

Des langages fonctionnels au traitement d'image temps réel: Un environnement de programmation pour une architecture flot de données

Jocelyn SÉROT, Georges QUÉNOT, Bertrand ZAVIDOVIQUE

Laboratoire Système de Perception
DGA/Etablissement Technique Central de l'Armement
16 bis Av. Prieur de la Côte d'Or, 94114 Arcueil Cedex
Tel: (33 1) 42 31 93 48 Fax: (33 1) 42 31 99 64 Email: jserot@etca.fr

Résumé

Le Calculateur Fonctionnel est une machine flot de données massivement parallèle dédiée principalement au traitement d'image temps réel. L'architecture matérielle comprend 1024 processeurs bas niveau dans un réseau tridimensionnel 16x8x8 et jusqu' à 36 transputers, pour une puissance crête de 50 milliards d'opérations par seconde. L'environnement de programmation est basé sur une étroite intégration du modèle d'exécution Flot de Données et des concepts issus de la Programmation Fonctionnelle. Il comprend une chaîne de compilation complète, produisant une configuration exécutable à partir d'une spécification fonctionnelle haut niveau, et d'une bibliothèque de primitives d'opérateurs. Plusieurs applications réalistes ont déjà été implantées et exécutées en temps réel sur des séquences vidéo à 25 Hz.

1 Introduction

L'intérêt de disposer d'une plate-forme de prototypage d'applications de vision temps réel a motivé dès 1987 le développement à l'ETCA d'une architecture flot de données massivement parallèle intégrant plusieurs types de processeurs au sein d'un modèle de programmation unifié [1][8].

L'approche adoptée est proche de celle proposée par Koren [2], dans laquelle un algorithme est d'abord représenté sous la forme d'un graphe d'opérateurs puis "plaqué" de manière directe sur un réseau de processeurs flot de données. Ces processeurs exécutent alors les opérations associées aux nœuds de ce graphe et les interconnexions entre ces processeurs servent à construire les chemins de données correspondant aux arcs de ce graphe.

Notre approche se distingue toutefois par une étroite intégration du modèle d'exécution flot de données [4] et des concepts issus de la programmation fonctionnelle [3]. Il est particulièrement adapté à l'expression et à l'implantation de traitements itératifs opérant sur de gros volumes de données structurées, comme des séquences d'images vidéo.

2 Architecture matérielle

L'architecture matérielle du Calculateur Fonctionnel est détaillée sur la figure 1.

Le cœur de cette architecture est un réseau tridimensionnel de processeurs conçus pour opérer sur des flots structurés de données se déplaçant de manière sérielle sur des liens physiques. Deux types de processeurs sont intégrés dans le réseau, dédiés respectivement aux traitements bas et haut niveau.

Le processeur bas niveau, nommé μ PCF a été développé au laboratoire, en technologie "full custom" [5]. Ce processeur, conçu pour être connecté au sein d'une maille cubique tridimensionnelle comprend 6 ports d'entrée-sortie, 3 piles FIFO d'entrée, 3 piles FIFO de sortie, un multiplieur, une UAL type 2901, une mémoire de données de 256 mots et une mémoire de programme de 64 mots. La partie contrôle est implantée sous la forme d'un automate à états finis programmable. Actuellement, 1024 μ PCFs sont intégrés sous la forme d'un réseau tridimensionnel de $16 \times 8 \times 8$ processeurs.

Les processeurs haut niveau sont des Transputers T800, organisés en sous-réseaux bidimensionnels de taille 3×3 . La programmation des opérateurs flots de données s'effectue sous la forme de processus indépendants en langage C.

La machine comporte aussi des dispositifs d'entrée-sortie vidéo couleur et noir et blanc, une série d'interfaces bas et haut débit et un contrôleur couplé à une station hôte (Sparc 2).

3 Environnement de programmation

La figure 2 illustre l'organisation générale des différents outils constituant l'environnement logiciel. Cette organisation met en évidence quatre phases indépendantes, communiquant entre elles via des niveaux de description bien définis:

- **Compilation:** passage d'une description textuelle à une représentation explicite du graphe d'opérateurs.
- **Translation:** prise en compte de l'implantation physique des opérateurs et génération d'un graphe de processeurs.
- **Placement-routage** du graphe de processeurs et génération d'une configuration de réseau.
- **Chargement** de cette configuration sur l'architecture physique: programmation des processeurs et de leurs interconnexions.

La phase de compilation, prend en entrée un code source écrit dans un langage fonctionnel de description de graphes flot de données inspiré du FP de Backus (Backus 78). Les *objets* de ce langage correspondent aux données manipulées par les processeurs, les types prédéfinis incluant par exemple:

- Le PIXEL: valeur entière 8, 16, ou 24 bits.
- La LIGNE: séquence structurée de pixels. Exemple: $\langle 1,2,3 \rangle$
- La TRAME: séquence structurée de lignes. Exemple: $\langle \langle 1,2 \rangle, \langle 3,4 \rangle \rangle$

Les formes fonctionnelles correspondent aux constructeurs élémentaires des graphes flots de données. Par exemple:

- La composition, définie par: $(f \circ g) : x = f : (g : x)$, correspond au placement en série d'opérateurs.
- La construction, définie par: $[f_1, \dots, f_n] : x = (f_1 : x, \dots, f_n : x)$, correspond au placement en parallèle d'opérateurs.

Le formalisme utilisé est purement fonctionnel en ce sens que la notion de *variable* disparaît complètement de l'expression des programmes.

Le traducteur assure la conversion du graphe d'opérateur généré par le compilateur en un graphe de processeurs, formé d'opérateurs implantables physiquement sur un ou plusieurs processeurs de l'architecture. Cette phase implique éventuellement l'expansion de macro-opérateurs définis comme des sous-graphes de processeurs encapsulés.

Le placement routage a pour but de "plaquer" le graphe issu du traducteur sur la maille cubique de processeurs. Il peut être effectué soit par un algorithme automatique [7] soit par un outil graphique interactif.

La bibliothèque d'opérateurs joue un rôle essentiel au sein de l'environnement logiciel. Outre le fait qu'elle constitue le "dictionnaire" dans lequel les programmeurs applicatifs viennent chercher les descriptions publiques des opérateurs de base, elle contient aussi toutes les informations utiles au traducteur. Elle regroupe en fait trois classes d'opérateurs:

- Des opérateurs implantés sur un seul μ PCF.
- Des macro-opérateurs implantés sur un groupe de μ PCFs.
- Des opérateurs implantés sous la forme de processus C sur un transputer.

Cette bibliothèque est aisément extensible de manière à faire face à des besoins spécifiques.

4 Applications

Plusieurs applications significatives de traitement d'image ont été reformulées au sein de notre modèle de programmation et implantées avec succès sur l'architecture. Les algorithmes impliqués vont de pré-traitements bas niveau (égalisation, détection de contours, ...) à des applications de type reconnaissance de forme, basées sur l'extraction de caractéristiques géométriques ou spectrales.

Le calculateur opérant intrinsèquement à la fréquence des capteurs vidéo (jusqu'à 25 MHz) ses performances se mesurent en nombre d'opérations effectuées par pixel à cette fréquence. La puissance théorique est de l'ordre de 2000 opérations/pixel, les performances réelles s'échelonnant entre 200 et 800 opérations/pixel, en fonction de la régularité des algorithmes et de la densité du placement-routage.

5 Conclusion

Un des objectifs du Calculateur Fonctionnel est de proposer aux spécialistes de traitement d'image, une plate-forme leur permettant de spécifier, implanter et tester une large classe d'algorithmes de vision temps réel.

Un autre objectif est la conception automatique d'automates de vision embarquables. L'idée est ici d'utiliser l'architecture comme un émulateur temps réel puis d'effectuer une réduction du graphe des ressources matérielles implanté.

Enfin bien que le Calculateur Fonctionnel soit principalement dédié au traitement d'image, les concepts sur lesquels il repose sont beaucoup plus généraux et l'approche adoptée peut être généralisée au sein de systèmes incluant des processeurs de grain différent. L'utilisation de processeurs flottants 32/64 bits par exemple pourrait permettre le développement d'un "super calculateur" flot de données dédié à la résolution d'équations aux dérivées partielles, issues de problèmes de mécanique des fluides par exemple.

Figure 1: Architecture matérielle

Bibliographie

- [1] E. Allart, B. Zavidovique. Functional Image Processing through Implementation of regular Data flow Graphs. *21st Annual Asilomar Conf on Signals, Systems on Computers*, Pacific Grove CA, USA, 2-4 Nov 1987
- [2] I. Koren, B. Mendelson, I. Peled, and G.B. Silberman. A data-driven vlsi array for arbitrary algorithms. *IEEE Computer*, Oct 1988.
- [3] P. Hudak. Conception, Evolution and Applications of Functional Programming Languages. *ACM Computing Surveys*, 21(3), 1989
- [4] A.H. Veen. Dataflow Machine Architecture. *ACM Computing Surveys*, 18(4), 1986
- [5] G.M. Quenot and B. Zavidovique. A data-flow processor for real-time low-level image processing. In *IEEE Custom Integrated Circuits Conference, San-Diego, CA*, may 1991.
- [6] J. Backus. Can programming be liberated from von neumann style ? a functional style and its algebra of programs. *Communications of the ACM*, 21(8), Aug 1978.
- [7] J.Y. Colin and M. Minoux. PRCF: Un placeur-routeur pour le Calculateur Fonctionnel de l'ETCA. *Proc RenPar5 - 5ièmes Rencontres du Parallélisme*, Brest, 25-28 Mai 1993.
- [8] J. Serot, G. Quenot, B. Zavidovique. Functional Programming on a data-Flow Architecture: Applications in Real Time Image Processing. [To be published in *Intl Journal of Machine Vision and Applications* - 1993]

Figure 2: Environnement logiciel