Analyse architecturale d'applications de compression de séquences d'images

François Charot, Charles Wagner

Irisa
Campus de Beaulieu, 35042 Rennes Cedex, France
email : charot@irisa.fr

Résumé

Dans cet article, nous analysons les algorithmes de compression de séquences d'images animées et étudions leur mise en œuvre en temps réel sur des architectures parallèles, spécialisées et programmables. Les propriétés de ces algorithmes permettent d'envisager des solutions matérielles hautement parallèles comme des architectures SIMD. La programmation de telles architectures ainsi que les fonctionnalités du processeur élémentaire sont discutées.

Abstract

In this paper, we analyse the image sequence coding algorithms and study their real-time implementation on specialized parallel and programmable architectures. The properties of these algorithms allow highly parallel implementations such as SIMD architectures to be considered. The programming method and the elementary processor architecture are discussed.

Introduction

Les applications de traitement numérique du signal vidéo impliquent des algorithmes de plus en plus complexes. Il ne s'agit plus uniquement de mettre en œuvre un algorithme de base mais de maîtriser toute une chaîne de traitements qui interagissent. Un exemple typique est la compression de séquences d'images animées, utilisée dans les applications de TVHD, de stockage et de transmission par réseaux ou satellites [8]. D'un point de vue de l'étude d'architectures matérielles, ce domaine d'application est intéressant à plusieurs titres. En effet, même si des schémas de codage sont maintenant normalisés[1] (34 mégabit/s [5], MPEG [4]), il n'en demeure pas moins que la mise en œuvre de certains traitements de l'application (choix de l'algorithme d'estimation de mouvement, critères de mode de codage, régulation) est laissée à l'appréciation du concepteur. De nouvelles approches de codage, telles que le codage en sous-bandes doivent également pouvoir être évaluées. Aussi afin d'étudier l'impact de ces choix algorithmiques, une simulation en temps réel est nécessaire avant la mise en œuvre matérielle à l'aide de circuits spécifiques. Cette phase de simulation fait partie intégrante du processus de conception d'un système matériel spécialisé [2].

Ces applications, très coûteuses en calculs, ne peuvent être simulées en temps réel que sur des architectures parallèles spécialisées programmables et configurables. En effet, les architectures spécialisées actuelles, rarement modulaires, et offrant peu de parallélisme, ne pourront satisfaire les besoins en calcul impliqués dans les traitements futurs. Les machines parallèles à usage général, commercialisées ou à l'état de prototype, ne possèdent pas d'entrées/sorties vidéo et ne sont donc pas adaptées aux contraintes du temps réel. Même si à terme, elles ont les caractéristiques souhaitées (temps réel, performances), elles resteront de toute façon difficile d'accès et d'un coût prohibitif.

L'objet de cet article est d'analyser les algorithmes de compression de séquences d'images animées et d'étudier leur mise en œuvre en temps réel sur des architectures parallèles spécialisées programmables de type SIMD. Le fonctionnement de l'architecture est illustré par l'exemple de l'estimation de mouvement basée sur l'approche exhaustive. La programmation d'une telle architecture ainsi que les fonctionnalités du processeur élémentaire sont discutées.

1 Principe des schémas de compression

Les chaînes de compression de séquences numériques d'images animées (codage à 34 mégabit/s, MPEG) utilisent conjointement une méthode de compression de type fréquentiel et une méthode prédictive avec une quantification éventuellement adaptative et un codage à longueur variable.

Les images sont découpées en macro-blocs, chaque macro-bloc étant constitué, dans la norme 34 mégabit/s, de deux blocs de luminance 8x8 pixels[2] et de deux blocs de chrominance 8x8 pixels. Ces chaînes de compression utilisent principalement trois techniques pour comprimer l'image.

- L'estimation de mouvement consiste, pour un macro-bloc donné, à rechercher dans une image précédente le macro-bloc le plus "ressemblant". Seule la différence

[1] Des circuits spécifiques (transformée en cosinus, estimation de mouvement par mise en correspondance de blocs basée sur une recherche exhaustive) sont maintenant commercialisés. Ils permettent par assemblage de quelques circuits la réalisation de codeurs, mais aucune programmabilité n'est permise.

[2] Dans la norme MPEG 1, le macro-bloc est constitué de quatre blocs de luminance.

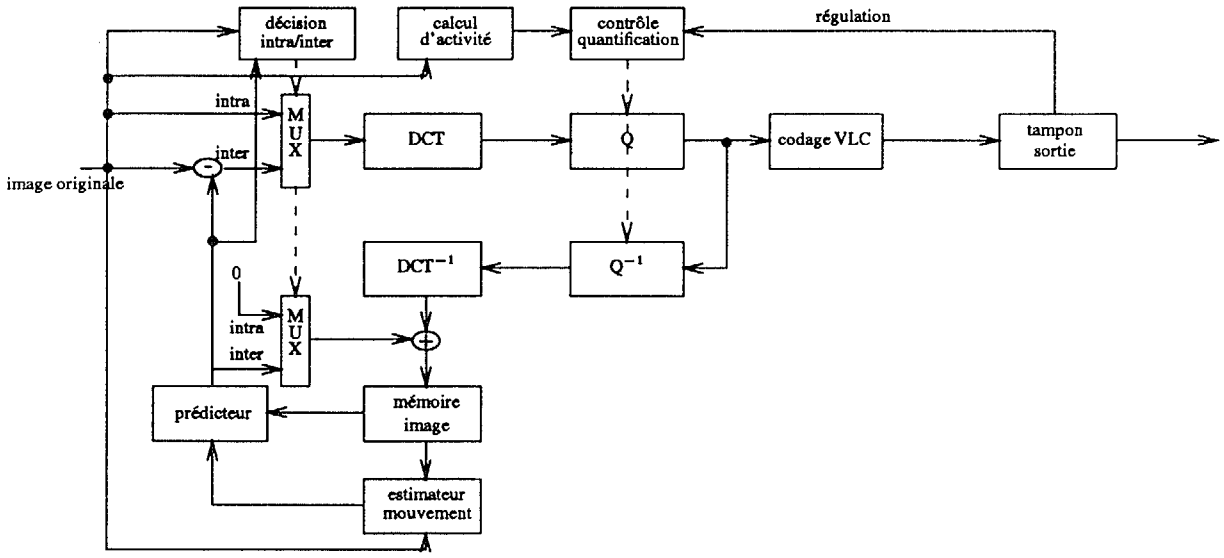

Figure 1 : Schéma de principe d'une chaîne de codage

entre les deux blocs et le vecteur de mouvement sont transmis (codage en mode inter). Lorsque le résultat produit par l'estimateur de mouvement n'est pas exploitable, le macro-bloc est transmis tel quel (codage en mode intra).

- La transformée en cosinus (DCT), appliquée à chaque bloc du macro-bloc, réalise une décorrélation des coefficients du bloc. Les coefficients sont ensuite quantifiés.
- La compression est réellement réalisée à l'aide d'un codage entropique qui consiste à remplacer les coefficients quantifiés par un code dont le nombre de bits est d'autant plus réduit que la probabilité d'apparition de la valeur est grande.

Le débit en sortie du codeur dépend du contenu des images. Les variations de débit sont absorbées par la présence de tampons d'émission et de réception au niveau du codeur et du décodeur. Le coefficient de quantification est ajusté en fonction du taux de remplissage du tampon d'émission (régulation du tampon).

2 Parallélisation de ces applications

L'étude des différentes tâches algorithmiques amène un certain nombre de constatations, sur lesquelles s'appuie notre étude architecturale.

- Les traitements de base sont réalisés sur des macro-blocs et sont par conséquent locaux.
- Dans la plupart des cas, le même traitement est appliqué à chaque bloc du macro-bloc.
- Les traitements des différents macro-blocs d'une ligne de macro-blocs voire même de l'image entière sont indépendants.
- Seule l'étape d'estimation de mouvement nécessite la connaissance de données appartenant à des blocs voisins pris dans l'image précédente, ces données constituent l'espace de recherche dont la taille est fonction du déplacement considéré.
- Le coût des traitements de base est très variable : 684 MOPS (million d'opérations par seconde) pour la DCT, 14,5 GOPS (giga opérations par seconde) pour l'estimation de mouvement avec une recherche exhaustive dans le cas du 34 mégabit/s.

Les bonnes propriétés de ces traitements permettent d'envisager des solutions matérielles hautement parallèles comme des architectures SIMD, systoliques ou pipelines. La difficulté principale, à laquelle le concepteur doit faire face, est de prendre en compte les interactions entre ces parties régulières de telle sorte que les problèmes liés à l'utilisation optimale des ressources mémoire, de la puissance de traitement requise, des aspects entrées-sorties soient résolus de façon optimale.

L'analyse algorithmique nous amène assez naturellement à privilégier le modèle d'architecture SIMD, assez couramment employé en traitement d'images [1]. Notre but est de montrer comment de telles applications peuvent être mises en œuvre en temps réel sur un réseau linéaire SIMD programmable.

2.1 Modèle d'architecture SIMD

Le modèle d'architecture considéré (figure 2) appartient à la classe des machines SIMD. Une architecture SIMD est constituée d'un ensemble d'unités opératives pilotées par une seule unité de contrôle, il s'agit par conséquent d'une architecture synchrone. L'unité de contrôle diffuse à chaque cycle, l'instruction qui est simultanément exécutée par toutes les unités opératives.

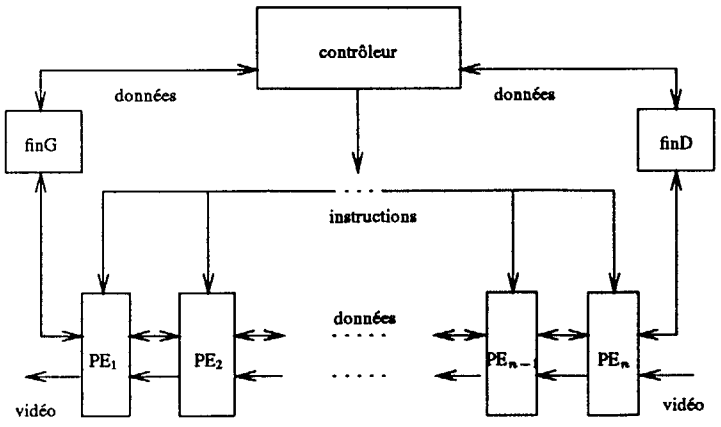

Figure 2 : Modèle SIMD

Précisons les hypothèses de base qui permettent d'une part, de mieux cerner le modèle retenu et d'autre part, de guider la parallélisation de tels schémas de compression.

- Les processeurs sont organisés en réseau linéaire.
- Chaque processeur communique localement avec ses voisins immédiats (gauche et droit). Les communications peuvent être effectuées en parallèle avec les traitements. Le rapport temps de calcul sur temps de communication est égal à un afin de supporter un parallélisme à grain fin.
- Le nombre de processeurs est au plus égal au nombre de pixels dans une ligne de l'image [3]. L'image est distribuée entre les processeurs de telle sorte que le processeur de rang j mémorise la colonne de pixel d'indice j. Il n'y a pas de duplication de données image.
- Chaque processeur a sa propre mémoire locale et dispose d'une capacité de stockage fonction de la taille des images (nombre de lignes) et du nombre d'images manipulées à un instant donné par l'application.
- Le réseau dispose de mécanismes d'entrées/sorties vidéo (acquisition des lignes d'images, restitution des résultats). Ces mécanismes sont distincts des mécanismes de communication inter-processeurs nécessaires au déroulement des calculs.

Le volume de calcul inhérent à ces applications est tel que seules des architectures massivement parallèles permettront de résoudre efficacement le problème et de satisfaire les contraintes de temps réel. Il convient cependant de noter que d'autres distributions de données sont possibles. Une répartition par blocs ou groupe de blocs (colonne de macro-blocs par processeur) est envisageable [7]. Cette granularité ayant pour conséquence la diminution du nombre de processeurs, il convient de reconsidérer la topologie et d'opter pour des approches du type réseau bidimensionnel de processeurs.

2.2 Mise en œuvre des traitements de base

La parallélisation de l'application de codage 34 mégabit/s repose sur les idées suivantes :

1. A un instant donné, un traitement de base (DCT, estimation de mouvement, ...) est appliqué à l'ensemble des macro-blocs MB d'une ligne de macro-blocs LB. En effet, les dépendances de données dans la boucle de codage, liées à la régulation, autorisent le traitement en parallèle de l'ensemble des macro-blocs d'une ligne. Il s'agit d'un parallélisme sur les données.
2. Du fait de la répartition des données, un sous-réseau de 16 processeurs a la charge du traitement d'un bloc BR_{YY} (composante de luminance). Chaque processeur du sous-réseau a connaissance d'une colonne de 8 pixels du bloc 8x16. Les huit premiers processeurs du sous-réseau traitent également le bloc BR_{Cr}, les huit derniers traitent le bloc BR_{Cb} (composantes de chrominance).
3. Les différents traitements de base sont successivement enchaînés sur le réseau dans l'ordre d'apparition dans la boucle de codage de la figure 1.

Exemple : l'estimation de mouvement
Le principe de l'estimation de mouvement par corrélation de blocs est défini comme suit. Pour chaque bloc de taille nxn de l'image courante, on recherche un bloc de même taille dans l'image précédente qui minimise la distance entre les deux blocs. Cette recherche est effectuée à l'intérieur d'une fenêtre de recherche de taille $(2d+n)(2d+n)$. Il s'agit de calculer pour toutes les valeurs k et l, telles que $-d \leq k, l \leq +d$, la distance :
$d(k,l) = \sum_{i=1}^{n} \sum_{j=1}^{n} |y(i+k, j+l) - x(i,j)|$,
puis à déterminer la distance minimale :
$u = min_{(k,l)} d(k,l)$.

Un sous-réseau de 16 processeurs réalise l'estimation de mouvement pour un MB. Un processeur PE_j mémorise en permanence une colonne j du MB courant et une colonne j de l'image précédente (espace de recherche). Les $x(i,j)$ sont les pixels de la colonne du MB résidants dans le processeur. Les $y(i+k, j+l)$ sont les pixels de la colonne du bloc déplacé dans l'espace de recherche pour un déplacement (k,l) situés à une distance l du processeur considéré.

Chaque processeur du sous-réseau calcule une distance partielle $\sum_{colonne} |y(i+k, j+l) - x(i,j)|$, celles-ci seront ensuite accumulées, sur un processeur particulier du sous-réseau, pour produire $d(k,l)$. Ce processeur réalise également la minimisation des distances $d(k,l)$, pour en déduire le vecteur de mouvement qui sera ensuite diffusé vers tous les processeurs du sous-réseau. Les colonnes des blocs déplacés sont propagées de proche en proche de la droite vers la gauche puis de la gauche vers la droite. L'espace de recherche est ainsi balayé de haut en bas. Cette opération de propagation est réalisée en parallèle avec le calcul.

2.3 Programmation du réseau

Le développement d'algorithmes parallèles comme celui décrit précédemment est facilité par l'utilisation du langage RELACS [6], développé à l'Irisa. Le modèle de programmation de RELACS permet au programmeur de s'abstraire de la machine cible, le parallélisme s'exprime par les données, le contrôle est séquentiel et s'applique de façon globale à tous les processeurs. Des opérateurs d'affectation

particuliers permettent de décrire explicitement les communications entre processeurs voisins et entre processeurs extrêmes et l'extérieur du réseau. L'utilisateur écrit en RELACS un seul programme à partir duquel le compilateur génère à la fois le code pour les processeurs du réseau et celui de l'interface. Cet environnement de programmation a été utilisé pour valider la mise en œuvre parallèle proposée. L'analyse du code RELACS permet de déterminer précisément les différentes classes de traitement réalisées par le processeur et de mesurer leurs importances les unes par rapport aux autres.

2.4 Fonctionnalités du processeur

La partie opérative du processeur doit être conçue de telle sorte que les calculs du type produit scalaire de vecteurs, somme de différence inter-pixels, somme de différence au carré, ... soient réalisés le plus efficacement possible en considérant que les données peuvent aussi bien être locales au processeur que provenir d'un voisin.

La réalisation efficace des calculs suppose la mise en œuvre de mécanismes de communication adéquats comme l'acquisition et l'émission de données à chaque cycle.

La table qui suit indique pour chacun des traitements de la chaîne de compression le nombre d'opérations arithmétiques effectuées par processeur pour un macro-bloc ainsi que le coût de traitement par processeur (en millions d'opérations par seconde) selon l'algorithme d'estimation de mouvement utilisé (4 pas, exhaustif). On a fait l'hypothèse que le processeur exécutait une opération arithmétique par cycle d'horloge. Il est à noter que la partie opérative peut être conçue pour réaliser plusieurs opérations arithmétiques simultanément. Le niveau de performance requis peut ainsi être satisfait.

traitement	macro-bloc	séquence 1 seconde
DCT	528	0,95 MOPS
Q	640	1,15 MOPS
DCT^{-1}	528	0,95 MOPS
Q^{-1}	640	1,15 MOPS
mvt 4 pas	2032	3,65 MOPS
mvt exhaustif	26534	48 MOPS
décision	222	0,4 MOPS
activité	316	0,6 MOPS
34 Mbit/s (exhaustif)	29186	54 MOPS
34 Mbit/s (4 pas)	4684	9 MOPS

2.5 Approche multi-réseau

Dans le but de satisfaire les contraintes de temps réel, l'utilisation de plusieurs réseaux, du type de celui présenté figure 2, est possible. Chaque réseau réalise alors un étage de la chaîne de compression et communique avec les réseaux voisins par l'intermédiaire de ses liaisons vidéo. Cette approche présente l'avantage de permettre un dimensionnement de l'architecture en fonction de la complexité du problème, chaque réseau pouvant par exemple être "taillé" sur mesure.

Le fonctionnement de l'ensemble est de nature "multi-SIMD ou pipeline. Bien que chaque réseau dispose de son propre contrôleur, l'aspect contrôle global de l'ensemble de la machine n'introduit pas de problème majeur.

3 Conclusion et perspectives

Les schémas de compression d'images actuels ont les bonnes propriétés qui permettent d'envisager des solutions hautement parallèles et performantes comme des architectures SIMD, réalisées à partir d'un processeur élémentaire spécialisé et programmable. Ce modèle d'architecture, couplé à un environnement de programmation efficace, constitue un outil essentiel pour l'émulation en temps réel d'applications de compression, étape indispensable avant toute mise en œuvre à l'aide de circuits spécifiques.

L'architecture interne du processeur élémentaire est en cours de définition. Les limitations du modèle totalement SIMD sont analysées et un enrichissement du modèle est étudié en vue d'aller vers des fonctionnements multi-SIMD par sous-réseaux de processeurs.

Bibliographie

[1] Charot (F.). – *Architectures parallèles spécialisées pour l'image.* – Publication interne n° 722, Irisa, avril 1993.

[2] Charot (F.), Frison (P.), Gautrin (E.), Lavenier (D.), Quinton (P.) et Wagner (C.). – *From equations to hardware. Towards the systematic mapping of algorithms onto parallel architectures.* – Publication interne n° 686, IRISA, octobre 1992. Second International Conference on Parallel Image Analysis, Ube, Japon, décembre 1992.

[3] Chin (D.), Passe (J.), Bernard (F.), Taylor (H.) et Knight (S.). – The Princeton Engine : A Real-Time Video System Simulator. *IEEE Transactions on Consumer Electronics*, vol. 34, n° 2, mai 1988, pp. 285–297.

[4] D. Le Gall. – MPEG : A Video Standard for Multimedia Applications. *Communications of the ACM*, vol. 34, n° 4, avril 1991, pp. 46–58.

[5] ETSI. – Specification of component TV codec for 32-45 Mbit/s. – Rapport technique ETSI, décembre 1990.

[6] Lavenier (D.) et Raimbault (F.). – *ReLaCS for Systolic Programming.* – Publication interne n° 726, IRISA, mai 1993.

[7] Tamitani (I.), Harasaki (H.), Nishitani (T.), Endo (Y.), Yamashina (M.) et Enomoto (T.). – A Real-Time Video Signal Processor Suitable for Motion Picture Coding Applications. *IEEE Transactions on Circuits and Systems*, vol. 36, n° 10, octobre 1989, pp. 1259–1266.

[8] Wagner (C.). – *De l'image vers la compression.* – Publication interne n° 737, IRISA, mai 1993.