Synthèse du Module Matériel dans une Conception Mixte Logiciel/Matériel: Application à l'Implantation d'un Algorithme d'Annulation d'Echo Acoustique GMDF alpha

A. Baganne, J. L. Philippe, E. Martin

LESTER, (Laboratoire d'Electronique des Systèmes TEmps-Réel),
2 rue Le coat Saint Haouen, 56100 Lorient
E-mail: nom@univ-ubs.fr , http://lester.univ-ubs.fr:8080/

RÉSUMÉ

Cet article décrit une méthodologie d'implantation mixte Logiciel/Matériel des algorithmes de TDSI. L'architecture cible est composée d'un DSP TMS320C4O (module logiciel) et d'un ASIC (module matériel). Un soin particulier a été accordé au processus de synthèse du module matériel à l'aide de l'outil de synthèse architecturale GAUT. La méthode de synthèse permet la conception de l'ASIC sous contraintes de temps d'exécution, de coûts (surface, consommation) et de contraintes fonctionnelles de communication d'E/S obtenues après partitionnement et synthèse du module logiciel. Cette méthode est illustrée par une implantation mixte de l'algorithme GMDF alpha, un algorithme de filtrage adaptatif pour l'annulation d'écho acoustique.

ABSTRACT

This paper presents a codesign methodology for mixed Hardware/software implementation of Digital Signal Processing and Image applications. The target architecture is based on DSP TMS320C4O (software module) and ASIC (hardware module). We focus on the hardware module design with a High Level Synthesis tool GAUT. The proposed method allows the ASIC design under a set of constraints such as execution time, area and power consumption costs, and I/O communication constraints defined after the partitionning step. As an illustration, we present a mixed implementation of the GMDF alpha algorithm, an adaptive filter well suited to acoustic echo cancellation, on both ASIC and TS320C40 DSP.

1. Introduction

L'augmentation de la complexité des applications de traitements du signal et e l'image (TDSI) nécessite de plus en plus l'utilisation d'architectures parallèles. Le temps de réponse imposé par le traitement est un problème important auquel est confronté le concepteur des systèmes électroniques. Ce problème peut être résolu par l'utilisation d'une implantation mixte composée d'un module logiciel (ML: TMS320C40, Transputer, ...) et d'un module matériel (MM : ASIC, FPGA, ...). Pour obtenir une architecture optimisée, le concepteur doit résoudre des problèmes divers tels que le choix du type de processeur, l'optimisation du partitionnement en modules MM et ML, l'ordonnancement des exécutions intra-module (MM ou ML) et inter-module (synchronisation et synthèse des communications).

Dans cet article, nous présentons une méthodologie d'implantation mixte Logiciel/Matériel des, algorithmes de TDSI. De telles applications sont généralement caractérisées par un paramètre T_0 appelé période d'itération, les calculs étant renouvelés à chaque itération. Nous nous intéressons tout particulièrement à la synthèse du module matériel (représenté par un ASIC) ainsi que les unités de communications entre modules MM et ML sous contraintes définies après partitionnement de l'algorithme. La conception de l'Asic est réalisée au moyen d'un outil de synthèse architecturale GAUT[2]. La synthèse du module Logiciel est assurée par des outils CAD tels que SynDex[3] Nous illustrons notre méthode par une implantation mixte d'un algorithme de filtrage adaptatif GMDFα. La partie 2 décrit notre approche de conception mixte Logiciel/matériel et de synthèse du module matériel. La partie 3 décrit brièvement l'algorithme GMDFα et développe son implantation mixte.

2. Conception mixte logiciel/Matériel :

La figure 1 illustre le flot de conception mixte Logiciel/Matériel. L'architecture hétérogène cible est composée d'un Asic (composant matériel) et d'un processeur à usage général (composant logiciel). Le flot de conception s'effectue en plusieurs étapes et est itéré jusqu'à l'obtention de résultats satisfaisants.

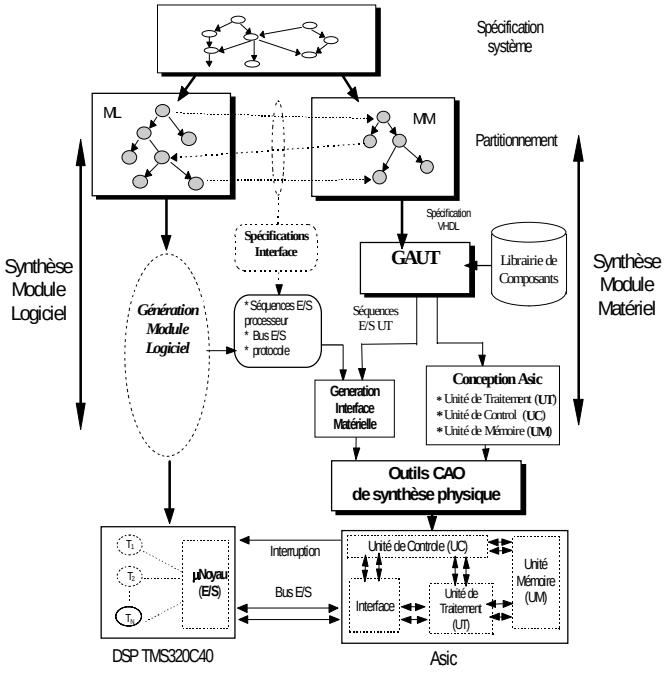

Figure 1 Flot de conception mixte

Les principales étapes qui interviennent dans notre méthodologie sont les suivantes: spécification des besoins du système et sa modélisation, partitionnement Logiciel/Matériel, synthèse des composantes matérielle et logicielle, synthèse de l'interface, cosimulation et intégration [1]. Dans ce qui suit, nous décrirons les principales étapes de notre approche de conception mixte

2.1 Spécification et partitionnement

Tout d'abord, l'algorithme de TDSI est initialement spécifié au niveau comportemental à l'aide d'un langage de spécification tel que SDL, Statecharts, C++, etc. Ensuite, l'algorithme est découpé en deux composantes logicielle (ML) et matérielle (MM). Ce partitionnement peut être réalisé manuellement ou à l'aide de certaines techniques [1] qui tiennent en compte des critères de coûts en temps d'exécution, de surface ou de consommation. L'objectif consiste à l'obtention d'une architecture hétérogène à faible coût et qui satisfait les contraintes d'exécution de l'application. La figure 2 représente un Graphe Flot de Donnée Mixte (GDFM) obtenu après la phase de partitionnement. Il est composé de deux sous-graphes : Logiciel $\mathbf{G_L(V_L,E_L,T_L, C_L)}$ et Matériel $\mathbf{G_M(V_M,E_M,T_M ,C_M)}$ où :

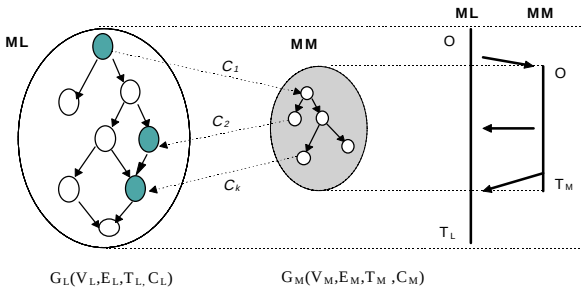

Figure 2 GFDM partitionné

- *Ei et Vi représentent respectivement l'ensemble des noeuds (ou taches) et arcs (ou liens) de chaque partition.*
- *T_M et T_L représentent les temps d'exécution estimés associés à chaque module.*
- *C_M et C_L représentent les coûts d'implantation estimés de chaque module, durant la phase de partitionnement.*

Notons que les coûts sont souvent exprimés en temps de cycles d'exécutions, pour les modules logiciel, et en surface et consommation pour le module matériel. Le graphe GFDM permet la spécification simultanée des deux partitions MM et ML ainsi que le volume des données échangées intra et inter-modules. Le transfert des données inter-modules est décrit par un ensemble de canaux de communications Logiciel/Matériel ***C ={Ci}*** . Chaque canal est caractérisé par un noeud source , un noeud destination et le type de donnée transférée (scalaire ou vectorielle).

2.2 Synthèse du module logiciel

La synthèse de la logicielle consiste à la conversion du graphe $\mathbf{G_L}$ en un Exécutif Temps Réel (ETR) configuré pour la machine matérielle cible (cartes multiprocesseurs, coeur de DSPs, etc...). Cet ETR peut être généré manuellement ou à l'aide d'outils automatiques tels que SynDex[3], COSSAP, etc. Dans notre approche, nous considérons que la composante logicielle a été déjà générée et que la gestion des communications avec le module matériel est assurée par un micro-noyau d'E/S . En outre, nous supposons que chaque support de communication Matériel/Logiciel est spécifié à un très bas niveau (séquences de transferts de données d'E/S et leurs spécifications temporelles).

2.3 Synthèse du module matériel

Il s'agit de concevoir le module matériel (ASIC) à partir du graphe GFDM. Les tâches destinées à être implantées sur l'Asic (sous-graphe $\mathbf{G_M}$) sont spécifiées en VHDL comportemental [2]. La conception de l'Asic est réalisée au moyen d'un outil de synthèse architecture GAUT[2] et son architecture est basée sur les quatre unités fonctionnelles suivantes (figure 3): l'unité de traitement UT, l'unité de contrôle UC, l'unité de mémorisation UM et l'unité de communication (ou *interface*) UCOM. L'outil GAUT permet la synthèse de haut niveau du graphe matériel sous contraintes de temps d'exécution T_M et de coût C_M. Afin de garantir la contrainte d'exécution T_M sur le module matériel, la conception de l'Asic débute par la synthèse de l'unité de traitement. Dans les paragraphes suivants, nous allons décrire le processus de synthèse de chaque unité fonctionnelle de l'Asic.

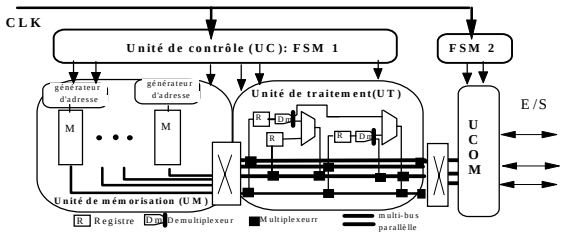

Figure 3 Architecture générique de l'Asic

Unité de traitement : la topologie de l'unité de traitement (figure 3) est basée sur des cellules élémentaires incluant un opérateur, des registres et des composants d'interconnexion optimisés pour le traitement à effectuer. Les cellules communiquent à travers un réseau multibus parallèle. Tout d'abord, GAUT effectue la tâche de sélection de composants, puis exécute simultanément les taches d'ordonnancement et d'assignation avant d'optimiser globalement l'architecture générée. Un module de sélection choisit le nombre et le type d'unités fonctionnelles à partir d'une librairie technologique donnée. Outre la description structurelle de l'UT, la conception fournit, dans une séquence de transferts, les dates de production et de consommation des données de l'ensemble des données nécessaire aux traitement. Cette séquence de transferts constitue un point d'entrée pour la synthèse des unités de mémorisation et de communication. Pour plus d'information, le lecteur pourra consulter [9].

Unité de mémorisation : Le modèle générique de l'unité de mémorisation est composé de bancs mémoires associés à des générateurs d'adresses. Les mémoires peuvent être internes ou externes à l'Asic en fonction de la surface disponible après synthèse de l'UT. La démarche de synthèse de l'UM est décomposable en trois grandes phases qui sont : la distribution des données dans les bancs, le placement des données dans chaque banc et la génération des adresses. La résolution de ces trois phases conduit à un problème *NP* complet. Afin de réduire l'espace de recherche, une approche hiérarchique de conception a été proposée[4].

Unité de contrôle : L'unité de contrôle est synthétisée sous forme d'une machine à états finis. Sa description est spécifiée au format VHDL qui est un point d'entrée aux outils de synthèse logique du commerce afin d'obtenir le plan de masse du circuit.

Unité de communication (interface d'E/S): La synthèse de l'unités de communication est une étape très importante dans le processus de conception du module matériel car elle doit d'une part assurer la cohérence des données échangées *(synchronisation)*

et d'autre part le respect des contraintes temporelles définies après *partitionnement Logiciel/Matériel* (temps d'exécution sur chaque partition, dates et ordre des transferts etc.).

Comme nous l'avons mentionné précédemment, l'interface de communication est synthétisée après l'unité de traitement. Par conséquent, sa conception doit tenir compte de deux types de contraintes :

1. Contraintes fonctionnelles issues de la synthèse de l'UT (que nous noterons **CUT**): dates de production et de consommation des données échangées avec le composant logiciel
2. Contraintes d'entrées/sorties (que nous noterons **CES**) sur les séquences de transferts d'E/S entre l'Asic et le processeur (dates et ordre des transferts de données) spécifiées durant la phase de génération de code du module logiciel.

Nous avons proposé en [6] une méthodologie de synthèse des interfaces matérielles qui offre :

- *Une modélisation des données de communication:* nous avons proposé un modèle global des transactions entre les deux composants sous forme d'un graphe de communication. Ce graphe permet la description des dépendances des données transférées sur les bus d'E/S entre l'Asic et le processeur, ainsi que leurs caractéristiques temporelles (dates de transferts, délai de transfert, contraintes mutuelles etc..)
- *Un modèle générique d'interface :* Nous avons proposé une méthode interactive de synthèse des interfaces de communication qui s'appuie sur un modèle architectural générique[6]. Ce modèle (figure 4) est composé de *modules de* bufferisation (FIFOs, LIFOs et Registres) et d'un *automate.*

La synthèse des modules de bufferisation passe par la transformation formelle des séquences d'E/S spécifiées. Nous avons pour cela défini trois classes de transformation : Tdemux qui sépare une séquence d'E/S complexe en plusieurs séquences élémentaires, Tmem qui mémorise les données et Tmux qui fusionne plusieurs séquences élémentaires en une seule. Tmem permet de cibler différents composants de bufferisation (fifo, registres, …) et est applicable sous certaines conditions (propriétés) à vérifier.

L'automate ; à base d'une machine à états finis, effectue l'implantation des protocoles de communications d'E/S et génère les séquences de contrôle pour chaque élément de l'interface.

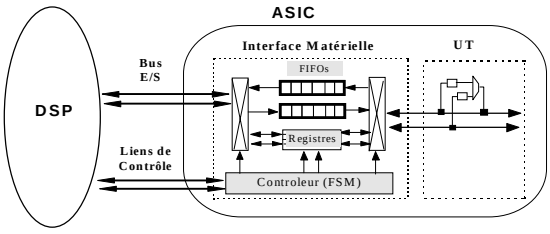

Figure 4 Modèle générique de l'interface matérielle

3. Application : Implantation du GMDFα

3.1 Algorithme GMDF alpha :

L'algorithme de filtrage adaptatif appelé Generalised Multi Delay Adaptive Filter α (GMDFα) [7] peut être considéré comme une formulation par blocs du filtre NLMS (Normalised Least Mean Square) avec adaptation dans le domaine fréquentiel. Il propose une segmentation de la réponse impulsionnelle (longue en téléconférence) en petits blocs et permet ainsi la réduction du retard de traitement. Cet algorithme bénéficie d'un parallélisme potentiel important, dépendant des paramètres de la décomposition de la réponse impulsionnelle et de la taille du filtre [7]. Le filtre adaptatif de longueur L est découpé en K segments de longueur N (L = K.N), ce qui réduit le temps de traitement. R nouveaux échantillons sont traités à chaque itération de l'algorithme et les coefficients de ce filtre sont mis à jour α (facteur de suréchantillonage) fois par bloc (R = N/α). La figure 5 représente le diagramme flot de donnée de l'algorithme. A chaque itération, l'algorithme exécute deux transformées FFT et une transformée inverse. La complexité algorithmique du GMDFα en version simplifiée est de $N[10K+3\log_2(N)-3]$ multiplications et de $N[8K+9\log_2(N)-10]$ additions par itération.

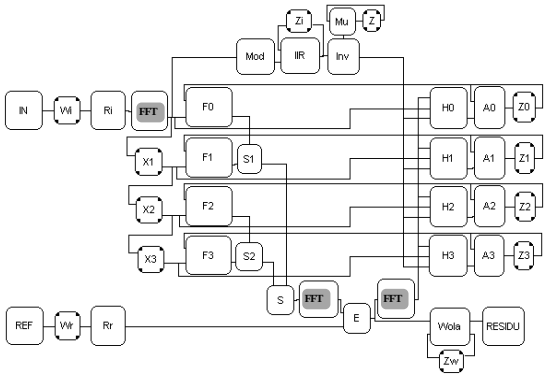

Figure 5 GFDM de l'algorithme GMDFα

Les implantations logicielles du GMDFα publiées [8] nécessitent deux TMS320C40 pour respecter un débit de 8K par seconde et pour une taille du filtre de 1024. Cependant , pour de longs filtres (L = 4096) il y a beaucoup de difficultés pour atteindre les contraintes temps réel. Nous avons proposé en [5] deux méthodes d'implantation de l'algorithme GMDFα: la première est une solution totalement logicielle qui utilise des outils CAD pour une implantation sur une machine multiprocesseurs à base de TMS320C40. La deuxième méthode offre une solution totalement matérielle en utilisant l'outil de synthèse architecturale GAUT pour la conception de circuits dédiés à l'algorithme sous forme d'Asic. Cette implantation respecte les contraintes de temps d'exécution mais offre une solution coûteuse (en surface et en consommation).

3.2 Partitionnement et contraintes :

Dans notre étude, nous considérons une configuration du filtre avec les paramètres suivants : L=1024, N=256, K =4, R= 64, α =4, avec une fréquence d'échantillonnage de 16 Khz. Par conséquent, il y a 1250 cycles disponibles durant une période d'échantillonnage (1 cycle C40 = 50 ns). Le filtrage par blocs du GMDFα engendre une période d'itération de l'algorithme de 80000 = Rx1250 cycles. Par conséquent, l'implantation mixte du GMDFα doit satisfaire un temps d'exécution maximal de 4 ms. Dans le cadre de cette étude, nous proposons un partitionnement manuel dans lequel la FFT est exécuté sur l'Asic. Tous les autres noeuds sont implantés sur le C40.

La figure 5 représente le **GFDM** du GMDFα. Les noeuds ombrés sont associés au composant matériel. La connexion DSP-ASIC est réalisé par un bus (liaison point-à point). Le protocole de

communication est du style Maître-Esclave où le processeur tient le rôle du maître. Le mode de communication entre les deux composants est asynchrone et bufferisé dans le but de réduire les délais de synchronisation.

3.3 Ordonnancement Logiciel/Matériel :

La partition matérielle, représentée par les noeuds de la FFT est décrite en VHDL comportemental. Le temps T_M, imposé à l'exécution de la FF, est de 200 µs (environ 4000 cycles C40), soit le tiers du temps d'exécution d'une FFT (512 points) sur un C40. Le volume de données échangées entre l'Asic et le DSP est de 3078 données de 16 Bits. Le tableau 1 représente l'architecture matérielle obtenue après synthèse par l'outil GAUT. La surface de circuit est de l'ordre de 38 mm2. La liste des composants est extraite de la librairie Compass (16 bits, Technologie *CMOS 1µm*).

La partition logicielle, décrite par un Graphe Flot de Donnée Conditionnée [3,5], est spécifiée sous l'environnement SynDex. Il s'agit d'un outil CAD d'aide à l'implantation des algorithmes de TDSI sur une machine cible multiprocesseurs TMS320C40. SynDex optimise la distribution et l'ordonnancement des tâches de l'algorithme, génère le code C pour chaque processeur ainsi que l'exécutif temps réel qui gère les échéances et transferts de données entre tâches.

La figure 6 décrit l'ordonnancement d'exécution de la partie logicielle sur le DSP produit par SynDex. Le diagramme temporel obtenu fait apparaître les échanges de données Asic-DSP des taches sur chaque composant. L'exécution de l'algorithme sur une architecture hétérogène mixte est de 50000 cycles. Les taux d'occupation de l'Asic et du DSP sont respectivement de 20% et de 65%. Remarquons que le partitionnement effectué manuellement peut être amélioré par l'utilisation de techniques spécifiques de partitionnement Logiciel-Matériel. Ceci s'avère nécessaire pour l'implantation du GMDFα avec des longueurs de filtre plus large (N =4096). Des taches implantées sur le DSP tels que l'adaptation des coefficients des filtres peuvent migrer vers l'Asic. Une autre approche d'optimisation de l'implantation mixte consistera à la duplication des tache sur les deux composants Asic et DSP permettant ainsi de réduire les délais de communication et de synchronisation.

4. Conclusion

Nous avons présenté dans cet article une méthodologie de conception d'implantation mixte Logiciel/Matériel des algorithmes de TDSI. Nous avons particulièrement développé le processus de synthèse du module matériel sous forme d'Asic à l'aide d'un outil de synthèse architecturale GAUT. La méthode de synthèse permet la conception de l'Asic sous contraintes de temps d'exécution, de coûts (surface, consommation) et de contraintes fonctionnelles de communication d'E/S obtenues après partitionnement et synthèse du module logiciel. Cette méthode est illustrée par une implantation mixte de l'algorithme GMDFα, la synthèse du module logiciel étant assurée par l'outil SynDex.

Remerciements

Les auteurs remercient Y. Sorel, C. Lavarenne de l'INRIA Rocquencourt, et A. Gilloire du CNET France Telecom Lannion, pour leurs discussions, et pour leurs contributions à l'implantation de l'algorithme.

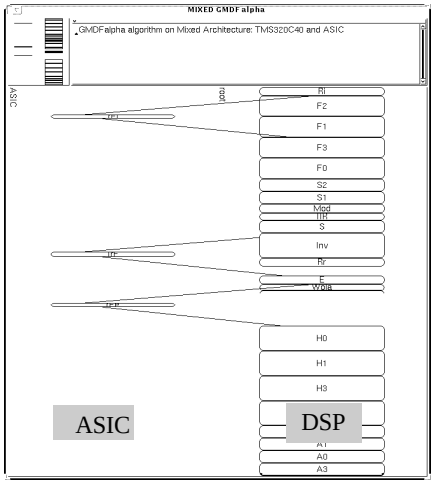

Figure 6 Ordonnancement et exécution du GMDFα

Module Matériel: FFT (512) points)	**Architecture des unités fonctionnelles**
Unité de Traitement (UT)	- 1 additionneur , - 1 soustracteur - 1 multiplieur - 10 registres - 4 multiplexeurs - 3 demultiplexeur **surface 16,18 mm2**
Unit de mémoire UM	RAM **surface 13,40 mm2**
Interface matérielle (UCOM)	2 FIFO - 8 registres - 3 multiplexeurs - 1 demultiplexeur **surface 8, 2 mm2**

Tableau 1 Résultat de la conception matérielle

5. Références

[1] De Micheli Giovanni, M. Sami 'Hardware/Software Codesign' Ed Kluwer Academic Publisher 95.

[2] E. Martin, O. Sentieys, J-L. Philippe, "Synthèse architecturale de coeur de processeurs de traitement du signal". TSI n° 2/1994, pp 251-279

[3] C. Lavarenne, Y. Sorel, "Spécification, optimisation des performances et génération d'exécutif pour application temps réel embarquée multiprocesseur avec SYNDEX", CNES International Symposium, novembre 1992.

[4] J-L. Philippe, D. Chillet, O. Sentieys, E. Martin " Memory aspects in signal processing and HLS tools : some results". EUSIPCO'96, Italie.

[5] A. Baganne, A. Gilloire, E. Martin, P. Martineau, J.P. Thomas "Méthodes d'implantation d'un algorithme d'annulation d'écho acoustique GMDFα" Quinzième colloque GRETSI, Juan les Pins, septembre 1995

[6] A. Baganne, J.L Philippe, E. Martin "Hardware interface design for Real Time embedded systems". GLSVLSI'97, Urbana, Illinois, Mars 1997

[7] O. Amrane, "Identification des systèmes à réponse impulsionnelle longue par filtrage adaptatif en fréquence : Application à l'annulation d'écho acoustique". Thèse de doctorat, ENST, 1992

[8] G. Le Tourneur, J.P. Thomas, A. Gilloire " Real time implementation of the GMDFα Algorithm on a multi DSP TMS320C40 board". EUSIPCO'94, Edinburg, Septembre. 94.

[9] O. Sentieys, J.-Ph. Diguet, J. L. Philippe, E. Martin 'Hardware Module Selection for Real Time Pipeline Architectures using Probabilistic Estimation'. 9th Annual IEEE Inter. ASIC Conf. and Exhibit, 1996, pp.147-150.