Vers la définition de composants virtuels au niveau algorithmique

Sébastien PILLEMENT, Olivier SENTIEYS, Daniel CHILLET

LASTI-ENSSAT, Université Rennes I
6, rue de kérampont, 22300 Lannion, France
http://archi.enssat.fr
prenom.nom@enssat.fr

Résumé –
Les méthodologies de conception à base d'Intellectual Property (Ip), sont des techniques prometteuses permettant d'envisager la conception de systèmes complet sur une puce (SoC). De plus, l'optimisation de ces SoC passe par une description de haut niveau de l'application complète.

L'originalité de l'approche consiste à définir des blocs réutilisables au niveau comportemental (i.e. algorithmique), ainsi qu'une méthodologie de conception indépendante de l'application considérée. Nous présentons dans ce papier un flot de conception permettant de spécifier et d'utiliser des Ips comportementaux, ces blocs sont alors spécialisés par l'utilisateur en fonction de ses besoins. Cette méthodologie est basée sur les concepts de générateur d'Ip et d'outil de synthèse comportementale universel (U-HLS)

Abstract –
It is well known now that Intellectual Property (Ip) design reuse is one of the most promising technique that solves the productivity gap problem. It is now accepted that a SoC will be synthesized from a high-level functional description using a common high-level language. Specifying the Ip at the behavioral level appears as the most promising solution to achieve a real efficiency of design reuse.

In this paper we propose a methodology to specify and use Behavioral Level Ip. Thus, Ip designer tasks are easier due to the unified representation offered by this level of abstraction. The genericity of a behavioral Ip permits efficient optimizations and makes application context adaptations a reality. We propose a unified framework to define an Ip at the behavioral level and to tune a particular block according to designer needs. Therefore, we define the Ip generator tool and the Universal High Level Synthesis concepts.

1 Introduction

Les dernières avancées technologiques au niveau des densités d'intégration sur silicium sont en train de bouleverser la conception des systèmes. Les progrès réalisés permettent l'intégration de systèmes complets sur une puce de silicium (SOC), à partir de macro-blocs réutilisables ("IP" pour Intellectual Property), tels que des cœurs de processeurs, des unités arithmétiques sophistiquées, des blocs de logique spécifique permettant un traitement adapté de l'information ou des blocs de logique reconfigurable permettant par exemple d'intégrer différents protocoles de communications [1].

Ce concept de réutilisation de parties matérielles a tendance à se généraliser aujourd'hui [2, 3]. La recherche de performances et la complexité de conception d'un SOC conduisent à la réutilisation de composants *virtuels*. Malheureusement, à l'heure actuelle, les concepteurs manquent d'outils et de méthodologies pour assembler ces blocs, ce qui constitue un frein à l'intégration d'applications complexes [4, 5].

Dans le contexte actuel, la mise en place de la notion d'Ip comportemental permet la normalisation et la simplification du processus de choix et de validation du bloc réutilisable [6]. L'objectif est d'offrir aux concepteurs un outil permettant de définir les caractéristiques générales d'une fonction souhaitée, le choix de l'implantation et des paramètres non définis se fait alors automatiquement pour un flot de conception particulier et ceci parmi un ensemble de solutions optimisées. De plus, l'augmentation du niveau d'abstraction, permet d'avoir un spectre plus large de solutions, et donc de satisfaire un plus grand nombre de clients.

Cette étude se situe dans le cadre du projet RNRT MIPAT [7]. Elle a pour objectifs la définition du concept d'Ip comportemental et des paramètres nécessaires à sa modélisation, mais aussi le développement d'une méthodologie de conception associée. Après avoir fixé les caractéristiques d'un composant virtuel de niveau algorithmique, nous présentons dans la troisième section le générateur d'Ip. La description d'une architecture logicielle pour ce générateur nous amènera à l'identification de paramètres nécessaires aux Ips comportementaux. Avant de conclure nous présenterons la spécification comportementale d'un Ip de filtrage numérique.

2 Caractéristiques générales d'un Ip comportemental

L'intérêt de la synthèse d'architecture est d'automatiser la tâche permettant la définition structurelle d'un circuit à partir de sa spécification algorithmique, appelée par la suite niveau comportemental. Ces outils permettent de réduire les temps de conception et les risques d'erreurs humaines. L'automatisation du processus de synthèse permet également d'ouvrir le domaine à des personnes non-spécialistes.

À partir d'une même description comportementale et d'un ensemble de contraintes, il est possible de générer un grand nombre de solutions architecturales permettant ainsi d'explorer l'espace de conception. Un Ip spécifié à ce niveau amènera donc une grande généricité permettant une réutilisation maximale de la fonctionnalité.

L'instance d'un Ip est le module caractérisé (en général pour une technologie particulière) que le *designer* va intégrer à son système. Nous pouvons définir un ensemble de critères qui permettent de cerner les objectifs que doit atteindre un Ip comportemental performant.

- La méthodologie de création de l'Ip doit admettre comme entrée une description de niveau comportemental.
- Le modèle doit adopter une structure uniforme quel que soit l'Ip. Le but est ici d'uniformiser les paramètres, et de créer des règles de choix entre les différentes implémentations disponibles.
- La méthode doit être transférable à la conception de n'importe quel module.
- L'utilisation d'un design réutilisable doit se faire avec un coût minimum.
- Le surcoût engendré par la création d'un Ip comportemental doit être amorti après un nombre de réutilisations raisonnable.
- L'utilisation d'un Ip ne doit pas entraîner des baisses de performances.
- Les caractéristiques de l'Ip doivent être les mêmes quel que soit le flot de conception utilisé par le concepteur. Ceci amène à la définition du concept U-HLS (pour Universal High-Level-Synthesis), c'est à dire d'outil de synthèse d'architecture universel.

Notre approche est indépendante du contexte et de l'application en cours de développement. L'objectif est de réduire le temps de conception et de s'appuyer sur l'expérience des concepteurs d'Ip. Une fonctionnalité est alors vue comme un générateur d'Ip. Les contraintes et/ou la fonctionnalité peuvent amener à la définition de plusieurs descriptions comportementales de l'application.

L'élaboration d'un Ip générique commence par l'étude des différents paramètres qui caractérisent la fonction. Comme nous allons le voir sur un Ip de filtrage numérique, cette étape détermine la généricité de la fonction.

On peut classer les paramètres en trois catégories :

- **les paramètres utilisateurs** caractérisent les besoins du concepteur en terme de performance ;
- **les paramètres algorithmiques** influent sur la sélection d'un algorithme particulier ;
- **les paramètres outils** sont spécifiques à chaque outil de synthèse.

3 Générateur d'Ip

La notion de générateur est spécifique à chaque Ip comportemental. Ce générateur peut se décomposer en plusieurs modules (Figure 1) :

- Une interface, elle permet au concepteur de spécifier les paramètres et les contraintes qui caractériseront l'instance d'un Ip.
- Un estimateur, il permet de guider l'utilisateur vers une bonne solution au regard des performances requises.
- Un générateur U-HLS, il accepte en entrée les paramètres fournis par le module d'interface et fournit un Ip U-HLS.
- Des dérivateurs, ils partent d'un Ip U-HLS et les dérivent pour obtenir des descriptions d'Ips spécifiques aux outils de synthèse.

Pour des raisons de diffusion et de simplicité d'accès, nous avons choisi une architecture logicielle basée sur des applets Java et des pages html. La configuration du générateur pour un Ip spécifique s'effectue alors par un fichier de paramètres. Ce fichier représente le degré de généricité de l'Ip.

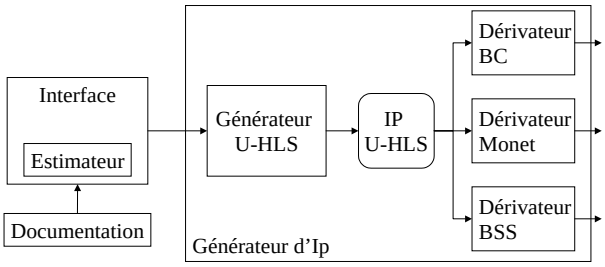

FIG. 1 – Architecture d'un générateur d'Ip.

4 Ip de filtrage numérique

4.1 Notions de base sur le filtrage numérique

L'exemple que nous présentons est un Ip de filtrage numérique. Deux types de filtres sont envisageables les filtres RII – ou récursifs – (équation 1) et les filtres RIF – ou non récursifs – (équation 2). Les équations 1 et 2 sont utilisées pour écrire le code comportemental de la fonction filtrage (Figure 2).

$$y(n) = \sum_{i=0}^{N} b_i.x(n-i) - \sum_{i=0}^{N} a_i.y(n-i) \qquad (1)$$

$$y(n) = \sum_{i=0}^{N-1} b_i.x(n-i) = \sum_{i=0}^{N-1} h(i).x(n-i) \qquad (2)$$

Le principal paramètre de généricité d'un tel filtre est le nombre N ainsi que la valeur des coefficients (a_i et b_i) qui composent la fonction de transfert $H(z)$. Ces coefficients peuvent êtres déterminés à partir d'une spécification réalisée à partir d'un gabarit en fréquence. Cependant, certains

filtres issus des normes peuvent être définis directement à partir de leurs coefficients.

4.2 Spécifications algorithmiques des filtres numériques

A chaque structure de filtre va correspondre une spécification en langage de haut niveau. Nous présentons dans la figure 2 la spécification VHDL générique d'un filtre RIF à structure directe pour l'outil BC [8] de Synopsys. Selon les paramètres retenus, ou le choix de l'utilisateur, une structure sera retenue comme modèle de spécification en vue de la synthèse.

```
entity  direct_fir is
  port(xn : in Integer range -2**($Bin-1) to 2**($Bin-1);
       yn : out Integer range -2**($Bout-1) to 2**($Bout-1);
       clk, rst : in Std_Logic );
end direct_fir;
architecture behavioral of direct_fir is

TYPE vectorH IS ARRAY (0 TO $N-1) OF Integer range ...
    ...
    main : Process
      Attribute dont_unroll of fir_loop : label is $LoopDUnroll;
      Variable X : vectorH;
      Variable acc : Integer range ...
      Variable mult : Integer range -2**($Bin+$Bcoeff-1)
      ...                          to 2**($Bin+$Bcoeff-1);
    main : loop

      tmp:=0;
      sig_evol : for i in 0 to x'length-2 loop
         x(i):=x(i+1);
      end loop sig_evol ;
      x(x'length-1):=xn;
      fir_loop : for i in 0 to h'length-1 loop
          mult:=x(i)*h(i);
          acc:=acc+mult;
      end loop fir_loop;
      yn<=acc;
      ...
    ...
```

FIG. 2 – Spécification VHDL générique pour un filtre FIR

Suivant la spécification définie au paragraphe 4.1, la taille du filtre est définie par le paramètre N et les tailles des données et des coefficients sont définies par les paramètres $Bin, Bout, BCoeff$.

4.3 Paramètres des Ips de filtrage numérique

Le niveau d'abstraction comportemental utilisé comme spécification des Ips étudiées dans ce projet permet d'obtenir un degré de généricité très élevé.

Les paramètres algorithmiques ont une influence sur le code de description comportementale, pour l'Ip de filtrage on trouve dans cette catégorie le choix du type de filtre (RII ou RIF), la forme (directe, cascade, parallèle) et la structure (directe, transposée, canonique) du filtre. La méthode de spécification du filtre (gabarit fréquentiel ou liste de coefficients) est aussi un paramètre de cette classe.

Les paramètres utilisateurs sont utilisés pour rendre synthétisable un code générique. Il représente les besoins et les contraintes d'un utilisateur. Ainsi on retrouve dans cette catégorie : N l'ordre du filtre (RII) ou la longueur de la réponse impulsionnelle (RIF), la valeur des coefficients (si le filtre n'est pas spécifié par un gabarit), la taille des données d'entrées/sorties ($Bin, Bout$), la taille de l'accumulateur ($Bacc$) ou le rapport signal à bruit.

Enfin les paramètres outils utilisent les stratégies des outils de synthèse comportementaux, par exemple la possibilité de dérouler les boucles.

Cependant l'ensemble des paramètres et des contraintes ont des répercussions les uns sur les autres. Il est donc nécessaires de spécifier les interactions entre les paramètres afin d'obtenir une solution réaliste.

4.4 Interaction entre paramètres

Cette partie de la spécification d'un Ip comportemental est très importante car c'est elle qui détermine les performances possibles pour une fonction particulière. Nous allons détailler dans ce paragraphe les dépendances de certain paramètres sur la cadence et le rapport signal à bruit de quantification ($SQNR$), pour un filtre RIF sous la forme directe. Les données d'entrées sont comprises dans l'intervalle [-1...1] dans un codage en virgule fixe. La généralisation pour toutes les structures des filtres a été réalisé dans notre générateur.

• Le rapport signal à bruit de quantification ($SQNR$) dépend de la valeur des coefficients et des paramètres N, Bin, $Bout$, $Bacc$, mais aussi d'autres paramètres qui doivent être définis, ce sont Qin, $Qout$, $Qcoeff$ représentant respectivement la quantification des entrées/sorties et des coefficients. Des variables internes (invisibles pour l'utilisateur final) sont alors évaluées, il s'agit de la dynamique des coefficients, et du nombre de bits de garde BG, qui garantissent qu'il n'y aura pas de dépassement lors du filtrage [9]. La taille de l'accumulateur $Bacc$ doit alors suivre la règle suivante :

$$\begin{aligned} Dynamique &= \sum_{i=0}^{N-1} |b_i| \\ BG &= log_2\lceil Dynamique \rceil - 1 \\ Bacc &\leq BG + Bcoeff + Bin \end{aligned}$$

Le rapport signal à bruit de quantification est alors donné par :

$$SQNR = \frac{\sigma_x^2 \sum_{i=0}^{N-1} b_i^2}{\sigma_e^2 \sum_{i=0}^{N-1} b_i^2 + \frac{q_{out}^2}{12}(1 - 2^{-2(Bacc-Bout)})}$$

A ce point l'utilisateur peut alors définir une contrainte sur le $SQNR$ et l'outil dérivera alors la taille de l'accumulateur nécessaire pour satisfaire cette contrainte.

• La cadence du filtre, dans ce cas la période d'échantillonage, a une influence sur les contraintes de temps données aux outils de synthèse comportementale. Mais elle impose surtout une contrainte sur le facteur de *déroulage (UF)* de la boucle de filtrage. Dans le cas du filtre RIF avec un temps de multiplication T_{mult}, ce facteur doit répondre à la règle :

$$UF \geq \lceil \frac{N.T_{mult}}{Throughput} \rceil$$

Ce déroulage de boucle correspond à la multiplication des ressources de calcul, permettant de faire du calcul parallèle. Ce paramètre découle donc directement de la contrainte de temps imposée par l'utilisateur mais influence aussi la surface finale de l'Ip.

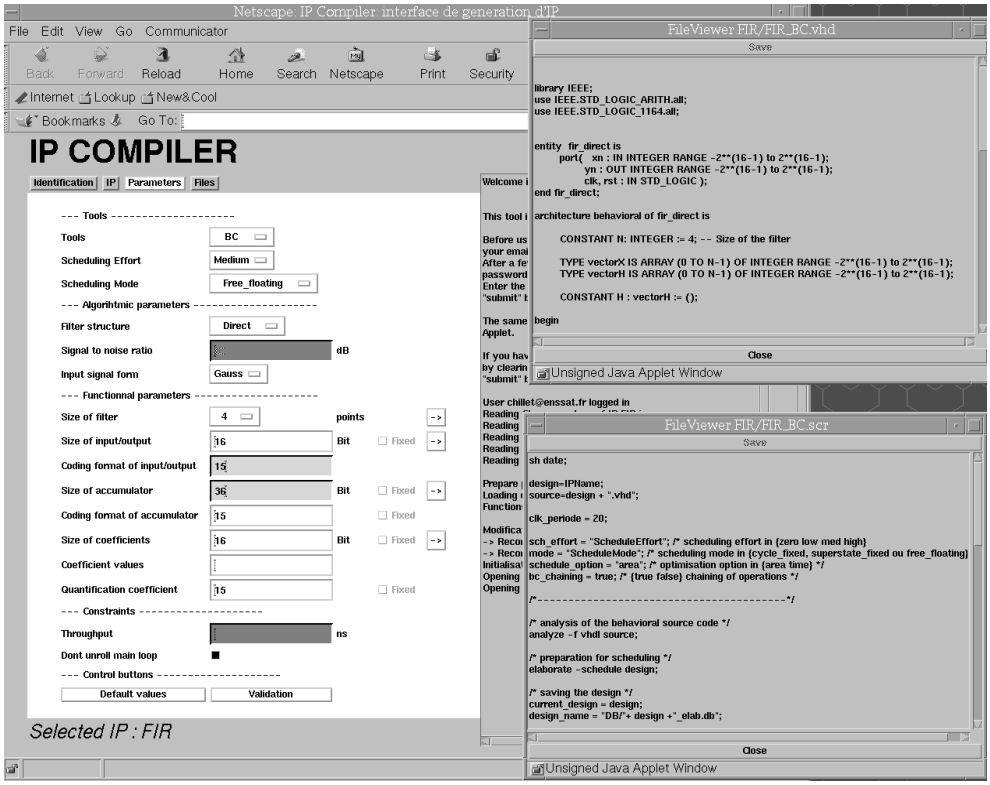

FIG. 3 – spécification des paramètres et fichiers générés

4.5 Générateur de l'Ip

Nous avons étudié et réalisé deux types d'Ips. Un Ip dont la généricité est obtenue directement sur le gabarit, et plusieurs Ips (RIF, RII) dont le nombre et la valeur des coefficients en sont les paramètres. La figure 3 représente un exemple possible de spécification de l'Ip au travers d'une page HTML [10]. La spécification est réalisée à partir des coefficients du filtre. Les fichiers de synthèse et de script sont générés automatiquement à partir des spécifications entrés par l'utilisateur.

5 Conclusion

Dans ce papier, nous avons présenté un flot de conception complet permettant à un intégrateur d'Ip de paramétrer un Ip de niveau comportemental pour l'intégrer dans son système avec ses propres contraintes. L'Ip comportemental est modélisé au travers d'un ensemble de paramètres et d'interaction, fourni par le créateur du générateur d'Ip. Cette méthodologie s'appuie sur les connaissances des concepteurs d'Ip.

A partir d'une interface Internet (Ip Compiler), l'intégrateur d'Ip met au point l'Ip pour son application. Au fur et à mesure de la spécification l'intégrateur est informé de la faisabilité de la solution qu'il développe (une résolution de règle semi-automatique est fournie), et peut estimer les performances générales du bloc. Finalement, l'utilisateur récupère un ensemble de fichiers (script de synthèse et VHDL) lui permettant de synthétiser l'Ip comportemental en respectant les contraintes fournies.

Références

[1] R. A. Bergamaschi and W. R. Lee. Designing systems-on-chip using cores. In *37 thProceedings of the Design Automation Conference*, pages 420–425, Los Angeles, June 2000. ACM Pres.

[2] A. Reutter and W. Rosenstiel. An efficient reuse system for digital circuit design. In *Proceedings of the Int'l Conference on Design Automation and Testing in Europe*, pages 38–43, Munich, Germany, March 9–12 1999. IEEE Computer Society Press.

[3] Semiconductor Industry Association. The national technology roadmap for semiconductors. Technical report, http ://notes.sematech.org, 1999.

[4] D. MacMillen, M. Butts, R. Camposano, D. Hill, and T.W. Williams. An industrial view of electronic design automation. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 19(12) :1428–1448, December 2000.

[5] D. Gajski, A. C-H. Wu, V. Chaiyakul, S. Mori, T. Nukiyama, and P. Bricaud. Essential issues for ip reuse. In *Proceeding of Asian and South-Pacific Design Automation Conference*, pages 37–42, Yokohama, Japan, January 25–28 2000. IEEE Computer Society Press.

[6] VSIA Architecture Document. http ://www.vsi.org.

[7] Projet RNRT MILPAT. http ://archi.enssat.fr/milpat/milpat.htm.

[8] Behavioral Compiler Synopsys. http ://www.synopsys.com/products/.

[9] D. Menard and O. Sentieys. Influence du modèle de l'architecture des dsps virgule fixe sur la précision des calculs. In *18e Colloque GRETSI sur le Traitement du Signal et des Images*, Toulouse, France, September 10–13 2001.

[10] Ip Compiler. http ://archi.enssat.fr.