Produit matriciel sur une famille de réseaux systoliques linéaires et unidirectionnels

Clémentin TAYOU DJAMEGNI

Faculté des Sciences, Département d'Informatique
BP. 812 Yaoundé Cameroun

Gabriel BOMBAMBO BOSEKO et Emmanuel KAYEMBE ILUNGA

Université de Kinshasa
Faculté des Sciences, Département de Mathématiques
BP. 110 kinshasa 11 Zaïre

Mots-clés: produit matriciel, réseaux systoliques, pipeline, accumulation, algorithmes optimaux, réseaux linéaires unidirectionnels, algorithmes parallèles, complexité en surface, complexité en temps, canal, registre, fonction de temps.

Résumé: Nous étudions le calcul du produit C=AB de deux matrices carrées A et B d'ordre n, sur une famille de réseaux systoliques, linéaires et unidirectionnels, dont chaque cellule effectue au plus une accumulation (une multiplication suivie d'une addition) entre deux tops.

Abstract: We study the computation of the product C=AB of two matrices A , B of order n on a family of unidirectional linear systolic arrays, where each cell performs at most one accumulation per step. We are interested in algorithms where the entries of a given matrix are read only once and flow through the array for the pipeline computation of C. We have a constant number of channels, a for matrix A, b for matrix B, and c for matrix C. On each channel, there is a constant number of registers per cell, X on each channel of A, Y on each channel of B, and Z on each channel of C. Such architecture is denoted XYZ_abc. Hereafter we assume that X>Y>Z, and we show that, the lower bounds for the number of cells and the execution time are, I/(X-Z) and XI/(X-Z) respectively, where I=max(n^2(X-Y)/(X-Z)a+n^2/b-1, n^2/a+n^2/c) . We exhibit on X21_1bb architectures new algorithms which require n^2(1/(X-1)+1/(X-2)b)+O(n) cells and Xn^2(1/(X-1)+1/(X-2)b)+O(n) steps. Then, we show how one can acheive optimal computation of C on X21_1min(X-1, b+1)b architectures. These results improve the best previously known bounds. We end the paper by showing that one can reduce futur studies on the architectures, X21_1bc where b≤c and XY1_2bc where c≥3.

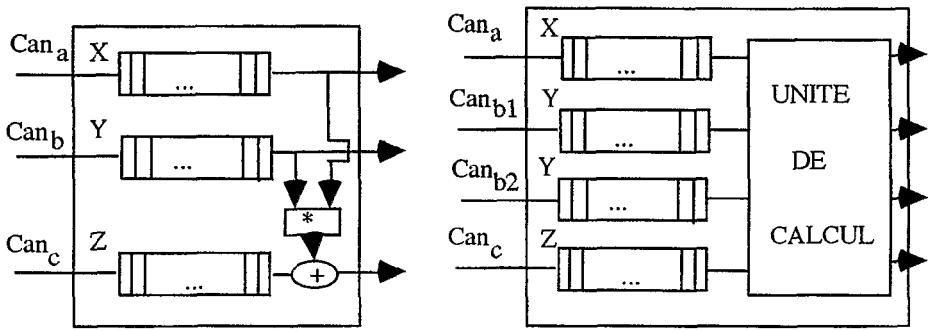
1. Introduction

Les systèmes systoliques ont été largement utilisés comme modèle de calcul parallèle. Ces systèmes sont constitués d'un grand nombre de processeurs identiques et élémentaires connectés localement, et de façon régulière. Chaque processeur reçoit les données de ses voisins, effectue des calculs et leur envoie les résultats. Seuls quelques processeurs particuliers situés aux extremités du réseau communiquent avec le monde extérieur. Les processeurs opèrent en parallèle et sont synchronisés par une horloge globale.

Le plus simple des modèles systoliques est le réseau linéaire. Dans un tel système, les processeurs sont interconnectés de façon linéaire. Malgré leur simplicité, les réseaux systoliques linéaires ont été utilisés pour la résolution des problèmes très variés [TCH 91], notamment le produit de convolution, les systèmes linéaires triangulaires, le produit matrice-vecteur et le produit matriciel. L'intérêt de tels réseaux réside dans leur régularité et leur nombre constant d'entrées/sorties qui limite les coûts de communication avec la machine hôte, car seules les cellules extrêmes communiquent avec l'hôte. D'autre part, ces réseaux sont adaptés à la tolérance aux fautes de conception [HUA 84, JOU 85, KUN 84, NAY 90].

Nous étudions le produit de deux matrices carrées A et B d'ordre n sur réseaux systoliques linéaires et unidirectionnels. La difficulté du problème provient du fait que les méthodologies de conception systématique d'algorithmes systoliques développées notamment par Moldovan [MOL 82], Quinton [QUI 84], Miranker et Winkler [MIR 84], Clauss, Mongenet et Perrin [CLA 92], Shako et Tchuenté [SAK 89] ne permettent pas d'obtenir des algorithmes systoliques sur des réseaux linéaires. En effet, les techniques d'allocation des tâches du graphe des dépendances aux processeurs proposées par les auteurs donnent des réseaux de dimmension k - 1 pour un graphe des dépendances de dimension k qui n'est linéaire que si k = 2, alors que le graphe des dépendances du produit matriciel est de dimension 3.

Dans ce papier, nous nous intéressons à la classe d'algorithmes où les entrées d'une même matrice sont lues une seule fois et se déplacent en continu dans le réseau à la même vitesse pour le calcul en pipeline du produit C=AB. Nous supposons que la cellule de base de l'architecture cible éffectue au plus une accumulation (addition suivie d'une multiplication) entre deux tops. Nous disposons d'un nombre constant de canaux, a canaux empruntés par les coefficients de A, b canaux empruntés par les coefficients de B, et c canaux empruntés par les coefficients de C. Sur chaque canal, chaque cellule possède un nombre constant de registres, X pour les canaux de A, Y pour les canaux de B, et Z sur les canaux de C. Une telle architecture est dite de type XYZ_abc. La figure 1 illustre la cellule de base d'une telle architecture.



(a) a = 1, b = 1, c = 1 (b) a = 1, b = 2, c = 1

Figure 1 Cellules de base des architectures XYZ_111 et XYZ_121

Nous admettons l'envoi de signaux de contrôle dans le réseau pour inhiber les accumulations invalides. Dans toute la suite, nous supposons que les inégalités, X>Y>Z, sont satisfaites.

Dans [RAM 84, RAM 85] Ramakrishan et Varman proposent des algorithmes s'exécutant sur réseaux bidirectionnels et partiellement pipelinés. Les réseaux bidirectionnels sont plus rapides que les réseaux unidirectionnels, mais ils s'adaptent mal aux fautes de conception.

Lee et Kedem [LEE 88] proposent une méthode permettant la synthèse automatique des réseaux linéaires et modulaires à partir des nids de boucle. Les complexités en temps et en surface d'un algorithme de calcul du produit C=AB obtenu par cette méthode pour l'architecture XY1_111, sont bornées inférieurement [TAN 94] respectivement par, min{1+(n-1)/(X-1)+(n-1)/(X-Y), 1+(n-1)/(Y-1)+(n-1)/(X-1)}n et min{1+(n-1)/(X-Y)+(n-1)/(X-Y), 1+(n-1)/(Y-1)+(n-1)/(X-1)}Xn. Toutefois, il n'est pas démontré que ces bornes sont atteintes asymptotiquement. De plus cette méthode nécessite un temps de prétraitement relativement long, car elle suppose la résolution d'un problème linéaire en nombres entiers dont le nombre de contraintes dépend de n.

Dans [KUM 88, KUM 89], Kumar et Tsai proposent une méthode de synthèse des algorithmes 1D à partir des algorithmes 2D. Les algorithmes de produit matriciel obtenus par

cette méthode nécessitent deux canaux pour B, $3n^2/(X-3) + O(n)$ processeurs et un temps de calcul de l'ordre de $2Xn^2/(X-3) + O(n)$.

Le meilleur algorithme connu à ce jour pour le calcul du produit C = AB sur une architecture X21_1bb sans signaux de contrôle a été proposé par Benaini et Tchuenté [BEN 88, BEN 89a]. Les performances atteintes par cet algorithme sont en $n^3/(bX^2) + O(n^2)$ pour la surface du réseau et en $n^3/(bX) + O(n^2)$ pour le temps de calcul. Dans [BEN 88, BEN 89b] les mêmes auteurs proposent une formulation géométrique pour le calcul du produit C=AB sur réseaux modulaires et linéaires. Cette approche est à la base de la méthode du parallélogramme.

Une extension de la méthode du parallélogramme est proposée dans [TAN 91, TAN 94] par Tangy Risset, et appelée "méthode des pavés avec la strategie en diagonales". Le nombre de cellules S et le temps de latence T d'un réseau X21_111 synthétisé par cette méthode verifient, $(1/(X-1)+1/(X-2))n^2 \leq S \leq (1/(X-1)+1/(X-2)+1/(X-1)^2)n^2$ et $(1/(X-1)+1/(X-2))Xn^2 \leq T \leq (1/(X-1)+1/(X-2)+1/(X-1)^2)Xn^2$. De plus, ces performances sont les meilleures connues à ce jour.

Nous étudions d'abord la complexité du problème. Nous montrons que les complexités en surface et en temps offertes par l'architecture XYZ_abc pour le produit matriciel sont bornées inférieurement respectivement par $I/(X-Z)$ et $XI/(X-Z)$, où $I = \max(n^2(X-Y)/(X-Z)a+n^2/b-1, n^2/a+n^2/c)$. Nous proposons sur l'architecture X21_1bb, un algorithme nécessitant $n^2(1/(X-1)+1/(X-2)b) + O(n)$ processeurs, et un temps de calcul de l'ordre de $Xn^2(1/(X-1)+1/(X-2)b) + O(n)$, ce qui n'est pas loin des bornes inférieures. Nous exhibons ensuite sur l'architecture X21_1min(X-1, b+1)b un algorithme dont les complexités en surface et en temps sont respectivement, $n^2(1/(X-1)+1/(X-1)b) + O(n)$ et $Xn^2(1/(X-1)+1/(X-1)b) + O(n)$, ce qui est optimal. Sur la base des résultats obtenus, nous montrons qu'on peut restreindre les études ulterieures de calcul du produit matriciel aux architectures X21_1bc où $b \leq c$ et XY1_2bc où $c \geq 3$.

2. Etude de la complexité du produit matriciel

Nous étudions dans cette partie la complexité du produit matriciel sur les architectures XYZ_abc. Afin d'accelerer la sortie des résultats, on peut supposer que $Z<\min(X,Y)$. Il est facile de voir que si X=Y, les coefficients de A et B ne se rencontreront jamais. Nous supposons donc que $X \neq Y$. Les cas $X>Y$ et $Y<X$ correspondent à des architectures symétriques. On peut alors supposer que $X>Y>Z$..

Théorème 1 Soient T et S les complexités respectives en temps et en surface d'un algorithme quelconque sur l'architecture XYZ_abc. Les inégalités suivantes sont satisfaites:

$$S \geq \frac{I}{(X-Z)} \quad \text{et} \quad T \geq \frac{XI}{(X-Z)} \quad \text{où} \quad I = \max\left(\frac{(X-Y)|A|}{(X-Z)a} + \frac{|B|}{b} - 1, \frac{|A|}{a} + \frac{|C|}{c}\right)$$

$|A|$ désigne le nombre de coefficients de la matrice A.

Preuve

Soit à effectuer le produit C = AB, où A et B sont des matrices carrées d'ordre n. Soient T et S les complexités respectives en surface et en temps d'un algorithme quelconque, ALG, sur l'architecture XYZ_abc. D'après le théorème de coupe [LEI 88], on peut se ramener à l'architecture pseudosystolique, X'Y'Z'_abc, où Z'= 0, sans modifier le nombre de processeurs. Soient donc T' et S' les complexités en temps et en surface de ALG sur X'Y'Z'_abc. Les égalités suivantes sont satisfaites :

$$X' = X - Z, \quad (2.1)$$
$$Y' = Y - Z, \quad (2.2)$$
$$Z' = 0, \quad (2.3)$$
$$S' = S, \quad (2.4)$$

$$T = T' + S'Z . \tag{2.5}$$

Plaçons nous dans le réseau X'Y'Z'_abc et posons, $A*_k = \{a_{1k}, a_{2k}, ..., a_{nk}\}$. b_{kj} doit rencontrer tous les coefficients de $A*_k$ pour le calcul des coefficients de $C*_j$. Etant donné X' > Y', tous les coefficients de $A*_k$ doivent entrer dans le réseau avant b_{kj} pour pouvoir le rencontrer. De même tous les coefficients de $B*_j$ doivent entrer dans le réseau avant c_{ij} pour pouvoir participer aux accumulations correspondantes au calcul de c_{ij}.

En résumé, c_{ij} entre dans le réseau après $b_{1j}, b_{2j}, ..., b_{nj}$, et b_{kj} après $A*_k$.
c_{ij} entre dans le réseau après $A*_1, A*_2, ..., A*_n$.
c_{ij} entre dans le réseau après A.

Puisque nous disposons de a canaux pour les coefficients de A et de c canaux pour les coefficients de C, il vient que:

$$T' \geq \frac{|A|}{a} + \frac{|C|}{c} \tag{2.6}$$

Nous allons maintenant borner inférieurement le temps nécessaire à l'entrée des coefficients de A et de B dans le réseau.

Soit bi1j1 le premier coefficient de B qui entre dans le réseau. Soient ta1 et ta2 (resp. tb1 tb2), les dates respectives de début et de fin d'entrée des coefficients de A (resp. B) dans le réseau. Puisque la premiere donnée à entrer dans le réseau est un coefficient de A, on peut supposer que ta1=1.

Puisque les accumulations commencent après l'entrée de tous les coefficients de A dans le réseau, un calcul simple montre que:

$$tb1 \geq \frac{X' - Y'}{X'} ta2 \tag{2.7}$$

Si l'inégalité, (2.7), n'est pas verifiée, bi1j1 se trouvera devant tous les coefficients de A à l'instant ta2, et ne pourra participer à aucune accumulation.

Puisqu'il y a a canaux pour A et b canaux pour B, on a :

$$ta2 \geq \frac{|A|}{a} \quad \text{et} \quad tb2 - tb1 + 1 \geq \frac{|B|}{b} \tag{2.8}$$

(2.7) et (2.8) entraînent que:

$$T' \geq \frac{(X - Y)|A|}{(X - Z)a} + \frac{|B|}{b} - 1 \tag{2.9}$$

Posons:

$$I = \max\left(\frac{(X - Y)|A|}{(X - Z)a} + \frac{|B|}{b} - 1, \frac{|A|}{a} + \frac{|C|}{c}\right) \tag{2.10}$$

On déduit de (2.6), (2.9) et (2.10) que:

$$T' \geq I \tag{2.11}$$

Soit a_{i1k1} le premier coefficient de A à entrer dans le réseau. Il est facile de voir que b_{k1j} effectue sa dernière accumulation avec a_{i1k1}. C_{i1j} est alors le dernier coefficient de la colonne j de C calculé par l'algorithme ALG. Il vient que le dernier coefficient de C calculé par l'algorithme effectue une accumulation avec a_{i1k1}. d'où:

$$S' \geq \frac{T'}{X'} \tag{2.12}$$

On déduit de (2.4), (2.11) et (2.12) que:

$$S \geq \frac{I}{(X - Z)} \tag{2.13}$$

(2.4), (2.5) et (2.11) entraînent que:

$$T - SZ \geq I \tag{2.14}$$

De (2.13) et (2.14), on a:

$$T \geq \frac{ZI}{(X - Z)} + I$$

Ceci entraîne que:

$$T \geq \frac{XI}{(X - Z)} \quad \square$$

3. Produit matriciel sur les architectures X21_1$\mu\mu$

Nous allons présenter de manière analytique une famille d'algorithmes baptisé ALG1 pour le produit de deux matrices sur les architectures X21_1$\mu\mu$. Nous admettons que les canaux de B(resp. C) sont numerotés de 0 à μ-1.

Posons,

$x = X-1$,
$n_1 = (x-1)x\mu q+1$,
$n_2 = x^2q$,
$T_0 = n_1n+n+1$,

où q est le plus petit entier naturel satisfaisant, $x^2q(x-1)\mu \geq n$.

Les dates d'entrée des coefficients dans le réseau sont définies comme suit :

$T_a[i,k] = xn_1k + (x-1)i$
$T_b[k,j] = n_1k + (x-1)n_2j + (x-1)T_0$
$T_c[i,j] = xT_0 - i + xn_2j$

Nous devons specifier le numéro de canal emprunté par un coefficient de B ou de C. b_{kj} emprunte le canal de numéro p si les inégalités, $p(x-1)n_2 < k \leq (p+1)(x-1)n_2$, sont satisfaites. De même, c_{ij} emprunte le canal de numéro p si les inégalités, $p(x-1)n_2 < i \leq (p+1)(x-1)n_2$, sont satisfaites. Puisque l'inégalité, $n \leq \mu(x-1)n_2$ est satisfaite, l'algorithme nécessite, μ canaux pour les coefficients de B et μ canaux pour les coefficients de C.

Nous allons établir quatre lemmes qui prouvent la validité de l'algorithme.

Lemme 3.1 Deux coefficients d'une même matrice qui empruntent le même canal, entrent dans le réseau à des instants distincts.

Preuve

(a) Supposons que $T_a[i,k] = T_a[i',k']$.

Ceci entraîne que $xn_1(k-k') = -(x-1)(i-i')$. (3.1)

Ceci entraîne que $(x-1)(i-i') \equiv 0 \bmod (xn_1)$.

Par ailleurs, pgcd(x-1, xn_1)=1, puisque pgcd(x-1,x) = pgcd(x-1,n_1)=1.

D'où, i - i' $\equiv$ 0 mod (xn_1).

Puisque $|i - i'| < n \leq xn_1$, il vient que i - i' = 0. (3.2)

(3.1) et (3.2) entraînent que i = i' et k = k'.

(b) Supposons que $T_b[k,j] = T_b[k',j']$ et que b_{kj} et $b_{k'j'}$ empruntent le même canal.

Ceci entraîne que, $n_1(k-k') = -(x-1)n_2(j-j')$ et $|k - k'| < n_2(x-1)$. (3.3)

Ceci entraîne que $n_1(k-k') \equiv 0 \bmod ((x-1)n_2)$. (3.4)

Par ailleurs, on a $n_1 - (x-1)\mu xq = 1$. D'où,

pgcd(n_1, x) =1, et
pgcd(n_1,(x-1)xq) = 1.

Il vient que pgcd(n_1,(x-1)n_2) = 1. (3.5)

(3.4) et (3.5) entraînent que k-k' $\equiv$ 0 mod ((x-1)n_2). (3.6)

(3.3) et (3.6) entraînent que k-k' = 0.

Il vient que j = j'.

(c) Supposons que $T_c[i,j] = T_c[i',j']$ et que c_{ij} et $c_{i'j'}$ empruntent le même canal.

Ceci entraîne que -(i-i') = -xn_2(j-j') et $|i - i'| < n_2(x-1)$. (3.7)

Ceci entraîne que $n_2(x-1) > xn_2|j - j'|$.

Ceci entraîne que j = j'.

Il vient que i = i' □

Lemme 3.2 Les coefficients a_{ik}, b_{kj}, et c_{ij} se rencontrent dans la cellule de numéro s(i,k,j) = $T_0 - n_1k - i + n_2j$.

Preuve

Un calcul simple montre que

$$T_a[i, k] + Xs(i,k,j) = T_b[k, j] + 2s(i, k, j) = T_c[i, j] + s(i, k, j)$$ □

Lemme 3.3 Chaque cellule effectue au plus une accumulation à chaque top.

Preuve

Supposons que c_{ij} et $c_{ij'}$ rencontrent simultanément un coefficient a_{ik} de A dans le réseau. Ceci entraîne que c_{ij} et $c_{ij'}$ entrent dans le réseau à la même date. Comme ils empruntent le même canal, le lemme 3.1 entraîne que j = j'. □

Nous allons maintenant montrer comment inhiber les accumulations invalides à l'aide des signaux de contrôle. Le lemme 3.2 montre qu'un coefficient b_{kj} de B participe aux accumulations successivement dans les cellules voisines, s(n, k, j), s(n - 1, k, j), ..., s(1, k, j),

dans cet ordre. On peut alors imaginer un mécanisme permettant de détecter le début et la fin d'une série d'accumulations successives. Le lemme suivant répond à cette question.

Lemme 3.4 Si b_{kj} rencontre $a_{nk'}$ et $c_{nj'}$, alors $k = k'$ et $j = j'$. De même, si b_{kj} rencontre $a_{1k'}$ et $c_{1j'}$, alors $k = k'$ et $j = j'$

Preuve

Supposons que b_{kj} rencontre $a_{nk'}$ et $c_{nj'}$. D'après le lemme 3.2, $a_{nk'}$, $b_{k'j'}$ et $c_{nj'}$ se rencontrent dans la cellule s(n, k', j'). Puisque $a_{nk'}$ et $c_{nj'}$ ne se déplacent pas à la même vitesse, ils se rencontrent uniquement dans la cellule s(n, k', j'). D'où $k = k'$ et $j = j'$. La démonstration de la deuxième implication se fait de façon analogue. □

Ce lemme montre que b_{kj} commence à participer aux accumulations quand il rencontre deux coefficients de la ligne n et cesse de participer quand il rencontre deux coefficients de la première ligne. On peut alors affecter une couleur aux coefficients de la dernière ligne de A et de C, et une autre couleur aux coefficients de la première ligne de A et de C, pour permettre a chaque coefficient de B de détecter les instants où il commence et termine une serie d'accumulations successives.

Nous déterminons à présent les performances de l'algorithme. Désignons respectivement par S_1 et T_1 le nombre de processeurs et le temps exigés par l'algorithme.

La valeur maximale de s(i, k, j) est atteinte pour i = 1, k = 1 et j = n. La surface du réseau est alors,

$$S_1 = s(1, 1, n) = \frac{n^2}{X - 1} + \frac{n^2}{(X - 2)\mu} + O(n$$

Le premier coefficient à entrer dans le réseau est a_{11} et le dernier est c_{1n}. Le temps d'exécution est alors:

$$T_1 = T_c[1, n] + S_1 - T_a[1, 1] = \frac{Xn^2}{X - 1} + \frac{Xn^2}{(X - 2)\mu} + O(n)$$

Ces performances améléorent les meilleures bornes précédemment connues [TAN 91, TAN 94]. De plus, elles ne sont pas loin des bornes inferieures.

4. Calcul optimal du produit C=AB sur les architectures X21_1min(X-1, μ+1)μ

Nous montrons dans cette partie comment calculer de façon optimal le produit C=AB, sur les architectures X21_1min(X-1, μ+1)μ. La famille d'algorithmes proposée dans cette partie est baptisée ALG2.

Posons,

$x = X-1$,
$n_1 = (x-1)\mu q+1$,
$n_2 = (x-1)q$,
$T_0 = n_1 n+n+1$,

où q est le plus petit entier naturel satisfaisant, $xq(x-1)\mu \geq n$.

Les dates d'entrée des coefficients dans le réseau sont définies comme suit :

$T_a[i,k] = xn_1 k + (x-1)i$
$T_b[k,j] = n_1 k + (x-1)n_2 j + (x-1)T_0$

$$T_c[i,j] = xT_0 - i + xn_2 j$$

Si $\min(X-1, \mu+1) = \mu+1$, le coefficient b_{kj} de B, emprunte le canal de numéro p si les inégalités, $p(x-1)n_2 < k \leq (p+1)(x-1)n_2$, sont satisfaites. Si $\min(x, \mu+1) = x$, b_{kj} emprunte le canal de numero p si les inégalités, $pn_1 < j \leq (p+1)n_1$ sont satisfaites. Le coefficient c_{ij} de C, emprunte le canal de numéro p si les inégalités, $pxn_2 < i \leq (p+1)xn_2$, sont satisfaites.

Nous allons établir trois lemmes qui prouvent la validité de l'algorithme.

Lemme 4.1 Deux coefficients d'une même matrice qui empruntent un même canal, entrent dans le réseau à des instants distincts.

Preuve

(a) Supposons que $T_a[i,k] = T_a[i',k']$.

Ceci entraîne que $(x-1)(i-i') \equiv 0 \bmod (xn_1)$.

Puisque $\text{pgcd}(x-1, xn_1)=1$, il vient que $i - i' \equiv 0 \bmod (xn_1)$.

Puisque $|i - i'| < n \leq xn_1$, il vient que $i = i'$, et $k = k'$.

(b) Supposons que $T_b[k,j] = T_b[k',j']$ et que b_{kj} et $b_{k'j'}$ empruntent le même canal.

Ceci entraîne que

$$n_1(k-k') = -(x-1)n_2(j-j') \text{ et } (|k - k'| < n_2(x-1) \text{ ou } |j - j'| < n_1) \qquad (4.1)$$

Puisque, $\text{pgcd}(n_1,(x-1)n_2) = 1$, il vient que

$$k-k' \equiv 0 \bmod (x-1)n_2) \text{ et } j-j' \equiv 0 \bmod (n_1) \qquad (4.2)$$

(4.1) et (4.2) entraînent que $k = k'$ et $j = j'$.

(c) Supposons que $T_c[i,j] = T_c[i',j']$ et que c_{ij} et $c_{i'j'}$ empruntent le même canal.

Ceci entraîne que $-(i-i') = -xn_2(j-j')$ et $|i - i'| < xn_2$.

Ceci entraîne que $j = j'$ et $i = i'$. □

Lemme 4.2 Les coefficients a_{ik}, b_{kj}, et c_{ij} se rencontrent dans la cellule de numéro $s(i,k,j) = T_0 - n_1 k - i + n_2 j$.

Preuve

Un calcul simple montre que:

$$T_a[i, k] + Xs(i,k,j) = T_b[k, j] + 2s(i, k, j) = T_c[i, j] + s(i, k, j).$$ □

Lemme 4.3 Chaque cellule éffectue au plus une accumulation à chaque top.

Preuve

La preuve est analogue à celle du lemme 3.3. □

L'inhibation des accumulations invalides se fait comme dans la partie 3.

Nous déterminons à présent les performances de l'algorithme. Désignons respectivement par S_2 et T_2 le nombre de processeurs et le temps de calcul exigés par ALG2.

$$S_2 = s(1, 1, n) = \frac{n^2}{X - 1} + \frac{n^2}{(X - 1)\mu} + O(n)$$

$$T_2 = T_c[1, n] + S_2 - T_a[1, 1] = \frac{Xn^2}{X - 1} + \frac{Xn^2}{(X - 1)\mu} + O(n)$$

Ces performances, et le théorème 1 montrent que ALG2 est optimal.

5. Comparaison des architectures XYZ_abc

Nous montrons dans cette partie, que l'on peut réduire l'etude du calcul du produit C=AB à une sous-classe particulière des architectures XYZ_abc. L'approche que nous adoptons consiste à comparer ces architectures, suivant quatre critères, le nombre de registres par cellule, le nombre de canaux d'entrées/sorties, et les performances en temps et en surface qu'elles offrent.

Définition 1 Soient deux architectures XYZ_abc et X'Y'Z'_a'b'c'. XYZ_abc est plus faible que X'Y'Z'_a'b'c' (XYZ_abc ≤ X'Y'Z'_a'b'c'), si et seulement si les conditions suivantes sont satisfaites :

(i)- $aX+bY+cZ \le a'X'+b'Y'+c'Z'$,

(ii)- $a+b+c \le a'+b'+c'$,

(iii)- Pour tout algorithme ALG' de calcul du produit C=AB sur X'Y'Z'_a'b'c', il existe un algorithme ALG de calcul du C=AB sur XYZ_abc, tel que ses complexités en surface et en temps de soient inférieures ou égales aux complexités respectives en surface et en temps de ALG'.

Par exemple, les performances de ALG1 et le théorème 1 prouvent que, $(X+1)21\_111 \le X21\_121$. Le lemme suivant se démontre facilement sur la base du théorème 1 et les performances de ALG1. Il montre que l'on peut restreindre l'étude du calcul du produit matriciel aux architectures, X21_1bc où b≤c et XYZ_2bc où c≥3.

Lemme 5.1 (i)- Si c≤a, alors $(aX-a+2)21\_111 \le XYZ\_abc$.

(ii)- Si c>a≥3, alors $(aX - a + 2)21\_1\lceil c / a\rceil\lceil c / a\rceil \le XYZ\_abc$.

(iii)-Si b>c alors $(X+2b-2c)21\_1cc \le X21\_1bc$

6. Conclusion

Dans ce papier, nous avons étudié le calcul du produit matriciel sur les architectures XYZ_abc. Nous avons donné des bornes inférieures pour les performances en surface et en temps offertes par ces architectures pour le calcul du produit matriciel. Ensuite, nous avons proposé des algorithmes simples, dont les complexités en temps et en surface améléorent les meilleures bornes précédemment connues. Sur la base du résultat de complexité obtenu et des performances des algorithmes proposés, nous avons montré que l'on peut restreindre l'étude du calcul du produit matriciel aux architectures, X21_1bc où b≤c et XYZ_2bc où c≥3.

Remerciements: Ce travail a été réalisé dans le cadre du projet Calcul Parallèle soutenu par l'Agence Aire Développement (Paris, France), et a bénéficié de l'appui du Microprocessors and Informatics Programme de l'Université des Nations Unies (Tokyo, Japon). Nous remerçions le Pr. Maurice TCHUENTE de nous avoir posé ce problème ainsi que pour ses conseils et suggestions judicieux.

Bibliographie

[BEN 88] A. Benaini, "Conception et validation des algorithmes systoliques", Phd Thesis, Institut National Polytechnique de Grenoble, France.
[BEN 89a] A. Benaini et M. Tchuenté , "Produit matriciel Sur un reseau lineaire", Revista de Matemmaticas Aplicadas, pp. 23-43.
[BEN 89b] A. Benaini, M. Tchuenté, "Matrix product on modular linear systolic arrays", In Parallel and Distributed Algorithms, M. Cosnard et al. eds., North Holland, pp. 79-88.
[CLA 92] Clauss, Ph., Mongenet, C., Perrin, G. R., "Synthesis of size-optimal toroîdal arrays for the algebraic path problem", Algorithms and Parallel VLSI architectures II. P. Quinton et al. eds., pp. 199-204, Elsevier Science puplishers.
[HUA 84] Huang, K.H., Abraham, "Algorithm Based Fault-Tolerance for Matrix Opérations", IEEE Trans. on Computer, 6.
[JOU 85] Jou, J.Y., Abraham, "Fault-Tolerant Matrix arithmetique and signal" Processing on Highly Concurent Computing Structure.
[KUM 88], Prasana Kumar V.K, Tsai Y. C. "Synthesizing optimal family of linear systolic arrays for matrix multiplications", in IEEE proc. of Int. Conf. on systolic arrays. pp. 51-60.
[KUM 89], Prasana Kumar V.K, Tsai Y. C. "On mapping Algorithm to Linear Fault-Tolerant Systolic Arrays", IEEE Trans. Computer. Vol 38, 3.
[KUN 84] H.T. Kung, "Why systolic architecture ?", IEEE Computer, i5(1), pp. 37-46.
[KUN 84] Kung, H.T. & Lam, M.S., Wafer Scal Integration and two Level Pipelined Implementation Systolics Arrays", J. of Parrallel and Distributed Computing.
[LAM 74] Lamport L., "The parallel execution of DO loops", Comm.. ACM, 17, 2, pp. 83-93.
[LEE 88] Lee P., Kedem Z.M., "synthesizing linear algorithms from nested for loops algorithms" IEEE trans. Computers 37 12, pp. 1578-1598.
[LEI 83] Leiserson C.E., Saxe J.B., "Optimal synchronous systems", J. VLSI and Computer Systems 1, 1 pp. 41-68.
[MIR 84] W.L. Miranker and A. Winkler, "A space-time representation of computational structures", Computing, 32, pp. 93-144
[MOL 82] Moldovan, D. I., "On the analysis and synthesis of VLSI algorithms", IEEE Transactions on Computers, C-31, 11, pp. 1121-1126.
[NAY 90] A. Nayak, N. Santoro, and R. Stefanelli, "Fault-Tolerance techniques for array structures used in supercomputing", IEEE Computer, FTCS'20, pp. 202-209.
[QUI 84] Quinton, P. "Automatic synthesis of systolic arrays from uniform recurrence equations", Proc. IEEE 11th International Conference on Computer Architecture, Ann Arbor, MI, pp. 208-214.
[SAK 89] Sakho, I.& Tchuenté, M.,"Méthode de Conception d'algorithmes parallèles pour réseaux réguliers", Technique et Science Informatiques, 8, 1, pp. 63-72.
[RAM 84] Ramakrishan I.V, Varman P.J., "Modular matrix multiplication on a linear array", IEEE trans. computers 33, pp. 952-958.
[RAM 85] I.V. Ramakrishan, P.J. Varman, "An optimal family of matrix multiplication algorithms on linear arrays", in proc. Int. Conf. ICPP'85, IEEE Press, pp.376-383.
[TAN 91] Tangy Risset, "Linear Systolic Arrays for Matrix Multiplication: Comparisons of Existing Synthesis Methods and New Results", Algorithms and Parallel VLSI Architectures II, P. Quinton and Y. Robert eds., pp. 163-174, Elsevier Science Puplishers.
[TAN 94] Tangy Risset, "Parallélisation automatique: du modèle systolique à la compilation des nids de boucles", Phd Thesis, Ecole Normale Supérieur de Lyon.
[TCH 91] Tchuenté, M. "Parallel Computation on Regular Arrays", Manchester University Press and John Wiley & Sons.