Implantation d'un algorithme de décomposition d'image sur une architecture multi-DSP à l'aide de SynDEx

*** Mustapha ASSOUIL, ** Olivier DEFORGES**

INSA, Laboratoire ARTIST, 20 Avenue des Buttes de Coësmes, 35043 RENNES Cedex, France

* E-mail: assouil@insa-rennes.fr Tel: (33) 2 99 28 64 00, Fax: (33) 2 99 28 64 95,
** E-mail: odeforge@insa-rennes.fr Tel: (33) 2 99 28 66 48, Fax: (33) 2 99 28 64 95,

Résumé : L'objet de ce papier est de présenter les résultats de l'implantation d'un algorithme de décomposition d'image par des opérateurs morphologiques sur une architecture à base de processeurs TMS320C40 en utilisant l'environnement d'aide à l'implantation SynDEx. Ce dernier repose sur la méthodologie d'Adéquation Algorithme Architecture dont le but est de définir une implémentation optimisée de l'algorithme sur une architecture parallèle

Abstract : This paper presents the results of an implementation of a parallel multi-resolution image decomposition algorithm onto a multi-DSP network based on the TMS320C40. The SynDEx CAD software takes advantages from the Adequation between Algorithm and Architecture methodology, which goal is to find out an optimised implementation of an application algorithm on a parallel architecture.

I. Introduction

L'implantation d'algorithme de traitement d'images sur une architecture multi-DSP du type MIMD peut s'avérer assez délicate, principalement en présence de contraintes temps réel. L'étude réalisée ici consiste à tester et valider une méthodologie d'Adéquation automatique Algorithme Architecture. Cette méthodologie est basée sur le langage synchrone SIGNAL [1] pour l'étape de la validation algorithmique et sur l'environnement de programmation SynDEx [2] pour la génération d'exécutif temps réel distribué. Après une présentation de la méthodologie, nous allons décrire l'algorithme de décomposition morphologique multi-taille (DMM) ainsi que les résultats de son implémentation sur une architecture multi-DSP à bas de TMS320C40.

II Adéquation Algorithme Architecture (AAA)

Nous nous proposons d'utiliser le logiciel d'aide à l'implantation SynDEx (acronyme de Exécutif Distribué Synchrone), développé à l'INRIA. Ce logiciel repose sur la méthodologie d'Adéquation Algorithme Architecture basée sur des modèles de graphes. Ces graphes permettent de spécifier (i) l'algorithme, par mise en évidence du parallélisme potentiel, au moyen d'un graphe flot de données conditionné (GFDC); (ii) l'architecture matérielle, par mise en évidence du parallélisme disponible. A partir de ces deux graphes, SynDEx génère un exécutif distribué sans interblocage.

II.1. Modèle flot de données

Dans la méthodologie AAA, l'algorithme de l'application est spécifié à l'aide d'un graphe flot de données éventuellement conditionné (graphe logiciel). Chaque sommet de ce graphe représente ou un calcul, ou un conditionnement, ou une mémoire, d'état ou une entrée-sortie. Deux sommets sont reliés par un arc qui représente un flot de données, c'est-à-dire un transfert de données entre les deux sommets. La figure suivante montre un exemple de graphe représentant un seuillage binaire d'une image.

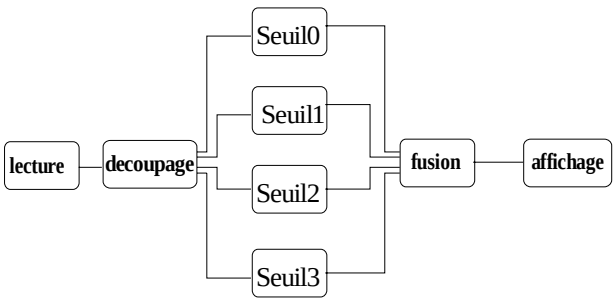

figure 1 : graphe logiciel d'un seuillage d'image

Ce modèle fait apparaître le parallélisme potentiel de l'algorithme (ordre partiel) qui permettra d'exploiter facilement le parallélisme disponible

de l'architecture. Dans l'exemple ci-dessus, les quatre fonctions de seuillage peuvent être exécutées simultanément car elles manipulent des données complètement indépendantes. Avec ce principe, les données sont connues à n'importe quel endroit du graphe puisqu'elles sont spécifiées.

Le graphe flot de données fournit une représentation synchrone d'un algorithme. En effet, il n'y a pas d'horloge apparente pour rythmer l'exécution du graphe qui est simplement conditionné par la présence ou non d'un flot de données à son entrée. Par conséquent, les caractéristiques matérielles de l'architecture ne sont pas prises en compte, ce qui revient à considérer que les calculs et les communications sont instantanées. Ces principes ont inspiré les chercheurs de l'IRISA et de l'INRIA pour créer un langage de programmation temps-réel synchrone SIGNAL [1]. Ce langage peut être vu comme le point d'entrée de SynDEx puisqu'il nous permet d'effectuer une vérification formelle de l'application et de générer un graphe flot de données directement exploitable par SynDEx.

II.2. Spécification du multi processeur.

La machine parallèle, comme le montre la figure suivante, est modélisée par un hypergraphe [3]] (*graphe matériel*) dont chaque sommet représente un processeur et chaque hyperarc une liaison physique de communication connectant plusieurs processeurs.

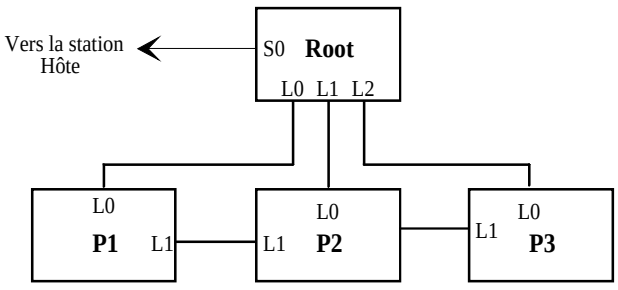

figure 2 : exemple de graphe matériel

II.3. Placement-ordonnancement sous SynDEx

L'implantation de l'algorithme sur l'architecture consiste à réduire le parallélisme potentiel de l'application au parallélisme matériel disponible. Ceci représente une distribution et un ordonnancement des actions sur les processeurs et des communications inter-processeur sur les liaisons physiques. Ces deux étapes sont statiques, c'est-à-dire qu'il n'y a pas de migration ni ordonnancement de processus au moment de l'exécution de l'application sur le réseau de processeurs.

Le Placement-ordonnancement dépend uniquement des durées de communication (calculées par SynDEx), des durées d'exécution des processus de calcul (spécifiées par l'utilisateur) et d'éventuelles contraintes de placement imposées par le programmeur. L'algorithme utilisé par SynDEx consiste à minimiser le temps de réponse ou latence c'est à dire la longueur du chemin critique du graphe logiciel.

II.4. Génération d'exécutif distribué

Après l'étape de distribution du graphe logiciel sur le graphe matériel, SynDEx génère un exécutif permettant l'exécution de l'algorithme sur l'architecture, limitant ainsi une grande partie de programmation bas niveau. Cette génération de code garantit que le comportement des entrées-sorties sera le même sur une architecture multi-processeurs que sur l'architecture mono-processeur sur laquelle l'application aura préalablement été deboguée.

III. Application: Implantation de la Décomposition Morphologique Multi-tailles (DMM)

L'algorithme de la DMM [4] consiste à décomposer une image en un ensemble d'images composantes, chacune d'entre elles ne contenant que les primitives d'une certaine taille. Ce traitement est surtout utilisé en analyse de texture pour permettre d'extraire de chaque image composante des attributs renfermant les informations de taille de primitives, de niveau de gris, etc. L'opération morphologique utilisée est *l'ouverture* définie comme étant *l'érosion* par un élément structurant suivie d'une *dilatation* par le

même élément structurant. Les différentes images composantes sont obtenues à l'aide d'un groupe d'éléments structurants de tailles décroissantes comme l'illustre la figure 3. L'extraction de primitives de grandes tailles nécessite donc l'utilisation d'éléments structurant au moins aussi grand lors de ma première ouverture de la chaîne.

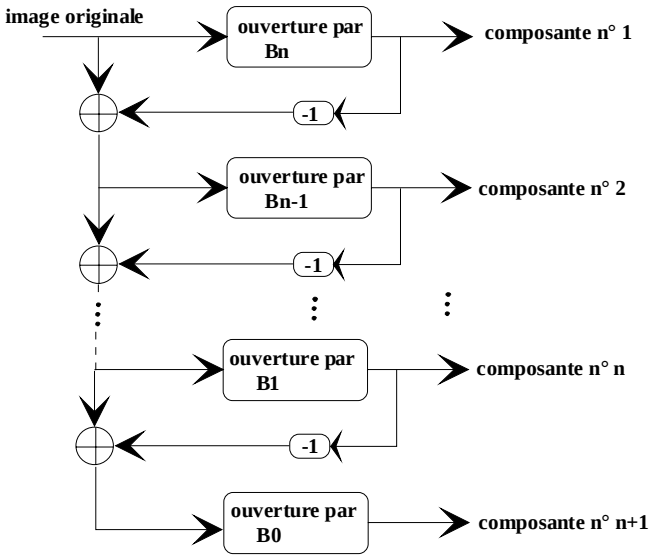

figure 3 : algorithme de la DMM

IV. Implémentation parallèle

En analysant l'algorithme de la DMM, on constate qu'il est en grande partie séquentiel. Nous avons donc décidé d'exploiter le parallélisme de données en découpant l'image originale en trois bandes. La solution ainsi retenue fait que chaque processeur doit exécuter la DMM sur l'imagette qui lui est destinée (Figure 4).

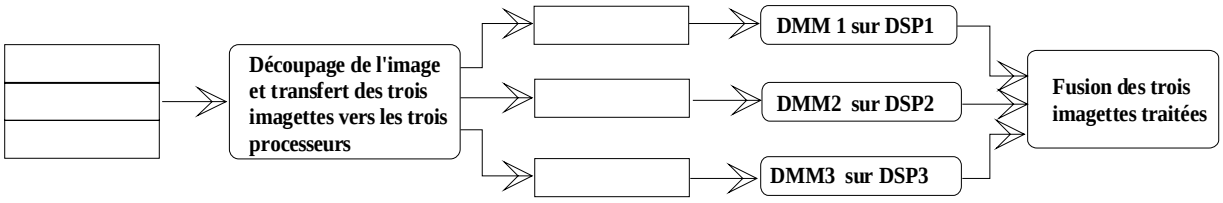

figure 4: exemple de distribution de l'image sur 3 processeurs

L'étape suivante consiste à décrire l'application complète sous forme de graphe flot de données directement sous l'environnement SynDEx, ce qui constitue le graphe logiciel comme l'illustre la partie supérieure de la figure 5. Le graphe matériel qui représente l'architecture utilisée est montré par la partie inférieure de la même figure.

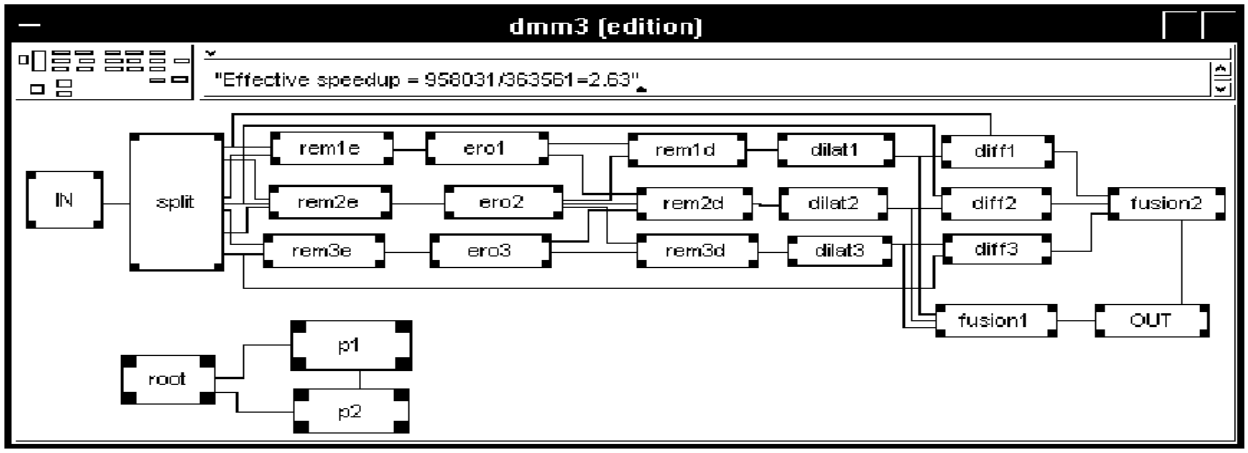

figure 5 : fenêtre d'édition graphique de l'environnement SynDEx

Les fonctions **rem***xy* ont pour rôle de mettre en forme les différentes imagettes avant les étapes d'érosion et de dilatation, de manière à ce que les zones frontières communes soient prises en compte lors des traitements. La fonction **split** découpe l'image originale en trois bandes alors que les fonctions **fusion***x* assemblent les imagettes traitées pour fournir les images composantes.

Pour spécifier complètement l'application et évaluer le facteur d'accélération (**speedup**), SynDEx a besoin de connaître les temps d'exécution de chaque tâche du graphe logiciel sur un seul processeur. Ceux-ci peuvent être estimés sur un seul processeur. SynDEx se charge alors de la gestion des communications inter-processeurs et du respect des contraintes de temps de réponse.

Exemple : L'image originale (a) de taille 128*128 est constituée essentiellement de deux primitives de tailles différentes. Après une ouverture par un élément structurant de taille 5*5, on obtient l'image composante (b) qui ne contient que les primitives de tailles supérieures au masque 5*5. La composante (c) est la différence entre les deux.

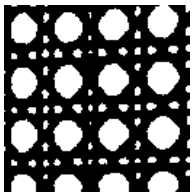
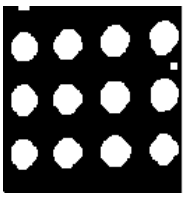


image originale image (b) image (c)

Le tableau suivant mentionne les temps obtenus pour différentes tailles de l'élément structurant pour une image de taille 128*128, et ce pour un nombre de processeurs utilisé allant de un à trois. L'architecture est constituée d'une carte mère de chez Sundance (SMT320) et de deux modules au standard TIM40 incluant un dispositif d'acquisition d'images et trois processeurs TMS320C40.

DMM d'une image 128*128 par un élé. Struct. 5*5			
nb de processeur	**Durée (ms)**	**Accélération**	**efficacité (%)**
1	1.42	1	1
2	92	1.53	0.76
3	65	2.18	0.72
DMM d'une image 128*128 par un élé. Struct. 9*9			
1	288	1	1
2	191	1.51	0.75
3	115	2.49	0.83
DMM d'une image 128*128 par un élé. Struct. 11*11			
1	364	1	1
2	241	1.51	0.75
3	141	2.57	0.85

V. Conclusion

Cet exemple d'implantation d'une application de traitement d'image a permis de valider le portage rapide d'un algorithme sur une architecture multi-DSP à l'aide de SynDEx. Les résultats peuvent être amélioré d'une part en optimisant les différentes fonctions de calcul, et d'autres part en cherchant d'autres flux de données ou d'autres manières d'exploiter le parallélisme de l'application. Ces différents test peuvent être menés rapidement à travers SynDEx puisque le code source à écrire par l'utilisateur fait abstraction de la topologie. Seule est à redéfinir la spécification en flot de données de l'algorithme.

Références

[1] P. LE GUERNIC, M. LE BORGNE, T. GAUTIER, et C. LE MAIRE. *Programming real-time applications with SIGNAL*. Rapport de recherche 1446, INRIA,Juin 1991.

[2] C. LAVARENNE, O. SEGHROUCHNI, Y. SOREL, et M. SORINE. *SynDEx, un environnement de programmation pour applications de traitement de signal distribuées.* Actes du treizième colloque GRETSI, Juin Les Pins, septembre 1991.

[3] C. LAVARENNE, Y. SOREL. *Spécification, optimisation de performance et génération d'exécutifs pour application temps-réel embarquée multiprocesseur avec SynDEx.* Les saintes Maries de la mer, CNES international symposium, 1992.

[4] D. Wang. *Décomposition morphologique multi-taille et segmentation adaptative de textures.* Thèse de doctorat au laboratoire ARTIST de l'INSA de Rennes. 1991.