IN PARTNERSHIP WITH:
CNRS

Université des sciences et technologies de Lille (Lille 1)

Activity Report 2015

Project-Team DREAMPAL

Dynamic Reconfigurable Massively Parallel Architectures and Languages

RESEARCH CENTER
Lille - Nord Europe

THEME
Architecture, Languages and Compilation

Table of contents

Project-Team DREAMPAL

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

Keywords:

Computer Science and Digital Science:

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

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]
Rabie Ben Atitallah [Univ. Valenciennes, Associate Professor, HdR]
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]

PhD Students

Karim Ali [Univ. Valenciennes]
Wissem Chouchene [Univ. Lille I]
Hana Krichene [until Oct 2015]
Venkatasubramanian Viswanathan [until Jun 2015, granted by CIFRE]

Post-Doctoral Fellow

Andrei Arusoaie [Inria]

Administrative Assistant

Corinne Jamroz [Inria]

Other

Jean Perier [Inria, intern from Mar 2015 until Jul 2015]

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. Highlights of the Year

4.1. Highlights of the Year

2015 has been a good year in terms of journal publications for Dreampal, with 8 articles mostly in very high-quality venues.

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 few optimizations (automatic inlining and more compact byte-code), a binary decoder and a code-generator for VHDL simulation. New features of the HiHope language are described in [19].

- Contact: Frédéric Guyomarch
- URL: https://gforge.inria.fr/frs/?group_id=3646

6. New Results

6.1. HoMade in 2015

6.1.1. Interruption support

In the last release of HoMade we introduced interruptions. Up to 7 interruptions are supported. The priority is static and each trap is associated to one of the 7 first VCs of the master, they are called trap1 .. trap7. Trap is par nature reflective. When a trap is raised the HoMade master reaches a no-preemptive kernel. Traps have no effect on the slaves, they can continue to work. At the end of trap execution, HoMade master resumes the sequential execution, trap codes should be clean and should restitute the stack as it was when they began. A WAIT instruction and a long IP cannot be interrupted. An example of interrupts is provided in the reconfiguration part later.

6.1.2. New assembly language

HoMade waits for two binary codes: one for the master and one for the slaves. These two codes are loaded via the UART port and triggers a global reset of all the softcores after. Binary codes are a sequence of 16 bits words finishing by a long word filled with 4 NULL. Our post fixed macro assembler generates some binary codes from text files. This assembly language introduces some flow controls like if for repeat. It is also based on PC and VC definitions. Now the particular operator := generates reflective behaviors via WIM instructions. The syntax is so simple than everybody can understand a program. A full new syntax description is available with the assembler on the official HoMade web site : https://sites.google.com/site/homadeguide/assembleur-homade-v6. Here is the code for a mono HoMade to implement a reflective execution of Fibonacci suite. Switches values are put on the top on the stack to indicate the position in the list we want to process. Different input buttons affect the execution: • Button 0 changes to soft fibo execution using some library IPs. SWAP ROT DUP = - + are IPs to change the tops of the stack or to process dyadic integer operators. • Button 1 changes to hard execution using fibo vhdl long IP • Other buttons process the current fibo (hard or soft).

```
:IP fibo $AC54 ; // fibo hard IPcode 54
// XX = 1 YY = 1
program
  : read
    $1f // immediate hexa
    btnpush // IP reads buttons pushed
    switch // IP reads switches
  ;
  : fibo_soft // function declare
    1 1 rot
    3 -
    for
      dup rot +
    next
    swap
    drop
```

```
    ;
    VC fibo_dyn := fibo_soft // VC init soft
  start
    begin
    read
    swap dup
    0 = // test button
    if // reflective process
      fibo_dyn := fibo_soft
    endif
    1 =
    if
      fibo_dyn := fibo
    endif
    fibo_dyn // call VC
    7seg // IP to print result
    $1f
    btn // button to pause
    7seg
    again // infinite loop
  endprogram
```

When the VC fibo_dyn is called, you call hard or soft Fibonacci version depending of the sequence of pushed button. The soft code is 7 time slower than the hard code. The extra cost due to reflective facility is 2 cycles by VC call.

6.1.3. Dynamic IP reconfiguration

Xilinx chips are offering capabilities to program some pre-reserved chip areas with different bitsreams and this during the execution itself. It is not instantaneous and even worse the reconfiguration time depends of the length of the bitstream (the size of the area). Do not abuse of partial reconfigurations! But for some applications where context evolves at a “human speed”, our design can benefit of this functionality to adapt the hardware to the current context. It is easy to introduce this notion in HoMade: just insert an IP! This IP has to manage the bitstream memory and the ICAP to load them in the predefined areas. We develop a such IP for the master, without broadcast of bitstream to the slaves for the moment. This IP reconfiguration only needs to know the bitstream address. Effectively for Xilinx, the data inside the bitstream are sufficient to achieve the reconfiguration. We introduced the new keyword ‘in the assembler in order to express IP reconfigurations. The declaration of reconfigurable IPs may also include the bitstream address. Now we can program dynamic partial reconfiguration of IPs using our dedicated IP that we developed. Furthermore we can couple the dynamic reconfiguration with the reflective notion. Here is a simple example with dynamic image filters. The filter processes 1 block of 3x3 pixels. The 9 pixels are stored on the 3 top of the stack by aggregation of 3 pixels per word. External actuators can change from one IP to the other. We used interrupts and traps to apply this migration.

```
  program // bistream addresses between ( )
    :IP IP_median $EC11 ($0);
    :IP IP_Sobel $EC22 ($49E);
    VC filter
    : T1
      IP_median ^^
      filter := IP_median
    ;
    : T2
      IP_moyenne ^^
```

```
      filter := IP_moyenne
      trap1 := T1 // interrupt level 1
      trap2 := T2 // Interrupt level 2
      : get3pix // must be defined &
    ;
  start
    begin
      $7D for
        get3Pix // 3x3 pixels on stack
        get3Pix
        get3Pix
        -rot swap
        $7D for
          filter // current IP
          get3Pix // next 3 pixels
          -rot
        next
      next
    again
  endprogram
```

Concerning dynamic reconfiguration of IPs, we are testing a dedicated IP to manage directly the ICAP of Xilinx. The different bitstreams are stored in DDR3 and this IP finds the starting address from the stack. Of course this is a long IP. Some optimization to broadcast efficiently the same bitstream towards different slave reconfigurable areas are still a big challenge with Xilinx architecture.

6.1.4. IP fusion

To be free from EDA companies, we are deploying IP fusion strategies to manage the dynamic reconfiguration by ourselves. We obtain good results concerning the reconfiguration time, but for large and very different IPs, the fusion works like an aggregation of two IPs and the surface gain is insignificant.

6.1.5. Using hardware parallelism for reducing power consumption in video streaming applications

In the PhD thesis of Karim Ali we exploited using a flexible parallel hardware-based architecture in conjunction with frequency scaling as a technique for reducing power consumption in video streaming applications. In this work, we derived equations to ease the calculation for the level of parallelism and the maximum depth for the FIFOs used for clock domain crossing. Accordingly, a design space was formed including all the design alternatives for the application. The preferable design alternative is selected in aware of how much hardware it costs and what power reduction goal it can satisfy. We used Xilinx Zynq ZC706 evaluation board to implement two video streaming applications: Video downscaler (1:16) and AES encryption algorithm to verify our approach. The experimental results showed up to 19.6% power reduction for the video downscaler and up to 5.4% for the AES encryption. The architecture and experimental results were published in a paper entitled "Using hardware parallelism for reducing power consumption in video streaming applications" at the 10th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC) in Jun 2015, Bremen, Germany [12].

In collaboration with NAVYA, we started the first steps to implement a stereo vision algorithm over a parallel architecture using FPGA technologies. The algorithm is based on a local approach for calculating the disparity map using sum of absolute difference between the right and the left image. As a first step, we exploited the possible optimization levels that can be applied at the software level. After that by using high level synthesis tool (Vivado HLS from Xilinx) the code was written in C in a way that facilitates its conversion into HDL files. Optimization techniques were applied to reduce both the hardware resources and time required for processing one frame. This design was tested experimentally to show around 50% decrease in the time required for processing one frame if compared to the software one. Currently, we are in the step of exploring more techniques for hardware optimization and decreasing the processing time to meet the industrial requirements of our partner.

6.1.6. A scalable flexible and dynamic reconfigurable architecture for high performance embedded computing

In collaboration with Nolam Embedded Systems (NES) and in the framework of the CIFRE PhD of Venkatasubramanian Viswanathan, we proposed a scalable and customizable reconfigurable computing platform, with a parallel full-duplex switched communication network, and a software execution model to redefine the computation, communication and reconfiguration paradigms in high performance embedded systems. High Performance Embedded Computing (HPEC) applications are becoming highly sophisticated and resource consuming for three reasons. First, they should capture and process real-time data from several I/O sources in parallel. Second, they should adapt their functionalities according to the application or environment variations within given Size Weight and Power (SWaP) constraints. Third, since they process several parallel I/O sources, applications are often distributed on multiple computing nodes making them highly parallel. Due to the hardware parallelism and I/O bandwidth offered by Field Programmable Gate Arrays (FPGAs), application can be duplicated several times to process parallel I/Os, making Single Program Multiple Data (SPMD) the favorite execution model for designers implementing parallel architectures on FPGAs. Furthermore Dynamic Partial Reconfiguration (DPR) feature allows efficient reuse of limited hardware resources, making FPGA a highly attractive solution for such applications. The problem with current HPEC systems is that, they are usually built to meet the needs of a specific application, i.e., lacks flexibility to upgrade the system or reuse existing hardware resources. On the other hand, applications that run on such hardware architectures are constantly being upgraded. Thus there is a real need for flexible and scalable hardware architectures and parallel execution models in order to easily upgrade the system and reuse hardware resources within acceptable time bounds. Thus these applications face challenges such as obsolescence, hardware redesign cost, sequential and slow reconfiguration, and wastage of computing power.

Addressing the challenges described above, we propose an architecture that allows the customization of computing nodes (FPGAs), broadcast of data (I/O, bitstreams) and reconfiguration several or a subset of computing nodes in parallel. The software environment leverages the potential of the hardware switch, to provide support for the SPMD execution model. Finally, in order to demonstrate the benefits of our architecture, we have implemented a scalable distributed secure H.264 encoding application along with several avionic communication protocols for data and control transfers between the nodes. We have used a FMC based high-speed serial Front Panel Data Port (sFPDP) data acquisition protocol to capture, encode and encrypt RAW video streams. The system has been implemented on 3 different FPGAs, respecting the SPMD execution model. In addition, we have also implemented modular I/Os by swapping I/O protocols dynamically when required by the system. We have thus demonstrated a scalable and flexible architecture and a parallel runtime reconfiguration model in order to manage several parallel input video sources. These results represent a conceptual proof of a massively parallel dynamically reconfigurable next generation embedded computers [16] [15]. The PhD of Venkatasubramanian Viswanathan has been defended in the 12th of october 2015 .

6.2. Language-Parametric Formal Methods

The HoMade assembly language is still evolving. Thus, our research in formal methods for programming languages kept the language-parametric nature that we decided upon when we started the project. The

techniques and tools developed here will be instantiated on the HoMade assembly when it stabilizes. Our results are also applicable to general programming languages in order to target a broader audience.

In 2015 we have consolidated the results obtained in previous years, by making them more generally available and publishing them in high-end venues.

6.2.1. Language Definitions as Rewrite Theories

In [9] we study the foundations of $\mathbb{K}$, a formal framework for defining operational semantics of programming languages. The $\mathbb{K}$-Maude compiler translates $\mathbb{K}$ 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. $\mathbb{K}$ 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 definition. In this paper we investigate the theoretical relationship between $\mathbb{K}$ language definitions and their Maude translations, between symbolic extensions of $\mathbb{K}$ definitions and their Maude translations, and how the relationship between $\mathbb{K}$ 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.

6.2.2. A Generic Framework for Symbolic Execution: Theory and Applications

In [10], [17] 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 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 $\mathbb{K}$ framework. A prototype implementation of our approach has been developed in $\mathbb{K}$. We illustrate it on the symbolic analysis and deductive verification of nontrivial programs.

6.2.3. Symbolic Execution by Language Transformation

In [2] we propose a language-independent symbolic execution framework for languages endowed with a formal operational semantics based on term rewriting. Starting from a given definition of a language, a new language definition is generated, with the same syntax as the original one, but whose semantical rules are transformed in order to rewrite over logical formulas denoting possibly infinite sets of program states. Then, the symbolic execution of concrete programs is, by definition, the execution of the same programs with the symbolic semantics. We prove that the symbolic execution thus defined has the properties naturally expected from it (with respect to concrete program execution). A prototype implementation of our approach was developed in the K Framework. We demonstrate the tool's genericity by instantiating it on several languages, and illustrate it on the reachability analysis and model checking of several programs.

6.2.4. Program Equivalence by Circular Reasoning

In [7] 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.

6.2.5. Verifying Reachability-Logic Properties on Rewriting-Logic Specifications

Rewriting Logic is a simply, flexible, and powerful framework for specifying and analysing concurrent systems. Reachability Logic is a recently introduced formalism, which is currently used for defining the operational semantics of programming languages and for stating properties about program executions. Reachability Logic has its roots in a wider-spectrum framework, namely, in Rewriting Logic Semantics. In the invited paper [10] we show how Reachability Logic can be adapted for stating properties of transition systems described by Rewriting-Logic specifications. We propose a procedure for verifying Rewriting-Logic specifications against Reachability-Logic properties. We prove the soundness of the procedure and illustrate it by verifying a communication protocol specified in Maude.

6.2.6. A Theoretical Foundation for Programming Languages Aggregation

This work was published as [11]. Programming languages should be formally specified in order to reason about programs written in them. We show that, given two formally specified programming languages, it is possible to construct the formal semantics of an aggregated language, in which programs consist of pairs of programs from the initial languages. The construction is based on algebraic techniques and it can be used to reduce relational properties (such as equivalence of programs) to reachability properties (in the aggregated language).

6.3. The SCAC Model : a weakly-coupled execution model for MPSoC

Synchronous Communication Asynchronous Computation (SCAC) is an execution model that separates the execution of communication phases from those of computation in order to facilitate their overlapping, thus covering the data transfer time. To allow the simultaneous execution of these two phases, we propose an approach based on three levels : two globally-centralized/locally-distributed hierarchical control levels and a parallel computation level.

G-MPSoC [5] is a SCAC System-on-Chip implementation based on a grid of clusters of Hardware and Software Computation Elements with different size, performance, and complexity. It is composed of parametric IP-reused modules: processor, controller, accelerator, memory, interconnection network, etc. to build different architecture configurations. The generic structure of G-MPSoC facilitates its adaptation to the intensive signal processing applications requirements.

The communication phase in SCAC System-on-Chip 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 [14] interconnection network to communicate data in our system. The SCAC-Net network is composed of communication modules as the number of the nodes used by the system. Using generic parameters, the topology of SCAC-Net network can be easily configured according to the needed communication which give more flexibility to the system.

7. Bilateral Contracts and Grants with Industry

7.1. 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.

8. Partnerships and Cooperations

8.1. International Initiatives

8.1.1. Inria International Partners

8.1.1.1. Informal International Partners

We have a long-lasting collaboration with the universities of Illinois at Urbana Champaign (USA) and Iasi (Romania), which has been particularly fruitful in 2015 with 5 co-signed articles published or accepted for publication in high-quality journals.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific events organisation

9.1.1.1. member of the organizing committee

F. Guyomarch, P. Marquet and S. Meftali were members of the ComPAS 2015 organizing committee (Lille, 2015).

F. Guyomarch and S. Meftali were members of the Archi summer school organizing committee (Lille, 2015).

9.1.2. Scientific events selection

9.1.2.1. responsable of the conference program committee

S. Meftali was responsible of the ComPAS 2015 conference program committee.

9.1.2.2. Member of the conference program committees

J.-L. Dekeyser was member of the ComPAS 2015 conference program committee and is member in the Reconfig 2016 and ReCoSoc 2016 program committees.

F. Guyomarch is member of the ComPAS 2016 program committee, which will take place in Lorient (http://compas2016.sciencesconf.org).

V. Rusu is member of the WRLA 2016 program committee, which will take place in Eindhoven (https://fmse.info.uaic.ro/events/WRLA2016/).

9.1.3. Research administration

V. Rusu is elected member of the Inria evaluation committee.

9.2. Teaching - Supervision - Juries

9.2.1. 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, 72h, L1, IUT-A (Univ. Lille 1)

Licence : F. Guyomarch, Conception de sites web, 8h, L1, IUT-A (Univ. Lille 1)

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

Licence : F. Guyomarch, Algorithmique avancée, 44h, 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

Licence : V. Rusu, Logique , 30h, M1, UFR IEEA, (Univ. Lille 1)

Master : V. Rusu, Architecture avancée des ordinateurs, 36h, M1, UFR IEEA, (Univ. Lille 1)

9.2.2. Supervision

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

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

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

9.3. 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, and more generally promoting computer science education.

Philippe Marquet is a member of the editorial board of 1024, the 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.

10. Bibliography

Publications of the year

Articles in International Peer-Reviewed Journals

[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] A. ARUSOAIE, D. LUCANU, V. RUSU. *Symbolic execution based on language transformation*, in "Computer Languages, Systems and Structures", 2015, 42 p. [*DOI :* 10.1016/J.CL.2015.08.004], https://hal.inria.fr/hal-01186008

[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 Mutual Program Equivalence*, in "Formal Aspects of Computing", 2016, forthcoming, https://hal.inria.fr/hal-01245528

[5] H. KRICHENE, M. BAKLOUTI, M. ABID, P. MARQUET, J.-L. DEKEYSER. *G-MPSoC: Generic Massively Parallel Architecture on FPGA*, in "WSEAS Transactions on Circuits and Systems", November 2015, vol. 14, https://hal.inria.fr/hal-01246675

[6] D. LUCANU, V. RUSU, A. ARUSOAIE. *A Generic Framework for Symbolic Execution: Theory and Applications*, in "Journal of Symbolic Computation", 2016, forthcoming, https://hal.inria.fr/hal-01238696

[7] D. LUCANU, V. RUSU. *Program Equivalence by Circular Reasoning*, in "Formal Aspects of Computing", 2015, vol. 27, n^{o} 4, pp. 701-726 [*DOI :* 10.1007/s00165-014-0319-6], https://hal.inria.fr/hal-01065830

[8] 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

[9] 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", 2015, 48 p. , forthcoming [*DOI :* 10.1016/J.JLAMP.2015.09.001], https://hal.inria.fr/hal-01186005

Invited Conferences

[10] 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

International Conferences with Proceedings

[11] S. CIOBACA, D. LUCANU, V. RUSU, G. ROSU. *A Theoretical Foundation for Programming Languages Aggregation*, in "22nd International Workshop on Algebraic Development Techniques", Sinaia, Romania, LNCS, Spriger Verlag, 2015, vol. 9463, https://hal.inria.fr/hal-01076641

Conferences without Proceedings

[12] 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

[13] J.-L. DEKEYSER, A. S. ALJENDI. *Adopting New Learning Strategies for Computer Architecture in Higher Education Case Study: Building the S3 Microprocessor in 24 Hours*, in "Workshop on Computer Architecture Education held in conjunction with the 42nd International Symposium on Computer Architeture", Portland, United States, June 2015, https://hal.inria.fr/hal-01152144

[14] 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

[15] 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

[16] 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

Research Reports

[17] A. ARUSOAIE, D. LUCANU, V. RUSU. *A Generic Framework for Symbolic Execution: Theory and Applications*, Inria, September 2015, n° RR-8189, 41 p. , https://hal.inria.fr/hal-00766220

[18] A. ARUSOAIE, D. NOWAK, V. RUSU, D. LUCANU. *Formal Proof of Soundness for an RL Prover*, Inria Lille - Nord Europe ; Alexandru Ioan Cuza, University of Iasi, December 2015, n° RR-471, 27 p. , https://hal.inria.fr/hal-01244578

[19] J. FORGET, F. GUYOMARCH, V. RUSU. *Programming with hardware/software functions*, Inria Lille Nord Europe, December 2015, n° 8835, 18 p. , https://hal.inria.fr/hal-01248163

Patents and standards

[20] G. AFONSO, W. GODARD, R. BEN ATITALLAH, J.-L. DEKEYSER. *Système de Simulation et de Test*, July 2015, n° WO 2015097105, https://hal.inria.fr/hal-01247395