Intégration Optimisée de Composants Virtuels orientés TDSI par la Synthèse d'Architecture

P. COUSSY, A. BAGANNE, E. MARTIN, E. CASSEAU

LESTER, Université de Bretagne Sud, Rue de Saint Maudé, BP 92116- 56321 Lorient Cedex

Prénom.nom@univ-ubs.fr

Résumé – La complexité grandissante des Systèmes sur Silicium (SoC) oblige les concepteurs à relever le niveau d'abstraction de leur description et à réutiliser des blocs fonctionnels préconçus, décrits au niveau RTL, de plus en plus complexes. La principale difficulté lors de la réutilisation de ce type de composants bas niveau provient de leur intégration (interfaçage avec le reste du système). Dans le cadre du projet *RNRT ALIPTA* nous proposons d'une part de relever le niveau d'abstraction des descriptions en introduisant la notion de *composant virtuel comportemental* et d'autre part de générer un cœur d'IP *RTL* en synthétisant sous contraintes d'intégration sa description comportementale à l'aide d'outils de synthèse haut niveau. Dans cet article nous rappelons brièvement le flot de conception système. Nous présentons les phases d'analyse garantissant la faisabilité de la synthèse en fonction des contraintes algorithmiques de l'IP et d'intégration et abordons la modélisation de l'ensemble de contraintes. Nous présentons ensuite une expérience de réutilisation de composants virtuels pour le traitement de l'image avec un quantificateur initialement connecté à une *DCT* puis à une *TO*. Nous comparons les résultats obtenus en appliquant une méthode classique d'intégration de composant virtuel au niveau *RTL* par synthèse d'un *wrapper* avec la méthode proposée.

Abstract *– The growing complexity of System-on-chip architecture lead designers to leverage the abstraction level of their description and also to re-use more and more complex function described at the RTL level. The main problem when re-using such low-level component comes from their integration. We propose to raise the abstraction level of IP synthetizable models by introducing the concept of behavioral IP and to generate a RTL IP core by synthesizing under integration constraints its behavioral description with HLS tools. In this paper, we present the system design approach and describe the IP design flow under constraints. As experiment, we trade-off the results obtained by applying traditional methodology, based on RTL IP and wrapper, and the proposed approach. The used application is a quantizer bloc first connected to a DCT component and re-use in an application based on the DWT transformation.*

1. Introduction

L'intégration des composants virtuels *(VC, IP)* est un problème majeur dans la conception des systèmes sur silicium SoC qui nécessite de nouvelles méthodologies pour être résolu. En effet, la complexité grandissante ainsi que la nature hétérogène des applications actuelles requièrent la réutilisation de composants de plus en plus complexes. Ces blocs peuvent représenter des fonctions de domaines spécifiques tels que le traitement du signal (DCT, FFT) ou les télécommunication (Viterbi, Turbo Codes). La principale difficulté lors de la réutilisation de ces composants provient de leur intégration : les problèmes de communication peuvent faire échouer la conception d'un SoC. Pour concevoir une unité d'interface d'un IP, l'intégrateur système doit donc prendre en compte les caractéristiques fonctionnelles du composant mais aussi les contraintes d'intégration.

Le problème d'intégration de composants virtuels peut être résolu de deux façons : (1) En réutilisant des IP préconçus ou pré routés et un adaptateur *(wrapper)*, (2) en utilisant la synthèse sous contraintes et un modèle générique d'interface.

La première solution est basée sur l'utilisation d'IPs *"soft"* qui sont, suivant la taxonomie VSIA [1], le plus haut niveau d'abstraction pour les modèles d'IPs synthétisables. Ces blocs fonctionnels sont fournis au niveau transfert de registre (RTL) dans un langage de description matériel (VHDL, Verilog) et sont synthétisés à l'aide d'outils de synthèse logique. Bien que de telles descriptions puissent être paramétrables, elles reposent sur un modèle d'architecture figée avec des degrés de personnalisation très réduits. Ce manque de flexibilité est particulièrement vrai pour l'unité de communication dont les ordres d'acquisition et de production de données, le comportement temporel et les protocoles d'échanges sont établis. Ces IPs sont donc connectés au bus système au travers d'interfaces spécifiques ou d'adaptateurs. Différentes approches tentent de faciliter ce mode d'intégration en définissant des méthodologies de conception [3]. La génération de *wrapper* reste cependant un processus largement manuel et sujet aux erreurs. Dans les applications TDSI, la quantité de données échangées entre les composants peut être très grande, faisant aboutir à la conception de wrappers dont la taille peut être aussi importante que l'IP lui-même. Ces interfaces de communication augmentent d'une part la surface mais surtout réduisent les performances du système. En effet, dans certains cas, la latence introduite par l'adaptateur peut conduire à la violation des contraintes temporelles d'E/S et de ce fait faire échouer la conception d'un SoC [2].

La deuxième solution, que nous proposons, génère un cœur d'IP RTL en synthétisant sous contraintes d'intégration, sa description comportementale. La solution consistant à utiliser la synthèse sous contraintes d'IP requière des outils de synthèse

haut niveau adaptés, une modélisation adéquate des contraintes et une phase d'analyse détectant les inconsistances des contraintes d'intégration. Certains travaux tentent de résoudre des problèmes spécifiques en utilisant la synthèse architecturale sous contraintes. Ainsi, dans [7] les auteurs réalisent un ordonnancemement relatif sous contraintes temporelles après la phase d'allocation. La solution proposée garantie l'existence d'une solution en présence de délais non bornés. BC, SystemC compiler [10], et Monet [11], en plus des modes de synthèse *"super state"* et *"free floatting"*, proposent tout comme CALLAS [9], un mode *"cycle fixed"* maintenant, s'il existe une solution d'ordonnancement, un comportement identique du composant avant et après synthèse [12]. Cependant, la communication n'est pas dissociée du traitement et est décrite à l'aide d'instruction de synchronisation *wait* dans la description source du composant. Ces outils ne prennent de plus pas en compte les fenêtre temporelles d'arrivée lors de la synthèse.

Dans notre cas, nous proposons une approche basée sur l'utilisation de la synthèse haut niveau, sous contraintes d'intégration système qui vise la spécification mais aussi la conception de l'IP. Nous étendons pour cela le formalisme proposé dans [7] dédiés aux applications dominées par le contrôle, afin de l'orienter vers des applications TDSI. Les contraintes temporelles prises en compte sont variables mais bornées.

Dans cet article nous formulons le problème de l'intégration d'IP sous contraintes et nous décrivons une approche pour la conception de SoC utilisant des IPs algorithmiques. Nous présentons brièvement le formalisme utilisé pour la modélisation des contraintes d'intégration développé dans le cadre du projet *RNRT ALIPTA*. Finalement nous présentons une expérience comparative, entre l'intégration classique d'un composant virtuel au niveau RTL par synthèse d'un "wrapper" et l'intégration par la synthèse d'architecture d'un composant virtuel comportemental.

2. Problématique d'Intégration

La conception d'un SoC débute par une spécification de la fonctionnalité de l'application visée. Le concepteur système sélectionne les IP dans une base de donnée en considérant certains critères tels que l'algorithme, la surface, la consommation … S'ensuit une phase d'exploration architecturale basée sur des techniques de co-design : partitionnement, analyse de performances, synthèse des communications et génération d'interface. L'architecture de communication du SoC décrit la façon dont les composants communiquent pour se synchroniser et échanger des données. L'étape de raffinement de la communication permet ainsi de déterminer : les protocoles utilisés, les contraintes temporelles pour chaque donnée ou groupe de données, format de bus... La communication entre un IP et les autres composants du système est réalisée aux travers de ports et de canaux par lesquels transitent les données.

La modélisation des contraintes doit supporter les caractéristiques précédemment décrites pour piloter efficacement la synthèse du composant virtuel (Figure 1). Les contraintes d'intégration comprennent les caractéristiques de communication mais aussi le débit de l'application (période d'itération) et les contraintes technologiques. Une intégration réussie d'un composant virtuel requière de la part du concepteur de SOC de (1) synchroniser les composants [15] (échanges de données, protocole) (2) temporiser les données pour garantir les contraintes temporelles et l'ordre des données [13] (3) convertir les protocoles entre blocs "incompatibles" [4], [5].

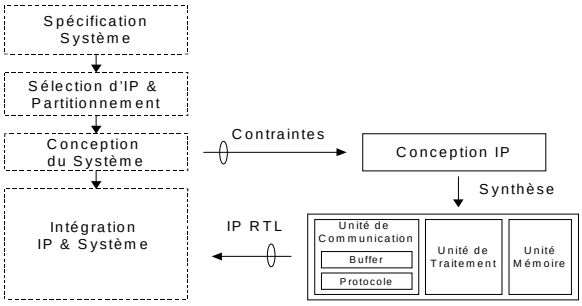

Figure 1 : Flot de conception d'un SoC

Un nouveau flot de conception basé sur la synthèse sous contrainte est nécessaire pour résoudre le problème de la réutilisation d'IP. Ceci inclus (1) un ensemble de modèles pour représenter les contraintes système et les contraintes algorithmiques (2) des méthodes et techniques pour synthétiser les différentes parties composant l'architecture d'un IP.

3. Approche de conception

Dans [20], nous avons proposé de relever le niveau d'abstraction des descriptions en introduisant la notion de *composant virtuel comportemental*, décrit sous une forme algorithmique dans un langage de haut niveau (*VHDL, SystemC*, etc.) et autorisant un haut degré de flexibilité par le biais d'un jeu de paramètres génériques. Outre l'algorithme qui modélise la fonction réalisée, ce type de composant virtuel doit comporter un modèle flexible de l'unité de mémorisation nécessaire au traitement. L'architecture d'un *IP comportemental* repose sur les unités fonctionnelles suivantes : unité de traitement (UT), unité de mémorisation (UM), unité de contrôle (UC) et unité de communication (UCOM). Dans le cadre du projet *RNRT ALIPTA* [18], nous proposons de générer un cœur d'IP RTL en synthétisant sous contraintes d'intégration sa description comportementale. La description des protocoles d'entrée/sortie ainsi que des caractéristiques des séquences de données, échangées entre l'IP et les composants du système, sont spécifiés par l'intégrateur. Le flot de synthèse décrit dans la Figure 2, s'appuie sur un ensemble successif de transformation et raffine une description comportementale en une description structurelle.

La spécification de l'IP est modélisée par un Graphe Flot de Signaux *(SFG)*. Un graphe de contrainte algorithmique *ACG* est ensuite généré en pondérant les arcs du *SFG* par les temps de propagation des opérateurs et la cadence. Une première phase d'analyse permet de vérifier la cohérence entre la cadence d'itération, les dépendances de données de l'algorithme et les contraintes technologiques. Pour supporter les architectures spécifiques des applications TDSI, nous avons étendu la modélisation de type *SIF (Sequential Intermediate format [7])*. Ce modèle formel nommé *IOCG (IO constraint Graph)* permet l'expression des contraintes d'intégration pour chacun des bus

connectants l'IP aux autres composants du système. Il supporte ainsi, la modélisation (1) des modes d'échanges de données, (2) du type de transferts, (3) de l'indéterminisme dans les dates d'arrivées des données dû aux phases d'arbitrage pour l'accès au bus partagé, (4) d'un séquencement des données échangées. Il permet en outre l'expression de mécanisme lié aux protocoles (synchrone / asynchrone) et les latences introduites par les conversions de format de données. Les contraintes d'intégration et les contraintes algorithmiques de l'IP sont ensuite fusionnées pour fournir un graphe détaillé des contraintes *GCG (Global Constraint Graph)* qui permet (1) d'analyser la pertinence des contraintes d'intégrations système vis à vis de l'algorithme et (2) de détecter les points de synchronisation entre l'environnement et le composant.

Basée sur les résultats de cette analyse, la synthèse de l'unité de traitement est réalisée à l'aide de l'outil GAUT (Générateur Automatique d'Unité de Traitement [18]) dont le cœur d'ordonnancement a été modifié afin de prendre en compte les contraintes temporelles spécifiées par l'intégrateur du composant virtuel. Cette unité de traitement représente le chemin de données de l'IP et inclut des opérateurs de calcul (additionneurs, multiplexeurs,...). L'exécution des unités fonctionnelles est pilotée par un l'unité de contrôle, symboliquement représenté par une FSM. La synthèse de l'UT génère un ensemble de contraintes modélisé sous la forme d'un *IPERM (IP Execution Requirement Model [2])*. Ces contraintes représentent (1) les dates de production et de consommation des données par l'UT (2) les bus associés aux transferts de ces données entre l'UCOM et l'UT.

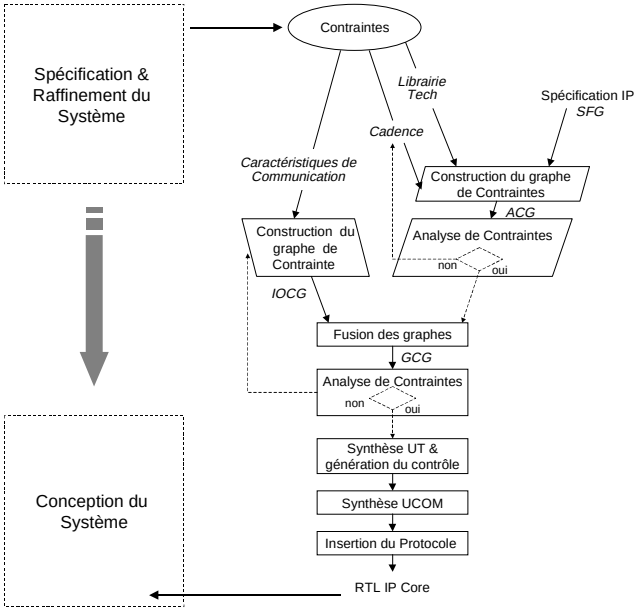

Figure 2 : Approche de conception

L'unité de communication est composée d'éléments mémorisants, de multiplexeurs, de démultiplexeurs assurant les échanges de données et d'une FSM qui implémente le protocole spécifié par le concepteur du SoC. Cette unité est synthétisée sous contrainte et a posteriori de la synthèse de l'UT. Pour cela les modèles *IPERM* et *IOCG* sont utilisés afin de dimensionner et d'optimiser d'une part les éléments mémorisants réalisant l'adaptation temporelle des séquences d'E/S système aux séquences de l'UT et d'autre part de générer les multiplexeurs et démultiplexeurs aiguillant les E/S entre les bus internes, les bus système et les éléments mémorisants.

Le découpage en plusieurs unités fonctionnelles sur lequel nous basons l'architecture de nos composants virtuels permet la séparation de la communication et du traitement [17]. La technique de synthèse utilisée, couplée au modèle d'architecture, garantie une intégration optimisée du composant virtuel puisque l'unité de traitement et l'unité de communication sont adaptées aux contraintes d'entrées/sorties imposées par le système.

4. Expérience

Dans cette section, nous présentons une expérience de réutilisation de composants virtuels de Quantification *(Q)*. Ce bloc de quantification provenant d'une application JPEG, basée sur une Transformée en Cosinus Discrète 8x8 *(DCT)*, et est réutilisé dans une application JPEG2000, basée sur une Transformée en Ondelettes 8x8 *(TO)*. Nous comparons en terme de coût (nombre de registres et d'opérateur, latence) les résultats obtenus en appliquant une méthode classique d'intégration de composant virtuel au niveau RTL par synthèse d'un *wrapper* avec la méthode d'intégration optimisée basée sur la synthèse d'architecture d'un IP comportemental que nous proposons.

Nous avons conçu, dans un premier temps, un bloc quantification au niveau RTL respectant les contraintes de la DCT et de l'application : 1 donnée lue sur un port tous les cycles de période 30ns (*Cad E*, Tab 1), 1 donnée produite sur un port tous les cycles *(Cad O)* après une latence de 5 cycles *(L)*. Ce composant contient 4 diviseurs, 1 multiplieur, un additionneur, 10 registres de traitements et 64 registres pour les coefficient de quantification (cf. Tab 1).

TAB 1 : Quantificateur initial

	Cad E	Cad O	L	Div	Mult	Add	Reg
RTL	1/1	1/1	5	4	1	1	74

L'ordre des entrées est l'ordre de production des résultats de la DCT à savoir : chaque bloc est fourni donnée par donnée, ligne à ligne (TAB 2). Nous avons réutilisé dans un deuxième temps ce composant Q dans une application JPEG2000 en le connectant à un bloc TO. Les contraintes définies pour l'application sont : 1 donnée lue sur un port tous les 3 cycles de période 60ns, 1 donnée produite sur un port tous les 3 cycles après une latence de 5 cycles. Le nouvel ordre d'acquisition des entrées est celui de production des résultats de la TO décrit dans TAB 3. L'ordre de production des résultats du bloc Q doit être celui de la TO et non plus celui de la DCT.

TAB 2: Séquence résultats produite par la DCT

R00	R01	R02	R03	R04	R05	R06	R07
R10	R11	R12	R13	R14	R15	R16	R17
R20	R21	R22	R23	R24	R25	R26	R27
R30	R31	R32	R33	R34	R35	R36	R37
R40	R41	R42	R43	R44	R45	R46	R47
R50	R51	R52	R53	R54	R55	R56	R57
R60	R61	R62	R63	R64	R65	R66	R67
R70	R71	R72	R73	R74	R75	R76	R77

TAB 3: Séquence résultats produite par la TO

R72	R02	R43	R05	R74	R16	R50	R21
R03	R01	R06	R04	R17	R13	R22	R20
R47	R10	R50	R14	R65	R24	R66	R27
R11	R07	R15	R12	R25	R23	R30	R26
R77	R32	R67	R35	R77	R51	R73	R54
R33	R31	R36	R34	R52	R44	R55	R53
R70	R40	R71	R45	R75	R57	R76	R63
R41	R37	R46	R42	R60	R56	R64	R61

Afin d'adapter l'ordre des séquences de la TO à celui du quantificateur nous avons réalisé manuellement un *wrapper* qui contient, afin d'effectuer le ré ordonnancement, des données 64 registres pour les entrées et 64 registres pour les sorties (cf. Figure 3). Les cadences d'acquisitions et de productions de l'application initiale JPEG étant différentes de la nouvelle application JPEG2000 le *wrapper* doit aussi assurer la temporisation des E/S. Cette phase d'adaptation introduit une latence de 64*3 cycles pour les entrées et 64*3 cycles pour les sorties puisque l'ordre de production des résultats du bloc Q doit être celui de la TO.

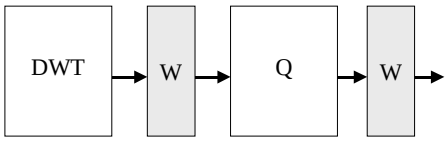

Figure 3 : wrapper

Avec les contraintes d'intégration précédemment décrites, la synthèse du composant virtuel comportemental, réalisant le même traitement (même algorithme de calcul), génère une solution optimisée pour le nouveau système basé sur la TO. Ainsi, l'unité de traitement du composant RTL généré contient 3 diviseurs, 1 multiplieur, 1 additionneur, 5 registres pour les calculs et 64 registres pour les coefficients de quantification (IP comp. Table 4). L'unité de communication inclus 5 registres afin d'adapter les contraintes système aux contraintes architecturales de l'UT.

Table 4 : Résulta de l'intégration de l'IP Quantification

	Cad E	Cad O	L	Div	Mult	Add	Reg
RTL + wrapper	1/3	1/3	133	4	1	1	202
IP comp.	1/3	1/3	5	3	1	1	71

La latence pour produire le premier résultat est de 133 cycles avec un méthode classique d'intégration contre 5 cycles en utilisant la méthode proposée. Le *wrapper* et le composant RTL totalisent 202 éléments mémorisants contre 71 dans l'IP RTL obtenu après synthèse de la description comportementale sous contraintes d'intégration.

Cette expérience permet de mettre en évidence les surcoûts matériels et temporels liés à l'utilisation de méthode d'intégration traditionnelle d'IP. Il est important de remarquer que la latence introduite par le *wrapper* peut faire échouer la réutilisation d'un composant virtuel. De plus les IP décrits au niveau RTL ne peuvent être réutilisés qu'à un débit inférieur ou égal au débit maximum pour lequel ils ont été conçus. La méthode proposée s'avère particulièrement intéressante pour les applications TDSI qui traditionnellement travaillent sur de forts volumes de données, et sont souvent soumises à des contraintes temps réel fortes.

5. Conclusion

Dans cet article, nous avons discuté du problème de la réutilisation des IPs. L'intégration de ce type de composant demande au concepteur de système de prendre en compte d'une part les aspects temporels et fonctionnels du composant virtuel et d'autre part les contraintes du système dans lequel il doit être inséré. Nous avons proposé une approche basée sur la synthèse haut niveau sous contraintes et l'utilisation de composants virtuels comportementaux. L'IP comportemental est dans ce cas adapté à chaque système dans lequel il doit être intégré. L'expérience proposée a souligné les surcoûts d'une intégration avec une méthode traditionnelle et a montré l'intérêt de l'approche proposée qui permet une intégration optimisée.

Remerciements :

Les travaux présentés dans cet article sont réalisés dans le cadre du projet RNRT ALIPTA.

Références

[1] Virtual Socket Interface Alliance, *http://www.vsi.org*
[2] P. Coussy, A. Baganne, E. Martin *A Design Methodology for IP integration*, Proc. of ISCAS, 2002
[3] M. Keating, P. Bricaud Reuse *Methodology Manual for System-on-a-Chip Design* Kluwer Academic Publishers, 1999
[4] R. Lysescky, F. Vahid, T. Givargis, *Techniques for reducing Read Latency of Core Bus Wrapper* , Proc. of DATE, 2000
[5] G. Cyr, G. Bois, M. Aboulhamid, *Synthesis of communication Interfaces for SOC using VSIA recommendation*, Proc. of DATE, 2001
[6] G. Savaton, P. Coussy, E. Casseau, E. Martin *A Methodology for Behavioral Virtual Component Specification Targeting SoC Design with High-Level Synthesis Tools*, Proc. of FDL 2001
[7] D. Ku and G. De Micheli, *Relative Scheduling Under Timing Constraints: Algorithms for High-Level Synthesis of Digital Circuits*, IEEE Trans. CAD/ICAS, vol. 11, p. 696-718, 1992
[8] R. Gupta and G. De Micheli, *VULCAN - A System for High-Level Partitioning of Synchronous Digital Circuits*, Technical Report CSL-TR-91-471, 1991
[9] A. Stoll and P. Duzy, *High-Level Synthesis from VHDL with Exact Timing Constraints*, Proc. of DAC, 1992
[10] T. Ly, D. Knapp, R. Miller and D. MacMillen, *Scheduling Using Behavioral Templates*, Proc. of DAC, 1995
[11] J. P. Elliott, *Understanding Behavioral Synthesis. A Practical Guide to High-Level Design*, Kluwer Academic Publishers, 2000.
[12] D. Knapp, and al. *Behavioral Synthesis Methodology for HDL-Based Specification and Validation*, Proc. of DAC, 1995
[13] A. Baganne, JL. Philippe, E. Martin *A Formal Technique for Hardware Interface Design*, Proc. of ISCAS, 1997
[14] Y; Liao and C. Wong, *An algorithm to compact VLSI symbolic layout with mixed constraints*, IEEE Trans. CAD/ICAS, vol. 2, pp62-69, 1983
[15] R. Gupta, *Co-Synthesis of Hardware and Software for Digital Embedded Systems*, Kluwer academic publishers, 1995
[16] R. Gupta and G. De Micheli, *A Co-Synthesis Approach to Embedded Systems Design Automation, Design Automation of Embedded Systems*, Vol. 1, No. 2, pp. 69-120, 1996
[17] A. Rowson and A.L. Sangiovanni-Vincentelli, *Interface-Based Design, Proc. of DAC*, 1997
[18] http://lester.univ-ubs.fr:8080
[19] P. Coussy *Modélisation des contraintes d'entrée/sortie* Rapport technique, 2003
[20] G. Savaton and al. *Composants Virtuels Comportementaux pour Applications de Compression d'Images*, GRETSI, 2001