Synthèse des unités mémoire pour le traitement du signal

Daniel Chillet, Olivier Sentieys et Michel Corazza

LASTI - ENSSAT - Université de Rennes 1
6 rue de Kérampont, BP 447
22300 Lannion, France

RÉSUMÉ

Les études concernant la synthèse automatique de systèmes numériques ont donné naissance à de nombreux outils synthèse architecturale s'appliquant à différents domaines d'application. Ainsi, les méthodes d'ordonnancement permettent de concevoir les unités de traitement (UT). Les unités de contrôle (UC) sont décrites sous forme d'une machine d'états qui peuvent être, dans la plupart des cas, matérialisées par un outil de synthèse logique du commerce. Les unités de mémorisation (UM) et de communication (UCom) restent, quand à elles, à la charge du concepteur. Cette étude propose une formalisation des problèmes liés à la synthèse des UM en vue d'une intégration dans un outil de synthèse d'architecture. Après une introduction de la synthèse d'architecture et des problèmes spécifiques au TS, nous examinerons l'état de la recherche concernant la synthèse des UM. Nous proposerons une démarche visant à générer ces unités de façon automatique, puis nous présenterons une première série de résultats. Enfin, nous concluerons et nous évoquerons les perspectives afin d'optimiser cette synthèse.

ABSTRACT

The studies concerning the automatic synthesis of numeric systems have produce a large number of high level synthesis tools for different application domains. Therefore the scheduling algorithms allow to conceive the units of treatment (UT). The units of control (UC) are described by state machine that can be, in the majority of cases, materialised by a commercial logic synthesis tool. The units of memorisation (UM) and of communication (UCom) are at the charge of the designer. This study proposes a formula for the problems in relation to the synthesis of the UM in view of an integration in a high level synthesis tool. After an introduction of the synthesis of architecture to the specific problems of TS, we will examine the state of research. We will propose a methodology to design automatically these units. Some results, produce by our tool, will be presented. Then we will conclude and give some way to optimize the synthesis.

1 Introduction

Dans le cadre de la méthodologie de conception de systèmes, deux approches prédominent : la synthèse qui permet d'affiner le niveau d'abstraction à partir de la description d'une application et l'analyse qui permet de modifier cette description afin d'utiliser au mieux les ressources d'une architecture donnée [1]. Ces dernières années, de nombreux outils de synthèse ont été développés permettant de concevoir des architectures dédiées (ASIC : Applied Specific Integrated Circuit) à partir de leur description comportementale [2, 3, 4], l'approche ASIC étant justifiée dès lors que la contrainte de temps ne permet pas l'implantation sur un DSP. Le parallélisme de l'application est alors exploité avec méthode afin de satisfaire la contrainte de temps tout en assurant la minimisation d'une fonction coût (surface, consommation, etc.). Dans un premier temps, les études en synthèse d'architectures ont proposé des outils d'aide à la conception de la partie opérative de l'application. Or, des travaux plus récents [5, 6] ont démontré que, lors de la conception d'un système complet, la mémoire pouvait devenir le point critique de la mise en œuvre. Des études concernant la définition d'une méthodologie de synthèse des unités mémoires ont débuté. Cette étude propose une formalisation des problèmes liés à la synthèse des UM en vue d'une intégration dans un outil de synthèse d'architecture. A terme, le concepteur bénéficiera alors d'outils puissants et rapides réunis dans un environnement de développement convivial, lui permettant de synthétiser des systèmes complexes et complets. L'étude présentée dans ce papier s'intéresse à la synthèse des unités de mémorisation. L'objectif est de proposer une méthodologie de synthèse de ces unités afin de compléter l'outil de synthèse d'architecture GAUT. Nous présentons une méthodologie de synthèse des unités de mémorisation pour des algorithmes de traitement du signal temps réel.

2 Méthodologie de conception

Nous présentons brièvement les étapes de synthèse développées dans notre outil de conception mémoire. Des présentations plus précises ont été faites dans [7, 8, 9, 10].

• **Analyse de la description comportementale**

Les contraintes de cadence des calculs et de dépendances de données exprimées dans le graphe flot de données conduisent la synthèse de la partie opérative à produire des contraintes temporelles sur les dates des transferts entre l'UT et l'UM. Rappelons que le modèle architectural utilisé dans le cycle de conception de notre outil impose que tout transfert soit effectué via un registre de l'unité de traitement. Bien que le temps d'accès mémoire soit pris en compte pour générer la séquence de transferts, celle-ci est, dans de nombreuses applications, irrégulière. La valeur maximale atteinte par ces

courbes fixe le nombre de bus à mettre en œuvre entre l'unité de traitement et l'unité mémoire. Ce nombre peut atteindre des valeurs rendant l'ASIC irréalisable. Dans ces conditions, il est alors intéressant d'essayer de "lisser" ces courbes afin d'obtenir une régularité du nombre de transferts au cours du temps. Nous proposons d'interfacer les séquences de transferts UT et UM par l'intermédiaire d'une "interface" (comme le montre la figure 1). La fonction de lissage est alors défini par $f_l : Tr(n) \overset{f_l}{\longmapsto} Tr'(n)$ Avec $Tr(n)$ la fonction discrète avant l'application de la fonction de lissage (figure 1.a) et $Tr'(n)$ la fonction discrète résultante du lissage (figure 1.b).

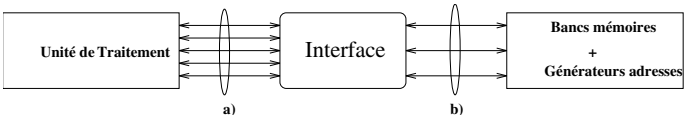

FIG. 1 — Modèle UT / UM

• **Sélection des mémoires**

A partir des besoins exprimés par la synthèse de l'unité de traitement, la sélection consiste à rechercher, dans une bibliothèque, l'ensemble de composants mémoires le mieux adapté à la mémorisation des données. En fait, comme dans la plupart des outils, cette phase sera remise en cause au fur et à mesure de l'avancement dans le processus de synthèse. Elle permet toutefois d'avoir une première évaluation des besoins d'un algorithme. La sélection s'appuie sur une évaluation du coût d'un ensemble de composants mémoires candidats à la résolution du problème. Un ensemble de composants mémoires candidats est un ensemble permettant la mémorisation de toutes les données. Cet ensemble de candidats est un sous ensemble des composants mémoires disponibles dans la bibliothèque. Afin que la sélection soit optimale, il est important de disposer d'une large bibliothèque. Le choix quant aux types de mémoires et leur taille respective est décisif vis à vis de la qualité de la solution finale. Les mémoires proposées sont de 2 types : mémoires dédiées pour lesquelles le nombre de points et la position (interne ou externe à l'ASIC) sont à définir ; mémoires du commerce pour lesquelles le nombre de points mémoires est connu et la position nécessairement externe à l'ASIC. En ce qui concerne les mémoires du commerce, il faut encore distinguer deux cas en fonction de la position du générateur d'adresses associé (interne ou externe à l'ASIC).

• **Distribution des données dans les bancs mémoires**

Il s'agit dans un premier temps de calculer le nombre de bancs mémoires à mettre en place pour résoudre les problèmes de cohérence de données et de simultanéité des transferts, puis, dans un deuxième temps, de ranger chaque donnée dans un banc mémoire particulier. La représentation que nous utilisons est un graphe $\mathcal{G}\{N, A\}$ de conflits entre données. L'ensemble des noeuds N est constitué par l'ensemble des données à mémoriser, alors que l'ensemble des arêtes A représente l'ensemble des conflits entre les données. Il y a "conflit" entre deux données s'il existe un temps discret tel que les deux données sont transférées simultanément. Dans ce cas, une arête est créée entre ces deux noeuds. La recherche du minimum de bancs mémoires nécessaires N_B pour mémoriser l'ensemble des données est équivalente à la recherche de "l'index chromatique" du graphe $\mathcal{G}$. Il s'agit de rechercher le nombre minimum de couleurs pour colorier tous les noeuds du graphe $\mathcal{G}$ sous la contrainte que deux noeuds reliés par une arête ne peuvent porter la même couleur.

• **Placement des données dans les bancs mémoires et synthèse des générateurs d'adresses**

Après la phase de distribution, chaque donnée de la séquence est affectée à un banc mémoire. L'organisation dans le banc n'est pas pour autant définie. En effet, le placement de chaque donnée à une adresse précise reste à spécifier. Le placement des données dans les bancs consiste donc à affecter une adresse précise à chacune des données. Cette affectation doit se faire de façon à limiter la complexité des générateurs d'adresses associés à chaque banc mémoire. L'impact du placement sur la complexité des générateurs d'adresses devrait nous conduire à considérer le placement des données dans les bancs au moment de la distribution. Toutefois, cette approche conduisant à une complexité de recherche trop importante, nous avons alors opté pour une solution prenant en compte le placement des données en aval de la distribution. Nous avons constaté que cette prise en compte tardive du problème de placement des données permet malgré tout de concevoir des générateurs d'adresses simples.

3 Le flot de conception complet

La méthodologie que nous avons développée a été implantée dans le flot de conception de l'outil de synthèse architecturale GAUT. Ce flot de conception est représenté par le schéma de la figure 2.

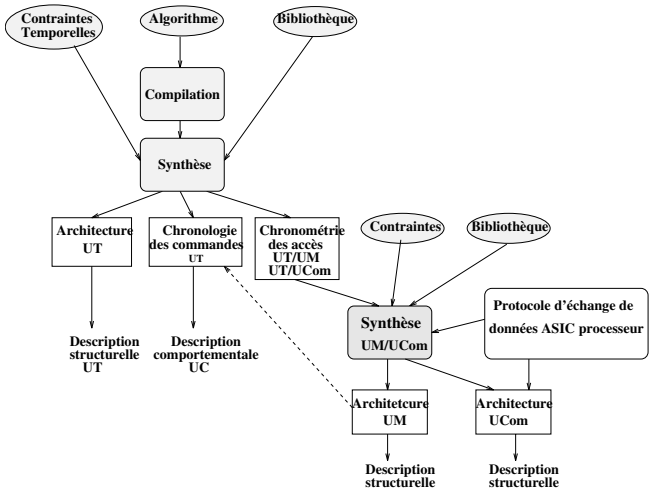

FIG. 2 — Flot de conception complet de GAUT

La conception se déroule en 2 phases :

– dans une première phase, les synthèses des unités de traitement et de contrôle sont réalisées. Celles-ci s'effectuent sous contrainte de temps réel et s'appuyent sur une bibliothèque d'opérateurs. Cette phase produit un certain nombre de contraintes que devront respecter les synthèses des autres unités ;

– dans la seconde phase, on s'intéresse à la synthèse des unités de mémorisation et de communication. La méthodologie présentée dans ce papier prend place à ce niveau et sera complétée, à terme, par la synthèse de l'unité de communication.

Notons que parallèlement à ces 2 phases de synthèse, des estimations de complexité et consommation du circuit en cours de conception peuvent être réalisées. Elles offrent au concepteur une vision probabiliste d'une fonction coût, et permettent alors à celui-ci de guider efficacement l'ensemble du cycle de conception.

Enfin, le développement d'une nouvelle version de l'outil GAUT est actuellement en cours. Elle offrira, à terme, l'ensemble des outils nécessaires pour la conception d'un circuit spécifique.

4 Résultats

Le tableau 1 présentent les résultats de l'évaluation de la complexité des différents algorithmes utilisables pour une application d'annulation d'écho acoustique. Tous ces filtres possèdent un vecteur d'échantillons vieillissant de 128 éléments. À chacun de ces algorithmes correspond une qualité de traitement et une complexité d'implantation. Dans une phase de prototypage d'algorithmes, l'analyse de la complexité mathématique consiste à dénombrer les opérations à effectuer (voir les deux premières lignes du tableau 1). Avec l'apparition des outils de synthèse d'architectures (tel que GAUT), cette analyse de complexité a été complétée par une connaissance précise des besoins et du coût de l'unité de traitement.

L'analyse de complexité, tant au niveau algorithmique qu'au niveau du nombre d'opérateurs et de registres, permet alors de comparer ces filtres. Ainsi la première ligne du tableau 1 montre que le filtre FIR est le moins complexe (mais il est le moins intéressant du fait de sa non adaptabilité au court du temps). Notons de plus que les deux premières lignes font apparaître une équivalence entre le filtre LMS et BLMS. Notre outil de synthèse mémoire permet de compléter les résultats d'implantation de chacun de ces filtres. En effet, alors que seul le nombre de données à mémoriser est connu après synthèse de la partie opérative, l'outil de synthèse fournit une description précise de l'organisation de la mémoire : nombre de bancs mémoire, nombre de points par banc et coût des générateurs d'adresses.

Nous remarquons que les synthèses UT des filtres LMS, BLMS et FTF ont produit des données intermédiaires de calcul, alors que les synthèses UT des filtres FIR et GAL parviennent, par une utilisation judicieuse des registres de l'UT, à diminuer le nombre de données à mémoriser.

La 4èmeligne montre l'effet du lissage sur le nombre de bus de données à implanter. Nous observons, notamment pour le filtre FTF, que l'application d'une fonction de lissage permet de diminuer de façon importante ce nombre. Le nombre de bancs mémoire nécessaires est ensuite fourni à la ligne 5 du tableau. Il est obtenu par un algorithme glouton qui peut ne pas fournir la solution optimale. Nous pouvons observer que ce nombre est différent du nombre de bus pour les filtres BLMS, GAL et FTF. En effet, pour ces trois filtres, les contraintes de cohérences imposent un nombre de bancs mémoire supérieur au nombre de bus de données.

Les lignes 6 à 9 indiquent la taille des mémoires à utiliser afin de mémoriser l'ensemble des données. Compte tenu de la faible contrainte de temps (qui permet à la synthèse UT de fournir une séquence d'accès faiblement contrainte), il n'y a pas de problème particulier pour assurer les cadences de transferts. La somme de toutes les tailles est naturellement légèrement supérieure au nombre de données à mémoriser. Indiquons, pour complèter le tableau, que tous les éléments de chaque vecteur vieillissant sont distribués dans un seul banc mémoire.

Les informations concernant les complexités des générateurs d'adresses sont également données aux lignes 6 à 9. Ces complexités sont données en fonction du type de générateur :

• pour une machine d'états, M : la complexité est donnée par le nombre de bascules du registre d'états ;

• pour un compteur, C : la complexité est donnée par le nombre de bascules du compteur ;

• pour une UAL et des registres, U : la complexité est donnée par le nombre bits de l'opérateur additionneur (donc par le nombre de bascules du registre de base à mettre en œuvre.

Nous donnons ensuite, à ligne 10 du tableau, une évaluation de la surface des unités de traitement et de mémoire. La surface de l'unité de traitement est fournie par GAUT, alors que notre outil de synthèse mémoire effectue une évaluation du coût de la mémorisation en utilisant une bibliothèque de composants.

Enfin, la dernière ligne du tableau montre l'importance de la mémoire par rapport au coût global de l'ASIC. La surface de la mémoire atteint, en effet dans la plupart des cas, plus de 65 % de la surface totale.

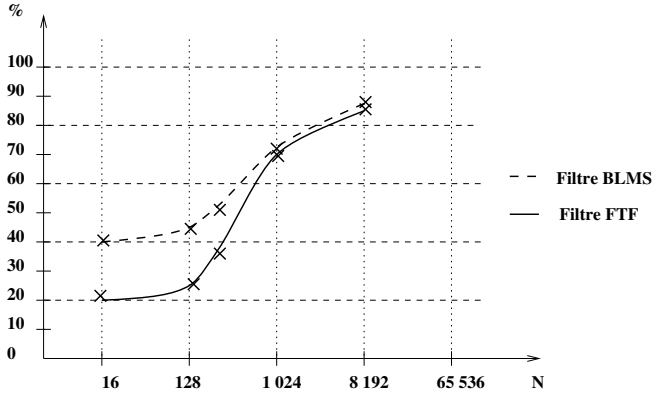

FIG. 3 — Variation du rapport $\frac{S_{UM}}{Surface\ Total} * 100$ en fonction du nombre de points des filtres FTF et BLMS

Les courbes de la figure 3 présentent l'évolution du pourcentage de surface nécessaire pour la mémoire en fonction de la taille et ceci pour les filtres BLMS et FTF. Ces courbes montrent que l'augmentation de la taille des filtres provoque une augmentation importante , plus la mémoire devient prépondérante par rapport à l'unité de calcul.

5 Conclusion

Nous avons présenté brièvement les méthodes de conception des unités de mémorisation que nous avons mis en œuvre dans le projet de synthèse architecturale GAUT. Rappelons que notre approche se situe en aval de la synthèse de la partie opérative et qu'elle est donc contrainte par cette dernière. L'outil développé, fournit une description précise de la structure de l'unité mémoire : nombre de bancs mémoires, nombre de points par banc, structure des générateurs d'adresses. La méthodologie de conception que nous avons adopté permet d'aboutir rapidement à une solution mémoire complète et optimisée. Couplé à un outil de synthèse d'architectures, notre outil permet alors d'affiner le prototypage d'algorithmes. En effet, les résultats de la synthèse d'architectures en terme de type et de nombre d'opérateurs et de registres, sont alors complétés par une évaluation précise du coût de mémorisation. L'ensemble fournissant une plate forme de développement et/ou de prototypage complète et rapide.

	FIR	LMS	BLMS	GAL	FTF
1	Complexité en opérations *, +	N, N	2N, 2N	2N, 2N	3N+5C †, 2N+4C †	8N+13, 8N+5
2	Complexité en opérateurs *, +, reg	1,1,4	1,1,4	1,1,4	1,1,4	1,1,4
3	Nb de variables déclarées, à mémoriser	259, 264	262, 385	607,720	772, 648	1299, 1968
4	Nb Bus généré par GAUT, nécessaire	2, 2	2, 2	4, 2	3, 2	8, 3
5	Nb Bancs	2	2	4	3	4
	Distribution : taille des bancs mémoire nécessaires					
	Placement : type ‡ de générateurs et complexité (en nb de bascules)					
6		256 ; C : 8	256 ; C : 8	256 ; C : 8	256 ; C : 8	512 ; C : 9
7		128 ; C : 7	256 ; C : 8	256 ; C : 8	256 ; C : 8	512 ; C : 9
8			256 ; C : 8	256 ; C : 8	256 ; C : 8	512 ; C : 9
9			128 ; C : 7			512 ; C : 9
10	Surface UT ; UM (mm^2)	2,12 ; 3,25	2,53 ; 4,03	2,54 ; 7,27	2,64 ; 6,04	2,41 ; 13,9
11	$\frac{S_{UM}}{Surface\ Total} * 100$ en %	61	61	74	69	85

† : le nombre C de cellules est compris entre 1 et N, ordre du filtre

‡ indication du type de générateur : C : Compteur ; U : UAL et registres ; M : Machine d'états

TAB. 1 — Complexité des filtres utilisés en annulation d'écho acoustique (128 points, 8Khz, données sur 16 bits)

Références

[1] D.Gajski, N.Dutt, A.Wu, and S.Lin. *High-Level Synthesis - Introduction to Chip and System Design*. Kluwer Academic Publishers, 1992.

[2] E.Martin, O.Sentieys, and J.L.Philippe. Synthèse architecturale de coeur de processeurs de traitement du signal. *Techniques et Sciences Informatiques*, 1(2), 1994.

[3] M.Potkonjak and J.M.Rabaey. Exploring the algorithmic design space using high level synthesis. *VLSI Design Methodologies for Digital Signal Processing Architectures*, Tome 257 :131–167, 1994.

[4] C.Y.Wang and K.K.Parhi. Resource-constrained loop list scheduler for dsp algorithms. *Journal of VLSI Signal Processing*, pages 75–96, 1995.

[5] F.Balasa, F.Catthoor, and H.De Man. Exact evaluation of memory size for multi-dimensional signal processing systems. In *IEEE International Conference on Computer Aided Design*, 1993.

[6] T.Kim and C.L.Liu. A new approach to the multiport memory allocation problem in data path synthesis. *Integration, the VLSI Journal*, pages 133–160, 1995.

[7] D.Chillet, J.P.Diguet, J.L.Philippe, and O.Sentieys. Méthodologie de conception des unités mémoires. *Techniques et Science Informatique*, 16(4) :483–512, 1997.

[8] O.Sentieys, D.Chillet, J.P.Diguet, and J.L.Philippe. Memory module selection for high level synthesis. In *IEEE Workshop on VLSI Signal Processing*, 1996.

[9] D. Chillet. Conception des unités de mémorisation pour un outil de synthèse architecturale dédiée au traitement du signal. Rapport de fin de 2^{eme} année de thèse, September 1996.

[10] D.Chillet. *Méthodologie de conception architecturale des mémoires pour circuits dédiés au traitement du signal temps réel*. PhD thesis, Enssat, Université de Rennes 1, January 1997.