Processus MOPCOM pour SoC/SoPC

Ali Koudri, Joël Champeau, Denis Aulagnier, Didier Vojtisek

Processus de développement UML/MARTE MoPCoM pour le codesign

Ali Koudri
ali.koudri@ensieta.fr

Joël Champeau
joel.champeau@ensieta.fr

Denis Aulagnier
denis.aulagnier@fr.thalesgroup.com

Didier Vojtisek
dvojtisek@inria.fr

Résumé

Dans cet article, nous présentons un flôt MDE de conception de SoC (System-On-Chip) basé sur l'utilisation du profile UML MARTE (Modeling and Analysis of Real-Time Embedded Systems), dédié à la conception et l'analyse de système temps réel embarqués et récemment standardisé par l'OMG. Le flôt présenté adresse les problèmatiques de la communauté de l'ESL (Electronic System Level) et s'inscrit dans le cadre du projet MoPCoM (http ://www.mopcom.fr).

1 Introduction

Le marché des System-On-Chip connait depuis quelques années une forte évolution. L'ITRS [ITR07] estime que d'ici à 2012, ce marché pésera 56 millards de dollars, ce qui représente une croissance annuelle moyenne de 24%. Ce succés est dû aux extraordinaires capacités d'intégration atteintes (22 nm), conformément à la loi de Moore. Ainsi, il est possible aujourd'hui d'intégrer sur une puce des systèmes complets, constitués de processeurs, de mémoires ou de senseurs.

Malheureusement, les flôts de développements et les outils de conception n'ont pas été aussi prompts à suivre de telles évolutions. Il en résulte aujourd'hui un gap de productivité accentué par les lois économiques. Les défis à relever consistent alors pour l'essentiel à diminuer les temps de developpement et les coûts de conception, de validation et de production. Nous pensons que l'ingénierie des modèles peut permettre d'atteindre ces objectifs à condition d'intégrer les préoccupations de la communauté de l'ESL, à savoir l'exploration d'architecture rapide, la réutilisation d'IP (Intellectual Properties) ou la synthèse de haut niveau. Nous verrons dans cet article que la prise en compte de telles préoccupations nécessite de revoir l'approche en Y du MDA standardisé par l'OMG.

Dans la section suivante, nous présentons l'état de l'art sur les approches MDE appliqués au codesign. Nous présentons ensuite notre flôt de conception MDA qui est basé sur l'utilisation du profile UML MARTE, dédié à la conception et l'analyse de système temps réél embarqués. Nous illustrerons notre approche sur un exemple de dévéloppement d'un système de radio intelligente implémenté sur une plateforme FPGA.

2 État de l'art

La transformation des exigences du client en une implémentation faisant de bons compromis entre les coûts et les performances est un processus complexe constitué de différentes activités de conception, d'analyse et de validation, et à différents niveaux d'abstraction. La bonne conduite de ces activités nécessite l'emploi de langages et d'outils appropriés. Depuis quelques années, la communauté de l'ESL propose des solutions pour faire face aux contraintes économiques et technologiques. Par exemple, l'explora-

tion d'architecture permet de réduire l'espace des implémentations possibles en fournissant au client des spécifications de système exécutables ou vérifiables à différents niveaux d'abstraction. Cette technique itérative permettant de réduire les risques est généralement couplée à d'autres techniques pour accélérer les développements. Parmi ces techniques, on peut citer :

- La synthèse de haut niveau [SVN06] qui permet de transformer des spécifications fonctionnelles en architectures optimisées,
- La réutilisation d'IP qui consiste à construire des systèmes par assemblage d'IPs vérifiés et documentés, et ce à différents niveaux d'abstraction et en supposant que les IPs aient des interfaces compatibles[KMD05],
- L'approche "Platform Based Design" [SVCBS04] qui consiste à configurer pour des applications spécifiques, des plateformes génériques, comprenants des composants reprogrammables (processeurs, FPGA, etc).

La modèlisation de SETR nécessite l'emploi d'un ou de plusieurs langages dédiés qui réifient les concepts permettant de représenter les caractéristiques les plus pertinentes pour un niveau d'abstraction donné. Par exemple, la notion de concurrence est essentielle dans la représentation des SETR. Dans [GT03], les auteurs soulignent quelles sont les concepts d'UML qui permettent d'exprimer la concurrence (classes actives, AND-State, Fork, etc). On constate dans la pratique que les plateformes évoluent plus rapidement que les fonctions qu'elles implémentent. Aussi, dans les développements de systèmes d'information, beaucoup de temps est consacré à traiter les obsolescences. Dans ce cadre, l'approche MDA [OMG03] suggèrent de séparer au mieux les préoccupations et de prévoir les mécanismes permettant leur tissage. Dans cette approche, le modèle fonctionnel est nommé CIM (Computational Independent Model), le modèle d'architecture fonctionnelle est nommé PIM (platform Independent Model), le modèle de plateforme est nommé PDM (plateform Deployment Model) et le modèle alloué, représentant l'allocation de l'architecture fonctionelle sur la plateforme PSM (Platform Specific Model). Ainsi, les fonctionnalités et les plateformes peuvent évoluer de manière indépendante. La modélisation de plateformes présentée dans l'approche MDA nécessite l'emploi de formalismes dédiés. Dans [EG03], on peut trouver un exemple intéressant de méthodologie MDA dans lequel les auteurs utilise UML pour modéliser l'architecture fonctionnelle et introduisent de nouveaux stéréotypes pour modèliser la plateforme. Mais la séparation fonction / plateforme constitue un cas simple d'orthogonalisation des préoccupations. Elle est raffinée dans [CSL+] où les auteurs soulignent la nécessité d'orthogonaliser différents aspects : calcul et communication, fonction et architecture, comportement et performance. Dans cet article, les auteurs présentent une méthodologie qui adapte l'approche "Platform Based Design" proposée par la communauté ESL à UML. Dans [RSRB07], les auteurs présentent une méthodologie nommée UPES (Unified Process for Embedded Systems) basé sur l'utilisation du profile UML for SystemC. Ce profile permet de réutiliser les efforts des partenaires du consortium OSCI autour du langage SystemC, dérivé du C++ et qui permet de décrire des architectures matérielles à différents niveaux d'abstractions (TLM PV, TLM PVT, TLM CC). Enfin, la méthodologie Gaspard [PAM+08] adresse des applications de traitement intensif du signal implémentées sur des grilles de processeurs, représentant un type particulier de plateforme qui comprend des structures répetitives.

Dans la section suivante, nous présentons un aperçu général de notre méthodologie et de l'application de Radio Intelligente qui nous a permis de valider notre approche.

3 Méthodologie MoPCoM

La méthodologie MoPCoM peut être considérée comme un raffinement de l'approche MDA intégrant l'exploration d'architecture, l'approche "Platform Based Design" et la réutilisation d'IPs. Elle prend en entrée une architecture fonctionnelle de type PIM exprimée en SysML [OMG08]. La figure 1 donne un aperçu du flôt MoPCoM et met en exergue 3 niveaux de modélisation :

- Le niveau *Abstract Modeling Level* (AML) adresse l'expression de la concurrence et de la communication sans présumer de la limitation

des ressources,
- Le niveau *Execution Modeling Level* (EML) définit une topologie physique abstraite permettant de faire des analyses à gros grains,
- Le niveau *Detailed Modeling Level* (DML) décrit la plateforme de manière détaillée et permet une analyse fine menant à l'implémentation du système.

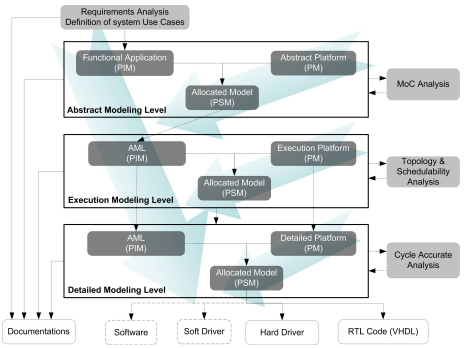

FIG. 1 – MoPCoM Process Overview

Pour chaque niveau, nous identifions quelles sont les sous-ensembles nécessaires et suffisant d'UML et de MARTE pour réaliser les activités de conception, d'analyse et de validation pour chaque niveau d'abstraction. Les régles méthodologiques qui en résultent sont saisies sous forme de contraintes KerMeta [MFJ05] dans un modèle de processus, exprimé dans un langage dérivé de SPEM (System and Software Process Engineering Modeling) et développé dans le cadre du projet MoPCoM.

Afin de valider notre approche, nous avons développé un système de radio intelligente [MJ01, HPM07] qui adapte son comportement suivant les caractéristiques de son environnement éléctromagnétique (localisation, bande passante, etc). Dans cet article, nous nous focaliserons sur l'étude du cas d'utilisation "Locate Radio Access Technology Source" qui consiste à selectionner une bande suivant sa charge d'utilisation et à configurer le système afin qu'il communique suivant le protocole correspondant. Le système est implémenté sur une carte FPGA Xilinx ML-506 qui comprend des composants analogiques et numériques.

3.1 Capture des exigences et analyse fonctionnelle

Dans la mesure où la capture, la formalisation et la traduction des exigences en architecture fonctionnelle dans le monde UML/SysML sont trés largement adressées par la littérature, nous ne nous attarderons pas sur ces points. Briévement, nous utilisons l'approche "Use Cases" pour formaliser les exigences puis l'analyse fonctionnelle permet de spéficier une architecture fonctionnelle flexible et réutilisable, qui applique l'ensemble des bonnes pratiques (design patterns) pour la résolution de problèmes spécifiques. Par exemple, l'expression de la reconfiguration dynamique du système au niveau fonctionnel utilise le pattern "Strategy". À noter toutefois qu'à ce niveau, nous utilisons certains sous-paquetages du profile MARTE, en particulier pour l'expression des contraintes non-fonctionnelles.

Par exemple, l'expression des contraintes temps réél utilise le paquetage "Time" de MARTE car il fournit un modèle de temps plus fin que celui de UML. Le profile MARTE fournit également un langage de spécification de valeurs nommé VSL (Value Specification Language) sous forme d'un métamodèle et d'une grammaire. Ce langage extensible permet d'annoter les modèles par des contraintes portant sur des grandeurs physiques (temps, puissances, longueurs, poids, etc).

Le modèle PIM issu de l'analyse fonctionnelle peut être utilisé en entrée d'un outil de synthèse comportementale de haut niveau moyennant une transformation de type "model-to-text". Le projet MoPCoM fournit de telles transformations pour les outils CatapultC de Mentor et GAUT développé par le LESTER (`http://web.univ-ubs.fr/lester/www-gaut/`).

3.2 Abstract Modeling Level

Alors que l'analyse fonctionnelle se focalise sur les aspects algorithmiques du système, ce niveau adresse l'explicitation de la concurrence et de la communication, généralement désignée par le terme "Modèle de Calcul" (MoC – Model of Computation).

Le modèle de calcul définit les régles qui régissent la concurrence et la communication. Il est caractérisé

par au moins un modèle de temps (causal, continu, discret), une représentation des données (type de données abstraits, vecteurs de bit, signal) ou un type de communication (IPC – Inter-Process Communication, variables partagées, FIFO, signaux).

La définition du MoC permet de mener différents types d'analyse comme par exemple des analyses de performance et d'ordonnancement ou la detection d'inter-blocages. Le choix du MoC dépend d'une part du système à implementer et d'autre part du type d'analyse à réaliser. Dans la mesure où les systèmes mélent généralement plusieurs modèles de calcul, il est important de pouvoir disposer d'outils permettant la conception et l'analyse de systèmes multi-Mocs. L'outil Ptolemy développé par l'université de Berkeley en est un exemple.

Pour exprimer la concurrence dans UML, on peut au niveau comportemental utiliser les And-States dans les statecharts ou les "fork nodes" dans les diagrammes d'activités. Au niveau de la classe, le méta-attribut "isActive" supporte la notion d'entité concurrente. Il est également possible d'utiliser le pattern "Concurrence" dans l'architecture fonctionnelle.

Cependant, toutes ces techniques pour exprimer la concurrence ne permettent pas de saisir les propriétés temps réél requis pour mener des analyses pertinentes. À cet effet, la notion de "RTUnit" (Real-Time Unit – Unité temps réél), fournit par le profile MARTE dans le paquetage "HLAM", supporte la notion d'entité concurrente. Elle est associée à un ensemble de méta-attributs (tags) précisant les caractéristiques temps réél. Un "RTUnit" est défini comme un bloc qui fournit/requiert un ensemble de services temps réél (RTService) et posséde au moins un comportement temps réél (RTBehavior) avec une queue bornée / non bornée. Les comportements temps réél peuvent être décomposés en actions temps réél (RTAction).

À ce niveau, toutes les communications sont en point-à-point et sont supportées par des connecteurs temps réél (RTeConnector) implementant des mécanismes de haut niveau (put/get ou read/write).

La figure 2 illustre l'allocation d'une architecture fonctionnelle ① sur la plateforme AML ②. Cette plateforme mélange 3 types de MoCs : CT (Continuous Time), KPN (Kahn Process Network) et DE (Discrete Event). Les comportement spécifiés au niveau fonctionnel sont transformés, lors de l'allocation, en comportement temps réél ③ pour lesquels un certain nombre de caractéristiques doivent obligatoirement être renseignés. Ces comportements doivent alors tenir compte des mécanismes de communication sous-jacents.

Le support des MoCs est rendu possible grâce à l'instanciation de classes spécifiques regroupés dans une librairie MoC fournit par le projet MoPCoM. Dans l'exemple de la figure 3, nous montrons comment le MoC KPN est supporté : pour chaque intance de "KPNConnector" ou "KPNPort", une classe de la librairie MoC est instanciée.

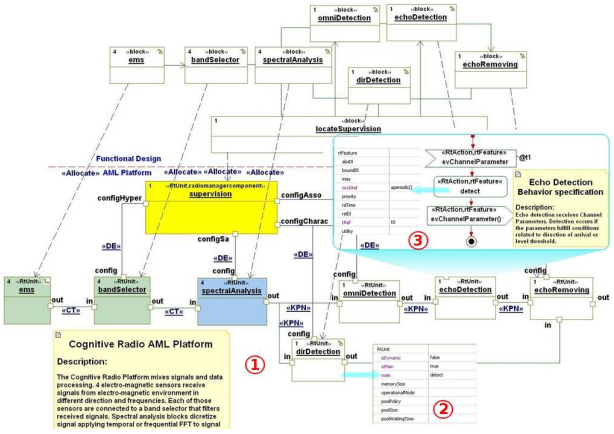

Fig. 2 – Functional to AML Platform Allocation

Par ailleurs, les contraintes d'allocation peuvent faire apparaître dans le modèle alloué des blocs spéciaux dédiés aux multiplexages / démultiplexages de données.

3.3 Execution Modeling Level

Le but du niveau EML est d'analyser l'exécution du modèle AML alloué du niveau précédent sur la définition d'une plateforme d'exécution gros grain. Les analyses menées à ce niveau abstraient les coûts liés au matériel (CPU, FPGA) ou aux couches d'abstrac-

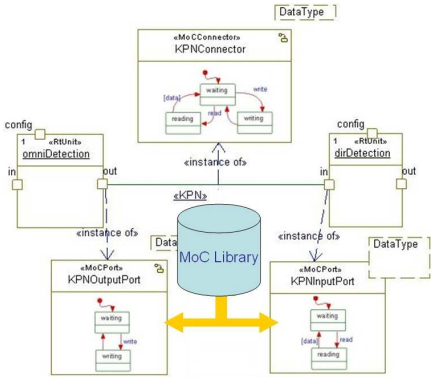

FIG. 3 – KPN MoC Support

tion (OS, Middleware) à travers des modèles statistiques [Kou96].

La définition de la plateforme et de l'allocation est réalisée au regard des compromis entre les performances et les coûts, et à travers un processus itératif mettant en avant la topologie de la plateforme. La topologie regroupe les noeuds de calcul ou de mémorisation, interconnectés à travers des bus qui implémentent des protocoles de haut niveau. Les données transitées sont exprimées au bit prés et le modèle de temps est approximatif.

Les ressources modélisées au niveau EML sont avant tout caractérisées par les services qu'elles offrent aux applications et les qualités de services associées. Les ressource possédent des vitesses d'exécution relatives. Les contraintes temps réél sont vérifiées grâce à l'utilisation d'une horloge idéale unique (pattern singleton) mesurant la progression du temps réél.

Les résultats des analyses conséquents aux allocations peuvent entrainer des refactorings au niveau des modèles de plateforme comme au niveau du PIM. Actuellement, ce processus de refactoring est réalisé manuellement et dépend essentiellement des compétences des ingénieurs qui en ont la charge. Cependant, nous pensons qu'il sera possible de prévoir par la suite des mécanismes de rétro-annotation permettant d'automatiser l'exploration d'architecture.

Afin de modéliser la plateforme du niveau EML, nous utilisons le paquetage GRM (Generic Resource Modeling) du profile MARTE. Ce paquetage réifie les concepts nécessaires pour modéliser des entités logiques ou physiques persistentes. Dans ce paquetage, une ressource est caractérisée par l'ensemble de ses services rendus aux applications et par ses propriétés non-fonctionnelles telles que la latence, le poids ou la consommation. Afin de procéder à des analyses gros grains de l'utilisation des ressources, nous utilisons également le paquetage GQAM (Generic Quantitative Analysis Modeling) et ses sous-paquetages PAM (Performance Analysis Modeling) et SAM (Scheduling Analysis Modeling). Ces analyses permettent notamment de détecter les goulots d'étranglement au niveau de bus de communication.

La figure 4 montre une vue simplifiée de l'allocation du modèle alloué AML① sur la plateforme EML ②. Cet exemple illustre une allocation sur deux niveaux de plateforme : le premier niveau spécifie la couche d'abstraction matériel (Operating System) alors que le second niveau représente la plateforme physique. L'allocation est représentée par des liens de dépendence stéréotypés "Allocate" spécifiant la dimension temporelle, spatiale ou hybride de l'allocation. Dans cette figure, le noeud "Supervision" représente un noeud de calcul en charge de la reconfiguration dynamique et caractérisé par une vitesse relative égale à 1.0 ④. La couche d'abstraction est constituée d'un ordonnanceur gérant des tâches avec une politique de type EDF (Earliest Deadline First) ③. Le bus de communication implémente des primitives de transports possédants un certain nombre de caractéristiques non-fonctionnelles ⑤. Les scénarios d'analyse mettent en exergue les débits de données, les latences, la mémoire occupée ou les coûts liés à la reconfiguration dynamique.

3.4 Detailed Modeling Level

Ce niveau vise à fournir un modèle détaillé de la plateforme par un raffinement des modèles du niveau précédent. Le modèle alloué de ce niveau doit contenir assez d'information pour permettre la génération du code d'implémentation pour les parties matérielles (VHDL) ou logicielles (C), ainsi que la glue logique (pilotes) permettant l'interfaçage des deux parties.

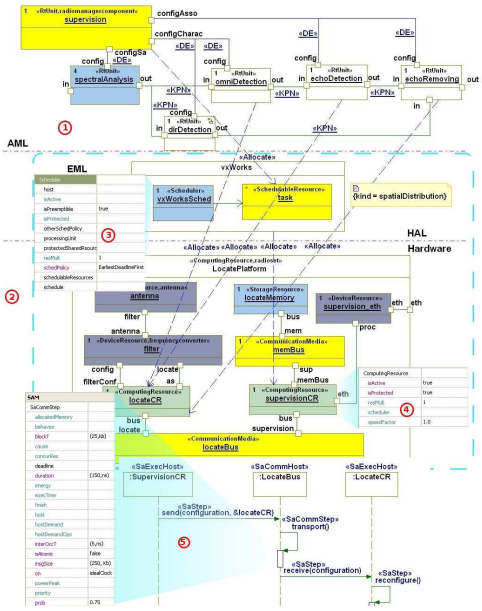

Fig. 4 – AML to EML Platform Allocation

Par exemple, les noeuds de calcul définis au niveau EML sont raffiné en processeurs ou en FPGA, les noeuds de mémorisation sont raffinés en SRAM ou en Flash, etc. On s'intéresse à ce niveau autant aux services fournis par les ressources qu'aux caractéristiques physiques précis (fréquence de fonctionnement, poids, prix, etc). Le raffinement de la plateforme implique celui des interfaces, des protocoles de communication ainsi que celui du modèle de temps sous-jacent. À ce niveau, les ressources sont "clockés" et les analyses sont menées au bit prés et au cycle prés (Cycle Accurate Bit Accurate). Les couches d'abstraction matérielle deviennent des OS ou des middlwares, et sont caractérisées par leurs API. La norme POSIX (http ://standards.ieee.org/regauth/posix/) constitue un exemple d'API.

Afin de modéliser la plateforme et les analyses à ce niveau, nous utilisons de MARTE les paquetages DRM (Detailed Resource Modeling) et ses sous-paquetage SRM (Software Resource Modeling) et HRM (hardware Resource Modeling), conjointement avec les paquetages GQAM comme vu dans la section précédente.

4 Gestion des métamodèles dans le processus

Le processus MoPCoM est constitué de trois niveaux d'abstraction avec pour chaque niveau plusieurs modèles : un modèle d'application, un modèle de plateforme, un modèle d'allocation, un modèle alloué et plusieurs modèles d'analyse (performance, ordonnancement, WCET, etc). Approximativement, le métamodèle UML comprend environ 240 métaclasses auquelles il faut ajouter 150 stéréotypes du profile MARTE. Un des objectifs du projet MoPCoM est de fournir des régles méthodologiques permettant de faciliter les activités de conception et d'analyse. Dans ce but, nous proposons une extension du profile SPEM appelé MODAL (MOdel Driven Architecture Language) et qui introduit d'une part des notions permettant de mieux comprendre et analyser les méthodologies et les processus de développement ; et d'autre part, qui raffine la définition des artefacts au regard des métamodèles qui les définissent et leur associe la notion de cycle de vie dans le but de contrôler plus finement l'exécution des processus. Ainsi, chaque activité du processus est liée à une intention (but) et applique une stratégie (plan) conformément à l'état de l'art du moment des outils et des langages. Dans ce langage, les contraintes portant sur l'utilisation de métaclasses ou de stéréotypes pour une activité donnée sont spécifiés en utilisant la syntaxe du langage KerMeta [MFJ05]. En outre, le langage KerMeta fournit des mécanismes permettant de tisser facilement des contraintes et une sémantique opérationnelle aux métamodèles.

5 Outillage

Le processus de codesign MoPCoM est supporté par un ensemble d'outils "conformes" aux standards de l'OMG (MDA, UML, MOF, XMI) ou aux technologies fournies par la plateforme Eclipse (EMF, Ecore, KerMeta, MDWorkbench) : Rhapsody de Telelogic est utilisé pour la modélisation UML de l'ap-

plication et des plateformes, MDWorkbench de Sodius est utilisé pour les transformations (model-to-model et model-to-text) et KerMeta est utilisé comme outil de méta-modélisation pour la spécification des contraintes d'utilisation du profile MARTE conformément aux niveaux d'abstraction AML, EML et DML.

Le langage d'action permettant de spécifier le corps des opérations ainsi que les guardes ou les actions dans les spécifications comportementales est un sous-ensemble du langage C++ complété par un ensemble de macros adressant par exemple l'envoi d'événements. Ce choix est essentiellement motivé par la flexibilité et l'extensibilité du langage ainsi que sa facilité d'apprentissage. Le projet MoPCoM fournit un parseur permettant de remonter la syntaxe du langage au niveau des modèles à des fins d'analyse et de génération de code de plus bas niveau (VHDL ou C).

6 Conclusion

Dans cet article, nous avons présenté un flôt de développement pour la conception de SoC. Ce flôt raffine le modèle en Y du MDA standardisé par l'OMG pour permettre l'exploration d'architecture. De plus, ce flôt a été formalisé par le biais d'un modèle décrit par un langage dérivé de SPEM. Un exemple d'application de Radio Intelligente implémenté sur une plateforme FPGA Xilinx nous a permis de valider notre approche. Ce travail a été réalisé dans le cadre du projet MoPCoM (http ://www.mopcom.fr), labelisé par l'Agence National de la Recherche et les pôles "Image et Réseaux" et "Cluster de clusters" des régions Bretagne et Pays de la Loire.

Références

[CSL+] Rong Chen, Marco Sgroi, Luciano Lavagno, Grant Martin, Alberto Sangiovanni-Vincentelli, and Jan Rabaey. Uml and platform-based design.

[EG03] Martyn Edwards and Peter Green. Uml for hardware and software object modeling. pages 127–147, 2003.

[GT03] Sebastien Gerard and Francois Terrier. Uml for real-time : which native concepts to use ? *ACM*, 13 :17–51, 2003.

[HPM07] Rachid Hachemani, Jacques Palicot, and Christophe Moy. A new standard recognition sensor for cognitive radio terminals. In *EURASIP*, KESSARIANI, GREECE, 2007.

[ITR07] ITRS. Design. Technical report, INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS, 2007.

[KMD05] Ali Koudri, Samy Meftali, and Jean-Luc Dekeyser. IP integration in embedded systems modeling. In *14th IP Based SoC Design Conference (IP-SoC 2005)*, Grenoble, France, December 2005.

[Kou96] A. A. Kountouris. Safe and efficient elimination of infeasible execution paths in wcet estimation. In *RTCSA '96 : Proceedings of the Third International Workshop on Real-Time Computing Systems Application (RTCSA '96)*, page 187, Washington, DC, USA, 1996. IEEE Computer Society.

[MFJ05] Pierre-Alain Muller, Franck Fleurey, and Jean-Marc Jézéquel. Weaving executability into object-oriented meta-languages. In *Proc. of MODELS/UML*, LNCS, Montego Bay, Jamaica, 2005. Springer.

[MJ01] III Mitola Joseph. Cognitive radio for flexible mobile multimedia communications. *Mob. Netw. Appl.*, 6(5) :435–441, 2001.

[OMG03] OMG. Mda guide version 1.0.1. Technical report, Object Management Group, 2003.

[OMG08] OMG. Systems modeling language specification v1.1. Technical Report ptc/2008-05-16, Object Management Group, 2008.

[PAM+08] Eric Piel, Rabie Ben Attitalah, Philippe Marquet, Samy Meftali, Smaïl Niar, Anne Etien, Jean-Luc Dekeyser, and Pierre Boulet. Gaspard2 : from marte to systemc simulation. March 2008.

[RSRB07] E Riccobene, P Scandurra, A Rosti, and S Bocchio. Designing a unified process for embedded systems. In *In the Fourth International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MOMPES)*, Braga, PORTUGAL, March 2007. IEEE Computer Society.

[SVCBS04] Alberto Sangiovanni-Vincentelli, Luca Carloni, Fernando De Bernardinis, and Marco Sgroi. Benefits and challenges for platform-based design. In *DAC '04 : Proceedings of the 41st annual conference on Design automation*, pages 409–414, New York, NY, USA, 2004. ACM.

[SVN06] Greg Stitt, Frank Vahid, and Walid Najjar. A code refinement methodology for performance-improved synthesis from c. In *ICCAD '06 : Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, pages 716–723, New York, NY, USA, 2006. ACM.