RESEARCH CENTRE

Lille - Nord Europe

IN PARTNERSHIP WITH:

Université de Lille, CNRS

2021

ACTIVITY REPORT

Project-Team

SYCOMORES

Symbolic analysis and Component-based design for Modular Real-Time Embedded Systems

IN COLLABORATION WITH: Centre de Recherche en Informatique, Signal et Automatique de Lille

DOMAIN

Algorithmics, Programming, Software and Architecture

THEME

Embedded and Real-time Systems

Contents

Project-Team SYCOMORES

Creation of the Project-Team: 2021 October 01

Keywords

Computer sciences and digital sciences

A2.1.9. – Synchronous languages

A2.3.1. – Embedded systems

A2.3.3. – Real-time systems

A2.4.1. – Analysis

A2.4.3. – Proofs

A7.2. – Logic in Computer Science

Other research topics and application domains

B6.6. – Embedded systems

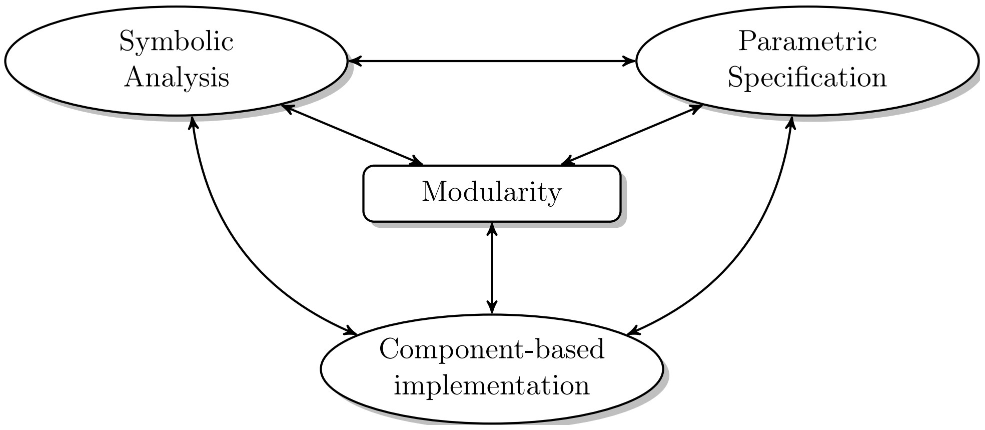

Figure 1: Ontology of terms in SYCOMORES.

1 Team members, visitors, external collaborators

Research Scientists

- Patrick Baillot [CNRS, Senior Researcher, from Oct 2021, HDR]
- Vlad Rusu [Inria, Researcher, from Oct 2021, HDR]

Faculty Members

- Giuseppe Lipari [Team leader, Université de Lille, Professor, from Oct 2021]
- Clement Ballabriga [Université de Lille, from Oct 2021, Associate Professor]
- Julien Forget [Université de Lille, Associate Professor, from Oct 2021]

PhD Students

- Fabien Bouquillon [Université de Lille, from Oct 2021]
- Frederic Fort [Université de Lille, from Oct 2021]
- Sandro Grebant [Université de Lille, from Oct 2021]
- Ikram Senoussaoui [Université Lille and Université Oran 1 (Algérie), From oct 2021]

Administrative Assistant

- Nathalie Bonte [Inria, from Oct 2021]

2 Overall objectives

The SYCOMORES project-team aims at developing a **framework for the design and the analysis of embedded real-time systems** based on **symbolic analysis of parametric components**.

To reduce the complexity, we will use a modular approach to the design, development and analysis of ERTS. In particular, we will decline *modularity* along several directions (Figure 1):

- a *component-based design and development* methodology; a ERTS system is decomposed in generic, configurable and reusable components that can be combined and instantiated in the final system;

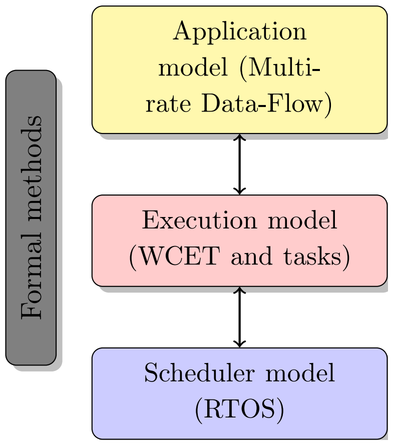
Application model (Multi-rate Data-Flow)
Execution model (WCET and tasks)
Scheduler model (RTOS)
Formal methods

Figure 2: Abstraction Layers

- *parametric specification* of inputs, configurations and properties; a component is characterised by its interface that will be specified using parameters whose concrete values are unknown at design time;

- *symbolic analysis* of components; by using symbolic techniques, the properties of a component will be proved correct at design time even when parameters have unknown values.

Finally, we will propose *composition operators* to integrate a set of components into a larger component, or into a complete system. The operators will allow to (partially or totally) instantiate parameters and connect component interfaces guaranteeing their compatibility and preserving their properties.

To this end, we will investigate several challenging problems at three different levels of abstraction, as shown in Figure 2. At the application level, we will focus on *multi-rate data-flow* programming languages like SCADE [6], SIMULINK [19], or in our case PRELUDE [21], as they are widely used in safety-critical domains like avionics and automotive. We want to tackle the notions of component, parametric interfaces and contracts in the ERTS context. We will investigate **modular code generation** (compilation) of a synchronous component (as opposed to a synchronous program) with real-time constraints. In the long term, we would like to **prove the semantic preservation of properties** of the program during compilation by using interactive proof systems.

The compiler will produce the component implementation as a set of concurrent real-time tasks with their scheduling parameters. At this level, we will work on **parametric and modular worst-case execution time (WCET) analysis** of tasks' code, and **symbolic schedulability analysis** of the components' tasks. We will also work on analysis of the correctness of system integration, and **safe dynamic replacement/upgrade** of components.

At the lowest level, we will work on the scheduler implementation in the Operating System. We will propose a **hierarchical scheduling architecture**, in order to provide temporal isolation to real-time components. We will formally prove the scheduler properties by using **modular proof techniques** in the Coq proof assistat [27], and we will provide a **correct-by construction implementation of the scheduler** by using code generation techniques from a Coq program specification.

As a cross-cutting activity, since safety will be a prime concern in our project, we will rely on *formal methods* throughout the project: synchronous programming languages, abstract interpretation, symbolic analyses and proof techniques. Our research objectives will be tightly coupled to ensure the safety of the final framework.

For instance, our tools and techniques will share proof facts (e.g. timing analysis is correct only if schedulability analysis and WCET analysis are both correct), models (e.g. the schedulability analysis, programming language and WCET analysis must share the same task model), and results (e.g. the WCET and schedulability results will impact the program design).

3 Research program

We detail the objectives by categorising them according to 3 major research axes. Within each axis, we distinguish Short Term (1-3 years) and Medium Term (2-4 years) objectives, and we describe how we will achieve them. We also detail how the objectives are related to one another. Finally, we present our Long Term (4+ years) objectives.

3.1 Axis 1: Design and implementation of RT-components

In this research axis we will focus on the programming of RT-components. We will cover programming aspects at different levels of the development process, from high-level design, to low-level implementation on distributed heterogeneous embedded platforms.

3.1.1 Short term

(A1.S1) Synchronous languages We will continue working on synchronous programming languages for generating correct-by-construction code. In particular, one of the objectives of the ANR projet CORTEVA (which is coordinated by G. Lipari, started in Jan. 2018, and will last for 48 months) is to extend the synchronous language PRELUDE [21] to address systems with extremely variable execution time.

The language is being extended (its semantics and its compilation) with constructs to dynamically change the behaviour of a subset of the system based on rare events, like the occurrence of a large execution time of a task. The proposed extensions will guarantee that the system will respect correctness properties even in the presence of such rare events.

Dependencies: We rely on Parametric WCET to detect the occurence of large execution times (A2.S1).

(A1.S2) Scheduling of complex task models Modern hardware platforms consist of heterogeneous processors with a large degree of parallelism. For example, the NVIDIA Jetson AGX Xavier chipthat is used in modern ADAS systems in the automotive domain, comprises 2 DLAs (Deep Learning Accelerators), 1 integrated GPU (Graphical Processing Unit), and 8 ARM 57 processing cores. Efficiently programming ERTS for such hardware is not easy, as the complexity of the interaction between software and hardware resources makes it difficult to predict performance. Building predictable real-time applications on these platforms requires appropriate programming models. In the recent literature, many graph-based task models have been proposed to exploit the parallelism of such architectures [26, 9]. Such models are neither based on components, nor are they parametric. It is our intention to investigate the possibility to apply component-based techniques to such complex task models.

3.1.2 Medium term

(A1.M1) Parametric scheduling analysis of Real-Time components. Analyzing the schedulability of a system under variations of the task parameters is a complex problem: the complexity of the analysis grows exponentially with the size of the system, and it explodes for medium-to-large sized systems.

The problem can be decomposed by analysing the schedulability of smaller real-time components under variation of some parameters. The idea is to follow a Resource Reservation strategy, where components are assigned and guaranteed a fraction of the system resources. In this way, the temporal behaviour of one component is *isolated* from the interference of other components.

First, we will introduce the notion of *parametric interface* for a component modelled as a multi-rate data-flow program. The interface will specify the temporal properties and constraints on the input/outputs of a component: periodicity and deadline constraints, dimension of the buffer for communicating with other components, and communication protocols.

To analyse the schedulability of a components under the hypothesis of variation of its parameters (the ones specified in the interface as well as the WCET of the tasks), we will extend the *sensitivity analysis* techniques, first proposed by Bini et al. [11, 10] to more complex task models and to hierarchical scheduling systems. The objective is to find the range of the parameters for which the component is schedulable.

(A1.M2) Predictable communication costs. ERTS are increasingly being built using multicore hardware. This transition is however still significantly delayed by the difficulty to provide *safe* and *tight* timing guarantees. Indeed, even though multicore architectures enable the simultaneous parallel execution of programs on different cores, these programs are not completely independent. They have to be synchronized at some point to exchange data and they also share some common resources, such as peripherals, communication bus and (parts of the) memory. Synchronizations and shared resource contentions cause hard-to-predict interferences and timing overheads, which significantly increase the complexity of timing analyzes (WCET and scheduling).

One solution for mitigating these overheads, and making them more predictable, consists in relying on multi-phase task models that decouple communication phases from computation phases [23, 14]. This greatly simplifies WCET analysis of computation phases, and also makes it possible to schedule communication phases so as to reduce their interference.

Memory architectures based on local addressable memories (scratchpads), instead of caches, are also being proposed [15, 22], to avoid the hard-to-predict timing effects of cache consistency mechanisms (which are used to speed-up access to the main shared memory).

We plan to integrate these two approaches in our project, since predictability is a major concern. The PRELUDE compiler will be extended to provide multi-phase task code generation, to be executed on scratchpad-based memory architectures. We will also develop the corresponding schedulability analysis method. Our objective is to devise our development framework such that the programmer abstracts from the implementation of communication mechanisms related on the target OS and hardware. This will enable the programmer to seamlessly transition between different architectures (unicore/multicore, cache-based/scratchpad-based).

3.2 Axis 2: Static Analysis of RT components

In this research axis, we will design static program analysis techniques for RT components at the binary level. While these techniques will mainly focus on WCET analysis, some results will also be reused to analyze security properties instead.

3.2.1 Short Term

(A2.S1) Symbolic WCET analysis. We will work towards improving static analysis techniques for Worst-Case Execution Time Analysis. In particular, we will extend the Parametric WCET method [8] with more powerful symbolic capabilities. Currently, Parametric WCET can represent the WCET of a function as a formula in a number of parameters (e.g. input data). It does however suffer from two important limitations. First, it cannot represent relations between distinct parameters. We will extend the Parametric WCET by abstract interpretation of binary code to detect linear relations between distinct parameters in the code, by using techniques similar to [7]. Second, it struggles to represent program properties that relate instructions of the program that are not close to one another, such as for instance infeasible execution paths [24]. We are currently extending this work to enable the representation of properties related to such *global execution contexts*.

3.2.2 Medium Term

(A2.M1) Modular WCET analysis Traditional WCET analysis is performed on a whole program. This approach is limited by two factors: first, performance concerns (analysis time tends to grow faster than the analyzed program size), and second, the analysis requires access to the complete program (including libraries, etc.). A modular and composable WCET analysis approach would reduce the analysis time, and enable the integration of third-party library or system calls in the analysis process.

To this end, we will extend our polyhedra-based abstract interpretation [7] to support inter-procedural analysis, based on function *summaries* [13], describing relations between the inputs and outputs of the function. This will enable us to compute WCET-related functional properties, such as e.g. loop bounds, in a composable way.

This work on composable analysis will lead to a full composable WCET analysis, by integrating it in our symbolic WCET computation framework [8].

Dependencies: To achieve this objective, we will need the advanced parameter handling techniques of S2.

(A2.M2) Security analysis of binary code Program analysis of ERTS has historically focused mainly on *safety*, i.e. ensuring the absence of system failures. However, in recent years ERTS have become increasingly more connected to other computer systems through communication networks. This makes ERTS more vulnerable to external attacks, as illustrated by various reports of hacks of highly computerized modern cars. This results in an increased need for analyses that focus on ensuring the *security* of ERTS, i.e. ensuring there is no vulnerability to external malevolent computer attacks.

Our work on abstract interpretation of binary code [7] enables to detect linear relations between the data-locations accessed by a binary program (registers, memory addresses and their contents). While this work was initially targeted for WCET analysis, we plan to apply the developped techniques to the security domain as well. In particular, we will analyze security properties specific to the binary structure (e.g. unauthorized accesses to specific memory sections) and study the discovery of potential security threats in closed-source software (to detect malwares).

3.3 Axis 3: Proof of RT Components

The formal proof activity in the project will focus on compositional techniques for proving RT components of operating systems. Our plan is to focus mainly on RT schedulers.

3.3.1 Short term

(A3.S1) A proof methodology for a standard scheduling policy We shall start with the standard scheduling policy EDF (Earliest Deadline First) and develop a proof methodology for it, which we shall later adapt to more advanced policies. The methodology incorporates a formal notion *refinement* as a means to master complexity and to smoothly descend from abstract definitions down to executable schedulers.

First, we are planning to model EDF at an abstract level in Coq. We shall formally prove at this level the *schedulability* property: under adequate hypotheses any given set of periodic hard real-time tasks can be scheduled, such that each task completes before its deadline. Since in the short term we shall only deal with strictly periodic, hard real-time tasks (reading data from sensors, performing some computation and sending the results to actuators), one only needs to consider the schedulability on finite executions, whose length equals the hyper-period: the least common multiplier of the task's periods. As a result, we expect that *induction*, well supported by Coq, will be the appropriate proof technique for this stage of the project.

Then, we shall refine the abstract EDF scheduling policy into a scheduling *algorithm* written in Coq's input language *Gallina*, a purely functional language, and shall prove again by induction that the algorithm preserves the already established properties of the policy.

Next, our plan is to further refine these *functional* algorithms into *imperative* Gallina programs, in order to get a step closer to executable code[1]. Imperative programs in purely functional languages such as Gallina are traditionally written using *monads* e.g., *state* monads, which enable variable assignments in functional code. For doing this we shall benefit from the experience of our colleagues in the 2XS team, our closest collaborators within the CRIStAL laboratory. They have been developing a minimalistic OS kernel called Pip [28] in imperative Gallina using the state-monad technology and have directly proved properties of the kernel in Coq. Unlike them, we do not prove properties directly on the monadic code, but shall prove a refinement step (from functional to imperative) ensuring that schedulability holds on the imperative scheduler.

The final step is translating the imperative Gallina scheduler to executable code. For this we shall use a translator also developped by the 2XS team, which essentially maps word-for-word imperartive Gallina programs to C programs with appropriate primitives that, after compilation of the C code, turn our schedulers into executable programs within Pip.

[1]Going directly from fonctional to executable code is not an option, since that would require a complex compilation process with a high risk of losing the schedulability properties already proved on the functional code.

3.3.2 Medium term

In the medium term we are planning to make progress on two directions: the RT components being proved and the proof techniques.

(A3.M1) More advanced schedulers and other RT components Once the validation of the proof/refinement methodology on the EDF example is complete, we are planning to progressively generalise it to other schedulers. The next likely candidate is the *Grub* scheduler, developped by Giuseppe Lipari [5, 17], which generalises EDF and makes it possible to mix periodic tasks (hard real-time) with sporadic tasks (soft real-time). The algorithm guarantees that the deadlines of the hard real-time tasks are met, while for the soft real-time ones a certain quality of service is guaranteed.

We are also planning to formally verify existing resource reservation hierarchical schedulers [18] by extending the proof/refinement approach with compositional features that exploit the component-based nature of the considered applications.

Depending on the availability of human resources, we would also like to formally verify other RT components, such as interruption multiplexers, memory managers, or synchronisation mechanisms.

(A3.M2) More advanced proof techniques We expect that different verification techniques will be necessary to prove these more advanced schedulers. The EDF scheduler can be proved correct by considering its behaviour over a limited period of time, which as explained above can be dealt with using induction. However, Grub also has to schedule sporadic tasks that, by definition, do not repeat themselves periodically. Without a periodic repetition the behaviour cannot be studied over a finite interval, infinite executions have to be considered. Hence, we are planning to use coinduction, the natural technique for defining and reasoning about infinite objects. As stated in the Preliminaries we already have experience with coinduction, especially, with improving the way it is dealt with in Coq to make it better suited for use in nontrivial applications. We are planning to continue doing this both for corecursive program definitions (e.g., reactive systems, including schedulers, are such programs) and for coinductive reasoning techniques.

3.4 Long term

In the long term, we will integrate the elements studied during the medium term phase, in order to provide a seamless framework that goes from high-level design to final implementation. Translations will be automated and proved correct. These objectives are transverse by nature, so all members of the project-team will participate. Since these objectives focus on integration of our previous results, they are related to all of our short and medium term objectives.

(L1) Integrated framework During the medium term phase, we will develop bricks that contribute to the design and analysis of ERTS. In the long term phase, we will integrate these bricks into a complete framework. This will raise several research topics.

First, we will have to evaluate the impact of scheduling on WCET analysis. Though this topic has already been studied before, our symbolic approach to both problems will raise new opportunities and challenges.

Second, we will evaluate the scalability of the approach with respect to realistic and complex real-time programs. In particular, the advent of powerful heterogeneous hardware platform permits to exploit their large scale parallelism. It is then important to check the suitability and the expressiveness of our framework with respect to these new powerful platforms.

(L2) Proving semantics preservation In our framework, an ERTS is first specified with a high-level data-flow semantics (in PRELUDE). Then, it undergoes translations to produce the final program (in C) embedded on the hardware platform. One of our long term objectives is to prove that the complete translation process, from PRELUDEto C, is semantics-preserving.

There exists previous work and techniques for the formal verification of compilers such as COMPCERT [16] (from C code to binary code) and the LUSTRE verified compiler [12]. However, these works focus on the preservation of the *functional semantics* (computing the correct values). A major novelty of our work will be to focus on the preservation of the *temporal semantics* (computing values at the correct time) as well. As far as we know, proving the preservation of temporal semantics was previously explored in theoretical models (e.g. timed automata, or Petri nets), but not in programming languages and compilers. It is an ambitious challenge, because it connects compilation with scheduling and WCET analysis.

(L3) Corecursion-preserving compilation and coinductive verification techniques An alternative view of reactive programs (such as PRELUDE programs) is that of corecursive transformers from infinite flows of inputs to infinite flows of outputs. We envisage an enhanced compilation chain that, in addition to what is planned in L2, enables the tracing of corecursion down to the executable code. Since corecursive calls typically compile to low-level instructions such as loops or jumps, such a tracing mechanism would enable recognising the corecursive calls at the low-level. This, in turn, would enable the formal reasoning about low-level corecursive programs, using coinductive techniques that we shall develop in A3. We note that schedulers can also be expressed as corecursive programs, hence the verification boils down to compositionally verifying compositions of corecursive programs. It is, again, an ambitious challenge at every step, since corercursive programs are notoriously difficult to define, compose, and verify. This envisaged works depends on progress in essentially all the objectives enumerated above.

(L4) Dynamic update of components ERTS may evolve over time, in particular, one component may be upgraded while the system is operational. In the traditional development process of critical software, once the system has been developed, tested and *certified*, it is not possible to change it anymore, with the only exception of correcting a critical bug. Dynamic updates are not possible since they would require to re-certify the whole system.

In the long term, we would like to apply our component based development process to the problem of guaranteeing the correctness of dynamic updates. This means that the system must be able to evolve over time, during operation, without jeopardizing the guarantee on existing properties.

Given the large complexity of the formal methods we will use for off-line analysis, it is unthinkable to apply the same type of analysis on-line. One possibility is to separate the analysis into two distinct parts: an off-line part which may be complex and does most of the work; and an on-line part which is much simpler but relies on the off-line computation. Alternatively, it is possible to off-load part of the analysis to the cloud, where there is abundance of computational resources.

4 Application domains

The long term research we propose to pursue in this project will advance the current development practice for embedded real-time critical systems.

First, it will impact the design and development of critical software for domains like avionics, automotive, train, etc. It is well known that developing safety-critical software is a long and costly process, where each error could endanger human life. It has been estimated that the cost to certify 30K lines of DAL A code is around 2M $ if the code has been developed by experienced programmers, and it jumps to 8M$ if the code has been produced by non-experienced ones.

The avionic industry is slowly adopting formal methods to reduce the cost by reducing (or, in certain cases, eliminating altogether) testing and improve confidence in the methodology. Our integrated framework (L1) will greatly reduce the development cost of safety-critical software because it will automatise many of the steps in the design and development methodology. For example, the use of semantic preserving transformations (L2, L3) will enhance the confidence in the correctness of the generated code, reducing the need for extensive testing, thus further reducing cost.

Other critical domains, like automotive, have not yet fully adopted safety-critical standard methodologies like in the avionic domain, mainly for cost reasons. Our framework will lower the cost of developing safety critical software, thus improving the safety of critical software in a wider range of domains.

Finally, thanks to the research in (L4), it will be possible to update a software component on-line without performing a complete analysis of the system, *while the system is operational.* An example of futuristic application will be the possibility to update one software subsystem of an autonomous vehicle while the vehicle is running, without compromising its functionality.

5 New software and platforms

Since the SYCOMORES team has been created on October 1, 2021, in this section we report some of the software developed by the members of the team in the past years and that will be used as a basis for future development in the project.

5.1 New software

5.1.1 otawa

Keywords: Static analysis, WCET

Functional Description: Otawa is an open source static analysis tool developed and maintained by the TRACES team at the University of Toulouse and co-developed by the team SYCOMORES at Inria lille Nord-Europe. Specifically, SYCOMORES has developed two plugins for OTAWA: Polymalys, for polyhedra-based analysis and loop bound estimation, and WSymb, a Control Flow Tree extractor and symbolic WCET evaluator.

Contact: Clement Ballabriga

Partner: Université de Toulouse

5.1.2 prelude

Keywords: Synchorous language, Real time

Functional Description: Prelude is a synchronous real-time language designed by Julien Forget. The language is currently being actively used at ONERA on industrial case studies. It is currently being extended to support adaptive applications.

Contact: Julien Forget

5.1.3 ptask

Keywords: Real time, Library

Functional Description: PTASK is a library for programming real-time systems in Linux. It serves as the target RTOS API in the SYCOMORES project. The PTASK library is authored by Giuseppe Lipari. It has been initially developed to support teaching real-time systems at the Scuola Sant'Anna. it has later been extended with additional capabilities and it is being used as target for design tools such as CPAL19 from RTaW20 and by other companies.

Contact: Giuseppe Lipari

6 New results

The SYCOMORES team has been created on October 1, 2021, therefore we only report the work performed or concluded in the last 3 months of the year.

6.1 Cache related preemption delay

The research described in this section has been done by Fabien Bouquillon, PhD student of the SYCOMORES team, co-directed by Giuseppe Lipari and Smail Niar (Université Polytechnique Haut de France).

The research concerns the estimation of the cost of preemptions due to additional cache misses. This cost is called *Cache Related Preemption Delay* (CRPD) and the CRPD analysis is a methodology for bounding the cost of cache reloads due to preemptions. Many techniques have been proposed to estimate upper bounds to the CRPD for Fixed Priority (FP) and Earliest Deadline First (EDF) scheduling. Given the complexity of the problem, existing methods make simplifying assumptions to speed up the analysis, but they also introduce large amounts of pessimism. We proposed two original contributions for reducing the pessimism in the CRPD analysis of real-time systems that use set- associative cache memories: a new way to compute a bound on the number of preemptions on a task under EDF scheduling, and a novel algorithm that trades off speed for precision in state-of-the-art analysis in the literature.

The article containing these results has been recently accepted at the ACM SAC 2022 international conference [2], and it will be presented in February.

Participants: Fabien Bouquillon, Giuseppe Lipari, Smail Niar (UPHF).

6.2 Real-Time scheduling on FPGAs

The research described in this section has been carried out in collaboration with the RETIS Lab of the Scuola Superiore Sant'Anna in the context of the PhD thesis of Marco Pagani, co-directed by Giuseppe Lipari and Giorgio Buttazzo. The research concerns the real-time scheduling of tasks on dynamically reconfigurable FPGAs.

Dynamic partial reconfiguration allows virtualizing the FPGA resources by sharing them among multiple hardware accelerators over time. Although very promising, FPGA-based hardware acceleration also introduces new challenges, such as managing and scheduling multiple concurrent acceleration and reconfiguration requests.

In this research, the FRED-Linux library has been proposed to address these challenges while preserving the predictability required by real-time systems. Fred-Linux allows developing rich applications while leveraging predictable FPGA-based hardware acceleration for performing heavy computations. This research has been published in [1].

Participants: Marco Pagani, Giuseppe Lipari.

6.3 Multi-mode synchronous programs

This work has been carried out by Fréderic Fort, PhD student of the SYCOMORE team, directed by Giuseppe Lipari and tutored by Julien Forget.

The research tackles the problem of designing and programming a real-time system with multiple modes of execution, where each mode executes a different set of periodic tasks. The main problem to tackle is that the period of Mode Change Requests (MCR) and the period of tasks are not all the same. Thus, not all tasks perceive MCRs in the same way. When programming such a system with traditional languages without mechanisms dedicated to mode changes (e.g. C), it is difficult to ensure a system is sound and deterministic.

We proposed an extension to synchronous dataflow languages to support mode changes. The semantics of the resulting language is defined formally, which prevents ambiguous programs. The language is flexible enough to support different types of mode changes. The compiler of the language includes a static analysis that rejects programs whose semantics is ill-defined. The extension consists in transposing Synchronous State Machines to the Prelude language. This requires to extend the semantics of Prelude, and to define a new clock calculus, based on refinement typing.

These results have been recently accepted at the ACM SAC 2022 international conference [4], to be held on February 2022.

Participants: Fréderic Fort, Julien Forget.

6.4 Scheduling for Networks on Chips

Mr. Chawki Benchehida has successfully defended his PhD Thesis titled "Mapping Hard Real-Time Tasks on Network-on-Chip Manycore Architectures" [3] at the University of Lille on the 9th of November 2021. His thesis was co-directed by Giuseppe Lipari and by Kamel Benahoua (University of Oran 1), and tutored by Houssam Zahaf (Université de Nantes).

The thesis tackles the problem of execution complex multi-thread real-time applications on modern Network-on-Chip architectures. In real-time systems, The communications that occur in the network, denoted by on-chip communications, must be predictable and as fast as possible to prevent deadline misses. Since the task position on the NoC determines its communication cost, the allocation of the application tasks on the chip cores is a crucial problem.

The thesis addresses the problem of allocating a set of real-time applications, each composed of several parallel tasks, whose structure is described by a Directed Acyclic Graph (DAG), onto a Network-on-Chip processor. Three main problems are addressed: 1) the problem of bounding the communication cost depending on the different message scheduling policies at the router level; 2) the problem of task scheduling and of verifying the schedulability of a given allocation; 3) how to reduce the complexity of the task allocation problem and its analysis cost. In particular, the thesis proposes a task mapping strategy through a meta-heuristic which performs an effective design-space exploration for DAG (Directed Acyclic Graph) tasks.

Participants: Chawki Benchehida, Giuseppe Lipari, Houssam Zahaf (Univ. Nantes), Kamel Benahoua (Univ. Oran 1).

6.5 Efficient tree-based symbolic WCET computation

This work is part of the PhD thesis of Sandro Grebant, directed by Giuseppe Lipari and tutored by Julien Forget.

The research concerns the symbolic WCET analysis. Contrary to classical analyses which produces a constant value, symbolic WCET analysis produces a formula with different user-defined parameters of the program such as loop bounds. It relies on a tree-based representation similar to a control-flow graph (CFG), albeit with a tree structure.

Classical WCET analyses rely on integer linear programming with the implicit path enumeration technique but a tree-based representation is more suitable for a symbolic analysis. The goal of our work is to be able to represent different hardware and software f acts in order to tighten the resulting WCET bound. A hardware fact represents the effect of a hardware component, e.g. processor caches, on the execution time of the program. A software fact represents the effect of an element of the program structure, e.g. a loop bound, on the execution time of the program. To achieve those representations, existing techniques to model hardware facts need to be adapted to the tree-based representation and the existing annotation language needs to be enhanced. Annotations provide a way to express several flow facts which are not always related to the program structure. A great example would be a lower bound on the execution count of a part of a loop. We are currently working on the representation of infeasible paths for tree-based WCET analysis. The tree will be annotated with the results of an infeasible paths analysis. It will allow to tighten the resulting WCET by excluding some nodes from the WCET path.

Some preliminary results on this work have been presented at the Compas Workshop in early 2021.

Participants: Sandro Grebant, Julien Forget, Clément Ballabriga.

6.6 Type-based analysis of program sensitivity

Patrick Baillot has been working with colleagues of Boston University (USA) (M. Gaboardi, A. Azevedo De Amorim, J. Wunder) on type-based analysis of program sensitivity. The context is that of data related applications, where one wants to reason not only about what a program computes but also about how changes in the input data can affect the output. This notion can be formalized using metric spaces: given a metric on the input and output spaces one can reason about how far apart two related inputs are mapped by a program. This is called program sensitivity.

The 2010 seminal paper [25] had shown how to reason statically about program sensitivity by means of a type system for a functional language, inspired by linear logic. Baillot and his colleagues are exploring how this type system can be extended with new types corresponding to metrics often used in privacy and optimization, the Lp distances. This is obtained by taking inspiration from Bunched logic [20]. They also show that this type system can help to reason on probability distributions in a natural way. A technical report is currently being written.

Participants: Patrick Baillot.

6.7 A formal correctness proof for an EDF scheduler implementation

The scheduler is a critical piece of software in real-time systems that orchestrates the various tasks to be executed. A failure can have serious consequences; therefore, it is important to provide strong guarantees for this software. In this work we propose a formal proof methodology that we concretely apply to an EDF scheduler. It consists first in proving the correctness of the election function algorithm, and then lifting this proof up to the implementation through refinements. The proofs are formalized in the Coq proof assistant, ensuring that they are free of human errors and that all cases are considered. Our methodology is general enough to be applied to any piece of software, including other schedulers or other types of system code. To the best of our knowledge, this is the first time that an implementation of EDF is proved correct.

This work is a collaboration with colleagues from the 2XS team of the Cristal laboratory in Lille. It is currently submitted for publication.

Participants: Florian Vanhems (2XS), Vlad Rusu, David Nowak (2XS), Gilles Grimaud (2XS).

6.8 Defining corecursive functions in Coq using approximations

We present two methods for defining corecursive functions that go beyond what the dedicated mechanisms of the Coq proof assistant for defining such functions are able to accept. The first method uses those same dedicated mechanisms, but in a manner that is substantially different from the standard one. It can be used to define corecursive functions as long as the functions' codomains are coinductive types not mutually dependent with inductive types. The second method is not subject to these restrictions, because it does not use Coq's dedicated mechanisms for corecursion but redefines them. Both methods amount to defining corecursive functions as limits of sequences of approximations, and both require a property of productiveness that, intuitively, means that for each input, an arbitrarily close approximation of the limit's corresponding output is eventually obtained. The methods are implemented in Coq and are illustrated with examples of corecursive functions that go beyond Coq's builtin capabilities. This gain in expressiveness is obtained by using a combination of axioms from Coq's standard library that enable reasoning in standard (i.e., not necessarily constructive) mathematics.

This work is also a collaboration with 2XS. It is currently submitted for publication.

Participants: Vlad Rusu, David Nowak (2XS).

7 Partnerships and cooperations

7.1 Participation in International Programs

Giuseppe Lipari participates in the PHC "Tassili" in collaboration with the University of Oran 1 (Algeria). The collaboration consists in the co-direction of two PhD theses:

- The PhD thesis of Chawki Benchehida, started on October 2018, and successfully defended on the 9/11/2021 at the Université de Lille, with title "Mapping Hard Real-Time Tasks on Network-on-Chip Manycore Architectures", co-directed by Giuseppe Lipari and Kamel Benhaoua (University of Oran 1).
- The PhD thesis of Ikram Senoussaoui, started on October 2019, co-directed by Giuseppe Lipari and Kamel Benahoua (University of Oran 1).

Participants: Giuseppe Lipari, Chawki Benchehida, Ikram Senoussaoui.

7.2 European initiatives

Giuseppe Lipari has participated to the submission of the "EDGEWARE" project proposal, a H2020 project on real-time edge computing.

Participants: Giuseppe Lipari.

7.3 National initiatives

Giuseppe Lipari has participated to the preparation and the submission of the PRCE ANR proposal SQUASH. The proposal is coordinated by Audrey Quedet (University of Nantes, principal investigator) and concerns the management and control of the energy consumption in IoT systems with real-time constraints.

Participants: Giuseppe Lipari.

8 Dissemination

8.1 Scientific events: organisation

- Giuseppe Lipari was Program Co-Chair of eRTNS 2021 conference:
- Giuseppe Lipari was Program Co-Chair of IEEE RTCSA 2021 conference;
- Patrick Baillot was Program Co-Chair of the LCC 2022 (Logic and Computational Complexity) workshop.

8.2 Member of the conference program committees

- DATE 2021, Julien Forget ;
- RTCSA 2021, Julien Forget ;
- RTNS 2021, Julien Forget ;

- ISPDC 2021, Giuseppe Lipari ;
- Working Formal Methods Symposium (FROM 2021), Vlad Rusu.

8.3 Teaching Responsabilities - Juries

- Giuseppe Lipari is *directeur d'études* of the « Internet des Objets » section of the *Master mention informatique* at the University of Lille. The IoT section is part of the Graduate Programme « Information and Knowledge Society » of the I-site of the university.
- Julien Forget was member of the PhD defence committee of Evariste Ntaryamira, 7/5/2021, Sorbonne Université ;
- Patrick Baillot was president of the PhD defence committee of Le Thanh Dung Nguyen, 3/12/2021, Université Sorbonne Paris Nord;
- Patrick Baillot was member of the PhD defence committee of Paul Gallot, 16/12/2021, Université de Lille;
- Giuseppe Lipari was member of the PhD defence committee of Jawher Jerray, Université de Sorbonne Paris Nord, 10/12/2021
- Giuseppe Lipari was member of the PhD defence committee of Agostino Mascitti, Scuola Superiore Sant'Anna, Pisa, Italy, 16/12/2021.

9 Scientific production

9.1 Publications of the year

International journals

[1] M. Pagani, A. Biondi, M. Marinoni, L. Molinari, G. Lipari and G. Buttazzo. 'A Linux-Based Support for Developing Real-Time Applications on Heterogeneous Platforms with Dynamic FPGA Reconfiguration'. In: *Future Generation Computer Systems* 129 (6th Oct. 2021), pp. 125–140. DOI: 10.1016/j.future.2021.11.007. URL: https://hal.archives-ouvertes.fr/hal-03512476.

Conferences without proceedings

[2] G. Lipari, F. Bouquillon and S. Niar. 'Improving CRPD Analysis for EDF Scheduling: Trading Speed for Precision'. In: The 37th ACM/SIGAPP Symposium On Applied Computing. Brno, Czech Republic, 18th Jan. 2022. DOI: 10.1145/3477314.3507027. URL: https://hal.archives-ouvertes.fr/hal-03531143.

Doctoral dissertations and habilitation theses

[3] C. Benchehida. 'Mapping Hard Real-Time Tasks on Network-on-Chip Manycore Architectures'. Université de Lille; Université d'Oran 1, 9th Nov. 2021. URL: https://hal-cnrs.archives-ouvertes.fr/tel-03514023.

Reports & preprints

[4] F. Fort and J. Forget. *Synchronous semantics of multi-mode multi-periodic systems*. 18th Jan. 2022. URL: https://hal.archives-ouvertes.fr/hal-03531360.

9.2 Cited publications

[5] L. Abeni, L. Palopoli, C. Scordino and G. Lipari. ‘Resource Reservations for General Purpose Applications’. In: *IEEE Trans. Ind. Informatics* 5.1 (2009), pp. 12–21. DOI: `10.1109/TII.2009.2013633`. URL: `https://doi.org/10.1109/TII.2009.2013633`.

[6] ANSYS. *SCADE Suite*. `http://www.esterel-technologies.com/products/scade-suite/`. 2018.

[7] C. Ballabriga, J. Forget, L. Gonnord, G. Lipari and J. Ruiz. ‘Static Analysis Of Binary Code With Memory Indirections Using Polyhedra’. In: *International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI’19)*. Lisbon, Portugal, Jan. 2019. URL: `https://hal.archives-ouvertes.fr/hal-01939659`.

[8] C. Ballabriga, J. Forget and G. Lipari. ‘Symbolic WCET Computation’. In: *ACM Transactions on Embedded Computing Systems (TECS)* 17.2 (Dec. 2017), pp. 1–26. DOI: `10.1145/3147413`. URL: `https://hal.archives-ouvertes.fr/hal-01665076`.

[9] S. Baruah. ‘Resource-Efficient Execution of Conditional Parallel Real-Time Tasks’. In: *Euro-Par 2018: Parallel Processing*. Cham: Springer International Publishing, 2018, pp. 218–231.

[10] E. Bini and G. Buttazzo. ‘The space of EDF deadlines: the exact region and a convex approximation’. In: *Real-Time Systems* 41.1 (2009), pp. 27–51.

[11] E. Bini, M. Di Natale and G. Buttazzo. ‘Sensitivity analysis for fixed-priority real-time systems’. In: *Real-Time Systems* 39.1-3 (2008), pp. 5–30.

[12] T. Bourke, L. Brun, P.-É. Dagand, X. Leroy, M. Pouzet and L. Rieg. ‘A formally verified compiler for Lustre’. In: *ACM SIGPLAN Notices*. Vol. 52. 6. ACM. 2017, pp. 586–601.

[13] R. Boutonnet and N. Halbwachs. ‘Disjunctive relational abstract interpretation for interprocedural program analysis’. In: *International Conference on Verification, Model Checking, and Abstract Interpretation*. Springer. 2019, pp. 136–159.

[14] G. Durrieu, M. Faugere, S. Girbal, D. G. Pérez, C. Pagetti and W. Puffitsch. ‘Predictable flight management system implementation on a multicore processor’. In: *Embedded Real Time Software (ERTS’14)*. 2014.

[15] A. Hamann, D. Dasari, S. Kramer, M. Pressler and F. Wurst. ‘Communication Centric Design in Complex Automotive Embedded Systems’. In: *29th Euromicro Conference on Real-Time Systems (ECRTS 2017)*. Dubrovnik, Croatia, 2017.

[16] X. Leroy. *The CompCert C verified compiler*. `http://compcert.inria.fr`. 2018.

[17] G. Lipari and S. K. Baruah. ‘Greedy reclamation of unused bandwidth in constant-bandwidth servers’. In: *12th Euromicro Conference on Real-Time Systems (ECRTS 2000), 19-21 June 2000, Stockholm, Sweden, Proceedings*. IEEE Computer Society, 2000, pp. 193–200. DOI: `10.1109/EMRTS.2000.854007`. URL: `https://doi.org/10.1109/EMRTS.2000.854007`.

[18] G. Lipari and E. Bini. ‘Resource Partitioning among Real-Time Applications’. In: *Proc. 15th Euromicro Conf. Real-Time Systems*. IEEE Computer Society, 2003, pp. 151–158. DOI: `10.1109/EMRTS.2003.1212738`.

[19] MathWorks. *Simulink*. `https://www.mathworks.com/products/simulink.html`. 2018.

[20] P. W. O’Hearn and D. J. Pym. ‘The logic of bunched implications’. In: *Bull. Symb. Log.* 5.2 (1999). DOI: `10.2307/421090`. URL: `https://doi.org/10.2307/421090`.

[21] C. Pagetti, J. Forget, F. Boniol, M. Cordovilla and D. Lesens. ‘Multi-task implementation of multi-periodic synchronous programs’. In: *Discrete Event Dynamic Systems* 21.3 (2011), pp. 307–338.

[22] C. Pagetti, J. Forget, H. Falk, D. Oehlert and A. Luppold. ‘Automated generation of time-predictable executables on multi-core’. In: *RTNS 2018*. Oct. 2018.

[23] R. Pellizzoni, E. Betti, S. Bak, G. Yao, J. Criswell, M. Caccamo and R. Kegley. ‘A predictable execution model for COTS-based embedded systems’. In: *Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE. 2011.

[24] P. Raymond. ‘A general approach for expressing infeasibility in implicit path enumeration technique’. In: *2014 International Conference on Embedded Software (EMSOFT)*. IEEE. 2014, pp. 1–9.

[25] J. Reed and B. C. Pierce. ‘Distance makes the types grow stronger: a calculus for differential privacy’. In: *ICFP 2010*. ACM, 2010. DOI: 10.1145/1863543.1863568. URL: https://doi.org/10.1145/1863543.1863568.

[26] A. Saifullah, D. Ferry, J. Li, K. Agrawal, C. Lu and C. D. Gill. ‘Parallel Real-Time Scheduling of DAGs’. In: *IEEE Transactions on Parallel and Distributed Systems* 25.12 (Dec. 2014), pp. 3242–3252. DOI: 10.1109/TPDS.2013.2297919.

[27] *The Coq proof assistant reference manual*. http://coq.inria.fr. 2021.

[28] *The Pip protokernel*. http://pip.univ-lille1.fr/. 2018.