TREIZIÈME COLLOQUE GRETSI - JUAN-LES-PINS DU 16 AU 20 SEPTEMBRE 1991

EVALUATION DU MULTIPROCESSEUR SIMD/MIMD GFLOPS SUR DES ALGORITHMES DE TRAITEMENT D'IMAGE

Dominique HOUZET, Jean-Luc BASILLE,
Jean-Yves LATIL

IRIT : 118 route de Narbonne 31062 TOULOUSE CEDEX
Tél: 61-55-63-08 Fax: 61-55-62-58

Résumé. L'objectif de cette étude est l'évaluation sur des algorithmes de Traitement d'Image de l'architecture GFLOPS (General Flexible Linearly Organized Processing Structure) conçue et simulée au sein de notre équipe. Cette architecture monodimensionnelle est une architecture à mémoire distribuée fortement couplée pouvant fonctionner sous différents modes (essentiellement SIMD et MIMD) ce qui permet de traiter des familles d'algorithmes de niveaux différents, en particulier en Traitement d'Image.

1. INTRODUCTION

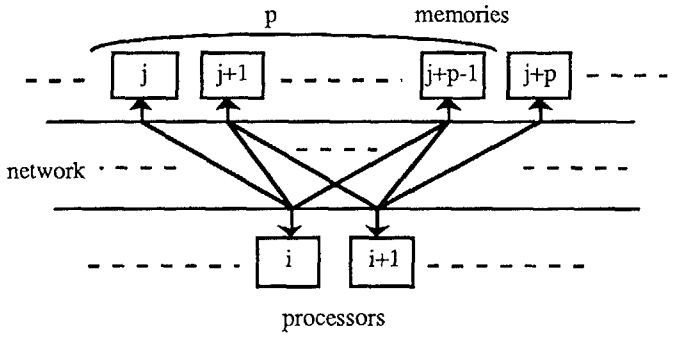
Le projet GFLOPS [HOUZ91] entre dans le cadre d'une étude plus générale sur les architectures de Traitement d'Image, menée au sein de l'équipe SYMPATI de l'IRIT. Cette étude a conduit à la réalisation du Processeur Ligne SYMPATI 2 développé en collaboration avec le CEA-Saclay [JUVI88].
L'objectif principal de cet article est de montrer que l'architecture GFLOPS permet de traiter efficacement une grande variété d'applications et particulièrement les algorithmes de traitement numérique des images par l'utilisation judicieuse des formes de parallélisme qui leur sont associées. Pour cela, l'architecture GFLOPS a été conçue afin de pouvoir fonctionner à la fois en mode SIMD qui est une forme de parallélisme synchrone, et en mode MIMD qui est une forme de parallélisme asynchrone. Dans cet article, nous nous intéressons plus précisément aux algorithmes de bas niveau de Traitement d'Image, bien adaptés au mode de programmation synchrone de par leur régularité, mais pouvant également tirer profit du mode de programmation de type asynchrone.
Nous avons pu évaluer cette architecture grâce à un simulateur associé à un assembleur qui nous ont permis de programmer et d'exécuter une trentaine d'algorithmes de Traitement d'Image. Les résultats obtenus font l'objet de la troisième partie de cet article.

2. L'ARCHITECTURE GFLOPS

Il s'agit d'une architecture à mémoire distribuée. Les images sont réparties sur l'ensemble des bancs mémoire. Les processeurs de GFLOPS sont reliés aux différents bancs mémoire au travers d'un réseau d'interconnexion monodimensionnel à un étage [HOUZ90] privilégiant les communications locales: celui-ci permet à chaque processeur d'être relié à un banc mémoire parmi un sous-ensemble local de bancs mémoire (figure 1). Nous avons choisi de faire communiquer les processeurs entre eux par l'intermédiaire des bancs mémoire.

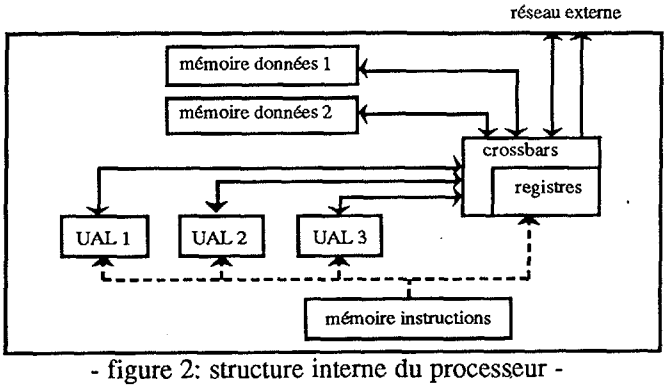
Abstract. We present in this paper the evaluation of GFLOPS (General Flexible Linearly Organized Parllel Structure) based on Image Processing algorithms. This architecture has been design and simulated in our team. It is a tightly coupled one-dimensional architecture using both synchronous and asynchronous forms of parallelism, that is SIMD and MIMD forms. Thus a great variety of algorithms may be implemented on this structure and more particularly in the Image Processing domaine

Un système de gestion des conflits d'accès mémoire est réparti dans le réseau pour permettre un fonctionnement en mode MIMD.



- figure 1: topologie du réseau -

La structure des processeurs [HOUZ91] est en accord avec la philosophie RISC (figure 2). De plus, pour profiter du parallélisme local, ceux-ci sont constitués d'un ensemble d'opérateurs élémentaires reliés à un ensemble de registres généraux par l'intermédiaire de plusieurs ports de communication. Cette structure se rapproche d'une structure de type VLIW [FISH83]. Les opérateurs sont des UALs banalisées pouvant aussi bien effectuer des calculs d'adresse que traiter des données.



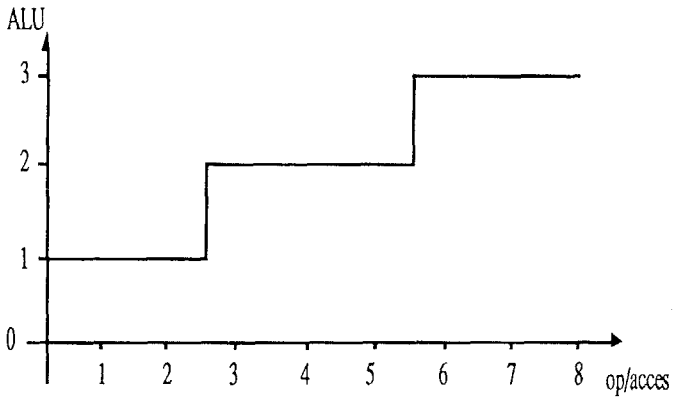
- figure 2: structure interne du processeur -

Une instruction est décomposée en deux cycles d'horloge. On peut effectuer en parallèle à chaque instruction, un accès mémoire, la recherche de l'instruction suivante et deux opérations par UAL (une opération par cycle d'horloge). En considérant une durée d'instruction de 100 ns on obtient pour une version de 128 processeurs, une puissance maximale de 5120 MOPS. Cette architecture peut être réalisée, en considérant des boitiers contenant chacun 4 processeurs (300 bornes pour 200000 Transistors), sur 4 cartes de type triple-europe.
Les processeurs fonctionnent de manière générale en mode MIMD. Ceux-ci peuvent se synchroniser par programme et fonctionner en mode synchrone (SIMD), le synchronisme étant maintenu par le programme. Ce mode de fonctionnement est purement implicite au niveau du code généré. En mode synchrone, le programme assure l'absence de conflits d'accès mémoire, ce qui permet d'optimiser les temps d'exécution. Grâce à ces deux modes de fonctionnement, il est possible de traiter efficacement des applications complètes de Traitement d'Image. La présentation des résultats d'évaluation de cette architecture fait l'objet de la suite de cet article.

3. EVALUATION DE GFLOPS

3.1 Nombre d'UALs par processeur

L'évaluation de notre structure s'est portée sur différents points spécifiques. Tout d'abord nous avons évalué l'intérêt de disposer de plusieurs UALs par processeur, en fonction de la quantité de traitement par rapport à la quantité de pixels traités. Cette évaluation s'est effectuée pour commencer sur des algorithmes de bas niveau de Traitement d'Image. Ce type d'évaluation a déjà été réalisé par différents auteurs [PLES88], mais les algorithmes utilisés sont généralement des algorithmes destinés à des machines "general purpose". Dans notre cas, nous avons simulé une trentaine d'algorithmes de bas niveau avec 2 types d'organisation des données en mémoire: une organisation de type tabulaire où chaque banc mémoire contient des colonnes de pixels de l'image, ou une organisation de type hélicoïdal où chaque banc mémoire contient des diagonales de pixels de l'image.
Ensuite, nous avons déterminé le nombre optimal théorique d'UALs nécessaires par processeur en fonction du taux de traitement (calcul d'adresse compris) par rapport au nombre d'accès mémoires nécessaires à un algorithme (voir figure 3). Cette évaluation tient compte du coût d'un processeur en fonction du nombre d'UALs.



- figure 3: nombre optimal d'UALs par processeur en fonction du nombre d'opérations par accès mémoire -

Par exemple, l'algorithme d'extraction d'un squelette après le calcul des distances aux frontières nécessite 18 opérations pour 5 accès mémoire Sachant qu'une UAL effectue deux opérations par instruction, on obtient les résultats de la figure 4.

	1 UAL	2 UALs
Coût en transistors	21000	29000
instructions par accès mémoire	1.78	1
amélioration	---	29 %

- figure 4: performances pour l'algorithme de squeletisation -

Dans ce cas, la puissance maximale est obtenue avec 2 UALs et le rendement maximal avec 2 UALs également.

3.2 Intérêt des mémoires locales

Nous avons également évalué l'intérêt de disposer de mémoires locales aux processeurs pour les algorithmes traitant un ensemble important de données localement. C'est le cas par exemple pour des calculs de formules mathématiques. Nous avons effectué cette analyse sur un algorithme de tri à bulles de N valeurs, pouvant être utilisé par exemple dans le calcul de la médiane du filtre de Tukey. On obtient les résultats de la figure 5, en considérant un coût par mémoire locale de 2000 transistors, et en effectuant un tri sur 25 valeurs. L'évaluation du coût a été obtenue à partir d'une estimation d'implantation sous forme d'ASIC. En intégrant la mémoire programme au processeur, on peut obtenir un circuit ne comportant pas plus de 50000 transistors et environ 200 bornes pour une version à 2 UALs. Ce type de circuit est réalisable en CMOS 1.2 µ à l'aide de SOLO1400 en vue d'une intégration par ES2. Nous avons commencé à étudier une telle réalisation.

	1 UAL 0 Mem.	1 UALs 1 Mem.	1 UALs 2 Mem.	2 UALs 0 Mem.	2 UALs 1 Mem.	2 UALs 2 Mem.	3 UALs 1 Mem.	3 UALs 2 Mem.
Coût en transistors	21000	23000	25000	29000	31000	33000	43000	45000
nombre d'instructions	715	390	390	480	270	270	270	155
amélioration par rapport au 1er cas	0 %	67 %	54 %	7 %	79 %	68 %	29 %	115 %

- figure 5: évaluation pour l'algorithme du tri à bulles -

Avec l'exemple de la figure 5, le nombre optimal d'UALs est 3, si on dispose de deux mémoires locales. L'amélioration a été définie de la manière suivante:

$$((C0*I0)/(Ci*Ii) - 1)/100$$

avec C le coût et I le nombre d'instructions. On peut remarquer que la puissance n'a pas augmenté lorsqu'on passe d'une mémoire locale à deux mémoires locales car les UALs sont déjà saturées pour une mémoire locale. De plus lorsqu'on passe de deux UALs à trois UALs en conservant une mémoire locale,

celle-ci étant utilisée au maximum, le fait d'ajouter une UAL n'apporte rien.
L'ensemble de ces simulations a été effectuée sur une structure ayant un chemin de données de 16 bits et avec un réseau ayant un voisinage d'accès de 7 bancs mémoire. Pour obtenir ces résultats, nous avons défini et simulé la structure la plus grosse contenant 3 UALs et 2 mémoires locales, et lors de sa programmation, nous n'avons utilisé que les parties nécessaires pour évaluer les différents cas.

3.3 Evaluation en mode synchrone

D'une manière générale, le fait d'avoir des instructions de même durée et une horloge commune permet à l'architecture de fonctionner de manière synchrone du point de vue du matériel. Lorsque nous parlerons de mode synchrone, il s'agira d'un mode de programmation assurant une exécution synchronisée du programme.
Nous avons évalué les performances de notre structure suivant son mode de fonctionnement. Nous avons commencé par évaluer le fonctionnement en mode synchrone: les programmes sont synchronisés à la génération du code de telle manière qu'un processeur ne puisse pas demander à accéder en même temps au même banc mémoire qu'un autre processeur. C'est un fonctionnement de type SIMD. Ainsi, les conflits mémoire sont évités et on conserve donc l'intérêt du mode SIMD. Dans un tel mode par exemple, les branches d'une structure conditionnelle peuvent s'effectuer en parallèle sur des processeurs différents, alors qu'en mode SIMD pur, celles-ci auraient dû être toutes effectuées, les unes après les autres. Avec les exemples que nous avons pris, la quantité de traitement étant faible par rapport à la quantité d'accès mémoire, nous avons obtenu un nombre optimal d'UALs de 2. Cependant, pour certains algorithmes comme la convolution 3*3 ou la DCT 8*8 utilisée en compression d'image, il est utile de disposer de plus de 2 UALs. Afin d'illustrer le fonctionnement du processeur, la figure 6 représente un exemple de programmation de la première passe de l'algorithme de dilatation 3*3 séparée. En utilisant des processeurs ayant 2 UALs, une instruction peut effectuer jusqu'à 4 opérations en parallèle. Chaque ligne décrit une instruction comportant plusieurs opérations effectuées en parallèle. Le registre d0 contient le numéro du banc mémoire accédé. Ce programme effectue pour chaque pixel le OU logique avec son voisin de gauche puis de droite et met le résultat dans une autre image qui commence à l'adresse D4. Les branchements conditionnels 'beq' et 'bne' (notations du 68000) s'effectuent à partir des résultats de l'intruction précédente.

```
      d2 <- m(d1)  d0 <- d0 - 1
      d2 <- m(d1)  d0 <- d0 + 2   d3 <- d2
L0:   d2 <- m(d1)  d3 <- d3 or d2  d0 <- d0 - 1  d1 <- d1 + 1
      d2 <- m(d1)  d3 <- d3 or d2  d0 <- d0 - 1
      d2 <- m(d1)  d1 <- d1 + d4   d3 <- d2   d1 - d5  beq L1
      m(d1) <- 1   d0 <- d0 + 2    d1 <- d1 - d4       bne L0
L1:   m(d1) <- 0   d0 <- d0 + 2    d1 <- d1 - d4       bne L0
```

d0 = banc mémoire demandé
d1 = adresse mémoire
d2 = donnée memoire
d3 = résultat
d4 = offset
d5 = nombre de pixel par banc mémoire
m( ) = accés mémoire

- figure 6: 1ère passe du programme de dilatation 3*3 séparée -

Nous avons effectué une série de simulations sur des algorithmes de Traitement d'Image de bas niveau. La figure 7 en présente certains résultats. Nous nous sommes intéressés au taux d'utilisation des ALUs et au taux d'utilisation des accès mémoire donc au taux d'utilisation du réseau externe. Ces résultats ont été obtenus à partir d'une version simulée de 16 processeurs ayant chacun 2 UALs et avec un temps de cycle instruction de 100ns soit 50ns par opération interne. Les images traitées sont de taille 256*256 pixels. En moyenne sur ces exemples, le fait de passer d'une UAL à 2 UALs a permis de diviser le temps d'execution par 1.5. Pour obtenir une amélioration identique, il aurait fallu augmenter de 50% le nombre de processeurs.
L'ensemble de ces algorithmes sont des algorithmes de bas niveau relativement simples et n'effectuant que peu d'opérations par pixel traité. Avec des algorithmes ou des structures de données plus complexes, comme par exemple avec la DCT 8*8 ou même la convolution 3*3, on atteint un taux d'utilisation de la seconde UAL de plus de 80%. Pour de tels algorithmes. Il peut donc être avantageux d'utiliser 3 UALs ou plus.
En ce qui concerne les échanges de données sur cette architecture, l'évaluation de l'algorithme de transposition de matrice révèle une capacité de communications globales performantes en mode synchrone. En effet, pour transposer une matrice, une colonne de pixels localisés dans un banc mémoire doit se répartir sur l'ensemble des bancs mémoire, pixel après pixel.

traitement	algorithmes	temps d'execution	UAL 1	UAL 2	accès mémoire
ponctuel	Tresholded	0.8 ms	87.5 %	25 %	100 %
	logical operation	1.2 ms	100 %	16.5 %	100 %
global	min / max	0.8 ms	75 %	0 %	50 %
	variance	1.3 ms	100 %	50 %	50 %
	histogram	1.1 ms	88.5 %	11.5 %	92.5 %
	image reduction	1 ms	86 %	46 %	86 %
	transposition	1 ms	100 %	70 %	77 %
local	skeleton	2 ms	100 %	60 %	100 %
	3*3 convolution	4 ms	100 %	85 %	90 %
	DRF edge detection	5.3 ms	100 %	37.5 %	75 %
	3*3 Prewitt	3.2 ms	87.5 %	44 %	87.5 %
	isolated points suppression	4 ms	50 %	11.5 %	61.5 %
	15*15 Huang median	33 ms	66.5 %	50 %	50 %
	DCT 8*8	5 ms	100 %	100 %	20 %
	split 3*3 dilation	2.4 ms	87.5 %	50 %	100 %
	boarder distances	3.6 ms	100 %	20 %	80 %
	moyenne		89.5 %	42 %	76 %

- figure 7: temps d'exécution et taux d'utilisation du processeur-

3.4 Evaluation en mode MIMD

Nous avons ensuite évalué le fonctionnement de notre structure en mode MIMD en utilisant le système de gestion des conflits mémoire associé au réseau. Il s'agit d'un mode de type asynchrone. Pour permettre le passage d'un mode à l'autre, il est nécessaire de se resynchroniser après une phase en mode MIMD: on utilise un indicateur dans chaque processeur signalant la fin de la phase MIMD. L'indicateur contenant le 'ou' des indicateurs précédents pour l'ensemble des processeurs, signale le début de l'exécution d'une phase synchrone (on utilise un 'ou' câblé). A titre d'exemple, nous avons comparé le fonctionnement de l'algorithme de suppression des points isolés d'une image binaire en mode SIMD et en mode MIMD. Nous avons obtenu les résultats indiqués sur la figure 8.

SIMD	MIMD	
	meilleur cas	moyenne
14	5	10

- figure 8: exemple de nombre d'instructions par pixel traité suivant le mode de fonctionnement -

La valeur moyenne de 10 instructions par pixel traité a été obtenue en utilisant des images contenant du bruit et ayant un nombre important de points isolés.
L'utilisation du mode MIMD nous permet d'améliorer les performances de notre structure même pour des algorithmes de bas niveau. Un exemple où cela est particulièrement vrai est l'algorithme de Huang pour le calcul du filtre médian [HUAN79]. C'est également le cas du Labelling en utilisant un graphe pour propager les étiquettes des région à labelliser.
L'intérêt du mode MIMD apparaît beaucoup plus clairement pour les algorithmes de haut niveau où les données à traiter représentent des caractéristiques de l'image issues des algorithmes de bas niveau. Ces données sont de nature irrégulière par définition et donc mal adaptées à un fonctionnement de type SIMD. Cependant, l'évaluation de ce mode de fonctionnement ne pourra se faire que sur des statistiques car les résultats dépendent fortement des données à traiter. De plus, les méthodes utilisées étant généralement différentes d'un mode à l'autre, il sera difficile de comparer les résultats. Nous avons commencé à simuler certains algorithmes comme le suivi de contours et le labelling sur des images particulières, en mode synchrone puis en mode MIMD. Nous avons également commencé à étudier certains algorithmes de manipulation non régulière d'image, utilisant des accès mémoire non réguliers comme pour la réduction d'image. En effectuant ces accès de manière synchrone, on obtient pour une réduction d'image 256 -> 128 un temps d'exécution d'environ 1 ms pour 16 processeurs.

4. CONCLUSION

Nous avons défini une architecture multiprocesseurs à mémoire distribuée constituée d'un réseau d'interconnexion permettant une extensibilité de manière linéaire de la structure grâce à son organisation sous forme modulaire. Ce réseau permet un accès direct de chaque processeur à un ensemble de bancs mémoire consécutifs. Cette structure peut fonctionner aussi bien en mode SIMD qu'en mode MIMD grâce au système de gestion des conflits d'accès du réseau. Les processeurs de notre structure sont des processeurs autonomes qui peuvent être synchronisés par programme. Ils sont constitués de plusieurs UALs banalisées. Ceci permet d'effectuer par exemple un calcul d'adresse hélicoïdale à chaque instruction, et d'optimiser le taux de traitements avec le taux d'accès mémoire. Nous avons simulé d'un point de vue fonctionnel l'ensemble de cette structure (processeurs-réseau) à la fois en mode synchrone et en mode MIMD. Nous avons pu déterminer certaines caractéristiques de performances grâce aux simulations effectuées, notamment en ce qui concerne le nombre d'UALs nécessaire par processeur et l'intérêt des mémoires locales. Nous avons commencé à évaluer cette structure sur des algorithmes de plus haut niveau (MIMD pur) pour pouvoir réaliser la simulation d'une chaîne complète de Traitement d'Image. Actuellement nous étudions l'implantation physique des processeurs. Nous pourrons entreprendre la réalisation matérielle de notre architecture dans le cadre d'un prototype contenant un faible nombre de processeurs, pour évaluer en situation réelle les performances d'une telle structure. Il nous faudra également étudier l'aspect logiciel de ce projet, c'est-à-dire définir un ensemble d'outils permettant de développer des applications réelles de Traitement d'Image.

5. BIBLIOGRAPHIE

[FISH83] J. A. Fisher, "Very Long Instruction Word Architectures and the ELI-512," 10th Symposium on Computer Architecture, IEEE Computer Society, 1983.

[FISK88] S. Fiske, J.W. Dally, "The Reconfigurable Arithmetic Processor," The 15th Annual International Symposium on Computer Architecture, Honolulu 1988, pp. 30-36.

[HOUZ90] D. Houzet, J.L. Basille, J.Y. Latil, "Réseau d'interconnexion incrémental pour une architecture de Traitement d'Image," 2nd Symposium Architectures Nouvelles de Machines, Toulouse 1990, pp. 219-242.

[HOUZ91] D. Houzet, J.L. Basille, J.Y. Latil, "GFLOPS: A General Flexible Linearly Organized Parallel Structure for Images," ASAP'91, Costa Brava, 2-4 sept. 1991.

[HOUZ91] D. Houzet, J.L. Basille, J.Y. Latil, "Processor Design for the Image Processing Parallel Architecture GFLOPS," ISMM Int. Workshop on Parallel Computing, Trani Italy, 1991.

[HUAN79] T.S. Huang, G.J. Yang, G.Y. Tang, "A fast two-dimensional median filtering algorithm," IEEE trans. Acoust., Speech and Signal Process., Vol. ASSP-27, N° 1, 1979, pp. 13-18.

[JUVI88] D. Juvin, J.L. Basille, H. Essafi, J.Y. Latil, "Sympati2, a 1.5D Processor array for image applications," in EUSIPCO Signal processing IV: Theories and applications, North-Holland, 1988, pp. 311-314.

[PLES88] A.R. Pleszkun, G.S. Sohi, "The Performance Potential of Multiple Functional Unit Processors," The 15th Annual International Symposium on Computer Architecture, Honolulu 1988, pp. 37-44.