IN PARTNERSHIP WITH:
CNRS

Université des sciences et technologies de Lille (Lille 1)

Activity Report 2014

Team DREAMPAL

Dynamic Reconfigurable Massively Parallel Architectures and Languages

RESEARCH CENTER
Lille - Nord Europe

THEME
Architecture, Languages and Compilation

Table of contents

Team DREAMPAL

Keywords: Embedded Systems, Reconfigurable Hardware, Programming Languages, Formal Methods

Creation of the Team: 2013 January 01*, updated into Project-Team:* 2015 January 01.

1. Members

Research Scientist
Vlad Rusu [Team leader, Inria, Researcher, HdR]

Faculty Members
Ahmad Aljendi [Univ. Lille I]
Rabie Ben Atitallah [Univ. Valenciennes, Associate Professor, HdR]
Sana Cherif [Univ. Lille III, until Aug 2014]
Jean-Luc Dekeyser [Univ. Lille I, Professor, HdR]
Frédéric Guyomarch [Univ. Lille I, Associate Professor]
Philippe Marquet [Univ. Lille I, Associate Professor]
Samy Meftali [Univ. Lille I, Associate Professor, HdR]
Chiraz Trabelsi [Univ. Valenciennes, until Aug 2014]

PhD Students
Karim Ali [Univ. Valenciennes]
Wissem Chouchene [Univ. Lille I]
Hana Krichene [Univ. Lille I]
Vlad Craciun [Univ. Iasi, Romania, since Sept 2014]
Venkatasubramanian Viswanathan [Univ Valenciennes & Nolam Industries]

Post-Doctoral Fellow
Andrei Arusoaie [Inria, since Nov 2014]

Visiting Scientists
Majdi Elhajji [Univ. Monastir (Tunisie), May 2014]
Dorel Lucanu [Univ. Iasi, (Romania), Professor, Jul-Aug 2014]

Administrative Assistant
Corinne Jamroz [Inria]

Other
Benjamin Danglot [Inria, Intern, from Apr 2014 until Jul 2014]

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: http://www.lifl.fr/west/team.html.
- 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. New Software and Platforms

4.1. New Software and Platforms

Download page : https://gforge.inria.fr/frs/?group_id=3646

4.1.1. HoMade

HoMade is a softcore processor that we have started developing in 2012. The current version is reflective (i.e., the program it executes is self-modifiable), and statically configurable; dynamically reconfigurable multiprocessors 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.

The V5 version of HoMade has been developed in the Spring 2014. It has been used by ∼140 4th-year computer science students at Univ. Lille enrolled in the hardware architecture course (https://sites.google.com/site/tpm1aev/home). The new features of V5 are listed in Section 5.2.

4.1.2. JHomade

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 5.3.

4.1.3. Kcheck

Kcheck is a tool for the symbolic execution of programs in arbitrary languages defined in the $\mathbb{K}$ framework (http://k-framework.org), such as C and Java as well as the languages HiHope and Homade machine-code languages developed in out team. It also allow users to formally verify programs against specifications written in Reachability Logic, a specification formalism that can be seen as a language-independent Hoare logic. More information about the theory underlying Kcheck is given in Section 5.5.

In 2014 we have developed a new and improved version of our tool, in order to keep up with the new modular infrastructure of the $\mathbb{K}$ framework. An online interface has been developed and is available at https://fmse.info.uaic.ro/tools/kcheck/. We have also started (since Nov. 2014) a development in the Coq proof assistant in order to obtained certificates for the program verifications performed by our tool.

5. New Results

5.1. Highlights of the Year

The papers [4] and [6] are published in journals (Software Testing Verification and Analisys, resp. Formal Aspects of Computing) that are among the best in their respective fields.

5.2. HoMade

HOMADE V5 is available from 03/2014. New features cover :

- new pipeline architecture with delayed conditional branch
- new unified FSM: Pipeline 2 stages
- renumbering of some IPs
- new activity management on the Slaves in 1D / 2D : by the master OnX , OnY, OnXY, and by the slaves the IPsleep removes the Slave from the next SPMDcall
- new bit per bit loading of program memories, for master and slaves
- new names for some components.
- new versions of a lot of IPs (inside)
- new communication network between Slaves: 2D torus ring with broadcast and communication on x or y axis
- new input binary file format (to respect !!)
- new test_bench for fast reading of instruction files
- new UART wrapper
- new assembler Hasm for those that do not speak binary
- nexys3 version for cheap platform experimentation (does not support more than 2x1 Slaves)
- V6 V7 xilinx supports up to 12 x 12 slaves
- Isim supports many more slaves !!!

More details can be found on www.lifl.fr/~dekeyser/Homade.

5.3. HiHope : A higher level language for the HoMade processor

HiHope is a programming language inspired by Forth used to program the HoMade processor. It includes language constructs for switching at runtime between hardware functions (implemented by IPs) and software functions in a transparent way. We also propose the notion of parallel function language construct. As a result, HiHope programs can use either hardware IPs or software functions, and can perform both sequential and parallel function calls, as well as sequential and parallel function redefinitions.

5.4. Integrating Profiling into MDE Compilers

This work [3] aims at improving performance by returning to the high-level models, specific execution data from a profiling tool enhanced by smart advices computed by an analysis engine. In order to keep the link between execution and model, the process is based on a traceability mechanism. Once the model is automatically annotated, it can be re-factored aiming better performances on the re-generated code. Hence, this work allows keeping coherence between model and code without forgetting to harness the power of parallel architectures. The example uses a transformation chain from UML-MARTE models to OpenCL code.

5.5. Language-Independent Symbolic Execution, Program Equivalence, and Program Verification

A significant part of our research project consists in applying formal techniques for symbolically executing and formally verifying HiHope programs, as well as for formally proving the equivalence of HiHope programs with the corresponding HoMade assembly and machine-code programs obtained by compilation of HiHope.

- Symbolic execution will detect bugs (e.g., stack undeflow) in HiHope programs. Additionaly, symbolic execution is the natural execution manner of HiHope programs as soon as they contain (typically, underspecified) hardware IPs;
- program verification will guarantee the absence of bugs (with respect to specified properties, e.g., no stack underflow, no invocation of unavailable IPs, ...);
- program equivalence will guarantee that such above-mentioned bugs are also absent from the HoMade assembly and machine-code programs obtained by compilation of HiHope source code.

Since these languages are still evolving we decided to work (together with our colleagues from Univ. Iasi, Romania) on language-independent symbolic execution, program-equivalence, and program-verification techniques. In this way, when all the languages in our project become stable, we will be readily able to instantiate the above generic techniques on (the K formal definitions of) the languages in question. We note that all the techniques described below are also independent of K: they are applicable to other language-definition frameworks that use similar rewriting-based formal operational semantics.

5.5.1. Symbolic Execution

In [15] we propose a language-independent symbolic execution framework. The approach is parameterised by a language definition, which consists of a signature for the language's syntax and execution infrastructure, a model interpreting the signature, and rewrite rules for the language's operational semantics. Then, symbolic execution amounts to performing a so-called symbolic rewriting, which consists in changing both the model and the manner in which the operational semantics rules are applied. We prove that the symbolic execution thus defined has the properties naturally expected from it. A prototype implementation of our approach was developed in the K Framework. We demonstrate the genericity of our tool by instantiating it on several languages, and show how it can be used for the symbolic execution, bounded model checking, and deductive verification of several programs. With respect to earlier versions of this work, we have redefined symbolic execution in a more generic way and have included applications to model checking and deductive verification. The current version of the report [15] is submitted to a journal and is based on Andrai Arusoaie's PhD thesis [1], defended in September 2014 at Univ. Iasi (Romania). Andrei was co-supervised by Vlad Rusu and has since joined Dreampal as a postdoc.

5.5.2. Program Equivalences

In [6] we propose a logic and a deductive system for stating and automatically proving the equivalence of programs written in languages having a rewriting-based operational semantics. The chosen equivalence is parametric in a so-called observation relation, and it says that two programs satisfying the observation relation will inevitably be, in the future, in the observation relation again. This notion of equivalence generalises several well-known equivalences and is appropriate for deterministic (or, at least, for confluent) programs. The deductive system is circular in nature and is proved sound and weakly complete; together, these results say that, when it terminates, our system correctly solves the given program-equivalence problem. We show that our approach is suitable for proving equivalence for terminating and non-terminating programs as well as for concrete and symbolic programs. The latter are programs in which some statements or expressions are symbolic variables. By proving the equivalence between symbolic programs, one proves the equivalence of (infinitely) many concrete programs obtained by replacing the variables by concrete statements or expressions. The approach is illustrated by proving program equivalence in two languages from different programming paradigms. The examples in the paper, as well as other examples, can be checked using an online tool. This work was started in 2012. With respect to earlier versions, the new journal publication [6] includes a new and more general presentation of program equivalence as a temporal-logic formula, the generalisation of the

approach to nondeterministic-confluent language semantics, substantially more compact proofs, and a new application to corecursive programs.

In another work [10] we deal with a different kind of equivalence: *mutual equivalence*, which says that two programs are mutually equivalent if they both diverge or they end up in similar states. 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 the case of different languages one needs an operation called *language aggregation*, which we present in [11] in more detail, that combine two languages into a single one. We introduce a language-independent proof system for mutual equivalence, which is parametric 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.

5.5.3. Program Verification

In [16] we present an automatic, language-independent program verification approach and prototype tool based on symbolic execution. The program-specification formalism we consider is Reachability Logic, a language-independent alternative to Hoare logics. Reachability Logic has a sound and relatively complete deduction system that offers a lot of freedom to the user regarding the manner and order of rule application, but it lacks a strategy for automatic proof construction. Hence, we propose a procedure for proof construction, in which symbolic execution plays a major role. We prove that, under reasonable conditions on its inputs (the operational semantics of a programming language, and a specification of a program, both given as sets of Reachability Logic formulas) our procedure is partially correct: if it terminates it correctly answers (positively or negatively) to the question of whether the given program specification holds when executing the program according to the given semantics. Termination, of course, cannot be guaranteed, since program-verification is an undecidable problem; but it does happen if the provided set of goals includes enough information in order to be circularly provable (using each other as hypotheses). We introduce a prototype program-verification tool implementing our procedure in the K language-definition framework, and illustrate it by verifying nontrivial programs written in languages defined in K. With respect to earlier versions of this work from 2013, program verification is now presented as a procedure (instead of a proof system), which leads to a direct implementation in the new version of our prototype tool. We also have a new theoretical result: *weak completeness*, which says that a negative answers returned by the verification procedure imply the fact that that the program does not meet its specification. Finally, since Andrei Arusoaie's arrival in the Dreampal team as a postdoc (Nov 2014) we have started working on certifying our verification procedure in the Coq proof assistant.

5.5.4. Language Definitions as Rewrite Theories

In [8] we study the relationships between language definition frameworks (e.g., the K framework) and rewrite theories (e.g., as those embodied in the Maude tool). K is a formal framework for defining the operational semantics of programming languages. It includes software tools for compiling K language definitions to Maude rewrite theories, for executing programs in the defined languages based on the Maude rewriting engine, and for analyzing programs by adapting various Maude analysis tools. A recent extension to the K tool suite is an automatic transformation of language definitions that enables the symbolic execution of programs, i.e., the execution of programs with symbolic inputs. In this paper we investigate more particularly the theoretical relationships between K language definitions and their translations to Maude, between symbolic extensions of K definitions and their Maude encodings, and how the relations between K definitions and their symbolic extensions are reflected on their respective representations in Maude. These results show, in particular, how analyses performed with Maude tools can be formally lifted up to the original language definitions. The results presented in this paper provide the theoretical underpinnings for the current version of the K-Maude tool.

5.6. Hardware chain for partial reconfiguration

The cost overhead due to the use of a softcore processor (MicroBlaze) to drive dynamic reconfiguration led us to explore alternative solutions. The one we have adopted is the use of a dedicated hardware IP (that can be invoked by HoMade) to control and manage dynamic and partial reconfiguration. This approach has led us to develop a complete hardware chain for partial bitstreams reads and writes. The proposed architecture is based on an external memory controller (DDR3) whose role is to manage bitstreams transfers from and to the DDR. Bitstreams loading are managed by a HoMade instruction implemented in a dedicated IP that drives the ICAP interface to transfer data into the reconfigurable area through the physical ICAP. One of the most important performance criteria of dynamic and partial reconfiguration is the reconfiguration time, that we always try to reduce while taking into account the compromise cost / area, speed and power consumption. Preliminary results give a transfer rate exceeding 500 MB/s. Such a result is clearly promising, especially since our hardware reconfiguration chain is constructed to be easily adaptable to SPMD (multi HoMade) needing parallel partial reconfiguration. This work has been the subject of a first communication in the GDR / SOCSIP conference in Paris: 11, 12, 13 June 2014.

5.7. Generic pixel distribution for parallel video processing application

In the frame of the PhD thesis of Karim Ali, we exploited this year the usage of parallel architectures for real-time image/video processing applications. Our main concern was the data distribution according to the parallelism level and respecting real-time processing constraint. As a first step, we proposed a generic pixel distribution model to be used with different image/video applications. Several parameters in the model can be configured according to the required size of the distributed macro-block with the possibility to control the sliding step in both horizontal and vertical directions. We have implemented our architecture on the Xilinx Zynq ZC706 FPGA evaluation board for two applications: the video downscaler (1:16) and the convolution filter. The experimental results showed the low hardware cost of the solution and how flexible is the model to be configured for different distribution scenarios. The architecture and experimental results were published in a paper entitled "A Generic Pixel Distribution Architecture for Parallel Video Processing" at Reconfigurable computing and FPGA international conference (ReConFig) in December 2014, Cancun, Mexico [7].

As a next step, we will reduce the operating clock frequency to decrease the power consumption while increasing the number of processing elements in the parallel architecture to maintain the same performance results. In this way, we will obtain a set of different design points differ in (area, power, other factors) and the system will have the ability to adapt its structure by moving between different design points according to the available resources to keep the same performance measurements. Furthermore, we will target intelligent transportation system, specially dynamic obstacle detection and tracking for autonomous vehicle navigation in collaboration with NAVYA (http://navya-technology.com).

5.8. Massively Parallel Dynamically Reconfigurable Multi-FPGA

In the frame of the PhD thesis of Venkatasubramanian Viswanathan, we conceived and validated a massively parallel and dynamically reconfigurable execution model for next generation high performance embedded systems. We have designed a multi-FPGA platform in order to conceive the massively parallel dynamically reconfigurable execution model. We have used several IP cores developed during the first two years of my PhD in order to test and validate the proposed model. We have proposed a new parallel dynamic reconfiguration mechanism for our architecture. We use our parallel reconfiguration model to reconfigure a subset or several IPs in parallel. We have proposed a partial reconfiguration model for next generation 3D FPGAs well-traced on the execution model (SPMD) in order to reconfigure in parallel a subset of the computing nodes. Finally, we have used the PicoComputing platform as an example to validate our proposed execution and reconfiguration models.

In order to demonstrate various features of such an architecture, we have implemented a scalable distributed secure H.264 encoding application with a FMC based high-speed sFPDP (serial Front Panel Data Port) data acquisition protocol to capture RAW video data. The system has been implemented on 3 different FPGAs, respecting the SPMD execution model managing several input video sources in parallel. We have measured various performance metrics of the proposed massively parallel dynamically reconfigurable system

and demonstrated several benefits. This work is going to be published in the FPGA 2015 conference as a poster titled "A Parallel And Scalable Multi-FPGA based Architecture for High Performance Applications" [13].

Later an ICAP controller was setup for dynamic partial reconfiguration in order to swap IPs during runtime on a single FPGA. We have used this IP along with the parallel communication feature of the multi-FPGA architecture, in order to broadcast a partial bitstream to all FPGAs at the same time and to do a parallel DPR in several FPGAs, thus emulating the reconfiguration model for next generation 3D FPGAs. These results represent a conceptual proof for a massively parallel dynamically reconfigurable next generation embedded computers that will use 3D PFGAs and reconfigure several logic layers in parallel.

5.9. HoMade-based MPSoC

The goal of this work is to build an MPSoC based on HoMade. The aimed system is a completely dynamically reconfigurable system. This mean that both the processing elements (HoMade) and the interconnection network are dynamically reconfigurable. The basic block in this system developped here is the interconnection network. It is a MIN (Multistage Interconnection Network) that would utilize oversizing techniques in order to reconfigure the network depending on the traffic.

5.10. Communication-Computation Overlap in Massively Parallel System-on-Chip

The Synchronous Communication Asynchronous Computation (SCAC) model is an execution model dedicated to the Massively Parallel System-on-Chip. This model proposes a novel processing paradigm, teh communication-computation overlap [17]. This concept does not only consider the programming level but also the implementation level. Using a decoupled control structure, the synchronous communication control is performed independently of the asynchronous computation control. Separating these two control phases allows the programmer to define programming strategies that overlap communication by computation to decrease the execution time.

To achieve this communication-computation overlap in SCAC architecture while avoiding the centralized control, in addition to the master controller, we define a second hierarchical control level, namely the slave controllers. The concept of this dual ontrol structure departs from the centralized configuration and instead of a uni-processor master controlling a set of parallel Processing Elements (PEs), the master cooperates with a grid of parallel slave controllers which supervises the activities of cluster of PEs. Based on this decoupled control structure, the programmer can manage the master-slave program to overlap communication by computation phase. Therefore, the basic idea to implement this paradigm is to divide the principal program into small blocks of parallel instructions, called Slave Program (SP), and send these blocks to the activated PEs of the system. Then, according to a predefined mask, the slave controllers send the begin execution orders. In parallel to computation, the slave controllers manage the synchronous inter-node communication. Distinguish communication from computation needs the separation of these two phases in different blocks. This repartition should be provided at programming level. Then, the overlapped execution of these blocks will be done in parallel according to the program description.

The aim of these last works is to define a new paradigm of a communication-computation overlap in massively parallel System-on-Chip. This paradigm allows to decrease the execution time of parallel programs using specific strategies in the programming level and a partially decoupled control system in the hardware level. The difficulty of implementing this paradigm lies in the coordination between the programming level and the architecture designing level in order to hide the communication cost.

6. Bilateral Contracts and Grants with Industry

6.1. Bilateral Contracts with Industry

Collaboration contract with Nolam Embedded Systems: In conjunction with the CIFRE grant of Venkatasubramanian Viswanathan, a collaboration contract is established with Nolam ES. The objective is to design an innovative embedded computing platform supporting massively parallel dynamically reconfigurable execution model. The use-cases of this platform cover several application domains such as medical, transportation and aerospace.

Collaboration contract with NAVYA: In conjunction with the doctoral grant of Karim Ali, a collaboration contract is established with NAVYA. The objective is to design an innovative embedded system dedicated for dynamic obstacle detection and tracking for autonomous vehicle navigation.

7. Partnerships and Cooperations

7.1. Regional Initiatives

The CPER has financed the visit of Prof. Dorel Lucanu from Univ. Iasi (Romania) in July and August 2014.

7.2. International Initiatives

7.2.1. Participation In other International Programs

Wissem Chouchene is financed by the Euramus Mondus programme.

7.3. International Research Visitors

7.3.1. Visits of International Scientists

Prof. Dorel Lucanu from Univ. Iasi (Romania) visited us in July and August 2014. We continued work on language-independent program-verification techniques and on the formal definitions of the HiHope and HoMade assembler languages, as well as on the formally proved correctness of communication IPs.

8. Dissemination

8.1. Promoting Scientific Activities

8.1.1. Scientific events organisation

8.1.1.1. member of the organizing committee

F. Guyomarch, P. Marquet and S. Meftali are members of the ComPAS organizing committee. This conference will take place in Lille in 2015.

F. Guyomarch and S. Meftali are members of the Archi organizing committee. This summer school will take place in Lille in 2015.

R. Ben Atitallah is a member of the Green Days@Rennes organizing committee. The scientific event was held in Rennes, July 2014.

8.1.2. Scientific events selection

8.1.2.1. responsable of the conference program committee

S. Meftali is responsable of the ComPAS conference program committee. This conference will take place in Lille in 2015.

8.1.2.2. member of the conference program committee

F. Guyomarch is a member of the ComPAS program committee.

V. Rusu is a member of the WRLA2014 program committee.

8.2. Teaching - Supervision - Juries

8.2.1. Teaching

Licence : Philippe Marquet, Introduction to Computer Science, 15h, Secondary Education Teatcher Training, Université Lille 1, France

Licence : Philippe Marquet, System Programming, 60h, L3, Université Lille 1, France

Master: Philippe Marquet, Design of Operating System, 60h, M1, Université Lille 1, France

Master: Philippe Marquet, Web of Things: Embedded System Programming, 20h, M1, Université Lille 1, France

Master: Philippe Marquet, Parallel and Distributed Programming, 24h, M1, Université Lille 1, France

Master: Philippe Marquet, Introduction to Innovation and Research, 15h, M2, Université Lille 1, France

Licence : Samy Meftali, Architecture des ordinateurs, 75h, L2, UFR IEEA, Université Lille 1, France

Licence : Samy Meftali, Programmation du matériel, 40h, L3, UFR IEEA, Université Lille 1, France

Master: Samy Meftali, Architectures évoluées des ordinateurs, 80h, M1, UFR IEEA, Université Lille 1, France

Master: Samy Meftali, Vérification des SoC, 16h, M2, UFR IEEA, Université Lille 1, France

Licence : Jean-Luc Dekeyser, Architecture des ordinateurs, 80h, L2, UFR IEEA, Université Lille 1, France

Master: Jean-Luc Dekeyser, Architectures évoluées des ordinateurs, 105h, M1, UFR IEEA, Université Lille 1, France

Licence : F. Guyomarch, Algorithmique et programmation, 144h, L1, IUT-A (Univ. Lille 1)

Licence : F. Guyomarch, Modélisation et théorie des langages, 64h, L2, IUT-A (Univ. Lille 1)

Licence : F. Guyomarch, Modélisation mathématique, 28h, L2, IUT-A (Univ. Lille 1)

Licence : Rabie Ben Atitallah, Introduction to Computer Architecture and Operating System, 36h, L2, Université de Valenciennes et du Hainaut-Cambrésis, France

Licence : Rabie Ben Atitallah, Algorithms and Language C Programming, 48h, L2, Université de Valenciennes et du Hainaut-Cambrésis, France

Master: Rabie Ben Atitallah, Tools for Embedded System Design, 32h, M2, Université de Valenciennes et du Hainaut-Cambrésis, France

Master: Rabie Ben Atitallah, Development and Compilation of Embedded Application, 32h, M2, Université de Valenciennes et du Hainaut-Cambrésis, France

Master : V. Rusu, Architecture avancée des ordinateurs, 42h, M1, UFR IEEA, (Univ. Lille 1)

Master : V. Rusu, Spécification et vérification de logiciels , 27h, M1, UFR IEEA, (Univ. Lille 1)

8.2.2. Supervision

PhD : A. Arusoaie, *A Generic Framework for to Symbolic Execution*, Univ. Iasi (Romania), Sept 2014. Supervisors: D. Lucanu, V. Rusu.

PhD in progress : H. Krichene, SCAC : *Modèle d'exécution générique faiblement couplé pour les architectures massivement parallèle sur puce*, March 2011, Philippe Marque & Jean-Luc Dekeyser

PhD in progress : W. Chouchene, *Partial reconfiguration model for dynamic and massively parallel architecture : towards 3D FPGAs*, 01/10/2013, Jean-Luc Dekeyser, Rabie Ben Atitallah and Samy Meftali

PhD in progress : K. Ali, *Massively parallel dynamically reconfigurable architecture for real-time vidéo processing applications*, 01/10/2013, Jean-Luc Dekeyser, Rabie Ben Atitallah

PhD in progress : V. Viswanathan, *Parallel and Dynamic reconfigurable computing system*, 01/02/2012, Jean-Luc Dekeyser and Rabie Ben Atitallah

PhD in progress : V. Craciun, *Hardware monitoring*, Sept 2014, F. Guyomarch and D. Lucanu.

8.2.3. Juries

Doctorat de Amel Khiar, Virtualisation des communications au sein d'une plateforme hétérogène et reconfigurable dynamiquement, Université de Cergy Pontoise, 5 novembre 2014, Samy Meftali (examinateur).

Doctorat de Feriel Ben Abdallah, Modeling and Formal Verification of Power Management for the Design of Systems-on-Chip, Université de Valenciennes, 12 décembre 2014, Samy Meftali (examinateur).

8.3. Popularization

Philippe Marquet is vice-president of the *Société informatique de France*, the French learned society in computer science.

Philippe Marquet is involved in scientific popularization and co-animate the group of people interested in science popularization within the Inria Lille - Nord Europe Research Center. He is also of member of the group for networking about computer science popularization inside Inria.

He organizes and participates to the visit of classrooms on the Inria Plateau at EuraTechnologies, promoting interactions between the scientific community and secondary school students and their teachers. He organizes and/or animates events with children and/or adults in order to initiate them to code via Scratch or to computer science via unplugged activities (Poitiers, February 2014; Paris, October 2014; Inria Lille, December 2014).

Philippe Marquet is a member of the editorial board of 1024, the new bulletin of the *Société informatique de France* that aims at showing informatics, science and technology, in all its dimensions. 1024 targets a wide audience, from high school students to researcher, including anyone interested in computer science.

Hana Krichene participated in a contest "my thesis in 180 seconds", March 2014 - University Lille 1. The aim of the competition is to present the PhD student researches in simple terms in 3 minutes, with a clear, concise and convincing presentation to a diverse audience.

9. Bibliography

Publications of the year

Doctoral Dissertations and Habilitation Theses

[1] A. ARUSOAIE. *A Generic Framework for Symbolic Execution:Theory and Applications*, Alexandru Ioan Cuza, University of Iasi, September 2014, https://hal.inria.fr/tel-01094765

[2] R. BEN ATITALLAH. *Dynamic reconfiguration and low power design : towards self-adaptive massively parallel embedded systems*, Université de Valenciennes et Hainaut-Cambrésis, December 2014, Habilitation à diriger des recherches, https://hal.inria.fr/tel-01104009

Articles in International Peer-Reviewed Journals

[3] V. ARANEGA, A. W. DE OLIVEIRA RODRIGUES, A. ETIEN, F. GUYOMARCH, J.-L. DEKEYSER. *Integrating Profiling into MDE Compilers*, in "International Journal of Software Engineering & Applications (IJSEA)", July 2014, vol. 5, n^{o} 4, 20 p. [*DOI :* 10.5121/IJSEA.2014.5401], https://hal.inria.fr/hal-01053031

[4] V. ARANEGA, J.-M. MOTTU, A. ETIEN, T. DEGUEULE, B. BAUDRY, J.-L. DEKEYSER. *Towards an Automation of the Mutation Analysis Dedicated to Model Transformation*, in "Software Testing, Verification and Reliability", April 2014 [*DOI :* 10.1002/STVR.1532], https://hal.inria.fr/hal-00988164

[5] A. A. E. CADI, R. B. ATITALLAH, S. HANAFI, N. MLADENOVIC, A. ARTIBA. *New MIP model for multiprocessor scheduling problem with communication delays*, in "Optimization Letters", September 2014, 15 p. [*DOI :* 10.1007/s11590-014-0802-2], https://hal.inria.fr/hal-01104613

[6] D. LUCANU, V. RUSU. *Program Equivalence by Circular Reasoning*, in "Formal Aspects of Computing", 2014, 15 p. , forthcoming [*DOI :* 10.1007/s00165-014-0319-6], https://hal.inria.fr/hal-01065830

International Conferences with Proceedings

[7] K. M. A. ALI, R. BEN ATITALLAH, S. HANAFI, J.-L. DEKEYSER. *A Generic Pixel Distribution Architecture for Parallel Video Processing*, in "International Conference on Reconfigurable Computing and FPGAs - ReConFig 2014", Cancun, Mexico, December 2014, https://hal.inria.fr/hal-01070541

[8] A. ARUSOAIE, D. LUCANU, V. RUSU, T.-F. SERBANUTA, A. STEFANESCU, G. ROSU. *Language Definitions as Rewrite Theories*, in "International Workshop on Rewriting Logic and Application", Grenoble, France, April 2014, To appear in Springer LNCS, https://hal.inria.fr/hal-00950775

[9] M. BOUAIN, V. VISWANATHAN, R. BEN ATITALLAH, J.-L. DEKEYSER. *Communication-centric design for FMC based I/O system*, in "ReCoSoC - 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip", Montpellier, France, May 2014, https://hal.inria.fr/hal-01104610

[10] S. CIOBÂCAˇ, D. LUCANU, V. RUSU, G. ROSU. *A Language-Independent Proof System for Mutual Program Equivalence*, in "ICFEM'14 - 16th International Conference on Formal Engineering Methods", Luxembourg-Ville, Luxembourg, Springer, November 2014, forthcoming, https://hal.inria.fr/hal-01030754

[11] D. LUCANU, S. CIOBACA, V. RUSU, G. ROSU. *A theoretical foundation for language aggregation*, in "22nd International Workshop on Algebraic Development Techniques", Sinaia, Romania, September 2014, https://hal.inria.fr/hal-01076641

[12] C. TRABELSI, R. B. ATITALLAH, S. MEFTALI, J.-L. DEKEYSER. *Model-Driven design flow for distributed control in reconfigurable FPGA systems*, in "Conference on Design and Architectures for Signal and Image Processing (DASIP 2014)", Madrid, Spain, October 2014, https://hal.inria.fr/hal-01104617

[13] V. VISWANATHAN, R. BEN ATITALLAH, J.-L. DEKEYSER, B. NAKACHE, M. NAKACHE. *Redefining the role of FPGAs in the next generation avionic systems*, in "FPGA - ACM/SIGDA International Symposium on Field-Programmable Gate Arrays", Monterey, United States, February 2014 [*DOI :* 10.1145/2554688.2554744], https://hal.inria.fr/hal-01104615

Conferences without Proceedings

[14] S. CIOBÂCAˇ, D. LUCANU, V. RUSU, G. ROSU. *Programming Language Aggregation with Applications in Equivalence Checking*, in "PAS - Third International Seminar on Program Verification, Automated Debugging and Symbolic Computation", Vienne, Austria, July 2014, https://hal.inria.fr/hal-00998930

Research Reports

[15] A. ARUSOAIE, D. LUCANU, V. RUSU. *A Generic Framework for Symbolic Execution*, Inria, March 2014, n^{o} RR-8189, 27 p. , https://hal.inria.fr/hal-00766220

[16] A. ARUSOAIE, D. LUCANU, V. RUSU. *Language-Independent Program Verification Using Symbolic Execution*, Inria, October 2014, n^{o} RR-8369, 28 p. , https://hal.inria.fr/hal-00864341

Other Publications

[17] H. KRICHENE, M. BAKLOUTI, M. ABID, P. MARQUET, J.-L. DEKEYSER. *Communication-Computation overlap in massively parallel System on Chip*, May 2014, Tunisian Workshop on Embedded Systems Design (TWESD'2014), https://hal.archives-ouvertes.fr/hal-01104157