An Approach for Integrated Specification and Design of Real-Time Systems

Y. Tanurhan S. Schmerler H.-P. Gölz K. D. Müller-Glaser

Forschungszentrum Informatik Karlsruhe
Dept. of Electronic Systems and Microsystems
FZI/ESM
Haid-und-Neu-Str. 10-14, D-76131 Karlsruhe, Germany
http://www.fzi.de/esm/esm.html
email: esm@fzi.de

Abstract

In this paper, a design methodology for the design of microelectronic systems which includes hardware and software for open-loop and closed-loop control will be presented. An integrated approach to specification and design, analysis and simulation of the overall system has been developed. This provides for a systematic, computer aided approach to requirements definition, specification and design as well as verification and validation of the results. As embedded systems often require real-time capabilities, the environment presented gives special consideration to these constraints. We give an example for concrete applications, which shows, how the design of real-time embedded systems is supported by the design methodology and environment.

1 Introduction

The design of embedded real-time systems, which includes hardware and software for open-loop and closed-loop control and sensor and actuator components, such as automotive Electronic Control Units (ECU) or On-Board Management Microsystems (OBMMS) requires the combination of expertise from different domains. Available methods and tools currently support certain parts of system design, such as the design of discrete open-loop control systems or continuous and discrete closed-loop control systems [18]. However, no integrated approach to specification and design with respect to real-time capabilities or to analysis and simulation of the overall system is available. This lack of a complete and consistent system description leads to integration problems that are often detected very late in the design cycle and therefore cause high redesign costs. The design of such complex and heterogeneous systems mandates a systematic, computer-aided approach to requirements definition, specification and design, as well as to verification and validation of the results.

The design methodology presented in this paper is called IRTISD (Integrated Real-TIme System Specification and Design Environment).

2 Specification of real-time systems

Real-time systems are typically electronic systems which demand special, fixed time-behaviour during execution. This means that the correctness of real-time systems does not only depend on a correct function but also on the delivery of the results at the correct time. The results have to be produced before a defined deadline, with other words, the real-time criterion has to be fulfilled. If this time-behaviour is crucial to the functioning of a whole system, the system is called a *hard* real-time system, if not we speak of a *soft* real-time criterion.

This paper deals with the specification of embedded real-time systems, i.e. systems which implement their functionality both through software and hardware-components. Normally, the software cannot be modified by the user. The number of embedded real-time systems has heavily increased during the last few years due to falling prices for micro processors and DSP´s and growth of the typical system complexity of embedded systems in industrial automation, telecommunication and aerospace.

Rising hardware-performance and higher computing power does not provide a solution to the problem of fulfilling the hard real-time criterion, as recent research results in the area of embedded systems have shown [13]. Increased computing power is used to increase system complexity rather than for the improvement of timing

EURO-DAC '96 with EURO-VHDL '96
0-89791-848-7/96 $4.00

behaviour. An improvement in this area can be achieved by systematic modifications to the scheduling concept and resource management.
More importantly, such improvements can be achieved through:

- The use of formal methods for the specification and verification of the design.
- The inclusion of programming languages and design tools especially for code generation for special target systems resulting from a formal specification.
- The development and use of operating systems and net management methods combined with a real-time-capable communication protocol to ensure predictable operations in a complex and non-predictable environment.
- The planning and use of fault-tolerance to increase reliability and ensure the timing behaviour of real-time systems.

This paper presents an integrated design and specification environment which fulfils the needs and constraints of embedded real-time systems described above. The following chapters describe details of the environment, the design methodology and example applications that have been designed with IRTISD. Related work [17] has been concerned with a design methodology for hybrid electronic/ mechanical systems but uses similar strategies as developed within IRTISD.

3 Overall strategy

The requirements and specification are defined as those stages in the system design cycle which precede the design of microsystem architectures, the implementation of system components, integration and test. These phases include the formulation and analysis of functional and behavioral requirements to the proposed system, as well as non-functional requirements and constraints that have to be met and considered.

The design process of complex, heterogeneous systems can be organized into a hierarchy of specification, design, implementation and integration processes at different levels of abstraction. Such systems consist of analog and digital hardware/software of different abstraction levels, as well as non-electronic components, such as actuators and sensors.

At the systems and subsystems level, implementation is replaced by a lower-level development process. At the component level, technology specific design processes are used to implement the components. Having successfully finished the implementation process at one level, the results are integrated at the next higher level up to the system level. Within this process, specification, design, verification and validation are performed at each level and therefore accompany the overall system design process. The definition of requirements at system level represents the initial, interactive process of formulating the customer's needs, as well as the technology-specific restrictions and common technical and economical restrictions. Feasibility studies are performed to validate the requirements. The formulation, validation and modification of requirements typically results in a specification document which includes a collection of formal and informal, functional and non-functional requirements, goals, constraints and marginal conditions.

The IRTISD methodology is based on three major approaches [1], [15]:

- Systematic acquisition of requirements documents
- Management of non-functional requirements that cannot be formalized based on coarse-granular data models
- Development, analysis and simulation of executable models for functional and behavioral specification with step-by-step refinement into actual component design

Embedded systems are characterized by a tight coupling with the usually time and value-continuous system environment (fig 1).

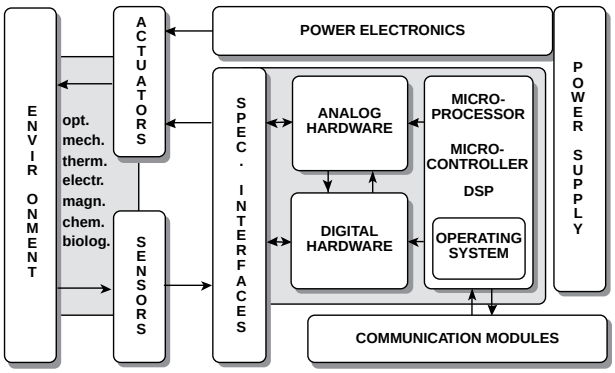

Figure 1. Structure of embedded systems

Specification techniques currently available, such as Structured Analysis for Real-Time Systems (SA/RT) [2], [3], Statecharts [4], SpecCharts [5], Specification and Description Language SDL [6], [7], [8], Hardware Description Languages [9] and object-oriented methods [10] concentrate mainly on functional and behavioural specification. Current work includes new insights derived from intensive usage of SDL for early system validation [11] in the field of aerospace technology. We use a specification model based on a combination of structured techniques similar to the SA/RT approach. It uses Statecharts for the reactive, open-loop parts of the behavioral specification, which are complemented by the use of mathematical specifications especially control design block diagrams, module generators or high-level modelling in VHDL and C for the description of SA/RT-processes. The SA/RT model can be partitioned and connected by models based on queuing theory, in order to analyse the system performance.

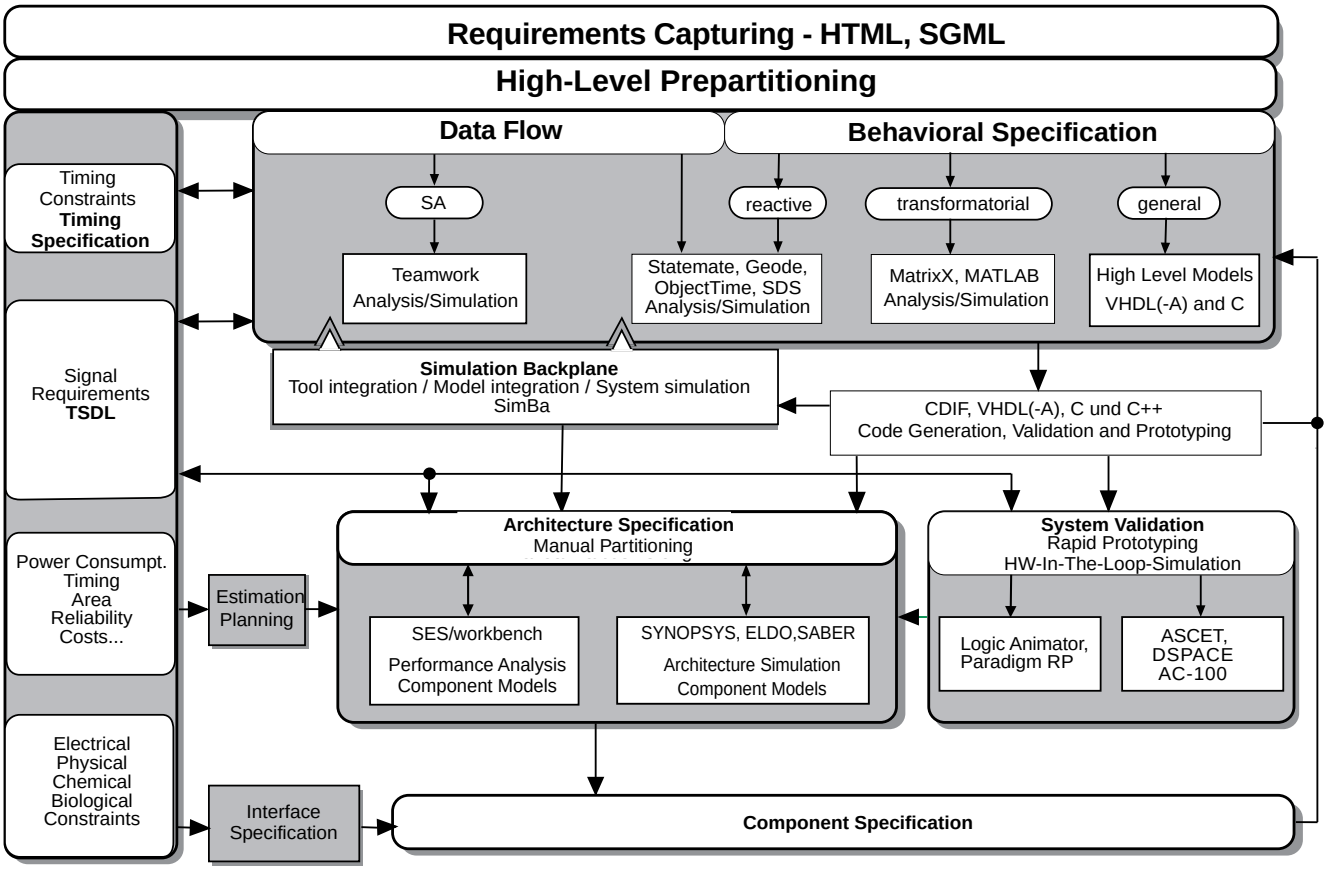

Figure 2. IRTISD-system

Fig. 2 shows the structure of the overall tool environment within IRTISD. Tool couplings in the area of cosimulation have been implemented between tools for description of time-discrete models (for instance Statemate) and of continuous systems (for instance MatrixX) on several levels. There are two mechanisms for code integration which is used for code generated by one tool communicating with the internal simulator of the other tool. This is a very convenient and, more than that, a quick way to get a coupled simulation. Code integration is a procedure performed automatically by the environment, which includes code modification and linking to executable programs. These are started in the background for communication with the running simulator. Besides code integration, direct couplings between Statemate and MatrixX have been implemented. Using these coupling mechanisms, system verification and validation can be performed in an interactive way - either by using the interactive simulator of Statemate or a graphical panel as frontend to a standalone program which resulted from code integration. The integration of continuous sub-models into the Activity and Statechart hierarchy allows for easy and intuitive design of heterogeneous systems. The management and handling of design and simulation data is the task of the framework and allows the user to restore former simulation runs and the corresponding design data at any time. Version management, code generation and code integration are performed by the tool servers of the integrated tools automatically. For tool data exchange, the standardized Format CDIF (Case Data Interchange Format) was chosen to achieve a tool independent exchange format [11].

4 Design process within IRTISD

The following example demonstrates the application of the different methods described above, in order to support the requirements description and specification of a system. The specification process using IRTISD includes:

- The requirements description:
  The acquisition of the requirements description is achieved using a hypertext-based frontend (HTML and SGML, see fig. 2). A predefined specification template for automotive control units is used. All functional,

behavioral and informal requirements and constraints are entered via text and table editors within the hierarchical HTML-pages. A table-based acquisition supports structured and traceable descriptions of the different functional requirements and the latest test procedures. This information is used to create checklists for the following formal specification of the system model and the test procedures.

- The formal interface and environment definition:
  A complete system specification must include the system´s interfaces and, if the system specification is to be simulated, (which is strongly recommended in IRTISD), simulation models of the environment. The system is described by a data flow diagram called context diagram. The context diagram represents the top-level system description which includes the environmental process, the information flow and the system interface. As the environment shows up as processes in the data flow model, these processes can be filled with the models of the environment when needed.
- The refinement in steps and formal control specification: The functional specification of the system to be designed is refined by a hierarchy of data flow descriptions. Each function described in the requirements description is designated by a black-box and a number of interconnecting data and control flows. Existing models are also represented as processes and are linked to the corresponding black box. System control is defined at each level of the data flow model hierarchy (as in SA/RT). Every control process is described using the Statechart formalism: in particular concurrent and hierarchical state machines are used. The activation/deactivation of processes is described formally and related to transitions in the state machine model.
- The formal process specification: Algorithms for feedback control or signal pressing are described using transfer functions and other elements known from classical and modern control design. As this usually requires a tool change (in this case Statemate to MatrixX), the interfaces are transferred into the control design tool.
- The simulation and prototyping: Using a cross-tool simulation the system behaviour is simulated. Alternatively, code can be generated by the participating tools and can be integrated automatically. This provides a software prototype which can be used to validate the system behaviour outside the simulators, especially in an real-time environment.
- The performance analysis: The software prototype is brought into the environment which provides for statistical analysis. Here, specific loads, dataflows or different communication protocols for implementation can be scrutinized. If the needed performance level is not achieved, the lower-level requirements are changed or sharpened in the hypertext requirements specification and the cycle is repeated.
- The code is generated in a language which leads to component development: VHDL on the digital hardware side, VHDL-A [15] on the analog hardware side, and C or ADA on the software side.

5 Example application

The following example demonstrates different methods which support the requirements description and specification of an embedded system. It also demonstrates the complete, integrated specification process of a typical automotive control unit based on a microcontroller. In cars being developed currently these control units are increasingly used for controlling different mechanical parts in automobiles such as motor management or transmission control.

The basic requirements for automotive control systems deal with multiple fields in system development such as hard real-time requirements for the operating system (i.e. injection control), continuous-time simulation (integrated RLC-networks), discrete-event simulation (microcontroller), network management and protocols (CAN), and robustness.

The ECU controls the engine by generating the pulses for the injection valves (IV1-4) and by sending the start signal to the electronic fuel pump (EFP) at the correct time. The ECU gets information about the state of the engine from different sensors. These sensors provide information about air temperature (TA), cooling water temperature (TCW), camshaft rotation (CAS) and crankshaft rotation (CRS). The results of the air mass measurement (AMM) are dependent on the mechanical position of the throttle valve (TV). CL15 indicates the on/off position of the ignition key (see fig. 3)

The ECU itself consists of an SAB80C166 microcontroller, which is connected via a local bus system to the I/O-Interfaces, to the CAN-controller and to RAM and ROM memory units. The software which runs on the SAB80C166 is the OSEK [16] operating system. OSEK is a configurable real-time operating system. It contains fixed and application-specific modules, e.g. for time- event- or interrupt-handling. The application itself is described in a variable number of tasks with different priorities.

The functional description of the electronic controller unit (see fig. 3) corresponds to the development methodology used in IRTISD. The specification starts with the definition of the highest level, in this case the electronic controller unit ECU. Then the design is refined layer after layer until the finest granularity is reached. This finest

granularity consists of basic components. Thus, the design methodology described in chapter 3 was applied.

At the first sub-level, the described ECU is divided into the following components: a microcontroller 80C166 to perform the basic computation, memory units, I/O-units, and a CAN controller to communicate with other ECU's in the automotive system.

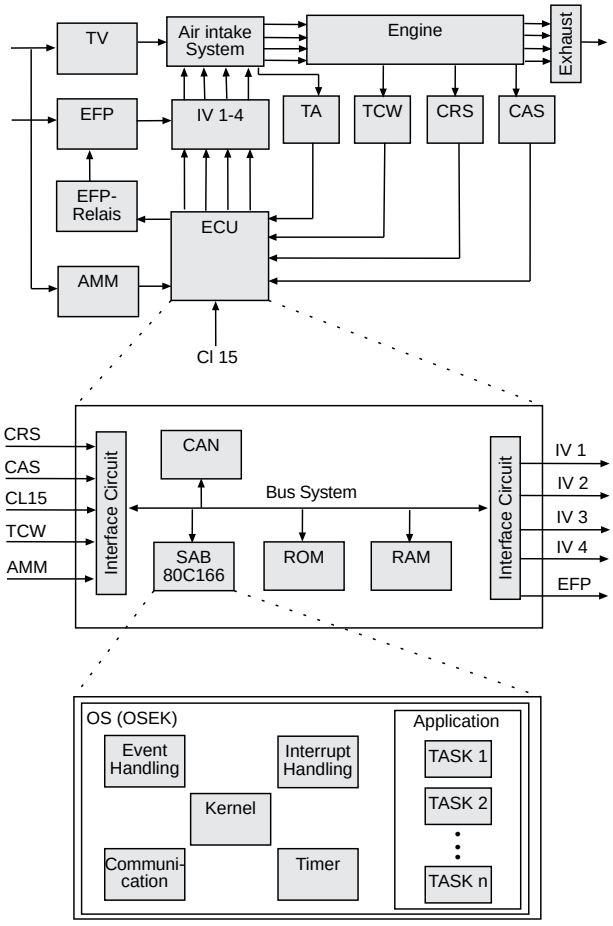

Figure 5. Functional components of an ECU

The microcontroller contains the operating system OSEK, which is refined into the parts concerning the functionality of the application, the different algorithms that control the engine.

These controlling algorithms have also been defined in the IRTISD environment by using the MATRIXx/System Build Simulator. The verified model was exported by MATRIXx via the C-code-interface. To obtain the OSEK-specific application format, the provided scheduler was removed from the generated code and the sub-models were extracted. To verify the extracted motor-management model, the simulation results from IRTISD were compared with the results from the simulation of the original model.

The ECU example shows the two different, but corresponding ways to go ahead with the system specification: the informal (conceptional) method and the formal method. The informal specification of the ECU uses textual and graphical methods to define the requirements, the functional decomposition and the interfaces between the components themselves and with the systems environment.The structure and contents of the informal specification can be transformed into a formal specification after an element has been described informally.

In addition to the top-down-methodology described above, IRTISD provides a bottom-up strategy for entering the specification process at the component level. In the ECU example the CAN unit has been developed in the past, and now it is being reused in the current design. So IRTISD combines the top-down and bottom-up methods to a meet-in-the-middle strategy.

The OSEK real time operating system [16], developed in a joint Project of the automotive industry, was modelled within the IRTISD environment. The OSEK standard specifies the kernel of the operating system including time-, interrupt- and event-handling. In order to integrate OSEK into the environment, its functionality was also modelled in Statemate. Now any application for OSEK can be developed and verified in IRTISD. An additional advantage of the OSEK-integration is the possibility of monitoring the operating systems behaviour during run time in order to get information about the real-time behaviour of the application.

The communication between different electronic controller units is handled by a CAN controller located within each unit. This CAN Controller was modelled in Statemate using the IRTISD methodology. For validation purposes, this simulation model was run on an FPGA- based logic emulation system. Then, the modelled controller was verified by connecting it to the emulated controller and to a integrated CAN-controller. In order to demonstrate the various coupling mechanisms, the software model of the CAN-controller was connected to real existing components and to emulated hardware components.

The environment MESA To verify the idea and basic concept of IRTISD, we implemented the design-, specification- and simulation environment for microelectronic systems MESA[1] in cooperation with the German automotive manufacturers BMW, Daimler-Benz, Porsche and VW and their electronic deliverers Bosch, Hella, Siemens and VDO within the MSR consort. The design methodology of IRTISD used in MESA allows an exchange of models under development between manufacturer and deliverer without giving away Know-How - but on the other side the sub-models of all partners were available and thus, cosimu-

1. See http://www.fzi.de/esm/projects/msr/msr.html

lation of the *whole* system under development is possible. Current research results in the area of Hardware/Software Codesign have been taken into consideration during the implementation of the Rapid Prototyping-interfaces to and from IRTISD [19], [20]. Thus, special interfaces to Rapid Prototyping-environments have been provided with the aim of running code generated within MESA on several commercial RP-systems. It can also be downloaded onto platforms running real-time operating systems like OSEK. Rapid Prototyping and -Verification has been performed in this manner for several ECU's on SAB80C166-platforms and the real-time operating system OSEK. This method has proven to be exact and a good support for ECU-designers in the automotive sector. MESA has been used successfully in several pilot projects between the members of MSR in the last year.

6 Conclusions and future work

A design methodology for real-time embedded systems has been developed. At the example of an electronic control unit (ECU) in an automotive system this methodology has proven to be useful for designers and especially meets the requirements for the specification of embedded real-time systems. An important improvement can be seen in the systematic discovery of errors or incomplete specifications in the early design phases. The IRTISD-environment as an implementation of the described methodology has been successfully used in several industrial projects in the automotive sector.

Future work will be concentrated on several subjects. The integrated HTML-based specification environment will be improved to support the designer additionally in the early design phases.

The integrated object-oriented specification and modelling language with the internal name ObjectCharts will be improved and systematically integrated into the design environment.

At last, the current work in the area of Hardware/Software-Codesign and Simulation Backplane [12] technology will be continued.

References

[1] K.D. Müller-Glaser, J. Bortolazzi, "An Approach to Computer Aided Specification" in *IEEE Journal of Solid-State Circuits*, Vol. 25, No.2 April 1990, pp. 335-345

[2] D.J. Hatley, and I.A. Pirbhai, "Strategies for Real-Time Specification", Dorset House Publishing, New York 1988

[3] T. DeMarco, "Structured Analysis and System Specification", Prentice-Hall, Englewood Cliffs, NJ, 1979

[4] D. Harel, A. Pnueli, J.P. Schmidt, and R. Sherman, "On the formal Semantics of Statecharts", in *Proc. 2nd Symp. on Logic in Computer Science* 1987, pp.54-59

[5] S. Narayan, F. Vahid, and D.D. Gajski, "System Specification and Synthesis with the SpecCharts Language", in *Proc. of ICCAD* 1991, pp. 266-269

[6] R. Saracco, J.R.W. Smith, and R. Reed, "Telecommunications Systems Engineering using SDL", North-Holland, Amsterdam 1989

[7] R. Gerlich, T. Stingl et al., "Experiences with an Extended SDL Environment", *ESTEC Workshop on Systems Engineering*, Nordwijk, 1995

[8] CCITT Recommendation Z.100, "Specification and Description Language SDL"

[9] VHDL System Design, The VHDL Consulting Group, Version 2.1, 1991

[10] J. Rumbaugh, P. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, "Object-Oriented Modelling and Design", Prentice-Hall, Englewood Cliffs, NJ, 1991

[11] J. Ernst, and K.D. Müller-Glaser, "Using The 1994 CDIF Standard As Data Exchange Format for Early Design Phases in Automotive Electronics", Technical Report FZI/ESM-Computer Science Research Center, 1995

[12] S. Schmerler, Y. Tanurhan, K.D. Müller-Glaser, "A Backplane Approach for Cosimulation in High-Level System Specification Environments", *Proc. of the 1995 European Design Automation Conference*, 1995

[13] Sang H. Son, Ed., "Advances In Real-Time Systems", Prentice Hall, Englewood Cliffs, 1995

[14] K. D. Müller-Glaser, J. Bortolazzi, Y. Tanurhan, "Towards a Requirements Definition, Specification and System Design Environment", *Proc. of the 1992 European Design Automation Conference*, 1992

[15] E. Sax, Y. Tanurhan, K.D. Müller-Glaser, "Integrated Design Process Support with VHDL-A", *Proc. of the 1995 EUROSIM Congress*, 1995

[16] OSEK consort,, "The OSEK Operating System - Application Program Interface", 1995

[17] A. Smailagic,D.P. Siewiorek, D. Anderson, et al., "Benchmarking an Interdisciplinary Concurrent Desin Methodology for Electronic/Mechanical Systems", *Proc. of the 32nd Design Automation Conference*, 1995

[18] J.-P. Soininen, T. Huttunen, K. Tiensyrjä, and H. Heusala., "Cosimulation of Real-Time Control Systems", *Proc. of the 1995 European Design Automation Conference*, 1995

[19] Th. Benner, R. Ernst, and A. Österling, "Scalable Performance Scheduling for Hardware-Software Cosynthesis", *Proc. of the 1995 European Design Automation Conference*, 1995

[20] G. Koch, U. Kebschull, W. Rosenstiel, "A Prototyping Environment for Hardware/Software Codesign in the COBRA Project", *3rd International Workshop on Hardware/Software Codesign*, 1994