Estimation probabiliste de la complexité de circuits VLSI pour le traitement du signal

J.Ph. Diguet[1,2], O. Sentieys[1], J.L. Philippe[3] et E. Martin[3]

[1] LASTI - ENSSAT - Université de Rennes, 6 rue de Kérampont, 22300 Lannion, France
eMail: sentieys@enssat.fr; http://www.enssat.fr/RECHERCHE/ARCHI
[2] IMEC, Kapeldreef 75, B-3001 Leuven, Belgique - Programme Lavoisier
[3] LESTER - Université de Bretagne Sud, 10 Rue Le Coat Saint Haoen, 56100 Lorient, France

RÉSUMÉ

Afin de raccourcir le cycle de conception d'un système VLSI, il est nécessaire de pouvoir estimer les performances (temps, coût, consommation) de différentes solutions architecturales et algorithmiques au niveau d'abstraction le plus élevé. Nous présentons une nouvelle approche d'estimation dynamique de la complexité matérielle, appliquée aux architectures pipelines sous contrainte de temps. Elle se situe au niveau algorithmique, tout en restant indépendante de la méthode de synthèse qui sera choisie ultérieurement. Nous employons une méthode probabiliste prenant en compte réellement les contraintes entre opérations, dans le but de guider le choix des transformations et des algorithmes impliqués dans la spécification. Enfin, nous présentons des résultats d'estimation, et les comparons avec des résultats de synthèse architecturale.

ABSTRACT

In order to decrease the design time of a VLSI system, it is necessary to estimate the performances (time, cost, power) of several architectural or algorithmic solutions, at the highest level of abstraction. This paper introduces an approach that aims to provide the designer with information to quantify the hardware complexity in order to guide the designer in his transformation choices. The method is based on probabilities, focuses the whole set of resources, and takes into account the real dependencies between operations. It firstly enables to combine the estimation with the most powerful algorithmic-transformations and secondly to be easily independent from the architectural model. Some estimation results are compared with high-level synthesis of digital signal processing applications.

1 Introduction

Plutôt que de créer des systèmes de synthèse architecturale totalement automatiques, ne pouvant être utilisés que pour des modèles d'architectures spécifiques, il semble plus intéressant de fournir au concepteur un outil interactif qui le guide dans ses choix [1], et lui permette de laisser l'outil réaliser les optimisations dans la bonne direction. Plusieurs techniques comme les transformations structurelles [2] ou la sélection des composants [3], [4] sont potentiellement efficaces pour optimiser une architecture dédiée. Cependant l'espace de conception au niveaux algorithmique, fonctionnel et structurelle, est tellement vaste que l'outil et encore moins le concepteur peuvent explorer l'ensemble des possibilité existantes. Le pouvoir réel d'optimisation des transformations est donc sous-utilisé. C'est pourquoi, il semble utile d'effectuer avant synthèse une analyse de l'application à traiter, de manière à avoir une vue d'ensemble de ses caractéristiques et propriétés. Les informations recueillies par cette étude préliminaire doivent être suffisamment précises pour permettent de réduire l'espace de recherche et d'identifier les étapes critiques de l'algorithme de manière à mettre en lumière la nature des techniques à utiliser pour améliorer l'adéquation algorithme-architecture. Le module estimation, a été développé dans cette esprit, il s'intègre dans le cycle de développement de GAUT [5], sous la forme d'une étude probabiliste. Le module est à la fois un estimateur de ressource et un outil de guidage pour la synthèse d'architecture. Après avoir commenté les travaux effectués dans le domaine de l'estimation et de la caractérisation, nous développerons le principe de l'estimation probabiliste, en détaillant plus particulièrement le cas des opérateurs. Cette partie s'achèvera, par un exemple significatif de l'intérêt de la méthode. Enfin, nous terminerons par un exposé de résultats sur différents algorithmes (filtrage récursif, filtrage adaptatif, FFT). Des mesures effectuées à partir de l'estimation de complexité des exemples accompagneront ces définitions.

2 État de l'art

Les travaux de Jain et Sharma [6] traitent de l'estimation des ressources et performances en synthèse d'architecture. La méthode qu'ils proposent, estime le nombre de ressource minimum en évaluant dans chaque unité de temps, la limite inférieure du nombre de noeuds du graphe impliqués. Cependant, les relations de précédence entre opérations ne sont prises en compte que partiellement. De plus, la complexité de la méthode devient rapidement prohibitive : $O(N^2C)$ ou $O(NC^2)$ où N est le nombre de noeuds du graphe flot et C le nombre d'unité de temps imposé ou prévu pour l'exécution de l'algorithme. Enfin, l'évolution dans le temps des régions critiques n'est pas évoquée. Potkonjak et Rabaey ont également étudié la question. Ils présentent dans [7], un modèle, qui traite l'estimation des ressources et performances de manière duale,

elle conduit à une complexité en $Nlog(N)$, la méthode est plus rapide mais moins précise que la précédente, de plus elle n'aborde pas le problème de la répartition temporelle des ressources. Enfin, comme précédemment, les dépendances sont résolues par le biais des intervalles ASAP-ALAP, donc sous-estimées. Une nouvelle approche est proposée dans [8], les auteurs se placent dans une optique de guidage en cherchant à extraire les propriétés de l'algorithme avant synthèse. Ces derniers sont dans l'esprit de ce que nous décrivons par la suite, à la différence que nous traitons les propriétés de concurrence et de durée de vie des variables différemment en incluant l'aspect temporel et en prenant réellement en compte les relations de précédence, le problème de régularité est lui aussi traité dynamiquement et s'accompagne de métriques statistiques.

3 Estimation Probabiliste

3.1 Introduction

Le module estimation, inséré dans l'outil GAUT se situe après les deux premières étapes que sont : la compilation générant le graphe flot de données et le module sélection qui optimise le choix les opérateurs et associe par la même des temps de traversée aux opérations du graphe. Son rôle est d'effectuer une analyse des besoins de l'application à synthétiser suivant un modèle d'architecture décrit dans [5]. Celle-ci est réalisée pour chaque unité de temps située entre la date 0 et T correspondant à la contrainte de temps, l'unité de temps étant définie par le module sélection. Elle fournit à l'utilisateur une estimation du nombre d'opérateurs, du nombre de registres, du nombre de bus et des connexions entre opérateurs (y compris la mémoire). L'estimation tient compte compte des précédences par un calcul de probabilités conditionnelles, elle fournit également une étude de faisabilité de synthèse des boucles et des statistiques sur les liaisons entre opérateurs. Les calculs sont détaillés dans [9].

3.2 Estimation datée du nombre probable d'opérateurs

Le nombre probable d'opérations du type n ordonnancées, à l'instant t s'obtient de la façon suivante :

$$\tilde{N}_n(t) = \sum_{i=0}^{\tilde{N}_{max}} i.P_n(\tilde{N}_n = i, t) \qquad (1)$$

où $P_n(\tilde{N}_n = i, t)$ est la probabilité que i opérations du type n soient ordonnancées à l'instant t. La complexité de (1) la rend, en pratique, inexploitable. Par contre, une fois développée et simplifiée, elle se résume à l'expression suivante :

$$\tilde{N}_n(t) = \sum_{i \in Eop_n} P_{O_i}(t) \qquad (2)$$

où Eop_n est l'ensemble des opérations réalisant une opération du type n et $P_{O_i}(t)$ la probabilité que l'opération i soit ordonnancée à l'instant t. Finalement, le nombre probable d'opérateurs exécutant une opération du type n à l'instant t est obtenu en prenant en compte les délais de calcul ($\Delta_{op}(O_i)$: temps de traversée de l'opérateur associé à O_i :

$$N_n(t) = \sum_{i \in Eop_n} \sum_{k=t-\Delta_{op}(0_i)+1}^{t} P_{O_i}(k) \qquad (3)$$

Les probabilités associées peuvent être de type :

• **non conditionnées** : on considère dans ce cas la probabilité de présence d'une opération O_n comme indépendante des ses prédécesseurs. $P_{O_n}(t)$ ne dépend que des dates au plus tôt et au plus tard des opérations (voir figure 1).

$$P_{O_n}(t) = \frac{1}{dp(O_n)} = \frac{1}{alap_{O_n} - asap_{O_n} + 1} \qquad (4)$$

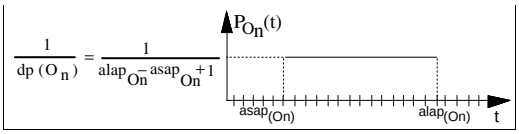

FIG. 1 — loi de probabilité uniforme

Il s'agit d'une densité utilisée en synthèse d'architectures pour mesurer la mobilité d'une opération et ainsi effectuer des choix d'ordonnancement [10] ou de transformations [7]. Cependant celle-ci est incomplète puisqu'elle exploite un parallélisme potentiel qui n'a pas lieu d'être. Dans leurs intervalles respectifs de mobilité maximale [asap ; alap] les prédécesseurs et successeurs sont considérés comme indépendants.

• **conditionnées** : la probabilité de présence de l'opération O_n dépend à présent de ses M prédécesseurs.

$P_{O_n}(t)$ est calculée en fonction de t, de sa densité de probabilité et de celle de chacun des N_{pred} prédécesseurs et des N_{suc} successeurs. Il n'est pas possible d'évaluer simplement $P_{O_n}(t+1)$ en fonction de $P_{O_n}(t)$. En effet, de manière symétrique, les probabilités conditionnées par les prédécesseurs peuvent être exprimées en fonction des dates passées alors que les probabilités conditionnées par les successeurs sont, elles, dépendantes des dates futures. Il y a blocage, le calcul récursif n'est pas permis, quelque soit de sens du parcours. Pour exprimer $P_{O_n}(t)$ en fonction de la probabilité de ses prédécesseurs et successeurs, il faut pouvoir disposer des valeurs $\{P_{O_n}(asap), ..., P_{O_n}(t-1), P_{O_n}(t+1), ..., P_{O_n}(alap)\}$ ce qui n'est envisageable que par approximations successives.

Le calcul exact, après convergence doit aboutir aux égalités suivantes :

$$P_{O_n}(t) = \prod_{i=1}^{N_{Pred}} \left(\sum_{t_1=asap(p_i)}^{t-\Delta(p_i)} P((p_i,t_1)|(O_n,t)).P_{O_n}(t) \right) \cdot \prod_{j=1}^{N_{Suc}} \left(\sum_{t_2=t+\Delta(O_n)}^{alap(s_j)} P((s_j,t_2)|(O_n,t)).P_{O_n}(t) \right)$$

où $P((A,t_1)|(B,t_2)).P_B(t_2)$ est la probabilité conditionnelle que A soit ordonnancée à t_1 sachant que B est ordonnancée à t_2. Ce calcul devant être effectué pour tous les ordonnancements possibles de l'ensemble des opérations du graphe, sa complexité interdit de l'utiliser si l'on souhaite conserver le caractère rapide de l'estimation.

Il est cependant possible d'effectuer ce calcul de manière beaucoup plus efficace, en acceptant une approximation. Le problème est alors vu de la manière suivante : nous considérons que la probabilité $P_{O_n}(t_k)$ est une simple proportion du nombre de possibilités d'ordonnancement offertes à O_n en $t = t_k$ par rapport à la totalité sur l'intervalle $[asap(O_n)\ ;\ alap(O_n)]$. Le calcul est donc effectué en associant une probabilité uniforme aux prédécesseurs et aux successeurs. Les probabilités exactes pourront être obtenues en répétant la méthode à partir de celles calculées à l'itération précédente. Pour des raisons de temps de calcul, que l'on souhaite court, nous nous arrêtons au premier ordre. Soit :

$$P_{O_n}(t) = \frac{1}{N_0} \quad . \quad \prod_{i=1}^{N_{Pred}} max[1; t - asap(p_i) - \Delta(p_i) + 1] \quad . \quad \prod_{j=1}^{N_{Suc}} max[1; asap(s_j) - t - \Delta(O_n) - 1]$$

avec :

$$No = \sum_{t=asap(O_n)}^{alap(O_n)} \prod_{i=1}^{N_{Pred}} max[1; t - alap(p_i) - \Delta(p_i) + 1] \quad . \quad \prod_{j=1}^{N_{Suc}} max[1; alap(s_j) - t - \Delta(O_n) - 1]$$

La complexité de calcul C se trouve extrêmement réduite, d'autant plus que No et P_{O_n} peuvent être calculées simultanément. Avec N le nombre d'opérations du graphe, la complexité est inférieure à :

$$O(N) \quad . \quad max_n\left\{[alap(n) - asap(n)]/\Delta_t\right\} \quad . \quad max_i\{N_{Pred}(i)\}.max_j\{N_{Suc}(j)\}$$

En conclusion, la méthode employée nous permet d'avoir une estimation à complexité arithmétique réduite. Elle offre également une exploration dans le temps du coût du matériel, conditionnée par les relations de précédence du graphe flot.

3.3 Estimation des bus, registres et interconnexions

Nous calculons en premier lieu, les probabilités de transferts pour chacune des variables du graphe flot. Celles-ci dépendent à la fois de l'ordonnancement de l'opérateur producteur et de l'opérateur consommateur. Nous délimitons ensuite les intervalles de temps pendant lesquels les variables occuperont un bus, registre, ou une connexion entre opérateurs. Le calcul des nombres probables s'effectue enfin, de la même manière que pour les opérateurs, par une somme des probabilités d'existence à l'instant t considéré. La complexité est similaire à la précédente, elle augmente dans le cas de la prise en compte du partage de données, où un bus (resp. un registre) est occupée par une variable destinée à plusieurs opérateurs.

Durant l'exploration du graphe, nous calculons les statistiques correspondant à la fréquence des liens entre les différents types d'opérateurs. Celles-ci sont calculées dans le but de fournir au concepteur une métrique lui permettant de juger de la régularité des liaisons et de décider par exemple de la fusion de deux opérateurs (classiquement un multiplieur-additionneur).

4 Résultats d'estimations

Différents algorithmes ont été testés pour différentes spécifications et contraintes temporelles *(Filtres de Voltera ; FFTs 1024 points ; IIR 7^{eme} ordre sous plusieurs spécifications modifiées par retiming [fig. 2] ; Filtre elliptique 5^{eme} ordre ; GAL 128 points - 30 cellules ; LMS 1024 points ; FIR 16 points ; LMS rapide [11])*. Nous avons calculé les moyennes des nombres probables d'opérateurs sur l'intervalle [1...T]. On observe que les moyennes obtenues sont identiques avec et sans condition sur les probabilités, ce qui est normal puisque l'aire (Nb probable x temps) est invariable, seule la forme de la courbe subit une déformation. Les résultats obtenus sont proches de ceux fournis par la synthèse, *le taux de corrélation étant de 0,95*. Ceci justifie l'utilisation, dans GAUT d'une simple moyenne pour initialiser la synthèse. Par contre, si l'on examine dans le détail les résultats, on note que la moyenne sous-estime la complexité dès lors que l'on teste un algorithme contenant des régions critiques associées à une contrainte de temps sévère. De telles observations ont été faites notamment avec le filtre Elliptic5, la FFT géométrique et un filtre IIR du 7^{eme} ordre. C'est la raison pour laquelle, l'analyse dynamique sur tout l'intervalle de temps est utile, les résultats associés à l'exemple suivant en sont l'illustration.

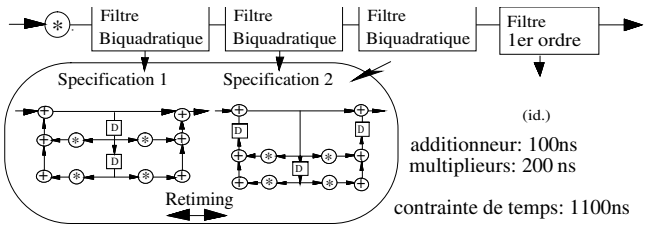

FIG. 2 — Spécifications du IIR7

Dans le tableau 1, nous présentons quelques résultats d'estimations concernant deux versions transformées d'un même algorithme. Cette comparaison entre différentes spécifications est actuellement la principale motivation de l'estimation de haut niveau.

La figure 3 donne les courbes d'estimation de complexité matérielle d'un filtre de Volterra du 2^{nd} ordre après différentes transformations successives de type *retiming*. Elle représente la distribution du coût probable dans les intervalles de temps [0 .. T]. Ces courbes peuvent être interprétées en utilisant les métriques de caractérisation présentées dans [12]. La figure 4 montre l'évolution du coût global du circuit après estimation (courbe en pointillés), et synthèse architecturale par l'outil GAUT (courbe en trait plein). Au fur et à mesure que les transformations sont appliquées, on note une certaine régularité dans la relation entre les estimations et les résultats de synthèse. Cette corrélation nous permet d'utiliser les estimateurs pour qualifier les transformations.

5 Conclusion

Nous avons décrit une nouvelle approche dans l'estimation de l'ensemble des ressources (unités fonctionnelles, registres,

Algo.	Volt.		FFT		IIR7		Ellip.	
	a	b	a	b	a	b	a	b
Esti.	3.14	2.91	23.6	21.2	17	13.5	8.1	7.2
Syn.	3.18	2.93	23	20.1	14.4	11.8	9.3	7.4
Algo.	GAL		LMS		Fir16		FE-LMS	
	a	b	a	b	a	b	a	b
Esti.	22	13.4	5.9	11.7	11.9	8.6	13.8	11.7
Syn.	19.3	13.3	4.1	12.1	10.9	7.6	13.6	11.2

TAB. 1 — Comparaison entre estimation probabiliste et synthèse architecurale par l'outil GAUT pour plusieurs algorithmes de TNS

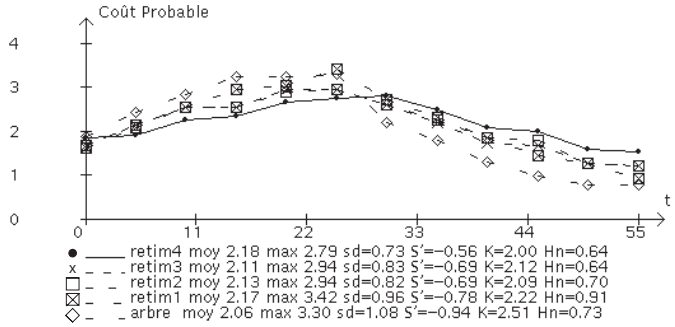

FIG. 3 — Coût probable de la spécification après tranformations du filtre de Volterra

bus et composants d'interconnexions) en synthèse architecturale d'unité de traitement. En utilisant les probabilités conditionnelles de placement de chaque noeud du graphe, la méthode permet une estimation rapide apportant une précision temporelle utile pour guider le concepteur dans ses choix algorithmiques ou ses options de transformations.

Ce module d'estimation de complexité est en cours d'intégration dans un environnement plus général donnant accès aux analyses prévisionnelles de la complexité des autres unités fonctionnelles (e.g. unité mémoire), et en utilisant d'autres contraintes ou critères (e.g. consommation) [12]. Il serait, par exemple, souhaitable de pouvoir interfacer de tels outils avec des langages de spécifications plus proches de ceux utilisés par les développeurs algorithmiciens (e.g Matlab, Ptolemy, etc.).

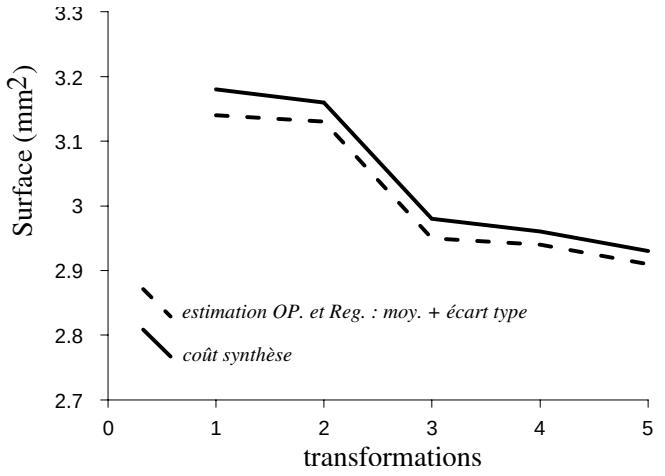

FIG. 4 — Comparaison entre synthèse et estimation de la complexité des filtres de Volterra

Ce *framework* doit pouvoir leur fournir rapidement des métriques afin d'estimer la complexité d'implantation matérielle de spécifications validées dans ces outils de simulation de traitement numérique du signal.

Références

[1] Working groups. Conclusions. In *IEEE Workshop on VLSI S.P.*, La Jolla, San Diego, October 1994.

[2] M. Potkonjak and J.M. Rabaey. Optimizing resource utilization using transformations. *IEEE trans. on C.A.D.*, 13(3), mar. 1994.

[3] S. Note, F. Catthor, G. Goossens, and H.J. De Man. Combined hardware selection and pipelining in high-performance data-path design. *IEEE Trans. on Computer-Aided Design*, 11(4) :413–423, April 1992.

[4] O. Sentieys, J.Ph. Diguet, J.L. Philippe, and E. Martin. Hardware module selection for real time pipeline architectures using probabilistic cost estimation. In *IEEE International ASIC Conference*, Rochester, NY, USA, September 1996.

[5] E. Martin, O. Sentieys, H. Dubois, and J.L. Philippe. Gaut, an architectural synthesis tool for dedicated signal processors. In *EURO-DAC*, pages 85–94, Hamburg, October 1993.

[6] A. Sharma and R. Jain. Estimating architectural resources and performance for high-level synthesis applications. *IEEE trans. on VLSI Systems*, 11, june 1993.

[7] J.M. Rabaey and M. Potkonjak. Estimating implementation bounds for real time DSP applications specific circuits. *IEEE Trans. on Computer-Aided Design*, 13(6), June 1994.

[8] L. Guerra, M. Potkonjak, and J. Rabaey. System-level design guidance using algorithm properties. In *IEEE Workshop on VLSI S.P.*, pages 73–82, San Diego, October 1994.

[9] J.Ph. Diguet, O. Sentieys, J.L. Philippe, and E. Martin. Probabilistic resource estimation for pipeline architecture. In *IEEE Workshop on VLSI S.P.*, pages 217–226, Sakai, Japan, October 1995.

[10] P.G. Paulin and J.P. Knight. Force-directed scheduling for the behavioral synthesis of asic's. *IEEE Trans. on Computer-Aided Design*, 8(6) :661–679, June 1989.

[11] J.Ph. Diguet, O. Sentieys, D. Chillet, and J.L. Philippe. Vlsi high level synthesis of fast exact least mean square algorithms. In *ICASSP'97*, 1997.

[12] J.Ph. Diguet, D. Chillet, J.G. Cousin, and O. Sentieys. A framework for cost-performance trade-offs in vlsi for signal processing. *Journal of VLSI Signal Processing Systems for signal, image and video technology, Special Issue on "Systematic Trade-off Analysis in Signal Processing Systems Design*, 1996. Kluwer Academic Publishers.