MODELISATION DES SYSTEMES DE TRAITEMENT DES SIGNAUX

Daniel POULTON, Elisabeth LAHALLE, Yann LE BERRE

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

RESUME

Pour concevoir des systèmes de traitement numérique du signal optimisés en termes de coût et performances, il est nécessaire de disposer d'outils permettant de modéliser efficacement leur architecture matérielle. Nous proposons dans cet article une telle modélisation, basée sur une structure d'accueil qui permet de représenter non seulement les performances (puissance de calcul, débits, ...) mais aussi les contraintes de réalisation (coûts, topologie, consommation, surface de silicium, ...).

Cette structure est destinée à être exploitée par d'autres outils (placement / ordonnancement, générateur automatique, ...) afin de tendre vers la synthèse automatique des architectures matérielles optimisées pour une classe donnée de traitements.

ABSTRACT

Design of optimized multiprocessor signal processing systems requires an efficient model of there hardware architecture. We propose in this paper a model based on a data structure which does not only takes into account the performances (processing power, throughtput, ...), but also the carying out constraints (cost, size, subset breaking up, consomption, silicium area, ...).

This structure is to be used by other tools (mapping / scheduling, automatic generation, ...) to do an automatic synthetisis of hardware architectures optimized for a known class of signal processing algorithms.

1. INTRODUCTION

L'augmentation de la complexité des applications de traitement du signal nécessite de plus en plus l'utilisation d'architectures matérielles multiprocesseur spécialisées.

L'évolution de la technologie met à la disposition des concepteurs des processeurs de plus en plus performants et s'il existe aujourd'hui des processeurs de traitement du signal conçus pour réaliser un système multiprocesseur (TMS320C40, Transputer, ...), leur utilisation n'est réellement efficace que pour l'exécution des traitements de bas niveau.

Il faut d'autre part noter que l'utilisation des outils d'I.A.O. électronique pour la réalisation de composants spécifiques conduit actuellement à des réalisations pour des prix et dans des délais réalistes.

Une architecture matérielle optimisée pour une application donnée devra alors comporter 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.

Les solutions à ces problèmes sont difficiles à obtenir et dépendent essentiellement des caractéristiques de l'application et du matériel disponible.

Comme il est irréaliste de construire des prototypes uniquement pour tester ou vérifier les performances de chaque architecture candidate, il est indispensable de disposer d'un modèle d'architectures qui permette à des outils d'aide à la conception (synthèse d'architectures matérielles, évaluation de performances mais aussi vérification de la conformité à des spécifications fonctionnelles et à des contraintes matérielles) de prédire la faisabilité d'une application sur une architecture matérielle [1][2][3][4][5].

Nous proposons dans cette communication une structure d'accueil permettant de modéliser l'architecture d'un système de traitement du signal. Cette structure d'accueil permet de déterminer si un traitement est implantable sur une architecture matérielle donnée compte tenu des contraintes de performances et de réalisation, et est une étape vers la conception optimale d'architecture.

2. PRINCIPES GENERAUX

Nous appellerons "entités de description" les éléments constitutifs d'un modèle de l'architecture d'un système considéré dans ses aspects matériels et logiciels. A chaque entité sont associées des informations quantitatives appelées paramètres. L'exploitation de ces paramètres doit permettre d'effectuer aussi bien la synthèse d'architectures matérielles optimisées que la validation d'un système.

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

Une architecture matérielle multiprocesseur sera représentée par un ensemble de ressources de calcul communiquantes. La détermination d'une architecture matérielle optimisée ou la validation d'un système nécessitent de connaître les relations de dépendance entre les primitives de traitement et les ressources de calcul.

Dans la structure d'accueil proposée, les entités de description comprennent les primitives de traitement, les ressources de calcul et de communication et les relations de dépendance entre les primitives et les ressources de calcul.

On pourra ainsi déterminer par exemple:

a) quand l'objectif est la synthèse optimisée d'architecture matérielle, le processeur optimal pour une primitive donnée selon un critère quelconque (temps d'exécution minimal, mais aussi contraintes, puissance dissipée, surface occupée, coût, ...),

b) quand l'objectif est la validation d'un système, le temps d'exécution d'une primitive sur une unité de traitement, les capacités des mémoires de programme et de données nécessaires, le coût associé à l'utilisation du processeur, la puissance dissipée.

3. ENTITES DE DESCRIPTION

3.1. Primitives de traitement

Dans cette structure d'accueil, une primitive est représentée par un nom symbolique et des paramètres qui peuvent dépendre d'une fonction mathématique du nombre d'itérations ou des points traités (exemple: $N * \log(N)$ pour la transformée de Fourier rapide).

```
Prim  { Nom,
        Performances [Type, NbPtItér, Complexité(N)]
        Entrées [(NomE, NbDonnées(N), Largeur),...],
        Sorties [(NomS, NbDonnees(N), Largeur),...],
        Autres  [ ... ] }
```

Figure 1: modélisation des primitives de traitement

Ils comprennent (Figure 1) le type de la primitive (fonction immédiate, retard, ...), le nombre de points traités ou le nombre d'itérations, la complexité fonction de N, la définition et les caractéristiques des entrées et sorties.

Le champ "Autres" (présent dans toutes les entités de description) permet éventuellement d'ajouter des paramètres au descriptif pour faire évoluer la structure d'accueil.

3.2. Ressources de calcul

Les entités de description de l'architecture matérielle du système seront les ressources de calcul (que nous appellerons processeurs par la suite) et leurs diverses interconnexions (liaisons bi-privées ou bus).

Chaque processeur peut posséder une mémoire locale, dispose de voies d'entrées et de sorties de données. Les communications inter-processeur peuvent se faire par passage de messages ou par utilisation d'une mémoire partagée.

Un processeur est représenté par un nom symbolique. Les paramètres caractéristiques optionnels précisent (Figure 2):

- la mémoire: capacités des mémoires de programme et de données disponibles,
- les performances: temps de cycle, capacité d'effectuer en parallèle des traitements avec des communications, coefficient de pondération du temps d'exécution des primitives (pour prendre en compte le temps de commutation d'une primitive à l'autre),
- les caractéristiques des voies d'entrées / sorties: temps de latence, débit maximal et largeur, en nombre de fils de connexion,
- les contraintes: coût impliqué par l'utilisation du processeur, surface occupée sur une carte et consommation.

Le champ "Autres" permet au concepteur, comme dans le cas des primitives, d'inclure dans la structure de données des paramètres supplémentaires.

```
Proc { Nom
       Processeurs [NomProc1, ... ]
       Mémoire [TailleRom, TailleRam]
       Performances [Horloge, DMA, Surcharge],
       Entrées [(NomE, Latence, DébitMax, Largeur ), ... ],
       Sorties  [( NomS, Latence, DébitMax, Largeur ), ... ],
       Contraintes [Coût, Surface, Consommation],
       Autres  [ ... ] }
```

Figure 2: modélisation des ressources de calcul

Le champ "Processeurs" permet de représenter des sous-ensembles (cartes, sous-systèmes, ...) incluant plusieurs processeurs et leurs voies d'intercommunication permettant la conception modulaire d'un système complexe ou la vérification de la réalisabilité du sous-ensemble moyennant une contrainte (surface occupée ou puissance dissipée, par exemple).

Les paramètres caractéristiques d'un sous-ensemble se déduisent des paramètres des éléments constitutifs correspondants.

3.3. Relations primitives / processeurs

Lors de la modélisation des relations entre les primitives et les processeurs, nous avons supposé que chaque processeur peut exécuter au plus une primitive à chaque instant et qu'une primitive est exécutée par un seul processeur. Enfin, une fois l'exécution d'une primitive de

traitement commencée, cette primitive occupe le processeur pendant une durée égale à son temps d'exécution, sans pouvoir être interrompue.

La modélisation que nous avons adoptée ici permet de caractériser les performances de chaque primitive (temps d'exécution , mais aussi taille des mémoires programme et données nécessaires, ...) sur chaque processeur. Pour ce faire, les informations suivantes pour chaque couple (Processeur, Primitive) sont données: le nombre de cycles par point ou itération et les capacités (fonctions mathématiques du nombre de points ou d'itérations) des mémoires de programme et de données nécessaires à l'exécution de la primitive sur le processeur (Figure 4).

Quand le temps d'exécution dépend de la valeur des données (exemple: minimisation de fonction), la donnée d'un nombre d'itérations maximal permet d'avoir accès au temps d'exécution maximal ("pire cas").

```
PriPro  { NomProc, NomPrim,
          Performances [NbCycles],
          Mémoire [TailleRom(N), TailleRam(N)],
          Autres  [ ... ] }
```

Figure 4: modélisation des relations primitives / processeur

3.4. Modélisation des communications

Les communications entre les ressources de calcul peuvent s'effectuer par l'intermédiaire de liaisons de type bi-privées, bus ou d'une mémoire partagée.

3.4.1. Liaisons bi-privées

Les liaisons bi-privées sont représentées par un nom et caractérisées par les connexions et leurs performances (débit effectif et priorité (voir modèle des bus ci-dessous)).

D'autres paramètres choisis en fonction des besoins peuvent également être ajoutés dans le champ "Autres" (Figure 5).

```
Lien  { Nom,
        Connexion [ProcEntrée, ProcSortie]
        Performances [DébitEffectif , Priorité],
        Autres  [ ... ] }
```

Figure 5: modélisation des liaisons bi-privées

3.4.2. Bus

Les bus sont modélisés comme des processeurs particuliers qui ne sont capables d'effectuer qu'une seule primitive particulière: la communication. Chaque processeur connecté au bus l'est alors par l'intermédiaire d'une liaison bi-privée.

Le paramètre "Priorité" de la liaison permet de hiérarchiser les processeurs lors d'un accès au bus. Il est alors possible d'obtenir de manière statistique le débit effectif d'un processeur sur le bus.

Il faut toutefois noter que lors de la réalisation du placement / ordonnancement statique des primitives sur les processeurs, l'occupation d'un bus est connue à chaque instant.

3.4.3. Mémoire partagée

La modélisation des mémoires partagées suit le même principe: la primitive réalisée par le "processeur" est cette fois une fonction unique de "mémorisation / communication". Ce "processeur" joue donc le double rôle de bus et mémoire et chaque processeur connecté à la mémoire l'est par l'intermédiaire d'une liaison bi-privée.

4. MODELISATION D'UN SYSTEME DE TRAITEMENT DU SIGNAL

4.1. Modélisation de l'application

Le domaine du traitement du signal se caractérise en particulier par des algorithmes peu décisionnels et des primitives de granularité grosse ou moyenne.

L'ensemble des primitives et de leurs relations entrées / sorties s'exprime comme un graphe "flot de données synchrone" (Figure 6): ce modèle est en effet particulièrement bien adapté [6] à la description des applications de traitement du signal.

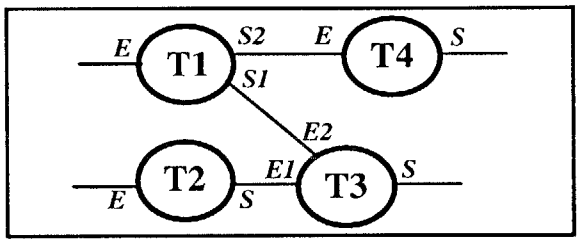


Figure 6: modèle des applications

L'application est donc décrite sous la forme d'un graphe orienté (graphe de calcul) dans lequel les nœuds correspondent aux primitives de traitement et les arcs (Figure 7) aux transferts de données.

```
Liaison  { Nom,
           Connexion [PrimEntrée, PrimSortie]
           Autres  [ ... ] }
```

Figure 7: modélisation des liaisons entre primitives

4.2. Modélisation de l'architecture matérielle

L'architecture matérielle du système de traitement est de manière comparable, modélisée (Figure 8) sous la forme d'un graphe (matériel) dans lequel un nœud correspond à une entité pouvant réaliser une fonction (calcul pour une primitive, communication pour un bus, mémorisation / communication pour une mémoire partagée), les arcs représentant les voies de communication bi-privées entre deux entités.

Les fonctions exécutées par les processeurs élémentaires s'expriment sous la forme cascade ("+" pour exécution séquentielle) des tâches du graphe de l'application, c'est-à-dire des primitives de traitement.

Dans le cas des sous-ensembles, l'opérateur de composition "/" traduit l'exécution parallèle des primitives sur des processeurs différents.

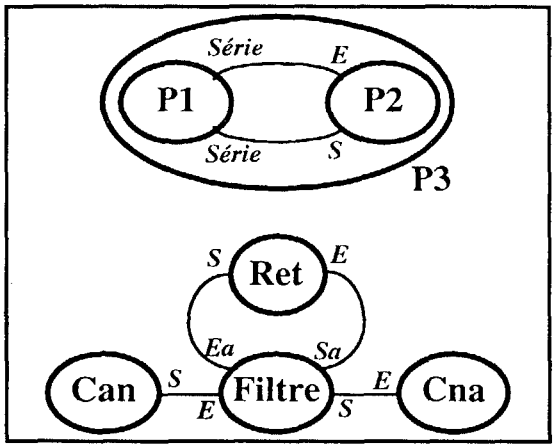
```
Nœud  { NomProc,
        Fonction [ (Prim + ... ) / ... ] }
        Mémoire [ TailleRom, TailleRam ]
        Contraintes [ Coût, Surface, Consommation ],
        Autres  [...] }
```

Figure 8: modélisation du système de traitement

Les paramètres "Mémoire" et "Contraintes", calculés à partir des paramètres correspondant des autres entités de description, permettent de vérifier la faisabilité de l'application sur l'architecture matérielle en cours d'étude.

5. EXEMPLE

L'exemple qui suit correspond à un système P3 comprenant deux processeurs P1 et P2 reliés par deux liaisons série de type bi-privées sur lesquels est implémentée une application de filtrage.




```
Prim      { Can, Imm, 1, 1 , Sort [ (S, 1 , 16) ] }
Prim      { Filtre, Imm, 64 , 2*N, Ent [ (E, 1, 16) , (Ea,
          63, 16) ], Sort [ (S, 1 , 16) , (Sa , 63, 16) ] }
Prim      { Ret, Retard, 63, N , Entrées [ (E, 63, 16) ],
          Sorties [ (S, 63, 16) ] }
Prim      { Cna, Imm, 1, 1 , Entrées [ (E, 1, 16) ] }
Liaison   { L1, Conn [Filtre.E, Can.S]}
Liaison   { L2, Conn [Filtre.Ea, Ret.S]}
Liaison   { L3, Conn [Ret.E, Filtre.Sa]}
Liaison   { L4, Conn [Cna.E, Filtre.S]}
Proc      { P1, Mém (32K, 16K), Perf [40M, DMA, 1.1],
          Ent [ (Série, 1µ, 2M, 1) ],
          Sort [ (Série, 1µ, 2M, 1) ],Cont [50, 4, 250m] }
Proc      { P2, Mém(8K, 1024), Perf [1M, NODMA, 1.8],
          Ent [ (E, 5µ, 2M, 1) ], Sort [ (S, 5µ, 2M, 1) ],
          Cont [6, 3, 75m] }
Proc      { P3, Proc [P1, P2] }
Lien      { A1, Conn [P1.Série, P2.S], Perf [ 2 ] }
Lien      { A2, Conn [P2.S, P1.Série], Perf [ 2 ] }
PriPro    { P1, Can, Perf [ 10 ], Mém [ 4, 1 ] }
PriPro    { P1, Cna, Perf [ 10 ], Mém [ 4, 1 ] }
PriPro    { P1, Filtre, Perf [ 3 ], Mém [ 12, 1 ] }
PriPro    { P1, Ret, Perf [ 1 ], Mém [ 6, 63 ] }
PriPro    { P2, Can, Perf [ 4 ],Mém [ 5, 1 ]}
PriPro    { P2, Cna, Perf [ 4 ], Mém [ 5, 1 ] }
PriPro    { P2, Filtre, Perf [ 7 ], Mém [ 4, 1 ] }
PriPro    { P2, Ret, Perf [ 5 ], Mém [ 4, 1 ] }
Nœud      {P1, Fonc [ ( Filtre + Ret ) ],Mém [18, 64],
          Cont [ 50, 4, 250m ] }
Nœud      {P2, Fonc [ ( Can + Cna ) ],Mémoire [10, 2],
          Cont [ 6, 3, 75m ] }
Nœud      {P3, Proc [P1, P2],
           Fonc [ ( Filtre + Ret ) / ( Can + Cna ) ]
          Mém [28, 68], Cont [ 56, 7, 325m ] }
```

6. CONCLUSION

La structure d'accueil que nous proposons pour modéliser les systèmes de traitement du signal permet de décrire le matériel: cartes, processeurs (en particulier les composants spécifiques), mémoires locales ou globales, ports d'entrées / sorties, supports de communication entre matériels (dont les bus) , l'application traitée (primitives) et leurs interrelations.

Elle permet d'évaluer les performances d'une architecture vis à vis d'une application (temps d'exécution, capacité des mémoires programme et données nécessaires, connectique et débits maximaux des liens de communication, ...) et d'en déterminer et / ou vérifier les contraintes topologiques et énergétiques ainsi que les coûts associés.

L'exploitation de cette structure de données par différents outils (synthèse d'architectures matérielles, placement / ordonnancement, vérification de conformité à des spécifications, à des contraintes de réalisation, ...) peut alors permettre à un utilisateur de disposer d'un outil de conception d'une architecture optimisée pour son application.

Références

[1] Y. Sorel. Langages synchrones et exécutifs distribués optimisés. *Conf. Adéquation Algorithme Architecture, Lannion 14 et 15 Sept. 1992.*

[2] D. Ferrari. Considerations on the Insularity of Performance Evaluation. *IEEE Trans. on Soft. Eng., SE-12(6):p678-683, June 1986.*

[3] O. Sentieys, H. Dubois, J. L. Philippe, E. Martin. GAUT, un outil de synthèse de cœur de processeurs dédiés au traitement du signal. *Conf. Adéquation Algorithme Architecture, Lannion 14 et 15 Sept. 1992.*

[4] Benchmarking and performance modelling of high performance computers. *Parallel Computing, vol. 17, 1991.*

[5] B. S. Haroun and M. I. Elmasry. Architectural Synthesis for DSP Silicon Compilers. *IEEE Trans. on computer-aided design, vol. 8, no 4, April 1989.*

[6] A. Benveniste. The synchronous approach to reactive and real-time systems. *Proceedings of the IEEE, VOL. 79, no 9, September 1991.*