Intégration rapide de services vidéo Mpeg sur architectures parallèles

Jean-François Nezan, Olivier Déforges, Virginie Fresse

Laboratoire ARTIST/INSA de Rennes

20, av des buttes de coësmes, 35043 Rennes Cedex

jnezan@insa-rennes.fr

Résumé - Le temps réel pour des applications audiovisuelles est une forte contrainte qui nécessite la mise en œuvre de plates-formes constituées de plusieurs unités de calcul. Le but de nos travaux est de développer un processus de prototypage rapide sur des architectures parallèles pour des applications de traitement d'image. Le processus de prototypage débute par la description des algorithmes grâce à une interface visuelle de programmation orientée objet. Cette description est ensuite transformée automatiquement pour pouvoir être utilisée par Syndex, un logiciel permettant d'évaluer et de générer l'ordonnancement des tâches de l'algorithme sur des architectures multiprocesseurs. Nous démontrons ici l'efficacité de notre méthodologie avec les développement d'une application Mpeg-2 conséquente et son implantation multi-DSP.

Abstract *– Real time audio-visual applications have very important time constraints, involving the use of several numerical calculation units. The aim of our work is to develop a rapid prototyping process dedicated to parallel architectures for image applications. The front-end of our prototyping process is an object-oriented visual programming interface for the description of the algorithms. This description is then automatically transformed to be compliant with Syndex entrance, a CAD software which evaluates and generates the tasks scheduling over the target multiprocessor architecture. We demonstrate here the efficiency of our methodology with the development of a consequent Mpeg-2 application and its multi-DSP implementation.*

1. Introduction

Le « Moving Picture Experts Group » (MPEG) est le groupe de travail ISO/IEC chargé du développement des techniques de compression, décompression, traitement et représentation des documents audio et vidéo. Les solutions disponibles pour les codages/décodages sont souvent des programmes logiciels pour processeurs standards, de par leur développement rapide. Cela permet de réaliser des applications comme la sauvegarde de données, manipulation de documents audio. Cependant, les outils mis à disposition dans les normes Mpeg permettent beaucoup plus : la manipulation de vidéo et audio mixées, l' interactivité entre autres. Les contraintes temps réel fortes des ces outils rendent nécessaire l'utilisation de technologies dédiées. Des composants spécifiques, disponibles sur le marché, permettent d'optimiser des calculs comme les transformations DCT, DCT inverse, estimation de mouvement et compensation. Mais leur intégration sur une plate-forme nécessite plusieurs personnes compétentes, spécialement pour le développement de plates-formes mixtes, constituées de processeurs et de composants matériels spécifiques. Généralement, ces plates-formes ne sont utilisées que pour une unique application.

Mpeg-4 [2] combine les algorithmes des autres standards Mpeg avec de nouveaux pour répondre aux besoins des futures applications multimédias. Actuellement, un composant incluant l'ensemble des outils Mpeg-4 ne peut être réalisé de part sa complexité [3]. De plus, une plate-forme peut avoir à passer d'un outil à un autre, ce qui nécessite des cibles programmables. Ces faits nous ont amené à développer une méthodologie nécessitant comme unique point d'entrée la description fonctionnelle de l'application. L'objectif a été d'offrir au concepteur la possibilité d'implanter son application de manière automatique dans une architecture complexe, sans qu'aucun pré-requis ne lui soit nécessaire. Une conséquence est de pouvoir porter de telles applications sur différentes plates-formes dédiées. Ces travaux ont été détaillés dans [1].

Nous présentons ici notre méthodologie appliquée sur application conséquente : le codage Mpeg-2 d'une séquence vidéo sur un architecture multi-DSP. Le problème est une pré-étape dans le cadre de l'intégration de services Mpeg-4.

2. Méthodologie de prototypage rapide

Nous décrivons ici la méthodologie [1] utilisée pour l'implémentation d'architectures parallèles homogènes, constituées de processeurs usuels (DSP).

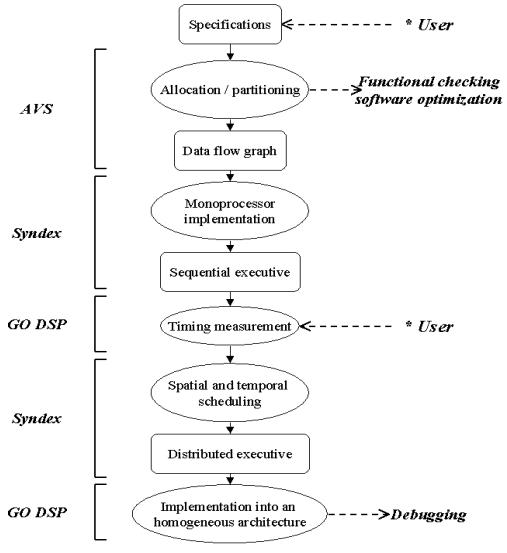

Fig 1 : Méthodologie automatique pour prototypage rapide d'architectures parallèles

Le point de départ du processus (fig 1) est la description fonctionnelle de l'application faite sur le logiciel de développement AVS [4]. AVS permet de définir des modules que l'on peut connecter entre eux afin de générer une application complète. Les outils de visualisation d'AVS permettent de développer et vérifier les modules indépendamment. Les modules peuvent être stockés dans des bibliothèques spécifiques pour de possibles re-utilisations.

La description fonctionnelle faite sous AVS est ensuite automatiquement traduite en un fichier d'entrée pour le logiciel Syndex [5], qui va optimiser la distribution des différentes taches de l'application sur les processeurs disponibles grâce à la méthode AAA (Adéquation Algorithme Architecture).

Une fois la description réalisée sous AVS, la seule intervention de l'utilisateur au cours du processus est la mesure temporelle. Cette étape consiste à déterminer la durée passée dans chaque fonction. Une méthode est de réaliser le chronométrage sur une implantation monoprocesseur, ce qui est possible grâce aux outils de débogage de GODSP : Code Composer. L'utilisateur peut facilement reporter ces temps mesurés dans le graphe du logiciel Syndex, pour avoir un placement optimisé.

3. Mpeg

3.1 L'organisation en profiles@levels

Depuis 1988, trois standards ont été développés : Mpeg-1, 2 et 4. Mpeg-7 et 21 sont en cours de spécification. Pour chacun d'eux, une multitude de schémas sont définis afin d'optimiser le codage en fonction des débits autorisés, de la complexité du document et de la qualité requise. Ces différents schémas sont organisés en profiles@levels.

Un profile est un sous ensemble précis dans la syntaxe du flux de bit Mpeg. Par exemple, Mpeg-4 est constitué de différents profiles traitant de la vidéo, de l'audio ou de la partie système (tout ce qui concerne le contrôle du flux) [3]. Pour chaque profile, les performances nécessaires en terme de codage ou de décodage sont très variables. Ainsi, chaque profile est divisé en niveaux (« levels »). Un niveau est un jeu de contraintes (nombre d'objets maximal, intervalle de débits …) imposé au flux Mpeg. Pour chaque application, le développeur doit mettre en œuvre un jeu de ces profile@levels et non la totalité de la norme.

3.2 Mpeg, partie vidéo

Notre centre d'intérêt se situe dans la partie vidéo de la norme. Cette partie (« visual part ») est constituée de 9 profiles dans Mpeg-4. Avec les niveaux associés, on obtient 23 manières différentes de coder et de décoder une séquence d'images. Il faut ajouter à cela les 13 schémas Mpeg-2.

Cependant, de fortes similarités entre ces schémas existent : le principe de la prédiction de mouvement d'une image à l'autre afin d'exploiter au maximum les redondances temporelles, par exemple. Mpeg fournit trois modes de codage : le codage intra (images I), le codage inter ou prédictif (images P) et codage bi-directionnel (images B). Une image I est codée indépendamment des autres images, une image P est codée à partir de la prédiction de cette image faite à l'aide d'une image précédemment codée, et une image B est codé à partir d' une image passée et d' une image future dans la séquence (fig 5). L'estimation de mouvement est réalisée à partir de la luminance (composante N&B d'une image), sur des blocs 16x16, ce qui permet un codage efficace ainsi que la resynchronisation du décodage en cas d'erreur.

Les codages Mpeg utilisent toujours des macroblocs, un macrobloc étant constitué d'un bloc 16*16 sur l'image des luminances, ainsi que des blocs 8*8 des composantes chromatiques correspondantes (sous-échantillonnées d'un facteur deux).

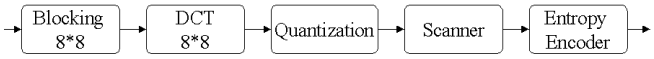

Fig 2 : Schéma de codage pour images I

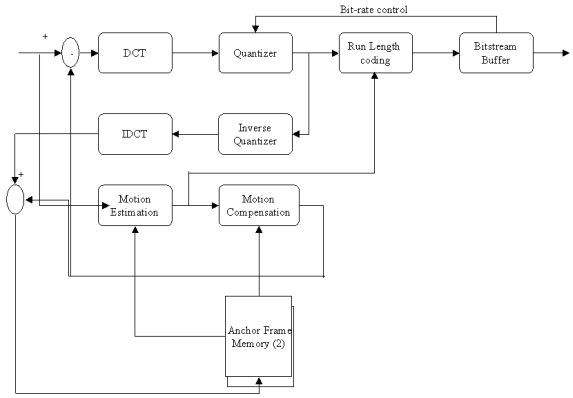

Fig 3 : Schéma de codage prédictif pour images P et B

Ensuite, les algorithmes Mpeg utilisent une DCT (discrete cosine transforms) afin de réduire la redondance spatiale dans les macroblocs. Les coefficients DCT sont calculés directement sur les pixels en mode intra, et sur la différence entre les pixels du macrobloc avec ceux du macrobloc le plus proche (close matching) pour les modes P et B. Après quantification, ces coefficients sont en grande partie nuls, ce qui permet de réaliser un codage entropique. Le résultat est un tableau à deux dimensions, l'une donnant le nombre de zéros consécutifs, l'autre donnant la valeur non nulle suivant la suite de zéros. Le vecteur de mouvement et les valeurs moyennes des coefficients issues des DCT sont également codés.

Mpeg-4 est plus complexe puisque chaque image est décomposée en objets de formes et de natures diverses, mais ces grands principes sont réutilisés pour le codage de la texture. On remarquera que la manière dont doivent être exécutées ces différentes étapes n'est pas spécifié dans le standard, seul le schéma global de décodage doit être respecté. Ainsi, les algorithmes peuvent être optimisés et adaptés aux applications visées. Par exemple, le décodage de forme est très simple pour une image rectangulaire, mais

relativement complexe pour une image de forme arbitraire et variable dans le temps.

3.3 Mpeg et notre méthodologie

Nous créons des librairies vidéo Mpeg constituées de modules et d'applications à l'aide d'AVS. Chaque application est constituée de modules, qui peuvent être utilisés dans plusieurs applications (ex : codage et décodage). Chaque module exécute une routine écrite en langage C. Un module peut également être une macro, constituée de plusieurs autres modules.

La vérification fonctionnelle et l'optimisation de code, des modules ou des applications, sont réalisées à haut niveau. L'optimisation d'un module peut dépendre des contraintes temps réel de l'application, mais aussi de la plate-forme cible.

Ensuite, la traduction automatique et Syndex sont utilisés. Les tâches peuvent alors être réparties entre les diverses unités de calculs, optimisant le temps global d'exécution en fonction de l'architecture décrite sous Syndex.

4. RESULTATS

4.1 Plate-forme matérielle

Nous avons choisis une plate-forme matérielle fournissant une architecture cohérente et modulaire. Notre plate-forme est constituée d'une carte mère Sundance possédant deux modules Texas Instrument (TIM). Le premier module est un module de calcul constitué de deux DSP TMS320C44. Le second est un module d'acquisition et de restitution d'images vidéo, basé sur un DSP TMS320C40. Ce module permet également de réaliser des calculs sur les images.

4.2 Développement des modules et des applications

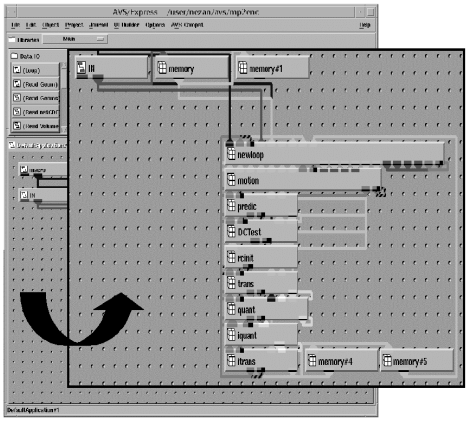

Fig 4 : Environnement de développement AVS

4.2.1 Développement

Nous avons créé des modules sous AVS (fig 4) pour le codage Mpeg-2 de séquences vidéo rectangulaires, au profile principal, niveau principal (main@main). Le codage fonctionne pour toutes les tailles d'image, que ce soit pour un codage Intra ou Prédictif (simple ou bi-directionnel), pour des séquences couleur ou N&B. Les modules développé sont : estimation de mouvement, compensation de mouvement, estimation du type de DCT, DCT, IDCT, quantification inverse, ainsi qu'un module de quantification et VLC (Variable Length Coding).

Nous avons également développé un module permettant de lire un fichier de paramètres, de façon à adapter tous les autre modules au format et au nombre d'images de la séquence, et un module écrivant l'en-tête du fichier codé.

Ensuite, nous avons adapté les modules pour une application adaptée à notre plate-forme, avec une taille d'image fixée à 128*128. Le graphe flot de donnée utilisé en entrée de Syndex version 4 doit être statique : les flots de donné entre les modules doivent être contraints, et toutes les images doivent suivre le même schéma quelque soit leur nature (I, P ou B).

4.2.2 Gestion des buffers

Le codage Mpeg de vidéo naturelles est basé sur l'estimation de mouvement. Le codage des images B nécessite la connaissance des images codées précédemment et successivement dans l'ordre de visualisation (fig 5).

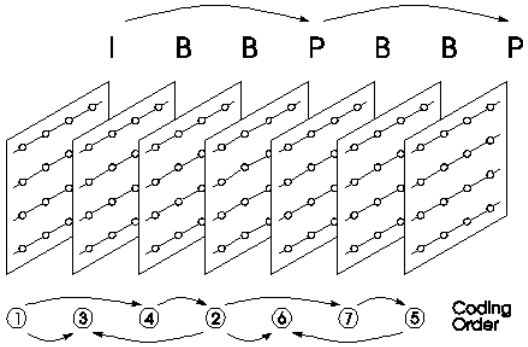

Fig 5 : Structure de la prédiction temporelle

L'ordre de codage est donc différent de l'ordre de visualisation. Il est possible de paramétrer le nombre d'images B entre deux images P. Nous avons choisis une unique image B, ainsi qu'une image I toutes les six images. Dans cette configuration, le codage d'une image implique de stocker les deux images précédemment codées, avant et après codage. Ces images sont stockées dans des modules « memory », le module « newloop » étant nécessaire afin de réorganiser les buffers et remettre à jour les variables avant le codage d'une nouvelles image. Le graphe flot de données correspondant est donné figure 4.

4.3 Implantation monoprocesseur

Nous avons tout d'abord réalisé une implantation monoprocesseur de notre algorithme. Sa description sous Syndex est donnée figure 6. Le temps de chaque tâche a été calculé en prenant la moyenne du codage de plusieurs images dans la séquence. En effet, la complexité des tâche dépend du type I,P ou B de l'image codée, mais également du contenu et de la rapidité des mouvements dans la séquence. De cette manière, le temps moyen de codage d'une

image est évalué à 910 ms, avec la répartition mise en évidence dans la figure 7.

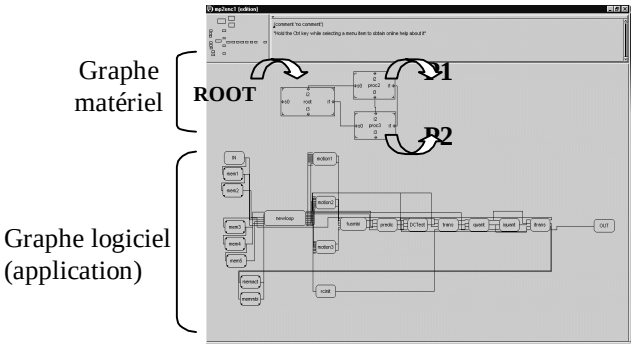

Fig 6 : schéma flot de donné sous Syndex

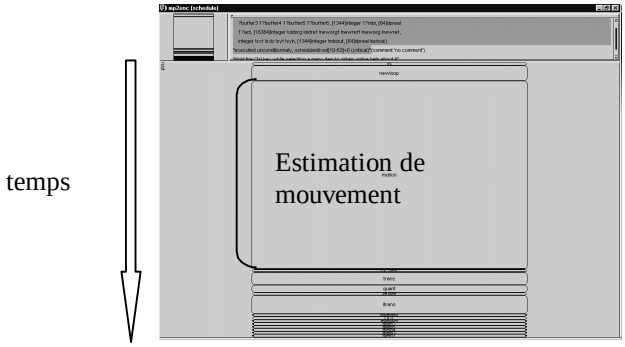

Fig 7 : diagramme de temps dans le cas monoprocesseur

75% du temps de codage est utilisé à l'estimation de mouvement, ce qui rend l'opération de codage bien plus critique de le décodage dans les applications temps réel.

4.4 Implantation Multi-DSP

La majeure partie des tâches doivent être exécutées séquentiellement. Pour partager les calculs entre les processeurs, il faut mettre en évidence les calculs pouvant être exécutés en parallèle dans chacun de nos modules. Généralement, on travaille sur les macroblocs de manière indépendante. Ils peuvent donc être divisés en plusieurs sous-modules en divisant chaque bloc en autant de modules indépendants qu'il n'y a de processeur, chacun travaillant sur une partie de l'image.

Syndex trouve l'adéquation optimale entre l'algorithme et l'architecture. Ainsi, nous pouvons mesurer l'importance des temps de transfert inter-DSP (fig 8). Utiliser trois DSP pour un tel codage permet d'accélérer l'estimation de mouvement d'un facteur 2, soit d'accélérer le codage complet d'un facteur 1.26. Le temps de codage d'une image est passé de 910 ms à 720 ms.

Cette première implantation multi-DSP nous permet d'adapter le nombre de macroblocs traité par chacun d'eux. En effet, de part son accès direct aux données, le processeur root peut traité plus de macroblocs que les processeurs P1 et P2, ce qui conduit à la nouvelle répartition figure 8. Le temps de codage d'une image est maintenant de 620 ms pour 910 ms dans le cas de l'implantation monoprocesseur, soit une accélération de 1.47.

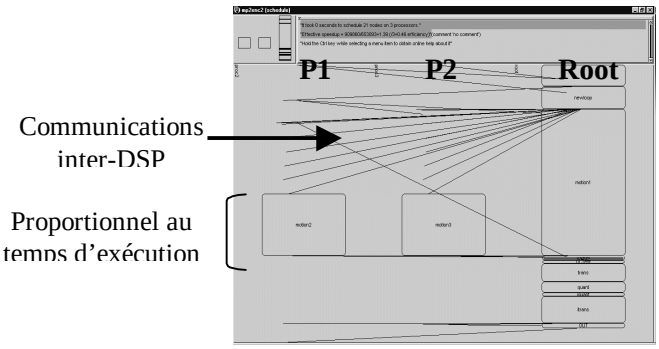

Fig 8 : diagramme de temps optimisé pour une architecture multi-DSP

5. Conclusions et perspectives

Nous avons montré ici l'efficacité de notre méthodologie pour des plates-formes multiprocesseurs. Les modules développés pour notre application Mpeg peuvent être re-réutilisés, avec la possibilité de les adapter et de les connecter entre eux facilement. Les fonctionnalités d'une nouvelle application peuvent être vérifiées à l'aide des outils de visualisation fournis sous AVS.

Pour créer une chaîne de traitement comme une application Mpeg, le processus partant de la description sous AVS, jusqu'à l'exécution sur une architecture multiprocesseur, est réalisé en une vingtaine de minutes. Cette rapidité permet à l'utilisateur de concentrer ses efforts sur les optimisations algorithmiques ou de modifier le partitionnement final.

Notre méthodologie est relativement indépendante de l'architecture cible. Nous travaillons actuellement sur la génération de code automatique pour DSP TMS320C6x à partir de Syndex.

Etant donné que la méthodologie utilisée est cohérente avec celle que nous avons développé pour les architectures mixtes, nous pensons améliorer nos résultats en implantant les opérations élémentaires et régulières (DCT, IDCT) sur la partie matérielle que comporte notre plate-forme.

Références

[1] V. Fresse, M. Assouil, O. Deforges. *Rapid prototyping of image processing onto a multiprocessor architecture*, DSP World ICSPAT, Orlando, Floride, USA, November 1-4 1999.

[2] Signal Processing : Image Communication, *special Mpeg-4*. Publication Elsevier Science B.V, janvier 2000.

[3] C. Miro, A. Lafage, Q.L. Nguyen-Phuc, Y. Mathieu. *Hardware Implementation of Perspective Transformations on Mpeg-4 Video Objects*, SPIE Proceedings, Volume 3655, Media Processors 1999.

[4] International AVS center, Université de Manchester. Site internet : http://www.iavsc.org

[5] C. Lavarenne, Y. Sorel. *Specification, performance optimization and executive generation for real-time embedded multiprocessor application with Syndex*, Proc of Real Time Embedded Processing for Space Applications, CNES International Symposium.