Algorithme-Architecture Parallèle Asynchrone pour la Segmentation d'Image par Ligne de Partage des Eaux

B. Galilée[1] M. Renaudin[2] P-Y. Coulon[3] F. Mamalet[1]

[1] France Telecom R&D, [2] TIMA - Grenoble, [3] LIS - Grenoble,

France Telecom R&D
28 chemin du vieux Chêne, BP 98, 38243 Meylan cedex
bruno.galilee@rd.francetelecom.com

Résumé

Une étude conjointe algorithme-architecture débouche sur l'implantation d'un algorithme de ligne de partage des eaux (LPE) parallèle à fine granularité sur un réseau de processeurs asynchrones. Pour cela, un réordonnancement original de cet algorithme a été étudié. Ce système nous permet de segmenter des images QCIF[1] en temps réel tout en limitant la consommation d'énergie.

La modélisation et la simulation de l'algorithme adaptées à l'architecture ciblée sont effectuées en "SystemC".

Mots Clef

Ligne de Partage des Eaux (LPE), algorithme parallèle, algorithme asynchrone, réseau de processeurs asynchrones, SystemC.

1 Introduction

Les projets liés aux mobiles de troisième génération nous promettent l'introduction de la vidéo ("streaming vidéo"), voire de la visiophonie, dans les terminaux. Par ailleurs, la norme MPEG-4 envisage d'utiliser la segmentation pour décomposer la scène en objets vidéo et obtenir de forts taux de compression en ne codant que les zones d'intérêt. Cependant, ces traitements sont gourmands en puissance de calcul et les terminaux portables ont de fortes contraintes de consommation. C'est pourquoi, l'objectif de notre travail est de proposer une architecture dédiée basse consommation, capable de segmenter en temps réel une séquence d'images (au minimum 15 images QCIF/s).

Contrairement aux circuits synchrones, les circuits asynchrones fournissent un résultat au plus tôt et non au pire cas, et l'énergie est par ailleurs réduite par une activité conditionnelle des éléments de calcul: seuls les processeurs qui disposent de données pertinentes calculent et donc consomment.

[1] Format d'image 176*144 pixels

Nous présentons un nouvel algorithme parallèle de segmentation par ligne de partage des eaux (LPE) ainsi qu'un schéma d'architecture asynchrone associé. Nous proposons une méthode de modélisation du couple algorithme-architecture, en langage de haut niveau, en vue d'une validation par simulation du système. Enfin, nous terminons par une conclusion et des perspectives.

2 Algorithmes de LPE et stratégies de parallélisation

De la biologie à la vidéo (MPEG-4) en passant par les études d'images satellites, l'algorithme de LPE est au cœur de nombreux projets de recherche. Le système que nous proposons permet de segmenter des images sans connaissance a priori et dans des conditions critiques (faible complexité du circuit, basse consommation, temps réel, ...).

2.1 Etat de l'art

Une étude comparative de nombreux algorithmes de LPE et leur implantation est présentée dans [8]. L'algorithme classique de LPE par immersion [10] à partir des minima régionaux est composé des étapes suivantes: détection des minima, étiquetage, immersion du relief et détermination des pixels frontières.

Une version parallèle consiste à partitionner l'image en sous-domaines rectangulaires où chaque processeur exécute l'algorithme séquentiel [1] [2] [4], le résultat global est alors obtenu par fusion des données locales. Bien que ces méthodes soient rapides, leur implantation matérielle ne répond pas aux critères de basse consommation et d'intégration dans un terminal de faible taille (architecture complexe [5]). Les algorithmes rapides de LPE utilisent les Files d'Attentes Hiérarchiques (FAH), mais leur mise en œuvre sur une architecture dédiée [3, Chap.6] est délicate car elle implique une architecture complexe du circuit. Des solutions sont proposées dans [5] pour simplifier ce système.

2.2 Solution développée

A partir de l'approche présentée dans [7], nous avons déterminé un couple algorithme-architecture rapide ($< 1ms$ par image) et faiblement consommant (énergie quasi-nulle pour les processeurs inactifs). Nous avons établi un désordonnancement de l'algorithme de LPE (4-connexité), pour une implantation parallèle sur un modèle de flot de données, en désynchronisant les traitements dans l'image, tout en garantissant bien sûr, un résultat final équivalent.

Description de l'algorithme: Soit $\mathcal{N}_G(p)$, le voisinage du pixel p dans la grille G 4-connexe, et soit f, l et m, 3 fonctions d'un treillis T sur un ensemble fermé des entiers positifs $\overline{\mathbb{N}}$, où $f(p)$ correspond au niveau de gris de p, $l(p)$ le label et $m(p)$ le niveau de gris du minima associé à p.
A l'initialisation, on détermine la topologie de p:

- *"pixel minima ou plateau"* (MP) si

$$\forall q \in \mathcal{N}_G(p)\,,\; f(q) \geqslant f(p) \tag{1}$$

- *"pixel non minima"* (NM) si

$$\exists q \in \mathcal{N}_G(p)\,,\; f(q) < f(p) \tag{2}$$

ainsi que les ensembles suivants:

$$\mathcal{N}_G^{+}(p) = \{q \in \mathcal{N}_G(p) \mid f(q) \geqslant f(p)\}$$

$$\mathcal{N}_G^{=}(p) = \{q \in \mathcal{N}_G(p) \mid f(q) = f(p)\}$$

$$\mathcal{N}_G^{I}(p) = \begin{cases} \emptyset & \text{si } \mathcal{N}_G^{+}(p) = \mathcal{N}_G(p) \\ \{q\} \mid & q \in \mathcal{N}_G(p) \\ & f(q) = min\big(f(r), r \in \mathcal{N}_G(p)\big) \\ & \text{et de label minimum} \end{cases}$$

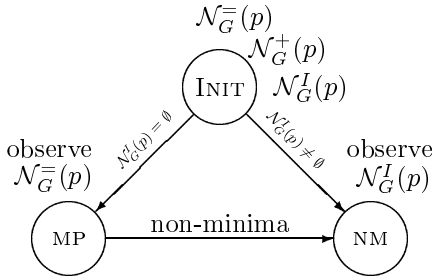

FIG. 1: Machine à 3 états associée à chaque pixel

L'algorithme est adapté à une architecture massivement parallèle (1 processeur par pixel) et fonctionne en mode SPMD. La phase d'initialisation (fig 1) consiste à déterminer $\mathcal{N}_G^{=}(p)$, $\mathcal{N}_G^{+}(p)$ et $\mathcal{N}_G^{I}(p)$ puis, la machine passe à l'état:

$$\varphi(p) = \begin{cases} \text{MP} & \text{si } \mathcal{N}_G^{I}(p) = \emptyset \\ \text{NM} & \text{sinon} \end{cases} \tag{3}$$

Au cours de l'immersion, chaque processeur a le comportement suivant:

- Si $\varphi(p) =$ MP, un processeur p écoute $\mathcal{N}_G^{=}(p)$ et met à jour son étiquette s'il reçoit $l(q) < l(p)$ (cas d'un plateau minima). S'il reçoit un niveau de gris $m(q) < m(p)$, cela signifie qu'il est situé sur un plateau non-minima. Il fixe $\mathcal{N}_G^{I}(p)$ en conséquence et passe à l'état $\varphi(p) =$ NM.
- Si $\varphi(p) =$ NM, le processeur p étant déjà innondé, n'écoute que le voisin $\mathcal{N}_G^{I}(p)$, et remet à jour son état s'il reçoit un niveau de gris $m(q) < m(p)$ (réimmersion), ou s'il reçoit un label $l(q) < l(p)$ (le label du minima a changé).

Si ses variables internes sont modifiées, le processeur p propage $m(p)$ et $l(p)$ vers ses voisins en amont $\mathcal{N}_G^{+}(p)$.

Caractéristiques: La particularité de cet algorithme est que l'immersion s'effectue simultanément à partir de *tous* les pixels. La détection des minima, étiquetage et immersion sont réalisées simultanément au niveau de chaque processeur. L'algorithme converge car :

- nous mémorisons la provenance de l'immersion ($\mathcal{N}_G^{I}(p)$ pour $\varphi(p) =$NM) et sa profondeur $m(p)$
- la remise en question des labels $l(p)$ suit une fonction décroissante de valeurs dans $\overline{\mathbb{N}}$.

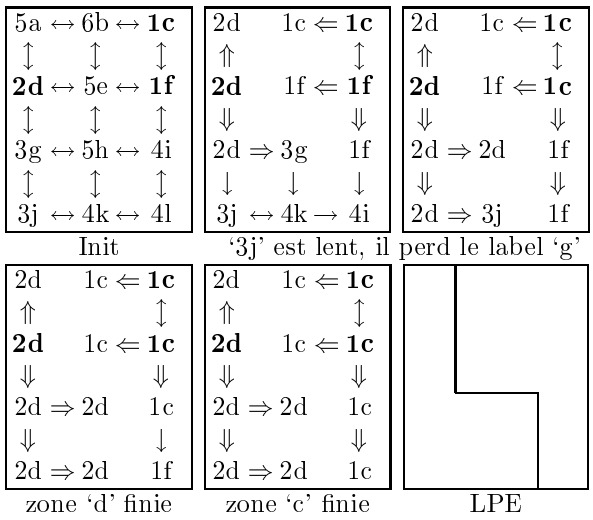

FIG. 2: Exemple d'immersion à partir d'une image gradient (minima en gras)

Exemple: La figure 2 présente un exemple d'immersion pour une image 3*4. Chaque pixel p est représenté par le couple 'xy' où $x = m(p)$ et $y = l(p)$. Afin de différencier plus facilement les étiquettes des niveaux de gris associés au minima, nous avons utilisé les lettres de l'alphabet comme label où les relations d'ordre sont basées sur l'ordre alphabétique. Une flèche mince ($\rightarrow$) représente un transfert possible des données entre 2

pixels ($\varphi(p)$ =INIT ou MP) alors qu'une flèche épaisse ($\Rightarrow$) représente une immersion ($\varphi(p)$ =NM).
Au point d'équilibre, l'image est composée des 2 étiquettes 'd' et 'c' qui localisent les 2 bassins versants de l'image. Nous obtenons une LPE d'épaisseur nulle située entre 2 pixels d'étiquette différente.

3 Réseau de processeurs asynchrones

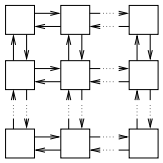

FIG. 3: Schéma synoptique du réseau

Un partitionnement classique consiste à attribuer une zone sans recouvrement carrée ou rectangulaire de l'image à chaque processeur. La structure du réseau cible (fig 3) peut atteindre le niveau de granularité le plus fin (un processeur asynchrone par pixel). Afin de simplifier cette structure, nous nous sommes placés en 4-connexité, sachant que ce choix, en contraignant les communications inter processeurs, a une influence sur la partition de l'image et sur l'algorithme.

3.1 Modèle d'exécution

Chaque processeur est doté d'une unité arithmétique et logique, d'une zone mémoire programme et d'une zone mémoire de données ($f(p)$, $l(p)$, $\mathcal{N}_G^I(p)$,...). Le programme, contenu au cœur de chaque processeur, effectue l'algorithme présenté au §2 d'une manière localement synchronisée. Si un nœud du réseau possède toutes les données nécessaires issues de ses voisins, alors il traite ces données, sinon, le processeur reste inactif. Ce comportement permet d'obtenir le résultat au plus tôt, avec un nombre minimum d'opérations, et donc une consommation minimale.

Contrairement aux systèmes synchrones, il est impossible de prédire l'instant de la terminaison du calcul mais, pour une image donnée, on peut déterminer une zone pire cas et un temps pire cas. Bien que le temps de calcul soit variable suivant les données, il est borné car l'algorithme ne boucle pas ni ne crée d'inter-bloquage. La terminaison sera détectée grâce à un "OU" global de l'état d'activité des processeurs ('1' actif, '0' inactif). Le contrôleur détectera le point de convergence du réseau dès que le résultat du calcul combinatoire vaudra '0'.

3.2 Modèle de communication

Aucun signal global d'horloge n'étant présent, seuls les signaux de communication, grâce à un protocole adéquat, vont ordonnancer les phases de l'algorithme en chaque nœud du réseau. Nous avons étudié des communications synchronisées dites "bloquantes", et non synchronisées dites "non-bloquantes".
Dans le premier cas, le processeur qui envoie une donnée reste bloqué tant que le processeur cible n'a pas renvoyé un acquittement. Cette synchronisation ou "point de rendez-vous", est implantée par un protocole de type "requête-acquittement", ou dit à "poignée de mains" [6]. Une file d'attente ou "FIFO", de taille variable placée sur chaque canal de communication, permet de relâcher plus ou moins la synchronisation entre les phases bloquantes de l'algorithme exécutées par les processeurs.
Dans le second cas, le processeur émetteur continue son programme, que la communication aie réussi ou non. Sur les canaux, on appellera "mémoire tampon", une file d'attente où les données les plus anciennes sont supprimées lorsqu'il y a saturation.

3.3 Simulation et validation

La modélisation d'un tel système nous permet de valider l'algorithme, et d'évaluer le spectre des architectures possibles dépendantes du modèle d'exécution et du modèle de communication inter-processeurs.
Afin de déterminer la meilleure adéquation algorithme-architecture, il faut que les outils de simulation puissent valider les différentes solutions étudiées, et évaluer les performances, la consommation et éventuellement la surface de silicium du circuit. Le langage CHP [6] est adapté à la description des systèmes asynchrones, mais, il souffre pour l'instant d'un manque d'outils. Une simulation VHDL est délicate car elle nécessite plusieurs heures pour segmenter une image QCIF. C'est pourquoi, nous avons choisi de simuler ce réseau en langage de haut niveau en utilisant "SystemC" [9].
Les communications s'effectuent via une mémoire tampon de taille 1. Un système de "*probe*" permet de tester la présence ou absence de données dans le canal et d'effectuer des communications non-bloquantes. Lors de la phase d'initialisation, chaque pixel attend d'avoir reçu les données issues de ses 4 voisins avant de passer à l'état MP ou NM. Comme les communications sont non-bloquantes, un processeur lent lors de la phase d'initialisation peut considérer des données d'immersion d'un voisin rapide (d'où perte des messages précédents) comme des données d'initialisation, sans que cela nuise au résultat final.
Lors de la phase d'immersion, un processeur est considéré inactif à partir du moment où aucune donnée n'est présente sur ses ports d'entrée le reliant avec $\mathcal{N}_G^=(p)$ si $\varphi(p)$ = MP (resp. $\mathcal{N}_G^I(p)$ si $\varphi(p)$ = NM). Dès qu'un processeur voisin le réveille, il devient actif, met à jour ses variables internes et propage ses résultats, si besoin est, vers $\mathcal{N}_G^+(p)$. Des données transitoires peuvent être perdues car tous les processeurs utilisent le même protocole non-bloquant, seuls les ré-

sultats les plus récents sont pris en compte.

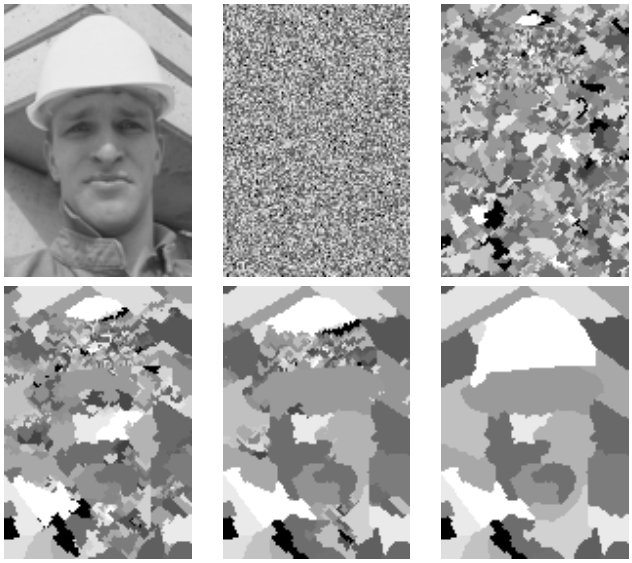

FIG. 4: Image originale et étiquettes du réseau au cours du temps (de la gauche vers la droite et du haut vers le bas)

La figure 4 présente quelques échantillons des labels des pixels au cours du temps (chaque niveau de gris correspond à un label particulier). L'image utilisée (96*140 pixels) est l'image gradient simplifiée de la tête de "foreman". La propagation désordonnée des étiquettes est particulièrement remarquable sur les larges régions (casque, front), les petites régions convergeant très rapidement.

4 Résultats et perspectives

Cet article décrit les résultats de simulation à l'aide de "SystemC" et présente une analyse conjointe algorithme-architecture.

Résultats: Seule une partition très fine de l'image (1 processeur par pixel) a été explorée pour l'instant. Nous avons décomposé l'algorithme sous forme d'opérations élémentaires effectuées par chaque processeur (lecture, écriture, tests,...).
Grâce au simulateur, nous avons établi des statistiques sur la repartition de l'activité du réseau. Pour atteindre le point de convergence de l'algorithme, chaque processeur utilise uniquement quelques variables et communique en moyenne une vingtaine de fois avec ses voisins. Par ailleurs, pour l'image de la figure 4, nous avons observé une chaîne critique de 806 opérations dont on a estimé un temps moyen de $17.5ns$ par opération. On obtient un temps de convergence du réseau de l'ordre $14\mu s$, contre $163ms$ pour l'algorithme séquentiel exécuté par un processeur embarqué tel que l'ARM9 (3 258 000 opérations avec une dérive de 10 instructions par opération pour une capacité de 200Mips), soit un facteur d'accélération proche de 12 000 pour une consommation d'énergie plus faible de plusieurs ordres de grandeur.

Perspectives: Une conception précise du cœur du processeur asynchrone nous permettra d'évaluer plus finement le temps de traversée des différents types d'opérations et donc, le temps de convergence du réseau, ainsi que la consommation du circuit. Les prochains travaux se focaliseront sur l'implantation microélectronique d'un tel circuit.

Références

[1] Bieniek (A.), Burkhardt (H.), Marschner (H.), Nölle (M.) et Schreiber (G.). – A parallel watershed algorithm. *10th Scandinavian Conference on Image Analysis (SCIA)*, june 1997. – Lappeenrante, Finland.

[2] Laurent (C.). – *Conception d'algorithmes parallèles pour le traitement d'images utilisant la morphologie mathématique: Application à la segmentation d'images.* – Thèse de PhD, Université Bordeaux I, 1998.

[3] Lemonnier (F.). – *Architecture électronique dédiée aux algorithmes rapides de segmentation basés sur la Morphologie Mathématique.* – Thèse de PhD, Ecole Nationale des Mines de Paris, 1996.

[4] Moga (A.). – *Parallel Watershed Algorithms for Image Segmentation.* – Thèse de PhD, Tampere University of Technology, Finland, 1997.

[5] Noguet (D.). – *Architectures parallèles pour la morphologie mathématique géodésique.* – Thèse de PhD, INPGrenoble, 1998.

[6] Renaudin (M.). – Asynchronous circuits and systems: a promising design alternative. *Microelectronics for Telecommunication: managing Microelectronics and mobility (MIGAS2000)*, vol. 54, n° 1–2, decembre 2000, pp. 133–149.

[7] Robin (F.), Privat (G.) et Renaudin (M.). – Asynchronous relaxation of morphological operators: a join algorithm-architecture perspective. *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 11, n° 7, 1997, pp. 1085–1094.

[8] Roerdink (J. B.) et Meijster (A.). – The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta Informaticae*, no41, 2001, pp. 187–228. – IOS Press.

[9] SystemC home page: a modeling platform that enables, promotes and accelerates system-level co-design and IP exchange. URL, http://www.systemc.org.

[10] Vincent (L.) et Soille (P.). – Watershed in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 13, n° 6, June 1991, pp. 583–598.