THE RECONFIGURABLE UML MACHINE PROJECT GROUP

Achim Rettberg [1], Tim Schattkowsky [1], Carsten Rust [1], Wolfgang Müller [1] and Franz Rammig [2]

[1] *University of Paderborn/C-LAB*
University of Paderborn, Germany
{achim, tim, car, wolfgang}@c-lab.de

[2] *University of Paderborn, Germany*
franz@upb.de

Abstract: This paper describes a seminar of a project group consisting of students that realizes a code generation from UML to a reconfigurable hardware platform. The usage of high-level modeling is nowadays essential in the software and in the hardware development. The reason for this is the complexity the industries have to cope with. The idea to specify a hardware system at such a high abstraction level should be realized for a given application within the project group. By dealing with the application the students learn to use these new development methods for system design, especially hardware design. Furthermore, an overview of the structure and organization of project groups are given.

Keywords: UML, Code Generation, Reconfigurable Hardware, Teaching Project Groups, Student Teams

1. INTRODUCTION

Common trends and common problems have emerged lately in the hardware and software industries. In the hardware industry, the growing complexity of hardware components (e.g., systems-on-chip), the challenge of design verification, and the need to meet the competing objectives of performance/power/time-to-market have made it essential to increase the level of abstraction in the hardware design process.
This leads to the usage of high-level modeling, using variants of traditional hardware description languages (SystemVerilog [15]) or general-purpose programming languages (SystemC [13], SpecC [14], Handel-C [1]). Similar pressures exist in the software industry: the increasing complexity of software systems coupled with the need for increased performance and lower cost have led

important sectors of the industry, e.g., avionics companies, to adopt model-based approaches to software development and to increase usage of modeling languages such as UML [11]. In both software and hardware, the availability of standard IP components has produced an urgent need for new methods to adapt and integrate these components to build reliable, cost-effective systems. The abstract models, languages, and analysis techniques produced by formal methods research provide a sound methodological basis for the high-level modeling, design, and development of both hardware and software and for adapting and integrating existing components to meet new requirements. In the context of hardware/software systems specification, executable UML became of major interest, see [12].
The objective of the project group is to convey these new trends and technologies to the students.

In this paper we describe the usage teaching of a project group in a seminar to realize a code-generation from UML to a hardware description language (HDL) that is synthesized to a reconfigurable hardware platform. First, we define the idea behind a project group at our University, see section 2. The project group of the reconfigurable UML machine consists of students that are separated in three different sub-groups. In section 3 we give a project description and we describe briefly the tasks of the sub-groups. The last section 4 concludes the paper.

2. STUDENT PROJECT GROUPS

In the event form "project group" a group of 8 to 15 students works on a topic provided of the organizer over the time period of a year (two semester). Topics of project groups are introduced in a common event. Interested parties can clear the prerequisites for the participation and cover a project group.

Project groups have on the one hand aims to support the personality of the participants and on the other hand aims which orientate themselves at the finished contents. Teamwork and organization of a project are practically proven and learned in the project group; through this, the participants are prepared for the later industrial professional practice. The students get to know extensive development processes in a self-organized team. The compulsion to report within the group and to hold the results about work of one's own arises from the job sharing.
The content of the project groups shall the students get close to current research themes, which typically are from the research areas of the organizers. Project groups in this respect are not but primarily also aids of the university research. For the students this means that the graduates of a project group are generally

predestined to take master theses in the connection from the corresponding research area.

The topic to be worked on or problem should reflect current research questions to the motivation of the group and as a preparation on possible connection work. On the other hand the topic should be adequate for the students. Independently, in which field of work a project group is organized, the participants should learn a methodical and systematic procedure adapted to the respective field of work within the work. If the implementation of software is the primary objective of the event, the methods and techniques of the development of the software should be used systematically as learned in the computer science course.
It must be made sure that the participants satisfy the formal prerequisites (admittance for the diploma examination) and the prerequisites as regards content and that they are interested in the topic. Project teams live on the motivation of the participants.
The project group should realize the far-reaching self-organization as the highest organization principle. This is reached through

- a discussion at the beginning of the project team about the topic together with the organizer;
- acquirement of the knowledge about and the choice of the systematic procedures, methods and tools relevant for the topic - typically in one initials seminar phase;
- consistent award of "position", i.e. distribution of responsibilities within the group;
- finding out and promoting of special talents within in the participants, yielded e.g. by seminar lectures or the task distribution;
- construction of a process oriented staff structure like an industrial development team; Delegating of sub-tasks to small groups which then report;
- regular lectures for the work progress single and of small groups;
- construction of a final report that shall be strongly distributed between the participants.

The self-organization finds her limits since the organizers must judge the participation individually at the end. It is necessary to pay attention on the following for the fairness sake:

- all participants should be consulted after possibility for works in all appearing activity profiles (e.g. programming, documentation, report construction, work organization);

- complete covering depth of the participants with special working tasks or avoidance of " task accumulation" by the participants;
- control of the complete job performance within the group and compensation at a dissimilar distribution;
- and control of largely complete presence of all participants during the two semesters takes.

3. PROJECT DESCRIPTION

For the development of complex embedded systems reconfigurable hardware is increasingly inserted to make prototypical implementations possible. As a development platform Field Programmable Gate Arrays (FPGAs) are applied by using hardware description languages like VHDL. This manner of the programming presupposes a high degree of expert knowledge. Besides this, a higher more abstract method for the hardware description is desired enriched with methods from software development. The Unified Modelling Language (UML) has established himself as standard for the model based development of software. The modeling of structure and behaviors of a FPGA implementation is possible with the present language size of the UML and therefore shall be pursued in this place.
An automatic generation of the software is also desirable besides the pure modeling out of the model. As a rule, UML software generators confine themselves to the version of the structure in the form of classes and methods. At this approach the behavior must be completed by the programmer later. As an alternative to it the software can be described in form of an executable model.The run time response is specified completely and the generated software requires only few till no customization by the programmer. Among other things behavior description can be made possible with condition or activity diagrams in UML. With the reduction of the language size such an executable model can be described. The aim of this project group is to develop UML subsets, which make the development of executable models possible. These shall serve as a higher description of software for reconfigurable hardware. Furthermore, the aim is the realize an automated synthesis of the model to the hardware and developing the missing tools.

Methods shall be developed in the project groups to be able to derive an implementation on reconfigurable hardware from the specification of an embedded system automatically.
The specification is carried out in the form of diagrams at a high abstraction level. Two approaches are followed up: Activity Diagrams as well as State Diagrams. The Unified Modeling Language (UML) forms the common frame. Though, the languages of the UML cannot be used on the full scale. It is nec-

essary to form well-defined subsets of the diagrams to enable the realization of a given specification in UML to an implementation in HDL. The HDL-Code is synthesized to a FPGA. A FPGA is an integrated circuit which still can be programmed (reconfigured) after his production.
Figure 1 show the design flow used in the project group. The essential step in the design flow is the translation of the specification into HDL. The following steps of the HDL for the implementation are already supported by existing tools. As HDL we use the C-based language Handel-C, see [1]. Handel-C extends the programming language C by constructs which are needed for the description of hardware or reconfigurable hardware. The expansions make for example the description of parallel processes or temporal aspects possible.

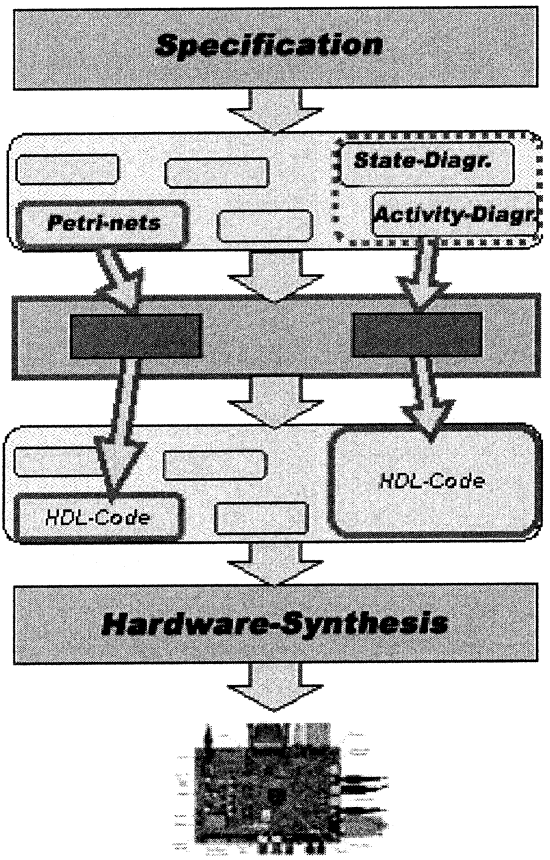

Figure 1. Design Flow

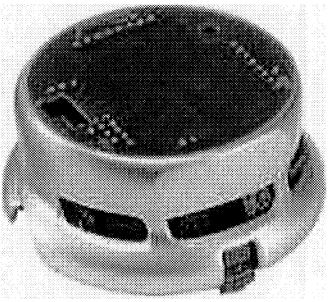

Figure 2. Khepera Roboter

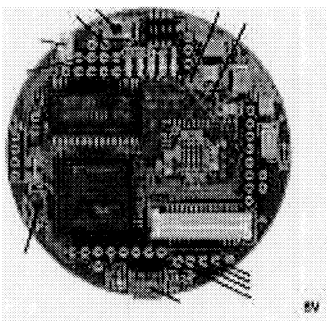

Figure 3. Khepera FPGA Module

The stepwise implementation of a specification in a FPGA implementation is carried out in the project group at a concrete example, i.e. a demonstrator. The target platform is the Khepera mini-robot ([2], [3] and [4]), see Figure 2, with the FPGA extension module depicted in Figure 3. The FPGA module is developed at University of Paderborn in the group of Prof. Rückert, see [8].
As demonstrator an application scenario in which several Kheperas can communicate over radio [5] solve a task together are realized. A possible application scenario is the represented intersection management with Kheperas at which several robots shall collision freely cross an intersection from different directions, see Figure 5.

In our research group we developed a tool to specify High-Level Petri-nets [7]. The tool enable the generation of SystemC and Handel-C code for a given Petri-net, see [6]. To use this tool one task of the project group is to develop transformation rules from High-Level Petri-nets to Activity Diagrams. As UML modeling tool we use Enterprise Architect (EA) from SparxSystems,

see [9]. EA offers XMI as a internal representation of a UML diagram. The project group is separated into four sub-groups, see Figure 4.

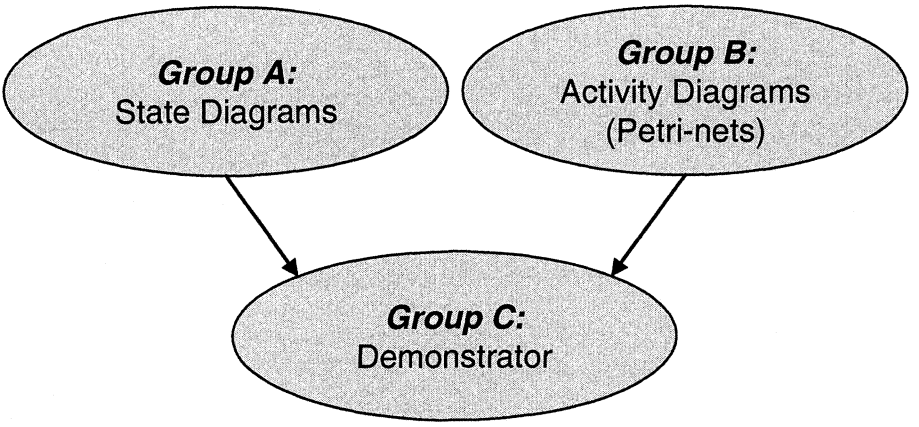

Figure 4. Sub-groups within the Seminar

The tasks to be done for the sub-groups are:

- *Group A* (State Diagrams): Specify the demonstrator with State Diagrams and implement a Handel-C code generation from XMI description.
- *Group B* (Activity Diagrams): Specify the demonstrator with Activity Diagrams and transform the diagram by transformation rules into a High-Level Petri-net and use the existing Petri-net tool to generate Handel-C code.
- *Group C* (Demonstrator): Implement small test models on the target platform and a communication library for the FPGA and the sensor and actuators of the Khepera mini-robot.

UML state and activity diagrams describe executable models in separate approaches. The complete language size of the two diagram types isn't supported in this place. Redundant and complex language constructs are limited or not used. For a detailed description of the permitted elements a subset for state and activity diagrams was defined, see [10].

The Khepera is started at a certain position in the intersection, see figure 5. By this way the initial position of the robot is set. After starting the Khepera recognizes and counts the gaps in the wall. The robot knows by the recognition of the gaps that he shall turn to avoid a collision with the wall. If no gap is recognized and the front sensors show an obstacle the Khepera assumes that another

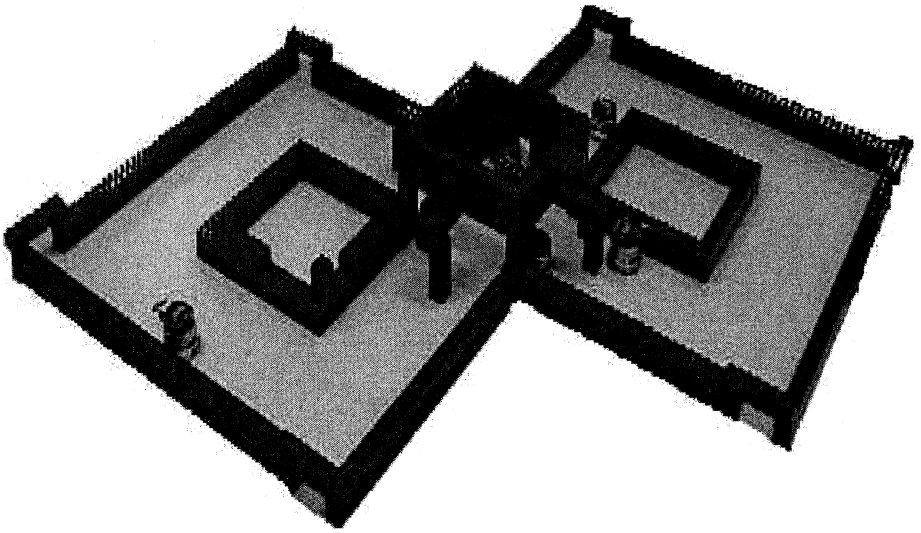

Figure 5. Test Track

robot stands in front of him and stops also until the obstacle is gone. After three turns the Khepera runs into the intersection. A message is also send to the communication module. The Khepera is in the so-called search zone now. The Khepera recognizes the entry to the action zone if the left sensor shows an obstacle. Markings corresponding to the intersection are integrated representing an obstacle on the left side. The Khepera can drive to the left, to the right or straight ahead now. In the dependence of the direction the internal Khepera controller is converted correspondingly. The controller has two conditions for the inner and the exterior wall. If the robot drives inside, no gaps can be counted. So the robot can detect if he shall turn again three times when e.g. he turns right to the left or to the right or no-one, if he drives straight ahead and reaches an inner wall.
The intersection is divided up into three areas: the search zone, the planning zone and the driving zone. To increase the efficiency of the algorithm at the right of way regulation the driving zone still is divided up in four areas (of sectors) so that four Khepera are able to come into the intersection at the same time.

4. CONCLUSIONS

The aim of the project team was to make an automated development of the executable model possible for the hardware description. Operational application scenarios should demonstrate the individual steps when translating

automatically and make a presentation of the results possible. The goal was accomplished in large portions by the project team. The use of UML activity and state diagrams was planned. It was achieved that models can successfully be translated from the formed subsets of activity and state diagrams into the target language HandelC.
The above approach targets at generating complete hardware implementations from Handel-C programs. With the developed methods a synthesis semantics for state and activity diagrams are given. Especially control-oriented behavior and parallelism map nicely to Handel-C. However, there are several critical issues when mapping the given UML subset to hardware.
The project group solved the problem under their own responsibility. The division of the project groups made sense and supported the team thought. By the application example, the participants were very motivated and received excellent knowledge in the research area. Many of them have already started with their master degree in research area.

REFERENCES

[1] Celoxica Ltd.: Handel-C Language Reference Manual, Document Number: RM-1003-4.2, 2003.

[2] http://www.k-team.com/, 2002.

[3] K-Team S.A. Khepera 2: User manual. K Team S.A. Ch. de Vuasset, CP111 1028 Preverenges Switzerland, 2002.

[4] K-Team S.A. Khepera: BIOS manual. K Team S.A. Ch. de Vuasset, CP111 1028 Preverenges Switzerland, 1999.

[5] K-Team S.A: Radio Base User Manual. K Team S.A. Ch. de Vuasset, CP111 1028 Preverenges Switzerland, 1999.

[6] Rust, Carsten; Rettberg, Achim; Gossens, Kai: From High-Level Petri Nets to SystemC. In: IEEE International Conference on Systems, Man & Cybernetics. Hyatt Regency, Washington, D.C., USA, 5 - 8 October 2003.

[7] Rust, Carsten; Rammig, Franz Josef: A Petri Net Based Approach for the Design of Dynamically Modifiable Embedded Systems. In: Kleinjohann, Bernd (Hrsg.): Design Methods and Applications for Distributed Embedded Systems IFIP WG 10.5, Kluwer Academic Publishers, 23 - 26 August 2004 Proc. IFIP TC 10 Conference DIPES 2004.

[8] Grünewald, Matthias; Rust, Carsten; Witkowski, Ulf: Using mini robots for prototyping intersectionmanagement of vehicles. In: Proceedings of the 3rd International Symposium on Autonomous Minirobots for Research and Edutainment (AMiRE 2005). Awara-Spa, Fukui, JAPAN, 20 - 22 September 2005

[9] http://www.sparxsystems.com

[10] Bühler, A.; Gerst, D.; Giefers, H.; Oette C.; Krivih, A.; Riemer, A.; Schilke, H.; Schilke, E.: Projektgruppe REUMA: AbschluSSbericht. Internal Report, University of Paderborn, 2005.

[11] The Object Management Group: Unified Modeling Language: Superstructure. OMG ad/2003-04-01, 2003.