Une Architecture Pyramidale à Base de Processeurs Programmables pour le Traitement des Images

Joël Le Roux et Diane Boucher
CNET-CCI, BP 98, 38243 Meylan Cedex

Résumé

Cette communication a pour but de présenter une étude en cours au centre de microélectronique du CNET à Grenoble et dont la finalité est la réalisation de deux circuits spécifiques permettant de construire une structure pyramidale. La base de cette pyramide est un plan de processeurs programmables soit de type classique soit de type processeurs de traitement du signal. L'étude comporte deux aspects : d'une part définition des caractéristiques des deux circuits envisagés, la mémoire à double accès et le noeuds du réseau de communication ; et d'autre part le développement d'un logiciel de simulation de fonctionnement de programmes sur cette structure parallèle permettant d'estimer les performances en vitesse de ce type d'architectures dans une application de traitement d'images (en particulier en synthèse et en détection de contours en vue de la reconnaissance des formes et de la transmission).

Abstract

The purpose of this communication is to present a study under development in the Centre National d'Etudes de Telecommunications in order to propose the realization of two specific circuits allowing the construction of a pyramidal structure. the basis of this pyramid is a plane of programmable processors (either classical or specialized in signal processing). There are two aspects in the study : firstly, the functional characteristic and the internal structure of the circuits are given ; secondly a software is developped for simulating the behaviour and evaluating the performances of image processing algorithms on such architectures (especially in image synthesis and contour detection for pattern recognition and image transmission).

Introduction

Si on veut réaliser une structure de calcul rapide relativement générale destinée au traitement des images (capable d'effectuer quelques milliards d'opérations par secondes), en écartant les structures directement liées à une application précise qui demandent souvent des solutions très spécifiques et très rigides comme le "pipe-line", on est amené à retenir un certain nombre de critères :

1 - La structure doit être parallèle et on doit pouvoir envisager un nombre variable de processeurs de calcul (il faut pouvoir considérer une structure comprenant plusieurs centaines de processeurs) ;
2 - L'information à traiter doit être répartie entre différentes mémoires ;
3 - Beaucoup d'applications supposent un traitement identique ("Single Program, Multiple Data") sur les différentes régions de l'image ; une forme raisonnable de parallélisme sera donc fondée sur une découpe spatiale de l'information, chaque mémoire contenant par exemple l'information représentant un carré de l'image à traiter ;
4 - Le processeur chargé de traiter une région de l'image doit pouvoir accéder préférentiellement à sa mémoire, mais aussi rapidement aux mémoires où se trouve l'information représentant les régions voisines ; il faut donc que le réseau de communication privilégie les communications locales (sans toutefois pénaliser excessivement les communications à plus grande distance) ;
5 - Si on veut dépasser un niveau élémentaire de fonctionnalités, il faut que les processeurs soient aisément programmabes en langage évolué ;
6 - Il faut que l'acquisition et la restitution des images par les mémoires de la structure se fasse sans que ces transferts ne freinent excessivement l'accès aléatoire normal par les processeurs de traitement ;
7 - Il faut qu'un contrôleur central ou un ordinateur hôte puisse échager aisément de l'information avec les processeurs du réseau (chargement de programmes, récupération d'information condensée en codage ou en reconnaissance des formes, description des objets en synthèse d'images, ...) ;

- De plus, avant d'entreprendre une réalisation ou une implantation d'un programme sophistiqué sur une structure de ce type, il faut pouvoir mettre en oeuvre une simulation. Celle-ci permettra d'une part de détecter d'éventuelles erreurs de programmation, ce qui n'est pas toujours simple dans une structure parallèle (effets de boucle, etc ...) et d'autre part de bien s'assurer que les temps de réponse d'un réseau d'une taille donnée sont bien adaptés au problème traité.

Une structure maintenent classique (voir les études citées en bibliographie) qui semble satisfaire correctement à cet ensemble de critères est celle des réseaux pyramidaux où un étage de numéro z contient un carré de $2^z x 2^z$ processeurs. Chaque processeur peut communiquer avec neuf voisins, un ascendant, quatre voisins dans le même plan et quatre descendants.

L'algorithme de traitement d'images est implanté dans les processeurs de la base de la pyramide. A chacun d'eux est associée une mémoire à double accès contenant l'information d'une région (un des 4^z carrés de l'image) ; soit un accès aléatoire normal par un bus et un accès direct vidéo permettant d'y écrire ou d'y lire une portion de ligne d'image (de longueur égale au côté du carré d'image mémorisé) sans que ces accès ne perturbent excessivement l'accès aléatoire.

Le réseau de communication permet aux processeurs d'envoyer (ou de recevoir) des messages vers les (des) mémoires, les autres processeurs ou le contrôleur situé au sommet de la pyramide. Les processeurs éventuellement associés aux noeuds des étages supérieurs de la pyramide seront plus spécialement chargés des traitements où l'information est condensée, sous une forme de plus en plus symbolique en fonction de l'étage considéré.
Remarquons néanmoins qu'une structure à bus unique peut s'envisager tant que le nombre de processeurs reste faible ; les limites d'une telle structure sont alors données par deux éléments : la durée nécessaire à l'arbitrage (~ 5 ns x nombre de processeurs sur la chaîne) et le temps d'accès à la mémoire (de l'ordre de 100 ns) ; dans ce cas, la répartition des données sur des mémoires locales ne présente guère d'intérêt si elles ne sont accessibles que par ce bus commun.

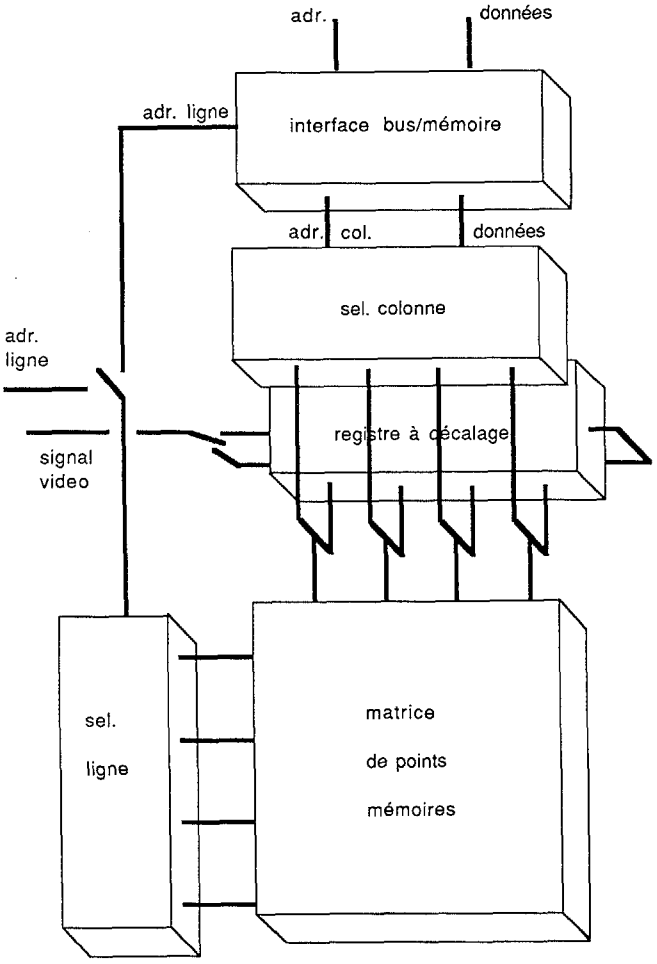
Notons aussi que les structures de communication fondées sur des réseaux de permutation possèdent des propriétés intéressantes et se prêtent à un bon nombre d'applications (cf. Lenfant, Auguin). Mais elles ne sont pas toujours les mieux adaptées au traitement d'image comprenant plusieurs niveaux d'algorithmes : elles ne privilégient pas les communications locales et permettent difficilement une spécification des

processeurs en fonction des tâches à effectuer. Ces hypothèses demandent à être confirmées ... Le logiciel de simulation proposé dans cette communication permettra de comparer objectivement les performances des différents types de structures parallèles envisagés actuellement (en ce qui concerne les architectures programmables spécialisées dans le traitement d'images en France, citons entre autres les réalisations et projets de Auguin et Al., Basille et Al., Bouville et Al., Merigot et Al, Vauquelin et Al.,).

En ce qui concerne les types de processeurs utilisés, notre étude est à rapprocher de celle de l'Université de Bordeaux où les processeurs de traitement sont des processeurs programmables classiques ; elle se différencie plus des études qui (comme à Orsay) préconisent la mise au point de processeurs spécifiques. Il nous semble en effet que la première approche permet d'utiliser un immense savoir-faire (langages, exploitation, normes et programmes existants) et de se poser avant tout les problèmes de parallélisation d'algorithmes et de méthodologie d'implantation, alors que dans la seconde approche tous les problèmes sont repris à la base.

Notons aussi que le Transputer d'Inmos permet d'envisager de telles structures et qu'il bénéficie d'un environnement logiciel intéressant (langage Occam). Son utilisation présente actuellement deux inconvénients : le nombre d'entrées sorties est limité à quatre, et le processeur de traitement est aussi chargé de la gestion des communication (réaiguillage des messages entre autres tâches).

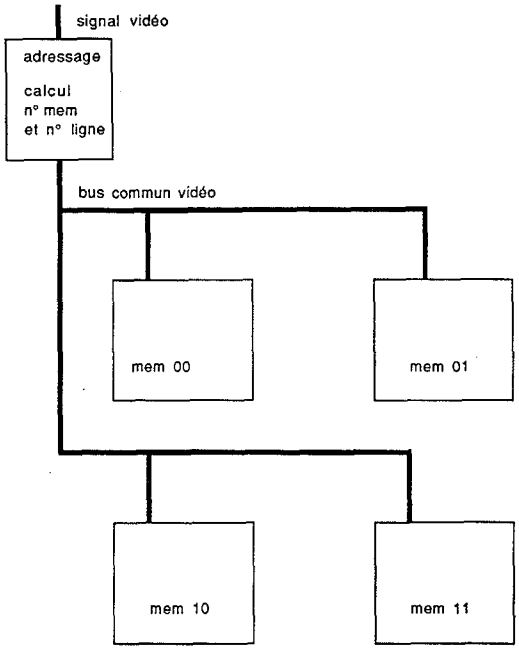
Dans cette communication nous présenterons les deux types de circuits dont nous envisageons la réalisation : la mémoire à double accès et le noeud du réseau pyramidal de communication ainsi que le logiciel de simulation et d'évaluation des performance. Ce point de notre étude nous parait essentiel : avnat d'implanter un algorithme sur une telle structure, il faut s'assurer de son bon fonctionnement en simulation et connaitre les durées d'exécution des programmes sur une telle structure.



Mémoire Double Accès Aléatoire et Vidéo

La mémoire à double accès

Pour mémoriser les portions d'images, nous envisageons une structure de mémoire à double accès maintenant classique et telle que le nombre de colonnes de leur matrice de bits soit égal à la taille du côté du carré d'image mémorisé. On accède à la mémoire soit par l'accès bus normal par transmission usuelle d'adresses et de données, soit par l'accès vidéo : sur le bus video, l'adresse de la mémoire associée à une région est affichée par un contrôleur ainsi qu'un numéro de ligne dans cette mémoire. En mode écriture, une séquence de valeurs dont la longueur correspond à la taille du côté de la région est transmise sur le bus video et emmagazinée dans un registre à décalage série/parallèle. Lorsque le registre est plein, les données présentes dans le registre à décalage sont transmises en parallèle dans la ligne sélectionnée de la mémoire à double accès. En lecture, l'information est transmise en parallèle de la mémoire vers les entrées parallèles du registre à décalage série/parallèle puis cette information est transmise sur le bus vidéo. Dans une telle mémoire, seul le millième du temps d'accès est utilisé par la vidéo qui ne perturbe donc guère l'accès par les processeurs de traitement.



répartition du signal vidéo
sur les différentes mémoires

Le noeud du réseau de communication

Ce circuit a pour fonction de récupérer un message (constitué d'une adresse d'expéditeur, d'une adresse de destinataire, de l'adresse d'un mot ou d'une suite de mots mémoires, d'une ou plusieurs valeurs de donnée et d'un ordre d'écriture) présent sur une de ses entrées et de l'aiguiller vers une de ses sorties en fonction de l'adresse du destinataire final, rapprochant ainsi le message de ce destinataire.

Il peut accéder à neuf voisins identiques à lui-même et éventuellement à un processeur de calcul local ainsi qu'à une mémoire. Il est composé de registres où sont mémorisés les messages à réexpédier, d'un bus interne auquel sont connectés ces registres, et d'une unité de contrôle (dispatcher) permettant de gérer les communications sur ce bus et de réorienter les messages en fonction de l'adresse de leur destinataire.

Le dispatcher scrute systématiquement l'état des registres d'entrée. Lorsqu'un de ces registres contient un message, le dispatcher récupère l'adresse du destinataire et déduit d'une stratégie très simple le numéro de la sortie par laquelle sera réexpédié le message. Si un message est en attente et si la sortie par laquelle il sera réexpédié est connue, alors le dispatcher demande l'accès à cette sortie (c'est à dire au registre d'entrée du noeud aval). Si cette sortie est occupée, le message reste en attente jusqu'à la scrutation suivante, sinon le message est transféré du registre d'entrée vers le noeud aval, ce qui libère le registre où était précedemment stocké le message.

Pour que les communications locales soient favorisées, la liaison entre le processeur de traitement et sa mémoire locale ne passe pas par le noeud de communication ; elle se fait par l'intermédiaire d'un bus local. La liaison entre le noeud et le bus local se fait par deux interfaces : d'une part un registre permettant de mémoriser les messages à destination ou en provenance du processeur ; d'autre part un contrôleur de mémoire capable de lire et d'écrire une donnée dans la mémoire, de reconstituer un message à partir d'une donnée lue en mémoire et de réexpédier ce nouveau message au processeur ayant demandé l'information.

La stratégie actuellement dans le calcul du numéro de porte de sortie permet de minimiser le nombre de noeuds que doit traverser un message :

Soit (Dx, Dy, Dz) la différence d'adresse entre le noeud où se trouve le message et le destinataire et D_zx, D_zy la différence d'adresse entre le noeud courant et le noeud ascendant du destinataire situé dans le même plan que ce noeud courant.

Si Dx = Dy = Dz = 0, le destinataire est soit la mémoire locale soit le processeur du noeud considéré ;

Si Dz < 0, le message est transmis à l'ascendant ;

Si Dz > 0 et $D_zx = D_zy = 0$, le message est transmis à celui des quatre descendants qui est un ascendant du destinataire ;

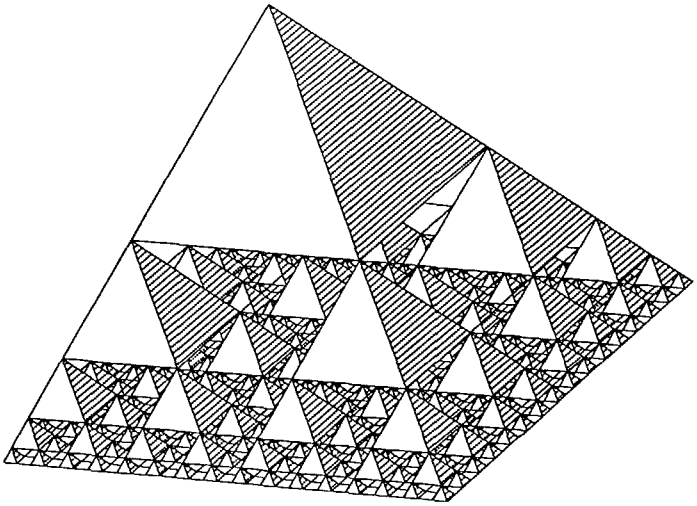
Si $|D_zx| + |D_zy| < 4$, le message est transmis dans le plan horizontal à celui des quatre voisins qui est le plus proche de l'ascendant du destinataire dans ce plan ; sinon le message est transmis à l'ascendant.

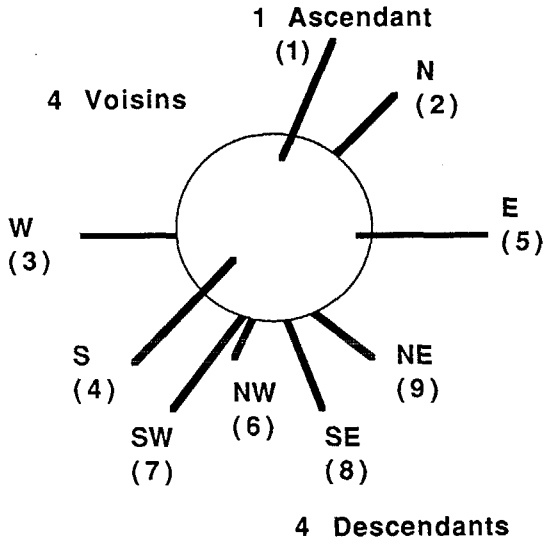
Remarque : la définition de ces deux circuits (mémoire double accès et noeud du réseau) se fait au moyen d'un langage de description fonctionnelle (FIDEL) développé au CNET (cf. El Tahawy) qui permet en particulier de comparer les performances temporelles de plusieurs variantes possibles du noeud du réseau.

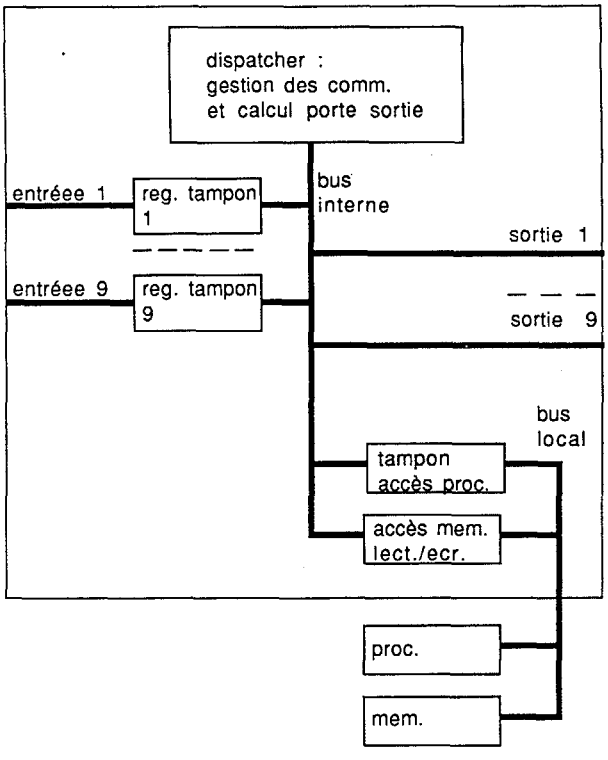
L'outil de simulation et d'évaluation des performances

L'implantation d'un algorithme sur une structure parallèle spécifique présuppose une utilisation répétitive de cet algorithme. La mise au point directe d'un programme relativement complexe sur une structure parallèle est délicate. Pour ces raisons, il nous semble bon de commencer une implantation de programme sur une structure simulée avant de le transporter sur la structure parallèle elle-même. De plus la structure simulée peut aisément prendre en compte un certain nombre de paramètres : nombre de processeurs, taille des mémoires, forme du réseau de communication, temps d'éxécution des instruction, ... L'utilisateur peut ainsi évaluer le temps d'éxécution d'un cycle du programme et choisir la forme de la structure finale avant l'implantation elle-même.

L'outil nécessaire à cette simulation se situe à un niveau supérieur à celui des langages de description de matériel. Il doit permettre de définir les différent organes séquenciels de la structure comme autant de processus fonctionnant en parallèle ; ces processus apparaitront comme des programmes écrits en langage évolué capables d'échanger des messages et de prendre en compte le temps. Il doit donc comporter une base de temps car le résultat qu'on attend de lui est une estimation précise de la durée d'éxécution d'un







noeud du réseau de communication

algorithme ; cette base de temps est aussi nécessaire à la synchronisation réaliste des processus parallèles. Il doit comporter une méthode de description connections des processus entre eux. Au delà du mécanisme d'échange de messages, il doit aussi faire intervenir un certain nombre de règles permettant une mise en place cohérente de ces échanges.

Le langage ADA nous a semblé un outil intéressant pour cet outil de simulation : les tâches permettent de bien simuler le parallélisme et l'échange d'information entre processus. La définition d'une tâche supplémentaire associée au temps permet de concevoir aisément une base de temps. L'adjonction de cette base de temps et le respect de quelques règles simples permettent ainsi de simuler une structure complète, bien que la lourdeur de la gestion des tâches (durée et taille mémoire) soit actuellement une limitation sérieuse du nombre de processus travaillant en parallèle.

Nous développons actuellement une seconde version du programme où seule est conservée la tâche associée au temps : les processus sont associés à des procédures classiques expédiant des messages à destination de la boite à lettre du destinataire dans une mémoire commune ou lisant dans cette boite à lettre les messages qui leur sont destinés. La synchronisation se fait par l'intermédiaire de la base de temps : à chaque instruction ou séquence d'instructions sans échange avec l'extérieur est associée une durée d'attente et donc un appel de la tâche temps qui lui donnera l'ordre de redémarrer le moment voulu. Une gestion correcte de la mémoire commune suppose qu'on évite de lire les boites à lettre ne contenant pas de message. Une bonne synchronisation suppose une différenciation nette des instants d'écriture (impairs) et des instants de lecture (pairs) des messages. Cette règle permet de contourner de délicats problèmes de priorité entre processus demandant d'accéder à l'unité centrale en même temps.

Conclusion

Nous avons présenté une étude sur une architecture multiprocesseur où nous insistons particulièrement sur l'importance des simulations, qui doivent précéder toute implantation. Ces simulations sont menées à deux niveaux : un niveau global où sont analysées les performances de la structure dans son ensemble grâce à un logiciel spécifique et un second niveau (analyse fonctionnelle classique) pour la définition précise des circuits à concevoir.

Bibliographie

Etudes en cours sur les structures pyramidales en traitement des images

En France :

M. Merigot, F. Devos (Orsay), "Projet de système pyramidal hiérarchisé pour le traitement numérique des images (SPHINX)", à l'institut d'éléctronique fondamentale, Université Paris XI.

B. Vauquelin (Bordeaux), "Projet de calculateur multiprocesseur à architecture pyramidale et sans mémoire commune (CHEOPS)", UER de mathématique et d'informatique, Université de Bordeaux I.

En Europe :

V. Cantoni, M. Ferretti, S. Levialdi and F. Maloberti (Rome), "A pyramid project using integrated technology", in Integrated Technology for parallel Image Processing, (S. Levialdi, Ed), 1985, pp 121-132

T.J. Fountain (Londres), "Array Architectures for iconic and symbolic image processing", Int. Conf. on Pattern recognition, Paris, Nov. 1986, pp 24-33.

Aux U.S.A :

C.R.Dyer (Chicago), "Pyramid algorithms and machines", in Multicomputers and image processing, algorithms and programs, Preston and Uhr, Eds, Academic Press, 1982.

S.L. Tanimoto (Seattle), "A pyramidal approach to parallel processing", Proc. 10th Annual Int. Symposium on Computer Architecture, Stockholm, June 1983, pp 372-378.

L. Uhr (Madison, Winsconsin), "Pyramid multi-computers and extensions and augmentation", in algorithmically specialized parallel computers (L. Snyder et al. Eds), Academic Press, 1985, pp 177-186.

Autres éléments bibliographiques

D.P. Agrawal and G.C. Pathak, "Design of VLSI based multicomputer architecture for dynamic scene analysis", in VLSI for pattern recognition and image processing, K.S. Fu, ed, Springer-Verlag, Berlin, 1984.

M. Auguin, F. Boeri (Nice), " Presentation des travaux du LASSY dans le domaine des archictectures paralleles", Séminaire sur les supercalculateurs, Paris, Février 1987.

J.L. Basille, S. Castan and J.Y. Latil (Toulouse), "Système multiprocesseur adapté au traitement des images" Workshop on new computer architectures and image processing, Ischia, Italie, 1980.

C. Bouville, R. Brusq, J.L. Dubois and I. Marchal (Rennes), "Generating high quality pictures by ray tracing", computer grapics forum, n° 4, 1985, pp 87-99.

J. Lenfant (Rennes), "Parallel permutation of data, a Benes network control algorithm for frequently used permutations", IEEE trans on computers, vol. 27, 1979, pp 637-647.

S.P Levitan, C.C. Weems and E.M. Riseman, "Signal to symbols : unblocking the vision communication/control bottleneck", in VLSI Signal Processing, P.R. Capello et al. eds, IEEE press, 1984.

P.A. Nagin, A.R. Hanson and E.M. Riseman, "Region relaxation in a parallel hierarchical architecture", in Real-time parallel computing : image analysis, M. Onoe K. Preston and A. Rosenfeld, Eds, Plenum press, New York, 1981.

P.C. Patton, "Multiprocessors : architecture and applications", Computer, June 1985.

K. Preston, "Languages for parallel processing of images", in Real-time parallel computing : image analysis, M. Onoe K. Preston and A. Rosenfeld, Eds, Plenum press, New York, 1981.

A. Rosenfeld, "Parallel image processing using cellular arrays", Computer, January 1983.

Sur la simulation fonctionnelle et en particulier l'utilisation du langage Ada

M.R. Barbacci et Al., "Ada as an hardware description language, an initial report", rapport CMU-CS-85-104, Dec. 1984.

H. El Tahawy et Al., "FIDEL, a multilevel hardware description and simulation language", Proc of the Int. Circ. Tech. Conf., Limeric, Ireland, Sept. 1986.

E.F. Girczic, R.J.A. Buhr and J.P. Knight, "Applicability of a subset of Ada as an algorithmic hardware description language for graph-based hardware compilation", IEEE Trans. on Computer aided design, vol CAD-4, n° 2, April 1985, pp 134-142.