PARADISE: *Design Environment for **Para**llel & **Dis**tributed, **E**mbedded Real-Time Systems*

W.Hardt, P. Altenbernd, C. Böke, G. Del Castillo, C. Ditze, E.Erpenbach, U. Glässer, B. Kleinjohann, G. Lehrenfeld, F.J. Rammig, C. Rust, F. Stappert, J. Stroop and J. Tacken

Today's embedded systems (ES) are characterized by more and more parallelism, distribution over different locations and hard real-time (RT) requirements. Consequently, the modern, structured design process has to deal with heterogeneous requirements and restrictions. The integration of several core competencies is essential for establishing a methodological structured design process for parallel distributed embedded real-time systems. In this paper we present an overall structuring of the design process as well as the Paradise Design Environment reflecting the design process structure. Our activities are based on wide, theoretically well-founded concepts and tool support is built upon these theoretically foundations.

1 Introduction

Today's embedded systems (ES) are characterized by more and more parallelism, distribution over different locations and hard real-time (RT) requirements. Consequently, the modern, structured design process has to deal with heterogeneous requirements and restrictions. Especially the integration of HW-design tasks with SW-design tasks and operating system functionality with respect to RT requirements is a main challenge. The integration leads to systems with enormous overall complexity because the complexity of each design partition itself has been doubled nearly every three years. This explains the need of well-founded theoretically paradigms, automation tools, and a globally applicable design methodology.
Traditionally several design domains are known and existing beside each other. Knowledge, algorithms and techniques used in a special design domain may be called

F. J. Rammig (ed.), *Distributed and Parallel Embedded Systems*

core competence. Core competencies needed for establishing a design methodology for today's ES are:

- specification and modeling
- analysis and verification
- RT SW-synthesis and RT operating systems
- HW-synthesis and rapid prototyping.

A core competence should cover all levels of abstraction within one design domain. A very good structuring of the HW-design domain has been suggested by Gajski. This structure is called Y-chart and distinguishes a behavioral, structural, and a geometry design-view. All three design-views are additionally divided into five different hierarchical layers, namely: the algorithmic, register-transfer, gate, symbolic-layout, and the electrical-layout-layer. Under security and productivity aspects a test-view has to be introduced as fourth design-view leading to the X-chart [Ra89]. The X-chart extends the Y-chart by the test-view. If this abstract structure is generalized for the design of today's ES a global applicable design methodology can be established.

In our approach the X-structure is applied to each design domain separately. Once the abstract structure is found domain specific interpretations can be brought together. This new abstract structuring of the design process of today's ES is illustrated in Figure 1. The five different layers of abstraction (algorithm to layout) of Gajski's Y-chart are depicted for each of the eight core competencies and each one is structured by Rammig's X-chart.

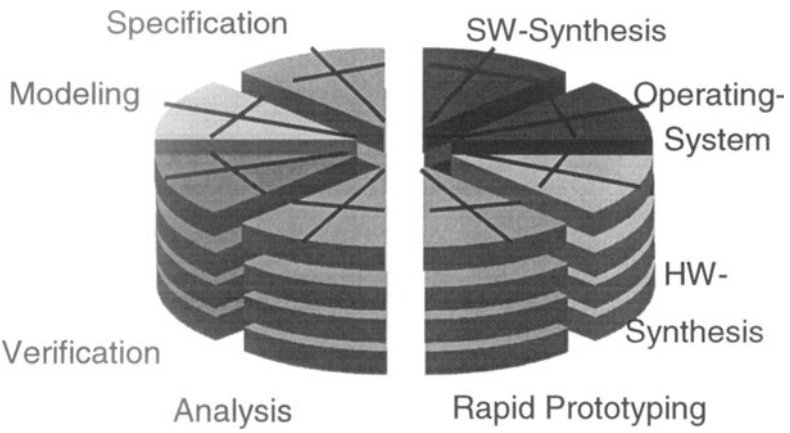

Figure 1. Abstract structuring of today's ES design.

Based on this concept a variety of automation tools for different design domains have been integrated within the PARADISE Design Environment. The PARADISE Design

Environment is understood as an approach towards an implementation of a globally applicable, integrated design methodology. In this paper we give an overview over the concept and the integrated tools. Next the theoretical concepts structured by the core competencies are explained. Then a summery of the tools integrated in the PARADISE Design Environment is given. Finally, the benefits resulting from the PARADISE design methodology are illustrated.

2 Concepts

Our core competencies are founded on the following theoretical concepts:

- abstract state machines (ASM)
- predicate transition nets (PrT-net)
- control-data flow graph (CDFG)
- state-charts (SC).

Abstract State Machines - formerly called Evolving Algebras [Gur95] - combine declarative concepts of first-order logic with the abstract operational view of transition systems in a unifying framework for mathematical modeling of discrete dynamic systems. The underlying computation model constitutes a simple yet powerful semantic basis to deal with concurrent and reactive behavior in a direct and intuitive way; it naturally enables operational interpretations of high-level system specifications and thereby facilitates machine-supported analysis and validation (e.g. through simulation) of the resulting models. Numerous ASM applications (see for instance the annotated ASM) to real-life systems engineering problems - in particular, formal semantics of programming languages, modeling languages and protocols - contributed to establish general abstraction principles needed to cope with the complexity of large systems. In contrast to many traditional formal methods with a monolithic character, the ASM method is open to be combined with other (e.g. application domain specific) modeling techniques, thus providing additional flexibility

The second concept, extended predicate transition nets (Pr/T-Nets), has been developed from petri nets. We choose a high level form of petri nets, since pure petri net models tend to become very large and therefore rather incomprehensible when modeling realistic systems. We extended the basic definition of Pr/T-Nets with hierarchy and timing concepts. Our timing concept supports the specification of real-time restrictions as well as execution and idle times for the actions assigned to transitions in a Pr/T-Net. Hierarchical Pr/T-Nets support a modular specification and allow the reuse of predefined nets in several models. The definition of hierarchical Pr/T-Nets within our formalism is closely coupled with a mechanism for graphical abstraction of Pr/T-subnets. Based on this graphical abstraction a domain specific representation of the internal model is possible.

The third concept, the control/data flow graph (CDFG), brings together control flow oriented information with data flow. During the history of research in HW-design automation both aspects have been evaluated intensively. As one result may be stated, that both aspects have valuable advantages. This has been realized by the definition of

the CDFG representing both and provide references in between. CDFGs are used by the core competencies HW-synthesis and rapid prototyping.
Last, the state-chart concept is well suited for the specification of reactive embedded systems. Closely related to the paradigm of synchronous languages, state-charts offer ideal abstraction primitives for a deterministic specification of a system's behavior. Work is going on to automatically transform such specification into software realizations not only for rapid prototyping but also for target systems.

3 Tool support

The previous section summarizes the conceptional base on which the core competencies have been implemented. All core competencies are supported by specialized tools. In this section we give an overview over the tools available in PARADISE Design Environment. Ongoing extensions ensure the adoptions to new algorithms, requirements or usability aspects. The modeling core competence is based on ASMs. This concept was implemented in the ASM-Workbench. This includes simulation and comfortable debugging of ASM-models (Figure 2). It has been found that ASM-model execution is much faster than traditionally used co-simulation. This is because the simulation granularity of traditional co-simulation is fixed.

ASM-SL: Specification Language for the tools of the ASM Workbench
ASM AST: Abstract syntax tree representation of ASM-SL
ASM runs: Representation of ASM simulation traces
MONTAGES: Methodology for the specification of programming languages: extends abstract state machines by a visual formalism for static semantics.

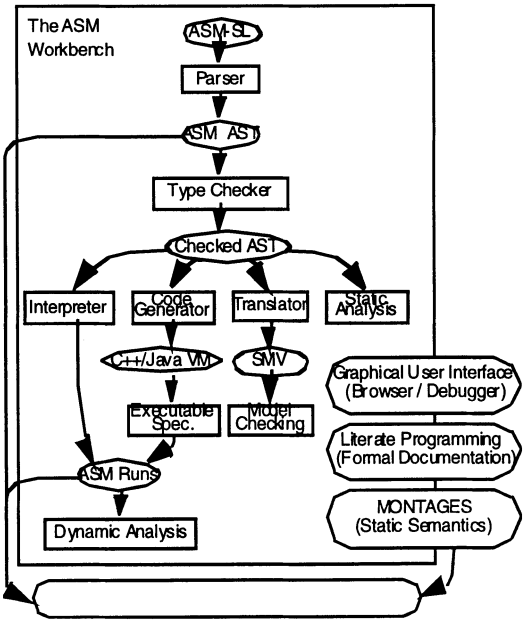

Figure 2. ASM Workbench

The specification and analysis tasks are based on extended Pr/T-Nets and are supported by the tool SEA. In addition to Pr/T-Nets, SEA allows the integration of other design domain specific languages due to the mechanism of graphical abstraction (Figure 3). For the resulting common extended Pr/T-Net, SEA provides powerful simulation and visualization facilities [KlKlTa96]. In order to analyze and verify the model, some methods known from petri nets have been applied to Pr/T-Nets. The analysis results are used for optimization of the model. For further processing by other components of the PARADISE Design Environment a description in a 'Meta Language' containing both, the model as well as the analysis results, may be generated.

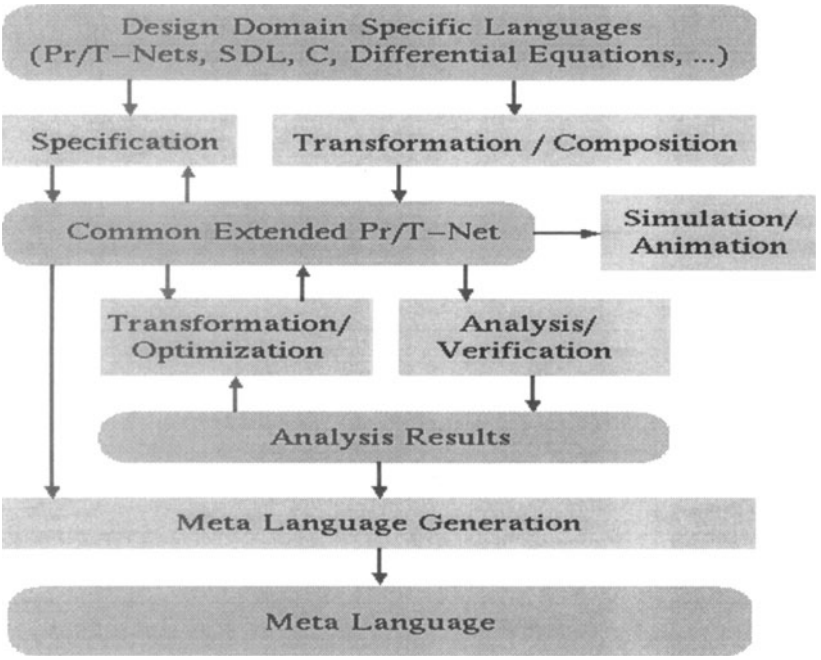

Figure 3. Concept of specification and analysis tool SEA

HW/SW-codesign partitions the specification extracted from SEA. Consequently the design tasks HW-synthesis, RT-SW-synthesis and RT-operating system design can start wherefore the PARADISE Design Environment provides automation tools:

PMOSS (HW-synthesis)
CHaRy (RT-SW-synthesis
TEReCS (RT-communication
DReaMS (RT-operating system design).

The main features of PMOSS are HW/SW partitioning, high-level transformation for early design optimization and high-level synthesis [HaRo97]. For implementation a modular concept has been chosen, i.e. classes of algorithms for all tasks have been

implemented. Thus each task can be adapted to the primer optimization goal by choosing the best algorithm. The main tasks are depicted in Figure 4. Several frontends read from tools within the PARADISE Design Environment, results are visualized by graphic editors and brought back by backends.

RT-SW-synthesis for parallel distributed systems can be performed by the CHaRy tool. As CHaRy processes deterministic C-code descriptions this toolset can ideally be combined with a state-chart language compiler currently under development. This compiler automatically transforms state-chart models into a software implementation based on C code. In order to generate production quality code, e.g. for microcontroller targets, analysis results of CHaRy will be considered during the automatic transformation. Several tasks for the guaranty of real-time behavior within a heterogeneous architecture are available (see Figure 5).

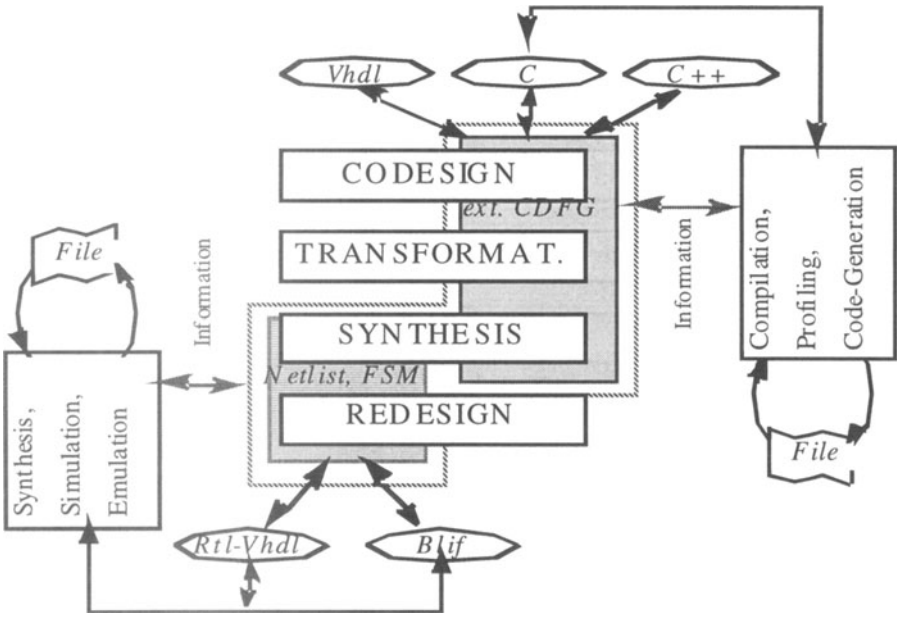

Figure 4. Main tasks of design tool PMOSS

The operating system design within the PARADISE Design Environment is driven by the today's ES's requirements. Experiences gained from the design of micro-kernel related to either high-performance or hard real-time computing have shown that customization plays a major role to enhance the performance of applications while maintaining a reusable and flexible software architecture.

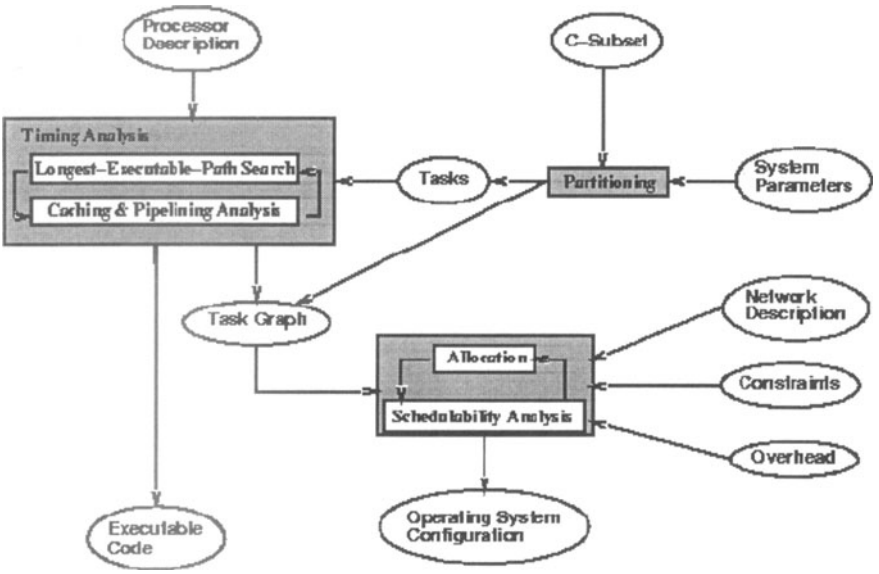

Figure 5. Conception of CHaRy

DReaMS [Di98] is a customizable library operating system intended to be used as a basis for the synthesis of either application-specific run-time platforms or operating system kernel in both fields (see Figure 6). Zero-DReaMS, the minimal core set of the system software implements only functions necessary to start application code on a native processor. All other services (including mechanisms for dynamic re-configuration and extensions) are optional features which can be integrated a priori

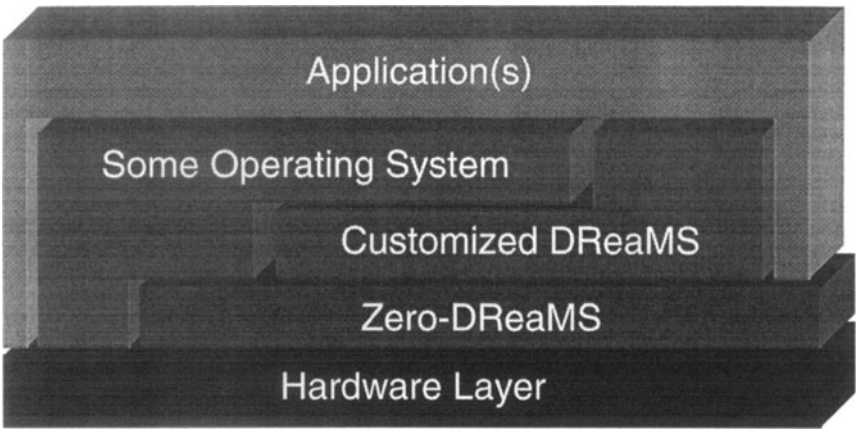

Figure 6. DReaMS overview

during the off-line configuration phase. By this way dedicated (RT) run-time platforms can be constructed in a predictable manner. Applications retain direct access to the hardware where required and system overhead is reduced to a minimum.

That is, only functionality which is really needed in the context in view is inserted. This removes redundancies and minimizes the system size. Beside the operating system functionality optimization can be applied to the communication system. This is brought into the PARADISE Design Environment by the TEReCS tool depicted in Figure 7.

Within TEReCS we intend to provide a tool for embedded RT communication systems. This toolkit should establish a design methodology for the construction of the software needed for the communication between applications on a distributed embedded controller environment. The main aspects persecuted are the automatic software generation using a library containing SW-services. Beginning from the specification of the communication connections through the definition of the SW-services for the library, the selection of needed services for each node, up to the verification of having enough resources and meeting all deadlines, each step will be supported by a task of TEReCS.

By this short overview we want to show that in the PARADISE Design Environment all core competencies needed for the design of today's ES are available in concept and automation tools. Table 1 gives a relation between a given core competence and the

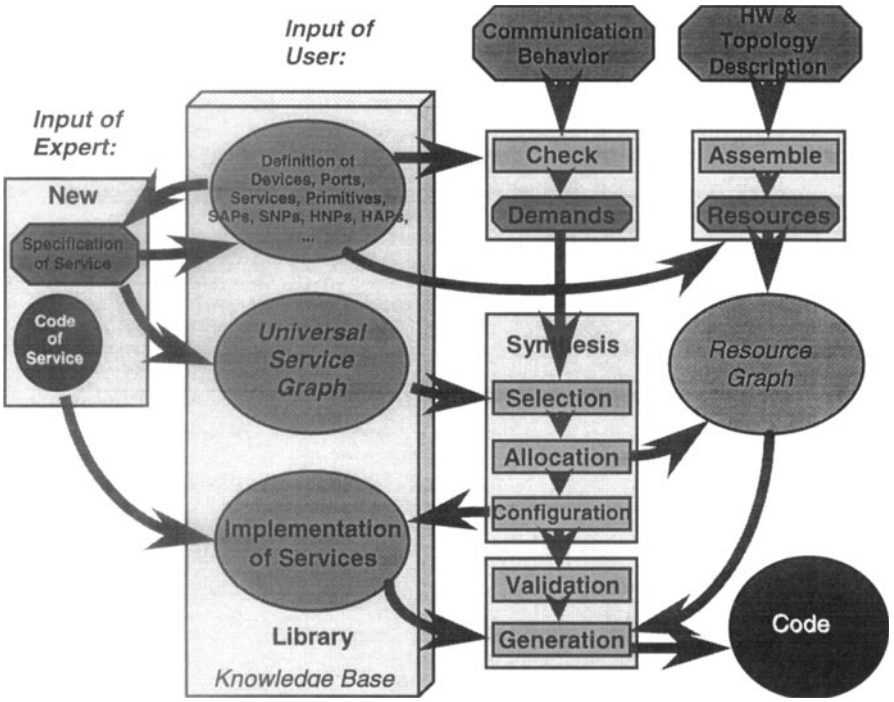

Figure 7. Conception of communication system design tool TEReCS

corresponding automation tool. It points out that different tools may be used within the same core competence, e.g. for analysis. This is due to the different levels of abstraction a core competence is structured by. The integration of tools is by now based on file-exchange.

	specification	modeling	SW-synthesis	operating system	HW-synthesis	rapid prototyping	verification	analysis
ASM-WB		X						X
SEA	X						X	X
PMOSS					X	X		
ChaRy			X					X
DreaMS				X				
TEReCS				X				

Table 1. Tools related to core competencies in the **PARADISE DESIGN ENVIRONMENT**

4 Integration

The mentioned tools are available within the PARADISE Design Environment. For an automated design path the question of integration comes up. We decided not to build up a tightly coupled framework because of the overhead caused by consistency checks, data base reaction time etc. Within PARADISE three types of integration with different degrees of coupling are distinguished:

1. **Shared data structures**: This is the strongest coupling. All algorithms are implemented on the same data structure which provides a very fast data exchange between algorithms. All tools within PARADISE use for their own algorithms unique data structures.

2. **Standardized exchange formats**: The definition of data exchange formats is a very commonly used manner to couple tools together. The complexity and expressiveness of such formats differ in wide ranges. We have found that data exchange formats must be adapted to the level of abstraction they refer to. On algorithmic level, e.g. after HW/SW-partitioning, the programming language 'C' has been found a very good formalism. Within the PARADISE Design Environment SEA and PMOSS are using this format for data exchange.

3. **Propagation**: On each design level and within each design step the design can be described by it's characteristics. The next task can be specified by

the requirements the design has to meet after performing the design task. This information has to be shared by several tools. Therefore we suggest to provide propagation mechanisms for design characteristics as well as for design rewirements.

Based on these integration concept a powerful design path can be provided. Each tool can be used autonomously as well as in combination with the other tools involved in PARADISE.

5 Conclusion

The integration of several core competencies is essential for establishing a methodological design process for parallel distributed embedded RT systems. in this paper we presented a overall structuring of the design process as well as an overview over the PARADISE Design Environment. Our activities are based on wide, theoretically well-founded concepts and tool support is built upon these theoretically foundations.

Further tasks needed within the PARADISE Design Environment should deal with visualization of results obtained from the different core competencies as well as from different levels of abstractions. This will help the designer to get a more global view of the design process because all results are brought together. Also a direct integration of the used concepts could be helpful for optimizing data exchange and tool performance.

6 References

[Di98] C. Ditze. Constructing a customizable library to support software synthesis for embedded applications and micro-kernel operating systems. In *Proc. of the 8. ACM SIGOPS European Workshop*, Sintra, Portugal, September 1998.

[Gur95] Y. Gurevich. Evolving Algebras 1993: Lipari Guide. Börger, Editor, *Specification and Validation Methods*. Oxford University Press, 1995.

[HaRo97] W. Hardt, W. Rosenstiel. Prototyping of Tightly Coupled Hardware/Software-Systems. *Design Automation for Embedded Systems*, vol. 2, no. 1 (1997).

[KlKlTa96] B. Kleinjohann, E. Kleinjohann, J. Tacken. "The SEA Language for System Engineering and Animation", *In Applications and Theory of Petri Nets*, LNCS 1091, Springer Verlag, 1996.

[Ra89] Franz J. Rammig. *Systematischer Entwurf digitaler Systeme*. B. G. Teubner, Stuttgart, 1989.