NOUVELLE ARCHITECTURE APPLIQUEE A LA PARALLELISATION DE DONNEES SEQUENTIELLES POUR LE TRAITEMENT D'IMAGE

B. TENEZE, P. BAYLOU, M. NAJIM

Equipe Signal et Image, ENSERB, 351 cours de la Libération, 33400 TALENCE CEDEX
GRECO-TDSI

Resumé :
Nous présentons dans ce papier une nouvelle architecture adaptée aux algorithmes récursifs, construite sur le principe d'une mise en correspondance entre architecture et algorithme. Ainsi dans le cadre du traitement d'image, ce principe nous a permis d'atteindre, par une parallélisation de données traitées par un réseau multitransputers, des performances élevées pour des traitements temps réel.

Abstract :
This paper deals with a new architecture built according to the architecture/algorithm interaction principle, giving an efficient implementation of recursive algorithm. Furthermore, in image processing, this principle allows, with a parallelisation of the data processed by a multitransputers network, high performance for real time application.

INTRODUCTION

L'objectif de temps réel en traitement du signal et de l'image amène à rechercher des solutions qui permettent de diminuer le temps d'exécution des algorithmes (1,2). La conception et la mise en correspondance de l'algorithme sur une architecture appropriée semble être une alternative permettant d'atteindre cet objectif. En effet, tous les algorithmes peuvent être implantés sur des ordinateurs monoprocesseurs de plus en plus puissants, mais au détriment d'une complexité technologique plus grande, et d'un prix de revient plus élevé. Cependant, la réécriture des algorithmes sous forme parallèle (3,4) ou sous forme adaptée au traitement en pipeline (5) autorise leur implantation sur des nouvelles structures matérielles très performantes, telles que les réseaux parallèles multiprocesseurs (T-Nodes, Connexion Machine,...), ou sur des structures de calcul en pipeline.

Afin de concrétiser un projet de traitement d'image temps réel, nous avons exploité les avantages de ce principe d'interaction entre l'algorithme et l'architecture. C'est ainsi que nous avons conçu de façon synchronisée d'une part une architecture matérielle dérivée de la machine de Türing (6) et de la machine de Von Neumann (7), et d'autre part des algorithmes récursifs pouvant être implantés sur cette architecture. Les algorithmes récursifs sont considérés comme une grande classe d'applications pour les signaux 1D et 2D. Dans ce dernier cas, le signal vidéo peut être vu comme un flux séquentiel de données multiplexées pour lequel les processus de traitement peuvent être aisément traduits sous forme d'algorithmes récursifs.

Nous exposons dans la première partie de ce papier un ensemble de remarques à partir desquelles nous dégageons les principes de conception de notre architecture. Dans la deuxième partie, nous précisons les choix matériels et algorithmiques découlant de ces remarques. Dans la troisième partie, nous décrivons notre architecture et l'algorithme récursif associé. Enfin nous décrivons dans la quatrième partie, une application concrète qui permet d'illustrer notre approche.

I REMARQUES PRELIMINAIRES

I.1 Opération élémentaire de Von Neumann

Notre développement s'est appuyé sur deux modèles d'architecture : celui de Türing (1936) (6) ou machine à bandes de longueur infinie contrôlée par un automate à états finis et celui de Von Neumann (7) ou machine à registres. La conception de cette dernière a été guidée par la recherche de l'universalité de son application et par la limitation de la quantité de mémoire disponible. Ceci implique un accés de type aléatoire aux informations (RAM) nous permettant alors de faire la première constatation suivante.

Une opération exécutée sur un système de Von Neumann, se déroule selon les trois étapes suivantes : la lecture de l'opérande sur le bus de données, l'accomplissement de l'opération, et enfin l'écriture du résultat dans la mémoire. Ces trois opérations sont effectuées séquentiellement mais durent cependant chacune le même temps.

La majeure partie du temps pris par les opérations de lecture et d'écriture est due à la nécessité d'adresser aléatoirement des emplacements mémoires, c'est à dire sans que l'on puisse déduire l'adresse suivante à partir des adresses déjà utilisées.

I.2 Données de base :

Les données sérielles à traiter peuvent être transmises multiplexées ou non. Nous nous sommes

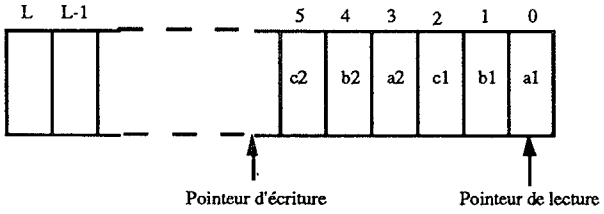
plus particulièrement intéressés au cas du transfert multiplexé qui est celui de la transmission sérielle de lignes constituant une image. Cependant, son étude englobe de plus celle du cas non multiplexé.

Dans le cas le plus courant où le même processus de traitement doit être appliqué à chaque ensemble de données multiplexées, le système matériel doit être à même d'effectuer une commutation de contexte la plus rapide possible et ne prenant qu'un minimum de temps à l'unité de calcul.

II STRUCTURES DE DONNEES ET ALGORITHMIQUES

II.2 Structure de données

L'emploi d'une file d'attente comme structure de données permet de s'affranchir du problème de l'accés aléatoire aux données. En effet, chaque écriture et chaque lecture se produit systématiquement à l'emplacement mémoire suivant directement celui en cours. De plus, les écritures et les lectures peuvent être asynchrones. Enfin, ces files réalisent implicitement une structure trés adaptée aux commutations de contexte. En effet, supposons que le traitement de chacune des parties des ensembles E1 et E2 de données à transmettre nécessite trois variables systèmes a_i,b_i et c_i, où i est le numéro de l'ensemble de données, le contenu d'une file, lors du deuxième changement de contexte, sera:



On remarque alors que le pointeur de lecture est automatiquement positionné sur la première donnée de l'ensemble E1 dont on avait abandonné le traitement deux commutations de contexte plus tôt.

II.2 Types de données à traiter

La restriction des données à traiter à celles transportées par un flot synchrone continu sériel permet également de s'affranchir de l'accès aléatoire à l'information.

Si le système de traitement fonctionne en temps réel, alors il est prêt à effectuer la nouvelle opération au prochain cycle d'horloge de synchronisation. La nouvelle donnée se présente alors sur le "buffer" d'entrée du système, remplaçant la précédente, et permettant ainsi que sa lecture se produise au même emplacement mémoire que la précédente.

II.3 Algorithme récursif

Un algorithme de type récursif a la particularité de bâtir des quantités dont les valeurs prises successivement au cours de son exécution mémorisent de façon implicite les valeurs des données à traiter déjà reçues.

Comme notre architecture est destinée au traitement d'un flot continu de données qu'il n'est évidemment pas possible de mémoriser exhaustivement, l'accés à ces quelques quantités évitera une perte de temps et un coût important dans la mémorisation des données à traiter.

Ces quantités seront stockées dans des files d'attente.

III ARCHITECTURE

L'application des trois principes du paragraphe précédent nous amène à la conception d'une nouvelle architecture, qui est un compromis entre la machine de Türing et celle de Von Neumann.

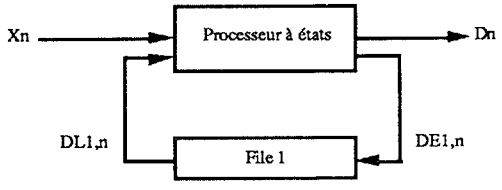
Le cœur de ce système est constituée de N unités arithmétiques et logiques (ALU) et d'un automate à R états. Nous l'appellerons dans la suite processeur à états. Le décodage de ces états génère les signaux d'écriture ou de lecture des files d'attente séparément les unes des autres. La commutation d'un état vers un autre est fonction à la fois des signaux d'entrée et de données lues sur certaines files.

Le fonctionnement de l'ensemble de cette architecture est synchronisé par l'horloge cadençant les arrivées des données de flot continu sériel. Nous associerons à chacune des variables de notre algorithme un indice n, précisant l'instant d'échantillonnage actuel. Le changement des états de l'automate sera également synchronisé par cette même horloge.

Adoptons tout d'abord les formes d'écriture suivantes :

X_n la valeur de l'échantillon d'entrée à l'instant n
$DE_{i,n}$ la valeur inscrite dans la file i à l'instant n
$DL_{i,n}$ la valeur lue dans la file i à l'instant n
O_n l'opération effectuée à l'instant n

Etudions le cas simple où l'opération effectuée On est toujours la même quelque soit l'instant considéré n. Cette application typique est représentée par le schéma synoptique suivant :



Architecture à file et opération unique

Dans ce cas très simple, les données $D_{1,n}$ sont identiques aux données D_n, sorties de notre système. De plus, la longueur de la file est ici constamment égale à 1. Nous nous apercevons alors que l'équation typique implantée par cette structure est la formule récursive suivante:

$$D_n = O (X_n , DL_{1,n})$$

associée à l'équation système

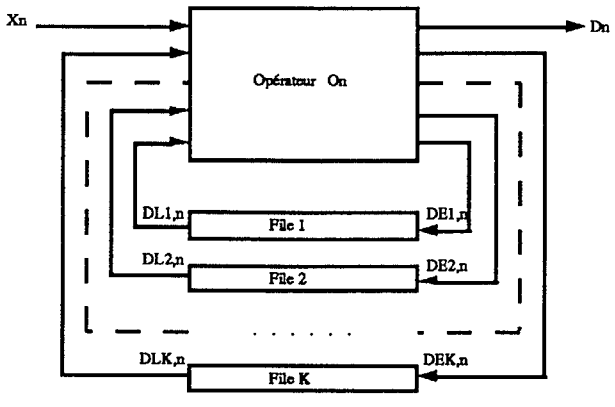
$$DE_{1,n} = D_n$$

Ces équations ainsi implantées peuvent être assimilées, par exemple, à l'équation récursive :

$$D_n = F(X_n) + D_{n-1}$$

Ces équations montrent que les données DL et DE circulant dans les files d'attente, sont des données composites mémorisant le passé de l'ensemble des opérations effectuée depuis l'instant origine jusqu'à l'instant n. Ces variables contiennent intrinsèquement une trace des données utilisées pour leur obtention, sans toutefois nécessiter le stockage exhaustif de l'ensemble de ces données passées.

Ce synoptique de base peut être étendu au cas dans lequel plusieurs ALUs cohabitent dans le processeur à états et où plusieurs files d'attente sont capables de fonctionner séparément les unes des autres. Ainsi, plusieurs données composites différentes peuvent être créées simultanément par des opérations arithmétiques et logiques disctinctes. L'automate à R états est alors chargé d'actionner soit en écriture soit en lecture une ou plusieurs files d'attente. Le schéma synoptique représentant ce principe devient alors le suivant:



Architecture dans le cas général

Ce schéma synoptique traduit l'équation récursive suivante :

$$D_n = O_n (X_n , DL_{1,n} , DL_{2,n} , ... , DL_{K,n})$$

associée aux équations systèmes suivantes:

$$DE_{1,n} = O_n (X_n , DL_{1,n} , DL_{2,n} , ... , DL_{K,n})$$
$$DE_{2,n} = O_n (X_n , DL_{1,n} , DL_{2,n} , ... , DL_{K,n})$$
$$...$$
$$DE_{K,n} = O_n (X_n , DL_{1,n} , DL_{2,n} , ... , DL_{K,n})$$

Toute opération O_n est accompagnée d'une ou de plusieurs lectures ou écritures d'une ou de plusieurs files. C'est ainsi qu'à chaque pas de déroulement de cet algorithme récursif, une à K lectures sont effectuées ainsi que une à K écritures dans les files, d'où l'utilité de ces dernières, qui "absorbent" ainsi les différences de rythme d'écriture et de lecture.

Les valeurs inscrites dans les files d'attente sont disponibles pour les ALUs à n'importe quel instant n+k (à condition que la file ne soit pas saturée pendant l'intervalle k). Cette particularité est mise à profit pour la mise en œuvre des algorithmes travaillant sur des données multiplexées.

Le schéma bloc de l'automate et de son décodage est représenté sur la figure suivante:

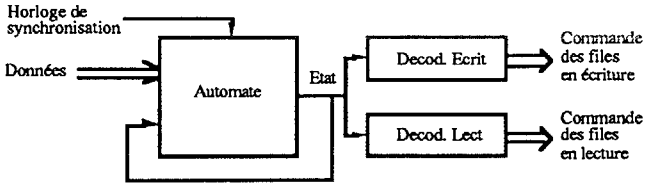
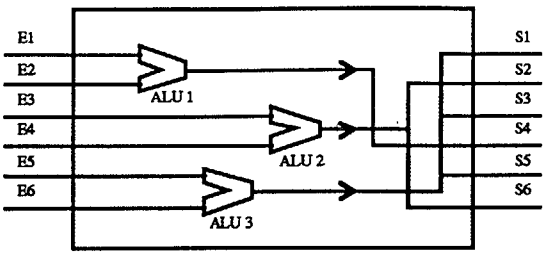


Schéma bloc de l'automate

Le fait que les données de base que nous traitons soient synchronisées par une horloge impose que les commutations d'états déclenchées par des conditions booléennes C_i soient également synchrones. Les conditions Ci sont déterminées par des combinaisons logiques des données du flux Xn et de celles de certaines files d'attente. La programmation de l'automate se résume donc à l'établissement de ces équations logiques.

Un exemple d'ensemble des ALUs présentes dans le processeur à états, est représenté sur le synoptique suivant:



Exemple du bloc d'unités de traitement logique et arithmétique

Chacune de ces ALUs est dotée de deux entrées (Ei), provenant de files d'attente ou des données Xn. Le cas d'ALUs à plus de deux entrées peut être envisagé si leur implantation est réalisée en logique

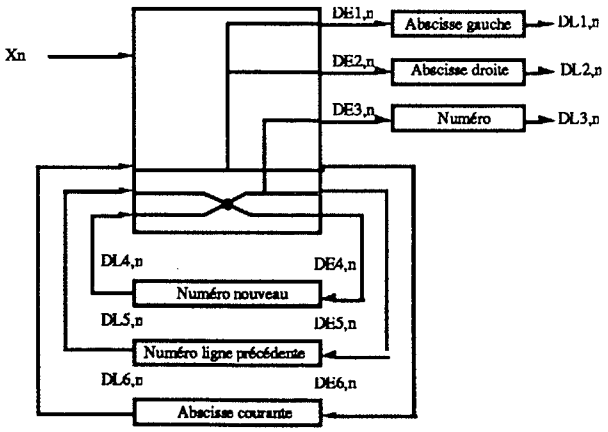
cablée. Leur sortie (S) est reliée à une ou plusieurs entrées de file d'attente. Il faut toujours avoir présent à l'esprit que l'automate, en actionnant la lecture des files d'attente procure des données nouvelles aux ALUs, et, en autorisant l'écriture dans ces mêmes files, prend en compte ou non les sorties de ces ALUs.

Remarque : une sortie de files peut être reliée directement à une ou plusieurs entrées de file d'attente, sans pour cela subir d'opération logique ou arithmétique.

Partie IV : application

Notre architecture a été utilisée dans le cadre d'une reconnaissance de forme en permettant une segmentation séquentielle écrite sous forme itérative des objets. Cette opération consiste à détecter les points de contour des objets, tout en les étiquetant avec un numéro affecté automatiquement à chaque objet. Le résultat est ainsi d'obtenir une parallélisation par les données permettant le traitement haut niveau par une structure de type SIMD.

Plus précisèment, l'architecture permet de détecter à la fréquence pixel le contour des objets d'une image binarisée apportant ainsi son résultat sans retard par rapport à l'acquisition de l'image. Le codage du contour des objets consiste, à stocker dans trois files le numéro du premier (file 1) et du dernier pixel (file 2) appartenant au contour d'un objet numéroté (file 3) intercepté le balayage séquentiel horizontale de l'image.



Architecture appliquée au traitement d'image

La file 4 contient une liste de chiffres arbitrairement choisis entre 0 et C, où C représente le nombre maximum d'objets pouvant être détectés simultanément. La file 5 contient les mêmes valeurs que la file numéro 3. La file 6 contient les chiffres 1 à N où N est le nombre de pixels par ligne.

Dans cette application, aucune ALU n'est employée.

Les files 1, 2 et 3 sont lues par un transputer maître servant d'aiguillage en émettant les abscisses des points de contour sur un transputer esclave effectuant la reconnaissance de forme, suivant le numéro de l'objet.

Ce procédé a débouché sur la fabrication d'une carte d'extension pour PC. Cette carte connectée à une caméra linéaire de 2048 pixels transmis à 10MHz permet d'obtenir l'analyse par la forme de plus de 200 objets par seconde, alors que la plupart des systèmes disponibles sur le marché sont limités actuellement à une dizaine d'objets par seconde (8).

Références :

(1) : S.W. WALLACE, Algorithms and Model Formulations in Mathematical Programming, Springer Verlag 1987.

(2) : J.F. TRAUB, A general theory of optimal algorithms, Academic Press 1980.

(3) : J.F TRAUB, Complexity of sequential and parallel numerical algorithms, Academic Press 1973.

(4) : M.V. ZILLY, Mathematical models for the semantics of parallelism, Springer Verlag, 1986.

(5) : P. DEWILDE : algorithmes et architectures, conférence prononcée à Bordeaux le 5/5/1991

(6) : J.BOND, Cours de calculabilité , Université de Bx.I, 1989.

(7) : A.W BURKS and J.V. NEUMANN, Preliminary discussion of the logical design of an electronic computing instrument, The institute for advanced study 1946.

(8) : B. TENEZE, thèse à soutenir, Université de Bordeaux I, 1991.