# Curriculum Vitae

Cécile Braunstein

Born October 19th, 1979
French citizenship
cecile.braunstein@lip6.fr
http://www-asim.lip6.fr/~cecile/

Address :
AGRA group - Bremen Universit
Bibliothekstrasse 1 - [MZH]
28359 Bremen
Tel : 0049 [421] 2 18 - 6 39 51

## Academic record

*2003 - 2007* **PhD thesis in computer science** (University of Paris 6) with high honors, Laboratory of computer science under the supervision of Emmanuelle Encrenaz, in the System on Chip team. Dissertation titled: "Incremental Design Process, Verification of Hardware Components and Abstraction Method for the Verification of System on Chip"
Defense date: May 14th 2007. Committee in charge:

- Dominique Borionne (Professor, TIMA Grenoble France)
- Béatrice Bérard (Professor, LAMSADE Paris France)
- Alix Munier-Kordon (Professor, LIP6 Paris France)
- Hans Eveking (Professor, TU-Darmstadt Germany)
- Alain Greiner (Professor, LIP6 Paris France)

*2002 - 2003* **M.Sc. in Computer Science** (University Paris 6) specialization in system on chip with high honors

*2001 - 2002* **B.Sc. in Computer Science** (University Paris 6) with high honors

## Work experience

*2007 - 2008* **Post-doctorant** (University of Bremen, Germany), in the AGRA group of Pr. Rolf Drechsler

*2006 - 2007* **Temporary assistant professor**, (University Paris 6) 200h
**Advisor** for a master's internship

*2003 - 2006* **Teaching assistant** (University Paris 6) 111h

*2003* **Master's thesis internship** Verification of temporal properties along the incremental design of wrappers VCI-PI

*2002* **Summer internship** Design and verification of CTL properties on wrappers VCI-PI with VIS

*2002* **Bachelor's degree internship** Multi-task kernel for the micro-controller PIC

## Thesis abstract

Title : Incremental design process, Verification of hardware components and Abstraction method for the verification of system on chip.

This thesis deals with formal verification of integrated system on chip by model checking. We propose an incremental design process for the verification of a hardware component. This method is a framework for designing a component by successively adding some new behaviours. The incremental design process guarantees, *by construction*, the non-regression of a component during the design process. Moreover, the specification is automatically derived with taking into account the new behaviours. We apply our method with success for the design of a protocol converter.

The formal verification process of system on chip suffers from the states explosion problem. Abstraction techniques aim at alleviating this problem. We state an abstraction algorithm based on the specification of each component. We construct a Kripke structure directly from a subset of the specification written with CTL formulas. This abstraction takes place in a counter-example guided abstraction refinement framework. The first experiments show an important benefit in terms of verification time and an increase in the size of the system we are now able to check. This result reinforces the interest of our abstraction method.

## Keywords

Hardware design, SoC, protocol converter, formal verification, Model Checking, CTL, Abstraction, CEGAR

## Context

Contemporary system-on-chip (SoC) design demands the use of pre-existing intellectual property (IP). IP's may come from different company or academic university. It is now crucial to insure that each IP is correct before it is used in SoC projects. The use of formal method may lead to produce a higher-quality reusable design. The issue of the formal verification in SoC is firstly to check IP individually and secondly to provide the verification of the whole system. As well known, the formal verification of large designs is limited by the complexity of the system due to state space explosion problem.

### The incremental design process

We first formalized an incremental design process to help designers at designing and specifying one component. In our context hardware components are defined by synchronous Moore machine and the specification logic used is CTL. This method stems from the observation of the way some hardware components may be designed. In some cases, hardware designers adopt an *incremental strategy*: after having defined the information flow of the design, the rough structure of the data paths and the control part, they proceed to the implementation of the simplest cases up to the most complex ones. This is accomplished by *adding* new

functionalities, thus building a more and more complex device. This is particularly true for devices implementing a pipe-line flow: stages of the pipe-line can be roughly drawn and then the stalling actions are added. We believe this incremental approach provides a framework that simplifies the design process, by treating difficulties one by one instead of having to face them altogether.

We are interested in exploring the links between properties that are true in an initial model and those that are true in the extended one. This might be expressed as: " May we transform the CTL formulae that are true in the initial model into other CTL properties that are true in the extended one, capturing the way the extension was performed ?". Given an additive increment, the initial model and the extended one, we showed that this CTL transformation is possible. The transformed CTL formulae, applied to the extended model, restrict the verification state-space traversal to a sub-graph isomorphic to the one derived from the initial model. We showed how these transformations can be used to perform non-regression analysis. In a general way, when a designer modifies a component, he has to insure that the modification did not induce *regression*: the correction does not disturb other correct functionalities.

We applied this method for the design and the verification of the wrapper VCI-PI. We develop a prototype tool to automatically transform the set of CTL properties.

**Abstract component from specification**

The major technique to handle the state-explosion problem is the use of abstraction The major approach used to perform such a verification is the simulation of component models, described in a hardware description language. Its main advantage is the ability to deal with large systems, encompassing numerous components; the counterpart is the partial verification it provides. On the opposite side, model-checking provides a complete verification, but is only applicable to small systems. In both cases, one is interested to build "lighter" models, containing less information than the complete ones, but still sufficient information to be able to state if the property is satisfied or not.

The abstraction of a component is derived from a set of logical properties the component satisfies. This set of properties is expressed in the temporal logic CTL. We develop an algorithm that build an abstract Kripke structure from CTL\X formulae. The abstraction is conservative, but not exact, the abstract model contains less information than the concrete one. Thus, checking an abstract model potentially produces incorrect results. The abstraction guarantees that if a property is true in the abstract model, it is also true in the concrete model. On the other hand if the abstract model invalidates a property the concrete one may still satisfy the property The abstraction may be too coarse to subtly capture the behaviour of the concrete component.

This abstraction is the first step to the definition of a counter-example guided abstraction refinement based on the set of specification of each components. Figure  shows the following loop : when a property is not satisfied by an abstract model, an abstract counter-example is provided by the verification tool (or the simulation engine). This counter-example must be analyzed in order to determine if it is a spurious counter-example. In this case, the abstract model can be iteratively constrained, by adding new properties in order to refine the abstraction.

We applied this method for the verification of a platform composed with initiator and target VCI, a PI-bus and master and slave wrappers. A first prototype tools transform a CTL formula into a verilog description of the abstract Kripke structure.

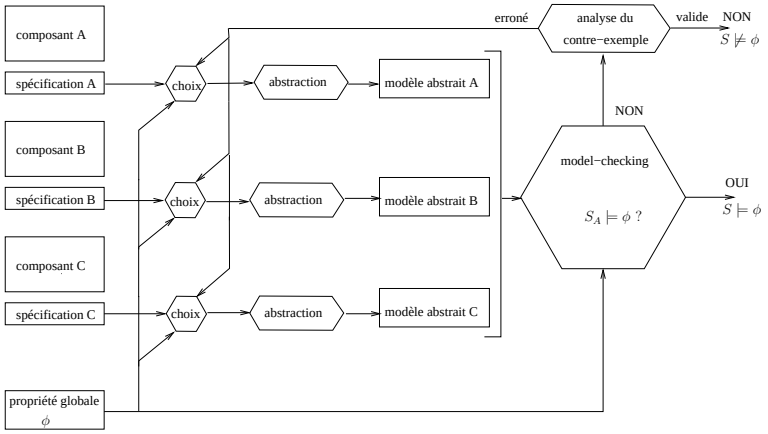


Figure 1: CEGAR Loop

## Publications

The following documents can be downloaded at http://www-asim.lip6.fr/~cecile

### Thesis

[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.

### International journals with selection committees

[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, February 2007.

### International conference with selection committees and proceedings

[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, Chania, Greece, June 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, London, England, September 2004.

### International conference with selection committees

[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/, november 2006.