

# Une approche modèle pour la conception conjointe de systèmes embarqués hautes performances dédiés au transport

Jean-Luc DEKEYSER , Sébastien LE BEUX, Philippe MARQUET

Inria Futurs Lille  
dekeyser@lifl.fr

**Résumé** – *Les technologies du génie logiciel ne sont pas réservées au développement d'applications Web. Nous montrons dans ce papier que l'approche dirigée par les modèles, associée à des modèles qui permettent l'expression à un haut niveau d'abstraction le parallélisme d'une application vont permettre de produire de façon automatique du code VHDL à partir de modèles exprimés en UML. Les types d'applications concernées recouvrent le traitement de signal systématique avec une partie contrôle. Le secteur du transport, en particulier l'automobile, est par-delà même un domaine d'application privilégié de notre environnement de conception.*

**Mots clefs:** modèle, parallélisme, traitement de signal, UML, VHDL, FPGA.

## 1 Introduction

Les « System On chip » permettent de développer des systèmes logiciel et matériel conjoints pour des applications la plupart du temps embarquées avec des contraintes de temps réel plus ou moins dures. Les premières générations de SoC sont déjà multi-processeur : on retrouve généralement un processeur General Purpose de type Risc (ARM etc), un DSP en particulier pour le traitement de signal, des composants matériels ASIC qui correspondent à des tâches câblées à partir d'une modélisation du logiciel puis de sa synthèse, enfin une surface du silicium peut être réservée pour une partie matérielle programmable de type FPGA.

L'évolution des capacités d'intégration des chips aujourd'hui permet de concevoir des systèmes plus « parallèles » où l'on retrouve plusieurs processeurs homogènes ou hétérogènes souvent reliés par un réseau interne au SoC. On parle encore de NoC pour « Network on Chip ».

Les prochaines générations de SoC se doivent de proposer encore plus de puissance tout en gardant un niveau de consommation énergétique faible et en utilisant au mieux les possibilités d'intégration offerte par la technologie. Une nouvelle classe de SoC apparaît, il s'agit de proposer non

plus une grille de cellule de hardware programmable tel que FPGA mais de proposer une grille de Processeur Elémentaires ou la topologie de la grille devient reconfigurable. C'est le cas par exemple du processeur développé par l'IMEC qui en amont propose un processeur maître de type VLIW et en aval une grille de PE

Les environnements de co-modélisation logiciel/matériel ou de co-design devront être capable de tirer partie de ces nouvelles architectures. Un compromis entre exécution parallèle du logiciel et consommation d'énergie doit être trouvé et ce à partir d'une modélisation unique permettant une évaluation du comportement d'une application placée sur cette architecture.

Dans cet article, nous allons montrer comment une approche dirigée par les modèles peut faciliter le développement conjoint de tels systèmes hautes performances. L'environnement Gaspard sera présenté, il est dédié aux applications et architectures à parallélisme régulier. Enfin une application parfaite pour Gaspard correspond aux algorithmes mis en œuvre dans les radars automobiles. Une démonstration de génération de code VHDL automatique à partir de la modélisation de l'algorithme parallèle à un très haut niveau d'abstraction montrera tout le potentiel en matière de développement de systèmes embarqués de ce type et l'intégration dans un FPGA pour une évaluation réel. Enfin pour conclusion, le développement d'une architecture particulière massivement parallèle pour ce genre d'application sera montré comme une perspective réaliste et novatrice sur un SoC.

## 2 Co-modélisation

Dans une approche classique les conceptions du logiciel et celle du matériel sont séparées et ce n'est qu'à la fin du processus de conception que les différentes parties sont testées ensemble. Le mariage de la conception logicielle et matérielle, appelé co-design, permet de maîtriser la conception des systèmes complexes et de promouvoir leur pérennité afin d'augmenter la productivité. L'analyse des performances avant la fabrication permet une

exploration rapide de plusieurs alternatives d'architecture, ce qui offre au concepteur une meilleure visibilité et une grande réactivité vis-à-vis des changements technologiques (fiabilité, optimisation, flexibilité, migration etc.).

Cette nouvelle approche constitue actuellement un enjeu dans l'accompagnement du développement de la technologie. En fait, le processus du co-design du SoC considère conjointement la conception des deux technologies (logicielle et matérielle) et couvre les différents cycles de développement, depuis la spécification jusqu'à la réalisation du SoC. Le processus comporte essentiellement les étapes de spécification, de co-synthèse et d'intégration. En plus, il doit inclure des tâches de vérification et de validation au cours de la conception.

L'objectif consiste donc à maîtriser la complexité grandissante des systèmes embarqués à travers une approche « co-design ». Celle-ci vise à aborder de front, et simultanément la conception (matériel et logiciel), de la spécification au prototypage de SoC.

En effet, l'exploration de l'espace des solutions, la co-synthèse et le prototypage représentent des tâches ultimes du co-design et visent à évaluer et affiner des solutions d'implémentation du système qui matérialisent les choix de réalisation de SoC. L'analyse des performances au cours du processus de conception peut être considérée comme une première estimation afin de guider l'exploration de l'espace des solutions [1]. Le prototypage permet une confirmation de cette évaluation. Tous deux tiennent compte d'une architecture cible.

## 2.1 Le modèle Gaspard

Le processus de co-design inclut donc plusieurs étapes de raffinement de la conception et des tâches de validation. Partant d'une spécification au niveau système, l'architecture sera raffinée jusqu'à la définition de l'architecture logicielle/matérielle de réalisation. Le processus de conception est donc un chemin progressif, à la fois pour déterminer une solution fonctionnellement appropriée et pour exprimer et évaluer les performances à chaque niveau de développement avec la prise en compte de l'aspect performances. Les étapes de conception sont:

- La spécification et la décomposition fonctionnelle : l'exploration concerne la recherche d'une conception, qui satisfasse aux besoins fonctionnels, mais aussi à des objectifs de performances. Souvent les performances considérées à ce niveau sont plutôt liées à des critères fonctionnels.
- Le partitionnement logiciel/matériel : l'exploration consiste à trouver une meilleure partition logicielle/matérielle en considérant un certain

nombre de contraintes à respecter et de performances à atteindre. On peut considérer la contrainte temps réel, le coût de réalisation, la consommation etc.

- La co-synthèse : Il s'agit d'assurer la synthèse des partitions logicielles et matérielles ainsi que la communication.
- Le prototypage virtuel et physique : au cours de cette phase, les solutions résultant des étapes de partitionnement et de co-synthèse sont validées. Il s'agit de vérifier le bon fonctionnement du système et la satisfaction des contraintes liées au contexte de l'application du système.

L'environnement Gaspard concerne l'utilisation du paradigme "data-parallel" et son application dans la conception de systèmes embarqués pour différents domaines d'application tel le traitement de signal intensif. Une attention particulière est accordée aux applications nécessitant l'utilisation le parallélisme régulier et aux architectures de systèmes multiprocesseurs sur puce (MPSoC).également régulière. Le placement de l'un sur l'autre nécessite également des outils de modélisation régulière (par exemple distribution par blocs des données en mémoires)

Dans le respect des recommandations faites par l'OMG en terme d'ingénierie dirigée par les modèles, Notre équipe a réalisé un environnement GASPARD pour la modélisation à haut niveau des applications et des plates-formes matérielles. L'outil offre d'un coté l'utilisation de techniques de vérification puissante avant la phase de prototypage et de l'autre coté la réalisation automatique de l'allocation et de l'ordonnancement des applications sur l'architecture.

Dans ce cadre, notre approche se traduit par la proposition d'un environnement construit sur le modèle du "Y-chart" (figure 1). Les technologies MDA (Model Driven Architecture) contribuent alors à exprimer les transformations de modèles entre les différents niveaux d'abstraction pour lesquels il existe un métamodèle associé. Dans le cas qui nous intéresse, ceci revient à réaliser des raffinements successifs entre les niveaux d'abstraction de la description depuis une spécification formelle jusqu'à un modèle synthétisable en FPGA. Ainsi des métamodèles ont été développés ou sont en cours de développement pour spécifier l'application, l'architecture et l'association entre l'application et l'architecture, chacun présente des possibilités d'exprimer l'aspect répétitif et régulier du système.

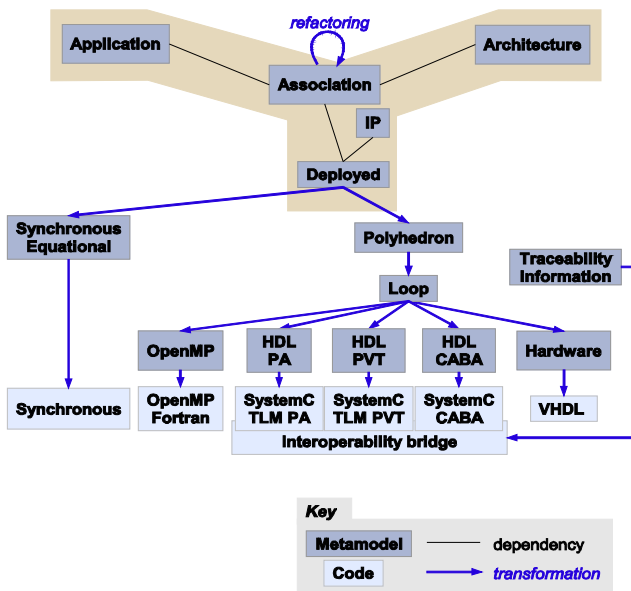


Figure 1. Métamodèles Gaspard

## 2.2 MDA

L'utilisation de l'approche MDA (Model driven Architectures) [4] pour la conception de SoC offre donc une nouvelle approche dans la conception de systèmes embarqués. Elle permet en particulier une séparation entre les modèles d'applications et/ou d'architectures cible d'un côté et les langages (ou plateformes) utilisés pour réaliser et implémenter ces modèles de l'autre côté. Elle ouvre les portes de la réutilisabilité de composants et de modèles. L'approche MDA est basée sur la définition de métamodèles et de règles de transformations entre les concepts introduits. On peut définir des profils UML comme langage de description des modèles de plus haut niveau d'abstraction pour l'application, l'architecture, l'association et le déploiement. Ce dernier fait appel à des métamodèles sur lesquels reposeront les modèles de description de l'architecture reconfigurable. Chaque modèle représente un certain niveau d'abstraction dans la description du système. Des transformations de modèle à modèle permettent de raffiner la description d'un niveau d'abstraction à un autre et permet aussi le déploiement du système sur différentes plateformes logicielles (SystemC, Java, VHDL, ...etc). Lors de ces transformations les bonnes propriétés du système doivent être conservées.

## 3 Application au transport

Les systèmes embarqués dans l'automobile en particulier un système de contrôle de vitesse réactif (cruise control) et un radar anti-collision nécessitent également d'outils et de techniques pour la réalisation de systèmes électroniques à bases de microprocesseurs fiables (Embarqués) à partir de systèmes de développement logiciel et de vérification

évolués. Ils devraient aussi contribuer à améliorer la maintenance des systèmes embarqués tout en garantissant un haut niveau de sécurité et ce malgré l'accroissement de la complexité de ces systèmes.

### 3.1 Le projet ModEasy

Le processus de conception des systèmes embarqués transforme des descriptions à un haut niveau d'abstraction (modèles de spécification) tels que les diagrammes de structure ou de comportement, vers une expression des détails d'implémentation à un bas niveau d'abstraction exprimés par des diagrammes de circuits microélectroniques (modèle de synthèse). Le but de ce projet Interreg III France/ Grande-Bretagne est de transformer la description à un haut niveau d'abstraction (Spécification du co-design) vers l'expression au niveau de détails de l'implémentation (Synthèse du co-design) pour diverses plateformes matérielles. L'objectif est de produire un outillage logiciel respectant ces principes et permettant alors le développement et la vérification de systèmes embarqués pour des composants liés à la sécurité automobile.

Les objectifs du projet sont:

- D'évaluer les besoins typiques des systèmes embarqués développés dans l'industrie automobile.
- De proposer des métamodèles, afin de formaliser la structure et le comportement des modèles de description à haut niveau. Ces modèles sont implantés dans un profil UML (Unified Modelling Language) [2]
- De définir les métamodèles pour le bas niveau lié à l'implémentation, pour SystemC et VHDL ainsi que les règles de transformations afin de produire ces modèles dépendant des plateformes.
- De définir les règles de transformation du processus de co-design depuis les modèles de haut niveau vers les modèles de bas niveau
- De concevoir pour chacun de ces modèles un outil d'évaluation de performances rapide et précis vérifiant la satisfaction des contraintes.
- De proposer une implantation des modèles dans une architecture reconfigurable de type FPGA (Field Programmable Gate Arrays).
- D'évaluer l'environnement pour des applications pré-industrielles telles que le contrôle de vitesse guidé par GPS et les radars anti-collision.
- De réaliser un prototype de ces systèmes en FPGA, et examiner la faisabilité d'une réalisation sur un System on Chip plus performant que les systèmes reconfigurables.

Nous décrivons dans cette partie les résultats obtenus lors de la validation par prototypage d'applications réelles et en particulier le contrôleur de vitesse couplé au GPS et au radar anticollision. Les étapes de réalisation recouvrent :

- modélisation de l'application et de l'architecture d'un SoC et du placement avec un corrélateur digital multibit, monobit et estimateurs d'ordre supérieur,
- application des transformations dans le profil UML,
- vérifications et génération par transformation de modèles du niveau RTL (Registre Transfer Level),
- réalisation du système embarqué en FPGA
- évaluation du système sur route dans un véhicule de test .

Ces différentes étapes sont illustrées dans la partie suivante. Elles ont été réalisées avec la collaboration de l'équipe valencienne de l'IEMN[3].

### 3.2 Du model UML à la route

Le type d'application que l'on considère regroupe les trois fonctionnalités radar anti-collision, cruise control et GPS afin d'améliorer la sécurité du système embarqué. Pour cela il faut pouvoir prendre en compte en même temps la partie contrôle du cruise control et la partie data intensif du radar. (Figure 2) Dans l'état actuel de Gaspard seule la partie data flow permet de générer du code VHDL, la suite de la démonstration ne concerne donc que cette partie.

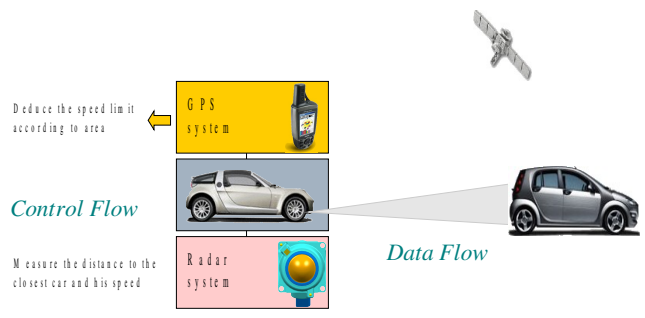


Figure 2. Data flow et control flow

Un algorithme de corrélation permettant la détection d'obstacle est développé par l'IEMN[3]. Cet algorithme peut alors être modélisé en UML par le biais de notre profil Gaspard. Dans le cas d'une mise en oeuvre sur FPGA, la modélisation du matériel est réduite à sa plus simple expression. Seulement lors du déploiement, on pourra spécifier les caractéristiques de celui-ci afin d'optimiser le placement/routage. (Figure 3)

Dans l'environnement Eclipse (Figure4) et par 4 transformations successives, nous produisons le code VHDL synthétisable (Figure 5) qui pourra ensuite être utilisé par les outils commerciaux fournis avec le FPGA. (Figure 6)

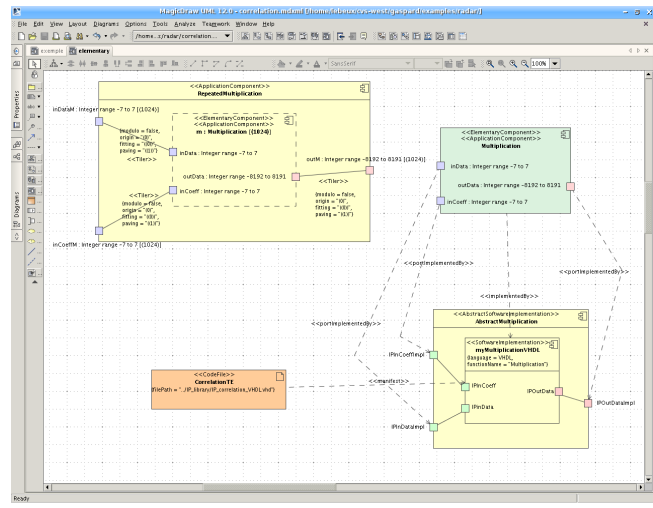


Figure 3. Modélisation en UML de la corrélation. Déploiement du composant élémentaire d'addition sur le code VHDL correspondant, le reste du code sera produit de façon automatique

```

genit0 : for it0 in 1 to 4 generate
  genit1 : for it1 in 1 to 4 generate
    instanceOfMyComponent : MyComponent
    port map(
      clk => clkConnectorlignecolonne,
      raz => razConnectorlignecolonne,
      InAMyComponent
        =>RepetitionConnector_G999b96(it0)(it1),
      OutMyComponent
        =>RepetitionConnector_14fe736(it0)(it1),
      InBMyComponent
        =>RepetitionConnector_1d1e713(it0)(it1));
    end generate;
  end generate;
end generate;

```

Figure 4. Les 4 modèles dans l'ordre : UML, Gaspard, HW (hardware) et VHDL sont produits automatiquement par transformation de modèles

```

genit0 : for it0 in 1 to 4 generate
  genit1 : for it1 in 1 to 4 generate
    instanceOfMyComponent : MyComponent
    port map(
      clk => clkConnectorlignecolonne,
      raz => razConnectorlignecolonne,
      InAMyComponent
        =>RepetitionConnector_G999b96(it0)(it1),
      OutMyComponent
        =>RepetitionConnector_14fe736(it0)(it1),
      InBMyComponent
        =>RepetitionConnector_1d1e713(it0)(it1));
    end generate;
  end generate;
end generate;

```

Figure 5 Exemple de code VHDL produit

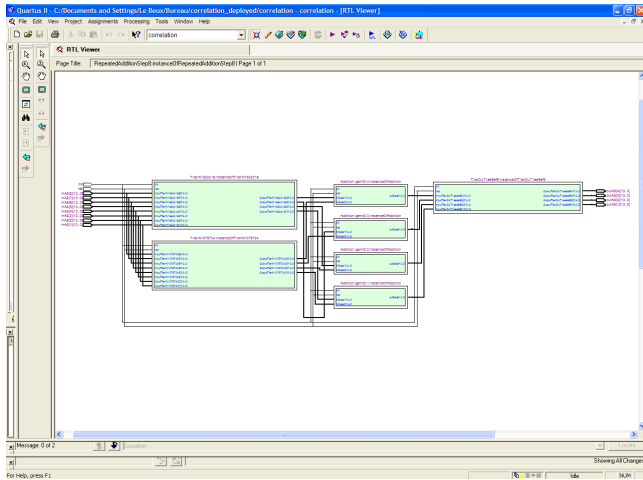


Figure 6. Synthèse produite pour un Altera

Enfin le FPGA a été placé sur une carte reliée à un détecteur radar et couplée à un PC pour la simulation GPS Cruise control. Des essais sur route ont pu être réalisés en collaboration avec l'INRETS de Lille. (Figure 7)



Figure 7. Essai sur route

## 4 Conclusions

Dans ce papier nous avons montré qu'il était possible de tirer partie des nouvelles technologies logicielles dans le contexte particulier des systèmes embarqués. Depuis une modélisation fonctionnelle en UML et par transformations successives de modèles nous sommes capables de produire du code VHDL synthétisable et synthétisé sur un circuit reprogrammable FPGA.

Dans la chaîne de co-design d'autres plateformes peuvent être nécessaires et Gaspard permet également d'obtenir du code SystemC pour la simulation conjointe logiciel/matériel

ou encore du code Lustre[5] pour permettre la vérification au plus tôt.

Nous pensons que la modélisation d'architectures massivement parallèles particulièrement régulières en parfaite adéquation avant ce type d'algorithme de corrélation devrait permettre une meilleure utilisation du FPGA et même de permettre d'en réaliser un système plus souple également synthétisable. Des travaux sont en cours sur la définition d'une architecture de type SIMD dédiée aux algorithmes de traitement de signal au sein de notre équipe.

Enfin, une expérimentation complète de la chaîne de transformation nous a permis, en collaboration avec l'IEMN de Valenciennes et l'INRETS de Lille, de tester sur route le résultat de ces travaux, qui inclue entre autre la conception d'un radar anti-collision embarqué dans un FPGA. Ainsi, nous pensons que le secteur du transport, et en particulier l'automobile, peut bénéficier de notre environnement de conception.

## References

- [1] Rabie Benatitallah, Smail Niar, Alain Greiner, Smail Meftali, Jean-Luc Dekeyser. Estimating energy consumption for an MPSoC architectural exploration. In ARCS'06: Architecture of Computing Systems, Frankfurt, Germany, March 2006
- [2] Arnaud Cuccuru, Jean-Luc Dekeyser, Philippe Marquet, and Pierre Boulet. Towards UML 2 extensions for compact modeling of regular complex topologies - A partial answer to the MARTE RFP. In MoDELS/UML 2005, ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems, pages 445-459, Montego Bay, Jamaica, October 2005. Lecture Notes in Computer Science vol. 3713.
- [3] Laila Sakkila, P. Deloof, Y. ElHillali, A. Rivenq, S. Niar «A real time signal processing for an anticollision road radar system.» IEEE VTC (Vehicular Technology) 2006 Fall 25 – 28 September 2006, Montréal, Canada
- [4] D. Schmidt. Model-Driven Engineering, in: IEEE Computer, February 2006, vol. 39, n° 2, p. 41-47.
- [5] P. Caspi, D. Pilaud, N. Halbwachs, J.A. Plaice. Lustre: a declarative language for real-time programming, in: Proceedings of the 14th ACM SIGACT-SIGPLAN symposium on Principles of programming languages (POPL'87), ACM Press, 1987, p. 178-188.