Environnement de conception AAA alliant cosimulation et prototypage rapide

Erwan Flécher, Mickaël Raulet, Ghislain Roquier, Marie Babel, Olivier Déforges

Environnement de conception AAA alliant cosimulation et prototypage rapide : cas d'étude fondé sur le codec LAR

Erwan Flécher, Mickaël Raulet, Ghislain Roquier, Marie Babel, Olivier Déforges

IETR CNRS UMR 6164/Groupe Image et Télédétection, INSA de Rennes
20 avenue des buttes de coësmes, 35043 Rennes cedex, France
{eflecher, groquier}@ens.insa-rennes.fr {mraulet, mbabel, odeforge}@insa-rennes.fr

Résumé – Dues aux contraintes de temps quelles imposent, les applications temps-réels de traitement du signal et des images rendent indispensable l'emploi d'unités de calcul dédiées. Mutualisant flexibilité et puissance de calcul, les architectures multi-composants dotées d'éléments programmables ont prouvées leurs efficacités. Cet article présente une méthodologie de conception rapide d'applications de traitement du signal et d'image. Les étapes de modélisation d'application, de cosimulation et de portage sur architectures hétérogènes sont disponibles dans un unique environnement et réduisent considérablement les temps de développement. De plus un haut niveau d'abstraction est offert grâce à une génération automatique de code. Finalement, l'intérêt d'une cosimulation Matlab/C est démontré dans un cas d'étude mettant en scène le codeur d'image couleur nommé LAR.

Abstract – Real-time signal and image applications have significant time constraints involving the use of several powerful calculation units. Programmable multi-component architectures have proven to be a suitable solution combining flexibility and computation power. This paper presents a methodology for the fast design of signal and image processing applications. In a unified framework, application modeling, cosimulation and fast implementation onto parallel heterogeneous architectures are enabled and help to reduce time-to-market. Moreover, automatic code generation provides a high abstraction level for users. Finally, the worthwhile nature of Matlab/C language cosimulation is illustrated on a still image codec named LAR.

1 Introduction

Les systèmes multimédia intégrés requièrent de plus en plus de puissance de calcul. Bien que leurs conceptions ne cessent de se complexifier, les délais de mise sur le marché sont de plus en plus courts. Or, l'utilisation de circuits dédiés qui n'ont pas de capacité d'ajustement, ne se prêtent pas à des développements rapides. Une solution alternative consiste à ajouter des composants programmables qu'ils soient softwares (DSP, ARM) ou hardwares (FPGA). [1] propose une méthode de co-conception Matlab/C/VHDL permettant de spécifier, cosimuler et co-synthétiser un système. Une spécification multi-langages est tout d'abord envisagée pour décrire le système et un modèle fonctionnel est ensuite réalisé. Progressivement, les modèles de langage et d'architecture sont raffinés et une validation du système est continument possible par cosimulation. En accord avec [1], les étapes de spécification au niveau système, de cosimulation et de prototypage sur architecture multi-composants dépendent directement de 4 concepts : concurrence ou parallélisme de l'exécution, hiérarchie de la description, communications et synchronisations entre processeurs. Dans [2], ces deux derniers concepts sont mis en oeuvre dans un système multi-langages composé de processus Matlab/JAVA. L'environnement proposé permet de distribuer une application Matlab sur des grappes hétérogènes de stations de travail (COW) synchronisées par un graph flot de contrôle (GFC).
Ce papier présente une méthodologie de conception d'application distribuée de traitement du signal et des images. Optimisant le temps de développement, les phases de *modélisation de l'application*, de *cosimulation* et de *prototypage rapide* sur architecture hétérogène sont supportées au sein d'un environnement basé sur l'outil de prototypage SynDEx[1]. Ainsi notre solution reprend les mécanismes tirant parti des noyaux SynDEx et en propose de nouveaux offrant une génération automatique de code pour la cosimulation et le prototypage.
Ce papier est organisé de la façon suivante : la section 2 introduit la méthodologie AAA ainsi que l'outil SynDEx. La génération d'exécutifs ainsi que les noyaux permettant la modélisation multi-langages et la cosimulation Matlab/C sont décrits en section 3. La section 4 présente notre méthodologie de conception dans son ensemble. La section 5 illustre cette méthodologie sur le codeur d'image couleur nommé LAR. Finalement, la conclusion et les perspectives sont données en section 6.

2 Méthodologie AAA & SynDEx

La méthodologie AAA vise à trouver la meilleure mise en correspondance (ou adéquation) entre une description haut niveau de l'application et une architecture multi-composants. L'algorithme est distribué afin de générer une solution optimisant un certain nombre de critères (temps d'exécution, mémoire) et respectant certaines contraintes. Une fois cette solution trouvée, un script générique est produit pour chaque processeur. La Figure 1 présente les entrées/sorties de la phase d'adéquation.

[1] www-rocq.inria.fr/syndex/

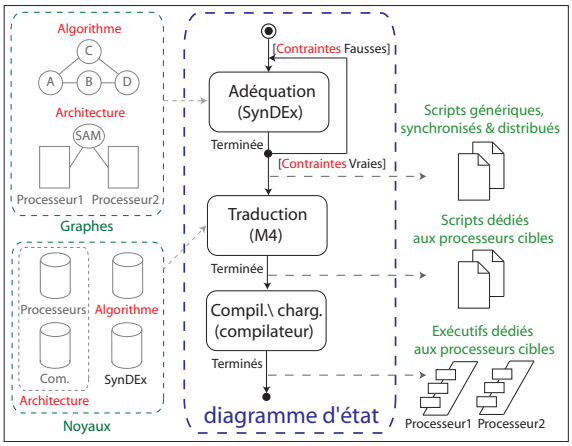


FIG. 1 – Diagramme d'état de l'environnement de conception basé sur SynDEx. L'architecture est composée de deux processeurs interconnectés par une SAM.

2.1 Adéquation Algorithme Architecture

Afin d'exploiter le parallélisme potentiel [3], l'application et l'architecture sont représentées par des hyper-graphes. L'application est modélisée par un graphe flot de données (GFD) où chaque sommet représente une opération. Les sommets du graphe d'architecture décrivent les processeurs (ou composants) qui disposent chacun d'un certain nombre de communicateurs. Ainsi, une architecture multi-composants est représentée par un réseau de machines à états finis (FSM) interconnectées par des média de différentes natures (FIFO, mémoire partagée, ...). Qualifiés de générique en raison de leur indépendance à tout langage de programmation, des scripts sont créés àl'issue de la distribution de l'algorithme. Ils sont composés d'une liste de macro-instructions spécifiant les allocations mémoire, l'ordonnancement de la séquence de calcul et de celui des séquences de communications.

2.2 Synchronized Distributed Executive

Conçu par l'INRIA Rocquencourt, SynDEx est un outil de CAO niveau système supportant la méthodologie AAA et dont les développements actuels sont réalisés en association avec l'IETR/Groupe Image. Bien que principalement développé pour explorer la distribution d'algorithmes, nous montrons ici que SynDEx peut aussi être exploité à des fins de modélisation multi-langages, de cosimulation et de vérification fonctionnelle. En effet, les opérations du GFD peuvent être manuellement partitionnées et exécutées sur des processeurs virtuels. Le mot "virtuel" est utilisé pour indiquer que le GFD est distribué mais exécuté en monoprocesseur afin de (co)simuler l'exécution concurrente de l'application.

3 Traduction de code générique

Les entrées/sorties liées à la phase de traduction sont représentées sur la Figure 1. Dépendant de l'étape de conception du système, deux points de vue concernant la nature du langage cible doivent être considérés afin d'adopter la traduction le plus appropriée. Bien que la mise en oeuvre de l'algorithme avec un langage spécifique à la plateforme cible permet d'en optimiser l'exécution, un développement rapide avec possibilité de cosimulation requière un langage de plus haut niveau tel que Matlab. Ainsi, la méthodologie de conception proposée vise à raffiner progressivement le modèle de langage en passant par des modèles multi-langages intermédiaires.

3.1 Modélisation multi-langages

Le but de ce paragraphe est de justifier le choix du langage associé au modèle fonctionnel de référence. Ce modèle développé rapidement avec un langage de haut niveau sera ensuite utilisé pour de la validation (dite au plus tôt) incluant des étapes de cosimulation et de vérification fonctionnelle. Considérant trois critères, [1] propose une classification de neuf langages de modélisation parmi lesquels Matlab. Le premier critère est le *pouvoir d'expression* qui prend en compte le pouvoir d'abstraction des communications, le modèle de temps et l'efficacité algorithmique. Les fonctionnalités avancées telle que l'interaction avec des outils complémentaires et les transformations possibles, sont évaluées par le deuxième critère qui est le *pouvoir d'analyse*. Le troisième et dernier est le *coût d'utilisation* qui décrit la pénibilité de la prise en main et celle de l'utilisation. Parmi ces neuf langages, Matlab présente plusieurs avantages. En effet, il figure parmi les plus populaires des langages interprétés spécialement dans la communauté des traiteurs du signal et de l'image. Il inclut des fonctions génériques de haut niveau indispensables à une modélisation rapide. De plus, les boîtes à outils Matlab offrent un modèle de temps, des fonctions de visualisation et de transformations nécessaires aux applications de traitement du signal et de l'image. Par contre l'absence de moyens de communication s'avère un handicap lorsque ce langage est utilisé à des fins de modélisation multi-langages et de cosimulation. En section 3.2, ce problème est surmonté en réalisant l'abstraction des communications à l'aide d'un modèle de mémoire SAM et de la génération automatique du code Matlab associé. Pour ce faire, des noyaux de communication et de synchronisation entre processus hétérogènes (au sens du langage) ont été développés.

3.2 Noyaux de traduction

Pour chaque processeur ou processus virtuel, l'exécutif se décline sous forme d'une *séquence de calcul* et d'autant de *séquences de communication* que de média connectés. Uniquement dépendants de l'architecture cible, *les noyaux de processeurs* et *les noyaux de liens de communication* contiennent la traduction des macro-instructions résultantes de l'adéquation. Le premier type de noyaux contient la traduction des instructions liées à la séquence de calcul telles que les allocations mémoires, la gestion des interruptions alors que la seconde prend en charge le transfert et la synchronisation des données.

Noyaux de processeurs : Lors de la phase d'adéquation (Fig. 1), le respect des contraintes est principalement lié au temps d'exécution. Présentée en section 4, notre

méthodologie de conception montre que la modélisation multi-langages ainsi que la cosimulation en monoprocesseur sont essentielles durant la phase de conception. Donc les contraintes liées au temps d'exécution sont lâchées et le regroupement est conditionné par la nature du langage de modélisation de l'opération. La similarité entre cosimulation en monoprocesseur et prototypage sur architecture multi-composants est mise en lumière si l'on considère que les processus hétérogènes sont associés à des processeurs virtuels dans le premier cas de figure et aux composants de l'architecture dans le second. Par conséquent, le GFD est inchangé et seul le graphe d'architecture diffère. Les noyaux de processeurs associés au language Matlab ont été développés pour la séquence de calcul afin de permettre la traduction automatique.

Noyaux de liens de communications : La plus part des noyaux de liens de communications proposés par [5] sont conçu autour d'un modèle pris en charge par la méthodologie AAA [3], le modèle SAM. Il définit un protocole basé FIFO entre deux processeurs ou processus virtuels dont les transferts de données sont synchronisés au moyen de signaux hardwares. Des librairies dédiées à des média classiques tels que TCP ou BIFO ont été développées en langage C pour des transferts multi-PCs et multi-composants. Dans le contexte proposé ici, une librairie associée au protocole TCP et consacrée à Matlab a été développée. Inclue dans un environnement basé sur SynDEx, notre solution permet une communication entre deux processus Matlab (MM), un en Matlab et un autre en C (MC) et plus généralement xM avec pour prérequis qu'un noyau de communication TCP compatible SynDEx ait été développé pour le processus associé au langage x. Rappelons que le transfert de données entre processus est pris en charge et synchronisé par SynDEx avec génération automatique du code associé. Un haut niveau d'abstraction est ainsi offert à l'utilisation n'impliquant qu'une faible connaissance des protocoles de communications.

4 Méthodologie de conception

Cette section présente une extension de la méthodologie proposée par [4] pour le développement rapide d'applications à travers des architectures hétérogènes. Les étapes de *vérification fonctionnelle*, *prototypage virtuel* et *prototypage rapide* sur architecture distribuée y sont définies et sont prises en charge dans un environnement SynDEx. Cette méthodologie est ici améliorée par les étapes de *modélisation d'application*, *cosimulation* et *vérification fonctionnelle* enrichie qui sont supportées par le même environnement de conception. Composée d'un cycle de cinq étapes (Fig. 2), cette méthodologie repose sur la possibilité de raffiner progressivement le modèle de langage en offrant un haut niveau d'abstraction à l'utilisateur. Bien entendu, l'efficacité de calcul se fera au détriment de la capacité de (co)simulation qui n'intervient que lors de la conception.

Conception de l'algorithme : dans cette première étape, un GFD est élaboré dans le but de produire une description haut niveau de l'application. Cette description sera exploitée tout au long du cycle de conception et plus particulièrement lors de l'étape de prototypage. Chacun des sommets du GFD est associé à une fonction réalisée au moyen d'un langage de haut niveau tel que Matlab afin de disposer de tous les avantages de modélisation. Le code Matlab (mono-processus) généré automatiquement par notre environnement SynDEx, produit un exécutif mono-PC et sera considéré comme modèle fonctionnel de référence pour une validation au plus tôt.

Modélisation hétérogène & cosimulation : La cosimulation a été définie comme une exécution concurrente de plusieurs modèles basés sur des langages hétérogènes [1] mais une exécution parallèle sur multi-PC peut aussi être aisément envisagée. A noter que l'objectif final est de produire une exécution respectant des contraintes fortes spécialement dans le cas d'applications de traitement d'image. Ainsi le raffinement du modèle de langage est indispensable et passe nécessairement par des phases de modélisation multi-langages intermédiaires. De plus, la cosimulation et la vérification fonctionnelle (respectivement étape 2 et 3 sur la Fig. 2) sont des outils de validation indispensable. Par exemple la vérification fonctionnelle d'une opération décrite par deux modèles de langage, l'un en langage C et l'autre en Matlab (résultant du modèle fonctionel de référence), peut être aisément réalisée.

Prototypage virtuel & prototypage rapide : Les deux dernières étapes du cycle sont similaires à celles proposées par [4], elles ne sont donc pas détaillées ici. L'étape de prototypage virtuel (étape 4) permet de vérifier la distribution et l'exécution parallèle du GFD en multi-PCs interconnectés par des liens TCP afin d'émuler la topologie de la platforme cible. Dans l'étape 5, SynDEx génère une distribution optimizée pour la platforme cible afin de produire une exécution temps-réel.

5 Application de traitement d'image

Nous proposons d'illustrer l'intérêt de modélisation multi-langages et de la cosimulation sur le codeur d'image couleur nommé LAR [6]. Le LAR (Locally Adaptive Resolution) est une solution efficace de codage d'image couleur à bas débit exploitant une approche région pour le codage des composantes de chrominances. Dans notre cas d'étude, l'accent est mis sur la segmentation auto-extractible (c.à.d à coût de codage minimal) développée dans [6]. En outre, il est montré que cette segmentation spatiale peut être efficacement décrite et évaluée au moyen d'un *graphe d'adjacence des régions* (RAG). Dans un RAG, chaque région est définie au moyen d'un sommet et chaque arc décrit le lien entre deux régions spatialement connexes. Nous avons donc développés une boîte à outils Matlab qui inclut entre autre une fonction d'affichage de RAG dont les ellipses représentants les sommets peuvent être transformées en fonction des propriété des régions. En effet, la position

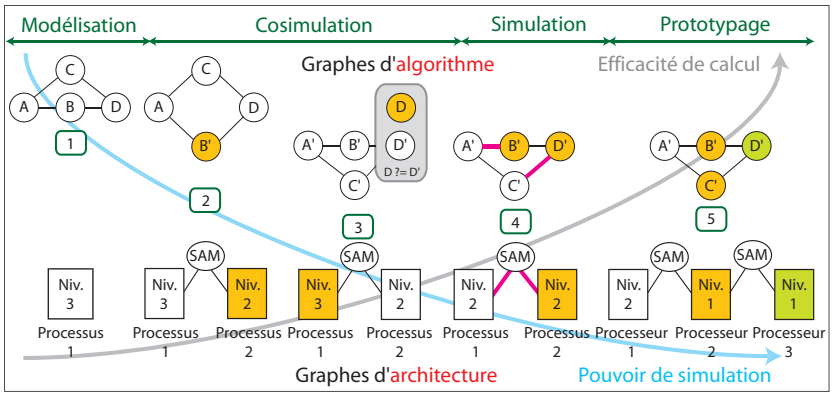
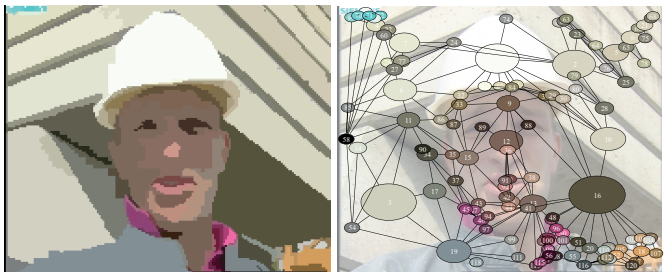


FIG. 2 – Méthodologie dont le cycle inclut cinq étapes : modélisation d'application (1), cosimulation (2), vérification fonctionnelle (3), prototypage virtuel (4) et prototypage rapide (5)

de l'ellipse est donnée par le baricentre de la région, la couleur moyenne des pixels contenus dans la région est attribuée à l'ellipse et sa surface est proportionnelle à celle de la région. Les Figure 3.a et 3.b sont respectivement la représentation en régions et le RAG extraits de l'image 50 de la séquence Foreman (CIF @25Hz).



(a) (b)

FIG. 3 – (a) Segmentation constituée de 120 régions (b) RAG produit par Matlab

Dans l'environnement SynDEx illustré par la Figure 4, les images et le RAG sont affichés à l'aide de fonctions Matlab alors que la segmentation est exécutée rapidement en langage C. L'exécution concurrente est réalisée en utilisant une architecture dédiée composée de deux PCs, l'un réel et l'autre virtuel. Des fonctions génériques de lecture et d'affichage de données peuvent être aisément insérées grâce à la génération de code automatique.

6 Conclusion

Nous avons présentés un environnement de conception d'application distribuée basé sur une modélisation multi-langages. Partant d'un langage de modélisation haut niveau, notre solution basée sur SynDEx permet de converger progressivement vers une application temp-réel avec un haut niveau d'abstraction. Ce travail s'inscrit dans l'élaboration d'un codeur vidéo de seconde génération nommé "LAR vidéo" qui est une extension du codeur d'image couleur. La modélisation et cosimulation C/Matlab permet de concevoir et tester rapidement des nouveaux algorithmes en Matlab alors que l'efficacité de calcul du language C permet un décodage temps-réel dans la séquence d'images.

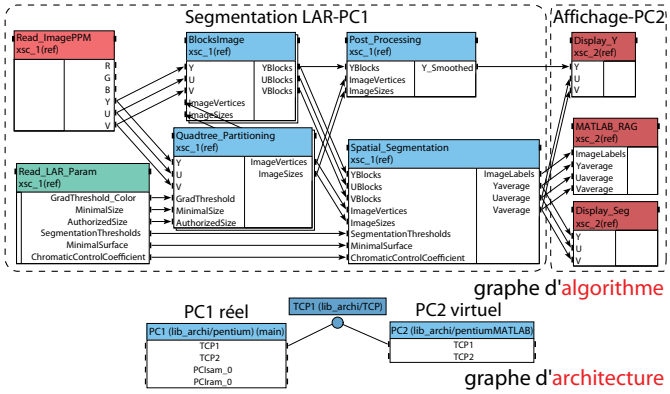


FIG. 4 – Interface SynDEx des graphes d'algorithme et d'architecture de la segmentation LAR

Références

[1] A. A. Jerraya, M. Romdhani, Ph. Le Marrec, F. Hessel, P. Coste, C. Valderrama, G. F. Marchioro, J. M. Daveau and N.-E. Zergainoh, "Multilanguage specification for system design," *System-level synthesis*, pp. 103–136, 1999.

[2] E.S. Manolakos, D. Galatopoullos and A. Funk, "Distributed Matlab Based Signal and Image Processing Using JavaPorts," in *Proc. ICASSP 2004*, Montreal, Canada, May 17-21. 2004.

[3] T. Grandpierre and Y. Sorel, "From algorithm and architecture specifications to automatic generation of distributed real-time executives : a seamless flow of graphs transformations," in *First ACM and IEEE International Conference on Formal Methods and Models for Co-Design*, Mont Saint-Michel, France, June 2003.

[4] M. Raulet, M. Babel, J.-F. Nezan, O. Déforges, and Y. Sorel, "Automatic Coarse Grain Partitioning and Automatic Code Generation for Heterogeneous Architectures," in *Proc. SIPS 2003*, Seoul, Korea, August 27-29. 2003.

[5] M. Raulet, F. Urban, J-F. Nezan, O. Déforges and C. Moy, "SynDEx Executive Kernels for Fast Developments of Applications over Heterogeneous Architectures," in *Proc. EUSIPCO 2005*, Antalaya, Turquey, September 2005.

[6] O. Déforges, M. Babel, L. Bédat and J. Ronsin, "Color LAR codec : a color image representation and compression scheme based on local resolution adjustment and self-extraction region representation," *IEEE Transactions on Circuits and Systems for Video Technology*, accepted, 2007.