52. IWK

Internationales Wissenschaftliches Kolloquium
International Scientific Colloquium

PROCCEDINGS | 10 - 13 September 2007

FACULTY OF
COMPUTER SCIENCE AND AUTOMATION

COMPUTER SCIENCE MEETS AUTOMATION

VOLUME II

Session 6 - Environmental Systems: Management and Optimisation

Session 7 - New Methods and Technologies for Medicine and Biology

Session 8 - Embedded System Design and Application

Session 9 - Image Processing, Image Analysis and Computer Vision

Session 10 - Mobile Communications

Session 11 - Education in Computer Science and Automation

TECHNISCHE UNIVERSITÄT
ILMENAU

Bibliografische Information der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der deutschen Nationalbiografie; detaillierte bibliografische Daten sind im Internet über **http://dnb.ddb.de** abrufbar.

ISBN 978-3-939473-17-6

Impressum

Herausgeber:	Der Rektor der Technischen Universität Ilmenau Univ.-Prof. Dr. rer. nat. habil. Peter Scharff
Redaktion:	Referat Marketing und Studentische Angelegenheiten Kongressorganisation Andrea Schneider Tel.: +49 3677 69-2520 Fax: +49 3677 69-1743 e-mail: kongressorganisation@tu-ilmenau.de
Redaktionsschluss:	Juli 2007
Verlag:	Technische Universität Ilmenau/Universitätsbibliothek Universitätsverlag Ilmenau Postfach 10 05 65 98684 Ilmenau www.tu-ilmenau.de/universitaetsverlag
Herstellung und Auslieferung:	Verlagshaus Monsenstein und Vannerdat OHG Am Hawerkamp 31 48155 Münster www.mv-verlag.de
Layout Cover:	www.cey-x.de
Bezugsmöglichkeiten:	Universitätsbibliothek der TU Ilmenau Tel.: +49 3677 69-4615 Fax: +49 3677 69-4602

Preface

Dear Participants,

Confronted with the ever-increasing complexity of technical processes and the growing demands on their efficiency, security and flexibility, the scientific world needs to establish new methods of engineering design and new methods of systems operation. The factors likely to affect the design of the smart systems of the future will doubtless include the following:

- As computational costs decrease, it will be possible to apply more complex algorithms, even in real time. These algorithms will take into account system nonlinearities or provide online optimisation of the system's performance.
- New fields of application will be addressed. Interest is now being expressed, beyond that in "classical" technical systems and processes, in environmental systems or medical and bioengineering applications.
- The boundaries between software and hardware design are being eroded. New design methods will include co-design of software and hardware and even of sensor and actuator components.
- Automation will not only replace human operators but will assist, support and supervise humans so that their work is safe and even more effective.
- Networked systems or swarms will be crucial, requiring improvement of the communication within them and study of how their behaviour can be made globally consistent.
- The issues of security and safety, not only during the operation of systems but also in the course of their design, will continue to increase in importance.

The title "Computer Science meets Automation", borne by the 52nd International Scientific Colloquium (IWK) at the Technische Universität Ilmenau, Germany, expresses the desire of scientists and engineers to rise to these challenges, cooperating closely on innovative methods in the two disciplines of computer science and automation.

The IWK has a long tradition going back as far as 1953. In the years before 1989, a major function of the colloquium was to bring together scientists from both sides of the Iron Curtain. Naturally, bonds were also deepened between the countries from the East. Today, the objective of the colloquium is still to bring researchers together. They come from the eastern and western member states of the European Union, and, indeed, from all over the world. All who wish to share their ideas on the points where "Computer Science meets Automation" are addressed by this colloquium at the Technische Universität Ilmenau.

All the University's Faculties have joined forces to ensure that nothing is left out. Control engineering, information science, cybernetics, communication technology and systems engineering – for all of these and their applications (ranging from biological systems to heavy engineering), the issues are being covered.

Together with all the organizers I should like to thank you for your contributions to the conference, ensuring, as they do, a most interesting colloquium programme of an interdisciplinary nature.
I am looking forward to an inspiring colloquium. It promises to be a fine platform for you to present your research, to address new concepts and to meet colleagues in Ilmenau.

Professor Peter Scharff
Rector, TU Ilmenau

Professor Christoph Ament
Head of Organisation

Table of Contents

C O N T E N T S

7 New Methods and Technologies for Medicine and Biology

8 Embedded System Design and Application

10 Mobile Communications

11 Education in Computer Science and Automation

52nd Internationales Wissenschaftliches Kolloquium
Technische Universität Ilmenau
10 - 13 September 2007

M. Müller / A. Pacholik / W. Fengler

Tool Support for Formal System Verification

Abstract

The design of complex real time systems requires verification within all stages of the development process. To ensure quality already in early design stages, formal verification has to be conducted on high abstraction levels and for potentially incomplete system definitions. In this paper we present an approach to formally verify the temporal behaviour of simulation models by introducing a tool chain based upon the system design environment MLDesigner and the verification tool UPPAAL.

1 Introduction

The design process of a complex embedded system aims to construct the right system in a correct way. To build the right system means to integrate the expected behaviour related to the users needs. This is ensured by simulation based validation [1]. For a system to be correct means, it is free of faults in a way, that specified properties can be ensured. This can be proved by formal verification [2].

Today simulation based validation is applied on system level, whereas verification is applied on implementation or block level. There are no techniques available to prove temporal properties in a high level validation model. A complete verification can only be conducted, if block level implementations are available. For example, SystemC is often proposed to be used for system level specifications. However, verification is either conducted dynamically, during simulations, or formally for SystemC models, for which RTL[1] code generation is possible. By the knowledge of the authors formal verification and constraint verification is not possible for generalized SystemC models.

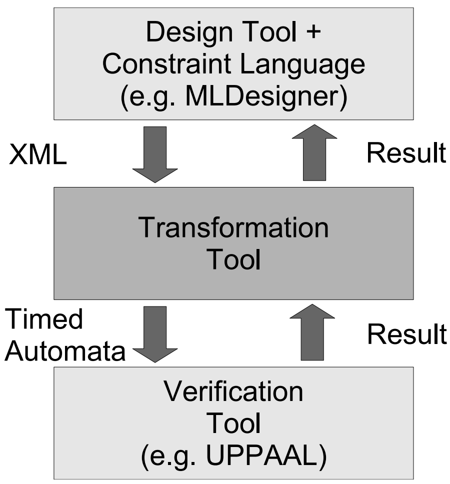

Fig. 1: Verification Flow

To overcome this gap, we present a constraint driven approach to verify system level properties in early design stages. The verification flow realized in our work is depicted in Figure 1. The simulation model is supplemented with formal constraint information. Then this validation model is transformed into a verification model, which is used as input for a verification tool. Results from the verification tool can be back-annotated into the simulation model.

This paper is organized as follows. First related work is presented. Then we describe the verification flow and the transformation tool in detail. The paper ends with our conclusions and an outlook to the future work.

[1]Register Transfer Language

2 Related Work

Based on the requirements identified and proposals made in [3] an approach for verifying a system model's temporal properties has been developed. As proposed, a structured constraint language has been defined and the transformation of time constraint expressions to systems of timed automata has been developed and implemented in [4]. By exploring the state-space of resulting automata the verification of the time constraints can be conducted. The tool UPPAAL is supposed to be utilized for this verification.

Two approaches for designing the systems have been considered, using MLDesigner discrete event models, as proposed in [3], or UML activity, sequence and state diagrams, as proposed in [4]. The tool presented in this paper has implemented the MLD-based approach.

2.1 System Modelling

MLDesigner [5] is a tool based on the ideas of the Ptolemy project [6]. System models can be easily created by using existing customizable functional blocks and connecting them. New blocks can be created using the Ptolemy Language, a facility to describe functional blocks in terms of C++ code fragments. The tool MLDesigner organizes the structural connection, creation, execution and debugging of models. The tool provides the ability to link different models of computation within one system. This enables to create efficient simulation models, but limits the capabilities of implementation code generation.

According to the underlying ideas we can find a lot of similarities between SystemC 2.0 [7] and the MLDesigner Discrete Event (DE) modelling domain. A scheduler is used to organize the execution (simulation) of functional blocks. Timing information is added to the simulation in terms of delay. Data is exchanged using memories and/or pointers. Signals are event based, using the scheduler to inform the functional blocks. When describing hardware, there are some drawbacks according to hardware specific support (e.g. data types) compared to SystemC. In our opinion MLDesigner represents the more general approach on resource and systems modelling. Furthermore it provides powerful debugging opportunities according to the discrete event modelling approach.

The MLD system models are optimized for fast simulation speeds. Due to these optimizations (extensive use of pointers, abstract data types, simulation management, describing functional behaviour in terms of C++ code), it is hard to extract a state model to apply model checking techniques directly. In our approach we substitute the C++ code with timing constraints, a structured language defined in EBNF describing the behaviour based on events, abstracting from data structures. Of course, finite state modules (domain FSM) can be embedded in discrete event models (domain DE).

2.2 Constraint Languages

There are different ways to classify properties. One is to distinct between constraints and assertions where constraints are limitations to both the environmental behaviour (regarding to the inputs) and the expected system behaviour (regarding to system outputs) [8]. Another is to state all side effect free specifications as system constraints, as done in the Object Constraint Language (OCL) [9].

In this paper every specified behavioural property is named *constraint*. The role of such a property depends on the view on the system or its fragments. A property can be expected to be already fulfilled or asserted to be fulfilled by the current device under verification. Regarding to the system modelling environment depicted above, we need to specify the use of (sub)systems within their environments. This includes to specify restrictions on inputs and outputs (safety properties), following called *specification constraints*. We also need to specify behavioural constraints between inputs and outputs, representing a (preferably minimal) subset of the systems expected behaviour, following called *implementation constraints*.

2.3 Verification Approach

There exist different kinds of models, that are suited for formal verification: clocked state models, timed automata, Petri nets, interval Petri nets, process algebra models, etc. These models provide different possibilities to model data, synchronization and timing. On the other side there are different, optimized internal data structures to represent state space and algorithms to apply state traversal and property checking, e.g. BDD[2] based symbolic verification. Because of the use of zero timed events in system models, we cannot use clocked state models. We need a state model involving timing in terms of clocks. Thus we decided to use timed automata, providing channel based communication, a synchronizing scheme, which is very similar to events. For verification we use the tool UPPAAL [10]. It provides a good support and is freely available.

3 Enhanced system models

Although well suited for developing event based behavioural system models, MLDesigner does not feature adequate means to describe the temporal behaviour of model elements [3]. So MLD model elements have to be artificially enhanced by Constraint Language expressions. There are four different model elements to be considered for applying temporal constraints: on one hand *System*, *Module* and *FSM* as complex (sub)model descriptions, on the other hand the subordinate model element *Instance*, which represents a functional part inside a complex model.

Of importance for model consistency is to keep the right *perspective*, when formulating the constraint expressions for a certain model element. This concept of perspective is originated in the design perspective of MLD, which limits the view to the elements residing in one hierarchical level and does neither provide nor require complete knowledge of the full model hierarchy. This has a major influence on the complexity of the event identifiers used within the constraint expressions.

Since the functional behaviour within a system is supposed to be provided by *Instance* blocks, they have to carry **implementation** constraint expressions, that describe the temporal dependencies between their input and output signals. The perspective is thereby limited to the boundaries of the particular instance.

The *System* and *Modules* as complex system definitions contain instances and their relations. The perspective is a *local perspective* set to the system's own hierarchical level, so the constraints expressed on this level have to be **specifications** related to the (sub)system's inner signals. Concerning verification the necessity may arise to make assertions on the occurrence of any signal from any instance on any subsystem level. So, for the *System* an additional *global perspective* needs to be defined, which exceeds the MLD provided view and requires knowledge of the complete model hierarchy. Here **global specification** constraints have to be formulated, that refer to signals throughout all model hierarchy levels.

[2]binary decision diagram

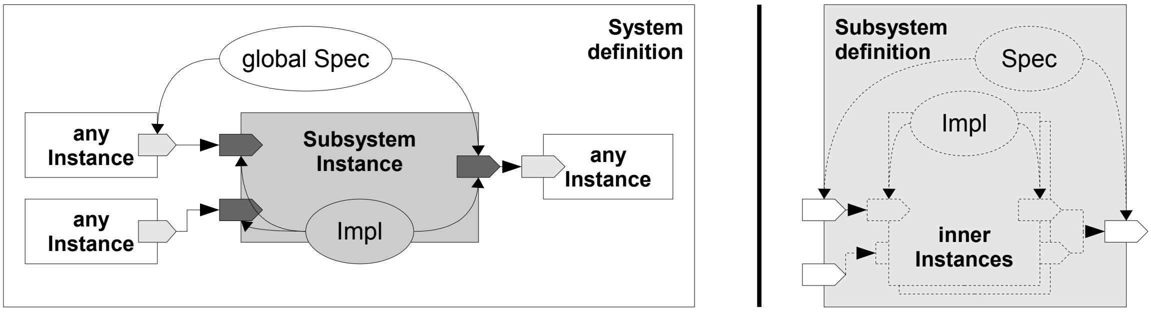

Fig. 2: MLD model elements and constraints

An *FSM* block contains an automaton structure (states and transitions). The perspective is limited to the boundaries of the FSM. The internal behaviour of an FSM depends on the *states*, so the **FSM implementation** constraints timing the state transitions are stored there. *Transitions* contribute to the internal behaviour by featuring *conditions* and *actions* - expressing the dependence of the transition from input events on one hand and the dependence of output events from transitions on the other.

The MLD modelling paradigm enables a once defined module or primitive to be instantiated in systems or modules various times on different hierarchy levels. Since the transformation produces a flat model, system wide unique identifiers have to be introduced. The concept of distinct constraint perspectives combined with respectively enhanced event identifiers eases the element handling and prevents conflicts on the resulting single system level.

4 The tool MLD2UPPAAL

The tool MLD2UPPAAL conducts the transformation from the enhanced MLD model to the UPPAAL timed automata system by passing three main conversion stages. Two intermediate model formalisms have been developed to support the transformation and provide a basis for different model manipulations. A summary of this workflow is shown in Figure 3.

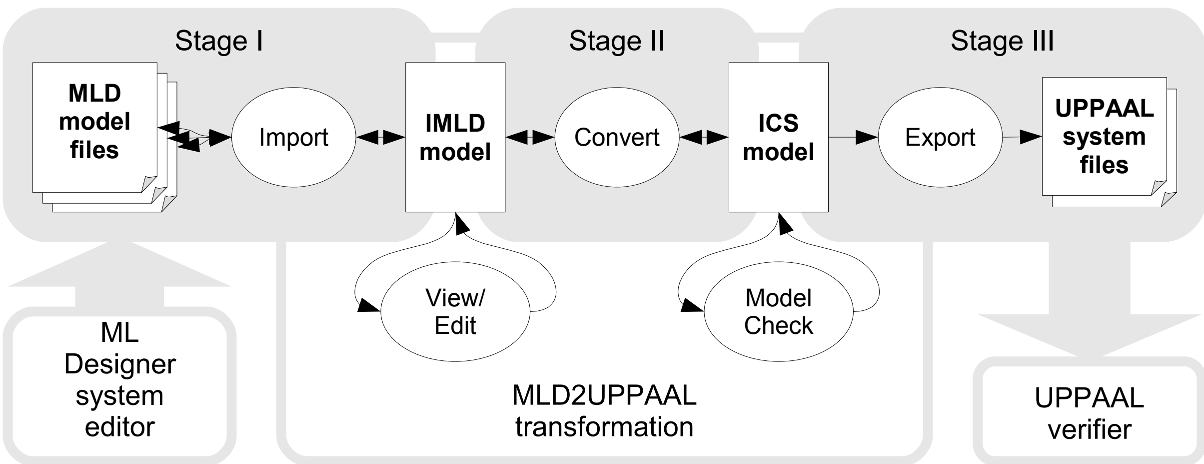

Fig. 3: MLD2UPPAAL workflow

Since an MLD model features a hierarchical structure, the model and each subsystem and element definitions are stored in separate files. In the first processing stage a given set of model files, including the top level system file and all referenced files, are imported into an *Intermediate MLD Model (IMLD)* instance. Main purpose of this intermediate formalism is to join the MLD system and subsystem definitions into one model while maintaining the hierarchy information. At this stage the application allows varying the level of hierarchy resolution and the revision of the constraint expressions. As the workflow indicates, the original model definition is kept consistent to the intermediate models by backpropagating content changes. The second stage converts the intermediate MLD model into an instance of the *Intermediate Constraint System Model (ICSM)*. During this conversion the system structure is flattened, which results in the substitution of all subsystems according to whether they are to be resolved or not. Unresolved subsystems are treated as simple instance blocks with implementation constraints, whereas resolved subsystems instances are replaced by their inner structure and local specification constraints to provide assertions about that system partition. The ICS model notation allows the check of syntactic and semantic constraint consistency in consideration of the concept of perspective mentioned above. Once the check is completed successfully, the ICS model is transformed to timed automata using UPPAAL templates to map the constraint properties. An UPPAAL-compliant XML-notation is stored in model and query files, that serve as input for the UPPAAL verifier.

The tool is implemented as a Java application, that appears as shown in Figure 4, and was successfully tested against selected examples.

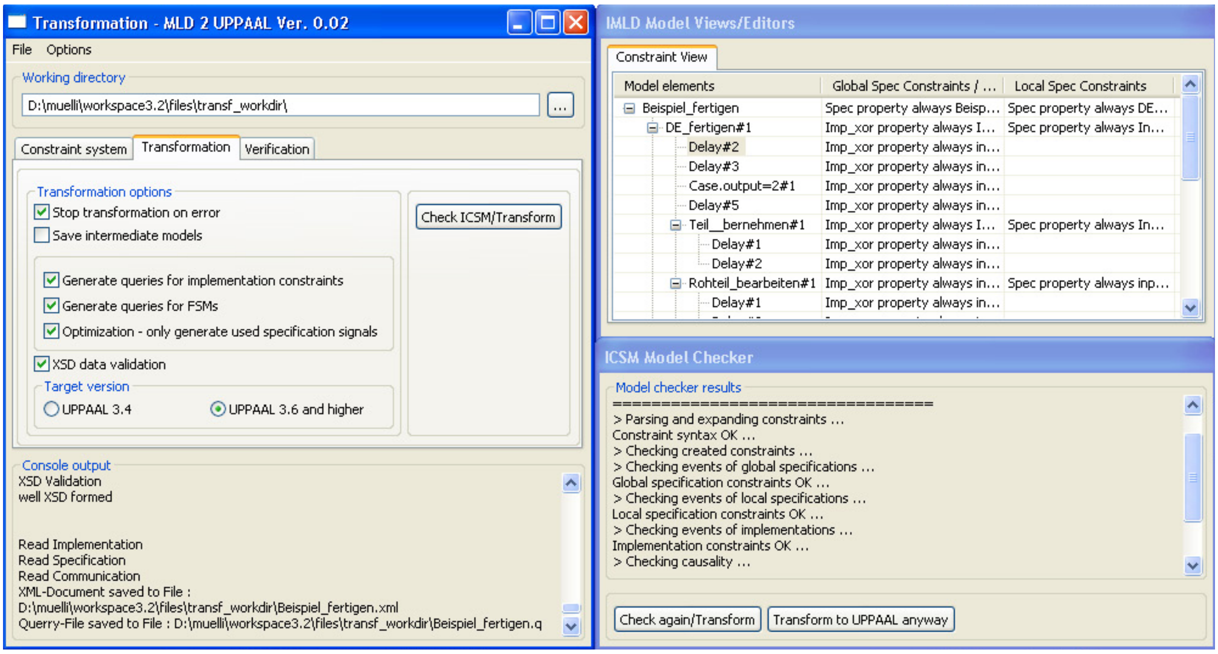

Fig. 4: MLD2UPPAAL screenshot

5 Conclusions and future work

By introducing the temporal constraint language and applying it to MLDesigner system models we showed the possibility to add formally verifiable properties to models with high degrees of abstraction. The tool implemented in this work's context gives an example of how to bridge the gap between design and verification formalisms. Since it has been derived from the application framework introduced in [11], a customization to various design sources and verification targets is easily possible. The given transformation of constraints to UPPAAL automata can be easily extended by adding custom templates. Currently we are going to apply our verification method to a number of design models to validate and benchmark it regarding model complexity. Using completely formally interpretable simulation models will make the models more directly verifiable and thus reduce the amount of additional information to be supplied in form of constraints. An according approach concerning extended finite temporal properties has already been presented in [12].

Acknowledgements

This work has been funded by edacentrum e.V. within project FEST.

References

[1] Schorcht, G. ; Unger, P. ; George, A. ; Troxel, I. ; Zinn, D. ; Salzwedel, H. ; Farhangian, K. ; Mick, C.K.: System-Level Simulation Modeling with MLDesigner. In: *IEEE / ACM MASCOT 2003 - 11th ACM / IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 12.-15. Oktober 2003, Orlando, Florida, USA* (2003)

[2] T.Kropf: *Introduction to Formal Hardware Verification*. Springer Verlag, 1999. – ISBN 3–540–65445–3

[3] Pacholik, A. ; Fengler, W. ; Salzwedel, H. ; Vinogradov, O.: Real Time Constraints in System Level Specifications Improving the Verification Flow of Complex Systems. In: *Net.ObjectDays 2005* (2005), September

[4] Klöckner, J.: *Transformation of UML System Descriptions to Verify Temporal Properties*. 2006. – Diploma Thesis, in German language

[5] MLDesign Technologies, Inc.: *MLDesign Technology Homepage*. : MLDesign Technologies, Inc., 2006. – http://www.mldesigner.com/

[6] Department of EECS, UC B.: *Official Homepage of the Ptolemy Project*. : UC Bercley, EECS, 2006. – http://ptolemy.eecs.berkeley.edu/

[7] Ghenassia, Frank: *Transaction-Level Modeling with SystemC*. Springer, 2005. – ISBN 0–387–26232–6

[8] Foster, H.D. ; Krolnik, A.C. ; Lacey, D.J.: *Assertion-Based Design 2nd Edition*. 2. Kluwer Academic Publishers, 2004. – ISBN 1–4020–8027–1

[9] Warmer, J. ; Kleppe, A.: *Object Constraint Language 2.0*. mitp-Verlag Bonn, 2004. – in german language. – ISBN 3–8266–1445–3

[10] Department of Information Technology, Sweden: *Official Homepage of the Uppaal Tool*. : Department of Information Technology, Uppsala University, Sweden, 2006. – http://www.uppaal.com

[11] Müller, M.: *Metamodel based Transformation of System Models - an Application Framework*. 2006. – Diploma Thesis, in German language

[12] Pacholik, A. ; Fengler, W.: A System Model for Formal Verification of TLM based Transaction Properties. In: *10th Communications and Networking Simulation Symposium (CNS) 2007* (2007)

Autors:

Dipl.Inf. Marcus Müller,
Dipl.Ing. Alexander Pacholik,
Prof. Dr.-Ing. habil. Wolfgang Fengler

Department of Computer Architecture
Technical University of Ilmenau
98684 Ilmenau, Germany

{marcus.mueller, alexander.pacholik, wolfgang.fengler}@tu-ilmenau.de