INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team aoste

Models and Methods for the Analysis and Optimization of Systems with Real-time and Embedded Constraints

Sophia Antipolis - Rocquencourt

THEME COM

Activity Report

2006

Table of contents

1. Team

Aoste is a joint team with UNSA (university of Nice/Sophia-Antipolis) and CNRS, through their I3S laboratory. It is also co-located between Sophia-Antipolis and Rocquencourt. Project members originate from the former INRIA Tick and Ostre teams, together with the I3S Sports team.

Head of the team
Robert de Simone [Research Director Inria]

Vice-head of the team
Yves Sorel [Research Director Inria]

Administrative assistant
Sophie Honnorat [Sophia-Antipolis, part-time]
Nelly Maloisel [Rocquencourt, part-time]
Viviane Rosello [I3S, part-time]

Staff member INRIA
Dumitru Potop-Butucaru [Researcher, Inria]
Daniel de Rauglaudre [Research Engineer, Inria]
Alix Munier [On temporary leave from University Paris XII, HdR]

Staff member UNSA
Charles André [professor at UNSA, HdR]
Frédéric Mallet [associate professor at UNSA]
Marie-Agnès Peraldi-Frati [associate professor at UNSA]

PhD. students
Julien Boucaron [ST Microlectronics contractual funding]
Omar Kermia [INRIA scholarship]
François Lagarde [codirected with CEA-List]
Aamir Mehmood [Pakistanese government scholarship, from December]
Patrick Meumeu Yomsi [INRIA scholarship]
Jean-Vivien Millo [BDE regional scholarship with ST Microelectronics]
Nicolas Pernet [INRIA scolarship, until July]
Mickaël Raulet [Mitsubishi Electric contractual funding, until May]

Post-doctoral fellow
Arnaud Cuccuru [until April]

Specialist Engineer
Gérard Cristau [on leave from Thales, part-time, until October]

Technical Staff
Benoît Ferrero [Project engineer, from December]
Fadoi Lakhal [Project engineer, from October]
Djamel Louar [Project engineer, from October]
Christophe Gensoul [Software development staff]

Student interns
Lyes Abdelfettah [University Paris 6]
Lionel Durand [University Bordeaux 1]
Fadoi Lakhal [University Paris 12]
Djamel Louar [University Paris 12]
Aamir Mehmood [UNSA STIC Research Master]
Manel Sghairi [UNSA STIC Research Master]

External collaborators
Liliana Cucu [Post-doctoral fellow, Université Libre de Bruxelles]
Thierry Grandpierre [Assistant Professor ESIEE Noisy-Le-Grand]

2. Overall Objectives

2.1. Overall Objectives

Keywords: *AAA methodology*, *Esterel*, *SynDEx*, *SyncCharts*, *SysML*, *UML*, *adequation*, *architectural models*, *automatic verification*, *code distribution*, *compilation*, *embedded systems*, *formal semantics*, *hardware synthesis*, *hardware/software codesign*, *model-driven engineering*, *multiprocessors*, *optimization*, *program analysis*, *real-time*, *scheduling*, *synchronous reactive formalisms*, *system-level design*, *systems-on-Chip*.

The main objective of the Aoste team is to promote the formal design of embedded systems, with their intrinsic concurrent, distributed and real-time aspects. We build upon previous experience by team members on ESTEREL, SYNCCHARTS and synchronous reactive formalismes, SYNDEX and the *Algorithm-Architecture Adequation* methodology (AAA) to reach this goal. Domains of application range from transport (automotive, aircrafts), electronic appliances (mobile phones, HDTV,...), to System-on-Chip design.

For the sake of presentation we split our activities in two: the definition of adequate description formalisms and models, with their formal semantics for precise representation of real-time embedded systems on the one hand; the design of relevant model transformation and analysis techniques to cover an efficient design flow methodology based on these formalisms on the other hand. By transformation here we mean various compilation and implementation techniques, considered as transforming a higher-level model into a lower-level one. Analysis instead work at the same modeling level.

Modern embedded systems combine complexity and heterogeneity both at the level of applications (with a mix of control-flow modes and mutilmedia data-flow streaming), and at the level of execution platforms (with increasing concurrency and multicore architectures). We use synchronous reactive models, their globally-asynchronous/locally-synchronous and multiclock extensions and formal semantics to describe application's behaviors as well as architectural timing constraints. The transformations under consideration include combinations of spatial distribution and temporal scheduling to map the application onto the platform, as well as various optimization techniques to improve compilation. Static and dynamic (model-checking) analysis are also used to provide insights on correctness and efficiency of models according to the prescribed formal semantics.

3. Scientific Foundations

3.1. Languages and Models

Keywords: *AAA methodology*, *Esterel*, *SynDEx*, *SyncCharts*, *UML*, *synchronous formalisms*.

Participants: Charles André, Julien Boucaron, Liliana Cucu, Robert de Simone, Frédéric Mallet, Jean-Vivien Millo, Marie-Agnès Péraldi, Dumitru Potop-Butucaru, Yves Sorel.

3.1.1. Synchronous reactive formalisms

Synchronous reactive models are those where concurrent processes all run at the speed of a common global clock, in a discrete logical time framework [48], [50], [43]. Simultaneous behaviors in a single global instant are thus allowed and even often required. The main issue is then to ensure correct causal behavior propagation inside a given instant, as well as across instants. Programs may indeed exist for which there is no such possible execution, and they have to be recognized and banned as ill-behaved. This issue is unfortunately too often overlooked by many existing specification formalisms in the field, as in Matlab/Simulink, VHDL/Verilog/SystemC, or StateCharts, to name a few.

Synchronous reactive formalisms, such as ESTEREL/SYNCCHARTS, LUSTRE/SCADE, or SIGNAL/POLYCHRONY were designed explicitly to offer dedicated programming models and proper specification primitives, to study advanced semantic issues and useful transformations schemes, in the realm of synchronous models [43], [50],[20], [1], [4], [3], [2]. ESTEREL and SYNCCHARTS are control-oriented state-based formalisms, while *Lustre* and *Signal* are declarative data-flow based formalisms.

These languages were mostly born out of INRIA or of collaborating French teams. The INRIA spin-off Esterel-technologies now markets ESTEREL STUDIO and SCADE. SynDEx and the AAA methodology also rely greatly on synchronous assumptions.

Synchronous reactive formalisms raise fascinating questions regarding their efficient and correct implementation onto heterogeneous real-time embedded platforms. Numerous research contributions have been provided in this direction for now over two decades [14], [3], [20]. In many ways synchronous reactive models can be viewed as playing a central role in embedded system design, as syntactic formalisms supporting the explicit representation of precise concurrency and scheduling patterns.

3.1.2. Globally-Asynchronous/Globally-Synchronous (GALS) and multiclock extensions

Over time the basic synchronous model was shown to suffer expressivity lacks when considering complex heterogenous systems. For such large designs the precise temporal allocation of instants to behaviors on a single global clock would impediment the general development flow, as it requires too many details from the designer. Instead, reasoning with several distinct clock domains, possibly loosely-coupled, provides necessary specification freedom. But this should not impair the underlying semantics, with all its correctness requirements; such is the price to pay to preserve the powerful implementation schemes as considered formerly.

GALS models were introduced to relax some of the synchrony hypothesis. They are used in the theory of *Latency-Insensitive Design (LID)*, which allows de-synchronization then re-synchronization of former synchronous specifications into implementations that are synchronous again, but this time will accommodate mandatory extra latencies provided by users. The final design is proven to be equivalent to the former, and requires no modification of the local (computation) IP components; it only adds an extra buffering and synchronization layer in between them. Similarly *multiclock* designs allow to consider hierarchical clock structures where not all components are active simultaneously. In both cases active research is conducted to study practical conditions under which extended models can be transformed into equivalent synchronous ones (in a way that respects behavior, and not necessarily timing). On the other hand the multiclock or LID extensions may provide extra properties that can be taken into account to improve even more the efficiency of embedded implementations. Endochrony [29] is an example of such property. Expansion of multiclock or GALS systems into plain synchronous ones takes the form of operational scheduling of concurrent applications.

LID theory bears strong ties with the former theories of Weighted Marked Graphs and its timed schedulings, either dynamic or static, such as described in [44], [51]. Results of this nature were already used in the context of software pipelining [47], and could certainly be better exploited in LID context.

3.1.3. UML modeling diagrams for Real-Time Embedded applications

Graphical models and diagrams were popular in Embedded System design long before the advent of the UML. In fact the ancestors of UML state, activity and sequence diagrams, namely statecharts, Petri nets and Message Sequence Charst, originated from this field. The long-standing object orientation philosophy of UML then disclosed many differences in spirit with the former semantics of such models (in the intuitive sense, since UML provides no formal semantics to its behavioral diagrams). This trend has been reversed with the introduction of components and ports in ROOM, then of block diagrams in SysML. But still, no semantics.

The benefits of expressing our models inside the UML framework would be to allow usage of UML editors and tools (textual or graphical), as well as to expose our result to a larger community. The new SysML profile for System Engineering is already a proof of interest towards such a modeling style. Of course this would require from us the definition of a timed semantics both formal and compatible with whatever there is of semantic concerns in current UML 2.1. We are currently taking part in a consortium named PROMARTE which, in part, aims at providing such an official UML profile at OMG level. The profile itself is named MARTE, which stands for *Modeling and Analysis of Real-Time Embedded systems*. The consortium was initiated by INRIA, Thales, and CEA-List, as part of a CARROLL initiative 7.3. The profile RFP (Request For Proposals) was voted early 2005, and the initial submission is due in March 2007.

We are exploiting our contributions to MARTE and our experience gained in meta-modeling in a number of downstream activities. We use model transformation techniques to define such a transformation of real-time distributed applications modeled in Marte into Time Petri Nets representations for analysis purpose [36]. In collaboration with other INRIA teams (Espresso and DaRT mostly) we provide meta-models for SYNDEX, and later of other synchronous formalisms to be coupled to MARTE.

Depending on the application domain a number of formalisms have been introduced for the representation of system structue and behavior, that were sometimes later embedded into some sort of UML syntax to benefit from tool vendor support. These include AADL in the avionics domain, EAST-ADL and AUTOSAR in the automotive domain, UML4SOC and to a lesser extend SPIRIT in the SoC design field. In all these cases we are seriously considering the connection with our MARTE contributions, to ensure our model enjoys enough semantic expressive power so as to represnet formerly the underlying concepts to these profiles.

3.1.4. AAA methodology

The purpose of the AAA methodology is to provide independent modeling of *applications* and supporting *architectures* in a first step. The mapping of applications onto architectures is realized only in a subsequent step, and is subject to algorithmic optimizations relative to the various timing constraints and costs involved. This approach is called *Algorithm-Architecture Adequation* (AAA) [16].

AAA allows the designer to specify "application algorithms" (functionalities) and "Embedded platform" (composed of hardware resources such as processors and specific integrated circuits all together interconnected) with graph models. Consequently, all the possible implementations of a given algorithm onto a given architecture can be described in terms of graphs transformations. An implementation consists in distributing and scheduling a given algorithm onto a given embedded platform. The Adequation amounts to explore, manually and/or automatically, the possible implementations, such that the real-time, and the hardware resources are best used. Furthermore, the adequation results are used, as an ultimate graphs transformation, to automatically generate two types of code: dedicated distributed real-time executives for processors, and net-lists (structural VHDL) for specific integrated circuits. Finally, fault tolerance is of great concern because the applications we are dealing with are often critical, that is to say, may lead to catastrophic consequences when fault occurs. Thus AAA generates automatically the redundant operations as well as data dependences (software redundancy) necessary to make transparent these faults when some hardware components fail.

Real-time embedded systems are, first of all, "reactive systems" that mandatorily must react infinitely to each input event of the infinite sequence of events it consumes, such that "input rate" and "latency" constraints are satisfied. The input rate corresponds to the delay between two successive input events, these events may be periodic, sporadic or aperiodic. The latency corresponds to the delay between an input event consumed by the system and an output event produced by the system in reaction, through the computation of several operations, to this input event. The term event is used here in a broad sense. It represents an information which is present relatively to a discrete time reference. When hard (critical) real-time is considered, off-line approaches are preferred due to their predictability and low overhead, and when on-line approaches are unavoidable, mainly to take into account aperiodic events, we intend to minimize the cost of the decisions taken during the real-time execution. When soft real-time is considered off-line and on-line approaches are mixed. The application domains we are involved in, e.g. automobile, avionic, lead to consider scheduling problems for systems of tasks with precedence, latency and periodicity constraints, whereas usually only periodicity is considered. We seek optimal results in the monoprocessor case where distribution is not considered, and sub-optimal results through heuristics in the multiprocessor case, because the problems are NP-hard due to distribution consideration. In addition to these timing constraints, the systems must satisfy embedding constraints, such as power consumption, weight, volume, memory, etc, it turns out that hardware resources must be minimized. In the most general case architectures are distributed, and composed of several programmable components (processors) and several specific integrated circuits (ASIC: Application Specific Integrated Circuit or FPGA: Field Programmable Gate Array) all together interconnected with possibly different types of communication media. We call such heterogeneous architectures "multicomponent" [17].

The AAA methodology is implemented in a system level CAD software called SynDEx (http://www.syndex.org). This software, coupled with a high level specification language, like one of the Synchronous Languages, Scilab/Scicos, or UML2.0, leads to a seamless environment allowing to perform rapid prototyping and hardware/software co-design while reducing drastically the development cycle duration and providing safe design.

3.1.4.1. Algorithm model

Our algorithm model is an extension of the well known data-flow model from Dennis [46]. It is a directed acyclic hyper-graph (DAG) that we call “conditioned factorized data dependence graph”, whose vertices are “operations” and hyper-edges are directed “data or control dependences” between operations. Hyper-edges are necessary in order to model data diffusion since a standard edge only relates a pair of operations. The data dependences defines a partial order on the operations execution, called “potential operation-parallelism”. Each operation may be in turn described as a graph allowing a hierarchical specification of an algorithm. Therefore, a graph of operations is also an operation. Operations which are the leaves of the hierarchy are said “atomic” in the sense that it is not possible to distribute each of them on more than one computation resource. The basic data-flow model was extended in three directions, firstly infinite (resp. finite) repetition of a sub-graph pattern in order to specify the reactive aspect of real-time systems (resp. in order to specify the finite repetition of a sub-graph consuming different data similar to a loop in imperative languages “data potential parallelism”), secondly “state” when data dependence are necessary between infinite repetitions of the sub-graph pattern introducing cycles which must be avoided by specific vertices called “delays” (similar to z^{-n} in automatic control), thirdly “conditioning” of an operation by a control dependence similar to conditional control structure in imperative languages, allowing the execution of alternative subgraphs. Delays combined with conditioning allow the programmer to specify FSM (Finite State Machine) necessary for describing “mode changes”, e.g. some control law graph is performed when the motor is the state “idle” whereas another one is performed when it is in the state “permanent”. Finite repetition and conditioning both exploit the hierarchy principle.

3.1.4.2. Architecture model

The typical coarse-grain architecture models such as the PRAM (Parallel Random Access Machines) and the DRAM (Distributed Random Access Machines) [55] are not enough detailed for the optimizations we intend to perform. On the other hand the very fine grain RTL-like (Register Transfer Level) [54] models are too detailed. Thus, our model of multicomponent architecture is also a directed graph [12], whose vertices are of four types: “operator” (computation resource or sequencer of operations), “communicator” (communication resource or sequencer of communications, e.g. DMA), memory resource of type RAM (random access) or SAM (sequential access), “bus/mux/demux/(arbiter)” (selection of a memory or arbitration to a shared memory for an operator), and whose edges are directed connections. For example, a typical processor is a graph composed of an operator, interconnected with memories (program and data) and communicators, through bus/mux/demux/(arbiter). A “communication medium” is a linear graph composed of memories, communicators, bus/mux/demux/arbiters corresponding to a “route”, i.e. a path in the architecture graph. Like for the algorithm model, the architecture model is hierarchical in order to abstract architectural details. Although this model seems very basic, it is the result of several studies [12] in order to find the appropriate granularity allowing, on the one hand to provide accurate optimization results, and on the other hand to obtain as quickly as possible these results during the rapid prototyping phase.

Our model of integrated circuit architecture is the typical RTL model. It is a directed graph whose vertices are of two types: combinatorial circuit executing an instruction, and register storing data used by instructions, and whose edges are data transfers between a combinatorial circuit and a register, and reciprocally.

3.1.4.3. Implementation model

An implementation of a given algorithm onto a given multicomponent architecture corresponds to a distribution and a scheduling of, not only the algorithm operations onto the architecture operators, but also a distribution and a scheduling of the data transfers between operations [11] onto communication media.

The distribution consists in distributing each operation of the algorithm graph onto an operator of the architecture graph. This leads to a partition of the operations set, in as many sub-graphs as there are operators. Then, for each operation two vertices called "alloc" for allocating program (resp. data) memory are added, and each of them is allocated to a program (resp. data) RAM connected to the corresponding operator. Moreover, each "inter-operator" data transfer between two operations distributed onto two different operators, is distributed onto a route connecting these two operators. In order to actually perform this data transfer distribution, according to the element composing the route as many "communication operations" as there are communicators, as many "identity" vertices as there are bus/mux/demux, and as many "alloc" vertices for allocating data to communicate as there are RAM and SAM, are created and inserted. Finally, communication operations, identity and alloc vertices are distributed onto the corresponding vertices of the architecture graph. All the alloc vertices, those for allocating data and program memories as well as those for allocating data to communicate, are used to determine the amount of memory necessary for each processor of the architecture.

The scheduling consists in transforming the partial order of the corresponding subgraph of operations distributed onto an operator, in a total order. This "linearization of the partial order" is necessary because an operator is a sequential machine which executes sequentially the operations. Similarly, it also consists in transforming the partial order of the corresponding subgraph of communications operations distributed onto a communicator, in a total order. Actually, both schedulings amount to add edges, called "precedence dependences" (compared to data dependences), to the initial algorithm graph. To summarize, an implementation corresponds to the transformation of the algorithm graph (addition of new vertices and edges to the initial ones) according to the architecture graph.

Finally, the set of all the possible implementations of a given algorithm onto a given architecture may be modeled, in intention, as the composition of three binary relations: namely the "routing", the "distribution", and the "scheduling" [19]. Each relation is a mapping between two pairs of graphs (algorithm graph, architecture graph). It also may be seen as a external compositional law, where an architecture graph operates on an algorithm graph in order to give, as a result, a new algorithm graph, which is the initial algorithm graph distributed and scheduled according to the architecture graph. Therefore, the "implementation graph" is of type algorithm that may in turn be composed with another architecture graph, allowing complex combinations.

The set of all the possible implementations of a given algorithm onto a specific integrated circuit is obtained differently because we need a transformation of the algorithm graph into an architecture graph which is directly the implementation graph. This graph is composed of two parts: the data-path obtained by translating each operation in a corresponding logic function, and the control path obtained by translating each control structure in a "control unit", which is a finite state machine made of counters, multiplexers, demultiplexers and memories, managing repetitions and conditionings [8].

3.2. Transformations and analysis

Keywords: *Compilation*, *allocation*, *architecture exploration*, *formal verification*, *optimization*, *synthesis*.

Participants: Charles André, Julien Boucaron, Liliana Cucu, Robert de Simone, Omar Kermia, Frédéric Mallet, Patrick Meumeu, Jean-Vivien Millo, Marie-Agnès Péraldi, Dumitru Potop-Butucaru, Yves Sorel.

3.2.1. Compilation of synchronous reactive formalisms

Syntactic constructs in synchronous languages are always provided meaning in terms of formal operational semantics on well-defined interpretation models. As a result, all kinds of compilation/synthesis, analysis and verification, or optimization methods can readily be caracterized as formal transformations on such mathematical models [20], [14]. While implementations may seek various optimality criteria, they should always in principle be established as "correct-by-construction" following semantic equivalence with the basic version.

Concerning Esterel/SyncCharts, compilation was first realized in the 1980's as an expansing into flat global Mealy FSMs; this produces efficient, but often unduly large code size. Then in the 1990's a translation was defined into Boolean equation systems (BES), with Boolean register memories encoding active control points. While such models are known in the hardware design community as Boolean gate *netlists*, they can be used in our context for software code production. Here the code produced in quasi-linear in size (worst-case quadratic in rare cases), but the execution consists in a linear evaluation of the whole equation system (thus each reaction requires an execution time proportional to the whole program, even when only a small fragment is truly active). Thus in the early 2000's new implementation frameworks were introduced, as in particular in the PhD thesis of Dumitru Potop-Butucaru; such compilation patterns rely on high-level control-data flowgraphs selecting the active parts before execution at each instant. This scheme is both fast and memory-efficient, but cannot cope with all programs (as the full constructive causality analysis underlying the synchronous assumption cannot then be realized at "compile time", and this check is of utterly importance for program correctness). Correctness is in this context ensured by a stronger, more restrictive *acyclicity* criterion, which provides a static evaluation order for signal propagation.

The advanced compilation techniques defined in the PhD thesis of Dumitru Potop-Butucaru [13], where the current hierarchical state and input signals are considered to determine the actual parts of the (concurrent) code which have to be active in the current reaction, are finding their way in the industrial compiler distributed by ESTEREL TECHNOLOGIES, where they are marketed as "fast C" compilation. Meanwhile work by Olivier Tardieu (formerly PhD student in the Aoste team and currently holding a postdoctoral position at Columbia University, NY) is establishing a number of internal semantic-preserving transformations from Esterel to (slightly augmented) Esterel [21]. These transformation steps combine the introduction of various `goto`-like primitives so that programs are rewritten into forms that enjoy "good" natural properties (such as *non-schizophrenia* for instance). These extensions are promising as a way to introduce synchronous formalisms to support several stages in the design flow of complex, heterogeneous embedded systems.

3.2.2. Dynamic analysis, automatic verification and model-checking

Because of the static structure of the Embedded Systems we consider, their models are frequently finite-state when considering only the control aspects. Thus, under some appropriate data abstraction, they are amenable to automatic verification (often known as "model-checking"). This is in fact put in use in many of the previous compilation schemes for Esterel/SyncCharts, which rely on computations performed at compile time, while these computations converge only in this finite-state case.

In the past years we developed the XEVE model-checker for Esterel, based on BDD symbolic representation techniques. We developed a number of approaches to partition the computation of the reachable state space according to the program structure [18]. In turn these methods have been shown to hold close relations with efficient compilation techniques as well. While these activities are currently in a somehow "stand-by" mode in the team, it is important to maintain some competency in this domain because of its interactions with other fields.

3.2.3. Time Refinement

In our design approach applications and architectural platforms are first described independently, both with their behavioral and structural aspects, and possibly with their logical time constraints. Asynchonous processes lead to loosely coupled or independent (virtual) clocks, synchronous systems rely rather on common global clocks. Clocks are of a more logical nature in applications, of more physical nature in the platform. The application mapping of application functions onto platform resources realizes an association between the whole set of clocks by solving constraints. In many simple case the solutions, which represent the scheduling of the application of the target platform, can be represented syntactically as "first-order objects" of the modelization framework.

We used this general philosophy in our work on static scheduling of latency-insensitive synchronous systems. Starting from a fully synchronous early specification, mandatory latencies on long communication wires are imposed from outside. The specification is de-desynchronized, then resynchronized to form a latency-independent version [45]. Then hardware implementation constraints impose the form and placement of specific buffering elements, named *relay-stations*, to realize the mandatory flow control. But as shown by classical results [44], [42], the resulting global behavior is ultimately k-periodic (for strongly connected systems). Then an explciit schedule can be effectively constructed, and used to optimized the allocation of *relay-stations*.

3.2.4. Distributed Real-Time Scheduling and Optimization

From algorithm and architecture models it is possible to express the space of all possible implementations. We must then choose a particular one for which the constraints are satisfied, while at the same time optimizing some criteria. In the case of a multiprocessor architecture the problem of finding the best distribution and scheduling of the algorithm onto the architecture, is known to be of NP-hard complexity.

The first problem we address consists in considering, in addition to precedences constraints specified through the algorithm graph model, one latency constraint between the first operation(s) (without predecessor) and the last operation(s) (without successor), equal to a unique periodicity constraint (input rate) for all the operations. We propose several heuristics based on the characterization of the operations (resp. communication operations) relatively to the operators (resp. communicators) which associate to each pair (operation, operator) (resp. (communication operation, communicator)) a set of values representing an execution time, a power consumption, a surface, etc. For example we minimize the total execution time of the algorithm (makespan) onto the distributed architecture, with a cost function taking into account the schedule flexibility of operations, and also the increase of the critical path when two operations are distributed onto two different operators inducing a communication, possibly through concurent routes [11]. We mainly develop “greedy heuristics” because they are very fast, and thus, well suited to rapid prototyping of realistic industrial applications. In this type of applications the algorithm graph may have up to ten thousand vertices and the architecture graph may have several tens of vertices. However, we extend these greedy heuristics to iterative versions which are much slower, due to back-tracking, but give better results when it is time to produce the final commercial product. For the same reason we also develop local neighborhood heuristics such as “Simulated Anealing”, and also “Genetic Algorithm”, all based on the same type of cost function.

New applications in the automobile, avionic, or telecommunication domains, led us to consider new problems with more complex constraints. In such applications it is not sufficient to consider the execution duration of the algorithm graph. We need also to consider periodicity constraints for the operations, possibly different, and several latency constraints imposed possibly on whatever pair of operations. Presently there are only partial and simple results for such situations in the multiprocessor case, and only few results in the monoprocessor case. Then, we began few years ago to investigate this research area, by interpreting, in our algorithm graph model, the typical scheduling model given by Liu and Leyland [53] for the monoprocessor case. This leads us to redefine the notion of periodicity through infinite and finite repetitions of an operations graph (i.e. the algorithm), thus generalizing the SDF (Synchronous Data-Flow) model [51] proposed in the software environment Ptolemy. For simplicity reason and because this is consistent with the application domains we are interested in, we presently only consider that our real-time systems are non-preemptive, and that “strict periodicity” constraints are imposed on operations, meaning that an operation starts as soon as it period occurs. In this case we give a schedulability condition for graph of operations with precedence and periodicity constraints in the non-preemptive case [5].

We also formally defined the notion of latency which is more powerful, for the applications we are interested in, than the usual notion of “deadline” that is not able to impose directly a timing constraint on a pair of operations, indeed when two deadlines are used the problem becomes overconstrained. Both operations of the pair are connected by at least one path as usually found in “end-to-end constraints”. In order to study schedulability conditions for multiple latency constraints we defined three relations between pair of paths, such that for each pair a latency constraint is imposed on its extremities. Using these relations called *II*, *Z* and *X*, we give a schedulability condition for graph of operations with precedence and latency constraints in the non-preemptive case. Then by combining both previous results we give a schedulability condition for graph of operations with precedence, periodicity and latency constraints in the non-preemptive case, using an important result which gives a relation between periodicity and latency. We also give an optimal scheduling algorithm in the sense that, if there is a schedule the algorithm will find it.

Thanks to these results obtained in the monoprocessor (one operator) case we study the problem of distribution and scheduling in the multiprocessor case (several operators) with more complex constraints than in the case previously studied, i.e. with precedence, and one latency constraint equal to a unique periodicity constraint. We proved this problem is NP-hard for systems with precedence and periodicity constraints, we proposed a heuristic which takes into account the communication times. We proved that operations with periods which are not co-prime can not be scheduled on the same operator. We proved this problem is NP-hard for systems with precedence and latency constraints, we proposed a heuristic which takes into account the communication times. This heuristic uses the schedulability results obtained in the case of one operator concerning the three relations *II*, *Z* and *X* between pairs of operations, on which latency constraints are defined. The latter results prove that the best way of scheduling operations is to avoid scheduling, between the first and the last operation of a latency constraint, operations which do not belong to this latency constraint. Finally, we proved this problem is NP-hard for systems with precedence, periodicity and latency constraints, we proposed a heuristic which takes into account the communication times. We proved that operations belonging to the same latency constraint must have the same period. A direct consequence is that the operations belonging to the same pair or to pairs which are in relation *II*, *Z* or *X* must have the same period. So, the heuristic may use the main ideas of the heuristic for the case of precedence and latency constraints and of the heuristic for the case of precedence and periodicity constraints. The performances of these three heuristics were compared to those of exact algorithms. These results show that the heuristics are definitely faster then the exact algorithms for all cases when the heuristics find a solution [6].

The aforementioned scheduling problems only takes into account periodic operations. Aperiodic operations issued from aperiodic events, usually related to control, must be handled on-line. We take them into account off-line by integrating the control-flow in our data-flow model, well suited to distribution, and by maximizing the control effects. We study relations between control-flow and data-flow in order to better exploit their respective advantages. Finally, we mix off-line for periodic operations and on-line approaches for aperiodic operations.

In the case of integrated circuit the potential parallelism of the algorithm corresponds exactly to the actual parallelism of the circuit. However, this may lead to exceed the required surface of an ASIC or the number of CLB (Combinatorial Logic Block) of a FPGA, and then some operations must be sequentially repeated several times in order to reuse them, reducing in this way the potential parallelism to an actual parallelism with less logic functions. But reducing the surface has a price in terms of time, and also in terms of surface but of a lesser importance, due to the sequentialization itself (instead of parallelism) performed by the finite state machines (control units) necessary to implement the repetitions and the conditionings. Then, we are seeking a compromise taking into account surface and performances. Because these problems are again of NP-hard complexity, we propose greedy and iterative heuristics in order to solve them [8].

Finally, we plan to work on the unification of multiprocessor heuristics and integrated circuit heuristics in order to propose “automatic hardware/software partitioning” for co-design, instead of the usual manual one. The most difficult issue concerns the integration in the cost functions of the notion of “flexibility” which is crucial for the choice of software versus hardware. However, this optimization criterion is difficult to quantify because it mainly relies on user’s expertise.

3.2.5. Generation of Distributed Code

As soon as an implementation is chosen among all the possible ones, it is straightforward to automatically generate executable code through an ultimate graphs transformation leading to a distributed real-time executive for the processors, and to a structural hardware description, e.g. synthetizable VHDL, for the specific integrated circuits.

For a multicomponent each operator (resp. each communicator) has to execute the sequence of operations (resp. communication operations) described in the implementation graph. Thus, this graph is translated in an “executive graph” [12] where new vertices and edges are added in order to manage the infinite and finite loops, the conditionings, the inter-operator data dependences corresponding to “read” and “write” when the communication medium is a RAM, or to “send” and “receive” when the communication medium is a SAM.

Specific vertices, called "pre" and "suc", which manage semaphores, are added to each read, write, send and receive vertices in order to synchronize the execution of operations and of communication operations when they must share, in mutual exclusion, the same sequencer as well as the same data. These synchronizations insure that the real-time execution will satisfy the partial order specified in the initial algorithm. Executives generation is proved to be dead-lock free maintaining the properties, in terms of events ordering, shown thanks to the synchronous language semantics. This executive graph is directly transformed in a macro-code [12] which is independent of the processor. This macro-code is macro-processed with "executive kernels" libraries which are dependent of the processors and of the communication media, in order to produce as many source codes as there are processors. Each library is written in the best adapted language regarding the processors and the media, e.g. assembler or high level language like C. Finally, each produced source code is compiled in order to obtain distributed executable code satisfying the real-time constraints.

For an integrated circuit, because we associate to each operation and to each control unit an element of a synthetizable VHDL library, the executable code generation relies on the typical synthesis tools of integrated circuit CAD vendors like Synopsis or Cadence.

3.2.6. Fault tolerance

For the applications we are dealing with, if the real-time constraints are not satisfied, this may have catastrophic consequences in terms of human beings lost or pollution. When a fault occurs despite formal verifications which allow safe design by construction, we propose to specify the level of fault the user accepts by adding redundant processors and communication media. Then, we extended our optimization heuristics in order to generate automatically the redundant operations and data dependences necessary to make transparent these faults. Presently, we only take into account "fail silent" faults. They are detected using "watchdogs", the duration of which depends on the operations and data transfers durations. We first obtained results in the case of processor faults only, i.e. when the communication media are assumed error free. Then, we studied, in addition to processors faults, media faults.

We propose three kinds of heuristics to tolerate both faults. The first one tolerates a fixed number of arbitrary processors and links (point-to-point communication medium) faults [10]. It is based on the software redundancy of operations. The second one tolerates a fixed number of arbitrary processors and buses (multipoint communication medium) faults. It is based on the active software redundancy of the operations and the passive redundancy of the communications with the fragmentation in several packets of the transfered data on the buses. The third one tolerates a fixed number of arbitrary processors and communication media (point-to-point or multipoint) faults. It is based on a quite different approach. This heuristic generates as much distributions and schedulings as there are of different architecture configuration corresponding to the possible faults. Then, all the distributions and schedulings are merged together to finally obtain a resulting distribution and a scheduling which tolerates all the faults [9].

Finally, we propose a heuristic for generating reliable distributions and schedulings. The software redundancy is used to maximize the reliability of the distribution and scheduling taking into account two criteria: the minimization of the latency (execution duration of the distributed and scheduled algorithm onto the architecture) and the maximization of the reliability of the processors and the communication media.

As soon as the redundant hardware is fully exploited, "degraded modes" are necessary. They are specified at the level of the algorithm graph by combining delays and conditionings.

4. Application Domains

4.1. Mobile robotics, automotive and transportation

Keywords: *embedded automotive electronics*, *transportation*.

With increasing functionality demands in powertrain, body comfort or telematics applications, modern cars are becoming complex systems including Real-Time OS (OSEK), complex data buses (CAN or TTP/FlexRay), with distributed intelligent sensors and powerful computing power. Software and electronic are now becoming a prominent part of both the price and added value. Still, no "ideal" hardware/software architecture is yet standardized, and the development methodologies are still at infancy. Proposals for high-level modeling and infrastructure platform organization have been proposed, as in the EAST-EEA and AutoSar consortium. We are taking part in the first initiative, mostly through the AAA methodology which proposes computer-aided mapping of synchronous applications onto heterogeneous platforms. This methodology was amply demonstrated in the framework of CyCab mobile robotic applications.

We are also involved in the definition of a methodology based on the new version of the ADL EAST-ADL2.0. EAST-ADL 2.0 is a UML2.0 profile dedicated to automotive systems. Our aim is first to formally introduce architectural concepts such as temporal characteristics, complex scheduling and resource allocation constraints in the process EAST and to found out the transformations rules from East model elements to formalisms such as Petri Net and synchronous models. The second objective is to apply validation technics for temporal verification, architectural constraints or ressource allocation of an EAST description at the different level of the EAST Process.

New standards are emerging in this field, such as AADL in avionics and AUTOSAR in automotive. They are strongly linked to UML representation models, as specific profiles. In both cases we are considering the connections with our MARTE profile effort, and the precise temporal semantics that we try to endow these formalisms with.

4.2. Mobile phones and other communicating objects

Such systems usually combine intensive (multimedia) data processing with mode control switching, and wireless or on-chip communications. The issue is often here to integrate design techniques for the three domains (data, control, communications), while preserving modular independence of concern as much as possible. At high-level modeling this translate into combining models of computations that are state-oriented (imperative) or datapath-oriented (declarative), with appropriate communication models, while preserving the sound semantics of the systems built from all such kinds of components. Dedicated architecture platforms here usually associate general-purpose processor (ARM) with specific DSP coprocessors. In the future the level of integration should become even higher, with the corresponding challenges for design methodologies.

4.3. System-on-Chip design

While design of digital circuits is already a fairly complex development process, involving many modeling and programming stages, together with intensive testing and involved low-leval synthesis and place-and-route techniques, SoC design adds yet new complexity dimensions to this process. Fully synchronous designs are not feasible anymore, and custom IP reuse becomes mandatory to integrate full processor cores into a new design. New approaches are being proposed, which try to depart only as little as possible form the synchronous/cycle-accurate prevailing design techniques, while allowing more timing flexibility at interfaces between blocks. These approaches are generally flagged as GALS (Globally-Asynchronous/Locally-Synchronous). They usually put a stress on proper mathematical modeling at every stage, thereby revisiting and associating known models with new intent. Synthesis seen as model-transformation seems here a nice way to bring some of the OMG MDA schemes into true existence.

Here again new standards are emerging such as SPIRIT for the representation of SoC structure and behavior, both at low (RTL) and high (TLM) description level. Again we are considering these in connection with our modeling approach.

5. Software

5.1. SyncCharts/Esterel

Keywords: *Esterel*, *SyncCharts*, *compilation*, *static analysis*, *verification*.

Participants: Charles André, Robert de Simone, Dumitru Potop-Butucaru.

The main software development activities concerning synchronous formalisms went to the ESTEREL TECHNOLOGIES company as it was spun-off from the former Meije team. We still carrry some experimental development on the former academic versions of Esterel and SynCharts, mostly to validate new algorithmic model transformations or analysis.

We are reviewing experts for the IEEE standardization of ESTEREL. In the next furture we should be able to contribute code processors to the commercial environnement by connecting into exchange format files.

5.2. K-passa

Participants: Julien Boucaron, Robert de Simone, Jean-Vivien Millo.

We developed this analysis software to charaterize the feasible K-periodic as-soon-as-possible static schedules of a (strongly connected) latency-insensitive system. The software is written in Java, and uses the mascOpt library developed in the MASCOTTE team. This library in turn is based on the commercial solver CPlex, by Ilog, for linear constraint solving.

5.3. SynDEx

Participants: Christophe Gensoul, Yves Sorel.

SynDEx is a system level CAD software implementing the AAA methodology for rapid prototyping and for optimizing distributed real-time embedded applications. It can be downloaded free of charge, under INRIA copyright, at the url: http://www.syndex.org. It provides the following functionalities:

- specification and verification of an application algorithm as a directed acyclic graph (DAG) of operations, or interface with specification languages such as the synchronous languages providing formal verifications, AIL a language for automobile architectures, Scicos a Simulink-like language, AVS for image processing, CamlFlow a functional data-flow language, etc,
- specification and verification of a “multicomponent” architecture as a graph composed of programmable components (processors) and/or specific non programmable components (ASIC, FPGA), all interconnected through communication media (shared memory, message passing),
- specification of the algorithm characteristics, relative to the hardware components (execution and transfer time, period, memory, etc), and specification of the real-time constraints to satisfy (latencies, periodicities),
- exploration of possible implementations (distribution and scheduling) of the algorithm onto the multicomponent, performed manually or automatically with optimization heuristics, and visualization of a timing diagram simulating the distributed real-time implementation,
- generation of dedicated distributed real-time executives, or configuration of general purpose real-time operating systems: RTlinux, Osek, etc. These executives are deadlock free and based on off-line policies. Dedicated executives which induce minimal over-head are built from processor-dependent executive kernels. Presently executives kernels are provided for: ADSP21060, TMS320C40, TMS320C60, i80386, MC68332, MPC555, i80C196 and Unix/Linux workstations. Executive kernels for other processors can be easily ported from the existing ones.

The distribution and scheduling heuristics, as well as the timing diagram, help the user to parallelize his algorithm and to explore architectural solutions while satisfying real-time constraints. Since SynDEx provides a seamless framework from the specification to the distributed real-time execution, formal verifications obtained during the early stage of the specification, are maintained along the whole development cycle. Moreover, since the executives are automatically generated, part of tests and low level hand coding are eliminated, decreasing the development cycle duration.

SynDEx was evaluated by the main companies involved in the domain of distributed real-time embedded systems, and is presently used to carry out new products, as for example in companies such as Robosoft, MBDA, and IFP.

5.4. SynDEx-IC

Participants: Mohamed Akil [Professor ESIEE Noisy-Le-Grand], Thierry Grandpierre, Pierre Niang [PhD. student ESIEE Noisy-Le-Grand], Yves Sorel.

SynDEx-IC is a CAD software for the design of non-programmable components such as ASIC or FPGA for which, the application algorithm to implement is specified with the graph model of the AAA methodology. It is developed in collaboration with the team A2SI of ESIEE. It allows to specify the application algorithm like in SynDEx, and automatically synthesizes the data path and the control path of the specific integrated circuit as a synthetizable VHDL program while real-time and surface constraints are satisfied. Because these problems are again of NP-hard complexity, we propose greedy and iterative heuristics based on "loop-unrolling" of the algorithm graph, in order to solve them.

This integrated circuit synthesis was tested on image processing applications. Using SynDEx-IC we specified and implemented, for example, several digital image filters onto the XC2S100 SPARTAN FPGA based on executive kernel for synthetizable VHDL. We extended the architecture model in order to support the specificities of FPGA: internal memories, configuration of computational units, communication unit with other components. Using this extended architecture model, we modelled the architecture of different commercial Boards (Virtex II pro card from Xilinx, Stratix board from Altera) and applyied the graph transformations needed to obtain an optimized hardware implementation on this kind of architecture.

Such non-programmable components designed with SynDEx-IC may be in turn used in SynDEx in order to specify complex multicomponent architectures composed of non-programmable and programmable components all together interconnected. Presently, both softwares SynDEx and SynDEx-IC are separated, the hardware/software partitioning phase of co-design being done manually. We plan in the future to integrate SynDEx-IC in an unique software environment, and also to provide heuristics to automatically perform hardware/software partitioning.

6. New Results

6.1. A UML profile for Real-Time Embedded System modeling

Participants: Charles André, Gérard Cristau, Fadoi Lakhal, Frédéric Mallet, Robert de Simone, Yves Sorel.

General-purpose UML 2.0 and its new companion formalism SysML (for system modeling) are both heavily relying on modeling elements such as state and activity diagrams for behavior, component and block diagrams for structure, that were already in use for a long time in embedded system design flows (as in Matlab/Simulink, Statecharts, SCADE and Esterel Studio, Scilab/Scicos, SynDEx and many others). UML compatibility opens the door to generic tools and broader audience, but the lack of formal semantics in its representation diagrams can downplay these advantages. Therefore we studied ways to endow classical UML and SysML with such formal semantics. We had to do so in a way that is both compatible with the existing standard, and at the same time refine its temporal aspects to allow the clear semantic representation of our desired models [31]. We conduct this work as part of a contractual collaboration with Thales and CEA-List in the CARROLL PROTES project 7.3, and then towards the OMG as an official MARTE profile definition being put up with a larger consortium PROMARTE.

This year we worked mostly on the definition of a full-fledged *Time Model* which allows logical and physical time definition. In particular it introduces (virtual) *clocks*, based on possibly loosely-related *time bases*, to represent asynchrony between concurrent entities. A primitive set of clock relations is provided and considered to link further the distinct clocks occurring in a system. Solving these clock relations amounts to providing a specific schedule for the system, and thus we also introduce notations to represent such schedules as first-class syntaxic components of the design approach. A preliminary report [39] and a technical workshop paper [32] describe this profile and the clock relations, and a keynote address was presented at IES'2006 (First IEEE international sympsium on Industrial Embedded Systems) on the general approach [38].

The Time modeling is then exploited to build useful Models of Computation and annotated both the Application models and the architectural Execution Platform model of a general UML system representation. A *Timed Allocation* model is also defined, extended similar notions introduced in SysML, which actually match already fairly closely the modeling approach promoted in Aoste for Application/Architecture Adequation.

We also realized a UML 2.0 meta-model of SynDEx, to become interoperable with the Polychrony and Gaspard tools developed respectively in the Espresso and DaRT Inria teams. This was implemented using the MagicDraw modeler. A SynDEx profile was designed and realized. It allows the specification in a UML framework of the application algorithm model, the distributed architecture model, and the association model of SynDEx. The association model representation characterizes the operations and the data dependences relative to the processors and communication media. These models may be translated into XMI files, which are in turn translated into SynDEx syntax files.

6.2. Formal modeling of latency-insensitive systems

Participants: Julien Boucaron, Robert de Simone, Jean-Vivien Millo.

Due to the the increasing complexity of embedded systems, the fully synchronous design paradigm is no longer always adequate. This is true in particular in *SoC* design (*Systems-on-Chip*), where large systems are built up from many so-called IP components. The global connection wires then become so stretched that communication latencies have to be taken into account, as they now dominate the computation time of local components.

Latency Insensitive Design (LID) was invented to cope with this problem. It starts by de-synchronizing the original, ideal synchronous specificication. Fixed arbitrary latencies are then provided (computed elsewhere by the designer), and a new re-synchronized version is constructed that can handle such arbitray latencies. In effect it introduces appropriate buffering mechanisms (called *relay-stations*) and back-pressure flow-control protocols so as to guarantee that late data can be awaited for across latencies.

Last year we established the formal connection of LID theory with older theories of Marked/event Graphs with capacity-2 bounded places. This is obtained after expanding the latencies into transportation sections, as with relay-stations. There are known results in this domain, mostly due to Carlier-Chretienne and Baccelli-Quadrat *et al*, which prove the existence of ultimately periodic schedules for such systems under natural conditions (strong connectivity). This year we explored the systematic application of such results to first compute explicit schedule representations, which can be viewed as static solutions, and then use these representations to greatly simplify the actual implementations: the whole flow-control protocol mechanism can be discarded, and the number of needed relay-stations can be optimized according to their effective utilization. These results are reported in [34], [33], and we implemented these ideas in a toll called K-PASSA (for K-Periodic As-soon-as-possible Static Scheduling Analysis).5.2.

Still, we discovered several pathological examples, where optimization in the general case is not as dramatic as it could be. This led us to define a notion of *smooth* schedules and token distribution in the system, in [27]. We have posed a number of conjectured which, if true, could lead to even better schedules of Latency-Insensitive Designs. We are currently working on these proofs.

This work is conducted in close relation with the SYS2RTL project of the Design platform, in the CIM PACA regional initiative 8.1.1.

6.3. Globally asynchronous implementations of synchronous specifications

Participants: Dumitru Potop-Butucaru, Robert de Simone, Yves Sorel.

We are exploring the distributed globally asynchronous implementation of synchronous specifications. In the implementations we consider, the correct functional and temporal behavior of the implementation is ensured by (1) the sequencing of the operations that are scheduled on each processor and (2) the exchanges of messages ensuring the inter-processor synchronization. Our objective is to optimize inter-processor synchronization by minimizing the number of messages in order to minimize the communication overhead.

The primary practical application is the extension of the class of specifications and implementations the AAA methodology can support. Currently, it only handles operations where all inputs and outputs are sent/received in each infinite repetition of the graph pattern of the algorithm. This repetition corresponds to the logical instants of the Synchronous languages. In synchronous terminology, all inputs and outputs of an operation have the same clock – they are all present or all absent in every execution instant. The objective is to allow the specification and scheduling of operations/components where only part of the inputs and outputs are present in an infinite repetition depending on the state and on input data.

We carried out a study (ongoing) on the relations between (1) the specification formalism and implementation process of AAA and (2) the main synchronous languages (Esterel, Lustre, Signal) and their compilation techniques. The comparison also includes significant related paradigms, such as latency-insensitive design and endochrony. We completed the language comparison, and we are adressing implementation aspects, where more marked differences exist, due to the fact that AAA takes into account physical time (periods and latencies). In [41], we defined a method for synthesizing the asynchronous executives that are driving the synchronous modules of a globally asynchronous, locally synchronous (GALS) system. The technique takes as input high-level synchronization constraints, under the form of multi-clock modular synchronous reactive specifications. For each synchronous module, our technique produces a multi-rate executive that drives the communication and the clock of the component using a mixed static/dynamic scheduling policy. The resulting GALS system is predictable and functionally correct with respect to the initial synchronous specification. The approach is based on the theory of weakly endochronous systems, and on a notion of atomic reaction which allow us to exploit the concurrency of the specification to improve the communication efficiency of the executives.

We determined the class of synchronous specifications that can be transformed into Kahn-like (deterministic) asynchronous implementations without adding supplementary synchronization messages to encode the absence of signals. Specifications of this class can be easily transformed into either purely asynchronous implementations, or into GALS components that perform a reaction as soon as the inputs needed for this reaction have arrived. For general synchronous specifications, doing this produces non-deterministic implementations, which means that supplementary synchronization messages must be added to ensure determinism. This result establishes practical limits to the minimization of the number of messages in AAA.

6.4. UML Patterns for hardware/software architectures

Participants: Charles André, Frédéric Mallet, Marie-Agnès Peraldi-Frati.

We developed here a quantitative timing analysis techniques. It starts from from UML2.0 and SysML-based architecture descriptions, including functional and structural specification, resource allocations and QoS modeling. A formal description of behaviour using Time Petri Nets is extracted from these representations, on which the analysis is then effectively conducted. It explores the solution space and proves the existence of valid schedulings. A refinement process allows different levels of description of the application.

We currently extend this approach in order to integrates multi clock representation in the models and to proposes transformations rules from this representation to synchronous based models.

Contributions have been presented at the OMER3 workshop in paderborn and the IES'2006 conference in Antibes [36].

6.5. Preemptive off-line scheduling in the monoprocessor case

Participants: Patrick Meumeu, Alix Munier, Dumitru Potop-Butucaru, Yves Sorel.

We address here the scheduling problem of real-time systems with precedence and strict periodicity constraints in the monoprocessor case. For such systems, the main challenge for the designer is to guarantee that all the deadlines of all the tasks are met, otherwise dramatic consequences occur. Guaranteeing deadlines is not certainly achievable because the preemption is approximated within the worst case execution time (WCET) of the tasks [53], [52] when using classical approaches such as RM (Rate Monotonic), DM (Deadline Monotonic), EDF (Earliest Deadline First), LLF (Least Laxity First), etc. This approximation may be wrong since it is difficult to count the exact number of preemptions of each instance for a given task even though the cost of one preemption is easy to be known for a given processor. Consequently, this approximation may lead to a wrong real-time execution whereas the schedulability analysis concluded that the system was schedulable. To cope with this problem, the designer usually allows margins which are difficult to assess, and thus in any case lead to a waste of resources. From now on, to clearly distinguish between the specification level and its associated model, we will use the term "*operation*" instead of the commonly used "*task*" which is too closely related to the implementation level. Thus, given a set of n periodic preemptive operations $\tau_i, 1 \leq i \leq n$ with precedence and strict periodicity constraints, we consider that each operation τ_i is an infinite sequence of instances τ_i^k, $k \in \mathbb{N}$, and is characterized by a WCET C_i, not including the approximation of the cost of the preemption and a period T_i.

In order to solve all the difficulties due to this approximation of the preemption cost, during the two last years, on the one hand we formally stated the problem, and on the other hand we gave preliminary results about the introduction of the preemption cost within the scheduling problem of real-time systems with precedence and strict periodicity constraints [5]. To do so, we have denoted by V the set of all systems of operations and we have given a partition of V into five subsets. Among these subsets, we showed that three of them were made of non schedulable systems. They formally consist of systems where at least two operations have co-prime periods, systems with at least two operations such that the time elapsed between their first start times is a multiple of the greatest common divisor of their periods and at last, systems with overlappings, i.e. where the start time of an instance of an operation occurs while the processor is executing another instance of a previously scheduled operation. We denoted by V_r, and V_i the two remaining subsets which are said to be potentially schedulable when the exact total load factor of the processor is less than 1. Note that V_r stands for the subset of regular operations, i.e. where the periods of all operations form a geometric sequence relatively to the precedence relations whereas V_i stands for the subset of irregular operations, i.e. where there exists at least two operations whose periods are not multiple [37]. For V_r, we proposed a scheduling algorithm which constructs all the possible preemptive schedules relatively to the strict periodicity constraint of the operations and to the possible durations that can be taken by each operation, this assumes that the duration of an operation cannot be greater than its period (implicit deadline). Although this scheduling algorithm gave the number of preemptions and although we got a new schedulability condition, unfortunately, that number did not include the preemptions due to the increase in the execution time of the operations because of the cost of preemptions themselves.

Thus, this year before solving the complex scheduling problem of sytems in V_i, firstly, we improved the schedulability condition for systems in V_r by giving the exact number of preemptions including those due to the increase in the execution time of the operations because of the cost of the preemptions themselves. This improvement leads to a schedulability condition which takes accurately into account the cost of the preemption when the cost of one preemption is known for a given processor, and also provides the first start time of all the operations when the systems is schedulable. This new condition always guarantees a correct execution and eliminates the waste of resources since no margins are necessary. To the best of our knowledge there are no result about this problem.

Secondly, we studied the simpler scheduling problem of independent preemptive periodic tasks in the same context as previously described. Our approach uses algebra, the set theory and the theory of numbers. It builds all the possible preemptive schedules relatively to all the execution times that can be taken by each task, when all tasks are released at the same time, i.e. the usual critical instant [53]. We use the availables time units left in each instance of a given task in order to predict the behavior of the next task relatively to the priorities of the other tasks. This prediction is made by using a modulo T_i algebra, where T_i denotes the period of a task τ_i. This method counts the exact number of preemptions including those due to the increase in the execution time of the tasks because of the cost of the preemptions themselves. This method leads to a stronger schedulability condition than the classical condition given in [53]. In addition, it will be reusable in the case of systems in V_i. Such a result is of great interest for the real-time scheduling community where the preemption is always approximated.

6.6. Non-preemptive off-line scheduling in the multiprocessor case

Participants: Liliana Cucu, Omar Kermia, Yves Sorel.

The last year we studied the non-preemptive multiprocessor scheduling of systems with precedence and periodicity constraints. There exists a lot of methods to solve this kind or problems, but because we aim at rapid prototyping the best suited ones are greedy heuristics that have are very fast and have quite good results. Then, according to this bibliographic studies, we decided to improve the greedy heuristic used in SynDEx which performs a distribution and scheduling for systems with precedence constraints but with only one period instead of multiple periods. In order to satisfy the multiple periodicity constraints, we extended this heuristic as follows. We execute a first algorithm which unrolls the initial graph of operations (algorithm graph) over the hyper-period H which is the Least Common Multiple of all different periods, where each operation of period P is repeated H/P times. The required edges to keep the data transfer are added to the graph. Then, we execute a second algorithm which assigns the operation classes, obtained according to the different periods, to the processors of the architecture (architecture graph). All the possibilities, due to the differences between the number of classes and the number of processors, are taken into account. Finally, the proposed heuristic is composed of these two algorithms executed in sequence followed by the heuristic used in SynDEx. The latter is extended with two conditions for satisfying periodicity constraints. The proposed heuristic has a $N_{unr}^2 M$ complexity (N_{unr} is the number of operations in the unrolled graph and M is the number of processors). This complexity is similar to the one of the heuristic used in SynDEx whose effectiveness was experimentally proved since a long time.

This year we focused our bibliographic study on data transfer in distributed periodic systems. It appears that the communicating operations must be at the same or multiple periods in order to guarantee that all the necessary data transfers are performed. Also, it appears that in realistic applications the number of different periods for a given system is small, and in addition these different periods often are multiple. Consequently, the hyper-period value is not very large. These two observations allowed us, on the one hand, to improve the graph unrolling algorithm. In this latter the edges addition between the repetitions of the operation which produce data and those which consume data, becomes simple. On the other hand we have the certitude that the unrolling graph could not be significantly larger than the initial one because of the small hyper-period value [35].

Because the multiprocessor scheduling with precedence and periodicity constraints is very complex, we performed a scheduling analysis with the same constraints in the monoprocessor case. Then, we use these results to propose multiprocessor scheduling heuristics. This analysis showed that to maximize the scheduling success ratio of a system onto a multiprocessor, the operations with the same or multiple periods must be executed on the same processor. In addition, this principle has the advantage of reducing the communication costs. This analysis led to scheduling conditions which allows us to know if some operations are schedulable on the same processor or not. The proposed heuristic has been implemented in the SynDEx software and tested extensively on academic examples.

In order to evaluate the performances of our heuristic we compared it to an optimal algorithm. We implemented a “Branch and Cut” algorithm which distributes and schedules systems with precedence and periodicity constraints. This algorithm effectively explores the search tree and uses the conditions obtained from the scheduling analysis to decide, at each step of the solution construction, if it proceeds or do back-tracking. The comparison between our heuristic and the optimal algorithm showed that our heuristic had a bad scheduling success ratio, and that the problem comes from the assignment algorithm. By looking closely to this latter it appeared that the increasing sort used to define the order the operations are assigned to the processors, reduces the scheduling possibilities. In order to make our heuristic more effective we replaced the increasing sort by a mixed sort which takes into account two criteria first the priority level and second the increasing order of the period. The priority order is given to every operation relatively to the number of operations the periods of which divide this operation period. Thus, we perform an increasing sort according to the priority level, and when several priority levels are equal we perform an increasing sort according to the period only. The tests performed on the new version of our heuristic are very satisfying: the scheduling success ratio is greater than 90% and it is very fast.

In the future we plan to study if it is possible to improve the assignment algorithm by taking into account the worst case execution times of the operations whereas presently we take into account only the periods. That could increase the scheduling success ratio. We plan also to investigate other distribution and scheduling heuristics, and especially meta-heuristics which will certainly have a best scheduling success ratio but will not be so fast.

6.7. Mixing off-line and on-line scheduling for aperiodic operations with latency constraints

Participants: Lyes Abdelfettah, Liliana Cucu, Nicolas Pernet, Yves Sorel.

The AAA methodology is presently based on off-line heuristics to find a schedule satisfying in addition to precedence and strict periodicities, a new type of real-time constraints [28]. A latency constraint concerns a pair of operations and defines the maximum amount of time which can separate the beginning of the first operation and the end the second one. We prove in [22] that a latency constraint can not be replaced by two deadlines without reducing the set of satisfying schedule and thus over-constraining the system. Off-line scheduling reduces execution overhead because it does not require a scheduler as on-line scheduling do. The drawback of these approaches is that they are not suited for aperiodic operations, that is to say, operations whose occurrence time is unknown contrary to periodic operations. Thereby, we aim at handling on-line aperiodic operations while satisfying all the real-time constraints of the periodic operations scheduled off-line.

Previous works on aperiodic operations rely on specific preemptive scheduling algorithms for periodic operations (Rate Monotonic, Earliest Deadline First). Consequently, they are unusable for solving our problem based on non preemtive scheduling with precedence, strict periodicity and latency constraints. The only method which does not impose such restrictions is the Slot Shifting [49].

The work, detailed in [22], consists in extending the Slot Shifting algorithm to deal with several scheduling constraints. Since the complexity of the algorithm depends on the number of constraints, we give three theorems which allow us to construct a subset of the off-line schedule constraints such as satisfying them implies to satisfy the whole set of constraints. Then we propose two approaches to extend Slot Shifting algorithms to take into account latency constraints.

The first one consists in translating, off-line, the latency constraints satisfied by the off-line schedule into deadlines. For a latency constraint on a pair of operations, the translation sets a deadline for the second operation and this deadline is related to the estimated start time of the first one. On-line aperiodic tasks execution may delay the start time of the off-line scheduled operation and consequently the deadlines corresponding to latency constraints may move. To take into account these changes, we propose an algorithm which has to be executed before each on-line decisions. Although this approach is optimal relatively to response time and aperiodic tasks acceptation, the complexity of the algorithm forbids its use in realistic real-time systems.

The second one consists in translating, on-line, the latency constraints satisfied by the off-line schedule into deadlines. For a latency constraint on a pair of operations, the translation is performed when the first operation of a latency constraint starts its execution, and it sets a deadline for the second one. Consequently, such deadline is fixed and it does not move. Nevertheless, in order to guarantee that all constraints are satisfied, this translation can not be applied to every latency constraint. We give an algorithm which, off-line, detects the latency constraints which can not be translated on-line without potentially causing a missing constraint.

These latency constraints are translated off-line into fixed deadlines. Contrary to the first approach, these deadlines are fixed and consequently the optimality is unreachable except if the set of off-line translated latency constraints is empty. However, this approach does not increase significantly the on-line algorithm and is totally usable in realistic real-time systems.

6.8. Improvements in SynDEx

Participants: Christophe Gensoul, Omar Kermia, Djamel Louar, Daniel de Rauglaudre, Mickaël Raulet, Yves Sorel.

This year we began a major redesign of the SynDEx software data structures to ease the verification tasks applied onto the hierarchical specification. We designed these data structures with performance in mind, and we rewrote the flattening algorithm (which turns the hierarchical specification into a flat graph) using these data structures. This work improved the performance of the flattening algorithm by a factor of 30 in least favorable situations and up to a factor of 1000 in the most favorable ones. It also greatly improved maintenability of this part of the software.

SynDEx allows repetition of subgraphs of operations which is the data-flow equivalent of loop in imperative languages. We designed a new algorithm to discover automatically, by reading the dimensions of input and out put data, repeated patterns inside hierarchical specifications. It allows the flattening algorithm to extract more parallelism from hierarchical specifications than previous versions. A new adequation algorithm taking periodic constraints into account is also in progress. This is a new important feature that the users are waiting for a long time

We continued our bug-fixing activities upon demand by SynDEx's users, and we continued to improve the code generator in collaboration with the ARTIST team of INSA Rennes (which works on memory optimizations). Finally, we continued to improve the software architecture of the OCaml SynDEx program, to help with its maintenance and its further evolutions.

SynDEx generates automatically a macro-code which is independent of the distributed architecture. Then, this macro-code is macroprocessed with M4, using architecture dependent executive kernels, in order to obain the executable codes running onto different the processors and communication media. Because RTAI/Linux is a free software real-time operating system becoming a standard in the academic world, and used more and more in the industry, we designed a Linux/RTAI executive kernel. We demonstrated is efficiency and flexibility on simple applications running onto several workstations under Linux/RTAI communicating through an Ethernet bus.

Now Scilab/Scicos is able to generate SynDEx code. When both tools are coupled this provides to the designer a seamless environment from the high level specification of automatic control models to their implementation onto distributed heterogeneous architectures (multicomponent) while satisfying real-time constraints and minimizing resources. Such coupling associated with compliant semantics for both tools increase the safety of the designed systems [15].

Last years, in collaboration with the ARTIST team of INSA Rennes, we studied the data memory optimization with colored graphs. We proposed a technique to minimize intra-processor data memory, and then data memories used for inter-processor communications. The latter is more complicated because it ask for modifying the synchronization mechanism used to guarantee that there is no inter-processor communication deadlock [23]. The first techniques was implemented in SynDEx reducing significantly the amount of data memory in each processor of the distributed architecture.

7. Contracts and Grants with Industry

7.1. ST MIcroelectronics

Participants: Julien Boucaron, Robert de Simone.

This collaboration takes place with the Smart Card division, located at the Rousset site. The goal is to study abstraction/refinement techniques for SoC specification, in a way inspired from the B Method, but using synchronous languages concepts instead. Julien Boucaron PhD thesis is largely funded on this contract. Clearsy is another partner in the project, and they are using the B method for similar aims. We are currently investigating a common case study around interrupt handling in embedded processors, with USB protocol support.

7.2. Texas Instruments

Participants: Julien Boucaron, Jean-Vivien Millo, Robert de Simone.

This Grant by Texas Instruments funds us to conduct further studies on the theory of *Latency-Insensitive Design*, in the scope of the so-called *timing closure* issue encountered when assembling a system composed of many IP components, where the lengths of long global wires may impair full synchronous semantics. The contract has been renewed three times now.

This year we focused on the potential benefits of static, ultimately periodic schedules of Weighted Marked Graphs to help optimize the allocation of additional *Realy-Stations* and back-pressure mechanisms as demanded by (dynamic) LID theory. The reults are exposed in section 6.2, and led to a presentation at the SAME forum [33]. They are also bearing close relations with our joint involvement in the CIM-PACA SYS2RTL project.

7.3. CARROLL PROTES

Participants: Charles André, Gérard Cristau, Anaud Cuccuru, Robert de Simone, Yves Sorel.

CARROLL is a joint initiative between Thales, CEA, and INRIA to launch collaborative projects, mostly on model-driven engineering. PROTES was initiated by CARROLL to foster a specific OMG UML profile on Real-Time Embedded Modeling and Analysis, named MARTE. We are participating together with the Espresso and DaRT INRIA teams. A dedicated consortium named PROMARTE was put up to write a proposal for the subsequent Request-For-Proposal (RFP), which was issued as a result of our efforts in january 2005. Together with the formal participants the consortium contains all prominent UML tool vendors, such as Rational/IBM, I-Logix/TeleLogic, Artisan, Mentor Graphics,... The initial submission should be ready by March 2007. The MARTE profile should then be used in a number of collaborative projects, such as openEmbeDD, MemVatex, or OpenDevFactory.

This work involved participation from Aoste members to several OMG Technical meetings in the US (Washington DC, Tampa, Boston, Anaheim), as well as internal progress meetings held in France, about every two months. Our results on time modeling presented in section 3.1.3 were included in the draft standard.

The project should be followed by the CORTESS follow-up in 2007.

7.4. MBDA

Participants: Christophe Gensoul, Yves Sorel.

MBDA develops with AAA/SynDEx a new automatic guidance application involving an algorithm with more than 6000 operations executed at different periods, whereas the architecture is made of several PowerPC and ASICs all interconnected through a crossbar.

8. Other Grants and Activities

8.1. Regional collaborations

8.1.1. CIM PACA

Participants: Jean-Vivien Millo, Robert de Simone.

This ambitious regional initiative is intended to foster collaborations between local PACA industry and academia partners on the topics of microelectronic design, though mutualization of equipments, resources and R&D concerns. We are actively participating in the Design Platform (one of the three platforms launched in this context). Other participants are UNSA, CNRS (I3S and LEAT laboratories), L2MP Marseille, CMP-ENSE Gardanne on the academic side, and Texas Instruments, Philips, ST Microelectronics, ATMEL, and Esterel Technologies on the industrial side.

Inside this platform we are coordinating a dedicated project, named Spec2RTL, on methodological flows for high-level SoC synthesis. Participants are Texas Instruments, NXP, ST Microelectronics, Synopsys, Esterel Technologies as industrial partenrs, INRIA, I3S (CNRS/UNSA) and ENST on the academic side. A pool of PhD students are funded on a par basis between industrial partners and local PACA PhD grants under the BDI programme. There are currently 4 such students, one of them hosted by the Aoste team in conjunction with ST Microelectronics. Main research topic is LID design for GALS systems.

8.1.2. Competitivity Poles

Participants: Charles André, Robert de Simone, Yves Sorel.

We are taking part in the OPENDEVFACTORY subproject of the regional SYSTEM@TIC *Software Factory* programme, launched in *Ile-de-France*.

Our contribution in this project is to extend the applicability of the MARTE UML profile by implementation into a commercial or public-domain modeling tool, by connecting it with analysis and transformation tools (such as SYNDEX, and by applying it to industrial case studies (in relation with IFP mostly).

The problem proposed by IFP as case study in OpenDevFactory is currently being modeled in Scilab/Scicos. We use the new features of Scilab/Scicos allowing SynDEx code generation to obtain the algorithm graph. Then we specify in SynDEx the architecture, composed of several Linux workstations. This enables us to run the case study on this architecture. Since Linux is not actually a real-time operating system we use the Linux/RTAI executive kernel to obtain a distributed real-time implementation of the case study.

8.2. Nation-wide collaborations

8.2.1. Relations with other INRIA teams

We have strong ties with INRIA teams ESPRESSO and DaRT through the PROTES initiative on synchronous and more generally RTE (Real-Time Embedded) modeling in UML. We conduct joint work with POP-ART on fault tolerance and adaptive scheduling for robotic applications. Together with the S4 team we regularly attend the same events gathering the “Synchronous languages” community. We wish to draw closer ties with ALCHEMY and PROVAL on the topic of synchronous and N-synchronous modeling, in relation to code distribution and parallel execution.

We also collaborate with IMARA team which develops with SynDEx new applications onto automatic vehicles such as the CyCab, and with METALAU on coupling of Scilab/Scicos with AAA/SynDEx. Historical links are preserved with the team SOSSO, on adaptive scheduling for applications mixing soft and hard real-time.

8.2.2. RNTL platform OpenEmbeDD

Participants: Charles André, Benoît Ferrero, Robert de Simone, Yves Sorel.

This is a large platform project aimed at connecting several formalisms with model-driven engineering tools, in the embedded domain. The project partners are: INRIA, CEA-List, Thales, Airbus, France Telecom, CS, LAAS, and VERIMAG. Four INRIA teams are involved (ATLAS, Triskell, Aoste and DaRT).

The focus is on the use of model-driven approaches to combine various specification formalisms, analysis and modeling techniques, into an interoperable framework. We contribute to this in several directions: first, we provide the definition and implementation of the MARTE profile, as described in 3.1.3; second, we contribute our work on compilation-by-transformation of synchronous program; last, we develop the meta-model and profile for the AAA-SynDEx methodology, and the transformation needed to couple the tools.

The various partner contributions in this project are assembled together by a dedicated engineer team of two people located at IRISA, as part of an INRIA forge.

8.2.3. RNTL project MemVatex

Participants: Djamel Louar, Fadoi Lakhal, Marie-Agnès Peraldi-Frati, Dumitru Potop-Butucaru, Yves Sorel.

The partners in this project are: Siemens-VDO, INRIA, CEA-List, CNRS-UTC, and Embelec. The focus is on the tracability and the validation of requirements in a methodology for automotive applications. This methodology has to be defined in the project and is based on the new standards EAST-ADL2 and Autosar. Both of them put UML and SysML has a the formalism in the design flow. The project is currently in the "homogeneous" phase centered arround UML formalisms. Inria contributes in the definition of the methodology by introducing a formal integration of temporal and architectural characteristics. This project provides an interesting and complex industrial case study for improving the MARTE Results on multiclock and our researchs on UML patterns for hardware software architecture. In a second phase called "heterogeneous" we will integrate synchronous formalism (ESTEREL, SYNDEX) in order to enrich the methodology to real-time models and to apply the associated validation technics and tools.

8.3. European collaborations

8.3.1. IST Network of Excellence ARTIST2

Participants: Julien Boucaron, Robert de Simone, Frédéric Mallet, Dumitru Potop-Butucaru, Yves Sorel.

Our participation here consists essentially (as for many other partners) in attending working group presentation meetings (without real collaborative work so far). We follow particularly the work of Working Group 1 on Hard Real-Time, with focus on Synchronous languages, Time-Triggered architectures and fixed priority scheduling.

Frédéric Mallet attended an international Symposium held in Vienna on automotive modeling (“Beyond AUTOSAR”) in this context.

8.4. Research exchange visits

Participants: Julien Boucaron, Luca Carloni, Robert de Simone, Stephen Edwards, Dumitru Potop-Butucaru, Olivier Tardieu.

This year was the second of our team-associationship with Columbia University, named HIDES. In the framework of this collaboration we had several exchange visits. Julien Boucaron and Jean-Vivien Millo visited Columbia for a week in June. Stephen Edwards and Olivier Tardieu visited us at Sophia-Antipolis in April for a month, and in Olivier’s case for another two weeks in July. Robert de Simone visited Columbia again in late October for a week.

As an important outcome of this collaboration, a book on Esterel compilation is in press [14].

9. Dissemination

9.1. Leadership within scientific community

Robert de Simone was program committee member for Memocode’06. He is also member of the *Commission de Spécialiste UNSA 27^{e} section*, and INRIA representative to the CIM PACA regional initiative on Microelectronics design; this includes being appointed to the Strategic Committee of the ARCSIS mother association, and member of the Board of Administrators of the Design Platform association. As INRIA leader of the CARROLL PROTES project he attended several OMG Technical Meetings at various locations in the US. He is member of the International Advisory Board for the CRC Press on Embedded Systems. He gave an invited keynote address at the First international Symposium on Industrial Embedded Systems in Antibes in November, and was reviewer for the thesis of O. Labbani (LIFL).

Charles André is member of the *Commission de Spécialiste UNSA 61^{e} section*. He was Program Chair of IES'2006, the First international Symposium of Industrial Embedded Systems, held in Antibes. He served as Program Committee member for MSR'2006. He was reviewer for the HDR of J-P. Babau, and the PhD theses of M. Feredj (Orsay), Ch. Mraidha (Evry) and S. Rouxel (Lorient).

Dumitru Potop-Butucaru and Gérard Berry, together with Stephen Edwards (U. Columbia), wrote a book on Esterel compilation and semantics currently in press.

Yves Sorel leads the Theme C Working Group (Adequation Algorithme Architecture) of the PRC-GDR ISIS (Information Signal Images et viSion). He is Program Member for the following conferences and workshops: JFAAA, ERTS, EUSIPCO, GRESTSI, JEAI, SYMPA, RTS. He is permanent member of the LCPC Scientific Committee, of the CARLIT-ONERA Scientific Committee, and of the DETIM-ONERA Evaluation and Orientation Committee. He participated to the following PhD jurys: A. Marchand, O. Marchetti, A. Meena, N. Pernet, M. Raulet.

9.2. Teaching

Robert de Simone taught courses on Formal Methods and Models for Embedded Systems in the STIC Research Master program of the university of Nice/Sopia-Antipolis (UNSA), and at ISIA-EMP (an engineering school located in Sophia-Antipolis), each time for approximately 24h.

Yves Sorel teaches at ESIEE (an engineering school located in Noisy-le-Grand), in the Research Master cursus at the University of Orsay Paris 11, and at ENSTA (an engineering school located in Paris), on topics comprising the AAA methodology, formal modeling and optimization of distributed embedded systems.

Charles André is a Professor at the University of Nice-Sophia Antipolis, department of Electrical Engineering. He teaches sequential circuits, discrete event systems, computer architecture and real-time programming. He also teaches “synchronous programming” and “UML for engineering systems” at the university polytechnic: EPU (options electrical engineering (Elec) and sofware engineering (SI)) and in the STIC research master. He provided didactic software as teaching material for basic architecture simulation [25].

Marie-Agnès Peraldi-Frati gives courses at different cursus levels of UNSA: A course and labs on real-time distributed systems in the STIC reaserch master (Embedded systems) and the STIC professionnal master (STREAM01/EPU), differents courses (Programming, Web developpment, Computer architecture) at the L1 level of the IUT Informatique. She is responsible for the option "Informatique embarquée et réseau sans fil” of the LPSIL cursus. She is member of the CERTEC (conseil d'études et de la recherche technologique) of the IUT of NICE.

Frédéric Mallet is Associate Professor at the University of Nice-Sophia Antipolis, department of Informatics. He teaches Object-oriented Programming at all levels from very beginners to Master level courses, and on all platforms from javacard, PDA, to standard operating systems. He also teaches Computer Architecture to undergraduate and graduate students.

10. Bibliography

Major publications by the team in recent years

[1] C. ANDRÉ. *Encyclopédie de l'informatique et des systèmes d'information. Tome 1: La dimension technologique des systèmes d'information. Section 7: Systèmes temps réel*, chap. L'approche synchrone pour le développement des systèmes temps réel - Chapitre 5, Vuibert, France, 2006, p. 151–166.

[2] C. ANDRÉ. *Representation and Analysis of Reactive Behavior: a Synchronous Approach*, in "Computational Engineering in Systems Applications (CESA)", IEEE-SMC, 1996, p. 19–29.

[3] A. BENVENISTE, P. CASPI, S. EDWARDS, N. HALBWACHS, P. L. GUERNIC, R. DE SIMONE. *Synchronous Languages Twelve Years Later*, in "Proceedings of the IEEE", January 2003.

[4] F. BOUSSINOT, R. DE SIMONE. *The Esterel Language*, in "Proceedings of the IEEE", September 1991.

[5] L. CUCU, Y. SOREL. *Schedulability condition for systems with precedence and periodicity constraints without preemption*, in "Proceedings of 11th Real-Time Systems Conference, RTS'03, Paris", March 2003.

[6] L. CUCU, Y. SOREL. *Non-preemptive multiprocessor scheduling for strict periodic systems with precedence constraints*, in "Proceedings of 23rd Annual Workshop of the UK Planning and Scheduling Special Interest Group, PLANSIG'04, Cork, Ireland", December 2004.

[7] L. CUCU, Y. SOREL. *Periodic real-time scheduling: from latency-based model to deadline-based model*, in "Proceedings of the Multidisciplinary International Conference on Scheduling: Theory and Applications, MISTA'05, New-York, USA", July 2005.

[8] A. DIAS, C. LAVARENNE, M. AKIL, Y. SOREL. *Optimized Implementation of Real-Time Image Processing Algorithms on Field Programmable Gate Arrays*, in "Proceedings of Fourth International Conference on Signal Processing, ICSP'98", 1998.

[9] C. DIMA, A. GIRAULT, Y. SOREL. *Static fault-tolerant real-time scheduling with "pseudo-topological" orders*, in "Proceedings of Joint Conference on Formal Modelling and Analysis of Timed Systems and Formal Techniques in Real-Time and Fault Tolerant System, FORMATS-FTRTFT'04", LNCS, vol. 3253, Springer-Verlag, 2004.

[10] A. GIRAULT, H. KALLA, Y. SOREL. *A Scheduling Heuristics for Distributed Real-Time Embedded Systems Tolerant to Processor and Communication Media Failures*, in "International Journal of Production Research", vol. 42, n^{o} 14, July 2004, p. 2877–2898.

[11] T. GRANDPIERRE, C. LAVARENNE, Y. SOREL. *Optimized Rapid Prototyping For Real-Time Embedded Heterogeneous multiprocessors*, in "Proceedings of 7th International Workshop on Hardware/Software Co-Design, CODES'99", 1999.

[12] T. GRANDPIERRE, Y. SOREL. *From Algorithm and Architecture Specification to Automatic Generation of Distributed Real-Time Executives: a Seamless Flow of Graphs Transformations*, in "Proceedings of First ACM and IEEE International Conference on Formal Methods and Models for Codesign, MEMOCODE'03, Mont Saint-Michel, France", June 2003.

[13] D. POTOP, R. DE SIMONE. *Optimizations for Faster Execution of Esterel Programs*, in "MEMOCODE'03", 2003.

[14] D. POTOP-BUTUCARU, S. EDWARDS, G. BERRY. *Compiling Esterel*, to appear, Springer, 2006.

[15] Y. SOREL. *From modeling/simulation with Scilab/Scicos to optimized distributed embedded real-time implementation with SynDEx*, in "Proceedings of the International Workshop On Scilab and Open Source Software Engineering, SOSSE'05, Wuhan, China", October 2005.

[16] Y. SOREL. *Massively Parallel Systems with Real Time Constraints, the Algorithm Architecture Adequation Methodology*, in "Proceedings of Conference on Massively Parallel Computing Systems, MPCS'94, Ischia, Italy", May 1994.

[17] Y. SOREL. *Real-Time Embedded Image Processing Applications using the AAA Methodology*, in "Proceedings of IEEE International Conference on Image Processing, ICIP'96, Lausanne, Switzerland", September 1996.

[18] E. VECCHIÉ, R. DE SIMONE. *Syntax-driven optimizations for Reachable State Space construction of Esterel programs*, in "International Journal of Embedded Systems", April 2005.

[19] A. VICARD, Y. SOREL. *Formalization and Static Optimization for parallel implementations*, in "Proceedings of Workshop on Distributed and Parallel Systems, DAPSYS'98, Budapest, Hungary", September 1998.

[20] R. DE SIMONE, D. POTOP-BUTUCARU, J.-P. TALPIN. *The Synchronous Hypothesis and Synchronous Languages*, in "Embedded Systems Handbook", chapter of the Embedded Systems Handbook, CRC Press, 2005.

[21] R. DE SIMONE, O. TARDIEU. *Loops in Esterel*, in "ACM Transactions on Embedded Computing Systems", 2005.

Year Publications

Doctoral dissertations and Habilitation theses

[22] N. PERNET. *Implantation distribuée temps réel de programmes conditionnés à l'aide d'ordonnancements mixtes hors-ligne/en-ligne de tâches périodiques avec contraintes de latence et acceptation de tâches apériodiques*, Ph. D. Thesis, Université Paris 6, jul 2006.

[23] M. RAULET. *Optimisation mémoire dans la méthodologie AAA pour code embarqué sur architectures parallèles*, Ph. D. Thesis, Institut des Sciences Appliquées de Rennes, may 2006.

Articles in refereed journals and book chapters

[24] C. ANDRÉ. *Encyclopédie de l'informatique et des systèmes d'information. Tome 1: La dimension technologique des systèmes d'information. Section 7: Systèmes temps réel*, chap. L'approche synchrone pour le développement des systèmes temps réel - Chapitre 5, Vuibert, France, 2006, p. 151–166.

[25] C. ANDRÉ. *Simulateur pédagogique d'architecture machine*, in "J3eA (Journal sur l'enseignement des sciences et technologies de l'information et des systèmes", Hors série 2, vol. 5, 2006.

[26] J. BOUCARON, J. MILLO, R. DE SIMONE. *Another Glance at Relay Stations in Latency-Insensitive Design*, in "Electr. Notes Theor. Comput. Sci.", vol. 146, n^{o} 2, 2006, p. 41–59.

[27] J. BOUCARON, JEAN-VIVIEN. MILLO, R. DE SIMONE. *Formal Methods of Scheduling for Latency-Insensitive Designs*, in "EURASIP journal on Embedded Systems", to appear, 2006.

[28] L. CUCU, N. PERNET, Y. SOREL. *Periodic real-time scheduling: from deadline-based model to latency-based model*, in "Annals of Operational Research", to appear, 2006.

[29] D. Potop-Butucaru, B. Caillaud, A. Benveniste. *Concurrency in synchronous systems*, in "Formal Methods in System Design", vol. 28, 2006, p. 111-130.

[30] D. Potop-Butucaru, B. Caillaud. *Correct-by-construction asynchronous implementation of modular synchronous specifications*, in "Fundamenta Informaticae", to appear, 2006.

[31] R. de Simone, C. André. *Towards a "Synchronous Reactive" UML profile ?*, in "International Journal on Software Tools for Technology Transfer (STTT)", vol. 8, n^{o} 2, 2006.

Publications in Conferences and Workshops

[32] C. André, A. Cuccuru, R. de Simone, T. Gautier, F. Mallet, J. Talpin. *Modeling with logical time in UML for real-time embedded system design*, in "MARTES, satellite workshop of Models'2006, Genova", oct 2006.

[33] J. Boucaron, Jean-Vivien. Millo, R. de Simone. *Latency Insensitive Design: Dynamic and Static scheduling with proper formal devices*, in "SAME Forum", SAME, 2006.

[34] J. Boucaron, Jean-Vivien. Millo, R. de Simone. *Latency-Insensitive Design and Central Repetitive Scheduling*, in "IEEE-ACM International Conference MEMOCODE'06", IEEE Press, 2006, p. 175- 183.

[35] O. Kermia, L. Cucu, Y. Sorel. *Non-preemptive multiprocessor static scheduling for systems with precedence and strict periodicity constraints*, in "Proceedings of the 10th International Workshop On Project Management and Scheduling, PMS'06, Posnan, Poland", April 2006.

[36] F. Mallet, M. Peraldi-Frati, C. André. *From UML to Petri Nets for non functional Property Verification*, in "First IEEE Symposium on Industrial Embedded Systems (IES'06), Antibes, France", 2006.

[37] P. Meumeu, Y. Sorel. *Non-Schedulability Conditions for Off-line Scheduling of Real-Time Systems Subject to Precedence and Strict Periodicity Constraints*, in "Proceedings of 11th IEEE International Conference on Emerging technologies and Factory Automation, ETFA'06, Prague, Czech Republic", sep 2006.

[38] R. de Simone. *Models at Work, an industrially (ir)relevant keynote address ?*, in "First IEEE Symposium on Industrial Embedded Systems (IES'06), Antibes, France", oct 2006.

Internal Reports

[39] C. André, A. Cuccuru, R. de Simone, J.-P. Talpin. *Modeling with logical time in UML for real-time embedded system design*, RR INRIA 5895, Technical report, 2006, http://hal.inria.fr/inria-00071373.

[40] C. André, F. Mallet, M.-A. Peraldi-Frati. *Non-functional Property Analysis using UML2.0 and Model Transformations*, INRIA RR 5913, Technical report, 2006, http://hal.inria.fr/inria-00124874.

[41] D. Potop-Butucaru, R. de Simone, Y. Sorel. *From multi-clock constraints to multi-rate GALS executives*, Submitted for publication, Research Report, n^{o} RR-6021, INRIA, November 2006, http://hal.inria.fr/inria-00114032.

References in notes

[42] F. BACCELLI, G. COHEN, GEERT JAN. OLSDER, J.-P. QUADRAT. *Synchronization and Linearity: an algebra for discrete event systems*, John Wiley & Sons, 1992, http://www-rocq.inria.fr/metalau/cohen/SED/book-online.html.

[43] A. BENVENISTE, G. BERRY. *The Synchronous Approach to Reactive and Real-Time Systems*, in "Proceedings of the IEEE", vol. 79, n^o 9, September 1991, p. 1270-1282.

[44] J. CARLIER, P. CHRÉTIENNE. *Problèmes d'ordonnancement*, Masson, 1988.

[45] L. CARLONI, K. MCMILLAN, A. SANGIOVANNI-VINCENTELLI. *Theory of Latency-Insensitive Design*, in "IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems", 2001.

[46] J. DENNIS. *First Version of a Dataflow Procedure Language*, in "Lecture Notes in Computer Sci.", vol. 19, Springer-Verlag, 1975, p. 362-376.

[47] V. V. DONGEN, G. R. GAO, Q. NING. *A Polynomial Time Method for Optimal Software Pipelining*, in "Conference on Algorithms and Hardware for Parallel Processing", http://citeseer.ist.psu.edu/vandongen92polynomial.html.

[48] S. EDWARDS. *Languages for Digital Embedded Systems*, Kluwer, 2000.

[49] G. FOHLER. *Joint Scheduling of Distributed Complex Periodic and Hard Aperiodic Tasks in Statically Scheduled Systems*, in "Procedings of IEEE Real-Time Systems Symposium", 1995, p. 152-161.

[50] N. HALBWACHS. *Synchronous Programming of Reactive Systems*, in "Computer Aided Verification", 1998, p. 1-16, http://citeseer.ist.psu.edu/10686.html.

[51] E. A. LEE, D. G. MESSERSCHMITT. *Static Scheduling of Synchronous Data Flow Programs for Digital Signal Processing*, 1987.

[52] J. LEHOCZKY, L. SHA, Y. DING. *The rate monotonic sheduling algorithm: exact characterization and average case bahavior*, in "Proceedings of the IEEE Real-Time Systems Symposium", 1989.

[53] C. LIU, J. LAYLAND. *Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment*, in "Journal of the ACM", 1973.

[54] C. MEAD, L. CONWAY. *Introduction to VLSI systems*, Addison-Wesley, 1980.

[55] A. ZOMAYA. *Parallel and distributed computing handbook*, McGraw-Hill, 1996.