Estimation de la consommation d'un algorithme C par analyse fonctionnelle

Johann Laurent, Nathalie Julien, Eric Senn, Eric Martin

Estimation de la consommation d'un algorithme C par analyse fonctionnelle

Johann Laurent, Nathalie Julien, Eric Senn, Eric Martin
Laboratoire LESTER
Centre de recherche rue Saint Maude
BP 92116 56321 LORIENT CEDEX
Tel :02/97/87/45/28 Fax :02/97/87/45/00
Email : prénom.nom@univ-ubs.fr

Résumé

Une méthode d'estimation de la consommation au niveau C pour processeurs commerciaux a été développée. Cette méthode est basée sur une analyse fonctionnelle de la consommation du processeur dont le résultat est un modèle de consommation de la cible. Ce modèle décrit le comportement en puissance du processeur à l'aide de paramètres algorithmiques et de configuration. Certains de ces paramètres peuvent être directement prédits à partir de l'algorithme C avec des connaissances simples sur l'architecture cible et le compilateur. Cette méthode, appliquée au TMS320C6201, fournit des résultats d'estimation avec une erreur maximale de 6% par rapport aux mesures pour des algorithmes classiques de traitement du signal et de l'image. De plus, une 'carte de consommation' informe le concepteur sur les variations de consommation de l'algorithme étudié et ses limites.

1. Introduction

Aujourd'hui, les applications portables se démocratisent ; elles requièrent de plus en plus de puissance de traitement ce qui augmente la consommation d'énergie de tels systèmes. Pour éviter d'avoir à sur dimensionner les batteries, il faut donc optimiser la consommation de ces applications. Ces optimisations peuvent se faire à tous les niveaux d'abstraction mais il existe un point commun entre elles : le besoin d'avoir une métrique de consommation rapide et fiable sans devoir faire de mesures physiques.

Les développeurs utilisent de plus en plus le langage C pour décrire leurs applications ; il est donc intéressant d'estimer directement la consommation de l'application à ce niveau. Ceci évite de passer par une phase de compilation donc augmente l'interactivité entre l'estimation de consommation et le développement de l'application. Le concepteur peut alors faire le meilleur choix d'algorithme pour une cible donnée ou s'il possède une librairie de cibles, choisir la meilleure pour son algorithme.

Des outils existent déjà pour les cœurs de processeur ou pour les ASICs (Wattch, Jouletrack, outils constructeurs etc.). En revanche, très peu d'études sont réalisées sur les processeurs commerciaux comme les DSP. Le problème avec ces cibles est le manque d'informations disponibles sur l'architecture et son fonctionnement. La plupart des outils et des méthodes d'estimation existantes sont basés sur une approche au niveau assembleur [1]. Avec cette méthode, il faut mesurer la consommation de tout le jeu d'instructions ainsi que de toutes les inter-instructions, ce qui représente un nombre prohibitif de mesures pour des architectures complexes récentes (ex : 1176 mesures pour un DSP 56K[2] et au minimum 71^8 pour un TMS320C6201).

La méthode que nous avons développée est basée sur une analyse fonctionnelle de la cible du point de vue consommation (Functional Level Power Analysis : FLPA) ; elle est indépendante du niveau d'abstraction (assembleur ou C). Grâce à cette analyse, un nombre limité de mesures suffit pour déterminer le modèle de consommation de la cible. De plus, notre méthode prend en compte toutes les fonctions du processeur que ce soit le contrôle du pipeline, les unités de traitement, les mémoires internes ainsi que les défauts de cache, ce qui n'est pas le cas avec les méthodes au niveau assembleur ; il faut alors compléter le modèle [6].

Nous ne présenterons dans cet article que l'estimation au niveau C, l'estimation au niveau assembleur ayant déjà fait l'objet de publications [3]. Le paragraphe 2 présentera la méthodologie d'estimation ainsi que le modèle du processeur. La méthode d'estimation au niveau C sera présentée dans le paragraphe 3 et enfin des applications sur des algorithmes de traitement du signal seront présentées dans le paragraphe 4 pour montrer la validité de notre approche.

2. Analyse de Puissance au Niveau Fonctionnel

2.1 Méthodologie d'estimation

La méthodologie d'estimation de la consommation de puissance, représentée sur la Figure 1, est constituée de deux parties : la définition du modèle et le processus d'estimation.

- La définition du modèle de puissance du processeur est réalisée une seule fois par cible. Elle est basé sur l'analyse fonctionnelle, d'un point de vue consommation, de l'architecture cible (dans notre cas un DSP). Cette analyse nous permettra de déterminer un modèle de puissance basé sur des lois de consommation qui représentent le comportement en courant du cœur du DSP. Ces lois sont des fonctions mathématiques déterminées à partir d'un

nombre réduit de mesures physiques réalisées sur la cible et dépendant de paramètres algorithmiques et de configuration. La FLPA permet de déterminer quels sont les paramètres pertinents d'un point de vue consommation pour un processeur donné. Les paramètres algorithmiques sont des ratios (variant de 0 à 1) qui représentent le taux d'activité entre chaque bloc fonctionnel du DSP ; par exemple, le taux de parallélisme, le taux de défaut de cache… Les paramètres de configuration sont définis par le concepteur.

- Le processus d'estimation est lui réalisé à chaque fois que la consommation d'un algorithme doit être déterminée. Au niveau C, les paramètres algorithmiques sont estimés en utilisant un modèle de prédiction. Il suffit ensuite d'utiliser les lois de consommation établies pour l'architecture cible pour connaître la consommation de l'application.

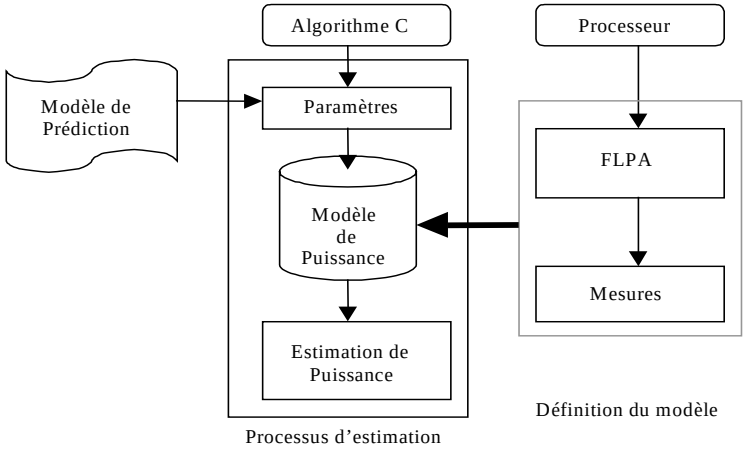

Fig. 1. Méthodologie d'estimation

2.2 FLPA et modèle d'estimation du TMS320C6201

La FLPA a été appliquée sur le DSP TMS320C6201 de Texas Instruments. Ce processeur possède une architecture complexe : pipeline profond (11 étages), jeu d'instructions VLIW (256 bits), possibilité de parallélisme (jusqu'à 8 instructions), deux mémoires internes (données et programme) ainsi qu'un EMIF (External Memory InterFace) permettant les accès aux mémoires externes [5]. Après avoir réalisé la FLPA du processeur, nous connaissons tous les paramètres qui influent sur la consommation (l'analyse FLPA est détaillée dans [3]). Ces paramètres sont présentés sur la Figure 2.

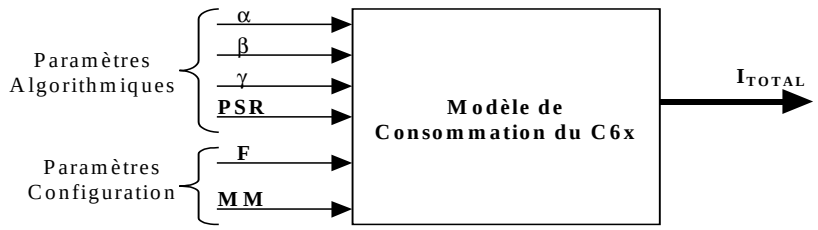

Fig.2. Modèle de consommation du C6x

- Il existe deux types de paramètres : les paramètres algorithmiques et les paramètres de configuration. Les valeurs des paramètres algorithmiques dépendent de l'algorithme à estimer : α représente le taux de parallélisme du programme, β le taux d'unité de traitement, γ le taux de défaut de cache et enfin PSR représente le taux de rupture de pipeline. Les paramètres de configuration sont la fréquence d'horloge du processeur et le mode mémoire.

Le C6x charge 8 instructions en même temps ; celles-ci forment un paquet de chargement (FP). Toutes les instructions exécutées en parallèle forment un EP. Chaque FP peut être constitué de plusieurs paquets d'exécution (EP). Les deux paramètres α et β sont alors calculés de la façon suivante :

$$\alpha = \frac{NFP}{NEP} \leq 1; \qquad \beta = \frac{1}{8}\frac{NPU}{NEP} \leq 1 \quad (1)$$

NFP et NEP représentent respectivement le nombre moyen de FP et de EP. NPU représente le nombre moyen d'unités fonctionnelles utilisées (toute opération sauf l'instruction NOP).

3. Méthode d'estimation

Pour utiliser la méthode au niveau C, il faut que l'algorithme soit constitué de boucles de taille suffisante pour pouvoir négliger tout ce qui n'est pas inclus dans la ou les boucles. Cette condition est vérifiée dans la plupart des algorithmes de traitement du signal et de l'image.

Pour estimer la consommation au niveau C, il faut déterminer les valeurs des paramètres algorithmiques. Pour déterminer exactement PSR et γ, il faut réaliser une trace du programme. En revanche, nous pouvons estimer les paramètres α et β, à partir du source C, en prédisant le nombre de paquets de chargement, d'exécution et le nombre d'unités de traitement utilisés par l'application.

Nous avons déterminé 4 modèles de prédiction basés sur une connaissance minimale de l'architecture cible et du compilateur :

- Le modèle SEQ : toutes les instructions sont traitées séquentiellement.
- Le modèle MAX : le compilateur exploite entièrement les possibilités de l'architecture
- Le modèle MIN : il n'autorise pas les load et/ou store en parallèle. En revanche, les opérations de calculs peuvent être effectuées en parallèle.
- Le modèle DATA : exploite les possibilités architecturales mais, les instructions load et/ou store en parallèle ne sont possibles que si et seulement si elles se font sur des données différentes.

Lorsque l'application est constituée de plusieurs fonctions et/ou boucles, nous prédisons les valeurs locales de α et β pour chaque fonction et/ou boucle. Les paramètres globaux α et β de l'algorithme sont obtenus en calculant une moyenne pondérée des valeurs locales. Un exemple de prédiction des paramètres est présenté dans [4].

4. Applications

Dans un premier temps, nous allons montrer les résultats obtenus avec notre méthode sur des applications de traitement du signal. Ces résultats montrent la validité de notre démarche puisque l'erreur entre la mesure de consommation et l'estimation est au maximum de 6%. Ensuite, nous proposerons une application de notre méthode à l'exploration de la consommation d'un algorithme.

4.1 Validation de l'estimation

Différentes applications ont été traitées : un filtre FIR, un filtre LMS bi voies, une FFT, une transformée en ondelette et un codeur de voix pour GSM. Les résultats sont donnés pour différents modes mémoire (M :mapped, C : cache, B : bypass) et différents placements des données (INT :interne ou EXT: externe).

Le but de ces applications est de valider notre méthode afin de montrer sa précision ; c'est pour cela que nous avons utilisé les valeurs réelles du PSR et de γ pour chaque application (ces valeurs sont issues d'un profiling dynamique). De plus, la fréquence d'horloge du DSP est de 200MHz pour toutes les applications.

Le tableau 2 présente les résultats des mesures effectuées (le temps d'exécution, la puissance et l'énergie consommée par les applications) ainsi que la comparaison entre ces mesures et les estimations obtenues pour les 4 modèles de prédictions.

Nous pouvons remarquer que le modèle SEQ donne les résultats les moins fiables puisque ce modèle ne prend pas en compte l'architecture de la cible. Le modèle MAX surestime toujours la consommation alors que le modèle MIN la sous-estime ; ces 2 modèles nous donnent les bornes supérieures et inférieures de consommation. Un seul cas donne des résultats contraires aux autres, c'est l'utilisation du mode mémoire (MM) Bypass ; ce mode mémoire génère un fort taux de rupture de pipeline car toutes les instructions sont chargées de la mémoire externe. Il faut donc au minimum 8 temps de cycle pour charger un FP au lieu de 1 en mémoire interne.

Les résultats les plus précis sont obtenus en utilisant le modèle DATA. Ce modèle engendre une erreur moyenne de 4% et une erreur maximale de 6% par rapport aux mesures. Ces résultats sont comparables à ceux obtenus avec une méthode au niveau assembleur.

ALGORITHME			**MESURES**			**ESTIMATION**				
APPLICATION	**M M**	**INT/ EXT**	**T_{EXE}**	**P(W)**	**ENERGIE**	**SEQ**	**MAX**	**MIN**	**DATA**	
FIR	M	INT	6.885µs	4.5	30.98µJ	2.745	4.725	3.015	4.725	+5%
FFT	M	INT	1.389ms	2.65	3.68mJ	2.36	2.97	2.57	2.58	-2.5%
LMS	B	INT	1.847s	4.97	9.18J	5.02	5.12	5.07	5.12	+3%
LMS	C	INT	165.75ms	5.665	939mJ	2.55	6	4.76	6	+6%
DWT 64*64	M	INT	2.32ms	3.755	8.71mJ	2.82	4.24	3.27	3.53	-6%
DWT 64*64	M	EXT	9.19ms	2.55	23.46mJ	2.295	2.63	2.4	2.46	-3.5%
DWT 512*512	M	EXT	577.77ms	2.55	1.473J	2.27	2.61	2.37	2.45	-4%
EFR VOCODER	M	INT	39µs	5.0775	198µJ	2.54	5.636	3.86	5.13	+1%
ERREUR MOYENNE						25%	7%	13%	***4%***	

Tableau 2 Résultats de mesures et d'estimation des applications

4.2 Exploration de la consommation de puissance d'un algorithme

Sans connaissance des paramètres PSR et γ, nous pouvons déterminer une carte de consommation de l'application. Cette carte permet au concepteur de visualiser la variation de consommation de son algorithme.

Comme exemple, nous allons présenter les résultats de l'exploration de la consommation pour le codeur de voix. La Figure 3 montre les résultats pour cette application lorsque le mode Mapped est utilisé. Dans ce cas, seul le PSR peut varier puisque γ=0 (pas de défaut de cache). La Figure 4 montre, elle, les résultats de cette même application mais cette fois ci lorsque le mode Cache est utilisé.

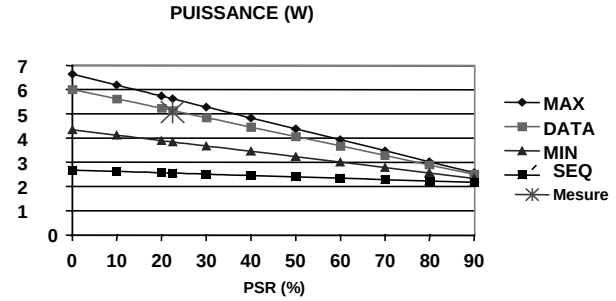

Fig.3. Exploration de la consommation en mode Mapped

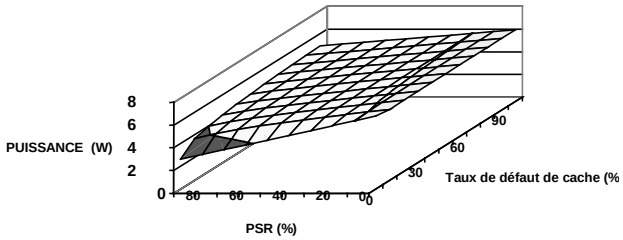

Fig.4. Exploration de la consommation en mode Cache

Certaines parties de cette carte ne sont pas réalistes comme par exemple le fait d'avoir un γ non nul et un PSR nul : dès lors qu'un défaut de cache est généré, il y a forcément un taux de rupture de pipeline puisque les instructions doivent être chargées de la mémoire externe.

Cependant, avec les processeurs actuels, il est rare que le taux de défaut de cache (programme) soit non nul vu la taille des mémoires internes (64Ko dans la plupart des cas). De plus, nous pouvons connaître l'ordre de grandeur du PSR puisque celui ci est, en particulier, dépendant du placement des données en mémoire. Cela permet au concepteur de localiser sur la carte de consommation la zone probable de variation de son algorithme.

L'exploration de la consommation permet au concepteur de vérifier que son algorithme respecte les contraintes de l'application. Si ces contraintes ne sont pas respectées, il peut connaître quelles sont les parties de son code les plus consommatrices. Il suffit pour cela d'analyser les valeurs locales de α et β : plus ces valeurs sont grandes et plus cette partie de code est optimisée. Cette connaissance lui permet d'optimiser seulement les parties de code critiques et lui évite une optimisation globale inutile.

5. Conclusion

Nous avons montré la faisabilité d'une estimation précise de consommation au niveau C. Pour réaliser cette estimation, il n'est pas nécessaire d'avoir une connaissance approfondie de l'architecture cible mais seulement une connaissance fonctionnelle du compilateur et de l'architecture cible. Le but de cette méthode est de donner au concepteur un retour rapide de la consommation de son algorithme. Pour cela nous pouvons donner une exploration de la consommation lorsque le taux de défaut de cache et/ou le taux de rupture de pipeline ne sont pas connus. Les travaux en cours sont le développement d'un outil automatique ainsi que l'application de cette méthode sur d'autres processeurs. Enfin, nous pensons appliquer cette méthode sur plusieurs exemples d'adéquation algorithme architecture.

Références :

[1] V. Tiwari, S. Malik, A. Wolfe "Power analysis of embedded software: a first step towards software power minimization." *IEEE Transaction on. VLSI Systems, Décember 1994.*

[2] B. Klass, D.E. Thomas, H. Schmit, D.F. Nagle "Modeling Inter-Instruction Energy Effects in a Digital Signal Processor" *Power Driven Microarchitecture Workshop ISCA, 1998.*

[3] J. Laurent, E. Senn, N. Julien, E. Martin "High Level Energy Estimation for DSP Systems" *in International. Workshop on Power And Timing Modeling, Optimization and Simulation,septembre 2001, pp311-316.*

[4] J. Laurent, N. Julien, E. Senn, E. Martin "Power/Energy Estimation a C-Algorithm based on Functional Level Power Analysis on a Digital Signal Processor" *in International Symposium on High Performance Computing, mai 2002.*

[5] Documentation Texas Instruments Technical Brief (SPRU 197D) février 1999.

[6] L. Benini, D. Bruni, M. Chinosi, C. Silvano, V. Zaccaria, R. Zafalon "A Power Modeling and Estimation Framework for VLIW-based Embedded Systems," in *Proc. Int. Workshop on Power And Timing Modeling, Optimization and Simulation PATMOS,* Sept. 2001, pp. 2.3.1-2.3.10.