Activity Report 2019

Project-Team TEA

Time, Events and Architectures

RESEARCH CENTER
Rennes - Bretagne-Atlantique

THEME
Embedded and Real-time Systems

Table of contents

Project-Team TEA

Creation of the Team: 2014 January 01, updated into Project-Team: 2015 January 01

Keywords:

Computer Science and Digital Science:

A1.2.5. - Internet of things
A1.2.7. - Cyber-physical systems
A1.5.2. - Communicating systems
A2.1.1. - Semantics of programming languages
A2.1.4. - Functional programming
A2.1.6. - Concurrent programming
A2.1.9. - Synchronous languages
A2.1.10. - Domain-specific languages
A2.2.1. - Static analysis
A2.2.4. - Parallel architectures
A2.3. - Embedded and cyber-physical systems
A2.3.1. - Embedded systems
A2.3.2. - Cyber-physical systems
A2.3.3. - Real-time systems
A2.4. - Formal method for verification, reliability, certification
A2.4.1. - Analysis
A2.4.2. - Model-checking
A2.4.3. - Proofs
A2.5. - Software engineering
A2.5.1. - Software Architecture & Design
A2.5.2. - Component-based Design
A4.4. - Security of equipment and software
A4.5. - Formal methods for security
A7.2. - Logic in Computer Science
A7.2.3. - Interactive Theorem Proving
A7.3. - Calculability and computability
A8.1. - Discrete mathematics, combinatorics
A8.3. - Geometry, Topology

Other Research Topics and Application Domains:

B5.1. - Factory of the future
B6.1.1. - Software engineering
B6.4. - Internet of things
B6.6. - Embedded systems

1. Team, Visitors, External Collaborators

Research Scientists

Jean-Pierre Talpin [Team leader, Inria, Senior Researcher, HDR]
Thierry Gautier [Inria, Researcher]
Rajesh Kumar Gupta [University of California San Diego, Inria Chair]
Shuvra Bhattacharyya [University of Maryland, Insa-Inria Chair]

PhD Students

Lucas Franceschino [Inria]
Stephane Kastenbaum [Mitsubishi Electric, from Oct 2019]
Simon Lunel [Inria, until Jan 2019]
Jean Joseph Marty [Inria]
Liangcong Zhang [China Scolarship Council]

Technical staff

Loic Besnard [CNRS SED, Senior Engineer]
Armelle Mozziconacci [CNRS, Project Assistant]

2. Overall Objectives

2.1. Introduction

An embedded architecture is an artifact of heterogeneous constituents and at the crossing of several design viewpoints: software, embedded in hardware, interfaced with the physical world. Time takes different forms when observed from each of these viewpoints: continuous or discrete, event-based or time-triggered: modeling and programming formalisms that represent software, hardware and physics significantly alter this perception of time. Therefore, time reasoning in system design is usually isolated to a specific design problem: simulation, profiling, performance, scheduling, parallelization, simulation. The aim of project-team TEA is to define conceptually unified frameworks for reasoning on composition and integration in cyber-physical system design, and to put this reasoning to practice by revisiting analysis and synthesis issues in real-time system design with soundness and compositionality gained from formalization.

2.2. Context

In the construction of complex systems, information technology (IT) has become a central force of revolutionary changes, driven by the exponential increase of computational power. In the field of telecommunication, IT provides the necessary basis for systems of networked distributed applications. In the field of control engineering, IT provides the necessary basis for embedded control applications. The combination of telecommunication and embedded systems into networked embedded systems opens up a new range of systems, capable of providing more intelligent functionalities, thanks to information and communication (ICT). Networked embedded systems have revolutionized several application domains: energy networks, industrial automation and transport systems.

20th-century science and technology brought us effective methods and tools for designing both computational and physical systems, such as for instance Simulink and Matlab. But the design of cyber-physical systems (CPS) is much more than the union of those two fields. Traditionally, information scientists only have a hazy notion of requirements imposed by the physical environment of computers. Similarly, mechanical, civil, and chemical engineers view computers strictly as devices executing algorithms. CPS design is, to date, mostly executed in this ad-hoc manner, without sound, mathematically grounded, integrative methodology. A new science of CPS design will allow to create machines with complex dynamics and high control reliability, and apply to new industries and applications, such as IoT or edge devices, in a reliable and economically efficient way. Progress requires nothing less than the construction of a new science and technology foundation for CPS that is simultaneously physical and computational.

2.3. Motivations

Beyond the buzzword, a CPS is an ubiquitous object of our everyday life. CPSs have evolved from individual independent units (e.g an ABS brake) to more and more integrated networks of units, which may be aggregated into larger components or sub-systems. For example, a transportation monitoring network aggregates monitored stations and trains through a large scale distributed system with relatively high latency. Each individual train is being controlled by a train control network, each car in the train has its own real-time bus to control embedded devices. More and more, CPSs are mixing real-time low latency technology with higher latency distributed computing technology.

In the past 15 years, CPS development has moved towards Model Driven Engineering (MDE). With MDE methodology, first all requirements are gathered together with use cases, then a model of the system is built (sometimes several models) that satisfy the requirements. There are several modeling formalisms that have appeared in the past ten years with more or less success. The most successful are the *executable* models [1] [2] [3], i.e., models that can be simulated, exercised, tested and validated. This approach can be used for both software and hardware.

A common feature found in CPSs is the ever presence of concurrency and parallelism in models. Large systems are increasingly mixing both types of concurrency. They are structured hierarchically and comprise multiple synchronous devices connected by buses or networks that communicate asynchronously. This led to the advent of so-called GALS (Globally Asynchronous, Locally Synchronous) models, or PALS (Physically Asynchronous, Logically Synchronous) systems, where reactive synchronous objects are communicating asynchronously. Still, these infrastructures, together with their programming models, share some fundamental concerns: parallelism and concurrency synchronization, determinism and functional correctness, scheduling optimality and calculation time predictability.

Additionally, CPSs monitor and control real-world processes, the dynamics of which are usually governed by physical laws. These laws are expressed by physicists as mathematical equations and formulas. Discrete CPS models cannot ignore these dynamics, but whereas the equations express the continuous behavior usually using real numbers (irrational) variables, the models usually have to work with discrete time and approximate floating point variables.

2.4. Challenges

A cyber-physical, or reactive, or embedded system is the integration of heterogeneous components originating from several design viewpoints: reactive software, some of which is embedded in hardware, interfaced with the physical environment through mechanical parts. Time takes different forms when observed from each of these viewpoints: it is discrete and event-based in software, discrete and time-triggered in hardware, continuous in mechanics or physics. Design of CPS often benefits from concepts of multiform and logical time(s) for their natural description. High-level formalisms used to model software, hardware and physics additionally alter this perception of time quite significantly.

In model-based system design, time is usually abstracted to serve the purpose of one of many design tasks: verification, simulation, profiling, performance analysis, scheduling analysis, parallelization, distribution, or virtual prototyping. For example in non-real-time commodity software, timing abstraction such as number of instructions and algorithmic complexity is sufficient: software will run the same on different machines, except slower or faster. Alternatively, in cyber-physical systems, multiple recurring instances of meaningful events may create as many dedicated logical clocks, on which to ground modeling and design practices.

Time abstraction increases efficiency in event-driven simulation or execution (i.e SystemC simulation models try to abstract time, from cycle-accurate to approximate-time, and to loosely-time), while attempting to retain functionality, but without any actual guarantee of valid accuracy (responsibility is left to the model designer). Functional determinism (a.k.a. conflict-freeness in Petri Nets, monotonicity in Kahn PNs, confluence in

[1] *Matlab/Simulink*, https://fr.mathworks.com/products/simulink.html
[2] *Ptolemy*, http://ptolemy.eecs.berkeley.edu
[3] *SysML*, http://www.uml-sysml.org

Milner's CCS, latency-insensitivity and elasticity in circuit design) allows for reducing to some amount the problem to that of many schedules of a single self-timed behavior, and time in many systems studies is partitioned into models of computation and communication (MoCCs). Multiple, multiform time(s) raises the question of combination, abstraction or refinement between distinct time bases. The question of combining continuous time with discrete logical time calls for proper discretization in simulation and implementation. While timed reasoning takes multiple forms, there is no unified foundation to reasoning about multi-form time in system design.

The objective of project-team TEA is henceforth to define formal models for timed quantitative reasoning, composition, and integration in embedded system design. Formal time models and calculi should allow us to revisit common domain problems in real-time system design, such as time predictability and determinism, memory resources predictability, real-time scheduling, mixed-criticality and power management; yet from the perspective gained from inter-domain timed and quantitative abstraction or refinement relations. A regained focus on fundamentals will allow to deliver better tooled methodologies for virtual prototyping and integration of embedded architectures.

3. Research Program

3.1. Previous Works

The challenges of team TEA support the claim that sound Cyber-Physical System design (including embedded, reactive, and concurrent systems altogether) should consider multi-form time models as a central aspect. In this aim, architectural specifications found in software engineering are a natural focal point to start from. Architecture descriptions organize a system model into manageable components, establish clear interfaces between them, collect domain-specific constraints and properties to help correct integration of components during system design. The definition of a formal design methodology to support heterogeneous or multi-form models of time in architecture descriptions demands the elaboration of sound mathematical foundations and the development of formal calculi and methods to instrument them.

System design based on the "synchronous paradigm" has focused the attention of many academic and industrial actors on abstracting non-functional implementation details from system design. This elegant design abstraction focuses on the logic of interaction in reactive programs rather than their timed behavior, allowing to secure functional correctness while remaining an intuitive programming model for embedded systems. Yet, it corresponds to embedded technologies of single cores and synchronous buses from the 90s, and may hardly cover the semantic diversity of distribution, parallelism, heterogeneity, of cyber-physical systems found in 21st century Internet-connected, true-time^{TM}-synchronized clouds, of tomorrow's grids.

By contrast with a synchronous hypothesis, yet from the same era, the polychronous MoCC is inherently capable of describing multi-clock abstractions of GALS systems. Polychrony is implemented in the data-flow specification language Signal, available in the Eclipse project POP [4] and in the CCSL standard [5] available from the TimeSquare project. Both provide tooled infrastructures to refine high-level specifications into real-time streaming applications or locally synchronous and globally asynchronous systems, through a series of model analysis, verification, and synthesis services. These tool-supported refinement and transformation techniques can assist the system engineer from the earliest design stages of requirement specification to the latest stages of synthesis, scheduling and deployment. These characteristics make polychrony much closer to the required semantic for compositional, refinement-based, architecture-driven, system design.

While polychrony was a step ahead of the traditional synchronous hypothesis, CCSL is a leap forward from synchrony and polychrony. The essence of CCSL is "multi-form time" toward addressing all of the domain-specific physical, electronic and logical aspects of cyber-physical system design.

[4] *Polychrony on Polarsys*, https://www.polarsys.org/projects/polarsys.pop
[5] *Clock Constraints in UML/MARTE CCSL*. C. André, F. Mallet. RR-6540. Inria, 2008. http://hal.inria.fr/inria-00280941

3.2. Timed Modeling

To formalize timed semantics for system design, we shall rely on algebraic representations of time as clocks found in previous works and introduce a paradigm of "time system" (types that represent time) in a way reminiscent to CCSL. Just as a type system abstracts data carried along operations in a program, a time system abstracts the causal interaction of that program module or hardware element with its environment, its pre and post conditions, its assumptions and guarantees, either logical or numerical, discrete or continuous. Some fundamental concepts of the time systems we envision are present in the clock calculi found in data-flow synchronous languages like Signal or Lustre, yet bound to a particular model of timed concurrency.

In particular, the principle of refinement type systems [6], is to associate information (data-types) inferred from programs and models with properties pertaining, for instance, to the algebraic domain on their value, or any algebraic property related to its computation: effect, memory usage, pre-post condition, value-range, cost, speed, time, temporal logic [7]. Being grounded on type and domain theories, a time system should naturally be equipped with program analysis techniques based on type inference (for data-type inference) or abstract interpretation (for program properties inference) to help establish formal relations between heterogeneous component “types”. Just as a time calculus may formally abstract timed concurrent behaviors of system components, timed relations (abstraction and refinement) represent interaction among components.

Scalability requires the use of assume-guarantee reasoning to allow modularity and to facilitate composition by behavioral sub-typing, in the spirit of the (static) contract-based formalism proposed by Passerone et al. [8]. Verification problems encompassing heterogeneously timed specifications are common and of great variety: checking correctness between abstract (e.g. the synchronous hypothesis) and concrete time models (e.g. real-time architectures) relates to desynchronisation (from synchrony to asynchrony) and scheduling analysis (from synchronous data-flow to hardware). More generally, they can be perceived from heterogeneous timing viewpoints (e.g. mapping a synchronous-time software on a real-time middle-ware or hardware).

This perspective demands capabilities to use abstraction and refinement mechanisms for time models (using simulation, refinement, bi-simulation, equivalence relations) but also to prove more specific properties (synchronization, determinism, endochrony). All this formalization effort will allow to effectively perform the tooled validation of common cross-domain properties (e.g. cost v.s. power v.s. performance v.s. software mapping) and tackle problems such as these integrating constraints of battery capacity, on-board CPU performance, available memory resources, software schedulability, to logical software correctness and plant controllability.

3.3. Modeling Architectures

To address the formalization of such cross-domain case studies, modeling the architecture formally plays an essential role. An architectural model represents components in a distributed system as boxes with well-defined interfaces, connections between ports on component interfaces, and specifies component properties that can be used in analytical reasoning about the model. Several architectural modeling languages for embedded systems have emerged in recent years, including the SAE AADL [9], SysML [10], UML MARTE [11].

In system design, an architectural specification serves several important purposes. First, it breaks down a system model into components of manageable size and complexity, to establish clear interfaces between components. In this way, complexity becomes manageable by hiding details that are not relevant at a given level of abstraction. Clear, formally defined, component interfaces allow us to avoid integration problems at the implementation phase. Connections between components, which specify how components interact with each other, help propagate the effects of a change in one component to the linked components.

[6] *Abstract Refinement Types*. N. Vazou, P. Rondon, and R. Jhala. European Symposium on Programming. Springer, 2013.
[7] *LTL types FRP*. A. Jeffrey. Programming Languages meets Program Verification.
[8] *A contract-based formalism for the specification of heterogeneous systems*. L. Benvenistu, et al. FDL, 2008
[9] *Architecture Analysis and Design Language*, AS-5506. SAE, 2004. http://standards.sae.org/as5506b
[10] *System modeling Language*. OMG, 2007. http://www.omg.org/spec/SysML
[11] *UML Profile for MARTE*. OMG, 2009. http://www.omg.org/spec/MARTE

Most importantly, an architectural model is a repository to share knowledge about the system being designed. This knowledge can be represented as requirements, design artifacts, component implementations, held together by a structural backbone. Such a repository enables automatic generation of analytical models for different aspects of the system, such as timing, reliability, security, performance, energy, etc. Since all the models are generated from the same source, the consistency of assumptions w.r.t. guarantees, of abstractions w.r.t. refinements, used for different analyses becomes easier, and can be properly ensured in a design methodology based on formal verification and synthesis methods.

Related works in this aim, and closer in spirit to our approach (to focus on modeling time) are domain-specific languages such as Prelude [12] to model the real-time characteristics of embedded software architectures. Conversely, standard architecture description languages could be based on algebraic modeling tools, such as interface theories with the ECDAR tool [13].

In project TEA, it takes form by the normalization of the AADL standard's formal semantics and the proposal of a time specification annex in the form of related standards, such as CCSL, to model concurrency, time and physical properties, and PSL, to model timed traces.

3.4. Scheduling Theory

Based on sound formalization of time and CPS architectures, real-time scheduling theory provides tools for predicting the timing behavior of a CPS which consists of many interacting software and hardware components. Expressing parallelism among software components is a crucial aspect of the design process of a CPS. It allows for efficient partition and exploitation of available resources.

The literature about real-time scheduling [14] provides very mature schedulability tests regarding many scheduling strategies, preemptive or non-preemptive scheduling, uniprocessor or multiprocessor scheduling, etc. Scheduling of data-flow graphs has also been extensively studied in the past decades.

A milestone in this prospect is the development of abstract affine scheduling techniques [15]. It consists, first, of approximating task communication patterns (e.g. between Safety-Critical Java threads) using cyclo-static data-flow graphs and affine functions. Then, it uses state of the art ILP techniques to find optimal schedules and to concretize them as real-time schedules in the program implementations [16] [17].

Abstract scheduling, or the use of abstraction and refinement techniques in scheduling borrowed to the theory of abstract interpretation [18] is a promising development toward tooled methodologies to orchestrate thousands of heterogeneous hardware/software blocks on modern CPS architectures (just consider modern cars or aircrafts). It is an issue that simply defies the state of the art and known bounds of complexity theory in the field, and consequently requires a particular focus.

To develop the underlying theory of this promising research topic, we first need to deepen the theoretical foundation to establish links between scheduling analysis and abstract interpretation. A theory of time systems would offer the ideal framework to pursue this development. It amounts to representing scheduling constraints, inferred from programs, as types or contract properties. It allows to formalize the target time model of the scheduler (the architecture, its middle-ware, its real-time system) and defines the basic concepts to verify assumptions made in one with promises offered by the other: contract verification or, in this case, synthesis.

3.5. Verified programming for system design

The IoT is a network of devices that sense, actuate and change our immediate environment. Against this fundamental role of sensing and actuation, design of edge devices often considers actions and event timings to

[12] *The Prelude language*. LIFL and ONERA, 2012. http://www.lifl.fr/~forget/prelude.html

[13] *PyECDAR, timed games for timed specifications*. Inria, 2013. https://project.inria.fr/pyecdar

[14] *A survey of hard real-time scheduling for multiprocessor systems*. R. I. Davis and A. Burns. *ACM Computing Survey* 43(4), 2011.

[15] *Buffer minimization in EDF scheduling of data-flow graphs*. A. Bouakaz and J.-P. Talpin. LCTES, ACM, 2013.

[16] *ADFG for the synthesis of hard real-time applications*. A. Bouakaz, J.-P. Talpin, J. Vitek. ACSD, IEEE, June 2012.

[17] *Design of SCJ Level 1 Applications Using Affine Abstract Clocks*. A. Bouakaz and J.-P. Talpin. SCOPES, ACM, 2013.

[18] *La vérification de programmes par interprétation abstraite*. P. Cousot. Séminaire au Collège de France, 2008.

be primarily software implementation issues: programming models for IoT abstract even the most rudimentary information regarding timing, sensing and the effects of actuation. As a result, applications programming interfaces (API) for IoT allow wiring systems fast without any meaningful assertions about correctness, reliability or resilience.

We make the case that the "API glue" must give way to a logical interface expressed using contracts or refinement types. Interfaces can be governed by a calculus – a refinement type calculus – to enable reasoning on time, sensing and actuation, in a way that provides both deep specification refinement, for mechanized verification of requirements, and multi-layered abstraction, to support compositionality and scalability, from one end of the system to the other.

Our project seeks to elevate the “function as type” paradigm to that of “system as type”: to define a refinement type calculus based on concepts of contracts for reasoning on networked devices and integrate them as cyber-physical systems [19]. An invited paper [20] outlines our progress with respect to this aim and plans towards building a verified programming environment for networked IoT devices: we propose a type-driven approach to verifying and building safe and secure IoT applications.

Accounting for such constrains in a more principled fashion demands reasoning about the composition of all the software and hardware components of the application. Our proposed framework takes a step in this direction by (1) using refinement types to make make physical constraints explicit and (2) imposing an event-driven programming discipline to simplify the reasoning of system-wide properties to that of an event queue. In taking this approach, our approach would make it possible for a developer to build a verified IoT application by ensuring that a well-typed program cannot violate the physical constraints of its architecture and environment.

4. Application Domains

4.1. Automotive and Avionics

From our continuous collaboration with major academic and industrial partners through projects TOPCASED, OPENEMBEDD, SPACIFY, CESAR, OPEES, P and CORAIL, our experience has primarily focused on the aerospace domain. The topics of time and architecture of team TEA extend to both avionics and automotive. Yet, the research focuses on time in team TEA is central in any aspect of, cyber-physical, embedded system design in factory automation, automotive, music synthesis, signal processing, software radio, circuit and system on a chip design; many application domains which, should more collaborators join the team, would definitely be worth investigating.

Multi-scale, multi-aspect time modeling, analysis and software synthesis will greatly contribute to architecture modeling in these domains, with applications to optimized (distributed, parallel, multi-core) code generation for avionics (project Corail with Thales avionics, section 8) as well as modeling standards, real-time simulation and virtual integration in automotive (project with Toyota ITC, section 8).

Together with the importance of open-source software, one of these projects, the FUI Project P (section 8), demonstrated that a centralized model for system design could not just be a domain-specific programming language, such as discrete Simulink data-flows or a synchronous language. Synchronous languages implement a fixed model of time using logical clocks that are abstraction of time as sensed by software. They correspond to a fixed viewpoint in system design, and in a fixed hardware location in the system, which is not adequate to our purpose and must be extended.

[19]Refinement types for system design. Jean-Pierre Talpin. FDL’18 keynote.

[20]Steps toward verified programming of embedded computing systems. Jean-Pierre Talpin, Jean-Joseph Marty, Deian Stefan, Shravan Nagarayan, Rajesh Gupta, DATE’18.

In project P, we first tried to define a centralized model for importing discrete-continuous models onto a simplified implementation of SIMULINK: P models. Certified code generators would then be developed from that format. Because this does not encompass all aspects being translated to P, the P meta-model is now being extended to architecture description concepts (of the AADL) in order to become better suited for the purpose of system design. Another example is the development of System modeler on top of SCADE, which uses the more model-engineering flavored formalism SysML to try to unambiguously represent architectures around SCADE modules.

An abstract specification formalism, capable of representing time, timing relations, with which heterogeneous models can be abstracted, from which programs can be synthesized, naturally appears better suited for the purpose of virtual prototyping. RT-Builder, based on the data-flow language Signal and developed by TNI, was industrially proven and deployed for that purpose at Peugeot. It served to develop the virtual platform simulating all on-board electronics of PSA cars. This 'hardware in the loop" simulator was used to test equipments supplied by other manufacturers for virtual prototyping of cars. In the advent of the related automotive standard, RT-Builder then became AUTOSAR-Builder.

4.2. Factory Automation

In collaboration with Mitsubishi R&D, we explore another application domain where time and domain heterogeneity are prime concerns: factory automation. In factory automation alone, a system is conventionally built from generic computing modules: PLCs (Programmable Logic Controllers), connected to the environment with actuators and detectors, and linked to a distributed network. Each individual, physically distributed, PLC module must be timely programmed to perform individually coherent actions and fulfill the global physical, chemical, safety, power efficiency, performance and latency requirements of the whole production chain. Factory chains are subject to global and heterogeneous (physical, electronic, functional) requirements whose enforcement must be orchestrated for all individual components.

Model-based analysis in factory automation emerges from different scientific domains and focus on different CPS abstractions that interact in subtle ways: logic of PLC programs, real-time electro-mechanical processing, physical and chemical environments. This yields domain communication problems that render individual domain analysis useless. For instance, if one domain analysis (e.g. software) modifies a system model in a way that violates assumptions made by another domain (e.g. chemistry) then the detection of its violation may well be impossible to explain to either the software or chemistry experts. As a consequence, cross-domain analysis issues are discovered very late during system integration and lead to costly fixes. This is particularly prevalent in multi-tier industries, such as avionic, automotive, factories, where systems are prominently integrated from independently-developed parts.

5. Highlights of the Year

5.1. Highlights of the Year

Loïc Besnard was promoted to the rank of Senior Engineer Exceptional Class by CNRS, acknowledging his remarkable career of research engineer as principal developer of Signal and Polychrony, as project manager and integrator with project teams EPATR (Signal), ESPRESSO (Polychrony), TEA (ADFG) and PACAP (Heptane).

6. New Software and Platforms

6.1. ADFG

Affine data-flow graphs schedule synthesizer

KEYWORDS: Code generation - Scheduling - Static program analysis

FUNCTIONAL DESCRIPTION: ADFG is a synthesis tool of real-time system scheduling parameters: ADFG computes task periods and buffer sizes of systems resulting in a trade-off between throughput maximization and buffer size minimization. ADFG synthesizes systems modeled by ultimately cyclo-static dataflow (UCSDF) graphs, an extension of the standard CSDF model.

Knowing the WCET (Worst Case Execute Time) of the actors and their exchanges on the channels, ADFG tries to synthezise the scheduler of the application. ADFG offers several scheduling policies and can detect unschedulable systems. It ensures that the real scheduling does not cause overflows or underflows and tries to maximize the throughput (the processors utilization) while minimizing the storage space needed between the actors (i.e. the buffer sizes).

Abstract affine scheduling is first applied on the dataflow graph, that consists only of periodic actors, to compute timeless scheduling constraints (e.g. relation between the speeds of two actors) and buffering parameters. Then, symbolic schedulability policies analysis (i.e., synthesis of timing and scheduling parameters of actors) is applied to produce the scheduller for the actors.

ADFG, initially defined to synthesize real-time schedulers for SCJ/L1 applications, may be used for scheduling analysis of AADL programs.

- Authors: Thierry Gautier, Jean-Pierre Talpin, Adnan Bouakaz, Alexandre Honorat and Loïc Besnard
- Contact: Loïc Besnard

6.2. POLYCHRONY

KEYWORDS: Code generation - AADL - Proof - Optimization - Multi-clock - GALS - Architecture - Cosimulation - Real time - Synchronous Language

FUNCTIONAL DESCRIPTION: Polychrony is an Open Source development environment for critical/embedded systems. It is based on Signal, a real-time polychronous data-flow language. It provides a unified model-driven environment to perform design exploration by using top-down and bottom-up design methodologies formally supported by design model transformations from specification to implementation and from synchrony to asynchrony. It can be included in heterogeneous design systems with various input formalisms and output languages. The Polychrony tool-set provides a formal framework to: validate a design at different levels, by the way of formal verification and/or simulation, refine descriptions in a top-down approach, abstract properties needed for black-box composition, compose heterogeneous components (bottom-up with COTS), generate executable code for various architectures. The Polychrony tool-set contains three main components and an experimental interface to GNU Compiler Collection (GCC):

* The Signal toolbox, a batch compiler for the Signal language, and a structured API that provides a set of program transformations. Itcan be installed without other components and is distributed under GPL V2 license.

* The Signal GUI, a Graphical User Interface to the Signal toolbox (editor + interactive access to compiling functionalities). It can be used either as a specific tool or as a graphical view under Eclipse. It has been transformed and restructured, in order to get a more up-to-date interface allowing multi-window manipulation of programs. It is distributed under GPL V2 license.

* The POP Eclipse platform, a front-end to the Signal toolbox in the Eclipse environment. It is distributed under EPL license.

- Participants: Loïc Besnard, Paul Le Guernic and Thierry Gautier
- Partners: CNRS - Inria
- Contact: Loïc Besnard
- URL: https://www.polarsys.org/projects/polarsys.pop

6.3. Polychrony AADL2SIGNAL

KEYWORDS: Real-time application - Polychrone - Synchronous model - Polarsys - Polychrony - Signal - AADL - Eclipse - Meta model

FUNCTIONAL DESCRIPTION: This polychronous MoC has been used previously as semantic model for systems described in the core AADL standard. The core AADL is extended with annexes, such as the Behavior Annex, which allows to specify more precisely architectural behaviors. The translation from AADL specifications into the polychronous model should take into account these behavior specifications, which are based on description of automata.

For that purpose, the AADL state transition systems are translated as Signal automata (a slight extension of the Signal language has been defined to support the model of polychronous automata).

Once the AADL model of a system transformed into a Signal program, one can analyze the program using the Polychrony framework in order to check if timing, scheduling and logical requirements over the whole system are met.

We have implemented the translation and experimented it using a concrete case study, which is the AADL modeling of an Adaptive Cruise Control (ACC) system, a highly safety-critical system embedded in recent cars.

- Participants: Huafeng Yu, Loïc Besnard, Paul Le Guernic, Thierry Gautier and Yue Ma
- Partner: CNRS
- Contact: Loïc Besnard
- URL: http://www.inria.fr/equipes/tea

6.4. POP

Polychrony on Polarsys

KEYWORDS: Synchronous model - Model-driven engineering

FUNCTIONAL DESCRIPTION: The Eclipse project POP is a model-driven engineering front-end to our open-source toolset Polychrony. a major achievement of the ESPRESSO (and now TEA) project-team. The Eclipse project POP is a model-driven engineering front-end to our open-source toolset Polychrony. It was finalised in the frame of project OPEES, as a case study: by passing the POLARSYS qualification kit as a computer aided simulation and verification tool. This qualification was implemented by CS Toulouse in conformance with relevant generic (platform independent) qualification documents. Polychrony is now distributed by the Eclipse project POP on the platform of the POLARSYS industrial working group. Team TEA aims at continuing its dissemination to academic partners, as to its principles and features, and industrial partners, as to the services it can offer.

Project POP is composed of the Polychrony tool set, under GPL license, and its Eclipse framework, under EPL license. SSME (Syntactic Signal-Meta under Eclipse), is the meta-model of the Signal language implemented with Eclipse/Ecore. It describes all syntactic elements specified in Signal Reference Manual[21]: all Signal operators (e.g. arithmetic, clock synchronization), model (e.g. process frame, module), and construction (e.g. iteration, type declaration). The meta-model primarily aims at making the language and services of the Polychrony environment available to inter-operation and composition with other components (e.g. AADL, Simulink, GeneAuto, P) within an Eclipse-based development tool-chain. Polychrony now comprises the capability to directly import and export Ecore models instead of textual Signal programs, in order to facilitate interaction between components within such a tool-chain. The download site for project POP has opened in 2015 at https://www.polarsys.org/projects/polarsys.pop. It should be noted that the Eclipse Foundation does not host code under GPL license. So, the Signal toolbox useful to compile Signal code from Eclipse is hosted on our web server.

- Participants: Jean-Pierre Talpin, Loïc Besnard, Paul Le Guernic and Thierry Gautier
- Contact: Loïc Besnard
- URL: https://www.polarsys.org/projects/polarsys.pop

[21]

SIGNAL V4-Inria version: Reference Manual. Besnard, L., Gautier, T. and Le Guernic, P. http://www.irisa.fr/espresso/Polychrony, 2010

6.5. Sigali

FUNCTIONAL DESCRIPTION: Sigali is a model-checking tool that operates on ILTS (Implicit Labeled Transition Systems, an equational representation of an automaton), an intermediate model for discrete event systems. It offers functionalities for verification of reactive systems and discrete controller synthesis. The techniques used consist in manipulating the system of equations instead of the set of solutions, which avoids the enumeration of the state space. Each set of states is uniquely characterized by a predicate and the operations on sets can be equivalently performed on the associated predicates. Therefore, a wide spectrum of properties, such as liveness, invariance, reachability and attractivity, can be checked. Algorithms for the computation of predicates on states are also available. Sigali is connected with the Polychrony environment (Tea project-team) as well as the Matou environment (VERIMAG), thus allowing the modeling of reactive systems by means of Signal Specification or Mode Automata and the visualization of the synthesized controller by an interactive simulation of the controlled system.

- Contact: Hervé Marchand

7. New Results

7.1. ADFG: Affine data-flow graphs scheduler synthesis

Participants: Loïc Besnard, Thierry Gautier, Jean-Pierre Talpin, Shuvra Bhattacharyya, Alexandre Honorat, Hai Nam Tran.

ADFG (Affine DataFlow Graph) synthesizes scheduling parameters for real-time systems modeled as synchronous data flow (SDF), cyclo-static dataflow (CSDF), and ultimately cyclo-static dataflow (UCSDF) graphs. It aims at mitigating the trade-off between throughput maximization and total buffer size minimization. The synthesizer inputs are a graph which describes tasks by their Worst Case Execution Time (WCET), and directed buffers connecting tasks by their data production and consumption rates; the number of processors in the target system and the real-time scheduling synthesis algorithm to be used. The outputs are synthesized scheduling parameters such as tasks periods, offsets, processor bindings, priorities, buffer initial markings and buffer sizes. ADFG was originally implemented by Adnan Bouakaz [22]. It is now being collaboratively developed with team Tea, Hai Nam Tran (UBO) Alexandre Honorat (INSA) and Shuvra Bhattacharyya (UMD/INSA/Inria).

ADFG is extended to support automated code generation of the computed buffer sizes and scheduling parameters for dataflow applications that are implemented in the Lightweight Dataflow Environment (LIDE) [23]. LIDE is a flexible, lightweight design environment that allows designers to experiment with dataflow-based implementations directly. LIDE actors and buffers (FIFOs) can be initialized with parameters, including buffer sizes. The usage of LIDE allows a systematic way to instantiate dataflow graphs with the buffer size parameters computed by ADFG.

Actor models and scheduling algorithms in ADFG have been extended to investigate the contention-aware scheduling problem on multi/many-core architectures. The problem we tackled is that the scheduler synthesis for these platforms must account for the non-negligible delay due to shared memory accesses. We exploited the deterministic communications exposed in SDF graphs to account for the contention and further optimize the synthesized schedule. Two solutions are proposed and implemented in ADFG: contention-aware and contention-free scheduling synthesis. In other words, we either take into account the contention and synthesize a contention-aware schedule or find a one that results in no contention.

[22] Real-Time Scheduling of Dataflow Graphs. A. Bouakaz. Ph.D. Thesis, University of Rennes 1, 2013.

[23] S. Lin, Y. Liu, K. Lee, L. Li, W. Plishker, and S. S. Bhattacharyya. 2017. The DSPCAD framework for modeling and synthesis of signal processing systems. Handbook of Hardware/Software Codesign (2017), 1185–1219.

ADFG is extended to apply a transformation known as partial expansion graphs (PEG). This transformation can be applied as a pre-processing stage to improve the exploitation of data parallelism in SDF graphs on parallel platforms. In contrast to the classical approaches of transforming SDF graphs into equivalent homogeneous forms, which could lead to an exponential increase in the number of actors and excessive communication overhead, PEG-based approaches allow the designer to control the degree to which each actor is expanded. A PEG algorithm that employs cyclo-static data flow techniques is developed in ADFG. Compared to existing PEG-based approach, our solution requires neither buffer managers nor split-join actors to coordinate data production and consumption rates. This allows us to reduce the number of added actors and communication overhead in the expanded graphs.

7.2. Parallel Composition and Modular Verification of Computer Controlled Systems in Differential Dynamic Logic

Participants: Jean-Pierre Talpin, Benoit Boyer, David Mentre, Simon Lunel, Stefan Mitsch.

The primary goal of our project, in collaboration with Mitsubishi Electronics Research Centre Europe (MERCE), is to ensure correctness-by-design in realistic cyber-physical systems, i.e., systems that mix software and hardware in a physical environment, e.g., Mitsubishi factory automation lines or water-plant factory. To achieve that, we develop a verification methodology based on the decomposition of systems into components enhanced with compositional contract reasoning.

The work of A. Platzer on Differential Dynamic Logic ($d\mathcal{L}$) held our attention [24]. This formalism is built upon the Dynamic Logic of V. Pratt and augmented with the possibility of expressing Ordinary Differential Equations (ODEs). Combined with the ability of Dynamic Logic to specify and verify hybrid programs, $d\mathcal{L}$ is particularly adapted to model cyber-physical systems. The proof system associated with the logic is implemented into the theorem prover KeYmaera X. Aimed toward automation, it is a promising tool to spread formal methods in industry.

Computer-Controlled Systems (CCS) are a subclass of hybrid systems where the periodic relation of control components to time is of paramount importance. Since they additionally are at the heart of many safety-critical devices, it is of primary importance to correctly model such systems and to ensure they function correctly according to safety requirements. Differential dynamic logic $d\mathcal{L}$ is a powerful logic to model hybrid systems and to prove their correctness. We contributed a compositional modeling and reasoning framework to $d\mathcal{L}$ that separates models into components with timing guarantees, such as reactivity of controllers and controllability of continuous dynamics. Components operate in parallel, with coarse-grained interleaving, periodic execution and communication. We present techniques to automate system safety proofs from isolated, modular, and possibly mechanized proofs of component properties parameterized with timing characteristics.

7.3. Multithreaded code generation for process networks

Participants: Loïc Besnard, Thierry Gautier.

As part of an in-depth comparison of process models, we have recently revisited the relation between the model of asynchronous dataflow represented by Kahn Process Networks (KPNs) and that of synchronous dataflow represented by the polychronous model of computation. In particular, we have precisely described in which conditions polychronous programs can be seen as KPNs. In this context, we have considered different cases of process networks, including so-called “polyendochronous processes”. Under some conditions expressed by clock equation systems, (networks of) processes exhibiting polyhierarchies of clocks are polyendochronous and, as compositions of endochronous processes, may be seen as KPNs.

Based on this characterization, we have developed in the open-source Polychrony toolset a new strategy of code generation for such (polyendochronous) process networks. Typically, after the clock calculus, a program P is organized as a composition of processes, $P = (|\ P1 \mid P2 \mid ... \mid Pn\ |)$, each one structured around a clock tree. When P is characterized as polyendochronous, it contains generally clock constraints such as $Clk1 = Clk2$, with $Clk1$ being a clock in the subtree corresponding to $P1$ and $Clk2$ a clock in the subtree corresponding to $P2$.

[24] *Differential Dynamic Logic for Hybrid Systems*, André Platzer, http://symbolaris.com/logic/dL.html

Such a constraint induces a synchronization between two parts ($P1$, $P2$) of the program when $Clk1$ or $Clk2$ occurs. The principle of the code generation for polyendochronous processes is based on the existing distributed code generation, but with the additional resynchronization of parts of the application induced by the constraints on clocks ($Clk1$, $Clk2$) not placed in the same clock trees. For distributed code generation, it is considered that each (clock) hierarchy will run on a specific processor. In this case, the purpose is mainly to partition the application, and the processors will be virtual ones.

The code generation of each partition consists in the definition of several tasks: one task per cluster (a cluster being a subpart that may be executed as soon as its inputs are available, without any communication with the external world); one task per input/output of the partition; one task for the cluster of state variables; one task that manages the steps. Synchronization between these tasks is obtained by semaphores (one semaphore per task). This code generation technique for the class of networks called “polyendochronous processes” has been added in the Polychrony toolset (http://polychrony.inria.fr) and a paper describing the comparison of process models is currently in submission.

7.4. Type theory for modular static analysis of system programs

Participants: Lucas Franceschino, Jean-Pierre Talpin, David Pichardie.

This Ph.D. project is about formal verification, with system programming applications in mind. Formal methods are essential for safety-critical software (i.e. transport and aeronautic industry). In the same time, more and more programming languages with a strong type system arise (such as Haskell, Rust, ML, Coq, F*, Idris...).

Formal methods come in different flavors: type theory, abstract interpretation, refinement types. Each of these "flavors" are both theoretical fields and are also being implemented concretely: *Astrée* ou *Verasco* for abstract interpretation, *Coq*, *Agda*, *F** or *Idris* dependent types, and *Liquid Haskell* for refinement types.

Our approach consists in positioning ourselves between type theory and abstract interpretation, and to leverage the power of both. The main intuition behind this idea is that abstract interpretation, suffering from expressiveness, would bring *invariant inference* power, while strong type systems, requiring manual annotations and proofs, would bring *expressivity*.

We formalized how one can enrich a weakest precondition calculus (WP) with an abstract interpreter. This work takes the shape of a WP calculus transformer: given a WP calculus, we generically construct a brand new WP calculus that produces easier (but sound, still) weakest preconditions, thanks to abstract interpretation.

Concretely, our work is being implemented as an F* effect transformer that leverage Verasco capabilities, for a low-level subset of F*, namely Low*.

7.5. Verified information flow of embedded programs

Participants: Jean-Joseph Marty, Lucas Franceschino, Niki Vazou, Jean-Pierre Talpin.

This PhD project is about applying refinement types theory to verified programming of applications and modules of library operating systems, such as unikernels, for embedded devices of the Internet of Things (IoT): TinyOS, Riot, etc. Our topic has focused on developing a model of information flow control using labeled input-outputs (LIO) implemented using F☆: project Lio☆.

As part of the development of Lio☆, we implemented a library that, thanks to static verification, ensures the containment of information in relation to a parameterized policy for information flow control. In collaboration with Niki Vazou (IMDEA) and Lucas Franceschino we have formalized and developed an automatic method to prove non-interference in Meta☆. Using the Kremlin code generator, programs using Lio☆ can be compiled into C code and run natively on embedded low-resource-constrained devices, without the need for additional runtime system.

In parallel we continued our collaboration with the ProgSys team on a second, now discontinued, project: Gluco☆. The goal of this project was to evaluate the capabilities to use the F* programming language to program an entire system by taking into account its software, hardware and physical constraints using type refinements [25].

8. Bilateral Contracts and Grants with Industry

8.1. Bilateral Contracts with Industry

8.1.1. Inria – Mitsubishi Electric framework program (2018+)

Title: Inria – Mitsubishi Electric framework program

Inria principal investigator: Jean-Pierre Talpin

International Partner: Mitsubishi Electric R&D Europe (MERCE)

Duration: 2018+

Abstract: Following up the fruitful collaboration of TEA with the formal methods group at MERCE, Inria and Mitsubishi Electric signed a center-wide collaboration agreement, which currently hosts projects with project-teams Sumo and Tea, as well as Tocata.

8.1.2. Mitsubishi Electric R&D Europe (2019-2022)

Title: A logical framework to verify requirements of hybrid system models

Inria principal investigator: Jean-Pierre Talpin, Stéphane Kastenbaum

International Partner: Mitsubishi Electric R&D Europe

Duration: 2015 - 2018

Abstract: The goal of this doctoral project is to verify and build cyber-physical systems (CPSs) with a correct-by-construction approach in order to validate system requirements against the two facets of the cyber and physical aspects of such designs. Our approach is based on components augmented with formal contracts that can be composed, abstracted or refined. It fosters the proof of system-level requirements by composing individual properties proved at component level. While semantically grounded, the tooling of this methodology should be usable by regular engineers (i.e. not proof theory specialists).

8.1.3. Mitsubishi Electric R&D Europe (2015-2019)

Title: Parallelism and modular proof in differential dynamic logic [1]

Inria principal investigator: Jean-Pierre Talpin, Simon Lunel

International Partner: Mitsubishi Electric R&D Europe

Duration: 2015 - 2018

Abstract: The primary goal of this Ph.D. project is to ensure correctness-by-design in cyber-physical systems, i.e., systems that mix software and hardware in a physical environment, e.g., Mitsubishi factory automation lines. We develop a component-based approach in Differential Dynamic Logic allowing to reason about a wide variety of heterogeneous cyber-physical systems. Our work provides tools and methodology to design and prove a system modularly.

9. Partnerships and Cooperations

9.1. International Initiatives

9.1.1. Inria International Labs

Sino-European Laboratory in Computer Science, Automation and Applied Mathematics
Associate Team involved in the International Lab:

[25]Towards verified programming of embedded devices. J.-P. Talpin, J.-J. Marty, S. Narayan, D. Stefan, R. Gupta. Design, Automation and Test in Europe (DATE'19). IEEE, 2019.

9.1.1.1. CONVEX

Title: Compositional Verification of Cyber-Physical Systems

International Partner (Institution - Laboratory - Researcher):

CAS (China) - State Key Laboratory of Computer Science - Naijun Zhan

Start year: 2018

See also: http://convex.irisa.fr

Formal modeling and verification methods have successfully improved software safety and security in vast application domains in transportation, production and energy. However, formal methods are labor-intensive and require highly trained software developers. Challenges facing formal methods stem from rapid evolution of hardware platforms, the increasing amount and cost of software infrastructures, and from the interaction between software, hardware and physics in networked cyber-physical systems.

Automation and expressivity of formal verification tools must be improved not only to scale functional verification to very large software stacks, but also verify non-functional properties from models of hardware (time, energy) and physics (domain). Abstraction, compositionality and refinement are essential properties to provide the necessary scalability to tackle the complexity of system design with methods able to scale heterogeneous, concurrent, networked, timed, discrete and continuous models of cyber-physical systems.

Project CONVEX wants to define a CPS architecture design methodology that takes advantage of existing time and concurrency modeling standards (MARTE, AADL, Ptolemy, Matlab), yet focuses on interfacing heterogeneous and exogenous models using simple, mathematically-defined structures, to achieve the single goal of verified integration of CPS components.

Inria@SiliconValley

Associate Team involved in the International Lab:

9.1.1.2. Composite

Title: Compositional System Integration

International Partners (Institution - Laboratory - Researcher):

University of California, San Diego (United States) - Microelectronic Embedded Systems Laboratory - Rajesh Gupta

Start year: 2017

See also: http://www.irisa.fr/prive/talpin/composite

Most applications that run somewhere on the internet are not optimized to do so. They execute on general purpose operating systems or on containers (virtual machines) that are built with the most conservative assumptions about their environment. While an application is specific, a large part of the system it runs on is unused, which is both a cost (to store and execute) and a security risk (many entry points).

A unikernel, on the contrary, is a system program object that only contains the necessary the operating system services it needs for execution. A unikernel is build from the composition of a program, developed using high-level programming language, with modules of a library operating system (libOS), to execute directly on an hypervisor. A unikernel can boot in milliseconds to serve a request and shut down, demanding minimal energy and resources, offering stealthiest exposure time and surface to attacks, making them the ideal platforms to deploy on sensor networks, networks of embedded devices, smart grids and clouds.

The goal of COMPOSITE is to develop the mathematical foundations for sound and efficient composition in system programming: analysis, verification and optimization technique for modular and compositional hardware-system-software integration of unikernels. We intend to further this development with the prospect of an end-to-end co-design methodology to synthesize lean and stealth networked embedded devices.

9.1.1.3. Inria International Chairs

IIC GUPTA Rajesh

Title: End-to-end system co-design

International Partner (Institution - Laboratory - Researcher):

University of California, San Diego (United States) - Rajesh Gupta

Duration: 2017 - 2021

Start year: 2017

9.1.1.4. Insa-Inria International Chair

Shuvra Bhattacharyya

Title: System design methodologies for real-time signal and information processing

International Partner (Institution - Laboratory - Researcher):

University of Maryland (United States) - Shuvra Bhattacharyya

Duration: 2018 - 2021

Start year: 2017

9.2. International Research Visitors

9.2.1. Visits of International Scientists

- Shuvra Bhattacharyya (UMD) visited project-team TEA and IETR in the context of his Insa-Inria Chair in May, July and December. He gave numerous talks and organized a workshop for the preparation of a European project proposal.
- Rajesh Gupta (UCSD) visited project-team TEA in the context of his Inria Chair in July and gave a seminar entitled: programming human spaces.
- Niki Vazou (IMDEA) visited project-team TEA in May and gave a presentation on her POPL'20 paper: "Liquidate your assets: reasoning about resource usage in Liquid Haskell".
- Yamine Ait Ameur (IRIT) visited project-team TEA in January on the occasion of Simon Lunel's Thesis defense.
- Naijun Zhan (ISCAS) visited project-team TEA in July, in the context of associate-project CONVEX.
- Delegates of the Sheng Yuan Honors College (BUAA) visited Inria-Irisa and Ecole Normale Supérieur de Rennes for the prospect of initiating an exchange program for graduate students, which will start in 2020.
- Zhang Bojun and Wang Zikai (BUAA) visited project-team TEA in July for an internship on verified modeling of blockchain protocols in Coq.
- Shenghao Yuan (NUAA) visited project-team TEA in July, in the context of associate-team CONVEX, and gave a presentation of the verified mini-Signal code generator developed at Nanhang University.

9.2.2. Visits to International Teams

Jean-Pierre Talpin visited UC San Diego in March, in the context of the associate-team Composite, and visited ISCAS, Beijing, in May and October, in the context of the associate-team CONVEX.

10. Dissemination

10.1. Promoting Scientific Activities

10.1.1. Scientific Events Selection

Jean-Pierre Talpin served in the program committee of the ACM LCTES'19, ACM SAC'19 and SCOPES'19 conferences.

Thierry Gautier reviewed articles for *Journal of Systems Architecture* (Elsevier).

10.1.2. Journal

Jean-Pierre Talpin is Associate Editor with the ACM Transactions for Embedded Computing Systems (TECS).

Thierry Gautier reviewed articles for *IEEE Access* and *Science of Computer Programming*.

10.1.3. Invited Talks

Jean-Pierre Talpin gave an invited presentation entitled "Towards verified programming of embedded devices" at DATE'19, Florence.

10.2. Teaching - Supervision - Juries

Jean-Pierre Talpin gave a one week graduate-level course at the Sheng Yuan Honors College, BUAA, entitled: "introduction to program verification".

Jean-Pierre Talpin co-supervises the PhD Theses of Stéphane Kastenbaum, Simon Lunel, Liangcong Zhang, Jean-Joseph Marty and Lucas Franceschino

Thierry Gautier served as external assessor for professor position application at Nankai University (China).

11. Bibliography

Publications of the year

Doctoral Dissertations and Habilitation Theses

[1] S. LUNEL. *Parallelism and modular proof in differential dynamic logic*, Université Rennes 1, January 2019, https://tel.archives-ouvertes.fr/tel-02102687

Articles in International Peer-Reviewed Journals

[2] T. GAUTIER, C. GUY, A. HONORAT, P. LE GUERNIC, J.-P. TALPIN, L. BESNARD. *Polychronous automata and their use for formal validation of AADL models*, in "Frontiers of Computer Science", August 2019, vol. 13, n^{o} 4, pp. 677-697 [*DOI :* 10.1007/s11704-017-6134-5], https://hal.inria.fr/hal-01411257

Invited Conferences

[3] J.-P. TALPIN, J.-J. MARTY, S. NARAYAN, D. STEFAN, R. GUPTA. *Towards verified programming of embedded devices*, in "DATE 2019 - 22nd IEEE/ACM Design, Automation and Test in Europe", Florence, Italy, IEEE, March 2019, pp. 1445-1450 [*DOI :* 10.23919/DATE.2019.8715067], https://hal.inria.fr/hal-02193635

[4] H. ZHAN, Q. LIN, S. WANG, J.-P. TALPIN, X. XU, N. ZHAN. *Unified Graphical Co-Modelling of Cyber-Physical Systems using AADL and Simulink/Stateflow*, in "UTP 2019 - 7th International Symposium on Unifying Theories of Programming", Porto, Portugal, October 2019, pp. 1-20 [*DOI :* 10.1007/978-3-030-31038-7_6], https://hal.inria.fr/hal-02193662

International Conferences with Proceedings

[5] S. LUNEL, S. MITSCH, B. BOYER, J.-P. TALPIN. *Parallel Composition and Modular Verification of Computer Controlled Systems in Differential Dynamic Logic*, in "FM 2019 - 23rd International Symposium on Formal Methods", Porto, Portugal, October 2019, pp. 1-22, https://arxiv.org/abs/1907.02881 - Long version of an article accepted to the conference FM'19, https://hal.inria.fr/hal-02193642

[6] H. N. TRAN, A. HONORAT, J.-P. TALPIN, T. GAUTIER, L. BESNARD. *Efficient Contention-Aware Scheduling of SDF Graphs on Shared Multi-bank Memory*, in "ICECCS 2019 - 24th International Conference on Engineering of Complex Computer Systems", Hong Kong, China, IEEE, November 2019, pp. 114-123 [*DOI :* 10.1109/ICECCS.2019.00020], https://hal.inria.fr/hal-02193639