6 milliards d'opérations logiques par seconde : simulation de phénomènes naturels par automates cellulaires en utilisant une matrice SIMD bidimensionnelle (GAPP) Application à la diffusion de particules

6 billions, mono-bit operations per second : the cellular automata approch to the simulation of natural phenomenon, using a systolic gapp array. Application to the diffusion

Cemal DRAMAN

ENSPS, URA CNRS n° 1207, 7, rue de l'Université, 67000 Strasbourg

Maîtrise de physique en 1977, DEA de physique des solides en 1978 à Strasbourg. Doctorat du 3e cycle en 1981, sur « les systèmes hybrides (optique et électronique) de traitement d'image », à l'École Nationale Supérieure de Physique de Strasbourg. Chercheur contractuel au sein de LSIT/ENSPS depuis 1981, entrecoupé par un poste d'Ingénieur d'Études (1983-1986) dans l'industrie, chez SECAPA Informatique, Dardilly.

Axes des recherches : architectures de machines de traitement d'images et de machines multi-processeurs, processeurs vectoriel et SIMD, environnement logiciel de systèmes multi-processeurs, algorithmes de traitement d'images, et de simulation par automates cellulaires.

Yannick HERVÉ

ENSPS, URA CNRS n° 1207, 7, rue de l'Université, 67000 Strasbourg

Élève de l'École Normale Supérieure de Cachan de 1982 à 1986. Maîtrise de Génie Électrique en 1984. Agrégation de Génie Électrique en 1985. Diplôme d'Études Approfondies en instrumentation et traitement du signal à l'IEF d'Orsay en 1986. Ancien Normalien Doctorant à partir de Septembre 1986 à l'École Nationale Supérieure de Physique de Strasbourg au sein du Laboratoire des Sciences de l'Image et de la Télédétection. Nouvelle Thèse de Doctorat en Avril 1988, sur l'optimisation des structures SIMD de calcul massivement parallèle et l'algorithmique bit-série. Maître de conférences au sein du LSIT depuis janvier 1989.

Axes des recherches : architectures de machines pour le traitement d'images, la simulation de phénomènes physiques et la visualisation scientifique, environnement et emploi optimaux des structures de calcul SIMD, interactions signal-algorithmes-architectures.

RÉSUMÉ

L'approche « automate cellulaire » des phénomènes physiques pose des problèmes de simulation difficiles à résoudre par des calculateurs classiques. Les temps de calcul pour des processus itératifs, l'accès pas toujours simple à de très gros ordinateurs, font que plusieurs équipes travaillent à l'étude de machines programmables dédiées à ce genre d'applications. Cette approche est étudiée dans les références [8] à [17]. Le LSIT de Strasbourg a développé une machine multi-processeurs de traitement d'images bas et moyen niveau orientée temps réel vidéo ([1] à [7]). La puissance de calcul apportée par une matrice SIMD de processeurs GAPP (« Geometrical Arithmetic Parallel Processors » de NCR) intégrée dans ce système nous permet de l'utiliser facilement et de façon très efficace pour les simulations de type « automates cellulaires » en visualisant toutes les itérations de calcul.

Après avoir présenté succinctement la machine, nous présenterons le modèle simulé. Nous donnerons quelques détails sur la gestion des entrées-sorties pour cet exemple, puis indiquerons les performances atteintes. Nous finirons en soulignant les limitations et les perspectives envisagées.

MOTS CLÉS

Automates cellulaires, systèmes de traitement d'images, systèmes multi-processeurs, systèmes SIMD, GAPP, calcul parallèle, parallélisation, simulation, modélisation, diffusion.

SUMMARY

The « cellular automata » approch (see the references [8] *to* [17]*) to the simulation of natural phenomenon leads up to computation problems essentially due to the required computation power and to the data transfer bottleneck between processing elements and memories.*

A real-time image processing system for low and medium level processing that we developped at LSIT, Strasbourg, enabled us to solve these problems. A systolic SIMD array (GAPP of NCR) gaved us enough processing power to handle the iterative cellular automata method.

In this paper, we will first describe our image processing system (MASYVE) and give the simulated diffusion model. We will then describe the management of data transfers. At last we will present the achieved performances, the limitations of the system and the perspectives.

KEY WORDS

Cellular automata, image processing systems, multi-processor systems, SIMD, GAPP, parallel computing, simulation, modeling, diffusion.

1. Généralités sur la machine MASYVE

1.1. INTRODUCTION

Développée dans le cadre du projet ESPRIT-1 P26, MASYVE (Machine à Architecture SYstolique et VEctorielle) a été conçue dans le but premier de résoudre les problèmes de temps de calcul en traitement d'images bas niveau (expansion de contraste, filtrage, seuillage, extraction de contours...) qui surchargent énormément les machines classiques dans des applications plus complètes.

MASYVE nous permet d'avoir une puissance de calcul de plusieurs GIGA_OPS (opérations/sec) pour des opérations locales (qui mettent en jeu un voisinage immédiat des pixels d'un image). Le processeur « GAPP » utilisé dans ces cas étant une association de 2 304 processeurs bit-série d'une puissance de 10 MIPS chacun, dont les instructions elles-mêmes permettent d'effectuer 5 opérations mono-bits en parallèle, la puissance de calcul globale dépend de la complexité de l'opération effectuée. Typiquement, on atteint 0.6 GIGA_OPS pour la somme de 2 octets et 1 GIGA_OPS pour une opération logique entre un octet et une constante.

Pour les opérations globales vectorisables (FFT, ROTATION), qui sont irréalisables par un réseau de type GAPP, on dispose, avec le processeur vectoriel ZIP 3216, d'une puissance de 10 MEGA_OPS sur les entiers 16 bits.

1.2. DESCRIPTION DE LA MACHINE

La figure 1 donne un schéma bloc de la machine générale. Les éléments constitutifs sont :

— un ordinateur hôte (type IBM PC-AT), qui sert à gérer les processeurs. Il est aussi utilisé comme l'interface-utilisateur ;

— un processeur de gestion d'images et d'entrée-sortie vidéo ICOTECH. Son intelligence locale (un microprocesseur 80186) lui permet de gérer le transfert d'images entre les processeurs ;

— un processeur GAPP qui est en fait un réseau SIMD de 48*48 processeurs mono-bit ;

— un réseau de processeurs vectoriels utilisé pour les opérations globales vectorisables (FFT, ROTATION).

Ces éléments sont connectés à un bus de contrôle MULTIBUS I et à un bus de données rapides qui nous permet de transférer simultanément jusqu'à 8 images en 40 ms (une image vidéo).

On ne décrira ici en détail que la partie utilisée comme automate cellulaire, le réseau « GAPP » de NCR, qui nous permet de faire les calculs de la simulation. Pour une description plus complète de la machine et de ses applications se reporter aux références [1] à [7].

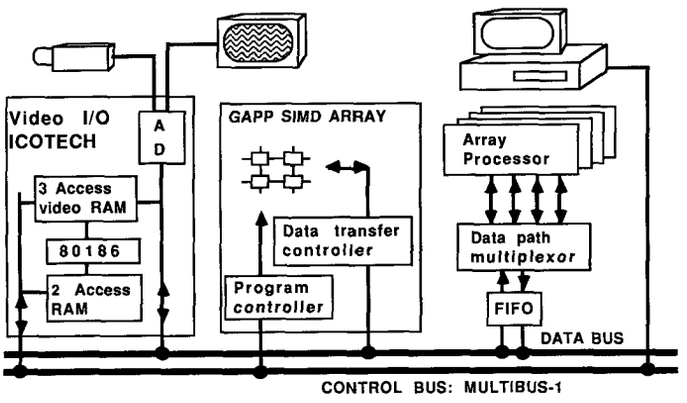


Fig. 1. — MASYVE, système multiprocesseur de traitement d'image.

1.3. LE PROCESSEUR SIMD (GAPP)

Le processeur SIMD est une matrice systolique bi-dimensionnelle de processeurs mono-bit utilisant des circuits GAPP (Geometric Arithmetic Parallel Processor) de NCR. Ce VLSI est une matrice de 6*12 processeurs mono-bit connectés aux quatre plus proches voisins (Nord, Sud, Est, Ouest ; symbolisés respectivement par *n*, *s*, *e*, *w*). Chaque processeur élémentaire (PE) est constitué d'une ALU bit série, d'une mémoire RAM statique de 128 bits et de quatre registres 1 bit. Trois de ces registres sont les entrées de l'ALU tandis que le quatrième permet d'effectuer des entrées-sorties (ES) à travers la matrice sans interrompre le travail de l'ALU. Une sortie de l'état global (GO = OR de tous les registres NS) permet de tester une condition d'ensemble sur la matrice. La cascadabilité de ce circuit permet de fabriquer des matrices d'une taille quelconque par multiple de 6*12 PEs. Cette structure fonctionne en mode SIMD. (Single Instruction stream-Multiple Data stream d'après la taxonomie de FLYNN). La figure 2 représente l'architecture interne d'un PE.

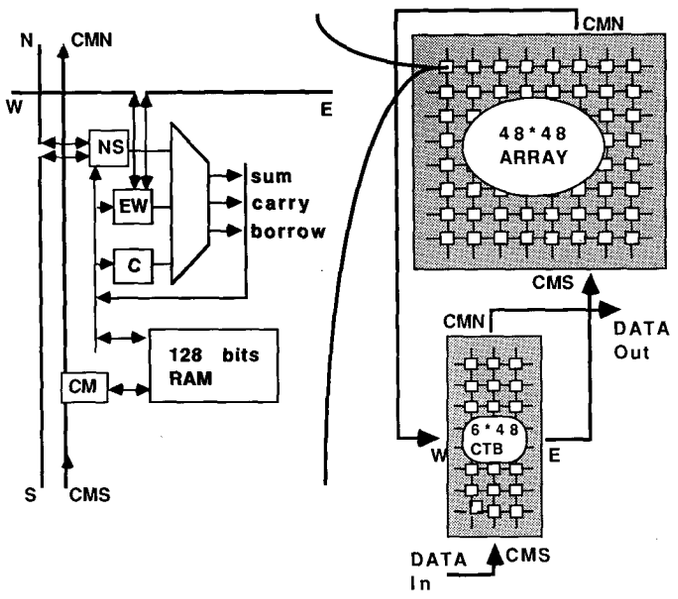


Fig. 2. — Processeur élémentaire et matrice GAPP.

Chacun des registres et la mémoire ont leur entrée pilotée par un multiplexeur. Les bits de sélection de chacun de ces multiplexeurs représentent le microcode assigné à chacun des registres. L'adresse de la mémoire et ces 5 micro-instructions sont représentés par un mot de 20 bits. Du fait de ce codage horizontal, le PE effectue les 5 micro-instructions (opérations binaires) en un cycle d'horloge. Le tableau suivant décrit les 6 champs.

Les instructions de la matrice de calcul GAPP :

bits	(0-1)	(2-4)	(5-7)	(8-10)	(11-12)	(13-20)
cm :=	cm	ns := ns	ew := ew	c := c	ram := « read »	(adresse
	ram	ram	ram	ram	cm	RAM)
	cms	n	e	ns	c	
	0	s	w	ew	sum	
		ew	ns	carry		
		c	c	borrow		
		0	0	0		
				1		

L'ensemble des 48*48 registres EW, BS, C et CM peut être considéré comme 4 plans de bits indépendants. Le fonctionnement SIMD fait que ces différents plans sont manipulables indépendamment de la taille de la matrice de calcul et sont indifférentiables selon les PE.

Pour l'entrée des données, on ne dispose que des bus mono-bits situés sur les bords de la matrice de calcul. La seule solution pour distribuer des données sur le processeur est donc la propagation : c'est le rôle des lignes CMS vers CMN. L'indépendance des registres CM (CoMmunication) et de l'ALU d'une part, le codage horizontal des micro-instructions d'autre part, permet l'exécution concurrente des entrées-sorties et du programme de calcul. Ceci permet de qualifier ce processeur de systolique puisqu'un flot continu d'E/S peut être obtenu. Pour notre part nous préférons qualifier ce mode de fonctionnement de SIMD quasi-systolique puisque le fonctionnement externe peut être assimilé à un fonctionnement systolique alors que le fonctionnement interne est purement SIMD.

1.4. LES CARTES DE CALCUL GAPP

Le processeur disponible au laboratoire est un prototype de la firme NCR. Il est composé de deux cartes MULTIBUS I : le processeur de calcul et le micro-séquenceur avec la mémoire programme. La matrice de calcul (MATRICE) est composée de 32 circuits organisés en une matrice de 48*48 PEs et de 4 circuits pour l'organe de formatage des données (Corner Turn Buffer ou CTB) organisés en une matrice de 48*6. Celui-ci utilise seulement un sous-ensemble des instructions, une partie des bits étant cablée à 0.

Les instructions de l'organe de formatage (CTB) :

bits	(0-1)	(2-3)	(4-5)	(6-12)
	ew := ew	cm := cm	ram := « read »	(adresse
	ram	ram	cm	ram)
	e	ns	c	
	w	ew	sum	

L'autre carte comprend le micro-séquenceur, la mémoire programme ainsi que toute la logique de contrôle. Cette mémoire a une taille de 32 Kmots de 47 bits (20 bits pour la matrice, 13 bits pour le CTB, 5 bits pour le séquenceur et 9 bits pour les adresses de sauts relatifs). La figure 2 montre les connexions entre la matrice et le CTB.

1.5. GESTION DES CARTES

Le programme qui pilote le processeur quasi-systolique est composé de trois modules concaténés : les modules de calcul, d'entrée-sortie (IO) et de formatage des données (TK ou Takeover).

Le module d'E/S (IO) concerne l'entrée-sortie du CTB qui est considéré comme un registre à décalage de 6 bits, de 48 bits de long. Les données à l'entrée et à la sortie du CTB sont des mots de 6 bits présentés les uns après les autres.

Le module de formatage (TK) concerne les transferts de données entre le CTB et la matrice. Il transforme les données (48 mots de 6 bits) en 6 plans de 48 bits (formatage).

Le module de calcul effectue l'algorithme proprement dit.

Le tableau suivant montre les ressources (registres et mémoire) utilisées par ces 3 modules :

Organe	Matrice		CTB					
Ressources / Module	NS	EW	C	RAM	CM	EW	RAM	CM
	Ressources utilisés							
Entrée-Sortie (IO)	—	—	—	—	—	—	RAM	CM
Formatage (TK)	—	—	—	RAM	CM	EW	RAM	—
Calcul	NS	EW	C	RAM	—	—	—	—

Les programmes IO et TK sont synchronisés au moment de l'écriture par le programmeur, mais ne sont écrits qu'une seule fois. Ces programmes dépendent de la taille du réseau de calcul et du CTB. Le programme de calcul est écrit indépendamment des deux autres et de la taille de la matrice de calcul. La synchronisation s'effectue au moment de la compilation par une concaténation horizontale des instructions des trois programmes, les conflits d'accès mémoire étant résolus à ce moment par le logiciel. La séquence d'entrées-sortie étant prioritaire sur le programme de calcul, en cas de conflit une instruction NOP est ajoutée au programme de calcul en regard de l'instruction d'accès mémoire du programme d'E/S. Ces trois programmes sont décrits en détails dans [3].

1.6. L'INTERFACE GAPP — BUS_VIDEO : LE FENETREUR

Ce système permet d'interfacer le processeur GAPP aux bus vidéo de ICOTECH. Il est principalement constitué de quatre plans de mémoire d'image et d'une logique de contrôle permettant de gérer le fenêtrage de l'image suivant la taille de la matrice de calcul, le retard entre l'entrée et la sortie, la gestion des points faux, le flux de données, l'horloge du processeur GAPP et le rôle des plans-image (*fig.* 3).

Le processeur étant plus petit que l'image à traiter il faut procéder au découpage de l'image. Différentes techniques peuvent être envisagées ; découpage en fenêtres correspondant à la topologie du processeur, traitement par lignes entières, ... Dans la suite de cet exposé les images sont découpées en fenêtres correspondant à la topologie du réseau de PEs et on associe un pixel à un processeur.

Pour une taille d'image donnée le nombre de fenêtres dépend de la taille du réseau utilisé et du voisinage « vu » par l'opérateur de traitement (les points calculés sur le bord du réseau étant faux, un recouvrement adéquat des fenêtres est nécessaire).

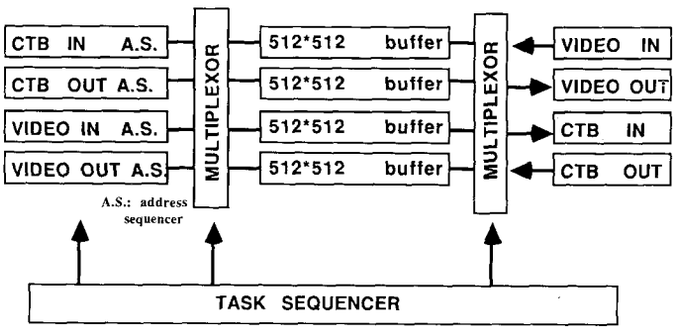


Fig. 3. — Fenêtreur.

Le nombre de fenêtres à calculer peut être exprimé comme suit :

$$Nx = \frac{Ix - Kx + 1}{X - Kx + 1} \quad \text{pour la direction X .}$$

Avec

Ix taille de l'image dans la direction X
X taille du processeur dans la direction X
Kx voisinage de l'opérateur dans la direction X
Nx entier directement supérieur ou égal au rapport.

On a alors le nombre de fenêtres total N

$$N = Nx * Ny .$$

La programmation de cette interface est effectuée à partir de l'ordinateur hôte au début de chaque application, en transférant la liste des coordonnées et la taille de chaque fenêtre à traiter par passe. Les transferts d'images ne sont possibles que toutes les 40 ms, synchronisés par les signaux d'E/S vidéo. Les transferts entre les plans images et le GAPP sont effectués en décodant « au vol » les instructions du programme GAPP (présence ou non d'une instruction de demande de donnée).

1.7. LE LANGAGE

Le langage de programmation du GAPP est le Gapp Algorithm Langage (GAL). C'est un sous-ensemble du langage C. Les structures de contrôle sont des structures du C (boucles, tests, opérations logiques et arithmétique...) et les instructions compilées sont en assembleur GAPP.

Ce langage permet de définir des variables images qui se trouvent dans la mémoire des PEs. On peut les déclarer de façon absolue en fixant l'adresse et la taille de chaque variable image, ou de façon relative, en indiquant seulement sa taille, le compilateur se chargeant de la gestion de leur adresse. On peut évidemment déclarer ces variables globalement ou localement.

Les variables non-images sont des entiers sur 16 bits. Elles servent d'index de boucles, de définitions de la taille des variables images ou de coefficients globaux pour une opération sur l'image. Une opération spéciale appelée SIZE établit le lien entre les deux types de variables. L'instruction s = size(IMA) fait correspondre à la variable entière s le nombre de bits de l'image IMA.

Les branchements et boucles se résument à :

— IF (expression) instruction_1 else instruction_2
— WHILE (expression) instruction_1
— FOR (expression_1 ; expression_2 ; expression_3) instruction

les expressions ne pouvant utiliser que des variables non-images.

La définition d'une instruction est la même qu'en langage C : c'est une instruction seule ou un ensemble d'instructions comprises entre accolades. Une instruction peut être un traitement par le langage C de constantes et de variables entières ou une instruction assembleur GAPP composée des CINQ champs de micro-instructions. Ceux-ci ont été décrits dans le chapitre 1.3.

Un exemple d'addition de deux images en calcul bit-série SIMD, le résultat étant rangé dans une troisième image :

```
image i1 : 6, i2 : 6, i3 : 7 ;        /* déclaration des images */
int i ;                               /* une variable compteur */
c:=0 ;                                /* le ';' sépare les instructions */
for (i=0 ;i<size(i1) ;i++)            /* boucle */
  {
   i1 : i  ew:=ram ;                  /* bit [i] de image_1 */
   i2 : i  ns:=ram ;                  /* bit [i] de image_2 */
   i3 : i  ram:=sum c:=carry ;        /* bit [i] du résultat */
  }
i3 : i  ram:=c ;                      /* le dernier bit */
```

Les champs de micro-instructions non remplis sont forces à NOP. Parallèlement à ce calcul, on programme l'entrée de la fenêtre suivante et la sortie de la fenêtre précédente en utilisant les registres CM.

Ici on voudrait rappeler, pour la compréhension des algorithmes décrits, un des mécanismes de programmation utilisées en SIMD. Le terme SIMD (Single Instruction, Multiple Data) indique que les branchements conditionnés par des données-image (séquence IF...THEN...ELSE...) ne sont pas réalisables, car tous les processeurs effectuent le même calcul quelque soit la donnée qu'ils utilisent. Une technique simple nous permet de simuler ce branchement et de pouvoir faire des calculs complexes. Il s'agit tout simplement d'effectuer les 2 branches de la séquence IF...THEN...ELSE... et de valider un des résultats en se servant de la condition de branchement (technique du multiplexeur).

Le branchement conditionnel

IF a
  code1 → r
else
  code2 → r

est implanté comme suit :

calcul de a
code1 → $r1$
code2 → $r2$
$r = r1 \,.\, a + r2 \,.\, !a$ [pour tous les bits]

où :
a est un booléen
r est une image
$r1$, $r2$ deux images intermédiaires
. est le ET logique
+ est le OU logique
! est le complément logique.

Un autre point important est l'indépendance du programme par rapport à la taille du réseau. Le programme est en effet écrit comme si le réseau avait une dimension infinie. Ceci se traduit tout simplement par le fait qu'un résultat qui dépend d'un voisinage qui déborde du réseau est faux. Il faut donc choisir les fenêtres-sources de façon que les fenêtres-résultats se juxtaposent. C'est précisément le rôle du fenêtreur. Le programme est alors exécuté sur chaque fenêtre, et n'a pas à tenir compte des tailles de l'image, du réseau et du voisinage.

1.8. QUELQUES AUTRES MACHINES

Avant d'aller plus loin, on voudrait situer MASYVE par rapport à quelques machines simulant des automates cellulaires. Ces machines ont des architectures différentes, leurs programmations sont plus ou moins compliquées, elles sont utilisées dans des domaines différents. On se bornera donc à rappeler ici leurs principales caractéristiques.

Abréviations :

MUPS Mega Update Per Seconde
HPP : modèle de gaz sur réseau carré
FHP : modèle de gaz sur réseau hexagonal

RAP-1 : [8], [9]
  machine spécialisée, dérivée du modèle FHP
  512*256 sites.
  Calcul par utilisation de LUTs (tables de correspondances)
  Traite un voisinage 3*3.
  16 bits par site.
  6.5 MUPS.

RAP-2 [8], [9] (projet)
  2*1024*1024 sites de 16 bits.
  Calcul par utilisation de LUTs.
  voisinage 5*5
  100 MUPs prévus.

OUPPI-1 [10], [11]
  utilise 2 processeurs GAPP
  12*12 PE, pas de mémoire externe
  14 MUPS sans affichage (modèle HPP, 84*84 sites).

MASYVE [1] à [7]
  voisinage quelconque, 512*512 sites
  6 ou 12 bits par site, 128 bits de travail par site
  23 MUPS avec 1 affichage sur 7 passes (modèle HPP)
  32 MUPS sans affichage (modèle HPP)
  prix estimé : 350 KF avec l'hôte.

à comparer avec CRAY-1 : 30 MUPS sans affichage (modèle FHP, [8]).

Les performances de ces machines sont toutes aux alentours de 10 MUPS. Les machines RAP se distinguent par leur spécificité à un certain type d'algorithme. OUPPI-1 (premier membre d'une famille) est complètement programmable (à base de GAPP) mais a une structure figée. MASYVE paraît être un bon compromis, étant une machine à structure ouverte, aux performances élevées, complètement programmable et à un prix abordable.

2. Application : simulation par automates cellulaires

2.1. LE BUT DE LA SIMULATION

La simulation présentée ci-après est un exemple qui sert à illustrer les possibilités de la machine. L'équipe dont est issu ce travail n'est pas spécialiste des phénomènes simulés. Le modèle développé illustre qu'un modèle microscopique simple peut générer un comportement macroscopique réaliste. Ce passage, « modèle microscopique » → « comportement macroscopique », nécessite de gros moyens de calcul et de transfert de données du fait de la répétition du calcul pour toutes les « cellules élémentaires ». Les résultats obtenus sont encourageants et une collaboration avec des physiciens est en cours pour l'implantation de modèles dont les comportements « réels » ont été démontrés. Pour l'étude théorique de la simulation par automates cellulaires, le lecteur peut consulter les travaux de d'Humières [8] notamment pour des gaz sur réseaux.

2.2. LE MODÈLE. DESCRIPTION

Le phénomène retenu pour l'exemple est la diffusion de particules. Ces particules pourraient représenter des molécules de gaz ou de liquide. Le modèle actuel ne tient compte que de la concentration locale, instantanée (donc pas de la vitesse de déplacement). C'est une description assez fidèle de la diffusion de substances dans un milieu poreux, du mélange de deux liquides miscibles...

Le « monde » de ce modèle est composée :

— de 2 enceintes étanches qui communiquent
— de particules qui remplissent ces enceintes.

Ces particules se déplacent selon des règles d'évolutions suivantes : on associe à chaque pixel une valeur représentant soit le nombre de particules sur ce site, soit l'état d'obstacle (les parois de l'enceinte). A chaque itération les particules se répartissent sur les plus proches voisins (ils diffusent), le nombre de particules en un site après diffusion étant la somme des contributions des voisins. Cette diffusion ne sera donc sensible que dans les zones à gradient de densité puisque dans les zones « homogènes » les diffusions mutuelles se neutraliseront.

2.3. LE MODÈLE. L'ALGORITHME

Un pixel de l'image est codé sur 6 bits (64 niveaux) et on décide d'effectuer le codage suivant pour la valeur du pixel :

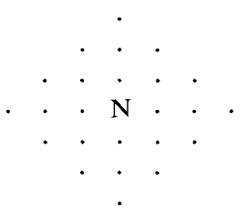
$x = 63$: obstacle ou enceinte
$x = 62{,}61$: réservé pour une utilisation future
$x < 61$: nombre de particules.

Le voisinage retenu est le V4 (4 plus proches voisins). La diffusion consiste alors à distribuer les particules d'un site aux 4 voisins cardinaux (le reste de la division par 4 est distribué aléatoirement). Pour augmenter la vitesse du traitement, on répète l'algorithme de diffusion 3 fois par image visualisée.

Cas d'un « paquet de particules isolé » :

$$N \xrightarrow{\text{diffusion}} \begin{matrix} & N/4 & \\ N/4 & 0 & N/4 \\ & N/4 & \end{matrix}$$

3 itérations de cet algorithme élémentaire produit une diffusion sur un voisinage 7*7 :



Exemple :

$$\begin{matrix} & & & & & & 3 & & & & 1\ 3\ 1 \\ & & 15 & & & 6\ 3\ 6 & & & 1\ 2\ 8\ 2\ 1 \\ 60 \rightarrow & 15\ 0\ 15 & \rightarrow & 3\ 3\ 12\ 3\ 3 & \rightarrow & 3\ 8\ 0\ 8\ 3 \\ & & 15 & & & 6\ 3\ 6 & & & 1\ 2\ 8\ 2\ 1 \\ & & & & & & 3 & & & & 1\ 3\ 1 \end{matrix}$$

Le séquencement des opérations est le suivant :

— détermination du type de pixel (gaz ou obstacle)
— remise à 0 si obstacle
— diffusion (3 fois)
  • calcul des particules qui partent vers e, w, n, s
  • distribution aléatoire du reste de la division/4
  • récupération des particules qui arrivent, somme
  • test : $n > 60 \rightarrow n = 60$
— réinitialisation de l'état d'obstacle.

Rappels :

— Ce calcul se fait en parallèle sur tous les sites d'une fenêtre.
— La technique du multiplexeur (chapitre 1-7) nous permet de distinguer les sites_obstacles des sites_non_obstacles.

2.4. RÉSULTATS DE LA SIMULATION, VISUALISATION

Chaque image calculée est visualisée à travers des tables de couleurs : les niveaux 61-63 sont affichés en blanc et une échelle de couleurs est utilisée pour représenter la densité locale de particules (niveaux 0 à 60).

Les photos 1, 2, 3 et 4 présentent différentes étapes de la diffusion. La photo 1 représente une vue générale des enceintes après 1 000 itérations, la photo 2, 3 et 4 des vues

aggrandies de l'orifice respectivement à l'instant initial et après 50 et 500 itérations.

2.5. COMMENTAIRES SUR LES RÉSULTATS

L'algorithme élémentaire

$$N \xrightarrow{\text{diffusion}} \begin{matrix} & N/4 & \\ N/4 & 0 & N/4 \\ & N/4 & \end{matrix}$$

est très fortement anisotrope : elle ne prend en compte que les 4 directions cardinales. Cette anisotropie se retrouve localement dans des zones plus ou moins grandes (photo 4) sous forme de structures microscopiques dont la maille élémentaire est de 2*2 pixels. Ceci nous indique qu'un moyennage du résultat sur 2*2 pixels ferait complètement disparaître cette anisotropie.

On peut en déduire qu'un algorithme très simple et très rudimentaire au niveau microscopique peut produire une évolution macroscopique du système qui est très proche de la réalité. Si on considère les avantages d'une telle approche, un des plus attractifs est le temps de développement minime qui correspond aux algorithmes d'application. Pour cet exemple, le temps de développement a été de 2 semaines-homme pour une personne avec une bonne connaissance des langages évolués et une expérience de 4 semaines en GAL.

3. Un autre exemple : soufflerie

En ajoutant très peu de code à l'algorithme décrit précédemment, il nous est facile de faire « bouger » uniformément les particules et d'obtenir un écoulement. Il suffit d'ajouter des zones de génération et d'absorption de particules pour avoir un écoulement continu.

Le « monde » de ce nouveau modèle est composé :

— d'une enceinte étanche
— d'un générateur de particules
— d'un absorbeur de particules (sortie de veine)
— d'obstacles à étudier.

3.1. LES NOUVELLES RÈGLES D'ÉVOLUTION

Comme avant, on associe à chaque pixel une valeur représentant soit le nombre de particules sur ce site, soit l'un des états d'obstacle, de générateur ou d'absorbeur. A chaque itération, le générateur produit des « paquets » de particules de façon homogène. Ces « paquets » sont animés d'une vitesse initiale unitaire, de la gauche vers la droite. Ceux qui arrivent sur « l'absorbeur » disparaissent. Autrement ils se répartissent sur les plus proches voisins non-obstacles.

L'algorithme qui en résulte est très semblable au précédent. Le tableau suivant donne la classification des pixels selon leurs valeurs.

$x = 63$: obstacle ou enceinte
$x = 62$: générateur de particules
$x = 61$: absorbeur (met à 0)
$x < 61$: nombre de particules.

Le séquencement des opérations est alors le suivant :

— détermination du type de pixel (obstacle, générateur, absorbeur ou gaz)
— remise à 0 si obstacle, générateur ou absorbeur
— génération : N = N + 31 sous le générateur
— déplacement du gaz de 1 pixel
— diffusion (3 fois)
  - calcul des particules qui partent vers *e*, *w*, *n*, *s*
  - distribution des particules aux voisins non-obstacles ; distribution aléatoire du reste
  - récupération des particules qui viennent (somme)
  - test : $n > 60 \rightarrow n = 60$
— réinitialisation de l'état d'obstacle, de générateur ou d'absorbeur.

3.2. LES RÉSULTATS

Les photos 5 à 7 représentent les différentes étapes de la simulation. 5 et 6 correspondent respectivement aux rencontres du gaz avec les obstacles et le régime quasi-stationnaire atteint à partir de $t0 + 600$ passes ($t0 + 15$ secondes). Cet état est quasi-stationnaire puisque dans les « sillages des ailes » on peut remarquer des turbulences et des instationnarités. La photo 7 représente un agrandissement d'une des ailes. On y remarque bien une surpression qui s'établit sur l'intrados mais on y remarque aussi que le modèle ne correspond pas du tout à un « gaz » classique mais plutôt à un liquide visqueux.

3.3. L'ANALYSE STATISTIQUE

Il est possible, sans diminuer les performances, de fournir l'image issue d'une itération à l'étage de calcul de moyen niveau, formé du réseau de processeurs vectoriels.

Il est donc envisageable d'effectuer des calculs complémentaires à la simulation (statistique, intégration locale, densité, répartition de vitesse...) sur le résultat, le temps de calcul dépendant bien sûre de la complexité du calcul effectué. Le transfert d'une image vers un processeur vectoriel se faisant à la cadence vidéo, pour un algorithme simple (temps de calcul =< temps de transfert), il est possible d'avoir un résultat tous les 2 images.

Cette voie prometteuse n'a pas encore été approchée dans l'attente des désidérata des physiciens utilisateurs de la machine.

4. La gestion de la machine pour ces exemples

4.1. LA GESTION DES IMAGES

Les quatre plans de mémoire d'image du fenêtreur sont numérotés de M1 à M4. Les différentes tâches affectables à ces plans mémoires sont les suivantes :

S source de calcul GAPP
R résultat de calcul GAPP
A acquisition sur bus vidéo
V restitution sur le bus vidéo (visualisation)
— attente.

Le séquencement des opérations pour la simulation est le suivant :

Étapes	Mémoires M1	M2	M3	M4	Remarques
1/	A	—	—	—	chargement BUS-VIDÉO → Mem.
2/	S	V	R		
3/	R	R	S	V	puis reprise à l'étape 2

Chacune des étapes 2 et 3 correspond à l'application d'une passe de la simulation sur l'image complète. Le résultat est écrit dans deux mémoires, l'une pour l'itération suivante, l'autre pour la visualisation.

L'image initiale chargée à l'étape 1 est une image stockée sur le disque de la machine hôte (voir photo 2). Chaque étape dure 40 ms. La visualisation « V » à chaque étape est en fait la restitution de l'image sur le bus vidéo. La station ICOTECH se charge alors d'effectuer la mémorisation du résultat et sa visualisation à travers des tables de correspondances en fausse couleurs.

4.2. LA GESTION DES FENÊTRES

Appliqué à une image 512*512, cet algorithme qui utilise un voisinage de 7*7 pixels, produit un résultat de 506*506 pixels. On n'en retient, par souci d'optimisation du temps de calcul, qu'une image 504*504 qui correspond à 12*12 fenêtres de 42*42 pixels, elles-mêmes résultat de fenêtres 48*48.

Le programme est itéré sur chacune des fenêtres de l'image. Le retard entre l'entrée et la sortie du processeur oblige l'interface à gérer des fenêtres fictives. Le tableau suivant représente l'organisation temporelle des traitements (entrée-sortie (ES) et calcul) en fonction du numéro des fenêtres, la technique utilisée est appelée « Double-Ram-Based » (les méthodes de gestion des ES sont décrites dans [2])

T1 : passage d'une fenêtre de la mémoire dans le CTB
T2 : passage d'une fenêtre du CTB à la MATRICE (TakeOverIn)
T3 : calcul sur la fenêtre dans la MATRICE
T4 : passage d'une fenêtre de la MATRICE au CTB (TakeOverOut)
T5 : sortie de la fenêtre du CTB vers la mémoire

n = numéro de la fenêtre

T1	1	2	3	4	5	6	...	144	—	—	—	—	
T2	—	1	2	3	4	5	...	143	144	—	—	—	
T3	—	—	1	2	3	4	...	142	143	144	—	—	
T4	—	—	—	1	2	3	...	141	142	143	144	—	
T5	—	—	—	—	1	2	...	140	141	142	143	144	

temps

On remarque que dans le cas présent il faut attendre 4 fenêtres pour récupérer les résultats. On gèrera donc un nombre fictif de 148 fenêtres. Ceci correspond à une passe de simulation.

5. Performances et limitations

5.1. PERFORMANCES OBTENUES

La machine dispose actuellement d'une matrice de 48*48 processeurs élémentaires (2 304 PEs) fonctionnant à 10 MHz (une instruction par cycle d'horloge).

Le voisinage scruté par itération étant de 7*7 pixels, l'image utile est de 504 par 504 pixels de 6 bits (254 016 pixels).

L'algorithme comporte 1 073 lignes de code utile, et génère 1 688 instructions après compilation. L'occupation mémoire est de 69 bits sur 128 bits disponibles par PE (54 %).

Le nombre d'opérations logiques (ol) par pixel et par itération est de 1 026. On entend par opération logique tout opérateur logique à trois entrées et une sortie.

Le nombre d'opérations binaire (ob) par pixel et par itération est de 2 901. Une opération binaire est toute opération portant sur un registre ou de la mémoire (sans compter les ES). Ce nombre est plus important que le nombre d'instructions du fait du microcodage des instructions : on peut réaliser au plus 4 microinstructions par cycle d'horloge et par PE (en fait 5 si on compte les ES).

Pour les entrées-sorties (ES) le programme comporte 2 413 instructions dont 1 688 (70 %) se déroulent en recouvrement temporel avec le calcul. L'image est découpée en 144 fenêtres utiles et 4 fenêtres de gestion du pipeline interne.

On a donc :

Image 512*512 pels sur 6 bits
voisinage 7*7 pels
matrice 48*48 PEs à 10 MHz
algorithme 1 688 instructions/fenêtre
(1 026 ol/pel
2 901 ob/pel)
gestion ES 2 413 instructions/fenêtre.

On obtient les performances temporelles suivantes :

40 ms entre deux affichages (imposé par la vidéo)
37,5 ms de calcul effectif avec ES
24,3 ms (68 % des 37,5 ms) pour le calcul seul.

Les performances suivantes sont données en cours d'utilisation, donc en supposant qu'une itération prend effectivement 40 ms.

6 350 400 pel/s en affichage (63,5 % du maxi)
6,51 Giga opérations logiques par secondes
18,42 Giga opérations binaires par secondes
19,2 Méga modif. de sites par secondes (3 passes par image).

5.2. LIMITATIONS ACTUELLES

La machine actuelle est limitée par la taille du pixel de 6 bits représentant un site de simulation. Il est possible sans modification, mais en divisant les performances par deux, de travailler sur des pixels de 12 bits.

La mémoire disponible par PE n'a pas été une limitation

pour les algorithmes étudiés, soit pour les automates cellulaires, soit pour les algorithmes de traitement d'images développés au laboratoire ; mais il est certain qu'avec la complexification des applications 128 bits seront insuffisants.

La mémoire externe disponible actuellement permet des simulations sur 512*512 pixels, pour certains phénomènes cette taille peut être trop juste.

Le maillage carré de l'espace n'est pas forcément le plus représentatif de notre monde du fait de l'inhomogénéité des distances aux voisins. Néanmoins, les résultats montrent que cette anisotropie n'existe qu'à l'échelle microscopique (voir le chapitre 2.5) pour les modèles étudiés. D'autre part, il existe aussi des modèles utilisant le pavage triangulaire (à 6 voisins) ou hexagonal (à 3 voisins). Ces modèles peuvent être simulés en utilisant un réseau carré (*fig.* 4). La performance, si elle est définie comme étant le rapport

$$\frac{\text{nombre de modification de sites par seconde}}{\text{nombre de voisins pris en compte}}$$

n'est pas affectée. Le problème principal est alors la gestion de l'affichage.

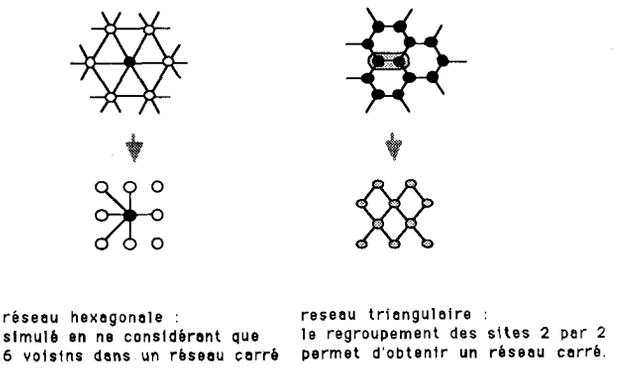


Fig. 4. — Réseaux carrés, hexagonaux, triangulaires.

La limitation la plus forte est sans doute le fait de se restreindre à un monde plan. La simulation 2D reste une approche pratique mais difficilement transposable aux phénomènes courants.

5.3. PERSPECTIVES

Le nombre de bit par pixel et la taille de l'image ne sont que des problèmes de coût et peuvent être résolus en s'en donnant les moyens.

Les problèmes de topologie du réseau peuvent se régler de façon satisfaisante mais les problèmes sont reportés sur les systèmes d'affichage.

Le problème le plus difficile à résoudre est la simulation en 3D avec les contraintes en performances et en coût qui en découlent. Bien que du matériel de visualisation 3D soit annoncé disponible sur le marché à court terme, le solution pour une machine parallèle ne peut, de façon satisfaisante, découler de la structure présentée ici. Un travail en profondeur doit être mené pour « penser » une machine spécialisée pour ce genre d'applications. Les solutions rencontrées actuellement sont souvent basées sur des réseaux de processeurs classiques, mais la gestion difficile de ceux-ci et les performances moyennes ne donnent pas entière satisfaction.

6. Conclusions

Cet article a présenté une machine de très hautes performances utilisable pour le traitement d'images bas niveau et la simulation par « automates cellulaires ». Ses performances nous permettent de simuler les phénomènes naturels macroscopiques en utilisant des modèles à l'échelle microscopiques.

L'intégration de plus en plus poussée, l'avènement du WSI (Wafer Scale Intégration), la montée des fréquences d'horloge permettent de penser que les machines du type de MASYVE ne sont qu'une étape intermédiaire dans la recherche de la puissance pour les outils mis à la disposition des scientifiques. A terme, chaque utilisateur pourra avoir sur (ou sous) son bureau une machine programmable dédiée à son application sans avoir recours aux supers ordinateurs centralisés d'une utilisation lourde et coûteuse. MASYVE est un premier travail qui a permis de bien comprendre les problèmes liés à l'utilisation de cette famille de processeurs.

Dès à présent un cahier des charges a été établi dans l'attente du nouveau composant ELSA (European Large SIMD ARRAY, [18], [19], [20]) de 128 par 128 PEs sur une tranche complète de silicium. Gageons que sa disponibilité nous fera avancer un peu plus vers les systèmes ultra performants demandés par les utilisateurs.

Contrats

* La machine MASYVE a été conçue dans le cadre du projet ESPRIT P26.

* L'application présentée a été développée dans le cadre du Pôle de Recherches Concertées — Architectures des Machines Nouvelles (CNRS).

* ELSA est développé dans le cadre du projet ESPRIT 824B.

* La construction de la machine autour de ELSA est financée par un projet ESPRIT II.

BIBLIOGRAPHIE

MASYVE et ses Applications

[1] R. Eugène, C. Draman, F. Pierre, Y. Hervé, S. Wendling, *Architecture Multi Parallèle, Application en Traitement d'Images.* 12e Colloque GRETSI, Juan-les-Pins, 12-16 juin 1989.

[2] R. Eugène, *Étude architecturale, modélisation et réalisation d'un processeur à base de GAPP pour le Traitement d'images et la simulation par automates cellulaires.* Nouvelle Thèse, février 1990, ULP, Strasbourg.

[3] Y. Hervé, *Mise en Œuvre et Optimisation de l'Utilisation d'un Processeur Quasi-Systolique dans une Chaîne de Traitement d'Images Temps Réel.* Nouvelle Thèse, avril 1988, ULP Strasbourg.

[4] Y. Hervé, F. Pierre, *Optimisation de l'Utilisation des Processeurs en Matrice : le Codage Optimal.* 1987 AFCET-RFIA, Antibes, pp. 1007-1014.

[5] F. Pierre, Y. Hervé, R. Eugène, C. Draman, S. Wendling, *Description d'une Machine de Traitement d'Images Temps Réel : l'Approche Multi-Parallélisme.* 1988 PIXIM, Paris.

[6] F. Pierre, Y. Hervé, R. Eugène, C. Draman, S. Wendling, *A Machine for Video Real Time Image Processing Using Parallel Polymorphism.* 1989 CA-DSP Hong-Kong.

[7] C. Draman, K. Zampieri, P. L. Wendel, Poste de traitement d'images configurable : ICOTECH 1986, Semaine Internationale de l'Image Électronique, Nice.

Les Automates Cellulaires

[8] D. D'Humières, A. Clouqueur, P. Lallemend, *Lattice Gases and Parallel Processors.* International Symposium on Vector and Parallel Processing for Scientific Computation. 2, Tome, Sept. 1987.

[9] A. Clouqueur, D. D'Humières, *R.A.P., a Family of Cellular Automaton Machines for Fluid Dynamics.* Helvetica Physica Acta Vol. 62, 1989.

[10] A. Perez, M. Cotton, G. Roger and J. Hervé, *OUPPI-1, A SIMD Computer using Integrated Parallel Processors.* CONPAR 88, Manchester, Sept. 1988.

[11] H. Ottavi, O. Parodi, Simulation of the ising model by cellular automata.

[12] J. Signorini, *Programming von Neumann's self-reproducing cellular automaton.* International Symposium on Computer Architecture and Digital Signal Processing, Hong Kong, October 1989.

[13] François Robert, Discrete Iterations, A Metric Study 1986 Springer Verlag.

[14] Tommaso Toffoly, Norman Margolus, Cellular Automata Machines, 1987 MIT press.

[15] Stephen Wolfram, Theorie and Applications of Cellular Automata 1986 World Scientific.

[16] Bernard Amy, Éric Decamp, Stéphane Guez, Réseaux d'Automates, Panorama des Recherches. 1987 national report IMAG.

[17] Éric Decamp, Bernard Amy, Neurocalcul et réseaux d'automates, le point sur les recherches et les applications. EC2, 1988.

Elsa et prospectives

[18] Saucier *et al.*, Tim Mindwinter *et al.*, Martin Hush *et al.*, ELSA : an European Large SIMD Array for Image Processing. EURO ASIC 89 pp. 363-380, Grenoble.

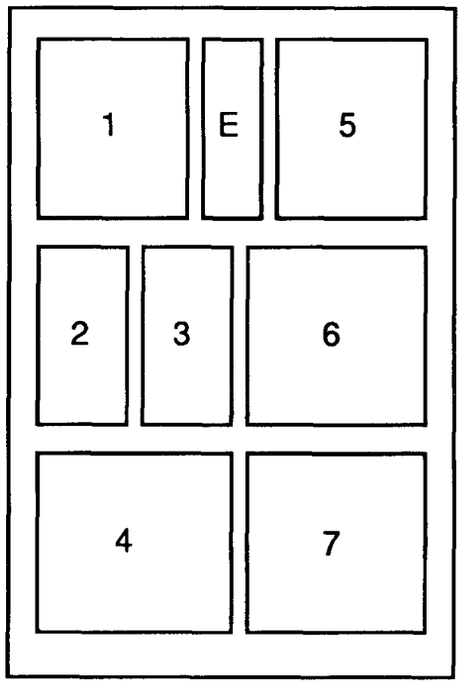
[19] Yannick Hervé *et al.*, Gabrielle Saucier *et al.*, *Étude et Réalisation d'une Machine massivement Parallèle pour le Traitement d'Images.* 1989 Journées d'architecture du PRC_AMN, Grenoble.

[20] Yannick Hervé, Yshu Wang, *ELSA in a Real Rime Image Processing Environment 1989 IFIP,* Parallel Architectures on Silicon, Grenoble.

Documentations techniques

Rapport ESPRIT 824B.
Rapport ESPRIT P26.
Documentation Technique GAPP, NCR Microelectronics.

Illustrations



L'échelle des couleurs (E) représente le nombre de particules par cellule, donc la pression.

Photos 1 à 4. — Simulation de la diffusion par automates cellulaires.

Photo 1. — Vue générale, après 1 000 itérations (40 sec).

Photo 2. — État initial, vue agrandie.

Photos 3 et 4. — État du système après 50 et 500 itérations. On peut remarquer les structures (« carrelages ») qui apparaissent à l'échelle microscopique (de la taille de 2*2 cellules).

Photos 5 à 7. — Simulation d'un écoulement par automates cellulaires.

Photo 5. — État du système au moment de la rencontre du fluide et des obstacles.

Photo 6. — État final, quasi stationnaire, qui s'établit après 500 itérations (20 sec.). On peut, ici, remarquer les « turbulences ».

Photo 7. — Agrandissement d'un des obstacles. Les couleurs indiquent une surpression sous l'intrados. Le vide qui s'établit sur l'extrados montre que le modèle du « fluide » ne correspond pas à celui d'un gaz ordinaire.

Manuscrit reçu le 9 avril 1990.