IN PARTNERSHIP WITH:
CNRS

Université des sciences et technologies de Lille (Lille 1)

Activity Report 2016

Project-Team DREAMPAL

Dynamic Reconfigurable Massively Parallel Architectures and Languages

RESEARCH CENTER
Lille - Nord Europe

THEME
Architecture, Languages and Compilation

Table of contents

1. Members **1**
2. Overall Objectives **1**
3. Research Program **2**
3.1. New Models for New Technologies 2
3.2. Multi-softcore on 3D FPGA 3
3.3. When Hardware Meets Software 3
4. Highlights of the Year **4**
5. New Software and Platforms **4**
5.1. HoMade 4
5.2. JHomade 5
6. New Results **5**
6.1. A Language-Independent Proof System for Full Program Equivalence 5
6.2. A Generic Framework for Symbolic Execution: a Coinductive Approach 5
6.3. Circuit Merging versus Dynamic Partial Reconfiguration -The HoMade Implementation 5
6.4. Language Definitions as Rewrite Theories 6
6.5. SCAC-Net: Reconfigurable Interconnection Network in SCAC Massively parallel SoC 6
6.6. Proving Reachability-Logic Formulas Incrementally 6
7. Partnerships and Cooperations **7**
8. Dissemination **7**
8.1. Promoting Scientific Activities 7
8.1.1. Scientific Events Selection 7
8.1.2. Research Administration 7
8.2. Teaching - Supervision - Juries 7
9. Bibliography **7**

Project-Team DREAMPAL

Creation of the Team: 2013 January 01, updated into Project-Team: 2015 January 01, end of the Project-Team: 2016 December 31

Keywords:

Computer Science and Digital Science:

1.1.2. - Hardware accelerators (GPGPU, FPGA, etc.)
1.1.12. - Non-conventional architectures
2.1.1. - Semantics of programming languages
2.4.2. - Model-checking

Other Research Topics and Application Domains:

6.6. - Embedded systems
7.2.1. - Smart vehicles

1. Members

Research Scientist

Vlad Rusu [Team leader, Inria, Researcher, HDR]

Faculty Members

Ahmad Shadi Aljendi [Univ. Lille I, Teaching Assistant, until Sep 2016]
Rabie Ben Atitallah [Univ. Valenciennes, Associate Professor, HDR]
Jean-Luc Dekeyser [Univ. Lille I, Professor, HDR]
Frederic Guyomarch [Univ. Lille I, Associate Professor]
Philippe Marquet [Univ. Lille I, Associate Professor]
Samy Meftali [Univ. Lille I, Associate Professor]

PhD Students

Karim Ali [Univ. Valenciennes, until Sep 2016]
Wissem Chouchene [Univ. Lille I]

Post-Doctoral Fellow

Andrei Arusoaie [Inria, until Feb 2016]

Administrative Assistant

Corinne Jamroz [Inria]

2. Overall Objectives

2.1. Executive Summary

Standard Integrated Circuits are reaching their limits and need to evolve in order to meet the requirements of next-generation computing. We anticipate that FPGAs (Field Programmable Gate Arrays) will play a major role in this evolution: FPGAs are currently only one or two generations behind the most advanced technologies for standard processors, and their application-specific hardware is an order of magnitude faster than software solutions on standard processors. One of the most promising evolutions are next-generation 3D-FPGAs, which, thanks to their fast reconfiguration and inherent parallelism, will enable users to build dynamically reconfigurable, massively parallel hardware architectures around them. This new paradigm opens many opportunities for research, since, to our best knowledge, there are no methodologies for building such architectures, and there are no dedicated languages for programming on them.

We shall thus address the following topics: proposing an execution model and a design environment, in which users can build customized massively parallel dynamically reconfigurable hardware architectures, benefiting from the reconfiguration speed and parallelism of 3D-FPGAs; proposing dedicated languages for programming applications on such architectures; and designing software engineering tools for those languages: compilers, simulators, and formal verifiers. The overall objective is to enable an efficient and safe programming on the customized architectures. Our target application domain are embedded systems performing intensive signal/image processing (e.g., smart cameras, radars, and set-top boxes)

3. Research Program

3.1. New Models for New Technologies

Over the past 25 years there have been several hardware-architecture generations dedicated to massively parallel computing. We have contributed to them in the past, and shall continue doing so in the Dreampal project. The three generations, chronologically ordered, are:

- Supercomputers from the 80s and 90s, based on massively parallel architectures that are more or less distributed (from the Cray T3D or Connection Machine CM2 to GRID 5000). Computer scientists have proposed methods and tools for mapping sequential algorithms to those parallel architectures in order to extract maximum power from them. We have contributed in this area in the past.
- Parallelism pervades the chips! A new challenge appears: hardware/software co-design, in order to obtain performance gains by designing algorithms together with the parallel architectures of chips adapted to the algorithms. During the previous decade many studies, including ours in the Inria DaRT team, were dedicated to this type of co-design. DaRT has contributed to the development of the OMG MARTE standard (http://www.omgmarte.org) and to its implementation on several parallel platforms. Gaspard2, our implementation of this concept, was identified as one of the key software tools developed at Inria: http://www.inria.fr/en/centre/lille/research/platforms-and-flagship-software/flagship-software.
- The new challenge of the 2010s is, in our opinion, the integration of dynamic reconfiguration and massive parallelism. New circuits with high-density integration and supporting dynamic hardware reconfiguration have been proposed. In such architectures one can dynamically change the architecture while an algorithm is running on it. The Dynamic Partial Reconfiguration (DPR) feature offered by recent FPGA boards even allows, in theory, to generate optimized hardware at runtime, by adding, removing, and replacing components on a by-need basis. This integration of dynamic reconfiguration and massive parallelism induces a new degree of complexity, which we, as computer scientists, need to understand and deal with order to make possible the design of applications running on such architectures. This is the main challenge that we address in the Dreampal project. We note that we adress these problems as computer scientists; we do, however, collaborate with electronics specialists in order to benefit from their expertise in 3-D FPGAs.

Excerpt from the HiPEAC vision 2011/12

> *"The advent of 3D stacking enables higher levels of integration and reduced costs for off-chip communications. The overall complexity is managed due to the separation in different dies, independently designed."*

FPGAs (Field Programmable Gate Arrays) are configurable circuits that have emerged as a privileged target platform for intensive signal processing applications. FPGAs take advantage of the latest technological developments in circuits. For example, the Virtex7 from Xilinx offers a 28-nanometer integration, which is only one or two generations behind the latest general-purpose processors. 3D-Stacked Integrated Circuits (3D SICs) consist of two or more conventional 2D circuits stacked on the top of each other and built into the same IC. Recently, 3D SICs have been released by Xilinx for the Virtex 7 FPGA family. 3D integration will vastly increase the integration capabilities of FPGA circuits. The convergence of massive parallelism and dynamic reconfiguration in inevitable: we believe it is one of the main challenges in computing for the current decade.

By incorporating the configuration and/or data/program memory on the top of the FPGA fabric, with fast and numerous connections between memory and elementary logic blocks (∼10000 connections between dies), it will be possible to obtain dynamically reconfigurable computing platforms with a very high reconfiguration rate. Such a rate was not possible before, due to the serial nature of the interface between the configuration memory and the FPGA fabric itself. The FPGA technology also enables massively parallel architectures due to the large number of programmable logic fabrics available on the chip. For instance, Xilinx demonstrated 3600 8-bit picoBlaze softcore processors running simultaneously on the Virtex-7 2000T FPGA. For specific applications, picoBlaze can be replaced by specialized hardware accelerators or other IPs (Intellectual Property) components. This opens the possibility of creating massively parallel IP-based machines.

3.2. Multi-softcore on 3D FPGA

From the 2010 Xilinx white paper on FPGAs:

> *"Unlike a processor, in which architecture of the ALU is fixed and designed in a general-purpose manner to execute various operations, the CLBs (configurable logic blocks) can be programmed with just the operations needed by the application... The FPGA architecture provides the flexibility to create a massive array of application-specific ALUs..The new solution enables high-bandwidth connectivity between multiple die by providing a much greater number of connections... enabling the integration of massive quantities of interconnect logic resources within a single package"*

Softcore processors are processors implemented using hardware synthesis. Proprietary solutions include PicoBlaze, MicroBlaze, Nios, and Nios II; open-source solutions include Leon, OpenRisk, and FC16. The choice is wide and many new solutions emerge, including multi-softcore implementations on FPGAs. An alternative to softcores are hardware accelerators on FPGAs, which are dedicated circuits that are an order of magnitude faster than softcores. Between these two approaches, there are other various approaches that connect IPs to softcores, in which, the processor's machine-code language is extended, and IP invocations become new instructions. We envisage a new class of softcores (we call them reflective softcores [1]), where almost everything is implemented in IPs; only the control flow is assigned to the softcore itself. The partial dynamic reconfiguration of next-generation FPGAs makes such dynamic IP management possible in practice. We believe that efficient reflective softcores on the new 3D-FPGAs should be as small as possible: low-performance generic hardware components (ALU, registers, memory, I/O...) should be replaced by dedicated high-performance IPs.

We are developing a sofcore processor called HoMade (http://www.lifl.fr/~dekeyser/Homade) following these ideas.

In the multi-reflective softcores that we develop, some softcores will be slaves and others will be masters. Massively parallel dynamically reconfigurable architectures of softcores can thus be envisaged. This requires, additionally, a parallel management of the partial dynamic reconfiguration system. This can be done, for example, on a given subset of softcores: a massively parallel reconfiguration will replace the current replication of a given IP with the replication of a new IP. Thanks to the new 3D-FPGAs this task can be performed efficiently and in parallel using the large number of 3D communication links (Through-Silicon-Vias). Our roadmap for HoMade is to evolve towards this multi-reflective softcore model.

3.3. When Hardware Meets Software

HIPEAC vision 2011/12: *"The number of cores and instruction set extensions increases with every new generation, requiring changes in the software to effectively exploit the new features."*

[1] Hereafter, by reflective system, we mean a system that is able to modify its own structure and behaviour while it is running. A reflective softcore thus dynamically adds, removes, and replaces IPs in the application running on it, and is able to dynamically modify its own program memory, thereby dynamically altering the program it is executing.

When the new massively parallel dynamically reconfigurable architectures become reality users will need languages for programming software applications on them. The languages will be themselves dynamic and parallel, in order to reflect and to fully exploit the dynamicity and parallelism of the architectures. Thus, developers will be able to invoke reconfiguration and call parallel instructions in their programs. This expressiveness comes with a cost, however, because new classes of bugs can be induced by the interaction between dynamic reconfiguration and parallelism; for example, deadlocks due to waiting for output from an IP that does not exist any more due to a reconfiguration. The detection and elimination of such bugs before deployment is paramount for cost-effectiveness and safety reasons.

Thus, we shall build an environment for developing software on parallel, dynamically reconfigurable architectures that will include languages and adequate formal analyses and verification tools for them, in addition to more traditional tools (emulators, compilers, etc). To this end we shall be using formal-semantics frameworks associated with easy-to-use formal verification tools in order to formally define our languages of interest and allow users to formally verify their programs. The K semantic framework (http://k-framework.org), developed jointly by Univs. Urbana Champaign, USA, and Iasi, Romania) is one such framework, which is mature enough (it has allowed defining a formal semantics of the largest subset of the C language to date, as well as many other languages from essentially all programming paradigms) and is familiar to us from previous work. In K, one can rapidly prototype a language definition and try several versions of the syntax and semantics of instructions. This is important in our project, where the proposed programming languages (in particular, the HoMade assembly language) will go through several versions before being stabilized. Moreover, once a language is defined in K one gets an interpreter of the language and one gains access to formal verification tools for free. We are also developing new analysis verification tools for K (in collaboration with the K team), which will be adapted and used in the Dreampal project.

4. Highlights of the Year

4.1. Highlights of the Year

2016 is the last year of Dreampal's existence as an Inria project-team. Due to different scientific objectives, three of the members (S. Meftali, J.L. Dekeyser, P. Marquet) will create a group within the Cristal laboratory, while the team leader V. Rusu will collaborate with the 2xs team within Cristal. Frédéric Guyomarch joined the L2EP laboratory, and external collaborator Rabie Ben Atitallah continues his activity in the LAMIH laboratory in Valenciennes.

This activity report has been written by the team leader, based on the information available to him at the time of its writing. Any activity, e.g., by other team members, not reflected in the report, is only missing because of lack of input from the people concerned.

5. New Software and Platforms

5.1. HoMade

KEYWORDS: SoC - Multicore - Softcore
FUNCTIONAL DESCRIPTION

HoMade is a softcore processor. The current version is reflective (i.e., the program it executes is self-modifiable), and statically configurable, dynamically reconfigurable multi-processors are the next steps. Users have to add to it the functionality they need in their applications via IPs. We have also being developing a library of IPs for the most common processor functions (ALU, registers, ...). All the design is in VHDL except for some schematic specifications.

- Participant: Jean Luc Dekeyser
- Partner: LIFL
- Contact: Jean Luc Dekeyser
- URL: https://sites.google.com/site/homadeguideen/home

5.2. JHomade

FUNCTIONAL DESCRIPTION

JHomade is a software suite written in JAVA, including compilers and tools for the HoMade processor. It allows us to compile HiHope programs to Homade machine code and load the resulting binaries on FPGA boards. It was first released in 2013. The second version in 2014 includes several new features, like a C-frontend, a binary decoder and a code-generator for VHDL simulation. New features of the HiHope language are described in more detail in Section.

- Contact: Vlad Rusu
- URL: https://gforge.inria.fr/frs/?group_id=3646

6. New Results

6.1. A Language-Independent Proof System for Full Program Equivalence

Two programs are mutually equivalent if, for the same input, either they both diverge or they both terminate with the same result. Mutual equivalence is an adequate notion of equivalence for programs written in deterministic languages. It is useful in many contexts, such as capturing the correctness of program transformations within the same language, or capturing the correctness of compilers between two different languages. In [11] we introduce a language-independent proof system for mutual equivalence, which is para-metric in the operational semantics of two languages and in a state-similarity relation. The proof system is sound: if it terminates then it establishes the mutual equivalence of the programs given to it as input. We illustrate it on two programs in two different languages (an imperative one and a functional one), that both compute the Collatz sequence. The Collatz sequence is an interesting case study since it is not known wether the sequence terminates or not; nevertheless, our proof system shows that the two programs are mutually equivalent (even if we cannot establish termination or divergence of either one).

6.2. A Generic Framework for Symbolic Execution: a Coinductive Approach

In [12] we propose a language-independent symbolic execution framework. The approach is parameterised by a language definition, which consists of a signature for the lan-guage's syntax and execution infrastructure, a model interpreting the signature, and rewrite rules for the language's operational semantics. Then, symbolic execution amounts to computing symbolic paths using a derivative operation. We prove that the symbolic execution thus defined has the properties naturally expected from it, meaning that the feasible symbolic executions of a program and the concrete executions of the same program mutually simulate each other. We also show how a coinduction-based extension of symbolic execution can be used for the deductive verification of programs. We show how the proposed symbolic-execution approach, and the coinductive verification technique based on it, can be seamlessly implemented in language definition frameworks based on rewriting such as the K framework. A prototype implementation of our approach has been developed in K. We illustrate it on the symbolic analysis and deductive verification of nontrivial programs.

6.3. Circuit Merging versus Dynamic Partial Reconfiguration -The HoMade Implementation

One goal of reconfiguration is to save power and occupied resources. In [13] we compare two different kinds of reconfiguration available on field-programmable gate arrays (FPGA) and we discuss their pros and cons. The first method that we study is circuit merging. This type of reconfiguration methods consists in sharing common resources between different circuits. The second method that we explore is dynamic partial reconfiguration (DPR). It is specific to some FPGA, allowing well defined reconfigurable parts to be modified during run-time. We show that DPR, when available, has good and more predictable result in terms of occupied area. There is still a huge overhead in term of time and power consumption during the reconfiguration phase.

Therefore we show that circuit merging remains an interesting solution on FPGA because it is not vendor specific and the reconfiguration time is around a clock cycle. Besides, good merging algorithms exist even-though FPGA physical synthesis flow makes it hard to predict the real performance of the merged circuit during the optimization. We establish our comparison in the context of the HoMade processo

6.4. Language Definitions as Rewrite Theories

K is a formal framework for defining operational semantics of programming languages. The K-Maude compiler translates K language definitions to Maude rewrite theories. The compiler enables program execution by using the Maude rewrite engine with the compiled definitions, and program analysis by using various Maude analysis tools. K supports symbolic execution in Maude by means of an automatic transformation of language definitions. The transformed definition is called the symbolic extension of the original definnition. In [14] we investigate the theoretical relationship between K language definitions and their Maude translations, between symbolic extensions of K definitions and their Maude translations, and how the relationship between K definitions and their symbolic extensions is reflected on their respective representations in Maude. In particular, the results show how analysis performed with Maude tools can be formally lifted up to the original language definitions.

6.5. SCAC-Net: Reconfigurable Interconnection Network in SCAC Massively parallel SoC

Parallel communication plays a critical role in massively parallel systems, especially in distributed memory systems executing parallel programs on shared data. Therefore, integrating an interconnection network in these systems becomes essential to ensure data inter-nodes exchange. Choosing the most effective communication structure must meet certain criteria: speed, size and power consumption. Indeed, the communication phase should be as fast as possible to avoid compromising parallel computing, using small and low power consumption modules to facilitate the interconnection network extensibility in a scalable system. To meet these criteria and based on a module reuse methodology, we chose to integrate a reconfigurable SCAC-Net interconnection network to communicate data in SCAC Massively parallel SoC. In [15] we present the detailed hardware implementation and discusses the performance evaluation of the proposed reconfigurable SCAC-Net network.

6.6. Proving Reachability-Logic Formulas Incrementally

Reachability Logic (RL) is a formalism for defining the operational semantics of programming languages and for specifying program properties. As a program logic it can be seen as a language-independent alternative to Hoare Logics. Several verification techniques have been proposed for RL, all of which have a circular nature: the RL formula under proof can circularly be used as a hypothesis in the proof of another RL formula, or even in its own proof. This feature is essential for dealing with possibly unbounded repetitive behaviour (e.g., program loops). The downside of such approaches is that the verification of a set of RL formulas is monolithic, i.e., either all formulas in the set are proved valid, or nothing can be inferred about any of the formula's validity or invalidity. In [16] we propose a new, incremental method for proving a large class of RL formulas. The proposed method takes as input a given RL formula under proof (corresponding to a given program fragment), together with a (possibly empty) set of other valid RL formulas (e.g., already proved using our method), which specify sub-programs of the program fragment under verification. It then checks certain conditions are shown to be equivalent to the validity of the RL formula under proof. A newly proved formula can then be incrementally used in the proof of other RL formulas, corresponding to larger program fragments. The process is repeated until the whole program is proved. We illustrate our approach by verifying the nontrivial Knuth-Morris-Pratt string-matching program.

7. Partnerships and Cooperations

7.1. International Initiatives

7.1.1. Inria International Partners

7.1.1.1. Informal International Partners

In 2016 we have continued our strong and long-term collaboration with Prof. Dorel Lucanu's group Univ. Iasi as witnessed by the co-authored publications (3 journals and 1 conference). Vlad Rusu serves as "external advisor for PhD students" in Prof. Lucanu's group. In 2016 we have also had notable interactions with Prof. José Meseguer (Univ. Illinois at Urbana Champaign, USA), which consisted in sharing ideas and mutual reading and commenting advanced drafts prior to submission in journals/conferences.

8. Dissemination

8.1. Promoting Scientific Activities

8.1.1. Scientific Events Selection

8.1.1.1. Member of the Conference Program Committees

V.Rusu was a member in the PC of the 2016 edition of the Int. Workshop on Rewriting Logic and Applications, and will be the organizer of the next edition of the event. He was was also PC member of Approches Formelles pour la Validation Logicielle (AFADL'2016).

8.1.2. Research Administration

Vlad Rusu is elected member at Inria's Evaluation Committee. As such he has been involved in several activities regarding promotions of researchers, team creations, and team evaluations.

8.2. Teaching - Supervision - Juries

8.2.1. Teaching

Licence : V.Rusu, Logic, 30hrs, L3, Univ. Lille, France.

Doctorat : V. Rusu, External adviser for PhD students, Univ. Iasi, Romania.

9. Bibliography

Major publications by the team in recent years

[1] A. AIT EL CADI, O. SOUISSI, R. BEN ATITALLAH, N. BELANGER, A. ARTIBA. *Mathematical programming models for scheduling in a CPU/FPGA architecture with heterogeneous communication delays*, in "Journal of Intelligent Manufacturing", April 2015, pp. 1-12 [*DOI :* 10.1007/s10845-015-1075-z], https://hal.inria.fr/hal-01247399

[2] K. M. A. ALI, R. B. ATITALLAH, N. FAKHFAKH, J.-L. DEKEYSER. *Using hardware parallelism for reducing power consumption in video streaming applications*, in "10th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2015", Bremen, Germany, June 2015 [*DOI :* 10.1109/ReCoSoC.2015.7238104], https://hal.inria.fr/hal-01247130

[3] M. BAKLOUTI, P. MARQUET, J.-L. DEKEYSER, M. ABID. *FPGA-based many-core System-on-Chip design*, in "Microprocessors and Microsystems", 2015, 38 p. [*DOI :* 10.1016/J.MICPRO.2015.03.007], https://hal.inria.fr/hal-01144977

[4] S. CIOBACA, D. LUCANU, V. RUSU, G. ROSU. *A Language-Independent Proof System for Full Program Equivalence*, in "Formal Aspects of Computing", 2016, vol. 28, n° 3, pp. 469–497 [*DOI :* 10.1007/S00165-016-0361-7], https://hal.inria.fr/hal-01245528

[5] D. LUCANU, V. RUSU, A. ARUSOAIE. *A Generic Framework for Symbolic Execution: a Coinductive Approach*, in "Journal of Symbolic Computation", 2016, Accepted for publication [*DOI :* 10.1016/J.JSC.2016.07.012], https://hal.inria.fr/hal-01238696

[6] D. LUCANU, V. RUSU, A. ARUSOAIE, D. NOWAK. *Verifying Reachability-Logic Properties on Rewriting-Logic Specifications*, in "Logic, Rewriting, and Concurrency - Festschrift Symposium in Honor of José Meseguer", Urbana Champaign, United States, Logic, Rewriting, and Concurrency Essays Dedicated to José Meseguer on the Occasion of His 65th Birthday, Springer Verlag, September 2015, vol. 9200 [*DOI :* 10.1007/978-3-319-23165-5_21], https://hal.inria.fr/hal-01158941

[7] D. LUCANU, V. RUSU. *Program Equivalence by Circular Reasoning*, in "Formal Aspects of Computing", 2015, vol. 27, n° 4, pp. 701-726 [*DOI :* 10.1007/S00165-014-0319-6], https://hal.inria.fr/hal-01065830

[8] V. RUSU, D. LUCANU, T.-F. ŞERBĂNUŢĂ, A. ARUSOAIE, A. ŞTEFĂNESCU, G. ROŞU. *Language Definitions as Rewrite Theories*, in "Journal of Logic and Algebraic Methods in Programming", 2016, vol. 85, n° 1, pp. 98–120 [*DOI :* 10.1016/J.JLAMP.2015.09.001], https://hal.inria.fr/hal-01186005

[9] V. VISWANATHAN, R. B. ATITALLAH, J.-L. DEKEYSER. *Massively Parallel Dynamically Reconfigurable Multi-FPGA Computing System* , in "IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2015", Vancouver, BC , Canada, May 2015 [*DOI :* 10.1109/FCCM.2015.13], https://hal.inria.fr/hal-01247116

[10] V. VISWANATHAN, R. BEN ATITALLAH, J.-L. DEKEYSER, B. NAKACHE, M. NAKACHE. *A Parallel And Scalable Multi-FPGA based Architecture for High Performance Applications*, in "The 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA '15", Monterey, California, United States, February 2015 [*DOI :* 10.1145/2684746.2689115], https://hal.inria.fr/hal-01247133

Publications of the year

Articles in International Peer-Reviewed Journals

[11] S. CIOBACA, D. LUCANU, V. RUSU, G. ROSU. *A Language-Independent Proof System for Full Program Equivalence*, in "Formal Aspects of Computing", 2016, vol. 28, n° 3, pp. 469–497 [*DOI :* 10.1007/S00165-016-0361-7], https://hal.inria.fr/hal-01245528

[12] D. LUCANU, V. RUSU, A. ARUSOAIE. *A Generic Framework for Symbolic Execution: a Coinductive Approach*, in "Journal of Symbolic Computation", 2016, Accepted for publication [*DOI :* 10.1016/J.JSC.2016.07.012], https://hal.inria.fr/hal-01238696

[13] J. PERIER, W. CHOUCHENE, J.-L. DEKEYSER. *Circuit Merging versus Dynamic Partial Reconfiguration -The HoMade Implementation*, in "i-manager's Journal on Embedded Systems(JES)", 2016, https://hal.inria.fr/hal-01245800

[14] V. RUSU, D. LUCANU, T.-F. ŞERBĂNUŢĂ, A. ARUSOAIE, A. ŞTEFĂNESCU, G. ROŞU. *Language Definitions as Rewrite Theories*, in "Journal of Logic and Algebraic Methods in Programming", 2016, vol. 85, n^{o} 1, pp. 98–120 [*DOI :* 10.1016/J.JLAMP.2015.09.001], https://hal.inria.fr/hal-01186005

International Conferences with Proceedings

[15] H. KRICHENE, M. BAKLOUTI, M. ABID, P. MARQUET, J.-L. DEKEYSER, S. MEFTALI. *SCAC-Net: Reconfigurable Interconnection Network in SCAC Massively parallel SoC*, in "The 24th Euromicro International Conference on Parallel, Distributed and Network-Based Processing", Heraklion, Greece, February 2016, https://hal.inria.fr/hal-01247298

[16] V. RUSU, A. ARUSOAIE. *Proving Reachability-Logic Formulas Incrementally*, in "11th International Workshop on Rewriting Logic and its Applications", Eindhoven, Netherlands, April 2016, forthcoming, https://hal.inria.fr/hal-01282379