ModEasy
(MOdel Driven dEsign for
Automotive Safety embedded sYstem)

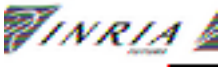
Investigation of the Problem

Deliverable D1

Edited by J-L Dekeyser

November, 2005

Version 5.0



Authors/Partners:

Partner	Author	Email
INRIA	Jean-Luc Dekeyser	dekeyser@lifl.fr
IEMN	Atika Rivenq	atika.menhaj@univ-valenciennes.fr
University of Kent	Klaus D McDonald-Maier	

Document History:

Date	Version	Editor	Description
02 may 05	0.1	Jean-Luc Dekeyser	Initial Document
21 june 05	0.1	Eric Rutten	
05 sept 05	V2	Eric Rutten	
22 sept 05	V3	Dave Akehurst	
07 oct 05	V4	Dave Akehurst	
03 nov 05	V5	Atika Rivenq	

Filename: DeliverableModEasy-1-nov05.doc

Table of Contents

1 Introduction

The current report represents the collective initial findings of the consortium forming the Interreg IIIa project entitled ModEasy (MOdel Driven dEsign for Automotive Safety embedded sYstem). It consists of a compound report incorporating material originally envisaged to comprise three separate reports on model, transformation and verification, the proposed application domain characteristics and requirements and SystemC, VHDL and FPGA synthesis. On considering the potential impact of the documentation produced, it was decided that these three reports would be best presented as a single coherent whole and the current report is the result of this decision. The overall structure of the document is designed to reflect the three primary constituent subject areas envisaged in the original deliverables however.

The ModEasy project seeks to develop software tools and techniques to aid in the development of reliable microprocessor based electronic (embedded) systems using advanced development and verification systems. The tools are to be evaluated in practical domains such as the automotive sector for reactive cruise control and anti-collision radar but are envisaged to be applicable for generic embedded systems in any safety and mission critical applications in the wider industrial domain. The project seeks to reduce development and production costs while maintaining existing high dependability and safety levels as embedded systems become more complex for many existing and new products across the Euro-region.

During the initial months of the project, considerable progress has been made in addressing the above goals and these are summarily documented below. Section 2 presents the progress achieved on the modelling and verification front and is divided into four sections, detailing the progress achieved in co-modelling, model driven engineering, investigating UML 2.0 feature enhancements and the implementation of Associations respectively. Each section introduces the goal of the particular investigation and details the development work achieved to date. Section 3 develops further ideas which have been examined with regard to verification techniques. Section 3.1 details the work undertaken with the regard to the problems and opportunities offered by Activity and Sequence diagrams found in UML while section 3.3 examines the opportunities offered by synchronous techniques in UML 2.0. Section 4 diverts the focus onto the problems presented by the simulation and synthesis of FPGA designs. It contains details of the investigations undertaken into SystemC and VHDL transformation tools and section 3.2 a survey of related formal verification approaches. Section 5 concludes the report with a specification of the two primary exemplars to be employed by the project in Automotive Cruise Control, GPS navigator characteristics and Anti-collision radar.

As a whole, the report may be taken as representing an overview of the current state of the art with respect to the topics under investigation and provides an insight into the current direction in which technology is progressing in addressing the issues under discussion in the modelling and formal verification of the embedded system development field.

2 Methodology

1.1 Co-modeling

Because of the vast scope of the encountered problems, of the quick evolution of the architectures, we observe a very great diversity as regards the programming languages. Portability of an application from one language to another (a new one for example) increases the workload of the programmer. This drawback is also true for the development of embedded applications. It is even worse, because the number of abstraction levels has to be added to the diversity of the languages. To develop an embedded application, it is essential to associate a target hardware architecture model to the application specification model, and to introduce as well a relationship between them.

These two models are practically always different; they are often expressed in two different languages. From this experience, one can derive some principles for the design of embedded application development environments:

To refrain from designing programming languages to express the two different models, application and architecture.

To profit from all the new systems dedicated to simulation or synthesis without having to reformalize these two models.

To use a single modeling environment possibly supporting a visual specification.

To benefit from standard formats for exchange and storage.

To be able to express transformation rules from model to model. Possibly the transformation tools could be generated automatically from this expression.

We believe that the Model Driven Architecture [MM03, Boa01] can enable us to design a new method of system design respecting these principles. Indeed, it is based on the common UML modeling language to model all kinds of artifacts. The clear separation between the models and the platforms makes it easy to switch to a new technology while re-using the old designs.

2.1.1 Y model

The multiplicity of the abstraction levels is appropriate to the modeling approach. The information is used with a different viewpoint for each abstraction level. This information is defined only once in a single model. The links or transformation rules between the abstraction levels permit the re-use of the concepts for a different purpose. In the "Y-chart" [GK83] approach, three domains are identified:

- Functional domain: algorithms, flowcharts, functional components.
- Structural domain: processors, memories, busses.
- Physical domain: delays, power consumption, hardware resources, real-time and embedding constraints.

We propose to specify specific informations in each of these three domains. The design

activities match a successive refinement process between each domain according to various abstraction levels.

The design flows from the functional domain, to the structural domain, finally to the physical domain, and so on while going down in the abstraction levels.

The Y-Model consists in the specification of an application, a hardware architecture and in the successive refinement of the mapping of the application on the architecture. In our MDA-style approach (Model Driven Architecure, a standard of the Object Management Group), the successive refinements are done by the way of model transformations: For instance an application model is associated with a hardware architecture model to produce a new model of the mapped application. This last model is then transformed into another mapped application model to allow functional simulation in SystemC, synthesis in VHDL, etc.

The semantics of the different models are expressed at the metamodel level. This expression of a precise semantic allows to derive transformations of well defined models. The transformation rules are also defined between metamodels.

This specification environment is based on the following principles:

- Component based UML 2.0 profiles (application and architecture)
- Definition of the metamodels using the MOF
- MOF to MOF transformations
- XMI interfaces with existing tools

1.2 Model Driven Engineering

The Model Driven Architecture (MDA) is an approach to IT systems development fostered by the Object Management Group (OMG). It is based on forming a separation between the specification of a systems essential functionality as a platform independent model (PIM) and the realisation of the system using more detailed and specific platform specification (PSM). MDA is the OMG's version of more general approach to system development that is comming to be known as Model Driven Development (MDD) or Model Driven Engineering (MDE). A good overview of the state of the art in MDD can be found in Part II of [InOp9.1].

As the number of approaches to addressing MDD grows, so too are the number of tools that claim to support MDD, MDE, MDA or some other variation. Some of these tools try to be fully fledged MDD environments, and some are simply code generation tools. We feel that there are at least three categories of MDD tool:

- **Code generation tools**: These can be considered as a form of compiler; generally taking a high level system description (e.g. a UML model) and automatically generating source code for a particular language (e.g. Java, C++, etc). Many UML tools offer such capabilities, although generally they are not particularly complex (see [AHM05]). Tools such as EMF [EMF], KMFstudio [KMF] are more dedicated to specifically generating code from (UML) models defined in other tools.
- **Transformation tools**: These tools offer the capability to specify model transformations and are often (necessarily) closely coupled with a model specification tool or technique. A compiler is a transformation tool, although a tool

that is built specifically for transforming from one language to another; in this respect, a compiler generation tool can be considered as a more general model transformation tool, although limited to the associated grammar based concepts. More generic Model based transformation tools are becoming available; spurred on by the OMG's RFP on QVT [QVT]. Such tools include: MTL Engine (UMLAUT), Atlas Transformation Language (ATL), MOD-Transf, Generative Model Transformer (GMT), AndroMDA , UML Model Transformation Tool (UMT), QVT-Partners Eclipse. A review of these tools and others can be found in [InOp7.3]

- **Environments for MDD**: The separation point between these tools and more basic Transformation tools is debatable; however, an MDD environment should provide means to specify models and transformations, and to have facilities to manipulating instances of those models and transformations as first class entities. In general it could be expected that these tools can be tailored or used as the basis to provide an application based on MDD for a particular purpose. We are unsure if any tool currently meets these requirements, although a likely candidate is XMF-Mosaic (Xactium, www.xactium.com).

2.1.2 Design Flow

SoC design covers a lot of different viewpoints including as much the application modeling by the aggregation of functional components, as the assembly of existing physical components, as the verification and the simulation of the modeled system, as the synthesis of a complete end-product integrated into a single chip. As a rule a SoC includes programmable processors, memory units (data/instruction), interconnection mechanisms and hardware functional units (Digital Signal Processors, application specific circuits). These components can be generated for a particular application; they can also be obtained from IP (Intellectual Property) providers. The ability to re-use software or hardware components is without any doubt a major asset for a codesign system.

The MDA contributes to express the model transformations which correspond to successive refinements between the abstraction levels. Metamodeling brings a set of tools which will enable us to specify our application and hardware architecture models using UML tools, to reuse functional and physical IPs, to ensure refinements between abstraction levels via mapping rules, to ensure interoperability between the different abstraction levels used in a same codesign, and to ensure the opening to other tools, like verification tools, through the use of standards.

All the previously defined models, application, architecture and association, are platform independent. No component is associated with an execution, simulation or synthesis technology.

Once all the components are associated with some technology, the deployment is realized. The diversity of the technologies requires interoperability between abstraction levels and simulation/execution languages.

From each of the resulting models we could automatically produce the execution/simulation code and the interoperability infrastructure by defining mapping rules.

The simulation results can lead to a refinement of the application, the hardware architecture, the association or the deployment models.

2.1.3 UML Profile for Real-time and Embedded systems

Among the many kinds of systems targeted at by UML, real-time and embedded systems have been the object of different proposals. A new request for proposals (RFP) has been issued, named MARTE (Modelling and Analysis of Real-Time and Embedded Systems), intended to update the previous UML profile for Schedulability, Peformance and Time (SPT) to take into account two major new developments within the OMG: the UML profile for Quality of Service (QoS) and Fault Tolerance, and the new version of UML: UML2.

This profile for Real-Time and Embedded Systems covers (in a non-exclusive way):
- Embedded systems, seen as interconnected devices containing software components as well as hardware, electronic, mechanical or other components. They are deployed into the physical environment, and are typically resource limited: hence the balance of functionality implementation and resource optimization is essential. They are naturally heterogeneous, distributed, reactive, safety-critical, and constrained in time and consumption.
- Reactive systems, continuously accepting inputs from their environment, which they are meant to control, or interact with, through the production of outputs. Their behavior is essentially cyclical: acquiring inputs (e.g. from sensors), computing reaction (possibly involving heavy data computation, and possibly depending on current local or global state, defining the current mode of action), and emitting the response (e.g., through actuators).
- Control applications implement control laws by issuing commands to actuators, in function of the sensor information, so that a convergence is obtained in the controlled process towards the required values. Regulation or servoing can be done in open-loop (using just the request as input) or in closed-loop (sensing of intermediate effects is taken into account in the regulation).

- Intensive data-flow computation is mainly encountered in signal processing (e.g., image and sound), and can benefit of parallelism for processing, reception, decoding, filtering and the like. Classically, a phase of systematic signal processing (regular computations) is followed by intensive data processing (irregular computations depending on values).

The definition of a UML profile for the modelling of these characteristics aims at their exploitation by specialized tools to facilitate the simulation, analysis and checking, optimization and implementation of embedded systems. The formalization within one common standard profile is also meant to improve communication in multi-expertise design teams inherent to the domain, involving systems, hardware and software engineers.

The integration of the software and hardware aspects into the same profile enables for the design of transformations based on them, so that an implementation-independent specification of an application can be combined with a description of the architecture in order for automated transformations to derive the implementation. The MDA approach related to UML is typical in that sense, and can serve as a definition framework for the Y-Model.

The temporal (behavioral) models of RTES considered cover different aspects:
- asynchronous/causal models are concerned with ordering of activities;
- synchronous/clocked models add the notion of simultaneity, as a discrete set of instants; this logical time is used in Simulink/Stateflow, synchronous languages, Statecharts, as well as hardware description languages like VHDL or SystemC;

- real/continuous time, taking durations into account, in the perspective of analysis and for scheduling.

1.3 New features of UML2.0 to express data flow and control flow

2.1.4 Integrating control in the MDE design flow

A well-advanced ongoing work at INRIA in the DaRT team is the definition of UML2.0 features to represent specifications of data intensive parallel systems, especially from the point of view of repetitions [CDMB05]. Another aspect is being studied, in a complementary way, concerning the control part of applications. In contrast to the regular part of e.g., signal processing applications [LDB05] [LDBR05], the control part involves reacting to events coming from the environment, or from the computations themselves, and switching between configurations, changing execution modes, going dynamically from state to state according to a specified behavior.

We are proposing control mechanisms in the framework of ArrayOL and Gaspard, and consider integrating them in the UML2.0 profile. This involves :

- giving semantics to activity states, starting, stopping, changing modes in the hierarchical and elementary tasks involved in parallel computations;

- integrating control issues in the existing constructs, and, according to need, proposing new language constructs to specify them;

- designing transformations to integrate them in the Y model, in order to extract the model of the control part from an application, to transfer it to analysis and verification tools, and to make efficient use of the control information along the transformations from applications, through association, to simulation and implementations.

2.1.5 Synchronous model of control

Our approach is to make our proposal for control mechanisms in terms of reactive systems, which are a way of modelling state and event based systems with :

- a clear and firm formal semantics, allowing for inambiguous specifications, and for well-founded analysis and transformation operations;

- a whole technology of design environments, structured specification languages, compilers with diverse optimizations, analysis and verification tools, and code generation tools targeting diverse execution platforms.

In particular, we are interested in the synchronous approach to reactive systems design (http://www.synalp.org) [NH93,NH98]. It gave birth to complete programming environments, around languages like Argos, Lustre(http://www-verimag.imag.fr/SYNCHRONE), Esterel (http://www.inria.fr/recherche/equipes/aoste.en.html), Signal/ Polychrony (http://www.irisa.fr/espresso/Polychrony), Syndex (http://www-rocq.inria.fr/syndex), Lucid Synchrone(http://www-spi.lip6.fr/lucid-synchrone) or Mode Automata(http://www-verimag.imag.fr/PEOPLE/Florence.Maraninchi/MATOU). This approach is characterized by the fact that it considers cyclic systems whose global steps can, by synchronous composition, encompass a set of events (known as simultaneous) on the resulting transition.

A relationship is being made between synchronous languages and models and UML2.0, [RdS-CA03], in particular in the context of the PROTES project, and the reply to the MARTE profile RFC. Our work on integrating control will be made in coherence with this work.

2.1.6 Implementing Associations

One primary task within the process of providing MDD based tools is the production and implementation of models or meta-models (models of models, or the model of a modelling language). For instance, to produce our ModEasy toolkit we require meta-models for a design language, for a verification language, and for any of the implementation frameworks we intend to map the designs to.

Within the current work, the design and verification support tools are provided within the Eclipse IDE [IBM] which requires us to implement our models using the Java programming language [Sun]. The Java programming language is not (and was never intended to be) a modelling or meta-modelling language. The Object Management Group's (OMG) [OMG] Meta Object Facility (MOF) [MOF2] has, however, been designed specifically as a set of concepts with which to define (or model) other languages. Therefore, in order to best support our needs to define meta-models of a system design language, verification and implementation frameworks, we chose to use the MOF as the language for defining those meta-models. Of course this then leads us to the issue of how to implement models in Java that are defined using MOF. There are a couple of choices regarding a general implementation approach,

- to implement the models directly in Java (using an approach such as EMF [EMF]),
- to implement the models as instances of a meta-model repository (using an approach such as MDR [M03]).

Both of these choices have their advantages and drawbacks. Use of a meta-model repository provides a quick way to provide support for your model. One need only provide a specification of the model (possibly using XMI [XMI2]) and the meta-model repository can be used as a repository for instances of your model. Another advantage is that the meta-repository should easily provide support for serialising and un-serialising instances of your model in a standard format (again potentially XMI). The drawbacks of this approach are

efficiency issues introduced by the meta-level objects that represent each of your model objects, and the consequential reliance on the (probably 3rd party) meta-model repository itself. This later issue is partially resolved by the standardization of the Java Metadata Interface (JMI) [JMI] that forms part of the libraries issued with Java 5, which provides a standard interface for MOF repositories, i.e. JMI (version 1.0) defines a Java mapping for the MOF (version 1.4).

The direct implementation approach requires a more effort up front (i.e. specifying the mappings from modelling concepts to programming concepts, as addressed in this paper [AHM05]) and the mapping from model instance to serialisation format is not quite as simple (although it is possible). The advantages of the direct approach are that there is no reliance on a 3rd party library, and if you have control of the code generation templates you can have complete control of the implementation patterns and hence some control over the efficiency choices.

Even though it is not desirable practice and a good round trip engineering tool should reduce the problem, it is still the case that code implementations and design models can easily become separated. Considering this, it is quite important that any code that is automatically generated from a model be produced in a way that is easy to understand. Thus we have two main goals for code generation from models:

1. Readable code.
2. No use of bespoke or 3rd party libraries.

Even though the modelling community has been using MOF and UML [UML2.0] based languages for several years, and there are a number of tools that generate Java code from UML models, there are still some significant issues regarding how to implement some of the concepts (especially due to the introduction of some new ones with the advance to UML 2.0). In particular, one of the major abstractions used in Object Oriented (OO) modelling languages such as MOF and UML is the concept of an Association. This concept does not exist in OO programming languages such as Java (or any of the other main stream OO languages). Consequently, in order to generate code from a UML or MOF model it is necessary to devise a mapping from Associations on to the chosen OO programming language.

The concept of an association is a complex abstraction facilitating a variety of specification variations all of which affect the semantics of the association. There are many modelling tools that claim to support code generation from UML models; however none of these tools address the generation of code for the complete set of possible variations to an association specification. In particular, some tools do not correctly implement the bi-directional semantics of an association and most do not address generation from the more complex qualified and n-ary associations. Additionally, the new UML 2.0 set of standards has been recently released including some new concepts such as redefinition and subsetting which of course the tools do not yet support. Supporting code generation from UML to Java is made easier for some of these new features by the changes to the Java programming language introduced with Java 5.

The paper [AHM05] focuses on providing solutions to the issues of mapping qualified associations and the new (UML 2.0) semantic variations of an association into the Java 5 programming language.

3 Verification techniques

1.4 Verification of state / Activity / Sequence diagram in UML2.0

UML State, Activity and Sequence Diagrams are diagrammatic languages for specifying system behaviour. To achieve the specification of a reliable system it is important to be able to verify the behaviour of the system; hence, we are investigating verification techniques for the behaviour specification languages of UML.

State diagrams are based on the language of Statecharts [H87] and there is a lot of material addressing the verification of state based behaviour specifications [AD94] and on the verification of Statecharts and their UML variant [DMY02, GHK00].

In earlier versions of UML, Activity diagrams are defined as a special case of State Diagrams, thus the verification of activity diagrams could be achieved in the same manner as their base. However, later versions of UML base the semantics of activity diagram more closely to that of Petri-nets, thus requiring us to look for separate techniques for their verification.

[GMP02] Discusses how Model Checking can be used in the context of UML Designs. Suggesting the use of Sequence diagrams to represent required of non-required behaviours, checked against a State Diagram description of the system behaviour.

[EW04] Illustrate a tool and technique for verifying workflow models described using activity diagrams. The technique maps the activity diagrams to a model checker, thus facilitating arbitrary propositional requirements to be checked against the workflow specification. [GL03] discus and move towards a technique for mapping a complete UML specification into Petri-nets for the purpose of verification; the technique uses meta-modelling and graph grammars.

[BCG04] proposes a methodology for verification of UML diagrams based on graphs and graph transformations and a temporal graph logic, enabling certain behavioural properties to be automatically checked.

Verifying Sequence Diagrams is not quite as simple, mainly because a sequence diagram usually is a partial specification, illustrating a single behavioural path. Sequence diagrams are based on the language of Message Sequence Charts (MCS) and later versions of UML incorporate more of the MSC semantics.

[BL01] suggest an approach involving Lamport diagrams, a linear time temporal logic and the construction of a Büchi automaton. In [AY99] the authors present a technique for model checking MCSs and Hierarchical MSC-graphs by construction of an automaton

There are a number of techniques proposed in the literature, but of primary importance is the level of completeness of the information provided by the diagrams. A Sequence diagram in particular is a partial specification of behaviour, thus verification of a partial specification is

required!

1.5 Verification and FPGAs

3.1.1 Survey of related formal verification approaches

Formal Verification in the context of FPGA design flow often refers to verifying that a final implementation is functionally equivalent to the higher level RTL design. This is more specifically referred to as Equivalence Checking. Model Checking is an alternative form of formal verification, which exhaustively proves a specific property about a specification.

Commercial FPGA support tools tend to support Equivalence checking rather than Model Checking. Model Checking is still a research exercise, although many useful tools are available, there is no standard property specification language and it suffers from state space explosion problems. However, it is a very useful technique and has much potential.

There are a variety of different approaches to verification, summarised by the following bullets taken from [K05]

- By construction - property is inherent.
- By verification - property is provable.
- By simulation - check behavior for all inputs.
- By intuition - property is true. I just know it is.
- By assertion - property is true. Wanna make something of it?
- By intimidation - Don't even try to doubt whether it is true

It is generally better to be higher in this list.

Within the context of formal verification we are generally looking at the second of these points, techniques for proving certain properties. There is a variety of formal specification languages that can/have be/been used to address the verification of FPGAs: Timed Automata, Petri-Nets (with timed extensions) and Timed Lotos are three specific languages that have been used in a number of publications, discussed in more detail below.

3.1.2 Timed Automata

Verification by Timed Automata is often referred to as Model Checking.

"Model checking is emerging as a practical tool for automated debugging of complex reactive systems such as embedded controllers and network protocols. In model checking a high level description of a system is compared against a logical correctness requirement to discover inconsistencies. Traditional techniques for model checking do not admit an explicit modeling of time and are thus unsuitable for analysis of real time systems whose correctness depends on relative magnitudes of different delays. Consequently, timed automata [AD94] were introduced as a formal notation to model the behavior of real time systems. Its definition provides a simple and yet general way to annotate state transition graphs with timing constraints using finitely many real valued clock variables. Automated

analysis of timed automata relies on the construction of a finite quotient of the infinite space of clock valuations. Over the years, the formalism has been extensively studied leading to many results establishing connections to circuits and logic, and much progress has been made in developing verification algorithms, heuristics, and tools." [Alur99]

Timed Automata are widely used as a means to analyse Real Time systems. [MP95] propose a technique for modelling asynchronous components using TA. [CYD98] use TA to perform timing analysis of asynchronous circuits illustrating their approach with an example of an asynchronous differential equation solver; they propose use of an approximating algorithm [CD97] that uses time separation of events giving much faster analysis results than a standard algorithm. [TKYBS98] use TA to model gate delays and investigate wire cross-talk. [CLY04] model real time asynchronous circuits using a Parameterised variation of TA [].

One of the issues with model checking is the state space explosion inherent in large systems. From a design perspective this can be improved by using hierarchical state machines. However, to verify properties they are traditionally flattened. There is some work [AY01] and[BGR01] that addresses verification of hierarchical SM directly but there are not as yet any tools available that implement the ideas.

There are a number of tools available that implement Timed Automata based model checking algorithms: [Uppaal] [Kronos] [HyTech] are three such tools. [AFMPY02] describes a tool based on Uppaal that is aimed specifically at schedulability analysis of embedded real-time systems.

3.1.3 Other Techniques for Formal Analysis

Another formal language often used as a basis for formal analysis is the language of Petri Nets. There is a lot of work at the ESLAB (http://www.ida.liu.se/labs/eslab/publications), However, to perform model checking on the Petri-net models, they are transformed into Timed Automata. [AEP03] [AEP02] [A01] [AEP01] [AEP00a] [AEP00b] give detail of their work.

[NHT96] and [HT99] look at using variation of LOTOS as a means to verify digital logic. However, the work of [HT99] again requires transformation to Timed Automata and [HT99] proposes use of Timed Automata as a means to perform the model checking.

3.2 Synchronous techniques in UML2.0

3.2.1 Integrating verification in the Y model

The Y design framework involves a set of transformations, going through different levels of abstraction and different views of the designs. In this context, different verification techniques can be integrated at different levels and for different purposes.

- Transformations can be checked for their preservation of properties to be kept holding across the levels in the MDA/E;
- Designs can be checked for application specific properties, or the association with an

architecture can be checked for more implementation-specific properties. This can be done at different levels of refinement between specification and implementation.

We are particularly interested in behavioral properties of the control part of an application. They can be analyzed by model-checking tools, with or without quantitative time. An alternative possibility is to consider, for the same class of automata-based models, to use more constructive techniques like discrete controller synthesis.

Technically, this involves extracting a model of the control, in terms of labeled transition systems : this can be done by a transformation from the application model towards a synchronous automaton model. Such models can then be used by the synchronous toolset, functionalities of which are sketched below.

3.2.2 Synchronous tools

The synchronous approach proposes formal methods in a very integrated way. Generally speaking, formal methods are often used for analysis and verification; they are much less often integrated in the compilation or generation of executives (in the sense of executables of tasks combined with the host real-time operating system). They are notoriously difficult to use by end-users, who are usually specialists in the application domain, not in formal techniques. This is why encapsulating formal techniques in an automated framework can dramatically improve their diffusion, acceptance, and hence impact.

Synchronous techniques do this by covering a complete range of design support :

- compilers, which are providing for:
  - analysis of the coherence of specification w.r.t. to synchronizations,
  - transformations from more or less declarative specifications to more operational or implementation-oriented forms,
  - generation of code towards different targets, software (C, VHDL) or hardware (FPGA).
- analysis and verification tools
  - simulation,
  - model-cheking,
  - discrete controller synthesis,
  - profiling/performance analysis, ...

4 Simulation and synthesis of FPGA

1.6 Observation of IP Interface standard development (Spirit project)

1.7 Investigation into SystemC to VHDL transformation tools

SystemC is a worldwide standard for modeling hardware and software systems using the C/C++ language with a library for hardware constructs.

For those who decide to use SystemC as an implementation language, the biggest issue appears to be the lack of a real offer of synthesizers taking SystemC as an entry.

There are a number of various SystemC to VHDL transformation tools existing on the market. Thus, SystemC design flow can be seamlessly linked to the backend flow in order to take advantage of existing HDL synthesizers.

The most important features of the synthesizable SystemC subset are the following:

- Hierarchical modules
- Loops (for, while)
- Template SystemC modules
- Member functions
- Conditional (if, switch)
- Enumeration
- Static type conversion
- Constructor functions
- Array (ports, signals, member variables, sub-modules)

Some of the most popular transformation tools are described next.

Cynthesizer (Forte):

Cynthesizer first translates behavioural SystemC to RTL SystemC, and then it translates RTL SystemC to VHDL/Verilog RTL for logic synthesis. Beyond this, it automatically creates multiple RTL implementations and optimizes them according to designer-specified constraints. Forte claims that their quality of results exceeds hand-coded design, specifically with timing and area optimization.

Prosilog SystemC Compiler:

This compiler offers the industry's largest synthesizable SystemC subset. The code generated by Prosilog is fully commented and accessible. It is totally integrated in the Microsoft Visual Studio C++ IDE.

CoCentric SystemC Compiler:

It offers a fast and high quality path from a system level hardware description coded in SystemC to gates or synthesizable RTL description.

1.8 Survey of SystemC and VHDL design flow for embedded system development with FPGA

There are two possible SystemC design flows: either a single language flow from SystemC to the logic gates implementation or a mixed language flow from SystemC down to RTL SystemC and then RTL SystemC is translated to the equivalent model into VHDL or Verilog.

The mixed language design flow is less time consuming, more efficient and more reliable.

The first advantage of this methodology is that the design process language remains the same from behavioural to RTL level, thus refinement down to RTL SystemC becomes easier and consequently the errors due to language translation can be avoided.

Another advantage is brought by the SystemC to HDL translator offering the possibility to automatically translate the RTL SystemC code into VHDL language. Thus, the SystemC code remains the only one that must be maintained.

For the verification steps, the present design flow allows to reuse the same test bench for the behavioural model, the SystemC RTL model and the translated HDL model (using SystemC-VHDL cosimulator).

In order to start the RTL refinement phase, a compiler can be launched on the behavioural model to identify which of its code elements are not synthesizable and to modify it to obtain a synthesizable model.

When the synthesizable model is available, it is necessary to verify that its functional behaviour is the same as the one of the behavioural model. To do this a verification project has to be carried out to compare two models.

After the RTL SystemC model has been approved it can be translated into VHDL language, which can then be used for the prototype synthesis on FPGA.

FPGA Design flow can be described as follows:

- Digital designs are entered as schematics or VHDL code. A design can be described using just one or a mixture of these methods.
- A functional simulator checks the operation of the compiled design and inspects errors.
- Implementation tools compile the list of gates and connections, or netlist, into a bitstream which is used to program an FPGA. For some devices, the implementation requires mapping the circuit to the FPGA architecture, placing the gates in specific CLBs, and then routing the wires using the programmable switch matrices.
- A timing simulation of the design can be run after implementation tools have determined the gate and routing delays associated with a particular mapping to an FPGA architecture.

- The bitstream is downloaded into the device.
- Debugging is performed by forcing inputs into the circuit board and observing the response.

5 Applications

1.9 GPS navigator application

The GPS navigator application concerns an automotive system of Intelligent Cruise Control, which aim is to help control the car speed depending on the limitations, which in turn depend on location of the car, given by a GPS.

It operates in different modes, where it can be informing the driver by emitting an alarm when the speed is exceeding limitation, or limiting the speed by forbidding to exceed the limitation (unless the driver really needs to, by kicking down the accelerator), or cruising in automatic mode at a given speed (which is deactivated by the brake). Special modes involve the GPS and a map to provide the limit speed or the cruising speed according to the limitations on the road where the car is located.

This system combines both control and data processing. The purpose of the case study is to illustrate and validate our methodological approach regarding separation of control and data aspects in the specification, as well as in the verification of the system.

In this project, a complete GPS navigator system will be used in order to inform the driver about the permitted speed in the concerned area. This system will alert the driver if the vehicle speed is above the speed limit on the road. Many research projects have been focused on this kind of navigator (LAVIA project, ISA project, INFATI project …). However, the ModEasy project still be different from all these projects especially concerning the following principles :

- The classification of the detected area located by the GPS.
- The association with the anti-collision radar (speed will depend on the limited speed and also the security speed indicated by the radar).
- A global UML model for the whole navigation system
- A Real time FPGA unit integrating the GPS and the radar decisions.

The study has analysed the existing commercial GPS system. This type of system will be completed by a real time algorithm which indicates the limit speed on the detected area. Then an original partition of the area into zones into limit has been proposed. To increase the security, the system to realise will take account also of the limit speed on the following area. The change of the speed will be done progressively and avoid disagreement to the driver.

The future works concern the implementation of the GPS navigator algorithm and the first test in real conditions. The next studies will be oriented to the radar and GPS information association.

1.10 Anti-collision radar specifications

The principle of a conventional RADAR system is to send a modulated microwave towards possible targets. If a target is within the range of the radar, it reflects a part of the transmitted wave to the receiver. Using the corresponding delay, we can compute the distance between the target and the radar. As several collision warning radar systems will come on the automotive market, and as some manufacturers plans to propose radar systems on board, researches are still carried out for a radar system that could avoid most longitudinal collision [RDAR99].

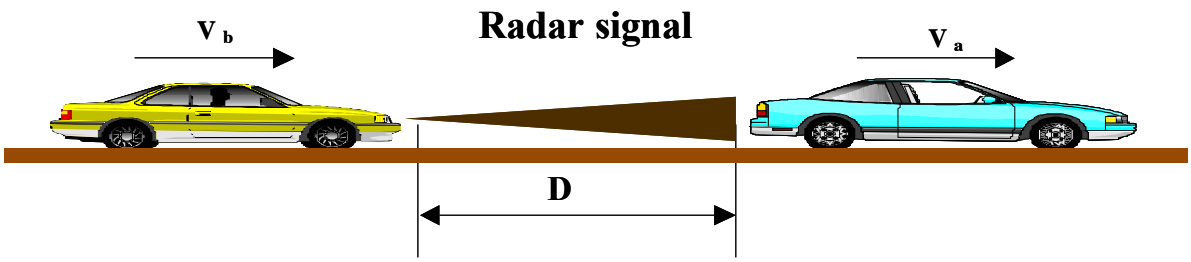


Figure 5.1 : The collision avoidance radar principle

The obtained distance is used to compute the safety distance as described by the following :

$$D_s = \frac{V_b^2}{2\ _b} - \frac{V_a^2}{2\ _a} + V_b T_r + M$$

Many studies have been focused for a long time on road collision avoidance RADAR system. The classical radars are the FMCW (Frequency Modulated Continuous Wave) radar and the pulsed radar. The first type is mainly used for road applications but is enable to detect large ranges due to the limitation of the transmitted power. The second type of radar is very simple to implement and send periodically a short duration pulse. However this radar couldn't detect small range. In fact, during the pulse transmission the radar is blind.

In the OAE department of the IEMN, many systems based on these two radars have been proposed. Then interesting tests have been realized using the corresponding mock-up in real conditions and with the INRETS-LEOST collaboration. The FMCW radar did not detect ranges upper then 60 m. The pulsed radar is enable to detect ranges under 30 m. Finally a bimodal radar has been implemented by the association of the two radars at the same system. An adapted decision unit gives the pertinent distances values between these calculated by the two sensors [RDAR99]. The covered range interval is very interesting and allows a perfect security.

However the proposed system is very encumbering. Then, in other approach we suggest an original kind of signal coding and processing in order to meet the sensor requirement of low emitting power with high detection and high range covering with good resolution.

The originality of the method resides in the use of numerical coding waveforms based on pseudo-random sequences (PRS). The receiver could use correlation receiver or other advanced detection algorithms [DM99], [FRDH00] and [RAHRB98].

The first radar mock-up uses a numerical correlation receiver. This receiver is studied in terms of Signal to Noise Ratio (SNR) enhancement. Thanks to a Digital Signal Processor (DSP), a numerical correlation receiver has been implemented on a pulsed microwave radar prototype emitting pseudo random sequences.

This receiver is known to be an optimal receiver because its Signal to Noise Ratio gain is the maximum value that could be reached in a receiver. Correlation or matched filter are widely used in communication and positioning systems (GPS) since they have the ability to detect coded signals buried in noise and to reveal a shift in phase between two signals. Therefore, it is a good idea to use PRS and a correlation receiver in a radar front end, since the radar aim is to detect the time shift between an emitted signal and its echo, in order to deduce the distance to the obstacle who generated it.

The new proposed system is called correlation radar. The principle of this radar system is to generate a random pseudo code (a signal which seems random if it is observed over a short duration, but which is repeated identically at each period) [RABH98]. This pseudo-random code modulates a microwave carrier and is transmitted towards the potential targets. After reflection on an obstacle, the return wave is demodulated and processed. The treatment consists in calculating the correlation between the emitted code and the signal it receives. The maximum of this correlation is detected, and the corresponding time position is noted n. Knowing the speed of the wave (3.10^8 m/s), the distance from the target can be deduced using the formula:

$D=n.c/2f$ where D is the distance to the target and f the RADAR code frequency.

This peak is the equivalent of the impulsion in the classic radar system.

The constant emission of the code as in continuous wave radar systems enables to have a lower peak power signal at the radar front end for a given average power. Our correlation receiver performs cyclic correlation between the received samples and the emitted ones (Figure 5.1). Samples are taken for one full period of the signal, one sample per bit, hence covering a full PRS period of the emitted code.

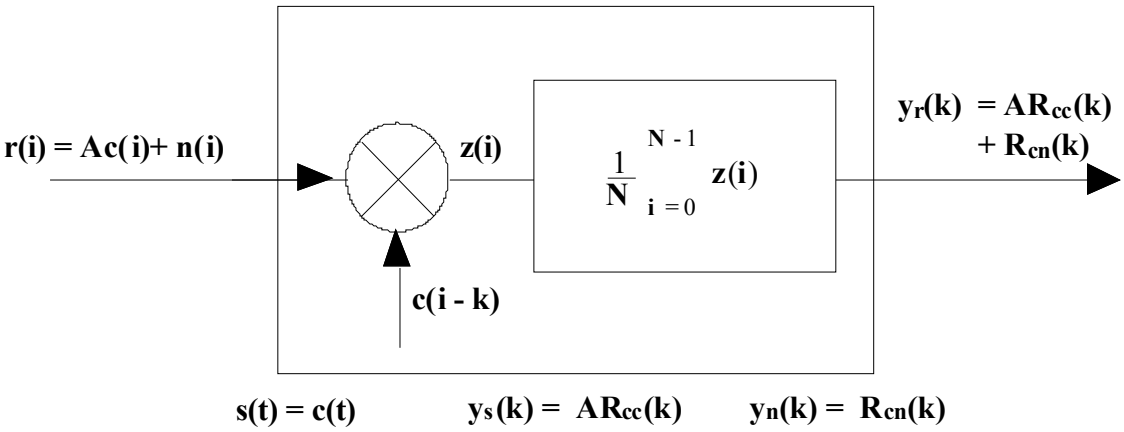


Figure 5-3-1 : principle of the numerical correlation receiver

The discrete outputs y_r of the correlation receiver are given by equation (1), in which we add each product of the received sample r(i) by the reference sample c(i) cyclically shifted.

$$y_r(k) \quad R_{rc}(k) \quad \frac{1}{N} \sum_{i \quad 0}^{N \quad 1} r(i)c(i \quad k) \text{ for k=0,1,…,N-1} \qquad (1)$$

The difference is modulo N, N being the number of samples of both signals. Cyclic Correlation can then be performed numerically the same way as a FIR filter is implemented. Because the system is linear, $R_{rc}(k)$ can be split as:

$$y_r(k) \quad R_{rc}(k) \quad AR_{cc}(k) \quad R_{cn}(k) \qquad (2)$$

in which R_{cc} is the autocorrelation of the reference signal c(i) and R_{cn} is the cross correlation between the noise and the reference signal. If the received signal is the emitted one attenuated and delayed in time with added noise, the receiver output will likely be the autocorrelation function of the coding sequence shifted accordingly.

For a radar application, the coding sequence should be chosen with a maximum value of the autocorrelation function at the origin and with weak sidelobes. We also have to take into account the fact that the sequences should be orthogonal (i. e. with null intercorrelation between them) to avoid interferences between different coding sequences. The family of PRS code was chosen because it satisfies the above requirements. Its autocorrelation function is presented on Figure 2 for a length N=127 bits bipolar code taking –A and A values. The result is equal to A at the origin and -A/N elsewhere, which gives us a 42 dB dynamic.

Because the system is linear, if several echoes are received, the output of the receiver will show several peaks accordingly, which permits us to detect more than one obstacle simultaneously, each distance from them given by the delay of the peak.

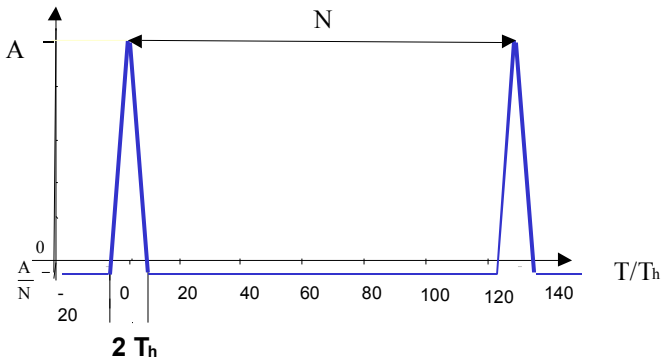


Figure 5-2 : autocorrelation function of a 127 bit PRS

The receiver presented on figure 1 has been studied in terms of Signal to Noise Ratio (SNR) enhancement [RDAR99]. We can notice that the receiver's SNR gain only depends on the length of the PRS used. The longer the code is, the better the SNR gain is, but also the heavier the computation level is.

5.1.1 Experimental Radar Prototype

We have implemented a PRS generator and a correlation based receiver on a 76_77 GHz pulsed microwave radar prototype (Fig 3). The length of the PRS is N=1023 clocked at 100 MHz. Modulation of a 76 GHz carrier is performed with the PRS, and this signal is emitted continuously.

At reception, demodulation is performed on the reflected signal. The baseband signal is then treated by the FPGA based numerical correlation board. The signal is sampled at a 100 MHz clock rate in order to get a full period (1023 samples) of the received signal.

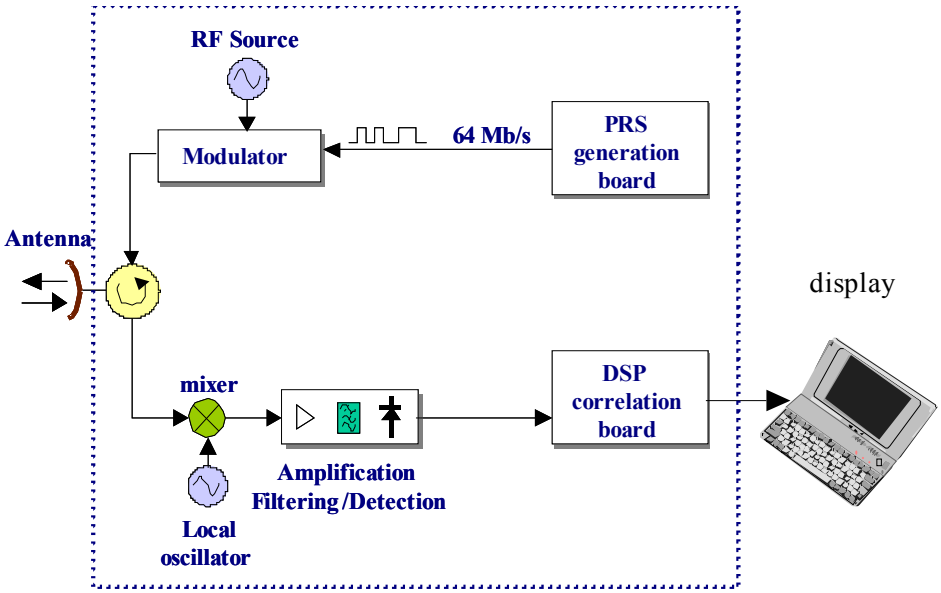


Figure 5-3 : The radar system bloc-diagram

The correlation should be computed by the FPGA between these samples and the reference signal coefficients (emitted PRS) stored in its program memory. The result is sent on a computer via a serial link. The result is displayed in real time through a window-based program. The computation time for the correlation is about 1 millisecond, which permits us to have a real time detection of echoes. Every result exceeding a given threshold is considered as a detected obstacle. Distances are easily computed, since a peak give the delay T between received and transmitted signal.

Several tests have been carried out on stationary targets with the prototype equipped with a narrow beam parabolic antenna. Since the SNR enhancement of the receiver is about 42 dB, we achieved the detection of weak echo even if they were buried below the ambient noise [DLGR05].

Figure 5.3 shows the correlation receiver output for an obstacle placed at 32 meters from the radar. The first 64 distance gates are not represented because they cover the range from 150 to 300 meters. The results are promising and in good accordance with the expected ones. We achieved unambiguous detection of a square meter panel, with a good SNR.

Thanks to the recent progress in speed of digital signal processor technologies, we can achieve a real time detection of obstacles with a resolution of almost 2 meters.

5.1.2 Conclusion

This concept of correlation receiver based RADAR is thus interesting. It makes possible to measure significant distances, between road vehicles, using only a low power. We will continue to work on this prototype in order to improve its performances of detection. For example, we can increase the frequency of the base band signal in order to increase the precision of the system.

The correlation receiver performs good detection only in AWGN (additive white gaussian noise) channel assumptions. In other conditions (non gaussian and non stationary noise), the only correlation receiver seems to be enable to detect noisy signal [RDER01] and [RRDE04]. Thus we have to use other detectors using new signal processing tools, such as wavelets or **higher order statistics**, or to test new electronic technologies like FPGA (Field Programmable Gate Array), in order to improve the integration and to reduce the calculation time of the system.

PRESENTATION OF THE CODE FAMILIES AND THEIR CORRELATION PROPERTIES

An interesting study have been realized to give comparison between the possible codes families.

The concerning codes are presented in this section. Table 1 introduces their characteristics. This table shows their numbers, their length and the maximum of their crosscorrelation.

These sequences look like noise and so have a spread spectrum. The selected codes have the same length: 2^n-1 [RDER01]. The pseudo-random sequences have two correlation levels: an autocorrelation peak and a low level only for the crosscorrelation.

The families are: the MLFSR (Maximum Length Feedback Shift Registers) [DLGR05], [RRDE04] used as a reference, the Gold codes and the Kasami codes that are available in a very large amount.

Family	Period	n	Size of family	Maximum autocorrelation	Maximum crosscorrelation
Gold	2^n-1	4^m+2, 2^m+1	2^n+1	2^n-1	$2^{(n+1)/2}$+1
Kasami (large set)	2^n-1	4^m+2	$2^{n/2}(2^n+1)$	2^n-1	$2^{(n+2)/2}$+1
MLFSR	2^n-1	2^m, 2^m+1	function of length	2^n-1	$2^{n/2}$+1

Table 1: *Codes families and their correlation properties.*

The Kasami codes exhibit the same performances as the other codes family. They behave as the Gold codes, the better detection probability obtained with the largest user's amount. As there are 500 more Kasami codes than MLFSR they are the solution we will use in this system. During all our study we called the used codes PRS (Pseudo-Random-Sequences).

Appendix

1.11 Literature

[A01] Luis Alejandro Cortes, "A Petri Net based Modeling and Verification Technique for Real-Time Embedded Systems", Licentiate Thesis No. 919, Dept. of Computer and Information Science, Linköping University, Dec. 2001.

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science.* 126:183-235. 1994

[AEP00a] Luis Alejandro Cortes, Petru Eles, Zebo Peng, "Verification of Embedded Systems using a Petri Net based Representation", 13th International Symposium on System Synthesis (ISSS 2000), Madrid, Spain, Sept. 20-22, 2000, pp. 149-155.

[AEP00b] Luis Alejandro Cortes, Petru Eles, Zebo Peng, "Formal Coverification of Embedded Systems using Model Checking", 26th Euromicro Conference (Digital Systems Design), Maastricht, The Netherlands, Sept. 5-7, 2000, vol. I, pp. 106-113.

[AEP01] Luis Alejandro Cortes, Petru Eles, Zebo Peng, "Hierarchical Modeling and Verification of Embedded Systems", Euromicro Symposium on Digital Systems Design, Warsaw, Poland, Sept. 4-6, 2001, pp. 63-70.

[AEP02] Luis Alejandro Cortes, Petru Eles, Zebo Peng, "An Approach to Reducing Verification Complexity of Real-Time Embedded Systems", 14th Euromicro Conference on Real-Time Systems (ECRTS 2002), Work-in-Progress Session, Vienna, Austria, June 19-21, 2002, pp. 45-48.

[AEP03] Luis Alejandro Cortes, Petru Eles, Zebo Peng, "Modeling and Formal Verification of Embedded Systems based on a Petri Net Representation", Journal of Systems Architecture (JSA), Special Issue on System and Circuit Synthesis and Verification, vol. 49, no. 12-15, December 2003, pp. 571-598.

[AFMPY02] Tobias Amnell, Elena Fersman, Leonid Mokrushin, Paul Pettersson, and Wang Yi. Times - A Tool for Modelling and Implementation of Embedded Systems. In proceedings of 8th International Conference, TACAS 2002, part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002 (Grenoble, France, April 8-12, 2002), pages 460-464, Springer-Verlag, 2002. Lecture Notes in Computer Science, Vol.2280

[AHM05] D.H.Akehurst, W.G.J.Howells, K.D.McDonald-Maier. Implementing Associations. Submitted to Software and Systems Modelling Journal. 2005

[Alur99] R. Alur. **Timed Automata.** *NATO-ASI 1998 Summer School on Verification of Digital and Hybrid Systems.* A revised and shorter version appears in *11th International Conference on Computer-Aided Verification*, LNCS 1633, pp. 8-

22, Springer-Verlag, 1999.

[AY01] Rajeev Alur, Mihalis Yannakakis. Model checking of hierarchical state machines. ACM Transactions on Programming Languages and Systems (TOPLAS). volume 23 , issue 3. ACM Press New York, NY, USA. ISSN:0164-0925. pp. 273 - 303. May 2001

[AY99] R. Alur and M. Yannakakis. Model checking of message sequence charts, Proceedings of the Tenth International Conference on Concurrency Theory, LNCS 1664, Springer, pp. 114--129, 1999.

[BCG04] P.Baldan, A. Corrandini, F. Gadducci. Specifying and Verifying UML Activity Diagrams via Graph Transofrmation. In proceedings Global Computing, Roverto. Springer, LNCS, Vol. 3267, pp18-33, 2004.

[BGR01] Michael Benedikt, Patrice Godefroid, Thomas W. Reps. Model Checking of Unrestricted Hierarchical State Machines. Proceedings of the 28th International Colloquium on Automata, Languages and Programming. Lecture Notes In Computer Science; Vol. 2076 Springer-Verlag, London, UK. ISBN:3-540-42287-0. pp.652 - 666. 2001

[BL01] B.Bollig, M.Leuker. Modelling, Specifying, and Verifying Message Passing Systems. Eighth International Symposium on Temoral Representation and Reasoning (TIME'01) p0240, 2001

[Boa01] OMG.A.Board. Model Driven Architecture (MDA), Technical report, OMG, 2001, n° ormsc/2001-07-01.

[CD97] S. Chakraborty and D. L. Dill. Approximate algorithms for time separations of events. In Proc. of ICCAD, 1997.

[CDMB05] Arnaud Cuccuru, Jean-Luc Dekeyser, Philippe Marquet, and Pierre Boulet. Towards UML 2 extensions for compact modeling of regular complex topologies - A partial answer to the MARTE RFP. In MoDELS/UML 2005, ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems <http://www.modelsconference.org/>, page 15 pages, Montego Bay, Jamaica, October 2005.

[CLY04] C. Chen, T. Lin, and H. Yen, Modelling and Analysis of Asynchronous Circuits and Timing Diagrams Using Parametric Timed Automata, in Proc. of the 23rd IASTED Int'l Conf. on Modelling, Identification and Control (MIC 2004), February 23-25, 2004 Grindelwald, Switzerland

[DLGR05] P. Deloof, C. Laderrriere, J.P. Ghys, A. Rivenq « Car collision sensor based on a noise radar » ITST 2005, Brest, 2005.

[DM99] P. Deloof, A. Menhaj, "The use of a car collision warning radar in the STATUE project for the detection and the identification of broken-down trains by the DIREP system" 6th ITS World congress Toronto 1999, technical session TS-F5, n° 2134, November 8-12 1999.

[DMY02] David A., Moller M. O., and Yi W., "Formal Verification of UML Statecharts with Real-Time Extensions," in R.-D. Kutsche and H. Weber (eds) proceedings 5th International Conference, FASE 2002, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002, Springer, LNCS, Volume 2306, Grenoble, France, April 2002.

[EMF] Eclipse Modelling Framework. www.eclipse.org/emf

[EW04] R.Eshuis, R.Wieringa. Tool Support for Verifying UML Activity Diagrams. IEEE Transactions on Software Engineering. pp437-447. July 2004.

[FRDH00] B. Fremont, A. Rivenq, P. Deloof, M. Heddebaut, “A cooperative collision avoidance and communication system for railway transports”, 3rd IEEE Conference on intelligent transportation systems, ref. : MP4-6, Dearborn (MI) USA, 1-3 Oct. 2000.

[GHK00] Graw G., Herrmann P., and Krumm H., "Verification of UML-based real-time system designs by means of cTLA," in proceedings 3rd IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC2K), IEEE Computer Society Press, Newport Beach, pp. 86-95, 2000.

[GK83] D. D. Gajski, R. Kuhn. Guest Editor Introduction: New VLSI-Tools, in: "IEEE Computer", December 1983, vol. 16, n° 12, p. 11-14.

[GL03] E. Guerra, J. Lara. A Framework for the Verification of UML Models. Examples using Petri Nets. In Jornadas de Ingeniería del Software y Bases de Datos, JISBD’03. Alicante.

[GMP02] M. Gallardo, P. Merino, E. Pimentel. Debugging UML Designs with Model Checking. Journal of Object Technology Vol. 1 No.2, July 2002.

[H87] Harel D., "Statecharts: A Visual Formalism for Complex Systems," Science of Computer Programming, vol. 8, pp. 231-274, 1987.

[HT99] Ji He and Kenneth J. Turner. Timed DILL: Digital Logic in LOTOS. Department of Computing Science and Mathematics, University of Stirling. Technical Report CSM-145. ISSN 1460-9673. June 1999.

[HyTech] http://www-cad.eecs.berkeley.edu/~tah/HyTech/

[InOp7.3] InterOp. Report: Model Transformation State of the Art. Sub-deliverable SD7.3-Part 1. Network of Excellence - Contract no.: IST-508 011. _www.interop-noe.org <http://www.interop-noe.org/>

[InOp9.1] InterOp. State-of-the art for Interoperability architecture approaches. Deliverable D9.1. Network of Excellence - Contract no.: IST-508 011. www.interop-noe.org

[K05] Koo, T. J., EECE 353-1 Real-Time Systems, January 2005 (http://www.vuse.vanderbilt.edu/~kootj/Class/2005/Spring/)

[KMF] Kent Modelling Framework. www.cs.kent.ac.uk/projects/kmf

[Kronos] http://www-verimag.imag.fr/TEMPORISE/kronos/

[LDB05] Ouassila Labbani, Jean-Luc Dekeyser, and Pierre Boulet. Mode-automata based methodology for scade <http://www.lifl.fr/west/publi/LaDeBo05HSCC.pdf>. In Springer, editor, Hybrid Systems: Computation and Control, 8th International Workshop, LNCS series, pages 386–401, Zurich, Switzerland, March 2005.

[LDBR05] Ouassila Labbani, Jean-Luc Dekeyser, Pierre Boulet, Eric Rutten. Introducing Control in the Gaspard2 Data-Parallel Metamodel: Synchronous Approach. International Workshop MARTES: Modeling and Analysis of Real-Time and Embedded Systems (in conjunction with 8th International Conference on Model Driven Engineering Languages and Systems, MoDELS/UML 2005), Montego Bay, Jamaica, October 4, 2005. http://www.martes.org/

[MM03] J. Miller, J. Mukerji (editors). DA Guide (Draft Version 0.2), 2003,http://www.omg.org/docs/ab/03-01-03.pdf.

[MP95] O. Maler, A. Pnueli. Timing Analysis of Asynchronous Circuits Using Timed Automata. ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems. University of Washington, Seattle, WA. November 1995

[NH93] N. Halbwachs. Synchronous Programming of Reactive Systems, Kluwer, 1993.

[NH98] N. Halbwachs. Synchronous Programming of Reactive Systems – a Tutorial and Commented Bibliography, in: "Proc. of the Int. Conf. on Computer-Aided Verification, CAV'98, Vancouver, Canada", LNCS Vol. 1427, Springer-Verlag, 1998.

[NHT96] Akio Nakata, Teruo Higashino and Kenichi Taniguchi: "Time-Action Alternating Model for Timed LOTOS and its Symbolic Verification of Bisimulation Equivalence", in Proc. of IFIP TC6/WG6.1 Joint Int'l Conf. on Formal Description Technique for Distributed Systems and Communication Protocols, and Protocol Specification, Testing, and Verification (FORTE/PSTV'96), Kaiserslautern, Germany, Chapman&Hall, pp.279-294, Oct. 1996.

[NS99] Simin Nadjm-Tehrani, Jan-Erik Stromberg, "Formal Verification of Dynamic Properties in an Aerospace Application.", Formal Methods in System Design, Volume 14, number 2, March 1999, pages 135-169.

[QVT] OMG, "Request for Proposal: MOF 2.0 Query / Views / Transformations RFP," Object Management Group, ad/2002-04-10, April 2002.

[RABH98] A. Rivenq-Menhaj, J. Assaad, O. Bazzi, F. Haine ” Statistical study of an acousto-optic correlator in terms of its bandwith”, Optical Engineering Journal, pp 1-6, Vol 37, N°3, Mars 1998.

[RAHRB98] A. Rivenq-Menhaj, J. Assaad, M. Heddebaut, J.M. Rouvaen, C. Bruneel, ”Combining two radar techniques to implement an anticollision system”, Institut Of Physics, Measurement Science and Technology, Avril 1998.

[RDAR99] A. Rivenq-Menhaj, P. Deloof, J. Assaad, J.M. Rouvaen ” Des systèmes radar

dédiés à l'anticollision", RTS, Dunod, Vol 62, p.24-34, 1999.

[RRDE04] A. Rivenq-Menhaj, J.M. Rouvaen, I. Dayoub, F. Elbahhar, «Detection of coded sequences buried in non gaussian noise using higher order statistics", AMSE Journal, Réf : 03 158 1D, 2004.

[RDER01] A.Rivenq-Menhaj, I. Dayoub, F. Elbahhar, J. Michel Rouvaen, "Detection of Coded Signals in non- Gaussian noise using Higher Order Statistics", ICISP' 2001, 3-5 Mai 2001, Agadir, Maroc.

[RdS-CA03] R De Simone, C André. *"Towards a Synchronous Reactive UML subprofile?"* Workshop on Specification and Validation of UML Models for Real Time and Embedded Systems (SVERTS), « UML » 2003, San Francisco (CA), October 2003.

[TKYBS98] Tasiran S., Khatri S. P., Yovine S., Brayton R. K., and Sangiovanni-Vincentelli A., "A Timed Automaton-Based Method for Accurate Computation of Circuit Delay in the Presence of Cross-Talk," in proceedings 2nd International Conference on Formal Methods in Computer-Aided Design, FMCAD'98, Springer-Verlag, Lecture Notes in Computer Science 1522, Palo Alto, CA, USA, pp. 149-166, November 1998.

[Uppaal] http://www.uppaal.com

P. Deloof, A. Menhaj, "The use of a car collision warning radar in the STATUE project for the detection and the identification of broken-down trains by the DIREP system". *6th ITS World congress Toronto 1999*, technical session TS-F5, n° 2134, November 8-12 1999.

M. Saint-Venant, J. Assad, P. Deloof, "Anticollision radar system based on a correlation receiver", ITS world congress, Turin, 6-9 novembre 2000.

M. Heddebaut, J. Rioult, M. Berbineau and D. Duhot "System and procedure for the

B. Fremont, A. Rivenq, P. Deloof, M. Heddebaut, "A cooperative collision avoidance and communication system for railway transports", 3rd IEEE Conference on intelligent transportation systems, ref. : MP4-6, Dearborn (MI) USA, 1-3 Oct. 2000.

J. Assaad, A. Rivenq-Menhaj, J. Assaad, J.M. Rouvaen , M. Heddebaut, C. Bruneel, " A new collision avoidance radar system using correlation receiver and compressed pulses", IOP (Institute Of Physics), Measurement Science and Technology, Vol. 9, Issue 2, pp 283-286, Février 1998.

A. Rivenq-Menhaj, J. Assaad, O. Bazzi, F. Haine " Statistical study of an acousto-optic correlator in terms of its bandwith", Optical Engineering Journal, pp 1-6, Vol 37, N°3, Mars 1998.

A. Rivenq-Menhaj, J. Assaad, M. Heddebaut, J.M. Rouvaen, C. Bruneel, ”Combining two radar techniques to implement an anticollision system”, Institut Of Physics, Measurement Science and Technology, Avril 1998.

A. Rivenq-Menhaj, P. Deloof, J. Assaad, J.M. Rouvaen ” Des systèmes radar dédiés à l’anticollision”, RTS, Dunod, Vol 62, p.24-34, 1999.

A.Rivenq-Menhaj, I. Dayoub, F. Elbahhar, J. Michel Rouvaen, "*Detection of Coded Signals in non- Gaussian noise using Higher Order Statistics*", ICISP' 2001, 3-5 Mai 2001, Agadir, Maroc.

P. Deloof, C. Laderrriere, J.P. Ghys, A. Rivenq « Car collision sensor based on a noise radar » ITST 2005, Brest, 2005.

B. Fremont, J-P. Ghys, A. Rivenq « Utilisation d'un RADAR coopératif afin d'améliorer la sécurité et la fiabilité des systèmes de transports guidés automatiques’ GRRT 2003.

A. Rivenq-Menhaj, J.M. Rouvaen, I. Dayoub, F. Elbahhar, «Detection of coded sequences buried in non gaussian noise using higher order statistics", AMSE Journal, Réf : 03 158 1D, 2004.

C. Tatkeu, Y. Elhillali, A. Rivenq, Pr. J.M Rouvaen , “Evaluation of coding’s methods for the development of a Radar sensor for localization and communication dedicated to guided transports” I EEE-VTC-2004, Transmission technology, Los-Angeles, Septembre, 2004.