Conception de circuits dédiés à une classe d'applications (ASIP)

compromis consommation - performances - flexibilité

Jean-Gabriel Cousin, Daniel Chillet et Olivier Sentieys

LASTI - ENSSAT - Université de Rennes
6 rue de Kérampont, BP 447
22305 Lannion, FRANCE
eMail: cousin@enssat.fr
http://www.enssat.fr/RECHERCHE/ARCHI

RÉSUMÉ

Devant la croissance des niveaux d'intégration et l'émergence récente des systèmes embarqués, les circuits intégrés VLSI[1] dédiés aux applications de traitement du signal et d'image, doivent être performants et consommer peu. Aussi, une exploration efficace de l'espace temps-surface-consommation paraît nécessaire, demandant l'utilisation d'outils de conception haut-niveau à rétroaction rapide.
Dans ce papier, nous présentons succinctement une classe de processeurs programmables performants, les ASIPs[2] VLIW[3], ainsi qu'une méthode haut-niveau d'estimation de la consommation électrique qui leur est appliquée.

ABSTRACT

Due to increasing integration levels and embedded systems, VLSI[1] integrated circuits that have been specifically designed for data-dominated signal processing algorithms, must be computing and power efficient. Then, fast exploration of the time-area-power space is essential and calls for high-level design tools with fast turnaround. In this paper, we briefly present a class of programmable processors, the VLIW[3] ASIP[2] architectures, and describe our high-level power estimator based on them.

1 Introduction

Les composants électroniques disponibles aujourd'hui et les possibilités de la technologie des circuits intégrés VLSI[1] permettent, et permettront de plus en plus, la conception matérielle de systèmes très complexes (satellite, radar, télévision numérique, GSM, compression vidéo, ...). Dans ce domaine, les efforts de développement et d'implantation dans une puce électronique étaient encore récemment focalisés sur les critères de temps d'exécution et de surface. Or, depuis quelques années, si les circuits ainsi conçus sont certes performants, leur consommation électrique relativement élevée tend de plus en plus à limiter cette conception. De ce fait, afin de répondre aux multiples contraintes imposées par l'état du marché (*time to market*), les circuits actuels et futurs doivent être (rester) performants tout en consommant peu.

Aussi, l'objet de notre recherche est l'analyse de la consommation électrique, ainsi que des performances de certains processeurs. Une telle estimation, effectuée à un haut niveau, permettra au concepteur de circuit de connaître tôt, dans le cycle de conception, la consommation électrique dissipée par l'application à mettre en œuvre : le concepteur pourra alors chercher à l'optimiser, soit en transformant l'algorithme de l'application, soit en modifiant l'architecture du processeur hôte (rétroaction).

Cette approche actuellement en développement (cf. figure 1), est appliquée sur des processeurs exécutant des applications de traitement du signal et d'image (algorithmes de filtrage, transformées de Fourier rapide et en cosinus, etc.), applications flot de données gourmandes en calcul et constamment évolutives. L'architecture de ces processeurs cibles est donc conçue de manière hétérogène, c'est-à-dire composée de parties dédiées performantes et de parties programmables capables de suivre l'évolution du marché, allongeant de ce fait la durée de vie généralement courte de ces circuits. Il s'agit des ASIPs[2] [4] à modèle VLIW[3].

Dans la section 2, nous présentons brièvement le concept de l'ASIP, i.e. ses domaines de prédilection et son architecture. Puis, dans la section 3, nous détaillons notre méthodologie d'estimation de la consommation électrique pour ces processeurs.

2 Les ASIPs : une analyse

L'utilisation de processeurs programmables est divisée entre deux classes d'applications, le calcul intensif et les systèmes embarqués, ces derniers incluant les DSPs[4], les microcontrôleurs, les cœurs de processeurs et les ASIPs. Les

[1] VLSI : *Very Large Scale Integration*

[2] ASIP : *Application Specific Instruction-set Processor*
[3] VLIW : *Very Long Instruction Word*
[4] DSP : *Digital Signal Processor*

applications nécessitant l'utilisations de tels processeurs sont essentiellement liées aux

- multimédia : vidéophone, compression vidéo MPEGx, HDTV, DVB, graphique 3D, jeux vidéo ;
- communications sans fil : GSM ou DECT européen, téléphone cellulaire nord-américain, assistant numérique personnel ;
- télécommunications : satellite, ATM.

Les industriels ont déjà développé de nombreux produits pour chacune de ces trois classes (cf. tableau 1). Une liste exhaustive de tels circuits peut être trouvée à partir des papiers de Paulin et Goossens [10, 5].

Concepteur	Processeur	Applications
Philips	TriMedia, VLIW ASIP	codeur/décodeur MPEG1, décodeur MPEG2, vidéophone H261
SGS-Thomson	VLIW DSP ASIP	Vidéophone, estimation de mouvement
France Telecom	LIW DSP ASIP basse consommation	Terminal ISDN Terminal GSM
Alcatel	ASIP DSP 16 bit	GSM
Philips	"KISS" DSP 16 bit "EPICS" DSP config.	GSM DECT
SGS-Thomson	"D950" DSP 16 bit + co-processeurs	GSM, cellulaire
AT&T	famille d'ASIP de type DSP16	GSM, cellulaire, modem

TAB. 1 — Exemples de processeurs embarqués

L'ASIP est un compromis entre deux classes de circuits, les processeurs généraux de traitement du signal (DSPs classiques) et les circuits intégrés dédiés à une application, les ASICs[5]. L'ASIP allie les avantages des architectures dédiées, c'est-à-dire l'ajout d'unités fonctionnelles spécifiques, donc performantes, et flexibilité de programmation de par son jeu d'instructions (plus restreint que les DSPs classiques). Il est en effet judicieux de conserver une certaine souplesse de programmation car cela offre, d'une part au processeur, la possibilité de suivre l'évolution du marché, lui évitant ainsi une durée de vie trop courte (inconvénient de l'ASIC), et d'autre part au concepteur de circuit, de pallier aux changements de dernière minute dans les dernières phases de la conception. L'ASIP est donc par définition un circuit spécifique programmable, reconfigurable rapidement.

[5] ASIC : *Application Specific Integrated Circuit*

Le modèle VLIW est une architecture parallèle basée sur plusieurs unités de traitement (UTs), exécutant simultanément différentes opérations (parallélisme des UTs). L'apport de ces opérations au processeur se fait par une macro-instruction VLIW contenant le code opération et opérande, code relatif à chaque UT (cf. figure 2).

L'ASIP apparaît alors comme un processeur dédié à une classe d'applications.

Le problème restant ouvert est la maîtrise de la conception de tels circuits. On connaît la faible efficacité des compilateurs *C* pour les architectures de type DSP, justifiant l'utilisation de l'assembleur dans 90% des lignes de codes de programme. Les ASIPs nécessitent donc un développement d'outils permettant leur conception, programmation, et co-simulation à partir d'un langage de haut niveau (C, VHDL), tout en conservant une efficacité raisonnable (taille du code généré, performance).

3 Méthodologie

Pour obtenir une telle efficacité, une exploration de l'espace temps-surface-consommation, appuyée par des outils de conception haut-niveau, est donc nécessaire. Nous développons actuellement une méthodologie d'estimation de la consommation électrique qui, effectuée à un haut niveau, permettra au concepteur de circuit de prendre en compte ce critère récent (excepté dans le domaine des circuits liés à des batteries (portables, régulateur cardiaque, etc.) ou des circuits sensibles aux faibles tensions) et de concevoir alors des circuits performants à faible consommation. Cette approche, représentée par la figure 1, se déroule en trois phases (numérotées 1, 2 et 3). Fort de nos acquis dans la conception de circuit dédié (ASIC), la phase 1, approfondie dans la section ci-après, détaille notre technique de conception des ASIPs effectuée autour de l'outil de synthèse architecturale GAUT[6] [8] tandis que les phases 2 et 3, regroupant l'estimation de la consommation, sont décrites dans la section 3.2.

3.1 Conception du circuit

Dans la phase 1 de notre méthodologie, il s'agit de modéliser et de concevoir des ASIPs par GAUT, notre outil de conception haut-niveau. À partir d'une description comportementale en langage VHDL d'une application cible à implanter (FIR, DCT, FLMS ...), GAUT extrait sous une contrainte de temps (d'exécution) le parallélisme inhérent à l'application et le traduit sous la forme d'un CDFG[7] ordonnancé (ordonné temporellement) ; il façonne ensuite le modèle architectural (ex. figure 2), c'est-à-dire le cœur du processeur (chemin de données ou UT), par le biais d'une bibliothèque formelle de composants (cf. figure 1), et cible le taux maximal d'utilisation du modèle ; il génère en conclusion une description structurelle (RTL[8]) en VHDL du circuit. La partie contrôle est alors déduite et décrite sous la forme de machines d'états ou de contrôleurs locaux.

[6] GAUT :*Génération Automatique d'Unité de Traitement*
[7] CDFG : *Control Data Flow Graph*
[8] RTL :*Register Transfer Level*

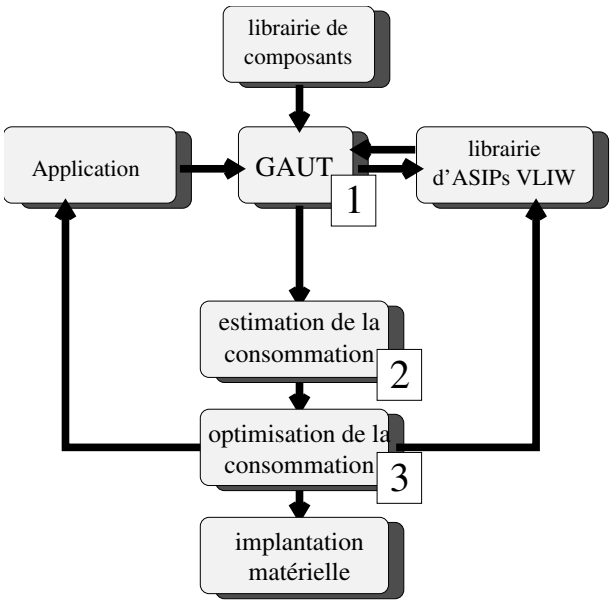

FIG. 1 — Méthodologie d'estimation de la consommation électrique

Outre la création d'une bibliothèque de circuits (librairie d'ASIPs de la figure 1), cette première phase nous procure des métriques sur la rapidité de calcul (la fréquence d'exécution), la surface (de silicium) du circuit conçu et la souplesse du jeu d'instructions de l'unité de contrôle. L'architecture structurelle obtenue est donc un compromis entre les performances et la flexibilité inhérentes au processeur.

À ce niveau de la méthodologie, le critère de consommation électrique n'est pas encore pris en compte. Deux possibilités sont alors offertes au concepteur : soit celui-ci désire estimer la consommation de l'application ayant servi de point de départ dans la conception du modèle architectural, et dans ce cas il exécute la méthode de la section 3.2 directement sur le CDFG, soit il veut estimer une autre application, et doit avant tout implanter la spécification algorithmique de cette nouvelle application sur le processeur hôte (adéquation algorithme-architecture). Aussi, une nouvelle phase d'ordonnancement est nécessaire : elle est actuellement en développement. La description VHDL de l'application est transformée, par compilation, en un CDFG, qui est alors ordonnancé par un algorithme par liste [9, 3] (*list scheduling*). L'architecture du processeur cible étant connue (conçue par GAUT en phase 1), les éléments ou ressources qui la composent et les interconnexions le sont également : l'ordonnancement s'effectue donc à ressources contraintes [2].

3.2 Critère de consommation

Nous avons défini notre méthode d'estimation de la consommation au niveau fonctionnel car, c'est à ce niveau que les gains espérés, après optimisation de la consommation, sont les plus prometteurs [1, 11]. À ce niveau, la puissance consommée est donnée par l'expression suivante,

$$P_{\text{consommée}} = \sum_{\forall i} N_i . C_{\text{ressource}_i} . V_{dd}^2 . f \qquad (1)$$

où N_i est le nombre d'accès à la ressource i (composant i), $C_{\text{ressource}_i}$ la capacité équivalente de commutation par accès, V_{dd} la tension d'alimentation et f la fréquence d'horloge. Les puissances respectivement statique et de court-circuit sont négligées à ce niveau.

Le modèle de l'architecture est connu (approche à ressources contraintes de la phase 1), les paramètres V_{dd}, f et $C_{\text{ressource}_i}$ sont donc fixés : f est déterminée lors de la conception du circuit en phase 1 ; V_{dd} et $C_{\text{ressource}_i}$ constituent deux des paramètres inhérents aux composants de la librairie [6] utilisée par GAUT. Par une lecture rapide du CDFG ordonnancé, nous estimons le nombre d'accès N_i de chaque ressource i et obtenons alors, par l'équation 1, la consommation $P_{\text{consommée}}$ de l'application implantée dans l'ASIP.

Le coût global de la consommation obtenu par cette analyse doit être minimisé, c'est la phase 3 de notre méthodologie. Le concepteur pourra appliquer des transformations algorithmiques et algébriques sur l'application afin de minimiser N_i, seule variable de l'équation 1, et comparer différentes versions d'un même algorithme par exemple. Les transformations les plus usitées sont le réordonnancement des instructions, l'élimination du code mort, les commutativité et distributivité sur les opérations, la minimisation des transferts mémoire, etc. [9, 12, 13].

Mais, si notre approche est à ressources contraintes, nous modifierons également le modèle architectural et le jeu d'instructions du processeur cible (rétroaction, cf. figure 1), complétant de ce fait notre librairie de circuit par des processeurs faible-consommation capables d'exécuter l'application dans le temps imparti (cf. section 2). Des transformations incluant le *retiming* et le *pipeline* seront alors applicables sur le CDFG.

3.3 Support d'études

Pour tester et valider notre méthode d'estimation de la consommation, nous avons développé un modèle architectural VLIW de 64 bits (taille de la macro-instruction, cf. figure 2), dédié à une classe d'applications : algorithmes de filtrage dans les domaines temporel et fréquentiel, annulation d'écho acoustique et compression vidéo. Ce modèle simple communique avec l'environnement externe via deux ports, l'un série, l'autre parallèle.

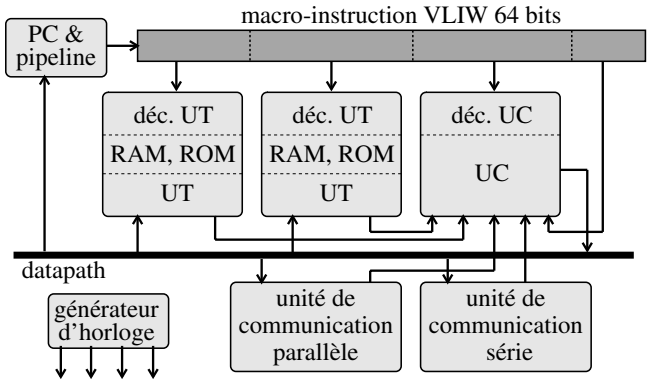

FIG. 2 — ASIP VLIW développé

L'architecture est *pipeline* (parallélisme temporel) et super-

scalaire : elle s'appuie sur deux unités de traitement (UTs) capables d'exécuter en parallèle des opérations *Mac* (Multiplication - ACcumulation) et *papillon*, respectivement en 1 et 9 cycles. Le *papillon* est l'opération élémentaire de la transformée de Fourier rapide, FFT, soit le produit de deux nombres complexes suivi d'une addition et d'une addition-soustraction, d'où l'architecture de la figure 3. Chaque UT est localement connectée à une ROM contenant les coefficients, poids ou table de cosinus, et une RAM pour les échantillons. Cet ASIP contient donc deux RAMs et deux ROMs. Le bloc UC gère le multiplexage du bus interne (*datapath*) du circuit.

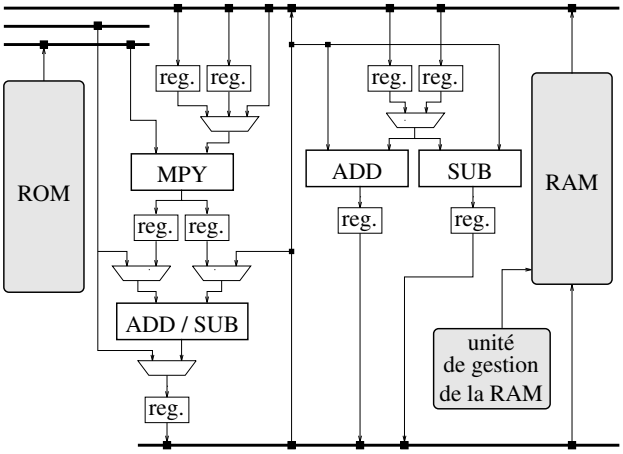

FIG. 3 — Architecture interne d'une UT

Ce prototype a été synthétisé, placé et routé par l'outil de synthèse logique COMPASS, puis fondu en technologie 0,7μm (ES2). Un homologue basse-consommation est en cours de développement, composée de la même architecture superscalaire.

4 Conclusion et perspective

Nous avons présenté rapidement dans ce papier le principe d'une méthode d'estimation à ressources contraintes de la consommation électrique, appliquée à des processeurs dédiés à une classe d'applications, les ASIPs VLIW. Cet estimateur en cours de développement s'applique à un haut niveau. De ce fait, tôt dans la conception, le concepteur connaîtra la consommation de l'application mise en œuvre et pourra l'optimiser.

À cet effet, nous estimerons par notre approche la consommation dissipée par différents algorithmes de traitement du signal (LMS, FLMS, FIR, FFT, DCT ...), implantés dans nos ASIPs récemment développés. Des comparaisons consommation - performances seront alors effectuées entre nos estimations, une simulation logique du modèle architectural (par l'outil POWERCALC sous COMPASS) et une mesure du courant sur le circuit réel [12].

Nos recherches futures porteront d'une part sur les moyens de minimiser cette consommation, c'est-à-dire sur le choix, voire l'ordre, des techniques d'optimisation à appliquer, et d'autre part, sur l'amélioration de l'ordonnancement (cf. section 3.1) afin d'obtenir l'optimalité du code pour une architecture cible [9, 7].

Références

[1] A.P. Chandrakasan and R.W. Brodersen. *Low Power Digital CMOS Design*. Kluwer Academic Publishers, 1996.

[2] F. Depuydt, G. Goossens, and H. De Man. Scheduling with register constraints for DSP architectures. *INTEGRATION, the VLSI Journal*, (18) :95–120, 1994.

[3] D. Gajski, N. Dutt, A. Wu, and S. Lin. *High-Level Synthesis : Introduction to Chip and System Design*. Kluwer Academic Publishers, 1993.

[4] G. Goossens, J. Van Praet, D. Lanneer, and W. Geurts. Programmable Chips in Consumer Electronics and Telecommunications : Architectures and Technology. *EUROCHIP Source on Methods and Tools for Digit System Design*, pages 1–30, September 1995.

[5] G. Goossens, J. Van Praet, D. Lanneer, W. Geurts, A. Kifli, C. Liem, and P. Paulin. Embedded Software in Real-Time Signal Processing Systems : Design Technologies. *Proc. of the IEEE*, 85(3) :436–454, March 1997.

[6] P.E. Landman and J.M. Rabaey. Black-Box Capacitance Models for Architectural Power Analysis. In *Proc. of Int. Workshop on Low-Power Design*, April 1994.

[7] S. Liao, S. Devadas, K. Keutzer, S. Tjiang, and A. Wang. Code Optimization Techniques for Embedded DSP Microprocessors. In *Design Automation Conference*, pages 599–604, 1995.

[8] E. Martin, O. Sentieys, and J-L. Philippe. Synthèse architecturale de cœur de processeurs de traitement du signal. *Technique et science informatiques*, 13(2) :251–279, avril 1994.

[9] P. Marwedel and G. Goossens. *Code Generation for Embedded Processors*. Kluwer Academic Publishers, 1995.

[10] P. Paulin, C. Liem, M. Cornero, F. Naçabal, and G. Goossens. Embedded Software in Real-Time Signal Processing Systems : Application and Architecture Trends. *Proc. of the IEEE*, 85(3) :419–435, March 1997.

[11] J.M. Rabaey and M. Pedram. *Low Power Design Methodologies*. Kluwer Academic Publishers, 1996.

[12] V. Tiwari, S. Malik, A. Wolfe, and M.T-C. Lee. Instruction Level Power Analysis and Optimization of Software. *Jour. of VLSI Signal Processing*, 1996.

[13] S. Wuytack, V.M. Catthoor, and H.J. De Man. Transforming Set Data Types to Power Optimal Data Structures. *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, 15(6) :619–629, June 1996.