SYNTHÈSE AUTOMATIQUE D'ARCHITECTURES MULTIPROCESSEURS HÉTÉROGÈNES

Mirjam KLENK, Elisabeth LAHALLE, Daniel POULTON

SUPELEC - Service des Mesures
Plateau de Moulon
91192 Gif sur Yvette Cedex

RÉSUMÉ

Concevoir rapidement des systèmes de traitement numérique des signaux optimisés à la fois en termes de performances (temps de réponse, cadence) et en termes de réalisation (coût, consommation, surface de Silicium, ...) est aujourd'hui un besoin largement exprimé.

Nous présentons dans cette communication une méthode de synthèse automatique d'architectures matérielles multiprocesseurs hétérogènes optimisées pour l'exécution d'un traitement ou d'une classe de traitements. La méthode est fondée sur une heuristique d'optimisation qui détermine une solution acceptable minimisant une fonction de coût (fonction du nombre de processeurs) sous différentes contraintes (temps de réponse maximal, nombre maximal de processeurs, ...). Elle exécute le placement / ordonnancement des primitives du traitement non plus sur des processeurs, mais sur différentes tranches de temps. L'architecture matérielle est construite simultanément.

ABSTRACT

There is today a strong need for designing optmimal signal processing systems. This optimization should be related not only to computing power (respond time, throughtput) but also to consistency with constraints (cost, consumption, silicium area, ...).

In this paper, we present a method for direct automatic synthesis of application-specific heterogeneous multiprocessor architectures. This method is based on an optimization heuristic which finds a solution that mimimizes a cost function (depending on the number of processors) under constraints (latency, maximum number of processors , ...). This heuristic uses a mapping / scheduling on different time slices, rather than on processors. The hardware architecture is constructed simultaneously.

1. INTRODUCTION

L'augmentation de la complexité des applications de traitement du signal nécessite de plus en plus l'utilisation d'architectures matérielles multiprocesseurs. L'évolution de la technologie met à la disposition des concepteurs des ressources de calcul diverses et de plus en plus performantes : processeurs généraux, processeurs de traitement du signal (TMS320C40, Transputer, ...), unités câblées (ASIC). La conception de systèmes de traitement numérique des signaux optimisés à la fois en termes de performances (temps de réponse) et en termes de réalisation nécessite alors l'utilisation d'une architecture matérielle constituée d'une structure non régulière de processeurs hétérogènes.

Pour déterminer une telle architecture matérielle optimisée, le concepteur doit résoudre des problèmes divers tels que le choix du nombre et du type des processeurs ou des sous-ensembles (cartes) à inclure dans le système, la sélection des types et des performances des moyens de communication, la répartition et l'ordonnancement des primitives sur les processeurs, ..., et ceci en tenant compte des contraintes de performances, de coûts, de réalisabilité. Afin de réduire les temps de conception, il est indispensable de disposer de nouveaux outils de conception [1], en particulier pour la synthèse automatique d'architectures matérielles multiprocesseurs à partir de spécifications fonctionnelles de haut niveau [2][3].

Les outils actuels se limitent toutefois à des architectures homogènes en ressources de calcul et à système d'interconnexions régulier ou à la

synthèse des systèmes logiques de bas niveau [4][5].

Nous proposons dans cette communication une méthode de synthèse directe et automatique d'architectures matérielles optimisées pour l'exécution d'un traitement ou d'une classe de traitements.

2. PRINCIPES GÉNÉRAUX

La méthode utilise une structure de données [6] contenant toutes les informations nécessaires pour modéliser les ressources matérielles disponibles (processeurs, liens de communication, mémoires), les primitives de traitement (complexité, entrées / sorties, ...) ainsi que les dépendances entre ces dernières et les processeurs. On peut ainsi déterminer, par exemple, le processeur optimal pour une primitive donnée selon un critère quelconque (temps d'exécution minimal, mais aussi contraintes : puissance dissipée, surface occupée, coût associé à son utilisation, ...).

Une application de traitement du signal est représentée par un ensemble de processus (primitives de traitements) communicants, liés par des contraintes de précédence sous la forme d'un graphe flot de données. Chaque primitive consomme et / ou produit des données en provenance et à destination d'autres primitives.

Une architecture matérielle multiprocesseur constitue un réseau de processeurs et est représentée par un graphe dans lequel les nœuds constituent les ressources de calcul (appelées par la suite processeurs) et les arcs les voies de communication.

L'objectif de la synthèse directe est d'obtenir une architecture matérielle optimisée pour l'exécution de l'application, construite progressivement à partir des processeurs disponibles, au cours d'un unique balayage du graphe de traitement. L'heuristique d'optimisation consiste à résoudre le problème dual d'un problème de placement / ordonnancement, c'est à dire effectue ces opérations non plus sur une architecture existante, mais sur des tranches de temps, action que nous appellerons portage.

Pour cela, à partir de l'analyse des chemins du graphe de précédence modélisant le traitement, les nœuds (primitives du traitement) sont classés par ordre de priorité décroissante de portage d'après un critère lié aux dates de début et de fin d'exécution " au plus tôt " et " au plus tard ". Simultanément, pour chaque type de processeur utilisable, est calculée son " aptitude ", coefficient représentatif de sa capacité à exécuter les primitives du traitement et servant de base pour le choix du type de processeur à ajouter à l'architecture matérielle en cours de synthèse.

L'initialisation de l'architecture est effectuée à partir du processeur optimal pour la primitive de priorité maximale et d'aptitude la plus élevée, dégageant un intervalle de temps pour l'exécution des primitives. Cette primitive est ensuite portée sur la tranche de temps ad-hoc ainsi disponible.

Pour chaque nœud suivant, un algorithme " glouton " :

- dans le cas où une tranche de temps nécessaire à son exécution est disponible, réalise le portage,
- sinon, intègre un nouveau processeur à l'architecture en cours de synthèse, puis réalise le portage sur les tranches de temps dégagées.

Ce processus itératif se termine quand toutes les primitives ont été portées.

3. CLASSIFICATION DES PRIMITIVES

Notations utilisées :

Nous utiliserons les notations suivantes :

T_i	primitive i du graphe de traitement
$S_s(T_i)$ $L_s(T_i)$	dates de début exécution " au plus tôt " et " au plus tard " de la primitive T_i
$S_r(T_i)$ $L_r(T_i)$	dates de fin exécution " au plus tôt " et " au plus tard " de la primitive T_i
P_i	processeur i
$d(P_i,P_j)$	durée de la communication entre les processeurs P_i et P_j
$D_{P_j}(T_i)$	durée d'exécution de la primitive T_i sur le processeur P_j
Π	ensemble des processeurs utilisables
$\Pi_i(T_j,T_i)$	ensemble des couples de processeurs optimaux pour l'exécution de la primitive T_i

Dates caractéristiques

A partir du graphe de précédence modélisant le traitement, des modèles des processeurs utilisables et des relations de dépendance, les primitives du traitement sont classées vis-à-vis du critère primordial : le temps de latence.

En supposant, ce qui ne restreint pas la généralité du problème, qu'il existe une primitive (appelée primitive initiale) devant exécutée avant toutes les autres primitives de l'application et une primitive (appelée primitive finale) devant être exécutée après que l'exécution de toutes les autres primitives soient terminée , nous commençons par déterminer les dates de fin et de début d'exécution " au plus tôt " de chaque primitive T_i en

parcourant les chemins du graphe de traitement de la primitive initiale à la primitive finale :

$$S_r(T_i) = \max_{T_k \in \text{préd}(T_i)} \left\{ S_r(T_k) + \min_{\substack{P_v \in \Pi_k(T_j,T_k) \\ P_\mu \in \Pi}} \left(d(P_v, P_\mu) + D_{P_\mu}(T_i) \right) \right\} \quad (1)$$

$$S_s(T_i) = \max_{T_k \in \text{préd}(T_i)} \left\{ S_r(T_k) + \min_{\substack{P_v \in \Pi_k(T_j,T_k) \\ P_\mu \in \Pi_i(T_i,T_i)}} \left(d(P_v, P_\mu) \right) \right\} \quad (2)$$

L'ensemble des couples de processeurs (P_v, P_μ) pour lesquels les relations (1) et (2) sont vérifiées définissent l'ensemble $\Pi_i(T_k, T_i)$ des couples de processeurs optimaux pour cette primitive tandis que les arcs correspondant définissent le chemin critique local pour cette primitive :

$$C(T_i) = \left\{ \bigcup_k \left(C(T_k), T_i \right) \right\} \quad (3)$$

Lorsque toutes les primitives du graphe ont été traitées, les caractéristiques de la primitive finale fournissent le temps d'exécution sous-optimal du traitement (compte tenu des processeurs utilisables) et le chemin critique global :

$$t_{latence\,opt} = S_r\left(T_{finale}\right) \quad (4)$$

$$C_{critique} = C\left(T_{finale}\right) \quad (5)$$

Pour obtenir pour l'application un temps de latence effectivement égal à $T_{latence_{opt}}$, il est nécessaire que les dates d'exécution de chaque primitive ne dépassent pas des seuils : les dates de début et de fin d'exécution " au plus tard ". Parcourant les chemins du graphe de traitement de la primitive finale à la primitive initiale, ces dates sont calculées pour chaque nœud par :

$$L_r(T_i) = \min_{T_k \in \text{succ}(T_i)} \left\{ L_s(T_k) - \max_{P_v, P_\mu \in \Pi} \left(d(P_v, P_\mu) \right) \right\} \quad (6)$$

$$L_s(T_i) = L_r(T_i) - \min_{P_\mu \in \Pi} \left(D_{P_\mu}(T_i) \right) \quad (7)$$

Classement des primitives

Toutes les primitives se trouvant sur le chemin critique devront pouvoir être placées sur l'un de leurs processeurs optimaux. Les relations (1) et (2) garantissent alors que le temps de latence déterminé pourra être effectivement atteint en parcourant le chemin critique de la tâche finale à la tâche initiale. Elles sont donc classées prioritairement par ordre décroissant de date de début d'exécution au plus tôt.

Afin de minimiser le nombre de processeurs utilisés, les autres primitives devront être placées dans toute la mesure du possible sur les processeurs de l'architecture déjà existante. Elles sont classées à la suite par ordre croissant de $S_s(T_i)$, puis par ordre décroissant de $S_r(T_i)$ en cas d'égalité.

Aptitude des processeurs

Le choix du type de processeur à inclure dans l'architecture matérielle en cours de synthèse dépend des primitives non encore portées. Un processeur sera d'autant plus apte qu'il est capable d'exécuter l'ensemble de ces primitives en un temps le plus faible possible.

Le critère utilisé, appelé aptitude, est fourni par :

$$A(P_i) = \sum_{\substack{T_k \in \text{primitives} \\ \text{non portées}}} \frac{1}{D_{P_i}(T_k)} \quad (8)$$

4- PORTAGE DES PRIMITIVES

Primitives du chemin critique

L'initialisation de l'architecture est effectuée à partir du processeur optimal pour la primitive de priorité maximale, et dont l'aptitude est la plus élevée. Un intervalle de temps entre $S_s\left(T_{initiale}\right)$ et $S_r\left(T_{finale}\right)$ est ainsi mis à disposition du portage pour l'exécution des primitives. Cette primitive est portée sur la tranche de temps nécessaire à son exécution.

Si un processeur optimal pour la primitive suivante et faisant partie du chemin critique existe déjà dans l'architecture en cours de synthèse et qu'au moins une tranche de temps correspondant aux dates de début et de fin d'exécution au plus tôt de cette primitive est disponible, la primitive est ensuite portée.

Sinon, le processeur optimal pour cette primitive et dont l'aptitude est la plus élevée est intégré à l'architecture en cours de synthèse, le portage de la primitive étant ensuite réalisé sur les tranches de temps ainsi mises à disposition.

Les paramètres régissant les choix sont réactualisés à chaque étape du traitement, en particulier l'aptitude des processeurs.

Autres primitives

L'algorithme qui réalise le portage des primitives ne se trouvant pas sur le chemin critique est très semblable au précédent. Les différences concernent les critères qui décident des choix effectués. Le portage d'une telle primitive est réalisé à l'intérieur de la tranche de temps correspondant aux dates de début et de fin d'exécution au plus tard si un processeur

susceptible d'exécuter la primitive existe dans l'architecture en cours de synthèse. Si aucune tranche de temps n'est disponible, le processeur susceptible d'exécuter la primitive et dont l'aptitude est la plus grande est intégrée à l'architecture.

6. EXEMPLE

La méthode présentée a été appliquée à l'application dont le graphe est représenté sur la figure 1.

Quatre types différents de processeurs étaient disponibles pour implémenter cette application.

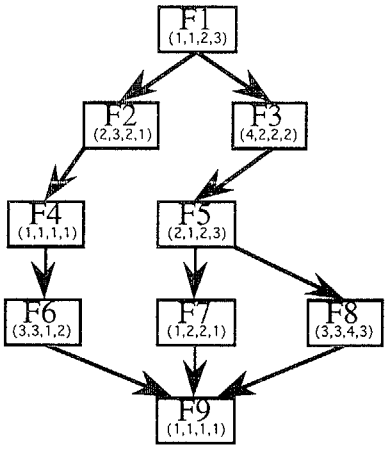

Figure 1 : Graphe de l'application

L'architecture matérielle synthétisée en utilisant notre méthode est donné figure 2 et le diagramme de Gantt correspondant au système obtenu après placement des primitives figure 3.

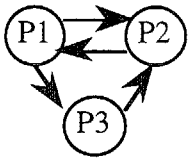

Figure 2 : Architecture synthétisée

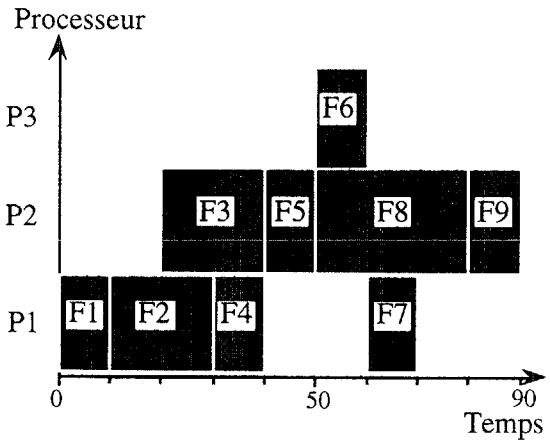

Figure 3 : Diagramme de Gantt correspondant

On peut remarquer que compte tenu des mauvaises performances du processeur P1 vis à vis du temps d'exécution de la primitive F6, le processeur P3 est a été naturellement inclus dans l'architecture synthétisée, l'objectif étant ici de minimiser le temps de latence.

7. CONCLUSION

L'outil de synthèse associé à la méthode présentée a été intégré dans un environnement d'aide à la conception des systèmes de traitement des signaux. Son exploitation, en permettant en particulier l'exploration de l'espace des solutions et par là même une évaluation rapide des choix de l'architecture du système fournit une aide efficace à la décision dès le début de la conception du système.

Références

[1] Y. Le Berre, " Elaboration de méthodes et d'outils d'aide à la conception d'architectures multiprocesseurs optimisées pour le traitement du signal ", Thèse de Doctorat, 10 Juin 1995.

[2] M. McFarland, A. Parker, R. Camposano, " The High-Level Synthesis of Digital Systems ", in *Proceedings of the IEEE*, VOL. 78, February 1990.

[3] C. Gebotys, " An Optimization Approach to the Synthetisis of Multichip Achitectures ", *in IEEE Trans. on VLSI Systems*, VOL. 2, n°1, pp. 11-30, March 1994.

[4] C. Hwang, J. Lee, Y. Hsu, " A Formal Approach to the Scheduling Problem in High Level Synthesis ", in *IEEE Trans. Computer-Aided-Design*, VOL. 13, n° 4, pp, 464-481, April 1191.

[5] Y. Chen, Y. Hsu, C. King, " Multipar : Behavior Partition for Synthetizing Multiprocessor Architectures ", in *IEEE Trans. on VLSI Systems*, VOL. 2, n° 1, pp. 21-32, March 1994.

[6] Y. Le Berre, E. Lahalle, D. Poulton, " Modèlisation des systèmes de traitement des Signaux ", Quatorzième colloque sur le Traitement du Signal et des Images, Juan-les-Pins, Septembre 1993.