Défis GDR GPL

Langages et outils de méthodes formelles pour la conception et la vérification de systèmes électroniques complets (hardware, software et firmware)

Porteur : Sylvain Conchon
Auteurs : Alexandre Chapoutot, Sylvain Conchon, Pierre-Loïc Garoche,
Matthieu Martel, Marc Pouzet

Contexte, positionnement et objectifs de la proposition

Aujourd'hui de nombreux systèmes électroniques sont pilotés et synchronisés par du logiciel. Un exemple d'architecture consiste à implanter des boucles de commande dans des circuits logiques programmables (FPGA) et des modules chargés d'assurer la coopération entre ces boucles de contrôle. C'est le schéma d'implantation choisi par exemple pour la conception des alimentations à découpage des scanners de dernière génération de General Electric Healthcare (GEHC).

La méthode de développement de ces systèmes suit habituellement une approche *model-based design* se basant sur des modèles Matlab/Simulink/SimScape ou Scade dont du code C (ou C++) est extrait (automatiquement) par un compilateur. Ce code est ensuite lié à des programmes écrits à la main avant d'être traduit vers un langage de description de matériel tel que VHDL pour être enfin exécuté par les FPGA. Par exemple, les scanners de GEHC sont constitués de plusieurs boucles de commande (émission, position, dose et taille du faisceau de rayons X) implantées dans des circuits logiques programmables (FPGA) ainsi qu'un module central chargé d'assurer la coopération entre ces boucles de contrôle. Seuls les modules de coopération restent programmés *à la main*. Ces derniers nécessitent une gestion fine de la synchronisation entre des composants dont les différentes horloges sont soumises à des phénomènes de dérive.

Le développement des logiciels embarqués dans de tels systèmes représente un coût de plus en plus important dans l'industrie. Cependant, la mise au point de ces boucles FPGA ou du code de synchronisation est extrêmement difficile et il est sujet à des erreurs qui sont découvertes très tard lors des tests finaux.

C'est pourquoi nous pensons qu'il est important de développer des techniques et des outils pour gagner en confiance sur ces modules de contrôle-commande ainsi que sur les modules de coopération. Bien que les approches existantes reposent sur une bonne méthodologie basée modèles et sur de la génération de code, il nous semble que l'écriture de ces systèmes dans des langages synchrones hybrides avec une sémantique bien définie, ainsi que des outils de méthodes

formelle pour vérifier certaines propriétés fonctionnelles, permettrait de trouver des bugs en phase amont et de mieux appréhender la complexité de ces architectures.

***Verrous et difficultés*.** Ce défi contient tous les ingrédients pour une mise à l'épreuve des langages et outils de méthodes formelles développés dans les GT LTP et Compilation (mélange discret/continu, méthodologie synchrone, vérification de modèles, génération de code). Les verrous à lever sont de plusieurs natures: (1) écrire dans un langage mathématiquement défini une spécification exécutable associée à des spécifications fonctionnelles des modèles, (2) modéliser et analyser les contraintes temps-réels et les problématiques de synchronisation et dérives d'horloges, (3) de mettre au point des méthodes de test/simulation/vérification permettant l'analyse du modèle hybride et (4) d'étudier les problématiques d'erreur de calcul sur le code produit (en virgule fixe ou flottante).

***Thématiques liées à ce défi*.** Les domaines de recherche liés à ce défi scientifique sont multiples. Une liste non exhaustive de thématiques est donnée ci-dessous. Elles sont issues des recherches menées dans les langages de programmation, la compilation, le test, l'interprétation abstraite ou la vérification de modèles.

- **Langages.** Les langages synchrone comme Scade et leurs extensions aux systèmes hybrides comme Zélus pourraient être utilisés pour écrire des programmes qui ne sont pas facilement modélisable en Matlab/Simulink, comme des contraintes de synchronisation ou certaines machines à états.

- **Test de systèmes hybrides**. Les techniques de test basées sur des propriétés (PBT), appelées également *QuickCheck*, semblent pertinentes dans ce contexte. Elles procèdent par génération aléatoire d'entrées filtrées par la propriété à tester. Pour les systèmes hybrides envisagés, cette génération devrait faire appel à un solveur de contraintes (par ex. SMT) qui saura traiter des équations différentielles et générer (et faire varier) des modèles.
  Une autre approche à développer consiste à utiliser les domaines abstraits de l'analyse statique pour construire des trajectoires abstraites. Ces trajectoires ensemblistes correspondent à des tests ou simulations à horizon borné mais capturent toutes les trajectoires faisables. Ces méthodes pourraient être développées et appliquées dans ce contexte de systèmes hybrides complexes combinant des sémantiques à temps continu décrites par des EDO avec des programmes; ces derniers pouvant décrire autant des composants logiciels que matériels (FPGA).

- **Vérification de modèles (Model Checking)**. Un des enjeux important de ce défi est la vérification exhaustive de propriétés sur les modèles hybrides. Des approches comme dReach proposent de combiner satisfiabilité et résolution d'ODE pour raisonner sur l'atteignabilité de propriétés sur des systèmes hybrides. Pour résoudre les problèmes de précision et de passage à l'échelle, il pourrait être pertinent d'intégrer plus finement ces deux méthodes.

- **Analyse numérique**. Les calculs décrits dans des outils comme Simulink sont exprimés dans le corps des réels, sans considérer les détails d'implémentation. Or, lors de la production du code, ces calculs doivent être implémentés en utilisant des nombres à virgule fixe ou

flottante. Les difficultés concernent la précision des calculs du code objet et de son efficacité en termes d'espace utilisé sur les FPGA. Il nous semble important de travailler à des outils de synthèse de code numérique de bas niveau, en virgule fixe et flottante, qui pourront prendre en compte la précision et la contrainte d'espace en s'appuyant sur techniques d'analyse statique et de transformation de programmes.

Impact et retombées

Ce défi va permettre d'améliorer l'expressivité et l'efficacité de plusieurs outils : extension des solveurs SMT au traitement des ODE, model-checking de systèmes hybrides, outil de synthèse de code numérique de bas niveau, langages de haut niveau pour les logiciels de contrôle/commande réactif et leur compilation. Il va également permettre de développer une nouvelle technique de test dirigée par les propriétés pour les systèmes hybrides, ainsi que de proposer une méthode de conception de systèmes multi-horloges avec des contraintes de synchronisation temps-réel fortes.

Nous attendons aussi de ce défi la possibilité de dépasser l'état de l'art en terme de certification pour les applications de contrôle/commande qui combinent logiciel et matériel (en particulier avec une cible de type FPGA).