Intégration d'un décodeur Mpeg-4 sur architecture multi-C6x

J.F Nezan, O Deforges, M Raulet
UMR CNRS 6164 IETR / Insa Rennes
20, av des Buttes de Coësmes
CS 14315 35043 RENNES Cedex, France
Tel : +33/0 223 238 459 Mail : jnezan@insa-rennes.fr

Résumé

Mpeg-4 apporte une réponse fonctionnelle pour le codage et la manipulation d'objets audiovisuels naturels et de synthèse. Les futures applications temps-réel manipulant ces types d'objets auront d'importantes contraintes, qui ne pourront être respectées qu'avec l'utilisation d'unités de calculs extrêmement performantes. Des solutions logicielles pour processeur standard commencent à être proposées. Toutefois, elles ne permettent pas une optimisation des ressources en fonction du niveau d'utilisation du décodeur, ni une migration simple sur une architecture parallèle. Nous présentons ici le processus de développement d'un décodeur Mpeg-4, à travers une méthodologie de prototypage et l'implantation résultante sur architecture multi-C6x à partir d'une génération automatique de code.

1 Introduction

Le "Moving Picture Experts Group" (MPEG) [4] est le groupe de travail ISO/IEC chargé du développement des techniques de compression, décompression, traitement et représentation des documents audio et vidéo. Mpeg offre un cadre standardisé pour les applications multimédia, ce qui accélère les temps de mise au point et facilite le développement des produits multimédia. Les futurs produits devront entre autres manipuler des images naturelles et de synthèse, permettre des interactions avec l'utilisateur.

Les solutions logicielles classiques peuvent réaliser du stockage de données, ou bien de la manipulation des documents, mais seront vite limitées pour ces applications, spécialement dans des contextes embarqués et temps-réels. D'autre part, l'utilisation d'un composant spécifique dédié (ASIC ou FPGA) n'apporte pas la flexibilité nécessaire aux applications Mpeg-4. En revanche, le temps-réel peut être atteint avec l'utilisation de plusieurs processeurs programmables. Le choix des DSPs permet d'allier souplesse d'utilisation, reprogrammabilité et performances de pointe pour les opérations classiques de traitement du signal. Nous présentons ici la méthodologie AVSynDEx [1] (concaténation d' AVS et de SynDEx) avec laquelle nous avons développé un décodeur temps-réel Mpeg-4 à un haut niveau d'abstraction, puis son implantation quasi automatique sur une architecture constituée de deux DSPs TMS320C6x.

Après une introduction des spécificités des algorithmes Mpeg-4 dans la partie 2, le chapitre suivant décrira la méthodologie AVSynDEx. Les résultats obtenus sur le décodage d'objets naturels vidéo Mpeg-4 seront ensuite détaillés chapitre 4. Les conclusions et perspectives feront l'objet de la partie 5.

2 Codage d'objets audiovisuels avec MPEG-4

Depuis 1988, le groupe Mpeg a développé 3 standards (Mpeg-1, 2 et 4). Mpeg-4 a vu le jour de part le succès grandissant de la télévision numérique, du multimédia et des applications graphiques interactives. Ce nouveau format apporte les éléments techniques de standardisation pour le développement de ces applications nouvelle génération.

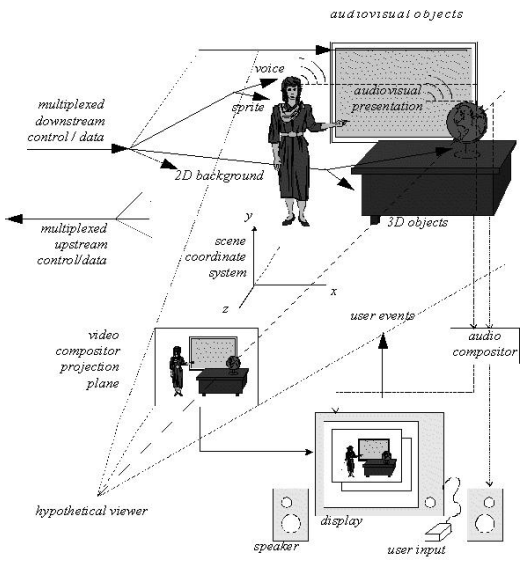

FIG. 1 – Exemple de scène audiovisuelle.

Un document Mpeg-4 est une scène constituée d'objets audiovisuels avec lesquels l'utilisateur peut interagir. La composition de ces objets et leurs ca-

ractéristiques dans la scène sont décrites dans le flux Mpeg-4 (fig.1). Ensuite, les données codées de chaque objet sont multiplexées et synchronisées, de manière à être envoyée sur un canal de transmission avec une qualité de service appropriée.

2.1 l'organisation du standard en profils et niveaux

Les schémas de codage sont définis en fonction du débit de l'application, de la complexité et de l'utilisation du document. Les logiciels produits à l'heure actuelle tentent d'intégrer l'ensemble des services, alors que seule une partie n'est véritablement utilisée. La minimisation des resources, entre autres de la mémoire, et l'amélioration des performances qui en dépend, reposent sur l'intégration des fonctionnalités strictement nécessaires. Mpeg-4 a été créé dans cette optique, avec une division en profils (profiles), eux mêmes divisés en niveaux (levels). Le standard est alors générique, couvrant une grande plage d'applications, débits, résolutions, qualités et services.

Le développeur d'une nouvelle application devra donc sélectionner un jeu des ces profils et niveaux, et ne pas utiliser toute la norme. De même, la manière de réaliser chaque étape n'est pas décrite dans le standard, seuls les schémas globaux de décodage sont normalisés. Les méthodes peuvent donc être optimisées et adaptées au contexte (rapidité, taille du code ou des ressources). L'utilisation de librairies Mpeg-4 constituées de modules pouvant être inter-connectés dans un graphe flot de donnée décrivant l'application complète est bien adaptée à cette organisation de la norme. Nous montrerons que cette description fonctionnelle est aussi le point d'entrée pour notre processus de portage.

2.2 Mpeg-4 part 4 : tests de conformité

Mpeg-4 partie 1 (system), 2 (visual) et 3 (audio) donnent les moyen de coder et de représenter les information audiovisuelles. La flexibilité est obtenue par l'inclusion de paramètres dans le bitstream qui définissent ses caractéristiques (taille des images par exemple). La quatrième partie de la norme spécifie comment les tests doivent être effectués afin de savoir si un décodeur ou les flux de bits créés par un codeur sont conformes au standard. Cette partie permet de tester chaque sous-partie de Mpeg-4. Les graphes décrivant les algorithmes Mpeg-4 sont divisés en parties pouvant être testées indépendamment grâce aux tests de conformité adéquats. La vérification fonctionnelle d'une application complète peut ainsi être réalisée de manière progressive.

3 Méthodologie de prototypage AVSynDEx

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

Le point de départ du processus est la description fonctionnelle de l'application faite sur le logiciel de développement AVS (fig.2). Ce logiciel permet de définir des modules que l'on peut interconnecter afin de générer une application complète sous forme d'un graphe flot de données. L'activation d'un module correspond à l'exécution d'une fonction C associée. Les outils de visualisation d'AVS sont utilisés pour développer et vérifier les modules indépendamment. Les modules peuvent être ensuite stockés dans des librairies spécifiques pour de possibles réutilisations.

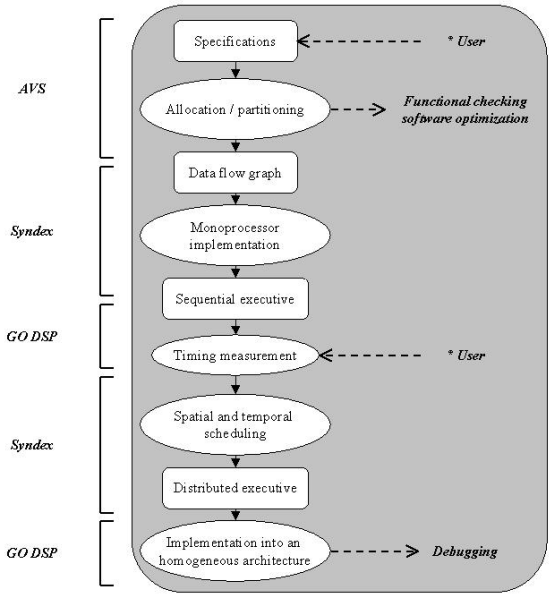

FIG. 2 – Methodologie AVSynDEx.

La description fonctionnelle faite sous AVS est ensuite automatiquement traduite en un fichier d'entrée pour le logiciel Syndex [3], développé par l'INRIA Rocquencourt (France). Celui-ci va optimiser la distribution des différentes taches de l'application sur les processeurs disponibles grâce à la méthode AAA (Adéquation Algorithme Architecture). Dans sa dernière version, il est possible de décrire des algorithmes avec hiérarchie et conditionnement. SynDEx génère un exécutif temps-réel minimum garantissant les ordres d'exécution et évitant les interblocages. L'utilisateur peut choisir d'y insérer ou non des primitives de chronométrage. Nous avons développé le noyau d'exécutif SynDEx pour DSPs Texas Instruments C6x [2].

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 avec un code monoprocesseur comportant les primitives de chronométrage généré par SynDEx. L'utilisateur peut facilement reporter les temps mesurés dans le graphe du logiciel Syndex, pour qu'il génère un placement optimisé des tâches sur les deux DSPs.

4 Prototypage d'un décodeur vidéo Mpeg-4

4.1 Platforme 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). Chaque module est constitué d'un DSP Texas Instruments TMS320C6201 de fréquence horloge à 200 MHz.

Un FPGA sur chaque module gère six ports de communications (CP, 20 MB/s) et deux bus SDB (Sundance Digital Bus, 200 MB/s chacun), ce qui permet les transferts de données entre les unités de calculs de la plateforme. Ces forts débits sont indispensables aux transferts rapides des images et données de taille importante rencontrées dans le codage vidéo.

4.2 Librairies de modules et applications

Les modules élémentaires de décodage de texture vidéo Mpeg-4 ont été entièrement réécrits sous AVS, puis stockés dans des librairies. Chaque application (codage et décodage par exemple) peut ensuite être construite à l'aide de ces modules.

Dans les algorithmes Mpeg-4, les images sont divisées en macroblocs, c'est-à-dire un bloc 16*16 de luminance (organisé en quatre blocs 8*8) et deux blocs 8*8 pour les chrominances associées (fig.3). C'est le niveau de granularité adopté dans la norme pour expliciter les traitements. C'est aussi celui que nous avons adopté car il correspond au grain le plus fin pour l'expression de l'ensemble des dépendances des opérateurs et des données. Le décodeur est donc décrit de manière hiérarchique, en utilisant la répétition de l'algorithme sur l'ensemble des macroblocs de l'image.

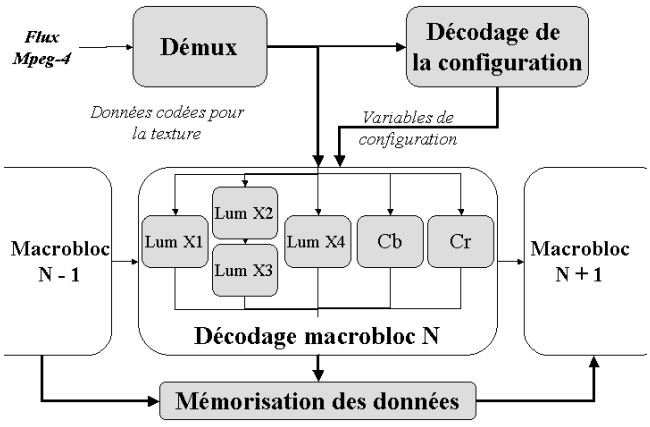

FIG. 3 – schéma hiérarchique d'un décodeur Mpeg-4.

Le flux Mpeg-4 est d'abord démultiplexé afin de séparer les données de chaque macrobloc, déterminant leur ordre de décodage. Une première bibliothèque contenant des modules de lecture et de manipulation de flux Mpeg-4 permet de récupérer un flux, puis de séparer les informations codées de configuration et les flux élémentaires de donnée.

Ensuite, chaque bloc d'un macrobloc est décodé selon le schéma fig.4. Les modules permettant de réaliser les opérations de décodage élémentaires (VLC inverse, direction de prédiction DC, scan inverse, prediction des coefficients DC et AC, quantification et DCT inverses) ont été regroupés dans une bibliothèque AVS. Un module peut être construit comme une macro constituée de plusieurs modules inter-connectés (description hierarchique). Chacun d'eux a été optimisé pour le décodage de macroblocs intra, et pour une implantation sur architecture VLIW : les boucles entrelacées ainsi que les tests conditionnels amenant des ruptures de pipeline, les allocations dynamiques et manipulations de fichiers sont évités.

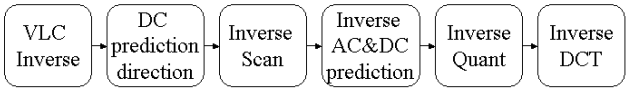

FIG. 4 – décodage d'un bloc 8*8 Mpeg-4.

Chaque bloc est prédit à partir d'un de ses proches voisins, impliquant un ordre fixe de décodage à l'intérieur de chaque macrobloc, mais aussi entre les macroblocs de l'image. Des données doivent alors être stockées dans des modules de mémorisation (fig.3).

Les librairies Mpeg-4 peuvent être utilisées dans toute application de codage et de décodage de textures intra, c'est-à-dire dans la majorité des profiles. Les fonctionalités ont été vérifiées grâce aux flux spécifiques fournis dans les tests de conformité du standard.

4.3 Implémentation monoprocesseur

Les modules AVS sont dans un premier temps implantés sur un unique DSP C6201, grâce à la génération automatique de code de SynDEx version 5 [2]. Comme le graphe de notre application utilise les fonctionalités de la version 6 (répétition, hiérarchie et conditionnement), le code de l'application complète a fait l'objet de quelques optimisations manuelles afin de pouvoir être implanté totalement en mémoire interne du C6x. En revanche, les images sont des données trop importantes et ont du être placées en mémoire externes du C6x. Dans ces conditions, les résultats numériques fig.5 ont été mesurés pour chaque opération élémentaire. Nous obtenons le temps-réel pour une image QCIF (176*144 pixels) avec un temps de 35,5 ms par image.

4.4 Implémentation multi-DSPs

L'application ne pourra être complètement vérifiée sur la plateforme que lorsque le générateur de code de SynDEx pour C6x prendra en compte toutes les fonctionalités de la version 6. Il est déjà possible de simuler l'implantation sur deux DSP en reportant les temps

Functions	Times (microsec)
MacroblockI()	3,2
Inverse VLC	171
DC prediction direction	0,7 * 6
Inverse scan	3,8 * 6
Inverse AC&DC prediction	7,4 * 6
Inverse Quant	11,1 * 6
Inverse DCT	1,8 * 6

FIG. 5 – Chronométrages réalisés sur un C6201.

chronométrés dans le graphe de l'application sous SynDEx. L'adéquation peut alors nous fournir un partitionnement optimisé de l'algorithme sur les deux DSPs de notre plateforme. Le diagramme temporel fournis par SynDEx (fig.6)permet de voir que les opérations sont bien partagées sur les deux DSP, avec un temps de décodage pour une image QCIF de 22,6 ms, soit une accélération de 1,57.

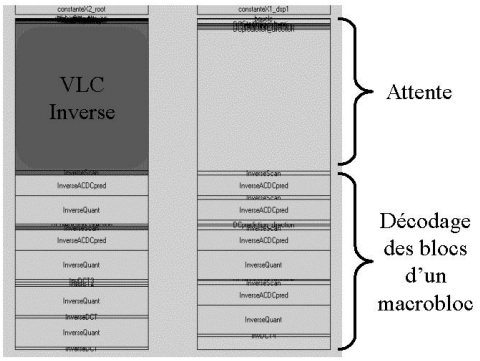

FIG. 6 – diagramme temporel SynDEx pour le décodeur Mpeg-4.

On peut facilement observer sur le graphe temporel de SynDEx (fig.6) que le second DSP doit attendre la fin du VLC inverse exécuté sur le premier. La description de cette opération au niveau bloc permettrait de diminuer cette attente de manière importante, et de permettre de se rapprocher d'une accéleration de facteur 2.

5 Conclusions et perspectives

Ce papier montre l'efficacité de la méthodologie AAA-SynDEx pour plateformes multi-DSPs. Les modules créés dans le décodeur Mpeg-4 présenté peuvent être reutilisés dans de futurs applications, avec la possibilité de les adapter et de les connecter entre eux à un haut niveau d'abstraction, avec l'aide des outils de visualisation interactifs, graphiques et des possibilités de debug offerts par AVS. Nous envisageons de les utiliser pour le codage Mpeg-4 de vidéo naturelle et de synthèse.

Le portage d'un processus de traitement numérique des images, après description sous AVS, est pratiquement automatique. Le faible temps d'implantation permet à l'utilisateur de se concentrer sur les fonctionalités de son algorithme, ou de modifier manuellement le partitionnement final. L'utilisateur n'a pas besoin de connaissances spécifiques concernant la synchronisation des taches, les chronométrages ou les transferts de données.

Chaque C6x de notre plateforme est relié à un FPGA qui gère les communications. L'implantation sur ces FPGA des calculs élémentaires répétitifs comme le VLC inverse, apporterait un gain certain. Étant donné que notre méthodologie est cohérente avec celle développée pour architectures mixtes (DSP + FPGA), nous prévoyons l'implantation de ces opérations sur la partie matérielle de notre plateforme.

Le positionnement en mémoire externe des données et le temps qui en résulte ont été des choix a posteriori. SynDEx ne permet pas pour l'instant de considérer plusieurs temps de traitement pour une tâche en fonction du type de mémoire utilisée pour le code et les données. Cette problématique fera l'objet de prochains travaux de recherche dans le cadre d'une thèse.

Références

[1] J.F.Nezan, V.Fresse, O.Deforges *"Fast prototyping of parallel architectures : an Mpeg-2 coding application"*, CISST'2001 proceedings, Las Vegas, USA, Juin 01.

[2] Y.Le Mener, M.Raulet, J.F.Nezan, A.Kountouris, C.Moy *"SynDEx executive kernel development for DSPs TI C6X applied to real-time and embedded multiprocessors architectures"*, Eusipco'2002 proceedings, Toulouse, France, Septembre 02.

[3] T.Grandpierre, C.Lavarenne, Y.Sorel, *"Optimized rapid prototyping for real time embedded heterogeneous multi-processors"*, 7th International workshop on Hardware/Software Co-Design, pages 74-78, Rome, Italie, Mai 1999. IEEE Computer Society, ACM SIGSOFT, IFIP.

[4] *"International Organization For Standardization, ISO/IEC JTC1/SC29/WG11, Coding Of Moving Pictures And Audio Documents"*, http ://www.cselt.it/mpeg.