Curriculum Vitae

Cécile Braunstein

Née à Paris, le 19 octobre 1979
cecile.braunstein@free.fr
http://www-asim.lip6.fr/~cecile/

Adresse professionnelle :
AGRA group - Bremen Universität
Bibliothekstrasse 1 - [MZH]
28359 Bremen
Tel : 0049 [421] 2 18 - 6 39 51

Expériences Professionnelles

2007 - 2008 **Post-doctorante** à l'Université de Bremen (Allemagne), dans le groupe d'architecture des ordinateurs dirigé par Pr. Rolf Drechsler

2006 - 2007 **ATER**, Attachée temporaire d'enseignements et de recherche à l'UFR d'informatique de l'Université Paris 6.
Encadrante stage de Master

2003 - 2006 **Enseignante vacataire** d'informatique en L3 et M2 à l'Université Paris 6, (111h)

2003 **Stage de DEA** : «Vérification de propriétés logico-temporelles lors de la conception incrémentale des wrappers VCI-PI», (laboratoire LIP6/ASIM)

2002 **Stage d'été** : «Modélisation et vérification de propriétés CTL sur les wrappers VCI-PI en VIS» (laboratoire LIP6/ASIM)

2002 **Stage de Maîtrise** : «Noyau multi-tâche pour micro-contrôleur PIC» (laboratoire LIP6/ASIM)

Formation

2003 - 2007 **Doctorat d'informatique** (Université Paris 6), mention très honorable, thèse intitulée : «Conception incrémentale, vérification de composants matériels et méthode d'abstraction pour la vérification de systèmes intégrés sur puce».
Soutenue le 14 mai 2007, devant le jury composé de :

- Dominique Borionne (Professeur, TIMA Grenoble France)
- Béatrice Bérard (Professeur, LAMSADE Paris France)
- Emmanuelle Encrenaz (Encadrante) (MdC, LIP6 France)
- Alix Munier-Kordon (Professeur, LIP6 Paris France)
- Hans Eveking (Professeur, TU-Darmstadt Allemagne)
- Alain Greiner (Professeur, LIP6 Paris France)

2002 - 2003 **DEA Architecture des systèmes intégrés et micro-électronique (ASIM)** (Université Paris 6), mention bien

2001 - 2002 **Maîtrise d'informatique** (Université Paris 6), mention bien

Recherche

Thématiques de recherche

- Application des méthodes formelles pour la vérification de systèmes intégrés sur puce
- Lien entre Conception et vérification de systèmes intégrés sur puce
- Model checking
- Méthode d'abstraction
- Boucle CEGAR (counter-example guided abstraction refinement)
- Diagnostique et localisation automatique de bugs

Mots clés : SoC, Model checking, CTL/PSL, Abstraction, CEGAR, Debug diagnosis, Fault localization

Thèse

Titre : Conception incrémentale, vérification de composants matériels et méthode d'abstraction pour la vérification de systèmes intégrés sur puce
Mots clés : Vérification formelle, CTL, Model Checking, Abstraction, CEGAR, Architecture matérielle
Soutenue le 14 mai 2007 avec la mention très honorable devant le jury composé de :

- Dominique Borionne (Professeur, TIMA Grenoble France), présidente
- Béatrice Bérard (Professeur, LAMSADE Paris France), rapporteur
- Emmanuelle Encrenaz (Encadrante) (MdC, LIP6 France), encadrante de thèse
- Alix Munier-Kordon (Professeur, LIP6 Paris France), Directrice de Thèse
- Hans Eveking (Professeur, TU-Darmstadt Allemagne), rapporteur
- Alain Greiner (Professeur, LIP6 Paris France), examinateur

Cette thèse a démarré en octobre 2003. Elle a été préparée au Laboratoire d'Informatique de Paris 6 (LIP6), dans le thème systèmes intégrés sur puce (SOC, anciennement ASIM), sous la direction d'Emmanuelle Encrenaz au sein de l'Ecole Doctorale d'Informatique, Télécommunications et Electronique de Paris (EDITE). Les travaux ont été financés par une allocation MENRT dans un premier temps, puis par un poste d'ATER à l'université Paris VI.

Contexte

Aujourd'hui, la conception de systèmes embarqués sur puce (ou "system on chip" SoC) exige l'utilisation de composants préexistants (appelés "cœurs" ou "IP cores" [4]). Ces cœurs peuvent provenir de compagnies industrielles ou d'universités différentes. Il est alors crucial de garantir ces IP ont un comportement correct avant de les intégrés dans la réalisation d'un SoC. L'enjeu de la vérification formelle de SoC est d'une part de vérifier chacun des composants individuellement, d'autre part de vérifier le système complet. L'utilisation de méthodes formelles permet ainsi d'obtenir des composants réutilisables

de meilleur qualité [5]. Le problème majeur de la vérification formelle est la limitation en taille et en complexité du système à vérifier, dû à l'explosion combinatoire du nombre d'états des algorithmes de vérification.

La vérification de composants matériels est souvent réalisée par simulation du modèle du composant écrit en langage de description de matériel (HDL). L'avantage de cette méthode est sa capacité à traiter des systèmes de grande taille (mettant en oeuvre plusieurs composants) ; la contrepartie est que ce type de vérification n'est pas complète et très coûteuse en temps. D'un autre côté, les techniques de model checking fournissent une vérification complète mais ne sont pas applicables sur des systèmes trop grands. Dans les deux cas, il est intéressant de construire des modèles plus "grossiers" qui contiennent moins d'informations que le système complet, mais avec assez d'information pour évaluer la validité d'une propriété du modèle. Ce type de technique est appelé abstraction [3, 2]. L'enjeu est alors de trouver une abstraction conservative, adaptée au model checker et pouvant être construite automatiquement afin de ne pas annihiler les méthodes de vérification automatique.

Une abstraction entraîne une perte de précision du modèle. Il est alors possible d'avoir une propriété vérifiée sur le modèle concret mais fausse sur le modèle abstrait. Dans ce cas on peut obtenir un contre-exemple décrivant un comportement incorrect du modèle abstrait (contre-exemple abstrait) qui ne correspond à aucun comportement du modèle concret. Ce type de contre-exemple est appelé *spurious counter-example*. Cela montre que l'abstraction est trop "grossière", il faut ajouter plus de détails au modèle abstrait, c'est à dire raffiner l'abstraction. Clarke et Grumberg [1] ont proposé une méthode automatique pour raffiner l'abstraction basée sur le contre-exemple fourni par le model checker (CEGAR loop). Cette boucle est réalisée de la manière suivante :

1. Création de l'abstraction initiale
2. Vérification de la propriété sur l'abstraction
3. Si la propriété est vraie alors elle est vraie sur le modèle concret (fin)
   Sinon un contre-exemple est fourni
4. Test de concrétisation du contre-exemple abstrait :
   Si le contre-exemple peut être appliqué au modèle concret alors la propriété est fausse (fin)
   Sinon le contre-exemple est *spurious*
5. Raffinement du modèle abstrait pour invalider le contre-exemple : générer un nouveau modèle abstrait et aller à 2.

Cette technique est de plus en plus utilisée pour la vérification de logiciel (BLAST [7] ...) et seulement depuis très récemment pour la vérification de matériel (VCEGAR [6]).

Références

[1] **Counterexample-Guided Abstraction Refinement**, E.M. Clarke and O.Grumberg and S. Jha and Y. Lu and H. Veith, CAV, 2000, LNCS vol. 1865, pages 154-169.

[2] **Construction of Abstract State Graphs with PVS**, S. Graf and H. Saïdi, CAV, 1997, LNCS vol. 1254, pages 72-83.

[3] **Model Checking, Abstraction, and Compositional Verification**, D. E. Long, PhD thesis, 1993, Carnegie Mellon University.

[4] **IP Reuse in System on a Chip Design**, R. Camposano and W. Savage and J. Chilton, VLSI Design, 2000, pages 20.

[5] **Using formal verification to create robust IP**, J. Littlefield, EE Times, 2004.

[6] **VCEGAR : Verilog CounterExample Guided Abstraction Refinement**, H. Jain and D. Kroening and N. Sharygina and E. Clarke,TACAS, 2007.

[7] **Software Verification with BLAST**, Henzinger, T.A. and Jhala, R. and Majumdar, R. and Sutre, G, SPIN, 2003, LNCS vol. 2648, 235–239.

Déroulement de la thèse

Mes travaux de recherche concernent l'intégration des méthodes formelles pour la conception et la vérification des systèmes intégrés sur puce (SOC). Plus précisément, dans la vérification de systèmes matériels modélisés par des machines de Moore et spécifiés en logique temporelle arborescente (CTL). Mes recherches se décomposent en deux parties. Dans un premier temps, je me suis concentrée sur la conception et la vérification d'un unique composant. Dans une seconde partie, je me suis intéressée aux techniques d'abstraction de systèmes pour la vérification d'un ensemble de composants.

Pendant ma thèse j'ai étudié le lien entre la conception de composants matériels et leur spécification. L'objectif était d'une part d'aider à la conception d'architecture, et d'autre part d'alléger la vérification par model checking. J'ai formalisé un cadre de conception incrémentale et établi des règles de transformations qui permettent de dériver automatiquement la spécification du composant lors de sa conception. Ces travaux ont donné lieu à plusieurs présentations lors des conférences RSP06, LPAR06 ainsi qu'à la publication d'un article dans le journal STTT. De plus j'ai appliqué la conception incrémentale à la conception de convertisseur de protocoles (entre les protocoles VCI et PI).

J'ai aussi commencé l'étude de méthodes d'abstraction dans le cadre de la boucle CEGAR. L'objectif est de vérifier des propriétés globales sur un système complexe composé de plusieurs IP cores. Ce travail est la première étape pour une nouvelle boucle CEGAR où l'abstraction est basée sur la spécification de chacun des composants. La spécification est un ensemble de propriétés CTL. J'ai défini un algorithme d'abstraction qui construit l'abstraction d'un composant à partir d'un sous-ensemble de sa spécification. Cet algorithme a été présenté lors de la conférence ACSD07 et un premier prototype a été réalisé.

Post-doctorat

Depuis octobre 2007, je suis en stage post-doctoral à l'université de Bremen (Allemagne), au sein de l'équipe Rechnerarchitektur dirigé par Pr. Rolf Drechsler. Mon projet de recherche s'inscrit dans le cadre de la vérification d'architectures matérielles par bounded model checking (BMC) à l'aide de propriété PSL. Les algorithmes de bounded model checking déroulent le FSM pour un nombre fixé d'étapes k et vérifient si une violation de la propriété peut être obtenue en au plus k étapes. La philosophie du BMC diffère du

model checking, dans le sens où le model checking essaie de prouver qu'un modèle vérifie sa spécification alors que le BMC tente de montrer l'existence de BUG dans le modèle.

Dans un premier temps, j'ai réalisé une boucle CEGAR pour la vérification par BMC. Cette implémentation s'insère dans la suite d'outils développée au sein du groupe. L'abstraction est effectuée par *localization reduction* [3]. La détermination d'un spurious contre-exemple se fait à l'aide d'un SAT solver. Une instance SAT est créée à partir du modèle concret déroulé pour la longueur du contre-exemple, cette instance est alors contrainte par les valeurs fournit par le contre-exemple. Dans le cas d'un spurious contre-exemple la formule est UNSAT. Les variables responsables du conflit (qui se trouvent dans l'UNSAT core) sont le point de départ du raffinement.

La seconde étape a été d'adapter cette boucle CEGAR pour la localisation de fautes dans une architecture ([1]). Cette technique vient enrichir les techniques de diagnostique et de recherches de bugs par SAT solver et vérification de propriété ([2]). Le but de ces techniques est de fournir un ensemble de portes ou de modules qui sont susceptibles d'être la source du mauvais comportement du modèle. A partir d'une propriété PSL, d'un modèle et d'un ensemble de contre-exemples, on utilise un SAT Solver pour déterminer un ensemble de candidat potentiellement responsable du BUG. L'utilisation d'une boucle CEGAR a pour objectif d'accélérer le diagnostique en se concentrant plus rapidement sur les parties fautives du modèle. De plus, nous cherchons à pouvoir gérer des modèles plus grands et plus complexes. Les différentes questions soulevées se concentrent sur les points suivant :

- La recherche de contre-exemples
- la gestion de fautes multiples
- La détermination d'une abstraction adaptée au niveau de description du modèle

Le premiers résultats montrent une forte diminution de la consomation mémoire ainsi que du temps de diagnostique. Nous voullons maintenant est de mettre en place une boucle de rafinement permettant le diagnostique complet pour une propriété donnée et non plus pour un seul ensemble de contre-exemples. Pour cela, l'idée est de produire de nouveaux contre-exemples en prenant en compte les informations obtenus lors du deboguage.

Références

[1] **Fault diagnosis and logic debugging using Boolean satisfiability** ,A. Smith and A. G. Veneris and M. Fahim Ali and A. Viglas, IEEE Trans. on CAD of Integrated Circuits and Systems, 2005, vol.24, pages 1606-1621.

[2] **Automatic Fault Localization for Property Checking**, S. Staber and G. Fey and R. Bloem and R. Drechsler, Haifa Verification Conference , 2006, pages 50-64.

[3] **Computer-Aided Verification of Coordinating Processes : The Automata-Theoretic Approach**, R. P.Kurshan, Princeton University Press, 1994.

Publications

L'ensemble de documents suivants peuvent être téléchargé à l'adresse suivante :
http://www-asim.lip6.fr/~cecile

Thèse

[1] *Conception Incrémentale, Vérification de Composants Matériels et Méthode d'Abstraction pour la Vérification de Systèmes Intégrés sur Puce*, Thèse de l'Université Paris 6, May 2007.

Journal International

[1] *CTL-Property transformation along an Incremental Design Process* , C. Braunstein and E. Encrenaz, International Journal on Software Tools for Technlogy Transfer (STTT), volume 9, 2007.

Conférences internationales avec comité de séléction et actes

[1] *Using CTL formulae as Component Abstraction in a Design and Verification Flow*, C. Braunstein and E. Encrenaz, *accepted* in the 7th International Conference on Application of Concurrency to System Design (ACSD 2007).

[2] *Formalizing the Incremental Design and Verification Process of a Pipelined Protocol Converter*, C. Braunstein and E. Encrenaz, IEEE International Workshop on Rapid System Prototyping, RSP'06, 2006.

[3] *CTL-Property Transformation along an incremental Design Process*, C. Braunstein and E. Encrenaz, Proceedings of the 4th International Workshop on Automated Verification of Critical Systems, AVOCS'04, Electronic Notes in Theoritical Computer Science (128), pages 263-278, 2004.

Conférences internationales avec comité de sélection

[1] *A Further Step in the Incremental Design Process : Incorporation of an Increment Specification*, C. Braunstein and E. Encrenaz, Logic for Programming Artificial Intelligence and Reasoning, LPAR'06, *short paper*,
proceedings online http://www.lix.polytechnique.fr/~hermann/LPAR2006/, 2006.

Enseignement

Activités d'enseignement

Mes activités d'enseignement ont commencé en 2003, lors de ma première année de thèse. J'ai dans un premier temps été vacataire pendant deux ans à l'Université Pierre et Marie Curie ainsi qu'à Polytech' Paris. J'ai lors de cette période enseigné le langage C et une introduction au système UNIX à des étudiants en chimie et biologie. J'ai été ATER à l'université Pierre et Marie Curie durant l'année 2006/2007. Mes enseignements ont été répartis entre l'UFR d'informatique et l'école Polytech' Paris. J'ai pu à cette occasion donner des enseignements très variés à des étudiants de tous niveaux. Le Tableau 1 montre le détail de mes enseignements, pour un total de 304 h équivalent TD.

Encadrement

Durant ma thèse, j'ai proposé et encadré un stages de Master 1 :
Implémentation d'un Algorithme de synthèse de propriétés CTL en automate de Moore, Fadila Drif.

Détails des enseignements

2006 - 2007 **Programmation impérative et structures de données en C - L2**, (70h) UFR Informatique Paris 6.
Chargée de TD et TP.
Encadrement de Projet Long PhP MySql- M1, (40h) 2ème année École Polytechnique Universitaire Pierre et Marie Curie (Polytech'Paris), spécialité Electronique Informatique.
Rédaction de sujets et évaluation des étudiants.

Automne 2006 **Machine et Représentation - L2**, (40h) UFR Informatique Paris 6.
Chargée de TD et TP.

2004 - 2007 **Informatique Générale - L3**, (126h) 1ère année École Polytechnique Universitaire Pierre et Marie Curie (Polytech'Paris), spécialité Agro-alimentaire et Matériaux.
Chargée de TD et TP, rédactions de TD, du contrôle continu et des examens finaux.

2005 **Preuve Formelle - M2**, (4h), spécialité Architecture et Conception de Systèmes Intégrés (ACSI)
Cours sur les relations de simulations/bisimulations, et présentation de mes activités de recherche.

2003 & 2004 **Projet MIPS - M2**, (16h), spécialité Architecture et Conception de Systèmes Intégrés (ACSI)
Encadrement d'un groupe de 4 étudiants, réalisation d'un processeur MIPS

micro programmé.

2003 **TP : Méthodologie de conception VLSI - M2**, (8h), spécialité Architecture et Conception de Systèmes Intégrés (ACSI)
Chargée de TP, aide à la prise en main des outils de la chaîne de conception ALLIANCE.

Année	Niveau	Intitulé	Nombre d'heures (eq. TD)
2006-2007	M1	Projet Long	40h
2006-2007	L2	C avancé	70h
2006-2007	L2	Machine et Représentation	40h
2004-2007	L3	Informatique Générale	126h
2005	M2	Preuve formelle	4h
2003-2004	M2	Projet MIPS	16h
2003	M2	Méthodologie de conception VLSI	8h
		total	304h

TAB. 1 – Récapitulatif des Enseignements

Projet d'enseignement

Les domaines d'enseignement dans lesquels je souhaiterais m'investir s'articulent autour des thèmes suivants :

- Algorithmique et programmation, quel que soit le langage,
- Systèmes informatiques et systèmes d'exploitation,
- Architecture matérielle et langage assembleur,
- Outils mathématiques appliqués à l'informatique,
- Techniques de vérification formelle