse de troisième cycle sur les réseaux systoliques pour la résolution de problèmes linéaires.



Maurice Tchuente est de nationalité camerounaise. Après des études supérieures effectuées successivement dans les universités de Yaoundé, Nantes et Grenoble, il a soutenu en 1982, sous la direction du professeur Noël Gastinel, une thèse de doctorat d'état sur les systèmes de type coopératif. Assistant associé à l'université de Marseille St-Charles en 1976-1977, puis chercheur au CNRS jusqu'en 1984, il est actuellement professeur associé à l'université de Grenoble II. Ses recherches portent sur les problèmes de calculabilité, de synchronisation et de dynamique dans les réseaux d'automates, ainsi que sur l'algorithmique parallèle pour VLSI.

PRESENTATION

Parmi les diverses tentatives faites pour proposer des architectures de machines de plus en plus rapides, l'approche systolique apporte une solution originale à la résolution de problèmes pour lesquels les solutions algorithmiques consistent à répéter un grand nombre de fois des opérations simples et peu nombreuses. C'est ainsi le cas de calculs numériques classiques : opérations matricielles, produit de convolution, etc...
Une solution systolique à de tels problèmes est composée d'un ensemble de cellules (chacune représentant une opération élémentaire) organisées en réseau présentant une certaine régularité géométrique et au travers duquel se propagent des flux de données. Dans cette approche, c'est donc l'architecture matérielle elle-même qui réalise la solution algorithmique : la notion de programme au sens de l'enchaînement d'instructions portant sur des opérateurs généraux disparaît au profit d'un « enchaînement câblé ».
La réalisation de systèmes systoliques pose sous un angle nouveau les problèmes bien connus de la conception d'algorithmes :

- Comment passer d'un énoncé de problème à un algorithme systolique ?
- Comment s'assurer de la correction d'un algorithme systolique ?
- Comment mesurer et améliorer l'efficacité d'algorithmes systoliques ?

L'article de L. Melkemi et M. Tchuente concerne à la fois l'aspect évaluation d'efficacité dans le cas d'algorithmes réalisant des produits de matrices et la conception de tels algorithmes. Les auteurs introduisent des techniques d'analyse combinatoire permettant d'encadrer le temps d'exécution d'un tel produit. Ils peuvent ainsi concevoir, à partir de ces évaluations, des algorithmes optimaux pour certaines tailles de matrices.

Jean-Pierre Finance

COMMENTARY

Amongst the many attempts that have been made to propose increasingly rapid machine architectures, the systolic approach offers an original way of solving problems which, in the algorithmic approach, are dealt with by repeating a small number of simple operations, a great number of times.
This is the case, for example, of classical numerical computation problems : matrix operations, convolution products, etc.
A systolic solution for such problems involves a series of cells (each of which represents an elementary operation) organized into a network with a certain geometric regularity, through which a flow of data moves. In this approach, it is the hardware architecture itself which implements the algorithmic solution : the concept of a program as a series of commands affecting general operators gives way to " hard-wired chaining ".
The implementation of systolic systems poses well-known problems of algorithm design in a new light :

— How do we move from the statement of a problem to a systolic algorithm ?
— How can we be sure a systolic algorithm is correct ?
— How can we measure and improve the efficiency of systolic algorithms ?

In this article, Melkemi and Tchuente consider both efficiency evaluation problem for algorithms implementing matrix products, and the design of such algorithms. The authors introduce combinatorial analysis techniques making it possible to set an upper limit to execution time of a product of this kind. They can then use these evaluations as a starting point for the design of optimal algorithms for certain matrix sizes.

Jean-Pierre Finance

TABLE DES MATIÈRES

1. Introduction

Les réseaux systoliques ont été introduits par Kung et Leiserson [Kung 79] et sont des architectures parallèles spécialisées, composées de cellules élémentaires interconnectées de manière locale et régulière. Par exemple, dans le réseau à connexions rectangulaires représenté dans la figure 1, la cellule en position (x, y) communique uniquement avec ses voisines qui sont en position $(x - 1, y)$, $(x + 1, y)$, $(x, y - 1)$, $(x, y + 1)$. Dans un tel réseau, les cellules opèrent de manière synchrone : à l'instant t elles reçoivent des variables en provenance des cellules voisines en entrée et après avoir effectué des transformations élémentaires sur ces variables, elles les envoient à l'instant $t + 1$ aux cellules voisines en sortie. Ce modèle présente un double intérêt.

• D'abord, la réalisation d'un réseau systolique est beaucoup plus aisée que la conception d'un circuit VLSI général. En effet, un tel réseau est généralement composé de cellules élémentaires en grand nombre mais n'appartenant qu'à un, deux ou trois types différents ; le concepteur peut donc dans un premier temps concentrer ses efforts sur l'optimisation des cellules types ; la deuxième phase qui consiste à interconnecter les copies des cellules types est facilitée par le caractère planaire local et régulier de la structure d'interconnexion.

• Ensuite, comme l'a fait remarquer Kung [Kung 82], les performances réelles d'une architecture parallèle générale de type SIMD lors par exemple du calcul du produit $C = A.B$ de deux matrices carrées d'ordre n, peuvent être très décevantes par rapport aux capacités de calcul de la machine, à cause du grand nombre d'accès-mémoire. En effet, si l'on dispose de n^2 processeurs qui doivent lire tous les coefficients a_{ik}, b_{kj}, $i, j, k = 1, ..., n$, en mémoire centrale et qui ne communiquent pas entre eux, alors tout algorithme de calcul en parallèle de C en temps linéaire, sur une telle structure, exige la lecture simultanée en mémoire d'un nombre de variables de l'ordre de n^2, ce qui est prohibitif.

Le modèle systolique permet de résoudre ce problème d'accès-mémoire. En effet, dans un réseau systolique, seules les cellules frontières communiquent avec l'extérieur ; par exemple, dans le réseau de figure 1, une cellule frontière en position $(x, 1)$ lit en mémoire un coefficient b_{kj} pour le calcul de l'accumulation $c_{ij} := c_{ij} + a_{ik}.b_{kj}$ et l'envoie, l'instant d'après à sa voisine de droite en position $(x, 2)$ qui peut s'en servir pour une accumulation $c_{hj} := c_{hj} + a_{hk}.b_{kj}$ puis l'envoyer à son tour, à sa voisine de droite $(x, 3)$ et ainsi de suite. Comme la remarque s'applique aussi aux coefficients a_{ik}, on peut ainsi obtenir

un algorithme qui s'exécute en temps linéaire sur un réseau de taille proportionnelle à n^2 et nécessite la lecture simultanée d'un nombre de variables linéaire en n.

Dans cet article, nous étudions le calcul en parallèle du produit $C = A.B$ de deux matrices carrées d'ordre n sur un réseau systolique. Contrairement à l'attitude qui prévaut dans la plupart des travaux antérieurs sur le sujet, et que nous citons en référence, le point de vue adopté ici ne consiste pas à rechercher une nouvelle architecture parallèle spécialisée permettant de calculer rapidement C.

Nous supposons plutôt que l'on dispose d'une architecture classique composée de $n.m$ processeurs élémentaires et identiques qui opèrent de manière synchrone et sont organisés selon un réseau rectangulaire d'ordre $n \times m$; dans ce réseau, le processeur en position (x, y) communique directement avec ceux qui sont en position $(x - 1, y), (x + 1, y), (x, y - 1), (x, y + 1)$. Le problème qui se pose alors est de programmer efficacement le produit matriciel sur une telle structure parallèle.

Nous nous restreignons à la famille d'algorithmes où, à tout instant t, chaque cellule reçoit a_{ik}, b_{kj}, c_{ij} en provenance des cellules voisines respectivement à gauche à droite et en haut, effectue l'accumulation

$$c_{ij} := c_{ij} + a_{ik}.b_{kj}$$

et transmet, à l'instant $t + 1$,
a_{ik}, b_{kj}, c_{ij} aux cellules voisines respectivement à droite, à gauche et en bas. Ceci peut se décrire plus simplement en disant que, dans le réseau, les a_{ik}, b_{kj} circulent horizontalement dans des directions opposées tandis que les c_{ij} circulent de haut en bas (*fig.* 1).

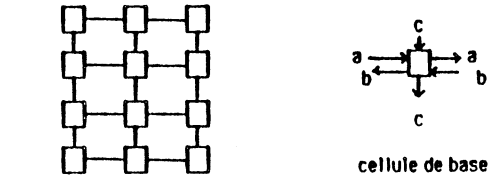


Figure 1.

Il convient de noter que nous nous restreignons aux réseaux comportant n lignes parce que, dans ce cas, tout coefficient c_{ij} initialisé à zéro, peut effectuer les n accumulations $c_{ij} := c_{ij} + a_{ik}.b_{kj}$, $k = 1, ..., n$ en passant une seule fois dans le réseau.

2. Formulation combinatoire

Identifions tout réseau d'ordre $n \times m$ avec un rectangle d'ordre $n \times m$ inclus dans $Z \times Z$ et dont les côtés sont parallèles aux axes. Nous notons M la bande horizontale de largeur n contenant le réseau.

Cas des réseaux linéaires ($m = 1$)

Lorsque $m = 1$, le réseau est linéaire et on est ramené à une structure pipeline classique (*fig.* 2). Chaque coefficient c_{ij} initialisé à zéro, circule verticalement (de haut en bas) et effectue successivement les accumulations $c_{ij} := c_{ij} + a_{ik}.b_{kj}$, $k = 1, ..., n$.

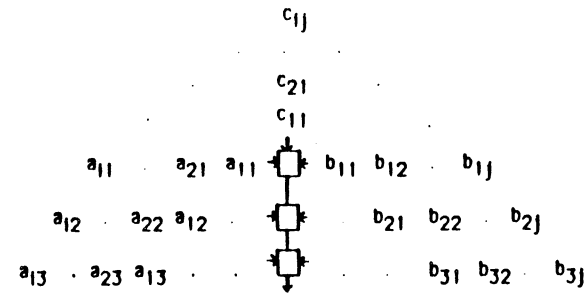


Figure 2. — Cas des réseaux linéaires ($m = 1$).

Tout algorithme systolique s'obtient alors de la manière suivante (à une permutation des indices près) :

(i) affecter les coefficients c_{ij} $i, j = 1, ..., n$ aux points situés au-dessus du réseau;

(ii) affecter les coefficients a_{ik} $k = 1, ..., n$ aux points situés à l'intersection entre M et toute droite de pente + 1 passant par un coefficient c_{ij};

(iii) affecter les coefficients b_{kj} $k = 1, ..., n$ aux points situés à l'intersection entre M et toute droite de pente − 1 passant par un coefficient c_{ij}.

Cas des réseaux rectangulaires ($m > 1$)

Dans le cas $m > 1$, il est clair que les coefficients c_{ij} situés au-dessus d'une colonne du réseau se calculent en pipeline dans les cellules de cette colonne. Dans l'exemple de la figure 3 nous avons écrit en caractères gras, les variables correspondant aux calculs qui s'effectuent sur la colonne de gauche du réseau.

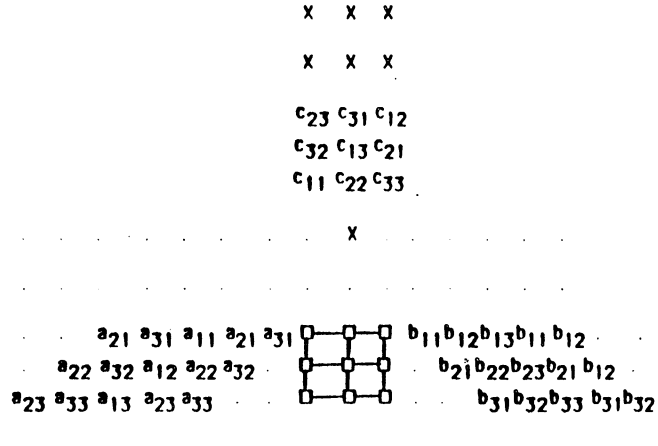


Figure 3. — Exemple de réseau rectangulaire ($n = m = 3$).

Tout algorithme systolique associé au réseau est donc une union de schémas pipeline correspondant aux colonnes du réseau.

Initialement, tous les coefficients a_{ik} sont hors du réseau; en conséquence, d'après la propriété (ii) ci-dessus, toute ligne a_{ij}, $j = 1, ..., n$ de A doit être affectée à un segment de pente + 1 contenu dans M et situé à gauche du réseau. De manière analogue, on déduit de la propriété (iii) ci-dessus que toute colonne b_{ij}, $i = 1, ..., n$ doit être affectée à un segment de pente − 1 contenu dans M et situé à droite du réseau.

Comme, d'après la propriété (i) ci-dessus, tout coefficient c_{ij} doit être affecté à un point situé à l'intersection entre une droite de pente + 1 contenant la i-ième ligne de A et une droite de pente − 1 contenant la j-ième colonne de B, on aboutit à la caractérisation suivante établie dans [Melkemi 84].

PROPOSITION 1 : *Identifions tout réseau de taille $n \times m$ avec un rectangle d'ordre $n \times m$ inclus dans $Z \times Z$ et dont les côtés sont parallèles aux axes ; par ailleurs appelons région (1) l'ensemble des points d'intersection entre les droites de pente + 1 qui passent à gauche du réseau sans le couper, et les droites de pente − 1 qui passent à droite du réseau sans le couper (fig. 4).*

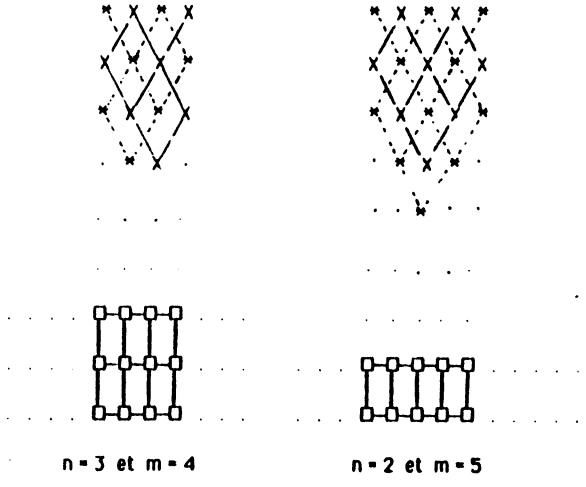


Figure 4.

La conception d'un algorithme systolique permettant de calculer le produit $C = A.B$ de deux matrices carrées d'ordre n sur un réseau d'ordre $n \times m$ revient à affecter les coefficients c_{ij} $i, j = 1, ..., n$ à la région (1) de manière à ce que :

(i) *si c_{ij} et c_{hq} appartiennent à une droite de pente + 1 alors $i = h$.*

(ii) *si c_{ij} et c_{hq} appartiennent à une droite de pente − 1 alors $j = q$.*

Commentaire : Dans la présentation usuelle d'une matrice C, les éléments c_{ij}, $j = 1, ..., n$ (resp. $i = 1, ..., n$) appartenant à une même ligne (resp. colonne) sont disposés horizontalement (resp. verticalement). La proposition 1 montre que, dans le modèle que nous étudions, la conception d'un algorithme pour le calcul du produit $C = A.B$ de deux matrices carrées d'ordre n sur un réseau systolique d'ordre $n \times m$, revient à présenter C dans la région (1) de manière à ce que toute ligne (resp. colonne) soit affectée à un ou plusieurs segments de pente + 1 (resp. − 1).

3. Notations et définitions de base

Dans la suite, les cellules du réseau sont numérotées (i, j), $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant m$, et celle située au coin inférieur gauche correspond à (1, 1). Pour un algorithme donné, nous considérons l'ordonnée maximale q^* des points de la région (1) auxquels ont été affectés des coefficients c_{ij} et nous notons D l'ensemble des points (x, y) de la région (1) tels que $y \leqslant q^*$.

Prenons pour unité de temps, le temps de cycle d'une cellule de base. Le temps de calcul d'un algorithme systolique est égal au délai entre la première entrée et la dernière sortie ; on vérifie aisément que le temps d'exécution d'un algorithme associé à une région D, est la distance entre la frontière supérieure de D et la frontière inférieure du réseau, et vaut donc $T = q^* - 1$.

On appelle longueur d'un segment de $Z \times Z$, le nombre de points de ce segment. Les segments de D de pente + 1 sont appelés L-segments et $N(L)$ désigne leur nombre ; les segments de D de pente − 1 sont appelés R-segments et $N(R)$ désigne leur nombre. Comme D admet un axe de symétrie verticale, on a $N(L) = N(R)$.

Notons que toute région triangulaire D vérifie

$$N(L) = N(R) \leqslant m \quad (\textit{fig. 5}). \qquad (1)$$

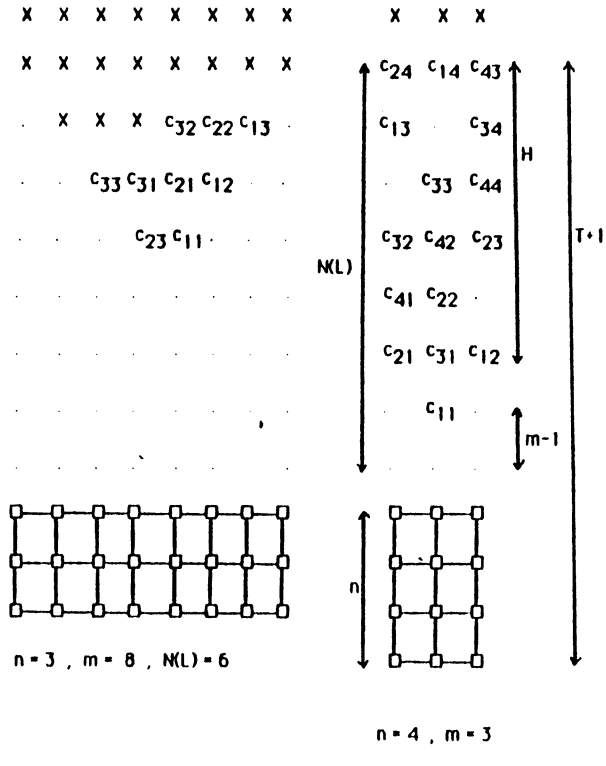


Figure 5.

Nous nous intéressons désormais au cas où $m < n$ (*fig.* 5). Les L-segments de D sont alors de longueur au plus m et il en faut au moins deux pour stocker toute ligne de C, donc $N(L) \geqslant 2n > m$. On déduit de (1) que D ne peut pas être triangulaire ; en notant H la longueur de

la frontière verticale de D, on vérifie aisément (*cf. fig.* 5) que

$$T = H + (m - 1) + n - 1 = H + m + n - 2 \quad (2)$$

(*cf. fig.* 5).

Par ailleurs, en comptant les points d'intersection entre la droite contenant la frontière verticale gauche de D et les L-segments de D on voit que $N(L) = H + m - 1$ ce qui, compte tenu de l'égalité (2) ci-dessus, entraine $T = N(L) + n - 1$.

Dans la suite nous noterons $T(n, m)$ le temps minimum nécessaire pour calculer le produit $C = A.B$ de deux matrices carrées d'ordre n sur un réseau d'ordre $n \times m$.

4. Introduction de la notion de grille

Notons D' l'ensemble des points (x, y) de D tels que $y + x = 0$ (modulo 2), et D'' l'ensemble des points (x, y) de D tels que $y + x = 1$ (modulo 2).

On vérifie aisément qu'un segment de D' de pente + 1 ou − 1, ne peut pas avoir de point commun avec un segment de D'' de pente + 1 ou − 1. Dans la suite, nous dirons que D' et D'' sont des grilles, ce qui est bien justifié au vu de la structure de leurs L-segments et R-segments (*fig.* 4).

La conception d'un algorithme systolique pour le calcul du produit $C = A.B$ de deux matrices carrées d'ordre n, revient donc à colorier les grilles D' et D'' de manière à ce que, pour tout $i, j = 1, ..., n$ il existe un L-segment de couleur i qui coupe un R-segment de couleur j.

Les grilles D' et D'' peuvent se construire séparément. Néanmoins, dans le paragraphe suivant, nous commençons par construire une grille périodique G dont les segments sont convenablement coloriés, puis, nous la découpons en deux morceaux pour obtenir deux nouvelles grilles (non périodiques) G' et G''. L'immersion de G' et G'' dans la région (1) permet alors d'obtenir la région D.

5. Borne générale

5.1. Énoncé du résultat

Dans ce paragraphe, nous donnons des bornes générales pour le temps de calcul du produit $C = A.B$ de deux matrices carrées d'ordre n sur un réseau d'ordre $n \times m$, lorsque $m \leqslant n$.

Proposition 2 : *Si* $m \leqslant n$ *et* $(k - 1)\, m < n \leqslant km$ *alors*

$$n(k + 1) - 1 \leqslant T(n, m) \leqslant n(k + 1) + m - 2 + f$$

où

$$f = \begin{cases} 0 \text{ si } m \text{ est impair ou } nk \text{ impair} \\ 1 \text{ sinon} \end{cases}$$

5.2. Plan de la démonstration

Comme tout segment de la région (1) est de longueur au plus m, il en faut au moins k pour stocker toute ligne de C donc $N(L) \geqslant nk$ ce qui entraîne

$$T = N(L) + n - 1 \geqslant n(k + 1) - 1.$$

Pour construire un algorithme correspondant à la borne supérieure de l'énoncé, nous procédons de la manière suivante :

Phase 1. Construction d'une grille périodique G

Nous considérons une grille G de largeur m, qui est périodique (c'est-à-dire se referme sur elle-même) et qui comporte nk L-segments $L_1, L_2, ..., L_{nk}$ et nk R-segments $R_1, R_2, ..., R_{nk}$. Cette numérotation est choisie de manière à ce que, pour tout p, le L-segment de rang p coupe les R-segments de rangs $j = p, p + 1, ..., p + m - 1$, l'addition étant effectuée modulo nk.

Phase 2. Coloration de la grille G

On colorie les segments de G de manière à ce que, pour tout couple (i, j), $1 \leqslant i, j \leqslant n$, il existe un L-segment de couleur i qui coupe un R-segment de couleur j.

Phase 3. Détermination de la région D

On découpe G dans le sens de la largeur en deux grilles G' et G'' dont les hauteurs diffèrent d'au plus une unité, et on les affecte à la région (1) en les tirant le plus possible vers le bas. On obtient ainsi la région D. Il convient de noter que dans le cas général, G' et G'' sont des sous-régions strictes de D' et D'' en raison notamment des points du triangle situé dans la partie inférieure de D. Le fait de tirer G' et G'' le plus possible vers le bas permet de réduire la hauteur de la région D et donc de réduire le temps d'exécution de l'algorithme correspondant.

Phase 4. Affectation des coefficients de C

On affecte c_{ij} à tout point de D situé à l'intersection entre un L-segment de couleur i et un R-segment de couleur j.

5.3. Présentation détaillée de la démonstration

Nous allons maintenant présenter les détails de cet algorithme.

Une suite de longueur sp et de période $(z_1, z_2, ..., z_s)$ est notée $(z_1, z_2, ..., z_s)^p$.

Les opérations sur les indices de segments seront toujours effectuées modulo nk et celles sur les couleurs modulo n.

Présentation détaillée de la phase 1

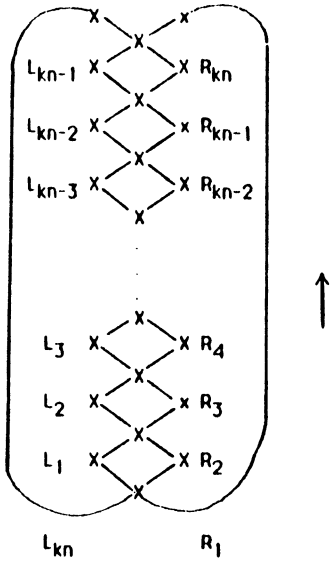
Dans la grille périodique G comportant nk L-segments (resp. R-segments) et de largeur m, tout L-segment coupe m R-segments consécutifs ; de même, tout R-segment coupe m L-segments consécutifs. Pour numéroter les segments de manière à ce que, pour tout i,

L_i coupe $R_i, R_{i+1}, ..., R_{i+m-1}$ (addition modulo nk), on procède comme suit (*fig.* 6*a*) :

- choisir arbitrairement un L-segment et lui affecter l'indice 1 ;
- choisir un sens de parcours de la grille et affecter successivement aux R-segments qui coupent L_1, les

indices 1, 2, ... m, puis aux suivants les indices $m + 1$, $m + 2$, ..., nk :

– conserver le sens de parcours précédemment choisi, et affecter, à partir de L_1, les indices successifs 2, 3, ... nk aux L-segments de la grille G.



le balayage de la grille se fait de bas en haut.

Figure 6a. — Phase 1, numérotation des segments.

Présentation détaillée de la phase 2 : étape 1 (fig. 6b)

(a) On affecte à tout segment R_p la couleur i, $1 \leqslant i \leqslant n$ telle que $i = p$ (modulo n). Les couleurs de $R_1, R_2, ..., R_{nk}$ correspondent donc à la suite $(1, 2, ..., n)^k$.

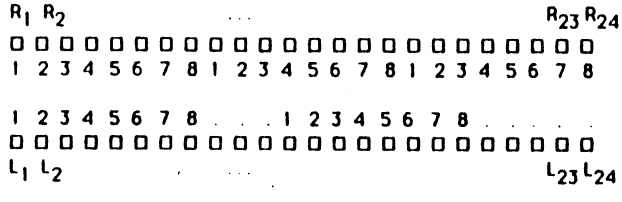


Figure 6b. — Phase 2, étapes 1 et 2. $n = 8$ et $m = 3$.

Notion de fenêtre

Pour tout sous-ensemble E de $\{1, 2, ..., n\}$, on dira qu'un L-segment correspond à une fenêtre de type E si l'ensemble des couleurs des R-segments qu'il coupe est E ; lorsque $E = \{p, p + 1, ..., q\}$, on le notera sous forme d'intervalle, $E = [p, q]$. On vérifie aisément qu'à l'issue de l'étape ci-dessus, on a :

(*) k fenêtres de type $[j, j + m - 1]$, pour $j = 1, ..., n$.

Présentation détaillée de la phase 2 : étape 2 (fig. 6b)

La seconde étape de coloration est alors la suivante :

(b) Pour $i = 1, ..., n$ et $q = 0, 1, ..., k - 2$, affecter la couleur i au segment $L_{i+q(n+m)}$. Si l'on part du L-segment L_1, alors cela revient à affecter aux segments rencontrés, des blocs de couleurs 1, 2, ..., n espacés de m les uns des autres.

Présentation détaillée de la phase 2 : étape 3 (fig. 6c)

On vérifie aisément que, à l'issue de l'étape précédente, les L-segments déjà coloriés correspondent aux fenêtres

(**) $[i, i+m-1], ..., [i+(k-2)m+1, i+(k-1)m-1]$, $i = 1, ..., n$.

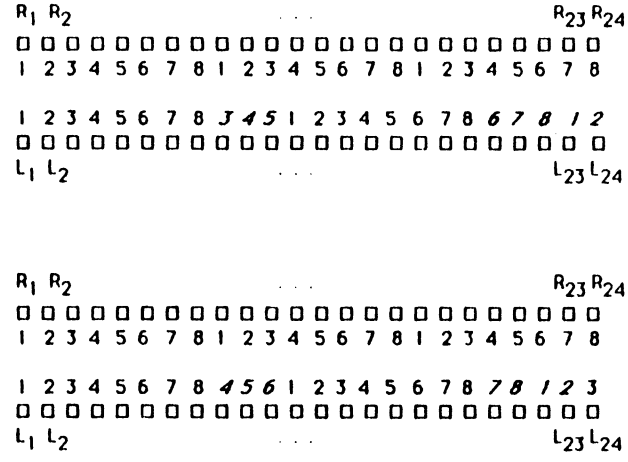


Figure 6c. — Phase 2. Les deux options de l'étape 3. $n = 8$ et $m = 3$.

On peut donc calculer les coefficients de C correspondant aux couples (i, j) tels que $0 \leqslant j - i \leqslant (k - 1)m - 1$. En conséquence, il suffit d'affecter maintenant aux L-segments restants, des couleurs de manière à ce que, pour $i = 1, ..., n$, le L-segment de couleur i corresponde à une fenêtre $[j, j + m - 1]$ contenant $[i + (k - 1)m, i + n - 1]$. D'après les propriétés (*) et (**) ci-dessus cela est possible et il suffit pour cela de respecter la contrainte $i + (k - 1)m - r \leqslant j \leqslant i + (k - 1)m - 1$ où $km = r + n$.

Présentation détaillée de la phase 3

On découpe la grille G dans le sens de la largeur en deux parties G' et G'' de hauteur nk. Comme nous le verrons plus loin, ces deux sous-grilles permettent de déterminer les deux sous-régions D' et D'' de D. De plus, pour m impair, on s'arrange pour que la frontière inférieure de G'' comporte $(m - 1)/2$ points. On remarque alors que G' et G'' se déduisent l'une de l'autre par translation si et seulement si nk est pair.

Cas 1 : m est impair. On considère une région D telle que $H = nk$. Si G' et G'' se déduisent l'une de l'autre par translation, alors on peut les affecter à D en faisant glisser G'' d'un cran dans la partie triangulaire de D (*fig. 7a*) ; dans le cas contraire, on les affecte à D de manière à ce qu'elles remplissent la partie rectangulaire (*fig. 7b*).

Cas 2 : m est pair. Si G' et G'' se déduisent l'une de l'autre par translation (ce qui correspond au cas où nk est pair), alors on les affecte à une région D telle que $H = nk + 1$ (*fig. 7c*) ; dans le cas contraire (ce qui correspond au cas où nk est impair), on les affecte à une région D telle que $H = nk$ et elles remplissent alors la partie rectangulaire de D (*fig. 7d*).

Remarques :

1. Lorsque $k = 1$, on a $m = n$, et, compte tenu du résultat de [Melkemi 84] qui dit que $T(n, n) = 3n - 2$

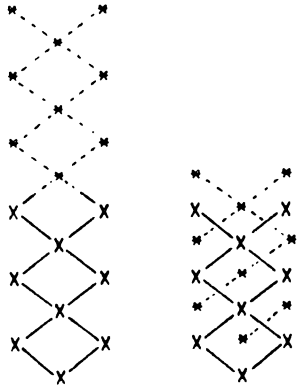


Figure 7*a*. — $m = 3$ (impair) et $nk = 6$ (pair).

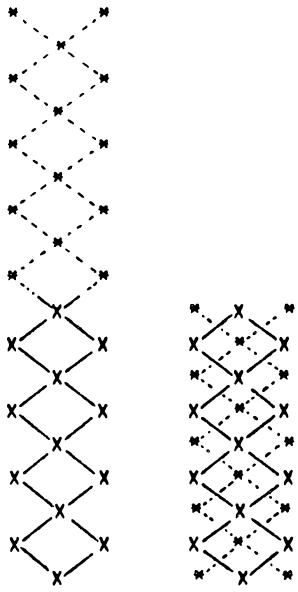


Figure 7*b*. — $m = 3$ (impair) et $nk = 9$ (impair).

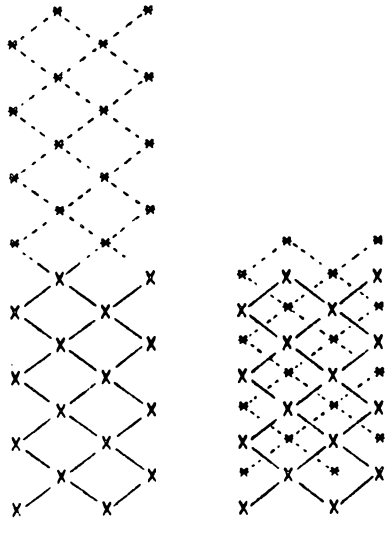


Figure 7*c*. — $m = 4$ (pair) et $nk = 8$ (pair).

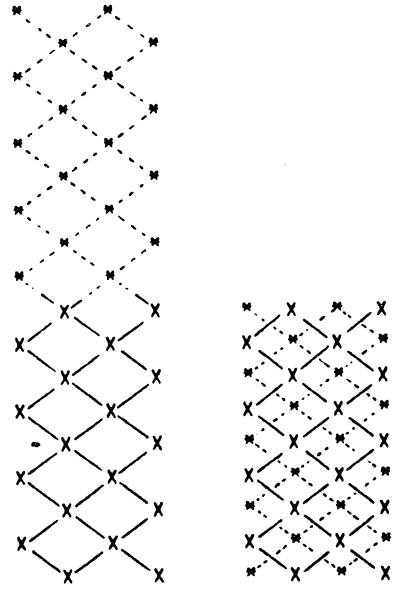


Figure 7*d*. — $m = 4$ (pair) et $nk = 9$ (impair).

(resp. $3n - 1$) si n est impair (resp. pair), la borne supérieure de la proposition 2 est atteinte.

2. Dans le cas général, l'algorithme de la proposition 2 est optimal à $m - f$ près, ce qui est largement satisfaisant dans le cas où n est très grand par rapport à m. Néanmoins, le problème de construction des algorithmes optimaux c'est-à-dire qui s'exécutent en temps $T(n, m)$ reste intéressant ne serait-ce que sur le plan théorique. Dans les deux paragraphes qui suivent, nous illustrons sur les cas $m = 2$ et $m = 3$, des techniques qui permettent d'affiner l'algorithme de la proposition 2 pour obtenir des solutions optimales.

6. Algorithmes optimaux pour les réseaux d'ordre $n \times 2$

DÉFINITION : *Une suite $(i_1, j_1), (i_2, j_2), \ldots, (i_s, j_s)$ est appelée chaîne alternée si deux éléments consécutifs $(i_r, j_r), (i_{r+1}, j_{r+1})$ appartiennent alternativement à la même ligne et à la même colonne. Si de plus $i_1 = i_2$ (resp. $j_1 = j_2$) alors nous dirons que la chaîne est de type 1 (resp. 2).*

Il convient de noter que si s est impair alors la chaîne est à la fois de type 1 et de type 2 selon le sens de parcours que l'on adopte.

Exemples :

(1, 1), (1, 3), (2, 3), (2, 5), est de type 1.

(1, 3), (2, 3), (2, 4), (5, 4), (5, 6) est de type 1 et de type 2.

Dans le cas d'un réseau comportant deux colonnes, la formulation combinatoire présentée dans la proposition 1, peut être précisée de la manière suivante :

PROPOSITION 3 : *La programmation en temps*

$$T = n + H,$$

du produit $C = A.B$ de deux matrices carrées d'ordre n sur un réseau d'ordre $n \times 2$, revient à recouvrir, les couples

(i, j), $1 \leqslant i, j \leqslant n$ par deux chaînes alternées de longueurs H dont l'une est de type 1 et l'autre de type 2.

Démonstration : Considérons un programme qui s'exécute en temps $T = n + H$. Comme le réseau comporte deux colonnes, ceci revient à affecter le couple (i, j).

$1 \leqslant i, j \leqslant n$ à un rectangle d'ordre $H \times 2$ avec les contraintes de la proposition 1. On remarque alors que ces couples peuvent être partitionnés en deux chaînes alternées dont l'une est de type 1 et l'autre de type 2 (*fig.* 8).

Figure 8. — La chaîne de type 1 est représentée par des étoiles ; la chaîne de type 2 est représentée par des croix.

Nous sommes maintenant en mesure d'établir des résultats sur les algorithmes optimaux associés aux réseaux d'ordre $n \times 2$.

PROPOSITION 4 :

(i) $T(2, 2) = 5$.
(ii) *Si $n > 2$ est pair alors* $T(n, 2) = n + n^2/2$.
(iii) *Si $n > 1$ est impair alors* $T(n, 2) = (n^2 + 3n)/2 - 1$.

Démonstration :

(i) Facile à vérifier.
(ii) Dans le lemme précédent, le rectangle de taille $H \times 2$ doit pouvoir contenir les couples (i, j), $1 \leqslant i, j \leqslant n$; en conséquence, $H \geqslant n^2/2$ et

$$T(n, 2) = H + n + m - 2 \geqslant n + n^2/2 .$$

Pour montrer que cette borne est atteinte, il suffit de recouvrir toute matrice carrée d'ordre $n \times n$, n pair avec deux chaînes alternées de longueurs $n^2/2$. Le recouvrement s'obtient comme généralisation immédiate des cas $n = 8$ et $n = 10$ traités ci-dessous (*fig.* 9).

(iii) On vérifie qu'il faut au moins $(n + 1)/2$ segments pour stocker une ligne de C, donc

$$N(L) \geqslant n(n + 1)/2$$

et

$$T(n, 2) = N(L) + n - 1 \geqslant (n^2 + 3n)/2 - 1 .$$

Pour montrer que cette borne est atteinte il suffit de considérer les recouvrements qui généralisent ceux de la figure 9.

7. Algorithmes optimaux pour les réseaux d'ordre $n \times 3$

Dans ce paragraphe, nous exhibons des algorithmes optimaux pour les réseaux d'ordre $n \times 3$.

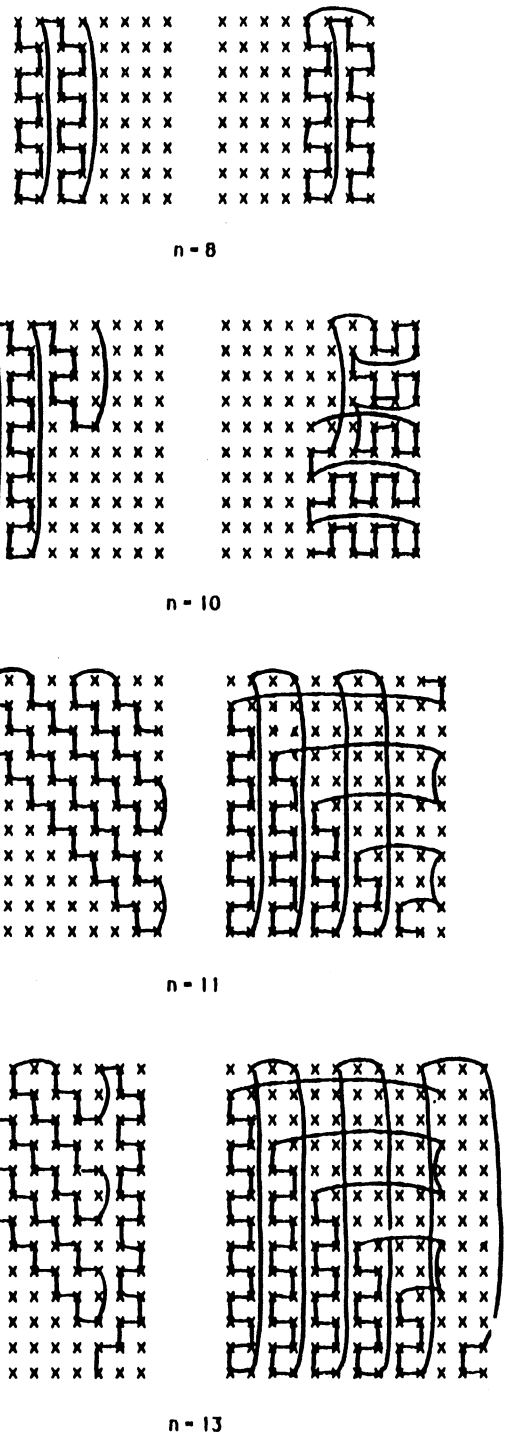


Figure 9.

PROPOSITION 5 :

(i) *Si $n = 3k$ alors* $T(n, 3) = n(k + 1)$
(ii) *Si $n = 3(k - 1) + 2 = 3k - 1$ alors*

$$T(n, 3) = n(k + 1) .$$

Démonstration :

(i) Soit D la région associée à un algorithme optimal ; elle contient exactement $3H + 1$ points donc $3H + 1 \geqslant n^2$, ce qui entraîne $H \geqslant (n^2 - 1)/3$ et $T(n, 3) = H + n + m - 2 \geqslant n(k + 1) + 1$. Par ailleurs,

la proposition 2 dit que $T(n, 3) \leqslant n(k + 1) + 1$ d'où $T(n, 3) = n(k + 1) + 1$.

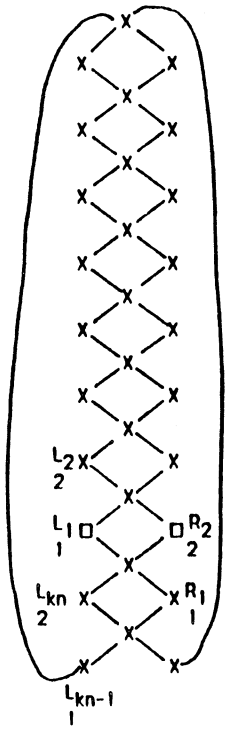
(ii) Comme le L-segment en haut à gauche est de longueur 1 la ligne de C qui la contient nécessite au moins $k + 1$ segments pour être stockée, d'où

$$N(L) \geqslant nk + 1 \text{ et } T(n, 3) = N(L) + n - 1 \geqslant n(k + 1).$$

Pour montrer que cette borne est atteinte, reprenons la démonstration de la proposition 2. A la fin de la deuxième étape de la phase 2, les segments déjà coloriés permettent le calcul de tous les coefficients c_{ij} tels que $j = i, i + 1, \ldots, i + n - 3$ (modulo n). En conséquence, il faut colorier les L-segments restants de manière à obtenir les couples $(i, i + n - 2)$ et $(i, i + n - 1)$, pour $i = 1, 2, \ldots, n$. Or on dispose pour cela de n fenêtres $[i, i + 3]$, $i = 1, 2, \ldots, n$. Associons alors, pour $i = 1, \ldots, n$, la couleur i au L-segment correspondant à la fenêtre $[i + n - 2, i + n]$ (addition modulo n). Tous les couples $(i, i) = (i, i + n)$ figurent alors en double. Examinons en détail la situation au voisinage de L_1 et R_1. On remarque alors (*fig.* 10) que :

— le couple de couleurs (1, 1) apparaît en double et les points correspondants sont à l'intersection de L_{kn-1} avec R_1 et de L_1 avec R_1 ;

— le couple de couleurs (2, 2) apparaît en double et les points correspondants sont à l'intersection de L_{kn} avec R_2 et de L_2 avec R_2.



les deux points que l'on peut enlever sont représentés par □

Figure 10.

En conséquence, nous pouvons supprimer dans la grille les points (L_1, R_1) et (L_{kn}, R_2). On se retrouve alors avec une grille comportant $3nk - 2$ points et ayant la forme

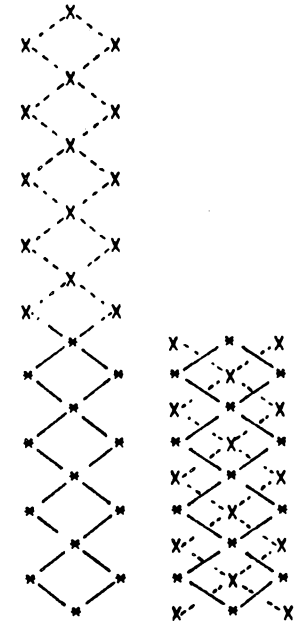


Figure 11.

illustrée à la figure 11. Coupons cette grille dans le sens de la largeur, en deux parties de hauteurs sensiblement égales. On vérifie (*fig.* 11) que l'on peut replacer les deux sous-grilles obtenues dans une région D de hauteur $H = nk - 1$. On obtient ainsi un algorithme qui s'exécute en temps $T = n(k + 1)$.

Remarque : Dans l'exemple traité à la figure 12, on a $n = 5$, $m = 3$ et on voit bien que la méthode exposée permet d'obtenir un algorithme optimal.

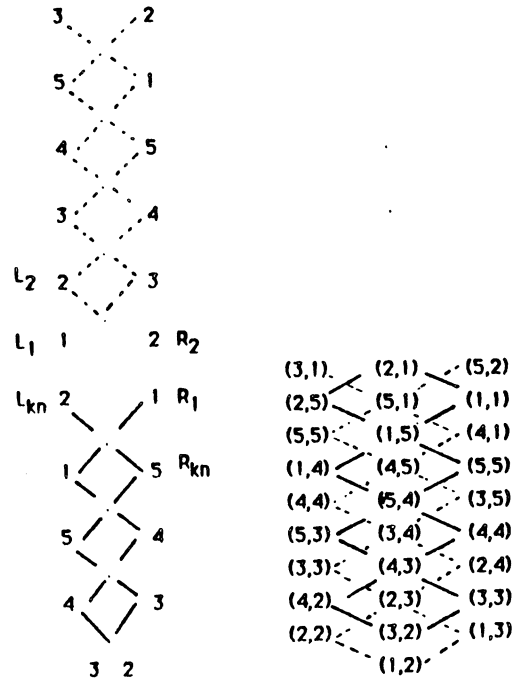


Figure 12.

PROPOSITION 6 : *Si* $n = 3k + 1$ *alors*

$$T(n, 3) = n(k + 2) - 1.$$

Démonstration : Il faut au moins $k + 1$ segments pour stocker toute ligne de C donc $N(L) \geqslant n(k + 1)$ et $T(n, 3) = N(L) + n - 1 \geqslant n(k + 2) - 1$. Considérons une région D telle que $N(L) = n(k + 1)$. Après avoir affecté aux segments les numéros 1, 2, ..., $N(L)$ de haut en bas, on les classe dans l'ordre 1, 3, 5, ..., 2, 4, 6, ..., c'est-à-dire celles de rang impair d'abord suivies de celles de rang pair. On adopte ensuite une technique de coloration semblable à celle utilisée pour la démonstration de la proposition 2. Pour faciliter la compréhension de la démonstration, nous introduisons le graphe biparti $F = (V, E)$ où V est l'ensemble des segments, et il existe une arête entre deux segments L_i et R_j si et seulement s'ils ont un point commun. Ce graphe F a la forme indiquée à la figure 13.

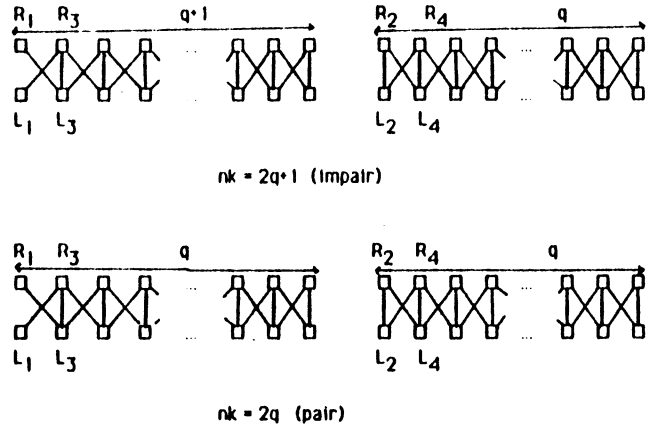


Figure 13.

Cas 1 : $k = 2r$.

Posons $q = 3r$, et considérons le tableau ci-dessous :

1357...				2468...
12 ... n1 ... n ...			1 ... n1 ... qn	$(q + 1)(q - 2)$ ... n1 ... n ... 1 ... n1 ... $(n - 1)$
× 12 ... n × × × 1 ... n × × × ... 1 ... n × × ×				× 12 ... n × × × 1 ... n × × × ... 1 ... n × ×
3	245	678	$q(q + 1)$ 1	$(q + 2)(q + 3)(q + 4)(q + 5)$ ... $(n - 1)n$

Ce tableau donne les indications suivantes :

— en première ligne, les indices des segments ;
— en deuxième ligne, les couleurs des R-segments ;
— dans les lignes 3 et 4, les couleurs des L-segments.

La preuve de validité est la suivante :

— comme dans la preuve de la proposition 2, les couleurs des lignes 2 et 3 permettent de calculer les coefficients c_{ij} tels que $j = i, i + 1, ..., i + n - 2$ (modulo n), c'est-à-dire tels que $j \neq i - 1$ (modulo n) ;
— les L-segments coloriés en ligne 4 correspondent de gauche à droite, aux fenêtres suivantes :

$\{2\}, [j, j + 2]$ $1 \leqslant j \leqslant q - 2$, $\{q - 1, q, n\}, \{q, n\}$ pour ceux de rang impair et $[q + 1, q + 2], [j, j + 2]$, $q + 1 \leqslant j \leqslant n - 3$, $[n - 2, n - 1]$ pour ceux de rang pair.

On vérifie aisément qu'en leur affectant de gauche à droite les couleurs 3, 2, 4, 5, 6, ..., q, $q + 1$, 1 (pour ceux de rang impair) et $q + 2, q + 3, ..., n$ (pour ceux de rang pair), on peut calculer les c_{ij} tels que $j = i + n - 1 = i - 1$ (modulo n).

Cas 2 : $k = 2r + 1$.
Posons $q = 3r + 2$.

On procède de manière analogue au cas 1 en suivant le tableau ci-dessous :

Les exemples de la figure 14 donnent une illustration de la coloration des sommets du graphe $F = (V, E)$ pour $n = 7$ et $n = 4$.

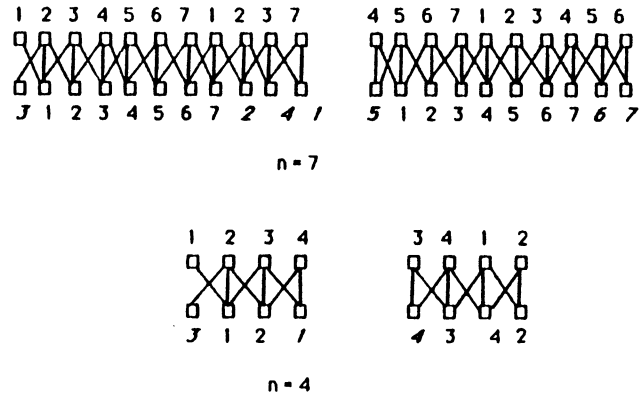


Figure 14.

13579...				2468...
12 ... n12 ... n ...			12 ... n	$(n - 1)n$12 ... n12 ... n ... 12 ... $(n - 2)$
× 12 ... n × × × 12 ... n × × × ... 1 ... q ×				× $(q + 1)(q + 2)$ ... n × × × 12 ... n × × × ... × × × 1 ... n ×
3	245	... 678	... 1	n $(q + 1)(q + 2)(q + 3)$ ... $(n - 1)$

8. Conclusion

Avec le développement des réseaux systoliques composés de cellules programmables [Fischer 83], l'une des branches du calcul parallèle consistera à écrire des programmes adaptés à de telles architectures. Une telle programmation consiste à définir la manière dont les données doivent être présentées aux portes d'entrée du réseau, au cours du déroulement de l'algorithme. Les bornes que nous avons établies dans le cas général, et les algorithmes optimaux exhibés dans les cas $m = 2, 3$ illustrent la puissance de la formulation combinatoire que nous avons introduite pour le calcul du produit matriciel sur un réseau systolique rectangulaire. Les différentes approches utilisées pour la construction d'algorithmes optimaux peuvent manifestement être étendues au cas $m > 3$. Néanmoins, le problème de déterminer la formule générale de $T(n, m)$ reste ouvert et nécessite probablement l'introduction d'outils combinatoires plus puissants que ceux présentés ici.

BIBLIOGRAPHIE

[Fischer 83] A. L. Fischer, H. T. Kung, L. M. Monier, H. Walker, Y. Dohi : *Design of the PSC : a Programmable Systolic Chip* ; Proc. Third Caltech Conf. on VLSI, R. Bryant ed., Computer Science Press Inc., March 1983, 287-302.

[Guibas 79] L. J. Guibas, H. T. Kung et C. D. Thompson : *Direct VLSI Implementation of Combinatorial Algorithms* ; Proc. Conf. on VLSI : Architecture, Design, Fabrication, Caltech, January 1979, 509-525.

[Kung 82] H. T. Kung : *Why Systolic Architectures* ; Computer Magazine, **15** (1), January 1982, 37-46.

[Kung 80] H. T. Kung et P. L. Lehman : *Systolic VLSI Arrays for Relational Data Base Operations* ; Proc. Int. Conf. on Management of Data, ACM-Sigmod, 1980, 105-116.

[Kung 79] H. T. Kung et C. E. Liserson : *Systolic Arrays for VLSI* ; in *Introduction to VLSI*, C. A. Mead and L. A. Conway ed., Addison Wesley, section 8.3, 37-46.

[Melkemi 84] L. Melkemi et M. Tchuente : *Complexity of Matrix Product on Orthogonally Connected Systolic Arrays* ; Rapport de Recherche Imag, April 1984, Grenoble.

[Mongenet 85] C. Mongenet : *Une méthode de conception d'algorithmes systoliques : résultats théoriques et réalisation* ; Thèse INPL Nancy, mai 1985.

[Preparata 80] F. P. Preparata et J. Vuillemin : *Area-time Optimal VLSI Networks for Multiplying Matrices* ; Information Processing Letters, **11** (2), 1980, 77-80.

[Quinton 83] P. Quinton : The Systematic Design of Systolic Arrays ; Rapport de Recherche N° 193, Irisa, April 1983.

[Robert 83] Y. Robert et M. Tchuente : Une approche Divide-and-Conquer pour des algorithmes systoliques ; Rapport de Recherche Imag n° 393, Grenoble, septembre 1983.

Article reçu le 8 juin 1984, version révisée le 8 février 1985.

I/O-TIME TRADEOOFS FOR MATRIX PRODUCT ON PROGRAMMABLE ARRAYS

Lamine MELKEMI and Maurice TCHUENTE
CNRS/INPG Laboratoire TIM3
BP 68. 38402 Saint Martin d'Hères

Abstract

We study I/O-time tradeoffs for the computation of the product C = A.B of two matrices on a two dimensional array of programmable cells. In the particular case where A and B are square matrices of order n, and the array is of order nxn, we exhibit algorithms which run in time 3n-2, 2.5n-2 and 2n-2 and whose I/O requirements are 2n, 3n and 4n respectively.

Résumé

Dans cet article, nous étudions le compromis entre d'une part le temps de calcul et d'autre part le nombre d'accès-mémoires pour le calcul du produit C = A.B de deux matrices sur la classe des réseaux bi-dimensionnels composés de cellules programmables. Dans le cas particulier où A et B sont des matrices carrées d'ordre n, nous présentons des algorithmes qui s'exécutent en temps 3n-2, 2.5n-2 et 2n-2 et dont les nombres d'accès-mémoires sont égaux respectivement à 2n, 3n et 4n.

A paraître dans les actes du colloque CONPAR85, Allemagne 1985

1. INTRODUCTION

Let A and B be two square matrices of order n. The sequential computation of any coefficient c_{ij} of the product C = A.B can clearly be realized according the following loop:

$c_{ij} := 0;$
for k := 1 to n do $c_{ij} := c_{ij} + a_{ik}.b_{kj}$.

Such a computation can be performed in n steps by the multiply-add cell illustrated in Figure1 below:

Figure1

Clearly, if we consider a collection of n^2 such cells, then C can be computed in n steps. However, such a parallel sheme is not realistic because it requires a simultanous access to $O(n^2)$ elements.

In this paper, we are interested in parallel shemes associated with square arrays of order nxn, and whose I/O requirements grow linearly with n.

2. PREVIOUS RESULTS

Many algorithms for the computation of the product C = A.B of square matrices of order n, on regular arrays of multiply-add cells have been proposed in the literature [3, 4, 5, 8, 9]. The best of these algorithms is due to the authors[5]: it runs in time 3n-2 on an array of size nxn, and requires the simultanous access to 3n elements.

On the other hand, a parallel sheme associated with a programmable array of size nxn has been proposed by Guibas, Kung and Thompson[1], Preparata and Vuillemin[2], and Kanota[3]. This sheme is illustrated in

Figure2, and works as follows:

Phase1

the cell at position (i,j) performs the n accumulation $c_{ij} := c_{ij} + a_{ik} \cdot b_{kj}$ necessary for the computation of c_{ij}. This accumulation starts at time i+j-2 and terminates at time n + i + j - 2.

Phase2

This phase starts at time 3n-2 and ends at time 4n-2; during this phase the elements c_{ij} are pumped leftwards (or upwards) in parallel, in order to be output from the array.

The drawback of this sheme is that the cell at position(n,n) is the farthest from the I/O pins and is the last to start its computation.

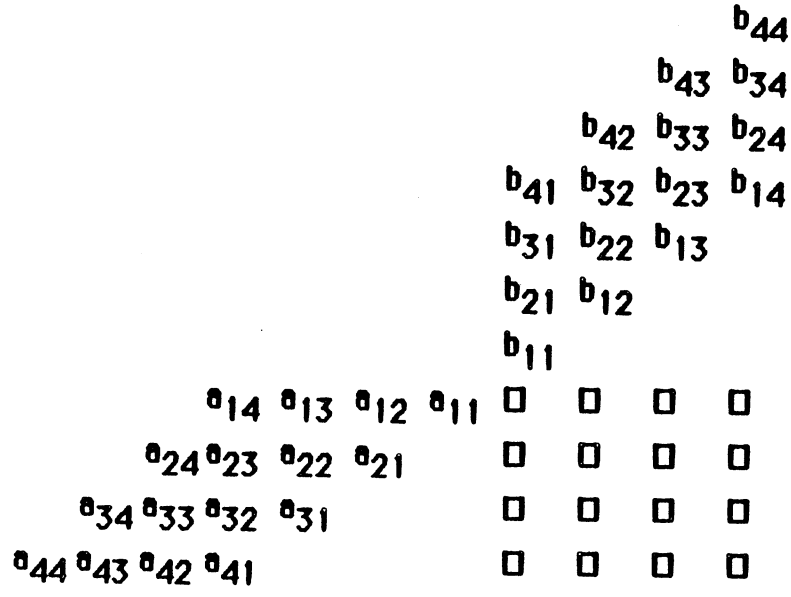


Figure 2

The modified sheme presented here, runs faster because it allows the computations of any cell to start as soon as possible.

3. PRODUCT OF SQUARE MATRICES

Proposition1

The product C = A.B of two matrices A and B of order n can be computed in time 3n-2 on an array of size nxn composed of programmable cells, and with only 2n I/O pins.

The corresponding sheme is illustrated in Figure3. In this sheme, the computation of the cell at position (i,j) starts at time max(i,j)-1 and terminates at time n+max(i,j)-1. As a consequence, the second phase can start at time 2n-1. this shows that the algorithm runs in time 3n-2.

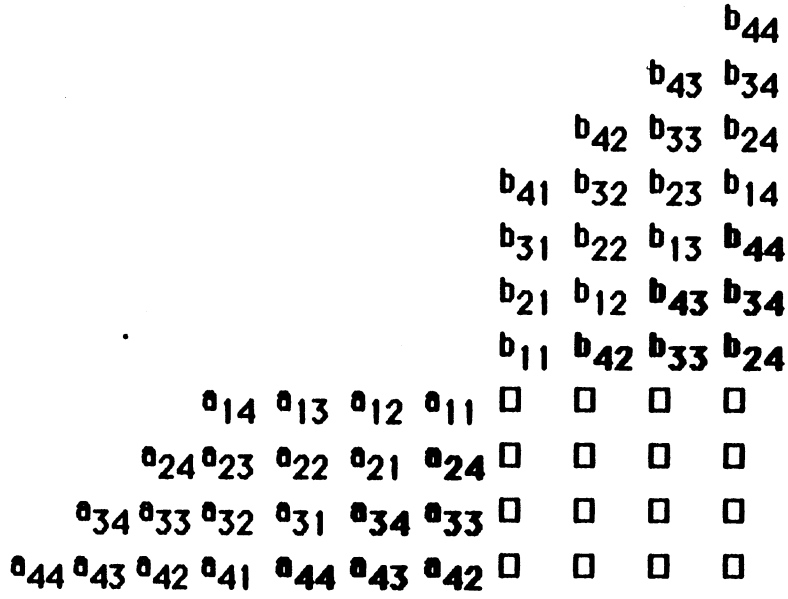


Figure 3: **the elements which do not appear in the solution of Figure 2 are darkened.**

It is important to note that the control of such network is easily implemented: we assign the bit 1 to any variable a_{ik} (resp.b_{kj}) which has been already received by the array, and we assign 0 to the others. Clearly, the cell at position (i,j) contains two 1's only at time T= n + max(i,j) -1. As a consequence, any cell is prevented from performing more than n accumulations whenever it contains two 1's .

The completion time of the above algorithm, illustrated in figure4, shows that there is not any overlapping if the cell at position (i,j) sends its result at time n + max(i,j) - 1 to that of position (i-1,j).

4	5	6	7
5	5	6	7
6	6	6	7
7	7	7	7

Figure 4: **The completion time of the systolic algorithm of Figure 3.**

Clearly, with this sheme, it is possible to compute the product C = A.B of two matrices A and B of orders nxq and qxn respectively, in time 2n + q -2 on an array of size nxn, which requires the simultanous access to 2n elements. This remark will be very useful for the proof of the forthcoming results.

Poposition2

The product C = A.B of two square matrices A and B of order n can be computed in time 2.5n - 2 on an array of size nxn composed of programmable cells, and with only 3n I/O pins.

The corresponding algorithms are obtained as follows: consider the decomposition $B = [B_1, B_2]$; $C = A.B = [A.B_1, A.B_2]$ and it is easily seen from the preceding remark, that $A.B_1$ and $A.B_2$ can be computed independently in time 2.5n - 2, on the subarrays illustrated in Figure5.

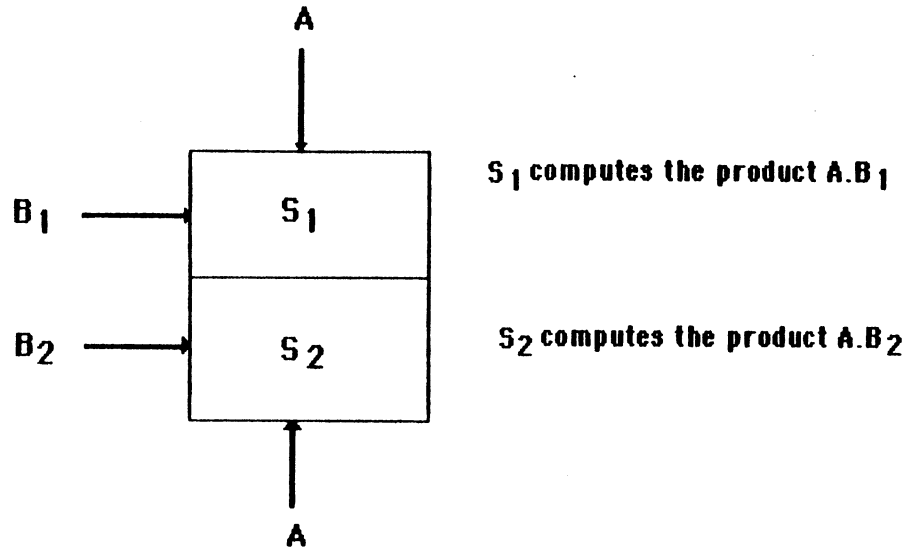


Figure 5

Proposition3

The product C = A.B of two square matrices of order n can be computed in time 2n - 2 on an array of size nxn composed of programmable cells, and with 4n I/O pins.

The idea is similar to the proof of proposition2 and is illustrated in Figure6.

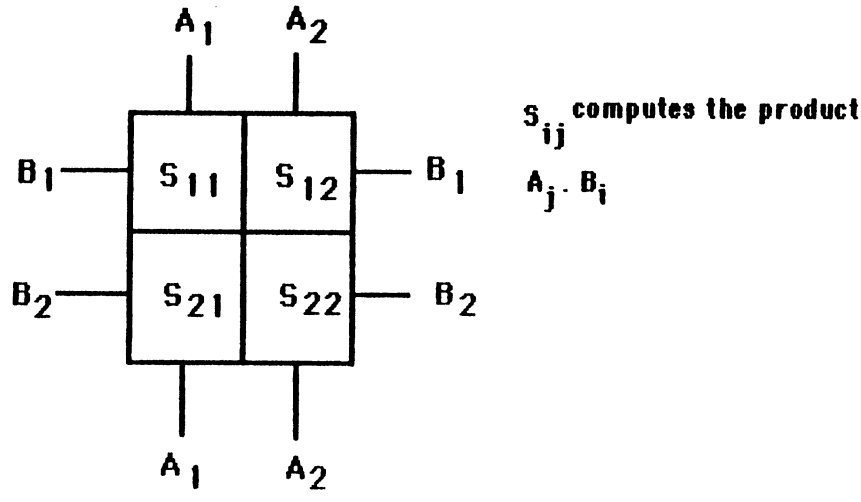


Figure 6 $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$

4- PRODUCT OF TRIANGULAR MATRICES

Proposition4

The product of two triangular matrices of order n can be computed in time 2n - 2 on an array of order nxn, with 2n I/O pins.

The algorithm is descibed in Figure7. Note that only half the cells are active, so the algorithm can run on a triangular array.

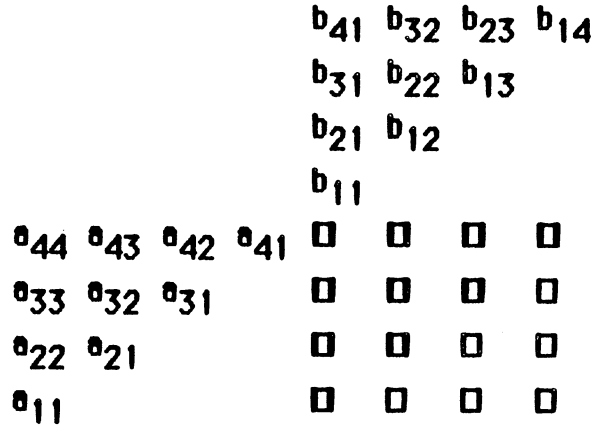


Figure 7: **The active cells are darkened.**

Proposition5

The product of two triangular matrices of order n can be computed in time 1.5n - 1 on an array of order nxn, with 3n I/O pins.

The corresponding algorithm is obtained by applying the combinatorial approach developped in [5]. The segments of slope +1 above the array store the elements of C belonging to the rows numbered 1, 2, ..., n, n, n-1, n-2, ... from bottom to top; similarly, the segments of slope -1 above the array store elements of C belonging to the columns numbered n, n-1,...,2,1,1, 2,... from bottom to top. The variables a_{ik} (resp. b_{kj}), $1 \le k \le n$, are assigned to the segment of slope +1 (resp.-1) which contains c_{ij} and is at the left (resp. right) side of the array ; in Figure 7, this construction is illustrated for i=5 and j=1.

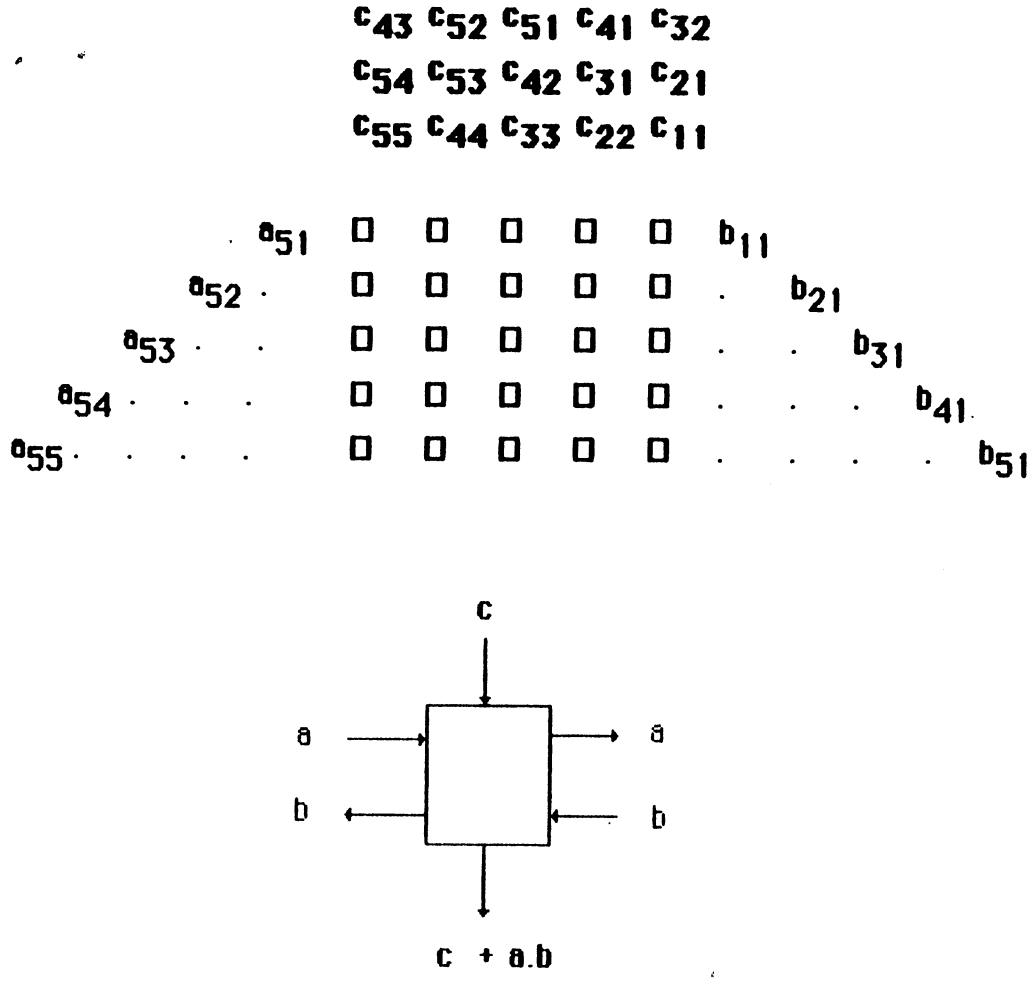

c_{43} c_{52} c_{51} c_{41} c_{32}
c_{54} c_{53} c_{42} c_{31} c_{21}
c_{55} c_{44} c_{33} c_{22} c_{11}
a_{51} b_{11}
a_{52} b_{21}
a_{53} b_{31}
a_{54} b_{41}
a_{55} b_{51}
c
a
a
b
b
c + a.b

Figure 8

5- CONCLUSION

In this paper, we have studied I/O-time tradeoffs for the computation of the product of two matrices A and B of order n on a two dimensional programmable array of size nxn. More precisely, we have exhibited algorithms which run in time 3n - 2, 2.5n - 2 and 2n - 2 and whose I/O requirements are 2n, 3n and 4n respectively.

The class of networks where the results c_{ij} , as well as the data variables a_{ik} and b_{kj}, flow in a pipeline way have been considered in the literature; recently, the authors[5] have proved that the minimum time that can be realized on a rectangular array of size nxn is 3n - 2.

Thus, when comparing these two diferent shemes, one may conclude that the programmable arrays are faster and have a more complicated control.

In the fourth section, we have presented two efficient algorithms for the computation of the product of triangular matrices. The first algorithm is realized on a triangular network of size n(n+1)/2 , and runs in time 2n - 1; while the second is realized on a two-dimensional one of size nxn and run in time 1.5n - 1.

In the case where any cell, in a square array of side n, performs exactly n accumulations; it is easily seen that the optimal algorithm takes at least 2n-2 times of unit. As a consequence, it is worthwile to analyse the time complexity that can be acheived on a rectangular array of size nxn

REFERENCES

[1] **Guibas. L. J, Kung. H. T and Thompson. C. D**
Direct VLSI implementation of combinatorial algorithms
Proc. Calt. Conf. on VLSI 509-521 (1979)

[2] **Kanota. E**
Cellular algorithms for binary multiplication
Cell Processors and Cell Algorithms; Vollmar et al eds
(Braunschweig 1981)

[3] **Kung. H. T and Lehman. P. L**
Systolic VLSI arrays for relational data base operations
Proc. Int. Conf. Manag. of Data, 105-116 (ACM-SIGMOD 1980)

[4] **Kung. H. T and Leiserson. C. E**
Systolic arrays for VLSI
Introduction to VLSI systems; Mead. C. A and Conway. L. A eds
Adison Wesley 1980

[5] **Melkemi. L and Tchuente. M**
Complexity of matrix product on orthogonally connected
systolic arrays
Submitted to IEEE. Transactions on Computers

[6] **Melkemi. L and Tchuente. M**
Programmation du produit matriciel sur un réseau
systolique rectangulaire
TSI (1985)

[7] **Preparata. F. P and Vuillemin. J**
Area-time optimal VLSI networks for multiplying matrices
Inf. Proc. Letters. 11, 2 77-80 (1980)

[8] **Robert. Y and Tchuente. M**
Une approche divide-and-conquer pour des algorithmes systoliques
R. R. IMAG. N° 393 (1983)

[9] **Weiser. U and Davis. A**
Mathematical representation for VLSI arrays
Report UUCS-8-111, University of Utah (1980)

CHAPITRE 3

Détection en temps linéaire des carrés d'un mot par des réseaux systoliques

1- INTRODUCTION

La reconnaissance des mots sans carrés est un problème classique qui a été étudié par de nombreux auteurs[1,2,3,4,8]. Le lecteur intéressé par les résultats les plus récents peut se referer à l'excellent article de synthèse de Berstel[3].

Récemment, Apostolico et Negro [1] ont proposé un réseau unidimensionnel, basé sur le réseau de reconnaissance de formes introduit pár Kung[7] et Foster et Kung[6], qui détecte en temps linéaire toutes les répétitions dans un mot; en conséquence, il peut reconnaitre en temps linéaire les mots sans carrés. Toutefois, la cellule de base de leur réseau est équipée d'un compteur initialisé à n, la longueur du mot à étudier. L'inconvénient est que ceci augmente la surface du réseau et réduit sa modularité. En effet, lorsqu'on applique le réseau à des mots de longueurs strictement supérieures à n, on est ramené, à chaque fois, à changer la structure de la cellule type; ce qui est prohibitif.

Dans ce chapitre, nous commençons par présenter un réseau bi-dimensionnel permettant de détecter en temps linéaire toutes les répétitions dans un mot. Contrairement à la solution proposée par Apostolico et Negro [1], ce réseau ne comporte pas de compteur; la structure de la cellule de base est donc indépendante de la longueur du mot. Ensuite nous établissons un lemme sur le chevauchement des carrés dans une chaîne. Ce résultat nous permet de déduire à partir du réseau bidimensionnel précédent, un réseau unidimensionnel pour la reconnaissannce des mots sans carrés.

Un algorithme linéaire de reconnaissance des mots sans carrés a été proposé récemment par Crochemore[5]. L'avantage de la solution présentée ici est sa rapidité lorsqu' elle est réalisée en circuits VLSI.

Soit A un alphabet et soit A^+ le monoide libre généré par A. Un élément de A est appelé caractère, et un élément de A^+ est appelé mot. Un mot x est donc complètement déterminé en écrivant $x = x_1x_2...x_n$ où les x_i sont des caractères. Un sous mot de x est un mot de la forme $x_ix_{(i+1)}...x_j$ avec $1 \le i \le j \le n$.

Un carré de longueur 2p est un mot de la forme $a_1a_2...a_pa_1a_2...a_p$. On dit qu'un mot est sans carré si aucun de ses sous mots n'est un carré.

Exemple 1

A= {a, b, c}

x = abcbacbca est un mot sans carré
x = abc**bacabacab** contient un carré de longueur 8 (le carré est présenté en gras)

Enfin, nous notons par 'blanc' un caractère spécial et par lxl la longueur du mot x, et nous supposons que 'blanc' ≠ 'blanc' .

2- SYNCHRONISATION DES CARACTERES

Description du réseau de base:

Considérons un réseau unidimensionnel simple composé de p cellules C_p, $C_{(p-1)},...,C_2,C_1$ pour la reconnaissance des carrés de longueurs inférieures à 2p+1. Chaque cellule C_i a deux registres A_i et B_i, n'ayant droit de contenir que des caractères ou le symbole spécial 'blanc'. Le registre A_i se situe en haut et le registre B_i se situe en bas.

Soit $x=x_1x_2...x_N$, N=2p, le mot à étudier. On fait défiler dans le réseau deux flux des données (x_i); le flux du haut passe par les registres A_i et celui du bas passe par les B_i. Précisément, le registre A_p reçoit le caractère x_i à l'instant t=i, et le registre B_p reçoit le caractère x_j à l'instant t=j+p.

De plus, sur le flux du haut il y a un registre de retard entre deux cellules consécutives; c'est à dire chaque caractère se trouvant à l'instant t dans une cellule $C_{(i+1)}$ est d'abord reçu par un registre de retard, et c'est à l'instant t+2 qu'il se trouve dans C_i. Par contre, sur le flux du bas, chaque caractère qui se trouve à l'instant t dans la cellule $C_{(i+1)}$ est directement transféré à l'instant t+1 à la cellule C_i.

Cette description est illustrée dans Figure 1.

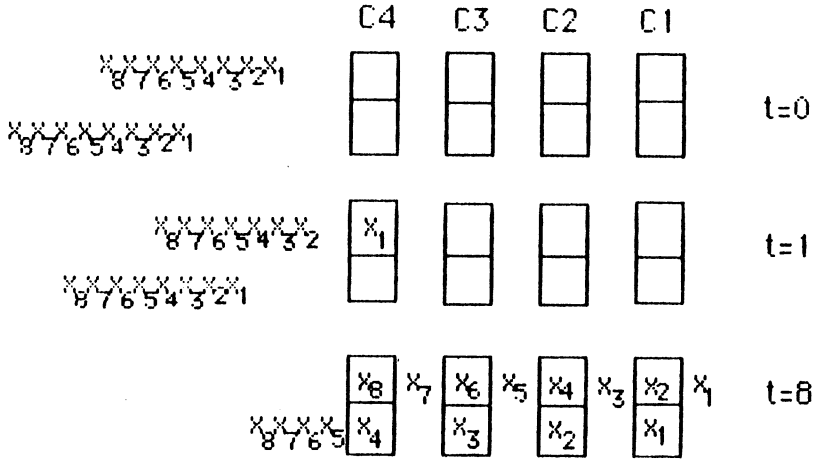


Figure 1

Dans ce qui suit, nous montrons que ces caractères sont bien synchronisés de telle manière qu'on puisse détecter les carrés de x.

Remarques:

1- La cellule C_i effectue les comparaisons $(x_h=x_q)$ avec $(h-q)=i$.

2- Si la cellule C_i détecte i égalités successives alors x contient un carré de longueur 2i.

On peut donc équiper la cellule C_i d'un compteur compt(i) initialisé à i, et obtenir ainsi un réseau permettant de détecter en temps linéaire tous les carrés de x. Dans ce cas, la cellule Ci exécute l'algorithme suivant:

1- Si (C_i détecte une égalité) et (compt(i)≠0) alors compt(i):=compt(i)-1;

2- Si (C_i détecte une égalité) et (compt(i)=0) alors énoncer un carré de longueur 2i;

3- Si C_i détecte une inégalité alors compt(i):=i.

3- DETECTION EN TEMPS LINEAIRE DES CARRES DE x

L'objectif de ce paragraphe est de supprimer le compteur compt(i) ainsi introduit, et le remplacer par un réseau unidimensionnel.

Pour ce faire, on joint à chaque C_i, (i-1) cellules $C_{i2}, C_{i3}, \ldots, C_{ii}$ interconnectées selon le réseau unidimensionnel illustré dans Figure2.

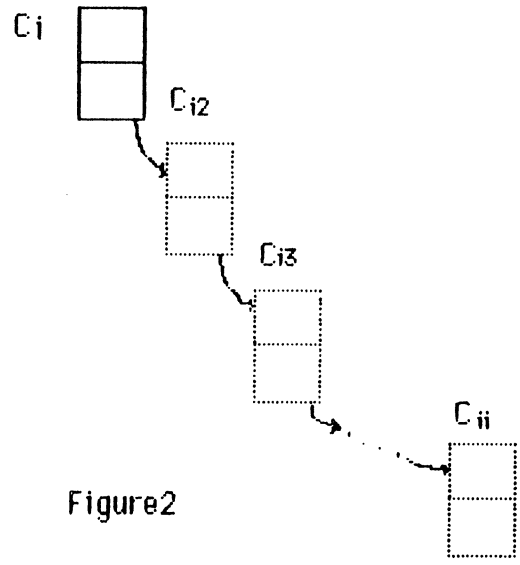


Figure2

Ce réseau se mobilise; bien entendu; sous l'effet des comparaisons effectuées par la cellule C_i. Lorsque C_i détecte une égalité, elle génère le signal 1; dans le cas contraire, elle génère le signal 0 (cf Figure3).

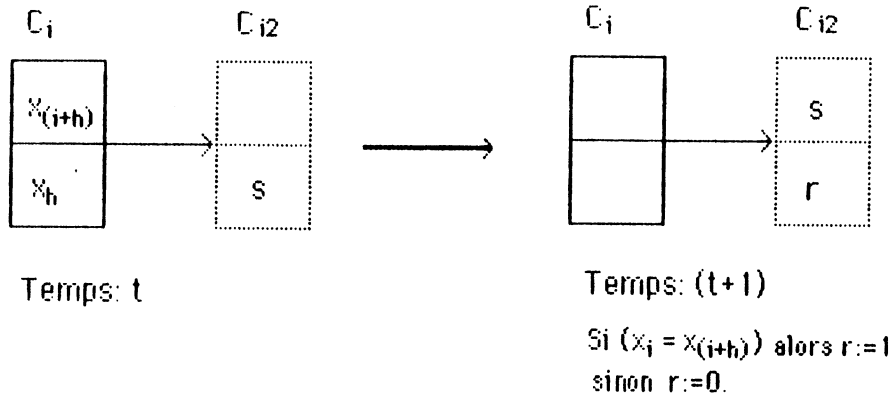


Figure 3

La cellule C_{ij} est menue de deux registres. A l'instant t, elle génère le signal 1 si elle ne contenait à l'instant (t-1) que des 1's, et elle génère le signal 0 dans le cas contraire. Son registre du haut aura la valeur du registre du bas, et celui ci reçoit le signal provenant de sa cellule voisine en entrée $C_{i(j-1)}$ (voir Figure4).

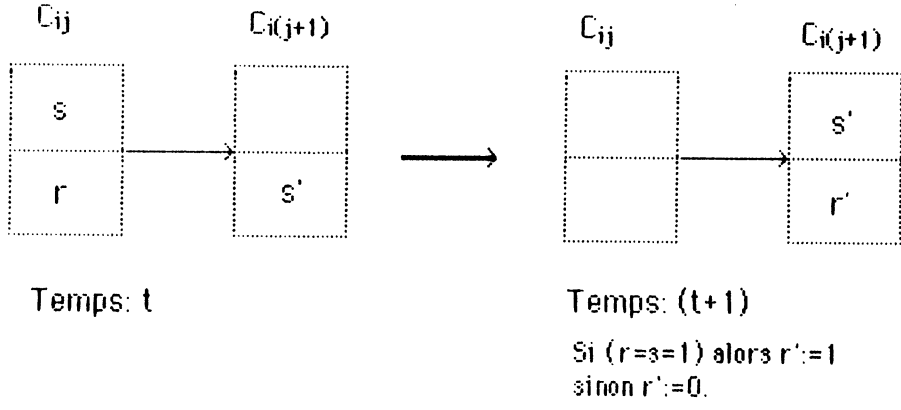


Figure 4

Il nous reste de savoir comment le réseau peut-il détecter les carrés de x. Par convention, on désigne par C_{i1} la cellule C_i; le lemme suivant est une réponse à cette question.

Lemme1:

Si à l'instant t+2p-2, la cellule C_{ip} ne contient que des 1's, alors la cellule C_i a détecté p égalités successives aux instants: t, t+1,..., t+p-1.

Démonstration:

Remarquons d'abord que si à l'instant t+2, une cellule C_{ip} ne contient pas de 0's alors la cellule $C_{i(p-1)}$ a généré deux 1's successifs aux instants t et t+1 (cf Figure 5).

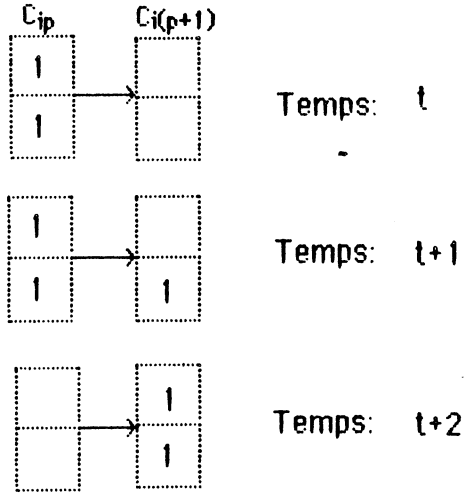


Figure 5

En remplaçant p par 1 dans Figure5, on vérifie aisément que le lemme est vrai pour p=2. Supposons qu'il est aussi vrai pour (p-1) et établissons le pour p.

Si à l'instant t+2p-2 la cellule C_{ip} ne contient pas de 0's alors la cellule $C_{i(p-1)}$ a généré, en vertu de Figure5, deux 1's successifs aux instants t+2p-4 et t+2p-3. En d'autres termes la cellule $C_{i(p-1)}$ ne contient à ces instants que des 1's.

Par hypothèse de récurrence, la cellule C_i a donc détecté (p-1) égalités successives aux instants t, t+1,...,t+p-2, et (p-1) égalités aux instants t+1,t+2,...,t+p-1. Il en résulte que C_i a détecté p égalités successives aux instants t,t+1,t+2,...,t+p-1; ce qu'il fallait établir.□

Il résulte de ce lemme que, si la cellule la plus à droite ne contient pas de 0's alors la cellule C_i a détecté i égalités successives, c'est à dire: le mot x contient un carré de longueur 2i.

On en déduit ainsi, un réseau bidimensionnel capable de détecter en temps linéaire tous les carrés de x (Voir Figure6).

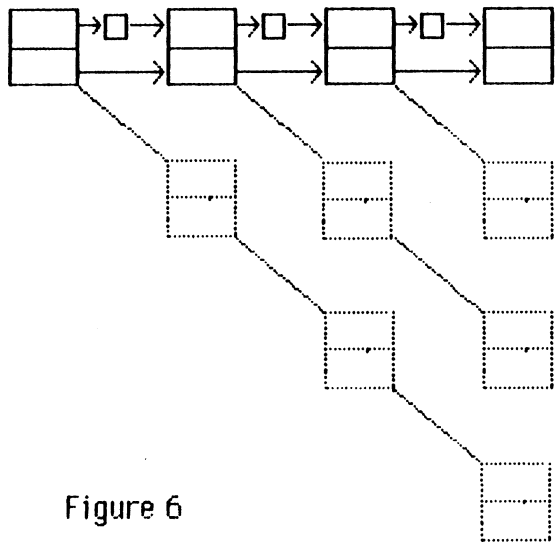


Figure 6

4- RECONNAISSANCE DES MOTS SANS CARRES

Dans ce paragraphe, on se restreint à la recherche en temps linéaire d'un carré dans le mot x; ce problème est équivalent à la reconnaissance en temps linéaire des mots sans carrés. En effet, le réseau souhaitable a à produire un résultat y de type booléen qui est égal à vrai si x contient un carré. La réponse (y=faux) nous permet donc de savoir que x est un mot sans carré. Notre but est de transformer le réseau bidimensionnel à un réseau linéaire et unidimensionnel.

Supposons sans perte de généralité que $N=2^{(m+1)}$, et divisons le réseau bidimensionnel précédent en (m+1) sous-réseaux $S_m, S_{(m-1)}, \ldots, S_1, S_0=C_1$. Chaque S_i représente le réseau composé des cellules:

$$[C_{2^i j}, C_{(2^i-1)j}, \ldots, C_{(2^{(i-1)}+1)j}]$$

où j est un entier variant entre 1 et i.

Voir l'exemple suivant. (Rappelons que $C_{i\,1}$ designe la cellule C_i).

Exemple2:

$$S_3 = \{C_{81}, C_{82}, \ldots, C_{88}, C_{71}, \ldots, C_{77}, \ldots, C_{51}, \ldots, C_{55}\}$$

$$S_2 = \{C_{41}, C_{42}, C_{43}, C_{44}, C_{31}, C_{32}, C_{33}\}$$

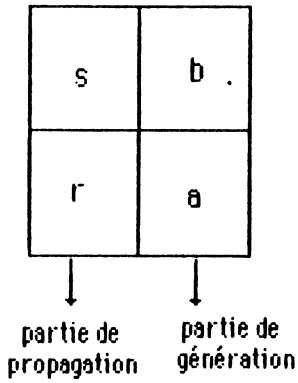
$$S_1 = \{C_{21}, C_{22}\}$$

$$S_0 = \{C_{11}\}$$

Propriété2:

La cellule C_{ij}, i≥j, appartient au sous réseau S_q si et seulement, si $\lceil \log_2 i \rceil = q$.

Pour la simplicité de l'exposé, on combine les deux cellules en faisant joindre l'une à l'autre; on obtient la cellule illustrée dans Figure7.



a, b ∈ AU{blanc} et

r,s ∈ {0,1}.

Figure. 7

Dans ce cas, cette cellule sera composée de deux parties: partie de génération qui effectue les comparaisons et génére les signaux, et partie de propagation qui reçoit et fait propager ces signaux.

Soit E(h,q) l'ensemble suivant:

$$E(h,q)=\{C_{ij};\ i-j=h \text{ et } \lceil \log_2 i \rceil = q\}$$

où h et q sont deux entiers naturels.

L'ensemble E(h,q) désigne tout simplement les cellules qui se situent sur une même verticale et appartiennent au sous réseau S_q.

Exemple3:

$$E(1,1)=\{C_2\},\ E(1,2)=\{C_{32}, C_{43}\},\ E(2,2)=\{C_3, C_{42}\}.$$

Considérons, le réseau unidimensionnel induit du réseau bidimensionnel précédent en faisant projeter et fondre toutes les cellules, qui apprtiennnent à E(h,q) à la cellule $C_{(h+k)k}$ avec $k=\min\{ j;\ C_{ij} \in E(h,q) \}$. Nous redressons les cellules se situant sur un même segment de pente -1 en les alignant sur l'axe horizontal, on obtient donc le réseau illustré dans Figure8

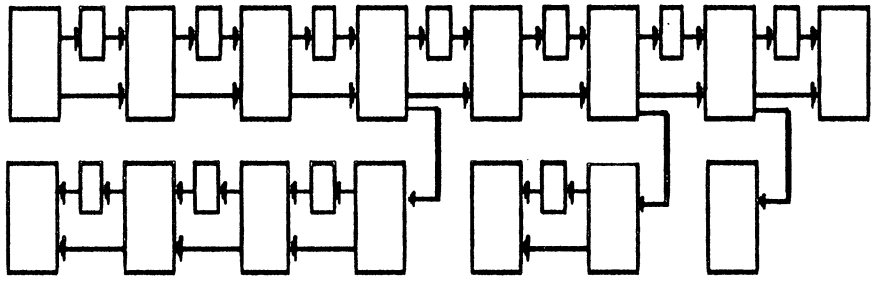


Figure.8

Toutefois si l'on se base sur l'algorithme décrit dans le paragraphe précédent, cette transformation donne lieu à un problème de chevauchement des signaux au cours de la détection des carrés de x. Chaque cellule est en effet contrainte d'utiliser ses deux parties simultanément afin de ne pas détecter un mot débuté par des égalités au détriment d'un carré.

On résoud ce problème, en donnant la priorité au carré ayant la plus grande longueur, et ce par inhiber la partie de génération de C_1 si et seulement si sa partie de propagation contient un signal égal à 1 (cf-Figure9).

Temps: t

Temps: t+1

Si (r=s=0) et (a=b) alors r':=1 sinon r':=r.s.

Figure.9

Cependant, on remarque que, compte tenu de cet algorithme, les caractères qui se trouvent dans une cellule ayant un signal 1 dans l'un de ses registres ne sont pas détectés. Ainsi, le réseau ne peut pas détecter les carrés qui contiennent de tels caractères.

Dans la suite, nous commençons par caractériser les carrés non détectables; puis nous montrons que, dans un mot contenant des carrés, le réseau peut détecter les carrés de longueurs minimales.

Pour cela, on suppose qu'un signal 1 provient originalement des égalités détectées par une cellule $C_{(j-i)}$ et se trouve à l'instant T dans l'un des registres de la partie de propagation de $C_{(j-i-k-1)}$. On vérifie aisément que les cellules $C_{(j-i)}$ et $C_{(j-i-k-1)}$ appartiennent à un même sous réseau; en conséquence:

propriété3:

$$\lceil \log_2(j-i) \rceil = \lceil \log_2(j-i-k-1) \rceil.$$

On envisage deux cas: le cas où ce 1 se trouve dans le registre du bas et le cas où il occupe le registre du haut.

cas1: le signal 1 se trouve en bas.

Posons T=t+2k+1 et désignons par $x_{(i+k)}$ et $x_{(j-1)}$ les caractères qui occupent la partie de génération de la cellule $C_{(j-i-k-1)}$.

On peut vérifier qu'à l'instant t+2k = t+2(k+1)-2, la cellule $C_{(j-i-k)}$ ne contient pas de 0's.

En vertu du lemme1, la cellule $C_{(j-i)}$ a alors détecté (k+1) égalités successives aux instants: t,t+1,...,t+k; ces égalités sont à fortiorie: "$x_i=x_j$", "$x_{(i+1)}=x_{(j+1)}$",...et "$x_{(i+k)}=x_{(j+k)}$".

cas2: le signal 1 se trouve en haut.

Posons T=t+2k+2, et désignons par $x_{(i+k+1)}$ et x_j les caractères qui occupent la partie de génération de $C_{(j-i-k-1)}$ à cet instant.

Il est facile de voir qu'à l'instant t+2k+1, les caractères qui se trouvent dans la partie de génération de $C_{(j-i-k-1)}$ sont $x_{(i+k)}$ et $x_{(j-1)}$; à ce même instant le signal 1 se trouve dans le registre du bas. On retrouve les conditions de cas1.

On peut conclure:

Proposition4:

Un carré ww n'est pas détectable si et seulement, s'il existe trois entiers naturels i, j et k tels que:

1) $\lceil \log_2(j-i) \rceil = \lceil \log_2(j-i-k-1) \rceil$
2) $x_{(i+h)} = x_{(j+h)}$ pour tout $h \in \{0,1,...,k\}$
3) le sous mot w contient, ou x_{j-1}(cas1) ou x_j(cas2)
4) $|w| = j-i-k-1$.

Notons par $x^*=x_N x_{(N-1)}...x_1$ le mot symétrique de x, et remarquons que si l'on remplace, dans cas1, x par x^* , la forme des caractères présentés dans le premier cas est la même que celle des caractères présentés dans le deuxième; et comme x est un mot sans carré si et seulement si x^* l'est, on peut se restreindre, sans perte de généralité, à l'étude du deuxième cas.

Le lemme suivant est le résultat principal de ce paragraphe.

Lemme5:

Soit $x=x_1x_2...x_N$ un mot et soient i, j et k trois entiers naturels vérifiant:

1- $2(k+1) < (j-i)$,
2- $x_{(i+h)} = x_{(j+h)}$ pour tout h dans {0,1,...,k}.

Si x contient un carré de longueur 2(j-i-k-1) qui contient les caractères $x_{(i+k+1)}$ et x_j alors le mot x contient un carré de longueur strictement inférieure à 2(j-i-k-1).

Commentaire:

Dans notre cas, le 1 se trouvant dans $c_{(j-i-k-1)}$ provient de $c_{(j-i)}$, les cellules $C_{(j-i-k-1)}$ et $C_{(j-i)}$ appartiennent donc à un même sous réseau S_q; on a alors (cf propriété3):

$$\lceil \log_2(j-i-k-1) \rceil = \lceil \log_2(j-i) \rceil,$$

ce qui donne: $2(k+1)$ est strictement inférieure à $(j-i)$.

On en déduit ainsi, d'après ce lemme, que si le réseau ne détecte pas les carrés de longueur $2(j-i-k-1)$ et contenant les caractères $x_{(i+k)}$ et $x_{(j-1)}$ (cas1) ou les caractères $x_{(i+k+1)}$ et x_j (cas2), alors le mot x contient un autre carré de longueur strictement inférieure à $2(j-i-k-1)$.

Démonstration du lemme:

Supposons que le mot x contienne un carré sous la forme suivante:

$$(x_{(i+d)}x_{(i+d+1)}\cdots x_{(i+d')})(x_{(j+e)}x_{(j+e+1)}\cdots x_{(j+e')})$$

où d,d',e et e' sont des entiers naturels vérifiant les conditions suivantes:

1- $j+e = i+d'+1$
2- $d \leq (k+1) \leq d'$ (c'est à dire le carré contient $x_{(i+k+1)}$)
3- $e \leq 0 \leq e'$ (c'est à dire le carré contient x_j)
4- $(d'-e')=(d-e)=(k+1)$
5- $x_{(i+r)}=x_{(j+s)}$ pour tout r et s tels que $d \leq r \leq d'$, $e \leq s \leq e'$ et $(r-s)=(k+1)$.

Les points (4) et (5) découlent du fait que le carré est de longueur $2(j-i-k-1)$. En conséquence, toutes ces conditions représentent la forme générale d'un carré de longueur $2(j-i-k-1)$, contenant les caractères $x_{(i+k+1)}$ et x_j. On note qu'au cours de la démonstration du théorème, on se refère souvent à elles.

On distingue trois cas:

cas 1: $d \leq 0$

Montrons que $(x_{(j-k-1)}x_{(j-k)}\ldots x_{(j-1)})(x_j x_{j+1}\ldots x_{(j+k)})$ est un carré de longueur $2(k+1)$.

Soit r un élément de l'ensemble$\{1,2,\ldots,k,(k+1)\}$. Comme $x_{(i+k+1-r)}=x_{(j+k+1-r)}$ et $d \leq 0 \leq k+1-r \leq k+1 \leq d'$, on a d'après condition 5: $x_{(j+k+1-r)}=x_{(i+k+1-r)}= x_{(j+s)}$ avec $(k+1)-r-s=(k+1)$. Ce qui donne $x_{(j+k+1-r)}=x_{(j-r)}$.

cas2: $e' \geq k$

Montrons que $(x_i x_{(i+1)}\ldots x_{(i+k)})(x_{(i+k+1)}x_{(i+k+2)}\ldots x_{(i+2k+1)})$ est un carré de longueur $2(k+1)$.

Soit r un élément de l'ensemble $\{0,1,\ldots,k\}$. Comme $x_{(i+r)}=x_{(j+r)}$ et $e \leq 0 \leq r \leq k \leq e'$, on a d'après condition 5:

$x_{(i+s)}=x_{(j+r)}=x_{(i+r)}$ avec $(s-r)=(k+1)$. Ce qui donne $x_{(i+r)}=x_{(i+k+1+r)}$.

cas3: $d>0$ et $e'<k$

Posons $m=(j-i)-2(k+1)$, et montrons que:

$$(x_{(i+d'-m+1)}x_{(i+d'-m+2)}\ldots x_{(i+d')})(x_{(j+e)}x_{(j+e+1)}\ldots x_{(j+e+m-1)})$$

est un carré de longueur $2m$.

Soit r un élément de l'ensemble $\{1,2,\ldots,m\}$. On peut aisément établir les quatres inégalités suivantes:

A) $e \leq d'+r-m-k-1 \leq e'$
B) $d \leq d'+r-m-k-1 \leq d'$
C) $d \leq d'-m+r \leq d'$
D) $e \leq e+r-1 \leq e'$

·De A) et C) on montre que $x_{(i+d'-m+r)}=x_{(j+d'-m+r-k-1)}$, et de B) et D) on montre que $x_{(i+d'-m+r-k-1)}=x_{(j+e+r-1)}$ (On fait appel aux conditions 1) et5)).

Par ailleurs, les propriétés A) et B) permettent d'observer que

$$1 \leq d'+r-m-k-1 \leq (k-1)$$

et de prouver ainsi que $x_{(i+d'-m+r-k-1)}=x_{(j+d'-m+r-k-1)}$.

En combinant toutes ces égalités, on déduit que $x_{(i+d'-m+r)}=x_{(j+e+r-1)}$.

Au vu de ces trois cas, on peut conclure qu'il existe toujours un carré de longueur strictement inférieure à 2(j-i-k-1) .□

Théorème6:

Il existe un réseau systolique linéaire de taille O(n), capable de reconnaitre en temps linéaire tous les mots sans carrés de longueurs n.

Démonstration:

Soit x le mot à étudier.

Cas1: x est un mot sans carré

Le résultat découle du réseau bidimensionnel présenté dans la troisième section.

Cas2: x contient un carré

Lemme5 nous assure que le carré de longueur minimale de x ne peut pas être dans une situation de chevauchement. En conséquence, il sera détecté par le réseau unidimensionnel qu'on vient de construire.□

5- CONCLUSION

Dans ce chapitre, nous avons présenté des réseaux systoliques permettant de détecter en temps linéaire des carrés dans un mot. Contrairement à la solution proposée par Apostolico et Negro[1], les réseaux présentés ici sont modulaires.

Nous avons commencé par construire un réseau bidimensionnel de $O(n^2)$ cellules capable de détecter tous les carrés dans un mot de longueur n. Ensuite, nous avons établi un lemme sur le chevauchement des carrés (Lemme5); ceci nous a permis de transformer la strucure bidimensionnelle à un réseau unidimensionnel pour la reconnaissance des mots sans carrés.

Il est important de noter que l'amélioration de la surface du réseau bidimensionnel est envisageable. En effet, si l'on remplace compt(i) par un compteur systolique dont on ne fait stocker que la représentation binaire de i, on obtient un réseau systolique de O(nlogn) cellules.

Un algorithme séquentiel de reconnaissance des mots sans carrés qui s'exécute en temps linéaire a été proposé par Crochemore [5]; l'avantage de notre solution est sa rapidité lorsqu'elle est réalisée en circuits VLSI.

6- REFERENCES

1- **A. Apostolico et A. Negro**
Systolic algorithms for string manipulations
IEEE. Trans. comput. c33, 4(1984)

2- **A. Apostolico et F. P. Preparata**
Optimal off-line detection of repetitions in a string
Theor. Comput. Sci. 22(1983)

3- **J. Berstel**
Some recent results on square free words
Lecture notes in computer science, n° 166 (1984)

4- **M. Crochemore**
An optimal algorithm for computing the repetitions in a word
Inf. Proc. Letters, 12 (1984)

5- **M. Crochemore**
Recherche linéaire d'un carré dans un mot
C. R. Acad. Sci. Paris, 296(1983)

6- **M. J. Foster et H. T. Kung**
The design of special purpose chips
IEEE. Computer Magazine, 13 (1980)

7- **H. T. Kung**
Why systolic architectures
IEEE. Computer Magazine, 15 (1982)

8- **L. Melkemi et M. Tchuente**
A one dimensional array for linear time recognition
of square free words
Cognitiva (1985)

CONCLUSION GENERALE

Au vu du travail mené dans l'ensemble des textes rassemblés dans cette thèse, on peut conclure que l'analyse des performances des algorithmes systoliques nécessite l'introduction d'outils de raisonnement, et se base sur la combinaison des trois critères suivants:

-étude du temps de calcul
-étude du nombre d'unités de traitement
-étude du nombre d'accès-mémoires

Au premier chapitre, nous avons étudié la complexité en temps du calcul du produit C=A.B de deux matrices sur la classe des réseaux rectangulaires. Dans le cas où A et B sont des matrices carrées d'ordre n, nous avons établi que le temps minimum nécessaire pour effectuer le calcul est égal à 3n-2.

Comme nous l'avons noté, l'approche combinatoire, qui est à la base d'un tel résultat, peut être adaptée à tout réseau systolique bi-dimensionnel; pour ce faire, nous n'avons qu'à modifier les directions des segments où les variables doivent être affectées.

Nous nous sommes intéressés, après, à l'étude du compromis entre le temps et le nombre d'accès-mémoires pour le calcul du produit matriciel sur la classe des réseaux programmables. Dans le cas du calcul du produit de deux matrices carrées d'ordre n, nous avons présenté des algorithmes qui s'exécutent en temps 3n-2, 2.5n-2 et 2n-2 sur des réseaux de taille nxn et ayant 2n, 3n et 4n accès-mémoires respectivement. Ceci montre que, si on a à minimiser le temps de calcul d'un problème sur une architecture systolique, on doit tenir compte du nombre de cellules communiquant avec le processeur hôte.

Au troisième chapitre, nous avons proposé des réseaux systoliques pour la détection des carrés dans un mot. Dans un premier temps, nous avons présenté un réseau systolique bi-dimensionnel permettant de détecter en temps linéaire tous les carrés dans une chaîne de caractères. Ensuite, nous avons établi un lemme sur le chevauchement des carrés dans un mot (Lemme5); ceci nous a permis de construire un réseau uni-dimensionnel capable de reconnaître les mots sans carrés. Le travail effectué dans ce chapitre mène à remarquer la nécessité d'introduire dans le contexte des réseaux systoliques, les outils usuels de raisonnement destinés à optimiser le coût des algorithmes séquentiels [18], [19].

D'une manière générale, les réseaux systoliques sont des processeurs intégrés spécialisés, particulièrement adaptés à la résolution des problèmes où le volume de calculs est beaucoup plus important que le nombre des variables manipulées. Ils sont destinés à capturer les concepts du parallélisme dans un formalisme mathématique unifié. Les différents thèmes qu'ils ont fait naître, et que nous avons cités en introduction, montrent bien que le modèle systolique est un sujet de recherche très fécond... dont nous n'avons abordé que quelques aspects.

REFERENCES

[1] **H. M. Ahmed, J. M. Delosme et M. Morf**
Highly concurrent computing structures for matrix arithmetic and signal processing
IEEE, Comput. magazine 15, 1(1982)

[2] **A. Apostolico et A. Negro**
Systolic algorithms for string manipulations
IEEE, Trans. Comput. 4(1984)

[3] **A. Apostolico et F. P. Preparata**
Optimal off-line detection of repetitions in a string
Theor. Comput. Sci. 22(1983)

[4] **A. J. Atrubin**
An iterative one-dimensional real-time multiplier
IEEE, EC 1965

[5] **J. Berstel**
Some recent results on square free words
Lecture notes in computer science, n°166(1984)

[6] **A. Bojanczyk, R. P. Brent et H. T. Kung**
Numerically stable solution of dense systems of linear equations using mesh-connected prcessors
T.R. CMU 1981

[7] **S. N. Cole**
Real-time computation by n-dimensional iterative arrays of finite-state machines
IEEE. Trans. Comput. c18, 4(1969)

[8] **M. Crochemore**
An optimal algorithm for computing the repetitions in a word
Inf. Proc. Letters, 12(1984)

[9] **M. Crochemore**
Recherche linéaire d'un carré dans un mot
C. R. Acad. Sci. Paris, 296(1983)

[10] **A. L. Fisher, H. T. Kung, L. M. Monier, H. Walker et Y. Dohi**
Design of the PSC: a programmable systolic chip
Proc. Third Caltech. Conf. on VLSI, computer science press Inc. 1983

[11] **M. J. Flynn**
Very high-speed computing systems
Proc. IEEE, 54(1966)

[12] **M. J. Foster et H. T. Kung**
The design of special purpose VLSI chip
IEEE Comput. Magazine, 13(1980)

[13] **W. M. Gentelman et H. T. Kung**
Matrix triangularisation by systolic arrays
Proc. SPIE 298, Real-time signal processing, 4(1981)

[14] **L. J. Guibas, H. T. Kung et C. D. Thompson**
Direct VLSI implementations of combinatorial algorithms
Proc. Caltech, conf. on VLSI, (1979)

[15] **D. E. Heller et I. C. F. Ipsen**
Systolic networks for orthogonal equivalence transformations and their applications
Proc. Conf. Advanced research in VLSI MIT. 1982

[16] **F. Hennie**
Iterative arrays of logical circuits
New York: Wiley, 1961

[17] **E. Kanota**
Cellular algorithms for binary matrix multiplication
Cellular processors and cellular algorithms: Vollmar et al eds
Braunschweig, 1981

[18] **D. E. Knuth**
The art of computer programming
Reading, MA: Addison-Wesly, 1973

[19] **D. E. Knuth, J. H. Morris et V. R Pratt**
Fast pattern matching in strings
SIAM J. Computing, vol.c-28 N°6 (1979)

[20] **A. V. Kulkarni et D. W. L. Yen**
Systolic processing and an implementation for signal and image processing
IEEE. Trans. Comput. 10(1982)

[21] **H. T. Kung**
Why systolic architectures?
IEEE. Comput. Magazine, 1(1982)

[22] **H. T. Kung et M. S. Lam**
Fault tolerant and two-level pipelining VLSI systolic arrays
T. R. CMU. 1983

[23] **H. T. Kung et P. L. Lehman**
Systolic (VLSI) arrays for relational data base operations
Proc. ACM/SIGMOD Int. Conf. Management of data (1980)

[24] **H. T. Kung et C. E. Leiserson**
Systolic arrays (for VLSI)
Proc. Symp. on Spase matrices computations
I. S. Duff et G. W. Stewart eds: Knoxville, Tenn. 1978

[25] **H. T. Kung et W. L. Lin**
An algebra for VLSI algorithmic design
T. R. CMU. 1983

[26] **M. Lam et J. Mostow**
A transformational model of VLSI systolic arrays
IEEE Trans. Comput. 2(1985)

[27] **F. T. Leighton et C. E. Leiserson**
Wafer-scale integration for systolic arrays
Proc. 23rd Ann. symp. on foundations of computer science IEEE, 1982

[28] **C. E. Leiserson et J. B. Saxe**
Optimising synchronous systems
Proc. 22th annual symp. foundations, computer science IEEE, 1981

[29] **G. J. Li et B. W. Wah**
The design of optimal systolic algorithms
IEEE. Trans. Comput. 1984

[30] **C. A. Mead et L. A. Conway**
Introduction to VLSI systems
Addison Wesly, 1980

[31] **R. Melhem et W. C. Rheinboldt**
A mathematical model for the verification of systolic networks
SIAM. Journal. Comput. 13(1985)

[32] **L. Melkemi et M. Tchuente**
Systolic arrays for connectivity and triangularisation problems
Proc. Conf. Dynamic Behaviour Automata: J. Demongeot et al eds
Academic press, 1983

[33] **L. Melkemi et M. Tchuente**
Complexity of matrix product on a class of orthogonally systolic arrays
soumis à IEEE. Transactions on computers

[34] **L. Melkemi et M. Tchuente**
Programmation du produit matriciel sur un réseau systolique rectangulaire
TSI. 4(1985)

[35] **L. Melkemi et M. Tchuente**
I/O-Time tradeoffs for matrix product on programmable arrays
Actes du colloque COMPAR85, Allemagne 1985

[36] **L. Melkemi et M. Tchuente**
A one dimensional array for linear time recognition of square free words
Actes du colloque Cognitiva. Paris, 1985

[37] **L. Melkemi et M. Tchuente**
Systolic algorithms for rectangular matrix product
"a complexity result"
Le Calcul Demain: P. Chenin et al eds
Masson et John Wiley, 1985

[38] **W. L. Miranker et A. Winkler**
Space-time representations of systolic computational structure
Research report IBM, New York: 1982

[39] **D. I. Moldovan**
On the analysis and synthesis of VLSI algorithms
IEEE. Trans. Comput. c31, 11(1982)

[40] **F. P. Preparata et J. Vuillemin**
Area-time optimal VLSI networks for mutiplying matrices
Inf. Proc. Letters, 1980

[41] **P. Quinton**
The systematic design of systolic algorithms
T. R. INRIA. 1983

[42] **Y. Robert**
Algorithmique parallele
Thèse. Grenoble, 1986

[43] **Y. Robert et M. Tchuente**
Une approche Divide-and-conquer for systolic algorithms
R.R. IMAG. 1983

[44] **Y. Robert et M. Tchuente**
Réseaux systoliques pour des problèmes de mots
RAIRO. Informatique théorique, 2(1985)

[45] **C. Savage**
A systolic data structure chip for connectivity problems
VLSI systems and computations: H.T.Kung et al eds
Computer science press, Rockville Md, 1981

[46] **P. Varman et D. Fussel**
Fault-tolerant wafer-scale architectures for VLSI
Proc. 9th ann. symp. on computer architecture, 1982

[47] **U. Weiser et A. Davis**
A mathematical representation for VLSI arrays
Report UUCS-8-111, University of Utah, 1980

AUTORISATION DE SOUTENANCE

DOCTORAT 3ème CYCLE, ~~DOCTORAT-INGENIEUR~~, ~~DOCTORAT USMG~~

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

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

Vu les rapports de M. ....M.a.u.r.i.ce.....T.CHUENTE.....

M. .....

M. .....MELKEMI.....Lamine..... est autorisé

à présenter une thèse en vue de l'obtention du ....Doctorat..de..troisième....

.cycle....en...Mathématiques...Appliquées.....

Grenoble, le 21 MARS 1986

Le Président de l'Université Scientifique

et Médicale

M. TANCHE