SympA'7
Paris, 24 - 27 Avril 2001
7ème Symposium sur les Architectures Nouvelles de Machines

Estimation de performances à un niveau comportemental pour l'implantation sur composants FPGA

S. Bilavarn, G. Gogniat, J. L. Philippe
LESTER - Centre de Recherche - BP 92116 - 56321 LORIENT Cedex
Tel : 02 97 87 45 60 - Fax : 02 97 87 45 00
bilavarn@iuplo.univ-ubs.fr

Mots Clés

Estimation architecturale, FPGA, vitesse, surface, consommation, exploration architecturale

Résumé

Cet article présente une méthode d'estimation des performances Temps-Surface-Consommation sur FPGA à partir d'un niveau comportemental. L'intérêt et l'originalité de la méthode est de permettre l'exploration rapide d'un espace de conception. En effet, le processus d'estimation est basé sur la détermination de courbes dynamiques et peu dépendant de la technologie grâce à la caractérisation du FPGA cible dans une librairie de composants. En plus des aspects temps et surface, l'estimation de la consommation est intégrée afin d'obtenir une caractérisation la plus complète possible de l'application. La méthode d'estimation décrite ici s'inscrit dans un flot de codesign qui a pour but la construction pragmatique d'une architecture hétérogène basée sur un processus itératif d'estimation du coût.

1. Introduction

Les progrès de la technologie et l'accroissement de la complexité des applications numériques conduisent à rechercher des architectures de systèmes mobiles optimisées, qui intègrent généralement des fonctions matérielles dédiées connectées à des processeurs programmables. Pour des raisons de coût, il n'est pas possible d'intégrer toutes les fonctions matérielles associées à tous les nouveaux services. Une approche consiste alors à utiliser des composants reconfigurables dynamiquement (type FPGA) connectés aux processeurs. Par conséquent, lors de la conception du système complet, il est nécessaire de prendre en compte cette possibilité dans le but d'obtenir des architectures optimisées en performances/surface de silicium/consommation d'énergie. L'évaluation des performances sur une technologie de ce type est un problème complexe du fait de l'augmentation des densités d'intégration et des récentes évolutions des architectures reconfigurables. Une estimation rapide du temps d'exécution, du taux d'occupation et de la consommation (performances Temps/Surface/Consommation) d'une application décrite au niveau comportemental est nécessaire pour permettre une exploration de l'espace de conception et choisir la meilleure solution d'implantation.

La méthode décrite dans cet article a pour but de guider le processus de conception d'architectures hétérogènes comme dans la conception conjointe logicielle / matérielle (co-design). En effet, une application complexe peut être vue comme un système composé de plusieurs fonctionalités que l'on cherche à implanter chacune dans un type de composant le mieux adapté à ses caractéristiques (ASIC, FPGA, DSP, processeurs RISC, CISC ...). Ainsi, la principale qualité des estimateurs est d'être rapide (aux dépens d'une précision absolue) de façon à explorer un vaste espace de recherche et de garantir le choix de la meilleure solution d'implantation lors de l'étape de partitionnement.

Beaucoup de techniques d'estimations sont décrites dans la littérature et s'appliquent à tous les niveaux de la conception, depuis la description comportementale (niveau système) [1][2] jusqu'au plan de masse (niveau layout) [3]. Les méthodes [4][5][6][7][8][9] permettent d'estimer le coût de l'unité de traitement (unités fonctionnelles, opérateurs arithmétiques et logiques, registres, multiplexeurs...). D'autres approches s'attachent à évaluer les caractéristiques de l'application sur le plan mémoire (taille, nombre de ports, complexité du générateur d'adresses, ...) [10] ou d'un point de vue de l'unité de contrôle (complexité de la machine d'état) [11]. Ces méthodes visent générale-

ment à caractériser l'application selon des critères de temps d'exécution et de surface occupée. [12][13] proposent des techniques d'estimation de la consommation pour FPGA. Dans [12], les différentes unités composant un FPGA (e.g. LUT, Registres, I/Os, arbres d'horloges) sont estimées séparément afin d'avoir un modèle précis des contributions respectives. A notre connaissance, il n'existe pas d'approche globale permettant d'estimer les trois facteurs TSC à partir d'une description comportementale. De plus, peu de travaux concernent les composants programmables. Dans [3], la méthode développée consiste à estimer les valeurs de temps et de surface à partir d'une netlist au niveau RTL résultant d'un outil de synthèse de haut niveau. Cette méthode est séduisante par la grande précision de ses estimations, mais possède deux inconvénients majeurs : le processus d'estimation est fortement dépendant de la technologie cible (i.e. XC4000) du fait de l'utilisation d'informations obtenues au niveau floorplan, et le temps d'estimation reste important si l'on souhaite explorer un vaste espace de conception, comme c'est le cas pour le co-design.

Notre approche permet à la fois d'être plus rapide (au détriment de la précision) et moins dépendant de la technologie grâce à la caractérisation du FPGA cible dans une librairie de composants. En plus des aspects temps et surface, l'estimation de la consommation est intégrée afin d'obtenir une caractérisation la plus complète possible de l'application. La méthode d'estimation décrite ici s'inscrit dans un flot de codesign qui a pour but la construction pragmatique d'une architecture hétérogène basée sur un processus itératif d'estimation du coût [1].

La suite de l'article se décompose de la façon suivante : le chapitre 2 présente le flot d'estimation en précisant l'intérêt des estimations dynamiques. Les chapitres 3, 4, 5 et 6 décrivent respectivement chacune des étapes du processus d'estimation. Dans cet article nous détaillons plus particulièrement l'étape d'estimation des CDFGs (chapitre 5). Le chapitre 7 présente les résultats d'estimations sur une application audio (filtre d'émission de la norme G722). Enfin, nous concluons et présentons les travaux futurs dans le chapitre 8.

2. Flot d'estimation

Le point d'entrée de l'estimateur est une description comportementale qui est saisie dans un langage de haut niveau (actuellement le langage C) puis traduite dans un modèle de spécification interne qui est un graphe flots de données et de contrôle hiérarchique (HCDFG) [14]. C'est sur ce graphe que le flot d'estimation est appliqué. Le modèle HCDFG correspond à un graphe G = {V, E} où chaque nœud $v_i \in V$ représente une partie de l'application. Un arc $e_{i,j}$ traduit la dépendance d'exécution entre les nœuds v_i et v_j. Il existe cinq types de nœuds différents (Figure 1).

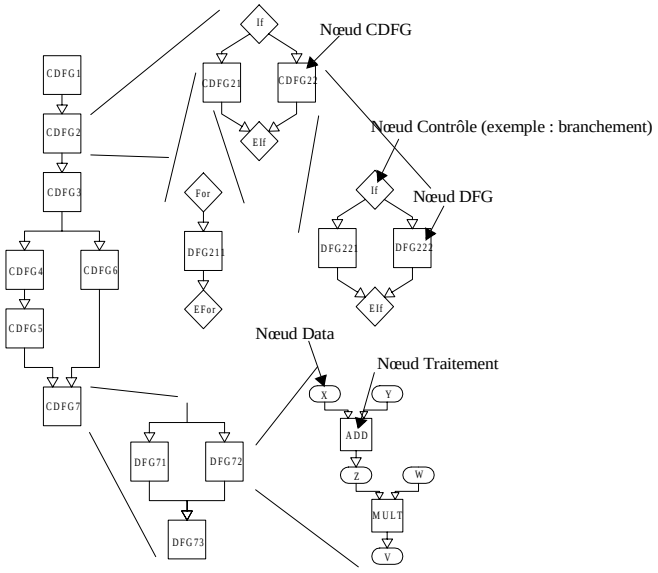
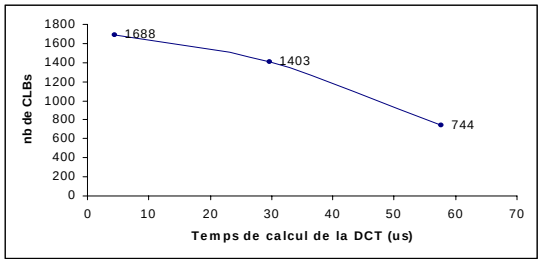


figure 1• Modèle de graphe HCDFG

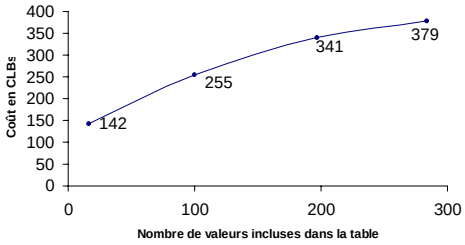
Les nœuds de type v_i^{CDFG} contiennent les structures hiérarchiques, c'est à dire qu'ils sont eux-mêmes composés de sous graphes. Ces nœuds permettent de gérer la compléxité des applications en évitant une mise à plat systématique des graphes, ce qui pourrait conduire à une explosion des algorithmes d'estimations. Les nœuds v_i^{CT} décrivent les structures de contrôle telles que les boucles et les branchements. Les nœuds v_i^{DFG} contiennent les séquences de codes linéaires de l'application (i.e. du type flot de données), donc sans aucune structure hiérarchique ou de contrôle. Ils constituent les blocs de bases (BB). Enfin, les nœuds v_i^{Data} modélisent les accès aux données et

les nœuds traitement v_i^{Trait} modélisent les opérations de calculs. La granularité des opérations peut être variable en fonction de la décomposition de l'application (e.g. addition, MAC, filtrage). Le choix du niveau de granularité est laissé au concepteur. Notons que les nœuds v_i^{Data} et v_i^{Trait} appartiennent aux nœuds v_i^{DFG} et que les nœuds v_i^{CDFG} encapsulent les nœuds v_i^{CT} et v_i^{DFG}.

Le modèle HCDFG présente l'intérêt d'exprimer explicitement les structures de contrôle ce qui le rend particulièrement adapté à une exploration architecturale. Un exemple classique concerne le déroulage partiel ou total des boucles afin d'estimer différents degrés de parallélisme de calcul. En effet, afin d'obtenir une estimation représentative des performances du système, il est nécessaire de considérer les différents degrés de parallélisme de l'application. Aussi, la méthode d'estimation que nous proposons estime les coûts TSC de manière dynamique sous la forme de courbes qui sont fonction d'une contrainte de temps variable. La contrainte de temps impose en effet d'exploiter plus ou moins le parallélisme de l'application. Pour illustrer le besoin qu'ont les concepteurs d'explorer plusieurs solutions architecturales, la Figure 2 présente les résultats issus d'une étude menée sur l'implantation partielle de MPEG4 sur un FPGA de chez Xilinx (XC4000) [15]. Lors de cette étude les concepteurs ont étudié manuellement plusieurs solutions afin d'évaluer différents compromis entre surface et vitesse. Les résultats de l'implantation d'une DCT-2D, en nombre de CLBs pour plusieurs contraintes de temps, et le coût d'implantation d'un décodeur VLC en fonction du nombre de mots d'une des tables de décodage sont présentés.



Surface de la DCT vs temps de calcul



Coût en CLB d'un décodeur VLC

figure 2• Exploration dynamique

L'étude de ces courbes a permis aux concepteurs de mieux évaluer plusieurs configurations et les temps d'exécutions correspondants. Dans notre exemple, les concepteurs pouvaient choisir d'implanter la DCT et le VLC pour un temps d'exécution de 4,4µs avec 16 valeurs incluses dans la table ce qui correspond à une occupation de 1688 + 142 = 1704 CLBs. Mais, ils pouvaient également retenir la configuration, t = 57,6µs, avec S = 744 + 379 = 1123, solution qui permet de minimiser la surface tout en augmentant le nombre de valeurs de la table. Cet exemple illustre la nécessité d'explorer plusieurs solutions avant de choisir l'implantation finale. Cependant, on perçoit bien ici la limite d'une approche manuelle qui impose d'effectuer une synthèse systématique de chacune des solutions. Il est donc essentiel d'apporter aux concepteurs des estimations de performances pour différentes configurations d'implantation. C'est clairement dans cette perspective que nous situons notre travail.

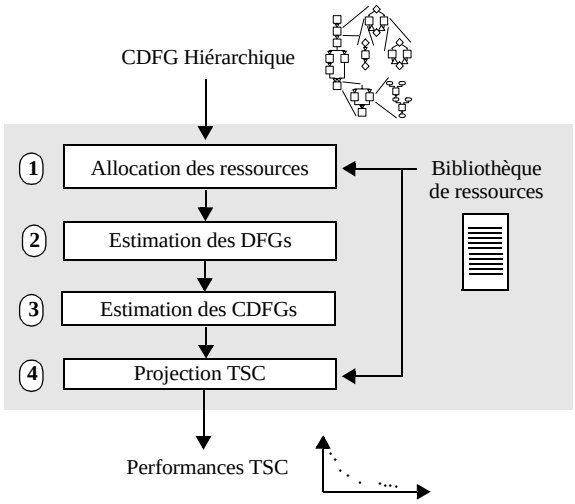


figure 3• Flot d'estimation

Afin d'estimer une application complète, le processus d'estimation (Figure 3) parcourt de façon incrémentale l'ensemble du graphe, selon une approche bottom-up, en partant des blocs de bases (les nœuds v_i^{DFG}) jusqu'aux nœuds v_i^{CDFG}. Cette approche permet de gérer la décomposition hiérarchique de l'application (ce point est plus particulièrement détaillé dans le chapitre 5).

De façon plus générale, le flot d'estimation est décomposé en quatre étapes : l'étape d'allocation (étape 1) associe à chaque opération du graphe une ressource (e.g. UAL, multiplieur, registre). Le nombre de ressources en fonction de plusieurs contraintes de temps (i.e. estimation dynamique) est ensuite estimé pour chaque bloc de base (i.e DFG) du graphe (étape 2). Dans l'étape 3, la hiérarchie du graphe est parcourue afin d'estimer l'application complète. Pour cela, les courbes dynamiques obtenues à l'étape 2 sont combinées en fonction du type de relations entre les différents des blocs de bases (exécution parallèle, séquentielle, hiérarchie, contrôle). A l'issue de cette étape, le nombre total de ressources nécessaires pour l'application est connu (actuellement, nous estimons uniquement les unités fonctionnelles). La dernière étape (étape 4) consiste à projeter ces résultats sur le composant FPGA cible (e.g. Virtex, Apex) afin d'obtenir des valeurs d'occupation et de consommation en fonction de plusieurs contraintes de temps.

3. Allocation des ressources

Afin d'effectuer l'estimation du nombre d'opérateurs alloués à l'application, il est tout d'abord nécessaire d'associer un opérateur à chaque opération du graphe. Pour cela, l'étape d'allocation se décompose en quatre parties :

- La première partie consiste à répertorier tous les types d'opérations du graphe (exploration de tous les nœuds v_i^{Trait}) et tous les formats des données (exploration de tous les nœuds v_i^{Data}).
- Ensuite, pour chaque opération répertoriée les opérateurs potentiels sont sélectionnés à partir d'une bibliothèque. Ce choix s'effectue en fonction du type d'opération et de la taille des données manipulées.
- Les opérateurs présentant les meilleurs compromis temps/surface/consommation sont alors sélectionnés. Au final, un seul opérateur est associé à chaque opération du graphe.
- Enfin, la période d'horloge est définie comme étant le temps de traversé de l'opérateur le plus rapide. On peut ainsi associer à chaque nœud v_i^{Trait} un nombre de cycles d'horloge à partir duquel les estimations sont effectuées.

Il est important de noter que chaque opérateur de la bibliothèque est caractérisé en temps/surface/consommation pour chacun des FPGA visés (e.g. Virtex, XC4000, Apex). Cette caractérisation doit être faite par le concepteur pour chaque cible qu'il veut évaluer et pour tous les opérateurs nécessaires. En général, les opérateurs sont transversaux à de nombreuses applications de TNS ce qui autorise la réutilisation de la bibliothèque pour différents designs.

Les caractéristiques de surface et de vitesse des éléments de la bibliothèque sont obtenues à partir des outils de synthèse (e.g. Foundation pour les FPGA de Xilinx). La consommation est directement mesurée sur le FPGA cible pour différents paramètres (e.g. format des données, fréquence d'horloge, type de données). Actuellement, la consommation d'un opérateur est caractérisée par sa consommation moyenne par accès. Pour cela, des vecteurs de test sont appliqués à l'opérateur et une consommation moyenne en est déduite. Nous sommes conscient des limites de cette approche, aussi des travaux sont en cours afin d'affiner le modèle utilisé. Le lecteur désirant en connaître davantage sur l'élaboration de la bibliothèque peut se référer à [16].

4. Estimation dynamique des DFGs

Chaque DFG du graphe est estimé selon une méthode classique d'estimation de DFG basée sur le calcul des dates au plus tôt (ASAP) et au plus tard (ALAP). Actuellement, nous utilisons la méthode présentée dans [4]. Cette méthode présente une précision suffisante pour une complexité réduite (il est important d'avoir à l'esprit que l'objectif de notre approche est d'effectuer une exploration de l'espace de conception, aussi il est nécessaire de considérer des méthodes d'estimation de faible complexité afin de réduire le temps de calcul de chaque solution). Toutefois, il est très facile d'adapter notre approche afin de considérer des algorithmes d'estimations plus précis [7], [17]. On peut imaginer de laisser le choix aux concepteurs du compromis précision/complexité en fonction de leurs applications et du nombre de solutions à explorer.

Les performances TSC sont caractérisées de manière dynamique, c'est à dire que chaque DFG est estimé pour un nombre de cycles variable. Ainsi, les résultats d'estimations (nombre de ressources vs nombre de cycles) permettent l'exploration de plusieurs solutions en prenant en compte les différents degrés de parallélisme des DFGs. Une contrainte de temps globale Ta_i est calculée pour chaque DFG_i du graphe et une zone d'exploration est définie autour de cette valeur afin de réduire le nombre de solutions explorées. En effet, les solutions estimées se situent

uniquement dans cette zone d'exploration. Le calcul de la zone d'exploration est défini de la façon suivante : Soit Δt_i l'intervalle d'exploration temporelle, et N le nombre de points d'exploration autour de Ta_i. Notons que le nombre de points d'exploration autour de Ta_i peut être défini manuellement où issu d'une analyse au niveau système [1]. Pour toutes les opérations des DFGs, on calcule le nombre d'unités fonctionnelles nécessaires et cela pour chaque contrainte de temps $Ta_i+k.\delta t_i$, avec $-N/2 \leq k \leq N/2$ et $\delta t_i=\Delta t_i/N$. La Figure 4 illustre la détermination de la zone d'exploration.

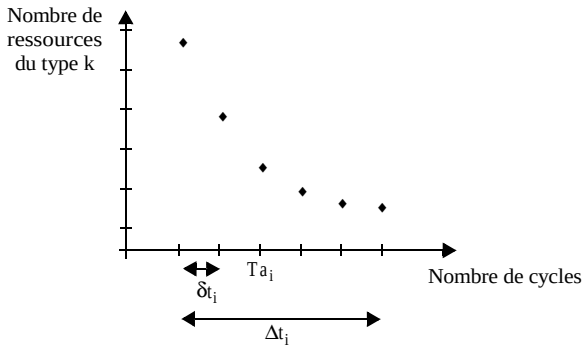


figure 4• Zone d'exploration

Cette étape d'estimation dynamique des DFGs est réalisée de façon automatique selon l'algorithme présentée ci-dessous. V est l'ensemble des nœuds du graphe, et R^{AB} est l'ensemble des unités fonctionnelles allouées aux opérations du graphe (étape 1).

```
DFGEstimation(V, R^AB, Ta, Δt, N)
E_P = {}
For all nodes v_i^DFG ∈ V^DFG
    E_P = {}
    E_P = DefineExplorationPoints(Ta_i, Δt_i, N)
    For all resources r^AB_i ∈ R^AB / r^AB_i is used in v_i^DFG
        For all timevalue ep_i ∈ E_P
        ComputeEstimationValue(ep_i, r^AB_i)
        End For
    End For
End For
```

A ce stade du flot, chaque DFG du graphe est estimé de manière dynamique (sous forme de courbes), c'est à dire pour un nombre de cycles variable.

5. Estimation dynamique des CDFGs

L'application est généralement modélisée de façon hiérarchique, aussi l'étape d'estimation des CDFGs vise à combiner les courbes dynamiques des différents DFGs du graphe afin de remonter progressivement au plus haut niveau de la description. Cette étape revient à prendre en compte le contrôle (i.e. boucles et branchements) et la hiérarchie (exécution parallèle/séquentielle des CDFGs). L'estimation dynamique des CDFGs est basée sur la construction et le regroupement des macroblocs (Figure 5). Au début du processus d'estimation des CDFGs, chaque DFG du graphe est labélisé comme étant un macrobloc. Les macroblocs sont ensuite regroupés s'ils sont caractérisés comme étant adjacents. Les macroblocs sont dits adjacents lorsqu'ils sont exclusivement parallèles, séquentiels ou lorsqu'ils sont inclus dans des structures conditionnelles ou des boucles. La Figure 5 présente les différents types de combinaisons. Par exemple, les DFG_{71} et DFG_{72} de la Figure 5.a peuvent être regroupés car ils sont caractérisés comme étant exclusivement parallèles. Ces deux macroblocs sont alors fusionnés en un nouveau macrobloc qui est caractérisé par la combinaison des courbes d'estimations de chacun des deux macroblocs précédents. Ce processus est itéré jusqu'à l'obtention d'un macrobloc final. La Figure 5.g représente le macrobloc final qui est donc une combinaison de tous les macroblocs du départ. Notons que la combinaison des macroblocs s'effectue deux par deux afin de limiter la complexité du processus de combinaison.

La combinaison des courbes d'estimations de chaque macrobloc dépend du type de dépendance qui leur est associée. Les rêgles de combinaison sont ainsi fonction du type d'interaction entre les macroblocs :

Cas 1• Deux macroblocs mb_1 et mb_2 s'exécutent en parallèle (Figure 5, cas 1). Dans ce cas, la contrainte

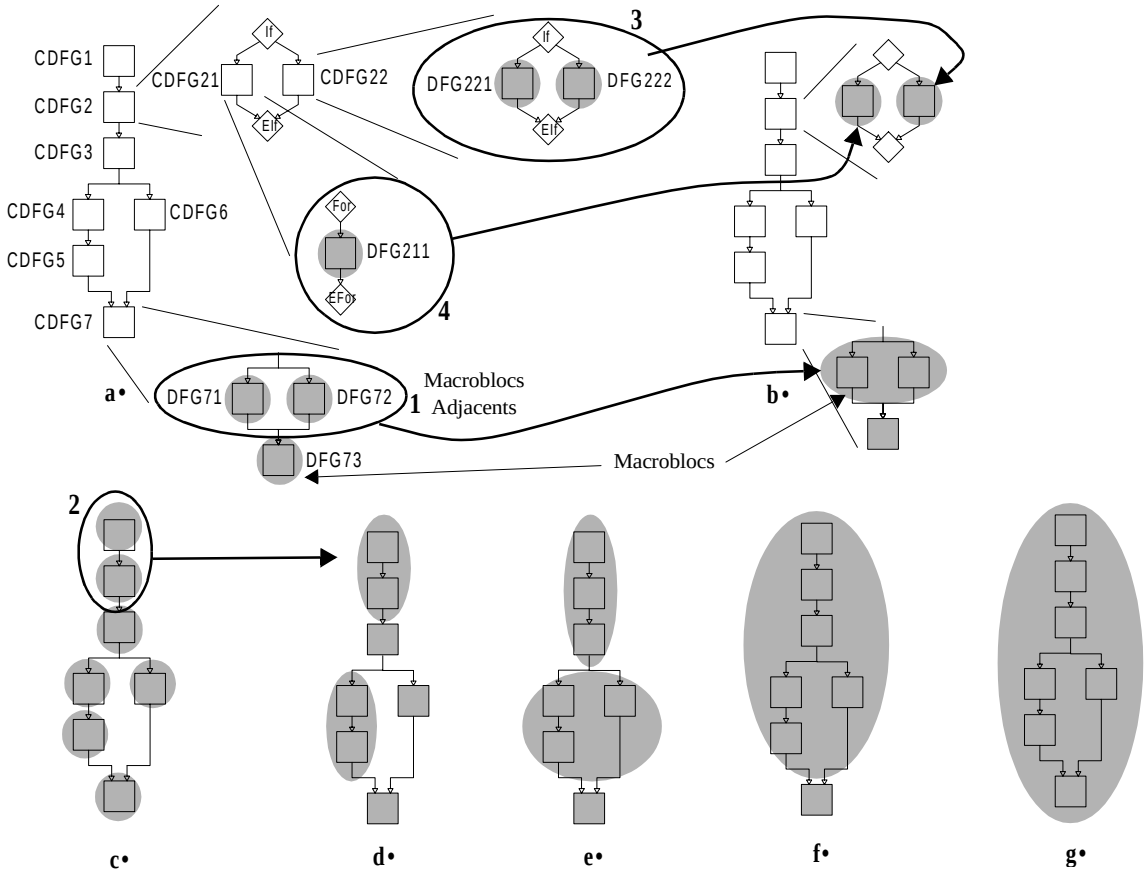


figure 5• Estimation hiérarchique du graphe

de temps T pour le macrobloc résultant est :

$$T = Max(T_{mb_1}, T_{mb_2})$$

où Tmb_1 et Tmb_2 sont les contraintes de temps respectives de chaque macrobloc. En ce qui concerne le calcul du nombre d'unités fonctionnelles, une distinction est faite selon le type d'opérateur :

• Analyse grossière (petits opérateurs) : le nombre d'unités fonctionnelles de type k $N_k(T)$ alloué pour un nombre de cycles T est :

$$N_k^{(mb_1, mb_2)}(T) = N_k^{mb_1}(T) + N_k^{mb_2}(T)$$

Cette méthode conduit à une surestimation mais possède l'avantage d'être très simple. C'est la raison pour laquelle, on la destine à l'estimation des petits opérateurs dont la taille et la consommation ont un faible impact sur la solution finale, afin de préserver le rapport précision / complexité de l'algorithme d'estimation.

• Analyse fine (opérateurs dont la surface et la consommation sont significatives). On étudie les profils (utilisation temporelle des opérateurs) pour optimiser le nombre de ressources (Figure 6).

$$N_k^{(mb_1, mb_2)}(T) = MAX_{t \in [0;T]}\left(\sum_{t=0}^{T} \left[N_k^{mb_1}(t) + N_k^{mb_2}(t) \right] \right)$$

Cas 2• Deux macroblocs mb_1 et mb_2 s'exécutent de façon séquentielle (Figure 5, cas 2). On favorise la réutilisation en prenant le maximum du nombre de ressources pour un opérateur de type k donné. Les temps d'exécution, eux, s'additionnent.

$$N_k^{(mb_1, mb_2)}(T) = MAX\left(N_k^{mb_1}\left(T_{mb_1}\right), N_k^{mb_2}\left(T_{mb_2}\right) \right)$$

$$T = T_{mb_1} + T_{mb_2}$$

Cas 3• Deux macroblocs mb_1 et mb_2 sont les branches d'une structure conditionnelle (Figure 5, cas 3). On

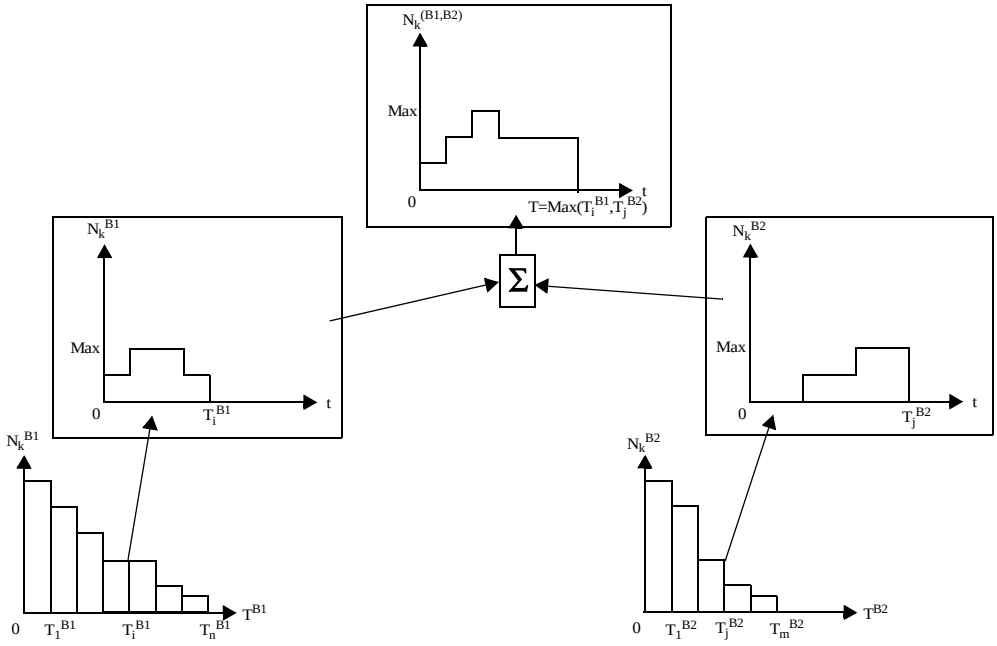


figure 6• Estimation fine des blocs parallèles

associe les probabilités de branchement $P(mb_1)$ et $P(mb_2)$ à chaque arc de transition [11]. Celles-ci peuvent être définies de trois manières : équiprobabilités (probabilité 1/2 de transition pour chaque branche), probabilités définies par l'utilisateur ou probabilités issues de la simulation (profiling).

Supposons que l'on ait deux branches mb_1 et mb_2 dont les nombres de ressources de type k sont connus pour chaque branche et valent $N_k^{mb1}(Tmb_1)$ et $N_k^{mb2}(Tmb_2)$, on calcule le temps d'exécution de la manière suivante :

$$T^{branch} = \left[P(mb_1) \cdot T_{mb_1}\right] + \left[P(mb_2) \cdot T_{mb_2}\right] + T_{ex}(CTL)$$

$$N_k(T^{branch}) = MAX\left(N_k^{mb_1}\left(T_{mb_1}\right), N_k^{mb_2}\left(T_{mb_2}\right)\right)$$

$T_{ex}(CTL)$ est le retard dû au test de branchement et peut être estimé comme n'importe quel autre DFG.

Cas 4• Un macrobloc constitue le corps d'une boucle (Figure 5, cas 4). On suppose connus les résultats d'estimation du corps de la boucle $N_k(T)$. On décompose les boucles en deux catégories selon que le nombre d'itérations dépend ou non des données.

- Boucles déterministes : le nombre d'itérations N est connu et fixe. On peut donc considérer plusieurs configurations possibles. Par exemple, soit une boucle FOR de 80 itérations, on peut choisir d'exécuter 80 fois une branche, 40 fois 2 branches en parallèle, 20 fois 4 branches, ...

Soit $N_k^{boucle}(T_{boucle})$ le nombre de ressources de type k nécessaires à l'exécution de la boucle, et N le nombre d'itérations. Pour une configuration donnée, i est le nombre d'itérations et b le nombre de branches ($i \bullet b = N$).

Le nombre de ressources est calculé de la manière suivante :

$$N_k^{boucle}(T_{boucle}) = N_k(T) \cdot b$$

$$T_{boucle} = T \cdot i$$

- Boucles non déterministes : on ne connait pas à priori N le nombre d'itérations. Dans ce cas, il ne peut y avoir qu'une seule configuration : une branche exécutée N fois. Pour obtenir des valeurs d'estimation, il faut connaître N. On le détermine à partir de vecteurs de test ou manuellement. On a :

$$N_k^{boucle}(T_{boucle}) = N_k(T)$$

$$T_{boucle} = T \cdot N$$

Le pseudo code de l'algorithme d'estimation des CDFGs est donné ci-dessous. Une première partie de l'algorithme analyse le graphe pour reconnaître les macroblocs adjacents afin de les fusionner. Ensuite, suivant le type de combinaison à mettre en œuvre (i.e. séquentielle, parallèle, branchement, boucle) la fonction correspondante est

```
HCDFGEstimation(V)
MN = {}
MN = CreateMacroNode(V^DFG)
While NumberOfMacroNode ≠ 1
    For all couple of MacroNode (mn_i, mn_j) ∈ MN
        if (mn_i, mn_j) are Adjacent MacroNode then
        ComputeMixedEstimation(mn_i, mn_j)
        CreateNewMacroNode(mn_ij)
        RemoveAdjacentMacroNode(mn_i, mn_j)
        If a hierarchical node v_i^CDFG ∈ V^CDFG is composed of only one MacroNode then
        ReplaceHNodewithMNode(v_i^CDFG, mn_ij)
    End For
End While
```

```
ComputeMixedEstimation(mn_i, mn_j)
Case AdjacentMacroNodeType(mn_i, mn_j)
    When Parallel ComputeParallelGraph(mn_i, mn_j)
    When Sequential ComputeSequentialGraph(mn_i, mn_j)
    When BranchComputeBranchGraph(mn_i, mn_j)
    When LoopComputeLoopGraph(mn_i, mn_j)
End Case
```

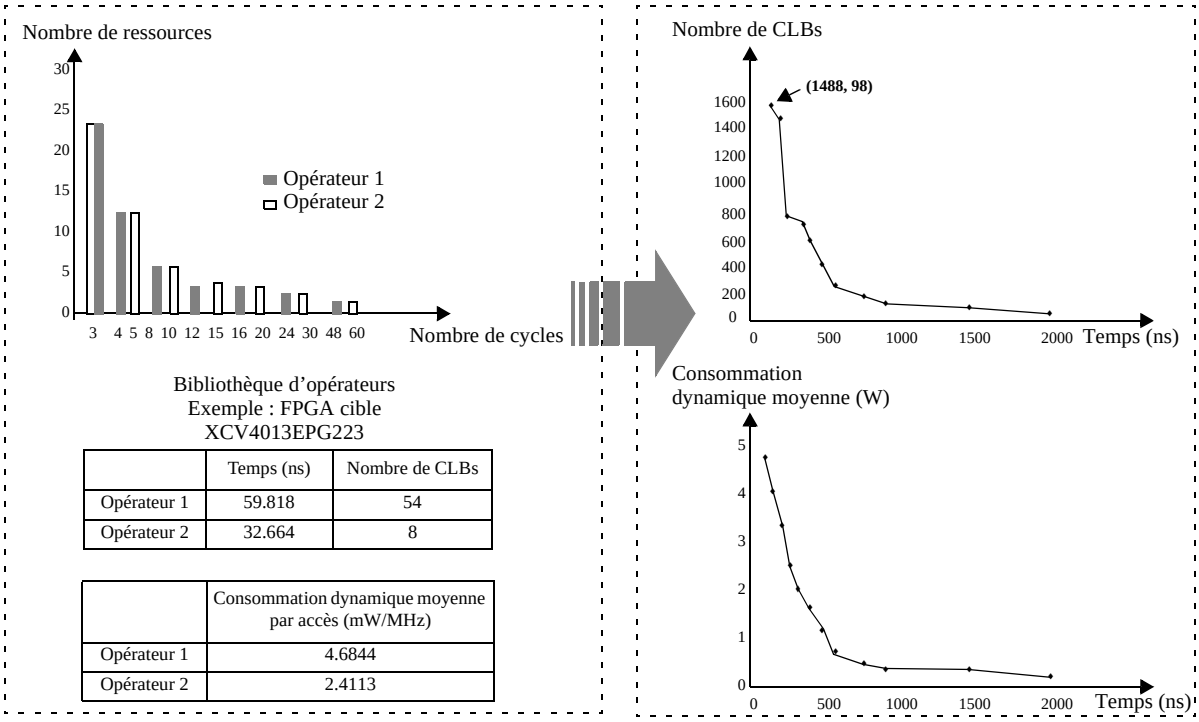
appelée.

Le processus se termine lorsqu'il n'y a plus qu'un seul macrobloc dans le graphe. On dispose alors des estimations en terme de nombre d'opérateurs vs nombre de cycles d'horloge. L'étape suivante a pour objectif d'associer des grandeurs physiques (e.g. nombre de CLB, durée) aux résultats de l'estimation.

6. Projection TSC

A ce stade du flot, le nombre d'unités fonctionnelles de chaque type pour un nombre de cycles variables est déterminé. De plus, la surface occupée (en nombre de CLBs), le temps de traversée et la consommation moyenne par accès de chaque opérateur sont disponibles dans la bibliothèque de description du composant cible.

Chaque temps d'exécution est calculé en multipliant la valeur de la période d'horloge par le nombre de cycles pour la configuration en cours. Le taux d'occupation du FPGA est obtenu en sommant la contribution de chaque unité fonctionnelle, connaissant la surface de chacune d'elle. Par exemple, dans la configuration 3 cycles de la Fi-



Bibliothèque d'opérateurs
Exemple : FPGA cible
XCV4013EPG223

	Temps (ns)	Nombre de CLBs
Opérateur 1	59.818	54
Opérateur 2	32.664	8

	Consommation dynamique moyenne par accès (mW/MHz)
Opérateur 1	4.6844
Opérateur 2	2.4113

figure 7• Projection TSC

gure 7, la solution requiert 24 opérateurs 1 et 24 opérateurs 2. Les opérateurs ayant une surface respective de 54 et de 8 CLBs, la surface finale est de 1488 CLBs. De plus, la période d'horloge étant fixée par l'opérateur le plus rapide (ici, l'opérateur 2), on obtient un temps d'exécution de 98 ns. Ce couple (1488, 98) se retrouve sur la courbe dynamique et correspond à une solution potentielle.

En ce qui concerne la consommation, le fichier de caractérisation du composant nous donne les valeurs de consommation dynamique moyenne par accès pour chaque opérateur en fonction de la fréquence d'horloge [16]. A partir de la consommation moyenne par accès pour l'opérateur de type k (PA_k) on calcule la consommation moyenne totale PA_k^{total} consommée par les $N_k^{operateurs}$ opérateurs de type k de la façon suivante :

$$PA_k^{total} = PA_k \times N_k^{operations}$$

La consommation totale est ensuite calculée en sommant toutes les contributions de tous les opérateurs du graphe. En considérant le même exemple que précédemment (configuration 3 cycles), on obtient une puissance moyenne de 5.2 W. En effet, 24 opérations sont exécutées sur l'opérateur 1 qui consomme 4.6844 mW par accès et par MHz, de même 24 opérations sont exécutées sur l'opérateur 2 qui consomme 2.4113 mW/MHz par accès. Ainsi pour les deux opérateurs, la consommation par MHz est de 0.17 W. Sachant que la fréquence de fonctionnement est de 30,61 MHz, on obtient donc une consommation moyenne totale de 5.2 W.

L'étape de projection TSC offre donc au concepteur un ensemble de solutions caractérisées à partir duquel il peut sélectionner une où plusieurs solutions à implanter. L'obtention des résultats d'estimation est un processus rapide ce qui permet au concepteur de n'effectuer la synthèse que pour les solutions qui lui semblent intéressantes.

7. Résultats

Afin de valider les estimateurs (précision vs complexité), nous avons choisi une application représentative du Traitement Numérique du Signal : le codage de la parole de la norme G.722. Le système de codage fait appel à la modulation par impulsions et codage différentiel adaptatif à sous-bandes (SB-MICDA). Nous présentons ici les résultats d'estimation du filtre d'émission en quadrature de phase. Cet exemple est intéressant dans la mesure où il est constitué d'une boucle déterministe, donc d'un parallélisme variable. Nous nous intéressons ici à la boucle FOR de 12 itérations. Le corps de la boucle est constitué de deux opérations de multiplication (opérandes 8 bits) et également de deux opérations d'addition (opérandes 14 bits). Les caractéristiques de ces deux opérateurs conduisent à une période d'horloge de 37,401 ns (25 Mhz). On attribue un temps de traversée de 1 cycle aux opérations d'addition et de 2 cycles à celles de multiplication. Les résultats du calcul du corps de la boucle sont combinés afin d'aboutir à l'estimation dynamique de la boucle de 12 itérations. Connaissant les courbes dynamiques (nombre de ressources vs nombre de cycles) des opérateurs, on en déduit la surface occupée en nombre de CLBs en fonction du temps exprimé en ns (Figure 8). Si on compare ces valeurs d'estimation avec les résultats de la synthèse de la boucle dans ses différents degrés de parallélisme, on constate que l'estimation du nombre de CLBs est très précise (1.75 % max). Pour les contraintes temporelles, l'écart est plus important et peut atteindre 39.9 % surtout pour les faibles valeurs de contraintes de temps où la surface est plus importante, le surcoût dû à la matrice d'interconnexions n'est alors plus négligeable.

Les résultats d'estimation de la consommation dynamique pour une fréquence d'horloge de 25 MHz sont aussi présentés Figure 8. Ces résultats peuvent être comparés à la méthode d'estimation fournie par Xilinx [16]. Les résultats montrent un écart entre les estimations et les valeurs issues du modèle numérique de Xilinx. Cet écart résulte de deux modèles simplifiés qui se combinent. En effet, la méthode d'estimation actuellement n'intègre qu'un modèle simple de la consommation dans un FPGA, et le modèle numérique de Xilinx ne prend en compte qu'un nom-

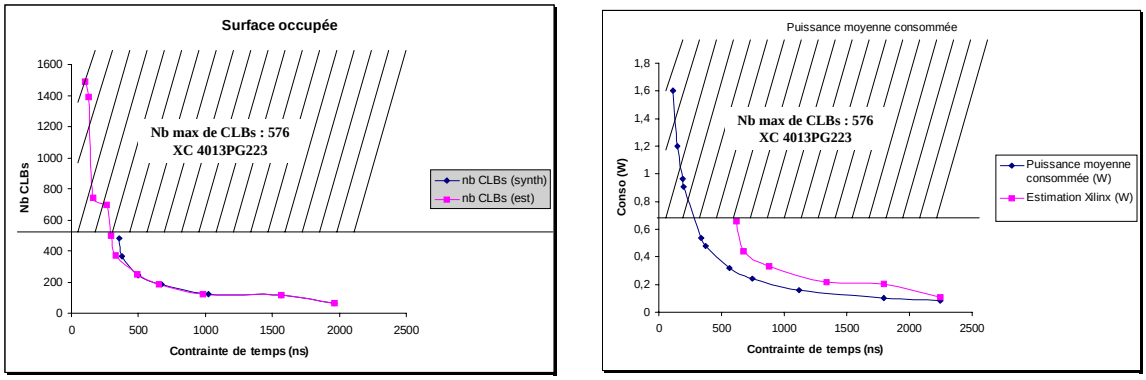


figure 8• Estimation du filtre d'emission en quadrature de phase

bre limité de facteurs influant sur la consommation. Aussi, avec le modèle d'estimation actuel, il faut considérer l'estimation de la consommation comme un indicateur plutôt que comme une valeur exacte. Cet indicateur permet de comparer deux solutions différentes afin de retenir, par exemple, celle présentant le meilleur compromis vitesse/ surface/consommation.

8. Conclusion et perspectives

Cet article présente une méthode d'estimation globale (temps, surface, consommation) sur FPGA permettant une exploration de l'espace de conception. La méthode est décomposée en 4 étapes successives : allocation des ressources, estimation des DFGs, estimation des CDFGs et projection TSC. Seule la dernière étape est fortement liée à une cible particulière ce qui rend la méthode générique et autorise de viser plusieurs cibles. Par ailleurs, les algorithmes d'estimation utilisés sont volontairement d'une complexité réduite (aux dépens de la précision) afin de rendre l'exploration d'un espace de conception efficace. En effet, les systèmes visés sont du type systèmes sur silicium pour lesquels il est essentiel de pouvoir évaluer rapidement plusieurs solutions. Actuellement, la méthode conduit à l'estimation des opérateurs de l'application, il est donc nécessaire d'étendre les travaux afin d'estimer également les structures de contrôle, les interconnexions et les unités de mémorisation. Comme le montre les résultats obtenus, il est indispensable de considérer l'impact des interconnexions dès que le taux de remplissage du FPGA devient important. En effet, le surcoût en vitesse et en consommation des interconnexions n'est alors plus négligeable. Enfin, dans le cas des systèmes sur silicium, des unités matérielles peuvent échanger des informations avec des processeurs, aussi il est important d'estimer l'impact des communications sur la solution finale.

9. Références

1. H. Thomas, J.P. Diguet and J.L. Philippe, *A methodology for an Application Profiling at a System Level*, SiPS, Taïwan, October 1999.
2. F. Vahid and D.D. Gajski, *Closeness metrics for system-level functional partitioning*, EDAC 95.
3. M. Xu and F.J. Kurdahi, *Accurate Prediction of Quality Metrics for Logic Level Design Targeted Toward Lookup-Table-Based FPGA's*, IEEE Transactions on Very Large Scale Integration Systems, vol. 7, No. 4, December 1999.
4. A. Sharma and R. Jain, *Estimating Architectural Resources and Performance for High-Level Synthesis Applications*, IEEE Transaction on Very Large Scale Integration Systems, Vol 1, No 2, June 1993.
5. M. Rabaey and M. Potkonjak, *Estimating Implementation Bounds for Real Time DSP Application Specific Circuits*, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, Vol 13, No 6, June 1994.
6. M. Rim and R. Jain, *Lower-Bound Performance Estimation for the High-Level Synthesis Scheduling Problem*, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, Vol 13, No 4, April 1994.
7. J.P. Diguet, *Estimation de Complexité et Transformations d'Algorithmes de Traitement du Signal pour la Conception de Circuits VLSI*, Thèse, Université de Rennes I, 1996.
8. J. Henkel, R. Ernst, *High-Level Estimation Techniques for Usage in Hardware/Software Co-Design*, IEEE/ACM Proc. Asia and South Pacific Design Automation Conference, Yokohama, Japan, February 1998.
9. J. Henkel, R. Ernst, *The Interplay of Run-Time Estimation and Granularity in HW/SW Partitioning*, Proc. Codes/ Cashe, Pittsburg, USA, 1996.
10. F. Balasa, F. Catthoor and H. De Man, *Background Memory Area Estimation for Multi-Dimensional Signal Processing Systems*, IMEC, Kapeldreef 75, B-3001 Leuven, Belgium.
11. S. Narayan and D.D. Gajski, *Area and Performance Estimation from System-Level Specifications*, Technical Report, University of California, 1992.
12. A. Garcia, W. Burleson, J.L. Danger, *Power Modelling in Field Programmable Gate Arrays*, 9th International Workshop on Field Programmable Logic and Applications, Glasgow, Scotland, August 1999.
13. K. Weib, C. Oetker, I. Katchan, T. Steckstor, W. Rosenstiel, *Power Estimation Approach for SRAM-based FPGAs*, 8th ACM International Symposium on Field Programmable Gate Arrays, Monteray, CA, USA, February 2000.
14. J.P. Diguet, G. Gogniat, P. Danielo, M. Auguin and J.L. Philippe, *The SPF Model*, FDL'2000, Tübingen, Germany, September 2000.
15. P. Chedmail, *Etude sur FPGA et DSP de l'implantation de fonctions d'un décodeur MPEG-4*, Master thesis, Université de Rennes 1, Juin 1999
16. G. Lenet, *Définition d'une bibliothèque d'opérateurs caractérisés en temps surface consommation sur FPGA*, Technical Report, LESTER,may 2000.
17. D.D. Gajski, N. Dutt, A. Wu and S. Lin, *High-Level Synthesis : introduction to Chip and System Design*, Kluwer Academic Publishers, 1992.