THE SPECIFICATION LANGUAGE SPECC WITHIN THE PARADISE DESIGN ENVIRONMENT

A. Rettberg[1] & F. J. Rammig[1] &
A. Gerstlauer[2] & D. D. Gajski[2] &
W. Hardt[3] & B. Kleinjohann[1]

[1]Department of Computer Science,
University Paderborn/C-LAB, Germany
E-mail: {achcad, franz, bernd} @c-lab.de
[2]Center for Embedded Computer Systems,
University of California at Irvine, USA
E-mail: {gerstlauer, gajski} @cecs.uci.edu
[3]Department of Computer Science (IPL),
University Paderborn, Germany
E-mail: {hardt} @upb.de

The design of embedded systems has to address several interacting design aspects, so-called dimensions, to capture parallelism, distribution over different locations and hard real-time requirements. Thus, a structured design process has been established with the PARADISE design environment. The design process covers all steps from behavioral specification to final chip realization. In this paper, we describe how system specification and refinement is covered in combination with the processes available in PARADISE. An example of an adequate specification and modeling language is considered and adapted for integration into PARADISE. First results show the feasibility of integrating the respective concepts.

1. Introduction

The design of embedded systems (ES) today has to address several interacting design dimensions to implement parallelism, distribution over different locations, and hard real-time (RT) requirements. Consequently, the modern, structured design process has to start with a system specification, and it has to deal with heterogeneous requirements and restrictions. Especially the integration of dedicated flows for HW-design and SW-design as well as RT-operating system issues into the ES design flow is a main challenge. The PARDISE[1] design environment provides tools for

[1] Design Environment for **PARA**llel, **DIS**tributed **E**mbedded real-time systems

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35409-5_23

B. Kleinjohann (ed.), *Architecture and Design of Distributed Embedded Systems*

behavioral specification, run-time analysis, and system synthesis. Additional needs are the evaluation of application-specific characteristics at the system specification level. PARADISE is an open design environment for parallel, distributed embedded real-time systems. The basis of this common design environment is a highly structured design view called P-chart.

This paper shows the extension of PARADISE with the specification language SpecC in order to support systematic refinement from a system specification. SpecC is a system-level design methodology and specification language developed at the University of California, Irvine [Ga+00].

Section 2 gives an overview of the PARADISE design environment followed by an overview of SpecC in Section 3. An example, which is implemented in SpecC, was chosen to demonstrate the extension of PARADISE (Section 4.). The extended design methodology is presented in Section 5. The results for the example are shown in Section 6. Finally, Section 7 draws some conclusions and summarizes the paper.

2. PARADISE Design Environment

The PARADISE [HaReKl99] [Re+00] design environment combines different design dimensions. Design dimensions needed for establishing a design methodology for today's ES are:

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

Each design dimension covers all levels of abstraction. A very good basis for the structuring of the HW-design domain has been suggested by Gajski [Ga88]. The so-called Y-chart distinguishes a behavioral, structural, and a geometrical design view. Design views are applied to different abstraction levels. The Y-chart differentiates between five hierarchical layers: algorithmic, register-transfer, gate, symbolic layout, and electrical layout layers. For testability, a test view has been introduced as a fourth design view leading to the X-chart [Ra89]. The abstraction of this basic structure leads to the P-chart design view first presented in [HaReKl99]. The P-chart design view applies the X-structure to each design domain separately. Based on this abstract structure, domain-specific methods and tools can be integrated. The abstract P-chart structuring is illustrated in Figure 1.

The different layers of abstraction (algorithm to layout) of the Y-chart are depicted for each of the eight design dimensions. In addition, each dimension is structured by the four views of the X-chart. Based on this concept, a variety of automation tools for different design domains and levels have been integrated within the PARADISE design environment. Thus, PARADISE can be understood as an

implementation of a general applicable, integrated design methodology for today's ES. A special Internet-based communication service allows remote access to each tool in a distributed environment [Astair99].

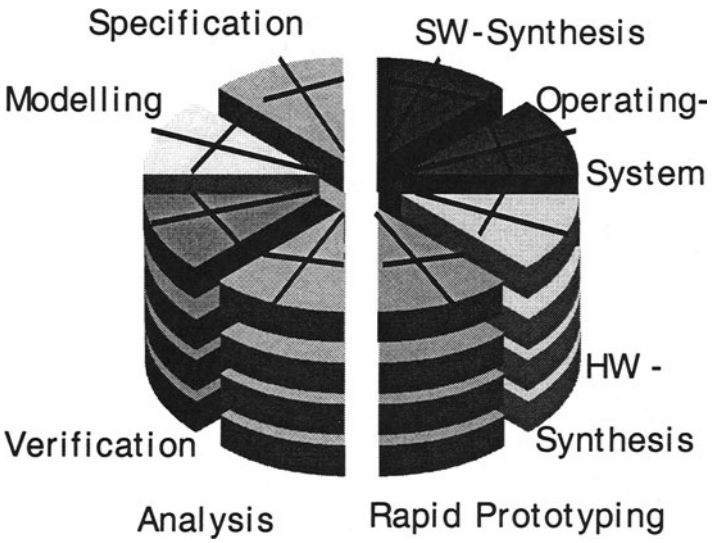

Figure 1. PARADISE design environment

3. SpecC Language and Methodology

Due to the increasing demand for analysis and evaluation of application specifications on the system level, the SpecC specification language has been introduced into the PARADISE design environment. SpecC allows to analyze functional aspects of a system level application specification through simulation or rapid prototyping in a very early design phase. The usage of SpecC within the PARADISE design environment accelerates the design process, which is important in order to keep today's time-to-market limits.

The SpecC specification language satisfies all the requirements for a codesign language and supports structural and behavioral hierarchy, concurrency, state transitions, exception handling, timing and synchronization in an explicit and orthogonal way. SpecC encourages reuse and supports integration of IPs. Since SpecC is a superset of ANSI-C, a large library of already existing algorithms can be used directly. A system design modeled in SpecC is executable, modular and complete. As a result, SpecC fulfills all of the requirements of a system specification language for the specification and modeling design dimensions on the different levels in PARADISE.

The SpecC-based design of complex systems, for example SOCs, is the process of implementing a desired functionality using a set of physical components. This process must begin with a specification of the desired functionality. The SpecC design methodology [Ga+00] starts with an executable specification as shown in Figure 2. This initial specification model describes the functionality as well as the performance, power, cost and other constraints of the design. The specification does not make any premature allusions to implementation details. During the specification of the desired functionality the designer has the ability to reuse existing code segments, functions or procedures by instantiating them out of an algorithm library.

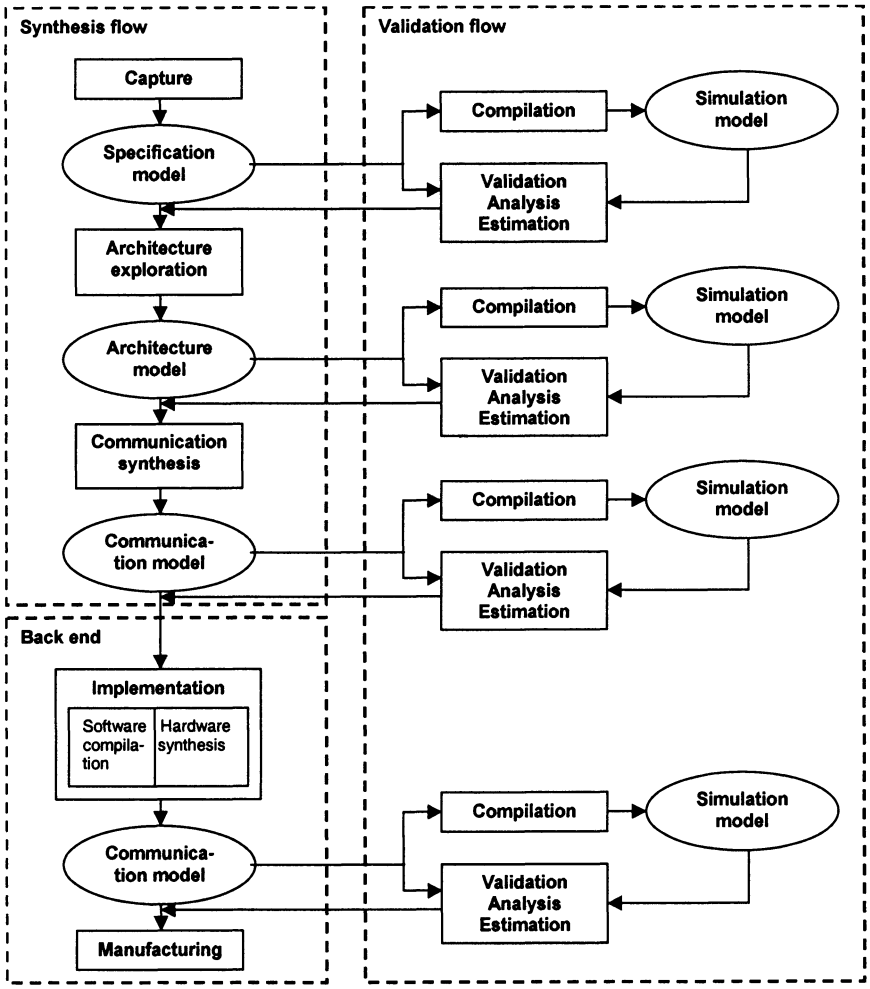

Figure 2. The SpecC methodology

The system-level synthesis flow of the SpecC design methodology consists of two major tasks: architecture exploration and communication synthesis. Through a series

of well-defined steps the initial specification is gradually mapped onto a target architecture. Architecture exploration, which refines the specification of the design into an architecture model, includes the design steps of allocation of processing components and busses, partitioning of behaviors, communication channels and variables, and scheduling.

The next step in the design flow is communication synthesis, which refines the abstract communication between behaviors in the architecture model into an implementation over the wires of system busses. The task of communication synthesis includes insertion of communication protocols, synthesis of interfaces and transducers, and inlining of protocols into synthesizable components. In the resulting communication model, the communication is described in terms of actual wires and timing relationships as described by bus protocols. The communication model, which is the resulting output from the system-level design process, describes the system design. It models the mapping of the specification onto components from the architecture model enriched by information of the communication structure and communication protocols.

The result of the synthesis flow is handed off to backend tools for compilation and high-level synthesis, as shown in the lower part of Figure 2 (back end flow).

In the following sections, we describe the integration of SpecC into the PARADISE design environment and the resulting new design methodology.

4. Example

To demonstrate the extension of the PARADISE design environment with SpecC, we choose an application example. The voice encoder/decoder (*vocoder*) which is implemented in SpecC is part of the European GSM standard for mobile telephone networks. The lossy codec scheme was originally developed by Nokia and the University of Sherbrooke [Ja97] and is based on widely used algorithms for speech encoding [Sa98]. The so-called Enhanced Full Rate (EFR) speech transcoding is standardized by the European Telecommunication Standards Institute (ETSI) as GSM 06.60 [ETSI96].

The GSM 06.60 standard for the EFR *vocoder* is accompanied by a bit-exact reference implementation of the *vocoder* functionality consisting of 13,000 lines of C code. This code describes the required functionality and was therefore used as the basis for the SpecC specification. At the top level the *vocoder* consists of independent coding decoding behaviors running in parallel. Encoding and decoding transform a stream of speech samples at a rate of 104 kbit/s into an encoded bit stream with a rate of 12.2 kbit/s, and vice versa. Coding is based on a segmentation of the incoming speech into frames of 160 samples corresponding to 20 ms of speech. For each speech frame the coder produces 244 encoded bits.

The SpecC block diagram of the encoding part is shown in Figure 3. Only the first levels of the behavior hierarchy of the encoding part are shown. All together, the SpecC description of the *vocoder* contains 43 leaf behaviors. At the top level, pre-filtering and framing, speech coding, and bit serialization run in a pipelined fashion. At the next level, the first step in the coding process is an extraction of linear-prediction filter parameters. Each frame is then further subdivided into subframes of

40 samples (5 ms). In two nested loops, open- and closed-loop analyses of pitch filter parameters and an exhaustive search of a predefined codebook are performed, followed by a filter memory update step. For a detailed information about the implementation of the *vocoder* in SpecC the reader is referred to [GZGH99]. The SpecC source code of the specification, architecture and communication models can be downloaded from the SpecC web page [SC].

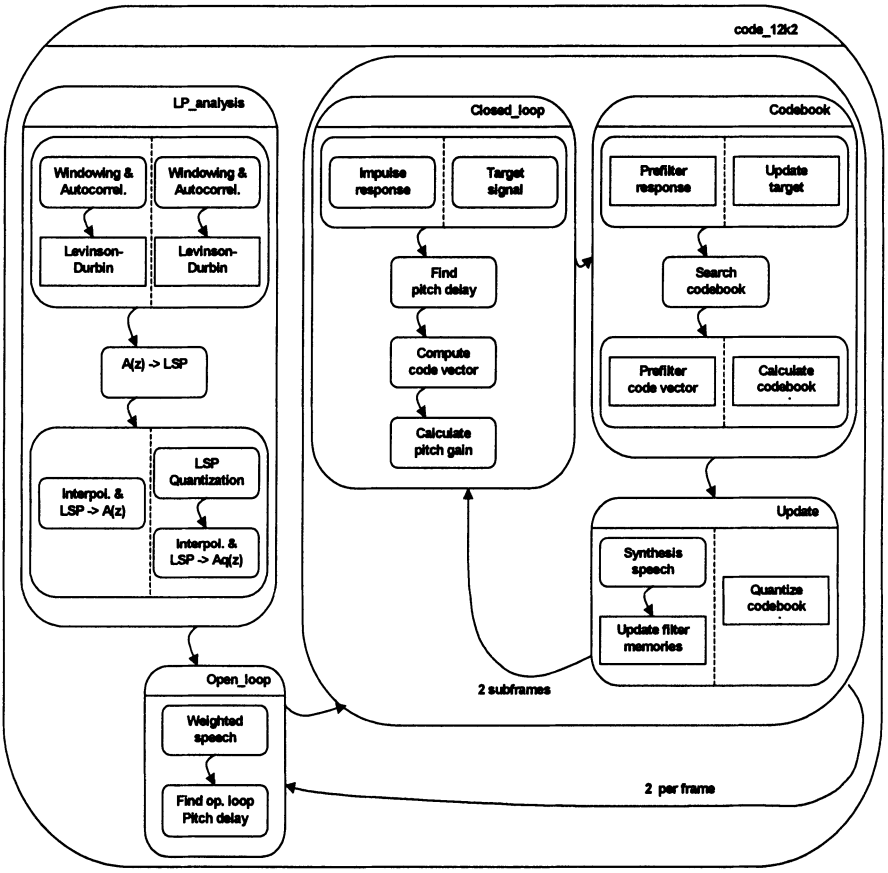

Figure 3. The *vocoder* encoding part

5. Design Methodology

The PARADISE design methodology is based on design dimensions and the structuring of each dimension by the P-chart (see section 2). SpecC introduces some new features to the design environment and the underlying methodology. Figure 4 shows the integration of SpecC into the P-chart based design process as

implemented within PARADISE. SpecC links the specification domain (left in Fig. 4) on the system level to the modeling dimension (right side in Fig. 4). The specification model is stepwise refined as presented in section 2. The designer of a complex ES compiles each SpecC model into an executable description. The simulation executable is used for prototyping and validation on the corresponding level. Once a model is validated, the designer passes it to the analysis design dimension within the PARADISE design environment. For example, the tool CHaRy[2] is used for timing analysis of the SpecC model in the analysis dimension, as depicted in the lower part of Fig.4.

CHaRy [Al96, Al97, StAl97] is a software synthesis tool for periodic controller applications. CHaRy allows to guarantee hard real-time conditions. Due to complexity reasons, CHaRy decomposes the overall problem of implementing periodic controllers on parallel embedded computers to the sub-problems partitioning, timing analysis, allocation, and schedulability analysis. Since all these sub-problems are of high complexity, CHaRy provides efficient heuristics for all these subjects. Hence, CHaRy supports the mapping of controller models to a number of tasks (partitioning), the extraction of their computation times (timing analysis), and the assignment of tasks to a processor network (allocation), such that all hard-real time conditions are guaranteed (schedulability analysis).

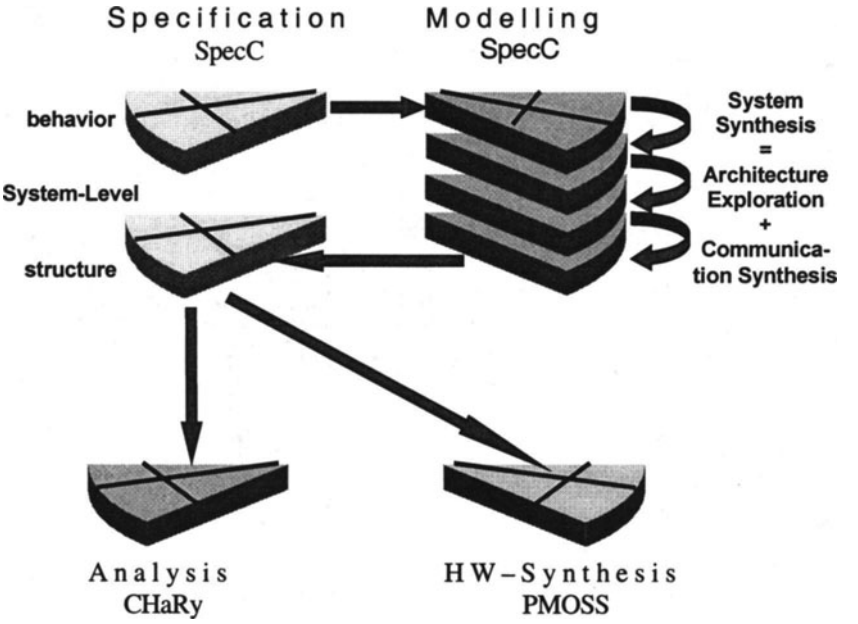

Figure 4. SpecC in PARADISE

The communication model of the SpecC design flow is the result of the system-level synthesis process, describing the structure of the system in terms of system components connected via system busses. High-level synthesis of the custom

[2] **C**-LAB **Hard** **R**eal-Time System

hardware components in the communication model is handled within the HW-synthesis design dimension. On the behavioral level, the PMOSS[3] system can be used. The PMOSS system is a powerful platform for high-level synthesis of embedded hardware out of a behavioral description. PMOSS divides the design process into several design tasks (e.g. HW/SW-partitioning, data-flow analysis and scheduling) which in turn may be subdivided into subtasks. Based on a well-structured database, at least one algorithm is available for each task or subtask which reads the database an writes all results back to the same database. Thus, subsequent tasks have immediate access to all the results. The same concept holds for data input which can be imported from different languages and for data output at the interface to commercial and public-domain tools for logic synthesis. For detailed information see [Ha95].

6. Results

In the following, we describe the results for some parts of the *vocoder* example. The *closed_loop* part of the *vocoder* model was analyzed with CHaRy. At the top-level the *closed_loop* is split into five different tasks (Figure 3): impulse response, target signal, pitch delay search, code vector computation and pitch gain calculation. CHaRy analyzed the worst-case execution time (WCET) for each of this task. The target architecture was a PowerPC. With CHaRy, we obtained the following WCET estimates for each individual task in terms of delay in µs and number of machine cycles as summarized in Table 1. The overall run-time for the *closed_loop* is 89854 µs and 8064786 cycles.

Task from *closed_loop*	**µs**	**cycles**
impulse response	4595	454261
target signal	11853	1085458
find pitch delay	60731	5432123
compute code vector	10747	923495
calculate pitch gain	1796	156312

Table1. WCET for *closed_loop* tasks

The *LP_analysis* (short term analysis) was analyzed with the high-level synthesis tool PMOSS. For our example, we use the high-level transformation and synthesis process from PMOSS. The SpecC description of the *LP_analysis* was transformed into a data-flow graph (DFG) and a control-data-flow graph (CDFG). Furthermore, PMOSS generates a controller and a datapath. The *LP_analysis* function is optimized by several high-level transformations. In fact, high-level design space exploration is enabled by the transformation task. Available high-level transformations include loop-unrolling, constant propagation, dead code elimination, elimination of temporary data elements as well as algebraic transformations. High-level transformations result in an optimized high-level description of the *LP_analysis*. All

[3] Paderborn **MO**dular System for High-Level **S**ynthesis and HW/SW-CodeSign

examined points of the design space can be visualized for feedback to the designer as transformation graph. The partitioned and optimized *LP_analysis* description can now be passed to the synthesis task, as shown in Figure 5. Within this task, the synthesis sub-tasks functional unit (FU) scheduling, FU allocation, FU binding, register (REG) allocation, REG binding, interconnection and finally netlist generation are performed. PMOSS provides several algorithms for each synthesis sub-task which allows to direct the optimizations. The DFG of the *LP_analysis* has 24 nodes and the CDFG has 128 nodes. The datapath contains 7 FU's, 7 registers and uses 13 multiplexers. The controller has 25 states and 57 transitions. The register transfer level netlist can be stored in different formats, for example VHDL.

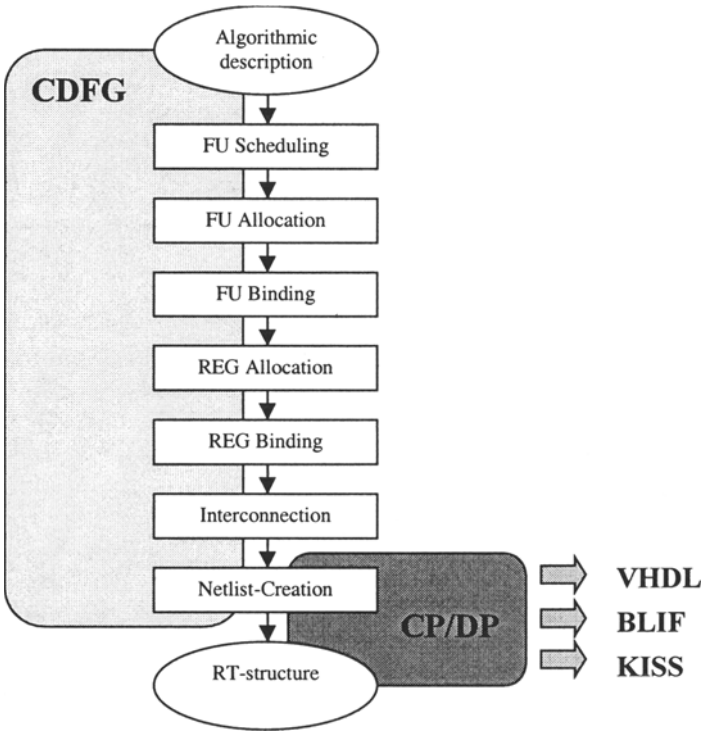

Figure 5. PMOSS synthesis process

7. Conclusion

In this paper we presented the integration of the specification language SpecC into the PARADISE design environment. SpecC fulfills all requirements for the design dimensions specification and modeling within PARADISE. The existing tools CHaRy and PMOSS are used for timing-analysis and high-level synthesis with a link to logic synthesis. Therefore, the integration of SpecC into PARADISE results in a closed design flow from system specification down to implementation. The results of

the *vocoder* example reflects the usability of the presented methodology and the design environment.

References

[Al96] P. Altenbernd: "Timing Analysis, Scheduling, and Allocation of Periodic Hard Real-Time Tasks". Dissertation, Paderborn, 1996.

[Al97] P. Altenbernd: "CHaRy: The C-LAB Hard Real-Time System to Support Mechatronical Design". In Proc. Of the International Conference on Engineering of Computer Based Systems (ECBS-97), Monterey, California, March 1997.

[ETSI96] European Telecommunication Standards Institute (ETSI), "Digital cellular telecommunications system; Enhanced Full Rate (EFR) speech transcoding (GSM 0.60)", Final Draft, November 1996.

[Astair99] http://www.c-lab.de/astair

[Ga88] D. D. Gajski: "Silicon Compilation". Addison Wesley Publishing Company, 1988.

[Ga+00] D. D. Gajski, J. Zhu, R. Dömer, A. Gerstlauer, S. Zhao: "SpecC Specification Language and Methodology". Kluwer Academic Publishers, 2000.

[GZGH99] A. Gerstlauer, S. Zhao, D.D. Gajski, A. Horak, "Design of a GSM Vocoder using SpecC Methodology", University of California, Irvine, Technical Report ICS-TR-99-11, Feburary 1999.

[Ha95]W. Hardt, "An Automated Approach to HW/SW-Codesign". In IEEE Colloquium: Partitioning in Hardware-Software Codesings. London, Great Britain, February 1995.

[HaReKl99] W. Hardt, A. Rettberg, B. Kleinjohann. "The PARADISE design environment". In Proc. of the 1st Embedded System Conference, Auckland (New Zealand), 1999.

[Ja97] K. Järvinen et. al., "GSM enhanced full rate speech codec". In Proceedings of ICASSP, pp. 771-774, 1997.

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

[Re+00] A. Rettberg, W. Hardt, J. Teich, M. Bednara. "Automated Design Space Exploration on System Level for Embedded Systems". In Proc. of the Ninth Annual International HDL Conference and Exhibition (HDL Conf. 2000), San Jose (USA), March 8 - 10, 2000.

[Sa98] R. Salambi et.al., "Design and description of CS-ACELP: a toll quality 8 kb/s speech coder", IEEE Transactions on Speech and Audio Processing, Vol. 6, No. 2, pp. 116-130, March 1998.

[StAl97] F. Stappert, P. Altenbernd: "Complete Worst-Case Execution Time Analysis of Straight-line Hard Real-Time Programs". in C-LAB External Report 27-97 Paderborn, 1997.

[SC] SpecC home page, http://www.cecs.uci.edu/~specc/