RENPAR'2002 proceedings - Rencontres Francophones du Parallélisme, Hammamet (Tunisie), 10-13 April 2002, pp. 341-345

RENPAR'14 / ASF / SYMPA
Hamamet, Tunisie, 10 - 13 avril 2002

Nouvelles méthodes de conception de systèmes électroniques intégrés

Damien Hubaux[1], Lotfi Guedria[1], Luc Vandendorpe[2], Michel Verleysen[3] et Jean-Didier Legat[3]

[1] *CETIC asbl*
rue Clément Ader, 8
6041 Gosselies - Belgique
damien.hubaux@cetic..be
lotfi.guedria@cetic..be

[2] *UCL - TELE*
place du Levant, 2
1348 Louvain-la-Neuve - Belgique
vandendorpe@tele.ucl.ac.be

[3] *UCL - DICE*
place du Levant, 3
1348 Louvain-la-Neuve - Belgique
verleysen@dice.ucl.ac.be
legat@dice.ucl.ac..be

RÉSUMÉ: Dans le cadre de notre projet de recherche, nous allons nous intéresser aux techniques de conception de systèmes intégrés. Nous mettrons l'accent sur des aspects comme la flexibilité, les possibilités de configuration et de réutilisation. Ces objectifs nous amènent à nous placer à un haut niveau d'abstraction. Nous allons appliquer ces méthodes à la modélisation et l'implémentation de nouvelles fonctions et techniques employées en télécommunications telles que le turbo-codage et la turbo-détection.

MOTS-CLÉS : Spécification, Modélisation, Méthodologie de design, développement niveau système.

1. Introduction

La méthodologie actuelle de design d'un système consiste, après spécification, à le partitionner en modules hardware et software, qui sont développés séparément. Le software est typiquement implémenté en langage C/C++ et le hardware en langage de description hardware (HDL, tel que Verilog ou VHDL). Ensuite, la co-simulation du système entier est réalisée. Elle constitue la phase cruciale vue sa lenteur: un surcoût important résulte de la complexité du design et de la communication entre software et hardware dans un environnement hétérogène C/HDL. Par ailleurs, la vérification se fait très tardivement dans le cycle du design, ce qui implique de lourdes et coûteuses modifications en cas d'erreurs.

Faire du design au niveau système passe d'abord par la définition et la validation d'une méthodologie de conception et de vérification, en tenant compte des réalités rencontrées dans ce domaine. En effet, la réduction de la taille des composants électroniques autorise aujourd'hui une intégration de plus en plus forte. Il est possible d'avoir sur la même puce des microprocesseurs, des DSP (Digital Signal Processor), des mémoires, de la logique programmable et de former ainsi des SoC (System on Chip). Le coût de développement de plus en plus important de ces SoC conduit à s'orienter vers le développement de plates-formes destinées à une classe d'applications, et donc à usage multiple. La mise en œuvre de ces plates-formes est une tâche complexe et la méthodologie traditionnelle de conception devient mal adaptée, notamment en terme de temps de simulation et techniques de vérification.

De nouveaux besoins du design au niveau système apparaissent: la possibilité d'intégrer et vérifier software et hardware plus tôt dans le processus, d'explorer très tôt les degrés de liberté en évaluant, par exemple, plusieurs scénarios de partitionnement hardware/software. Ainsi il est possible de

définir une architecture efficace et de pouvoir réutiliser ou importer plus facilement des parties d'un autre design.

2. Emergence de langages pour la conception de haut niveau

Les nouveaux besoins en matière de développement ont favorisé l'émergence d'environnements de conception. Étant donné que le software est typiquement développé en C/C++ et le hardware en langage HDL, on assiste à deux principales approches: certaines initiatives tentent d'adapter le C/C++ à la description hardware, en tenant compte de ses aspects fondamentaux tels que la concurrence et l'aspect temporel, d'autres cherchent à étendre les HDLs à la description système [1]. Une troisième voie est la spécification de nouveaux langages.

2.1 Aperçu de certaines initiatives et des approches associées

Superlog [2]: langage tendant à étendre le Verilog, langage HDL, au design système en y incluant des possibilités de programmation C, système et vérification. D'origine privée, Superlog a été donné à Accellera (organisme indépendant de normalisation de langages HDL).

SpecC [3]: langage de description et de spécification système basé sur le C, avec des extensions syntaxiques (qui le rendent incompatible à la norme ANSI). Ces extensions incluent, par exemple, des types de données adéquats, des commandes pour supporter la concurrence et la description de machines à états. Il a été développé à l'université de Californie, Irvine. Au delà du langage lui même, les travaux menés par l'équipe de D. Gajski introduisent des concepts intéressants pour la modélisation système et les techniques de raffinement [4]. Le développement de SpecC se fait dans le cadre du STOC (SpecC Technology Open Consortium).

SystemC [5]: SystemC est une plate-forme de modélisation composée de librairies de classes C++ et d'un noyau de simulation permettant de faire du design au niveau système, comportemental et au niveau de l'implémentation. Certains concepts de SystemC comme les interfaces, les canaux et les événements sont inspirés[6] de SpecC. Cette initiative se développe dans le cadre de l'OSCI (Open Source systemC Initiative)

Ptolemy [7]: Projet de recherche de l'université de Berkeley visant la modélisation, la simulation et le design de systèmes concurrents, temps réel et intégrés. Il s'agit d'un ensemble de modules Java.

Rosetta [8]: langage en cours de création qui compte offrir un support de modélisation avec des sémantiques adaptées aux différentes 'facettes' de la conception, telles que la consommation, le comportement ou le timing. En d'autres termes, Rosetta vise à mettre ensemble, dans l'environnement du langage, des informations hétérogènes appartenant à des domaines différents. Chaque domaine utilise sa propre sémantique pour décrire ses modèles. Rosetta n'impose pas un modèle commun, mais permet de partager des informations entre des modèles hétérogènes.

2.2 Les points clés de l'approche C/C++ en design système

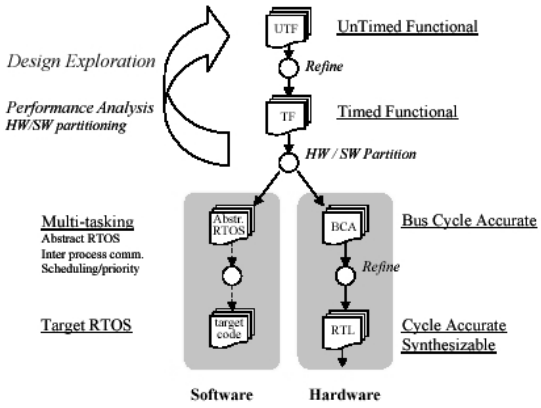
Dans la guerre actuelle des langages système, les approches basées sur le C/C++, notamment SystemC, semblent récolter le plus d'intérêt. Le C/C++ est largement utilisé par les ingénieurs système et software, et il est couramment adopté pour la spécification. L'utilisation du C/C++ pour modéliser toutes les parties d'un système autorise une meilleure flexibilité. On pourra par exemple traduire, plus aisément, des blocs software en hardware et vice versa. Un modèle abstrait en C/C++

peut rapidement être simulé pour valider et optimiser le design, explorer divers algorithmes et fournir aux développeurs du hardware et du software une spécification exécutable.

Le design basé sur C de systèmes sur puces(SoC) a déjà donné des résultats intéressants en terme de gain en temps de développement et vérification [9]. Cependant, une approche système ne repose pas simplement sur le langage. Il est essentiel d'avoir un support en terme de méthodologie et une disponibilité d'outils EDA (Electronic Design Automation). Dans ce sens, l'initiative SystemC semble être plus avantagée par rapport aux autres approches dans la mesure où elle est supportée par les principaux vendeurs EDA du marché. Le but poursuivi est de s'imposer en tant que nouveau standard de design système et échange d'IPs (blocs protégés par Propriété Intellectuelle).

3. Méthodologie de design

Il est important de pouvoir modéliser un système entier à plusieurs niveaux d'abstraction, allant de la spécification à l'implémentation [10]. Partant d'un niveau fonctionnel abstrait, le modèle est affiné en vue d'être partionné suivant une spécification architecturale. Cet affinement se poursuit progressivement en vue de tenir compte des détails de l'implémentation et des contraintes de synthèse [11]. Il est ainsi possible de surveiller la cohérence du système tout au long du processus du design. Les simulations peuvent être réalisées avec le même ensemble de test et le design peut ainsi être vérifié à chaque niveau, ce qui évite de lourdes modifications en phase finale. La fiabilité du design est améliorée en créant une spécification détaillée du système et en permettant de tester le hardware et le software par rapport à cette spécification tout au long du cycle de conception. Par ailleurs, le fait de garder le même langage pour tous les niveaux d'abstraction permet d'éviter des traductions propices aux erreurs.



Niveaux d'abstraction SystemC (source : OSCI)

L'adoption d'un langage système nécessite la définition d'une méthodologie de design adaptée. Il est envisageable de recourir à des outils formels en première phase de spécification tel que SDL ou UML [12] ou d'employer Matlab. La méthodologie doit spécifier clairement le processus d'affinement progressif des modèles et la façon dont la transition se fait d'un niveau d'abstraction à un autre. Par ailleurs il est aussi essentiel de définir les procédures de simulation, de vérification à suivre ainsi que les techniques de partitionnement.

Un autre point important dans le design système est la réutilisation et la flexibilité des modèles. En ce sens, il faut différencier l'aspect fonction de l'aspect communication des modules [13]. La

communication peut être modélisée à divers niveaux d'abstraction [14] offrant, par conséquent, plus de flexibilité au design.

Cependant, il ne faut pas s'attendre à avoir un outil de synthèse totalement automatisé pour générer systématiquement le design à partir de la spécification abstraite. Pour aller de la description au niveau système à l'implémentation, les différents niveaux d'abstraction doivent être bien définis et les méthodes de raffinement bien précises et claires. Les modèles peuvent ainsi être dérivés de leur prédécesseurs hiérarchiques par des transformations vérifiables [15].

4. L'exemple de SystemC

SystemC introduit de nouveaux concepts (par rapport au C++ classique) afin de supporter la modélisation et la description du hardware et ses caractéristiques inhérentes comme la concurrence et l'aspect temporel. Ces nouveaux concepts sont implémentés par des classes C++ tels que les modules, ports, interfaces, canaux, processus (threads et methods).

Concernant la communication, la librairie Master-slave de SystemC implémente un protocole abstrait qui peut être ensuite affiné en un protocole explicite.

D'autre part, seulement un sous ensemble du langage est synthétisable. Il est équivalent à celui d'un langage HDL classique (Verilog ou VHDL), et les outils actuels utilisent d'ailleurs le même noyau de compilation pour synthétiser les descriptions. Ce sous ensemble n'est certainement pas figé et pourrait varier en fonction des outils futurs.

Le support du software n'est pas encore assuré pour des parties aussi importantes que le système d'exploitation temps réel (RTOS) par exemple, mais SystemC se développe en ce sens. De plus, des travaux de recherches sont menés dans l'optique de générer le système d'exploitation à la carte à partir de librairies en partant du modèle abstrait de l'application décrit en SystemC [16].

Les fonctionnalités de SystemC sont en train de s'enrichir avec le développement de librairies de classes génériques ou spécifiques (SystemC-SV [17]). Les outils associés qui apparaîtront vont permettre de définir des méthodologies de design plus élaborées et complètes.

5. Conclusion

Notre travail consiste, en une première étape, à étudier et implémenter un système de turbo-codage et décodage. Nous disposons pour ce faire d'une plate-forme de design configurable à base de microprocesseur. Cet essai nous permettra d'évaluer les outils disponibles et déterminer les problèmes auxquels nous serons confrontés en spécification, simulation, synthèse et implémentation. Ce premier travail nous aidera à définir une méthodologie de design et de vérification nous ouvrant la voie pour implémenter et évaluer d'autres fonctions plus complexes tout en élargissant notre maîtrise de la conception système.

6. Références

[1] D.MALINIAK, "Design Languages Vie For System-Level Dominance" *ELECTRONIC DESIGN*.October2001,pp.53-60.

[2] *http://www.superlog.org/*

[3] *http://www.specc.org/*

[4] D.D.GAJSKI,J.ZHU,R.DÖMER,A.GERSTLAUER AND S.ZHU., "SpecC: Specification Language and Methodology," *Kluwer Academic Publishers, Dordrecht, Hardbound*, ISBN 0-7923-7822-9, March 2000.

[5] *http://www.systemc.org/*

[6] SYSTEMC LANGUAGE WORKING GROUP., "Functional Specification for SystemC 2.0," *Source: http://www.systemc.org/* , Version2.0-P, October 2001,pp. 11.

[7] *http://ptolemy.eecs.berkeley.edu/*

[8] *http://www.sldl.org/*

[9] K.WAKABAYASHI,T.OKAMOTO, "C-Based SoC Design Flow and EDA Tools: An ASIC and System Vendor Perspective," *IEEE transactions on computer aided design of integrated circuits and systems*, VOL.19,NO.12,December 2000,pp.1507-1522.

[10] A.GHOSH,J.KUNKEL,S.LIAO, "Hardware Synthesis from C/C++," *Proceedings of the DATE'99 conference*, pp.382-383, March99.

[11] L.SÉMÉRIA, AND A.GHOSH, "Methodology for Hardware/Software Co-verification in C/C++," *http://www.systemc.org/technical_papers.htm*

[12] SVARSTAD,G.NICOLESCU, AND A.A. JERRAYA, "A Model for Describing Communication between Aggregate Objects in the Specification and Design of Embedded Systems," http://www.systemc.org/ technical_papers.htm

[13] K.KEUTZER,S.MALIK,A.R.NEWTON,J.M.RABAEY AND A.SANGIOVANNI-VINCENTELLI., ,"System-Level Design: Orthogonalization of Concerns and Platform-Based Design," *IEEE transactions on computer aided design of integrated circuits and systems*, VOL.19,NO.12,December 2000,pp.1507-1522.

[14] W.O.CESÁRIO,G.NICOLESCU,L.GAUTHIER,D.LYONNARD, AND A.A.JERRAYA, "Systèmes de processus légers: concepts et exemples", *IEEE Design & Test of Computers,September*-October 2001,pp. 8-19.

[15] D.D.GAJSKI, "System Level Design, Quo Vadis?," *DATE2002 Tutorial: Specification Beyond RTL*, March 2002.

[16] S.YOO,G.NICOLESCU,L.GAUTHIER ANDA.A.JERRAYA, "Automatic Generation of Fast Timed Simulation Models for Operating Systems in SoC Design," *DATE2002 Proceedings*, March 2002,pp.620-627.

[17] R.SIEGMUND, D.MÜLLER, "SystemC-SV : An Extension of SystemC for Mixed Multi-Level Communication Modeling and Interface-Based System Design", *DATE2001 Proceedings*, March 2002.