SpearDE : Plateforme de développement de chaînes de parallélisation sur architectures distribuées hétérogènes

Teodora PETRISOR, Eric LENORMAND, Rémi BARRERE, Michel BARRETEAU

THALES Research and Technology, Laboratoire de Systèmes Embarqués
Campus Polytechnique, 1 Av. Augustin Fresnel, 91767 Palaiseau Cedex, France
{claudia-teodora.petrisor, eric.lenormand, remi.barrere,
michel.barreteau}@thalesgroup.com

Résumé – Avec l'émergence des nouvelles architectures de calcul parallèle, de nouvelles méthodes qui ne font pas partie du savoir-faire logiciel traditionnel doivent être mises en œuvre. Par conséquent, leur introduction dans un monde industriel fortement orienté vers des gains de productivité et réduction de coûts, tarde à se produire malgré les avantages que ces architectures peuvent amener. L'offre d'outils de conception permettant de faire le pont entre une description fonctionnelle de l'application et son implémentation sur une architecture distribuée reste très limitée, en général spécialisée soit sur un type d'applications, soit sur une architecture parallèle particulière. Dans la perspective industrielle de contrôle des coûts de développement et des expertises requises, le besoin d'adresser, selon les applications, des cibles de calcul différentes en général hétérogènes a poussé à développer une plateforme logicielle adaptable aux architectures candidates et capable d'outiller le processus d'implémentation à différents niveaux dans le flot de conception.

Abstract – With the emergent manycore architectures new programming skills need to be acquired and this slows down the process of switching to parallel hardware in the industry despite its advantages. This is particularly challenging for critical mission systems where the impact of development costs is not only very high at the design stage but also during the long life cycle of the products. To our knowledge, in most cases tools target either a specific application type or, more often, a specific parallel architecture (such as the Cell/BE, the GPU or one custom MPSoC). The need for productivity gains and for small development costs while preserving portability has lead to the development of a software platform tooling the implementation process from a high-level application description down to performance evaluation and code generation. This environment is designed to be adaptable to different candidate parallel architectures without topology assumptions.

1 Contexte

La programmation d'architectures de calcul parallèle demande de nouvelles méthodes qui ne font pas partie du savoir-faire logiciel traditionnel, et elle tarde à s'installer dans le monde industriel. Le challenge est d'autant plus grand dans un contexte industriel adressant les systèmes de mission critiques, où l'impact des coûts de développement est très important à la fois à la conception et lorsque des produits à long cycle de vie connaissent des changements de spécifications et/ou d'architectures cible. Dans ce contexte, le besoin d'outils de conception de haut niveau (« System Level Design ») est particulièrement patent. Bien que la demande de l'industrie soit claire, l'offre d'outils de conception [5], [6], [11] permettant de faire le pont entre une description fonctionnelle de l'application et son implémentation sur une architecture distribuée reste très limitée. En plus de la complexité du problème posé et de l'absence pour l'instant de langage approprié standard, la taille du marché potentiel pour ces outils reste incertaine, réduite de plus par leur adhérence à une famille d'architectures cible particulières, malgré les efforts substantiels de création de plateformes de parallélisation en cours dans la communauté (par exemple avec [4], [10], [9], [12]). Dans la perspective industrielle de contrôle des coûts de développement et des expertises requises, le besoin d'adresser, selon les applications, des cibles de calcul différentes - à plus forte raison hétérogènes, par exemple lorsqu'elles associent FPGAs et processeurs - a poussé à développer une plateforme logicielle adaptable aux architectures candidates et en même temps capable d'outiller le processus d'implémentation sur une architecture distribuée à différents niveaux. Ce processus inclut le placement sur l'architecture depuis une description des applications à haut niveau, l'évaluation de performances et la génération de code. Cette plateforme a été bâtie sur un certain nombre de choix, issus de l'expérience acquise à Thales sur les architectures parallèles et leur programmation :

- **sur l'adaptation aux architectures cible** : la plateforme offre un ensemble de modèles de base utilisables pour représenter les architectures cible, éventuellement hétérogènes, à la fois sur l'aspect topologique des unités de calcul, des mémoires et des moyens de communications, et sur l'aspect comportemental en temps réel.
- **sur le format de capture des applications** : on préfère aux langages C/C++ ou Matlab, souvent proposés par les outils [9], [7], en l'absence d'un meilleur standard plus approprié, une approche basée sur des modèles de com-

posants spécialisés par domaine d'application (e.g. radar). Ce modèle offre de meilleures conditions pour utiliser les outils de parallélisation qui suivent, par exemple en traduisant les modèles domaine en combinaisons de modèles de calcul (angl. « Models of Computation ») porteurs d'informations pertinentes pour la parallélisation et l'ordonnancement

- **sur le domaine d'applications** : on porte une attention particulière aux graphes composés de nids de boucles (« static affine nests of loops »), car le parallélisme de données / boucles est le vecteur le plus important de gain de performance sur les architectures massivement parallèles. La plateforme offre des fonctionnalités génériques d'allocation, de parallélisation et d'ordonnancement relevant de ce domaine. On y trouve en particulier les opérations qui permettent de découper et d'ordonnancer les applications sur des architectures présentant des topologies de mémoires hiérarchisées. En associant ces modèles avec des modèles de calcul à base de machines à états (tels que les Modèles Modaux [1]), on adresse un champ d'applications plus vaste comme le radar multi-mode, où les parties à base de boucles intensives doivent être parallélisées alors que le contrôle n'est pas critique en temps de calcul.
- **sur l'implication du concepteur** : on ne peut pas attendre que chaque architecture cible, à plus forte raison hétérogène, ait un outil de parallélisation automatique fiable, i.e. garantissant un certain niveau d'optimisation. Tout en étant capable d'intégrer les méthodes et outils jugés suffisamment fiables pour des architectures particulières, la plateforme doit conserver la possibilité pour le concepteur de choisir les principales options du placement, en lui présentant graphiquement une vision synthétique des possibilités.
- **sur l'itérativité du processus de conception** : l'évaluation de performances, par simulation ou métrologie sur cible après génération de code, est nécessaire et doit fournir rapidement au concepteur les indications de performance lui permettant d'explorer plusieurs options de placement, voire plusieurs architectures cibles.
- **sur la génération de code** la plateforme est chargée de produire automatiquement le code qui résulte du déploiement et de l'ordonnancement de l'application sur l'architecture, sous une forme utilisable par les outils de backend attachés aux composants de l'architecture.

2 Le flot de conception de SpearDE

En partant des critères énoncés ci-dessus, les étapes essentielles du flot de conception dans SpearDE peuvent se résumer comme le montre la Figure 1.

Une spécification fonctionnelle d'une application se fait à travers un graphe de type flot de données basé sur le formalisme Array-OL [3], proche sémantiquement du modèle de calcul flot de données synchrone, multidimensionnel (« Multi-Dimensional Synchronous Dataflow »angl.), rendant explicites les dépendances de données et le parallélisme de boucles potentiel des nœuds du graphe. Les arcs du graphe transportent des tableaux de données multidimensionnels, et les nœuds représentent des nids de boucles statiques accédant à leurs ports d'entrées-sorties suivant des lois affines. Ce modèle est agnostique par rapport à l'architecture de calcul sur laquelle l'application sera projetée.

Le deuxième point crucial dans le flot de conception industriel est la possibilité de spécifier une architecture particulière sur laquelle on souhaite déployer l'application. Le modèle générique de capture d'architecture permet de décrire des structures construites à partir de CPUs, de mémoires et de moyens de communication sans restriction sur leur topologie d'interconnexion (mémoire commune/distribuée, etc.). L'interface permet aussi de spécifier en SystemC le comportement temporel de chaque élément de l'architecture, en vue de la simulation de performances permettant d'évaluer une allocation d'application particulière (exploration de l'espace de design (DSE)). C'est un des avantages de l'outil d'offrir une interface permettant de décrire les architectures cible, potentiellement hétérogènes, au niveau d'abstraction qui convient pour l'utilisateur : la possibilité est conservée de représenter un cluster de processeurs dans tout son détail ou comme un processeur unique virtualisé. Quelques exemples d'architectures déjà modélisées sont également donnés dans la Figure 1. Les propriétés utiles pour la parallélisation sont le nombre de CPU-s, la répartition des mémoires et leurs types, les bus de communication. Des caractéristiques physiques de chacun de ses éléments (tailles, vitesses) sont spécifiés à ce stade, également.

Les saisies d'application et d'architecture représentent les briques de base du flot de conception, guidant par la suite des choix de partitionnement et d'allocation (« mapping »angl.) qui, en absence d'une heuristique universelle doivent être laissées à la portée du concepteur, donc non-automatisés. La traçabilité de ces choix est un critère indispensable, vue la durée de vie des produits dans les systèmes embarqués critiques. Les décisions de placement prises par le concepteur sont en pratique un nombre limité de commandes formalisées par SpearDE qui sont enregistrées et rejouables aisément par la suite.

L'étape suivante dans le flot de conception est le partitionnement. Des groupes de nœuds du graphe sont alloués à des sous - ensembles d'architecture distincts (parallélisme de tâches), chaque sous-ensemble étant appelé un segment. Un segment est potentiellement composé de plusieurs elements de calcul (PEs de « Processing Element ») identiques, et le partitionnement consiste aussi à répartir les itérations des boucles entre les PEs (parallélisme de données). Le partitionnement est une allocation physique des calculs sur l'architecture. Elle doit en général être suivie par une phase d'insertion de nœuds de communication dans le graphe dont la fonction est d'assurer les redistributions de données entre les mémoires nécessaires pour mettre les données à disposition des PEs qui doivent les traiter. Il faut noter que des ré-alignements de données sont aussi nécessaires lorsque les données se trouvent naturellement à por-

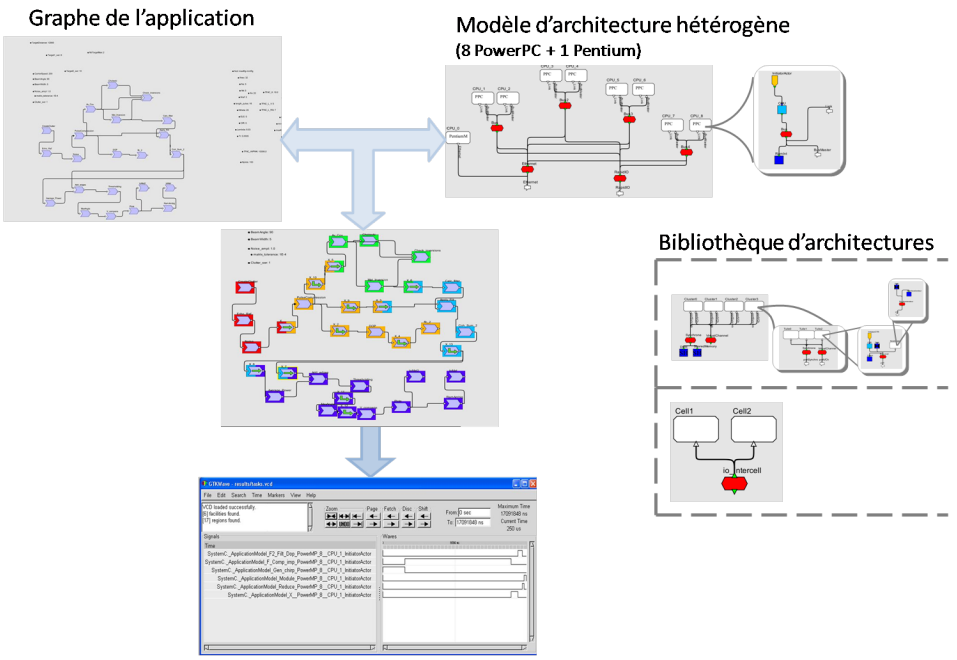

FIGURE 1 – Quelques étapes du flot de conception de SpearDE

tée des PEs qui doivent les utiliser, mais qu'elles sont écrites dans un mauvais ordre : les applications considérées utilisent des tableaux multidimensionnels et leurs traitements peuvent s'appliquer sur des dimensions différentes. L'étape suivante est l'ordonnancement dans le temps au sein de chaque PE. Il passe par des opérations de fusion de boucles qui consistent à factoriser les espaces d'itérations des nœuds du graphe (y compris des nœuds de communication), principalement pour limiter la taille de stockage des données intermédiaires, comme dans l'opération de strip-mining classique pour les DSPs et leur mémoires scratchpad. Le résultat de cette phase de placement est une allocation physique des calculs aux ressources et un ordonnancement statique légal. L'ordonnancement entre segments suit une sémantique similaire aux processus de Kahn (KPN, voir [7]. Tout au long du placement, en se basant sur les relations entre les indices de boucles et les indices des données manipulées, l'outil valide en permanence la légalité du placement, indique la nécessité des opérations de communication et les crée à la demande, et propose des schémas de fusion de boucles en rapport avec les capacités mémoire de l'architecture. Il s'appuie pour cela sur une bibliothèque de fonctions de manipulation de polyèdres en nombres entiers. L'aide à l'opération de fusion passe par une recherche itérative qui ajuste la factorisation des espaces d'itérations d'un ensemble de nœuds en calculant un layout mémoire résultant et en cherchant à le rendre compatible avec la capacité disponible donnée par le modèle d'architecture. Dans la Figure 1 nous avons présenté également le graphe applicatif après l'étape de mapping (pour une architecture à base de PowerPCs et une application de filtrage radar adaptatif espace-temps). Les couleurs dénotent des segments et les boites avec des flèches définissent des communications qui ont été introduites par l'outil. Une fois le mapping réalisé, le concepteur procède à l'étape de génération de code-glue spécifique pour l'architecture et / ou à la simulation de performances (les courbes données en bas de la Figure 1 illustrent l'évaluation de performances avec SystemC [8] pour l'application, l'architecture et la stratégie de mapping considérées). Cette évaluation étant très rapide (quelques minutes, par rapport à des mois de développement en absence d'outil) elle facilite à la fois le processus d'exploration d'architectures, mais aussi d'exploration de stratégies de mapping, tout en gardant une trace des choix effectués. Ceci étant dit, quand l'architecture le permet (en d'autres mots quand la stratégie de mapping optimal est connue) une automatisation du processus sera mise en œuvre.

3 Exemple

Dans [2], une architecture de calcul dédiée au traitement d'images, basée sur un CPU couplé avec un accélérateur multiprocesseur sur FPGA (dans la famille Xilinx Virtex4) et commercialisée dans des produits Thales, a été proposée. Cet accélérateur consiste en 128 Processeurs Élémentaires (PEs) composés à partir de blocs DSP48 Xilinx, capables d'exécuter de façon itérative des fonctions élémentaires disponibles en bibliothèque (par exemple des convolutions). Les PEs sont disposés en anneau et fonctionnent en SIMD sous contrôle d'un processeur softcore de type μBlaze. Bien que très efficace, cette architecture de calcul a un modèle de programmation à trois niveaux de granularité : un niveau gros grain gérant les appels entre le CPU hôte et l'accélérateur, un niveau à grain moyen gérant les activations de PEs dans l'accélérateur et finalement un niveau grain très fin exécutant des calculs en assembleur VLIW

entre les PEs. Il faut noter que, s'agissant d'une architecture non-COTS, en absence d'outil de programmation parallèle générique sur le marché, la tâche de déploiement d'applications revient en général aux concepteurs et reste bien loin de la portée des algorithmiciens. Ceci ralentit fortement le processus de développement industriel, qui se trouve alourdi par les itérations entre algorithmiciens et développeurs matériel, par les diverses phases de debug et test à considérer à tous les niveaux, mais aussi par le manque de flexibilité qui fait que la répercussion d'un changement au niveau applicatif peut mener à une réécriture quasi-totale du code déployé sur la plateforme. A travers SpearDE, cette architecture devient facilement programmable dès lors qu'un modèle fonctionnel de l'application à été validé, grâce à la génération de code automatique. Ce modèle fonctionnel est en général validé à travers la fonctionnalité SpearDE de génération de code C séquentiel exécutable sur un PC local à partir du graphe flot de données. Il est intéressant de remarquer que le graphe applicatif peut être construit par l'algorithmicien ou à partir d'un modèle de référence fourni par l'algorithmicien (sous la forme d'un code C/C++, Matlab, etc.), ce qui permet en plus un prototypage rapide des applications, puisque le temps de développement du graphe applicatif est en général comparable à celui d'un code séquentiel. Dans le cas particulier de l'accélérateur indiqué ci-dessus, qui est un développement interne, on sait définir une stratégie de mapping automatique fiable pour exploiter les 128 PEs de manière optimale. Par conséquent, le mapping d'un graphe ou d'un sous-graphe applicatif sur l'accélérateur, ainsi que la génération du code pour le processeur de contrôle du SIMD est pris en charge par SpearDE sans intervention de l'utilisateur.

En ce qui concerne le code généré, outre son avantage d'être correct par construction, il est important de signaler que dans les architectures distribuées la proportion du code glue est largement supérieure au code effectif fonctionnel. L'écriture manuelle de ce code est à la fois très longue est très sujette aux erreurs.

4 Conclusion

Nous avons présenté la problématique industrielle dans les systèmes embarqués critiques, relative au déploiement massif d'applications parallèles sur des architectures distribuées génériques. Dans le cas de Thales, il a été nécessaire de développer une plateforme permettant la description d'un flot complet pour la parallélisation des applications de type flots de données, capturées avec les spécificités de leur domaine et à travers des Modèles de Calcul, pour lesquelles un prototypage et une évaluation de performances à différents angles, rapides et répétable sont indispensables. Il faut noter que SpearDE n'est pas conçu dans le but d'améliorer les performances précises d'une paire application-architecture parallèle, mais surtout dans le but d'outiller le processus de développement d'application parallèle de traitement de signal multi-dimensionnel, à tous les niveaux afin de répondre à des critères similaires à ceux énoncés dans [4] : productivité, performance, portabilité, traçabilité, prédictibilité. Ce sont surtout ces critères qui gouvernent les choix d'outils dans le milieu industriel. La performance maximum de parallélisation sur une architecture cible serait plutôt une conséquence du flot de développement, de l'exploration de l'espace de conception et du processus itératif d'allocation des modules de l'application sur les modules de calcul. Par ailleurs, dans les systèmes embarqués critiques l'accélération des calculs est parfois en deçà des contraintes de puissance consommée ou de prédictibilité des résultats, ce qui fait qu'une variante sous-optimale en vitesse pourrait être préférée. Garder la largeur de spectre de la plateforme SpearDE impose de laisser la liberté dans ces choix de modélisation. En outre, en supposant qu'une stratégie de mapping, optimisant les découpages de données et l'utilisation de ressources a été trouvée, le déploiement de l'application sur l'architecture est aussi performant que le code décrivant les opérations élémentaires exécutées à l'intérieur des boucles imbriquées dans le graphe applicatif.

Références

[1] UC Berkeley. `http://ptolemy.berkeley.edu/ptolemyII/`.

[2] P. Bonnot, F. Lemonnier, G. Edelin, G. Gaillat, 0. Ruch, and Gauget P. Definition and simd implementation of a multi-processing architecture approach on fpga. In *Design, Automation and Test in Europe (DATE)*, 2008.

[3] P. Boulet. Array-ol revisited, multidimensional intensive signal processing specification. Technical report, RR 6113 INRIA, 2007.

[4] H. Chafi, A. K. Sujeeth, K. J. Brown, H. Lee, A. R. Atreya, and K. Olukotun. A domain-specific approach to heterogeneous parallelism. In *Proceedings of PPoPP'11*, 2011.

[5] T. Grandpierre, C. Lavarenne, and Y. Sorel. Optimized rapid prototyping for real-time embedded heterogeneous multiprocessors. In *Proceedings of 7th International Workshop on Hardware/Software Co-Design, CODES'99*, Rome, Italy, May 1999.

[6] S. Ha, S. Kim, C. Lee, Y. Yi, S. Kwon, and Y.-P. Joo. Peace : A hardware-software codesign environment for multimedia embedded systems. *ACM Transactions on Design Automation of Electronic Systems*, 12(3), 2007.

[7] W. Haid, K. Huang, I. Bacivarov, and L. Thiele. Multiprocessor soc software design flows. *IEEE Signal Processing Magazine*, 26(6) :64–71, 2009.

[8] Open System C Initiative. `http://www.systemc.org`.

[9] B. Kienhuis, E. Rijpkema, and E. F. Deprettere. Compaan : Deriving process networks from matlab for embedded signal processing architectures. In *Workshop on Hardware/Software Co-design (CODES'2000)*, San Diego, CA, USA, May 3-5 2000.

[10] R. Leupers and J. Castrillón. Mpsoc programming using the MAPS compiler. In *ASP-DAC*, pages 897–902, 2010.

[11] H-w. Park, H. Oh, and S. Ha. Multiprocessor soc design methods and tools. *IEEE Signal Processing Magazine*, 26 :72–79, 2009.

[12] BlueBee Multicore Technologies. `http://www.bluebee-tech.com/index.php?q=node/2`.