Embedded System Design 2.0: Rationale Behind a Textbook Revision

Peter Marwedel
TU Dortmund, Informatik 12
Dortmund, Germany

Michael Engel
TU Dortmund, Informatik 12
Dortmund, Germany

ABSTRACT

Seven years after its first release, it became necessary to publish a new edition of the author's text book on embedded system design. This paper explains the key changes that were incorporated into the second edition. These changes reflect seven years of teaching of the subject, with two courses every year. The rationale behind these changes can also be found in the paper. In this way, the paper also reflects changes in the area over time, while the area becomes more mature. The paper helps understanding why a particular topic is included in this curriculum for embedded system design and why a certain structure of the course is suggested.

General Terms

Embedded Systems

Keywords

Embedded systems, embedded system education, cyber-physical system, curriculum, text books

1. INTRODUCTION

Back in 2003, we published our first text book on embedded system design. The goal of the book was to suggest the standard level of knowledge about embedded system design for senior undergraduate students. In this way, **we want to propose a standard curriculum for embedded system design** and to support this curriculum with the necessary teaching material.

At the time of writing the first book, the books by W. Wolf [25] and F. Vahid [23] were the only text books on embedded system design which were discussing the principles of embedded system design. Therefore, pioneering work was required in order to define the standard knowledge. The knowledge finally selected was based on conference publications and journal papers. In 2007, a German translation of the English original was published. The German translation did already include some updates, like more detailed proofs.

WESE 2011 Taipei, Taiwan

In 2010, knowledge on embedded system design had advanced to a level where a new edition of the book became necessary. Various extensions were deemed appropriate. In a few cases, material was also dropped. Some of the chapters were completely reorganized. More emphasis was placed on cyber-physical systems, a term which did not exist when the first edition was written. The current paper explains the changes which were incorporated into the second edition. In this way, it provides an update on the list of topics which should be included —from the author's point of view— in a curriculum on embedded system design and it also explains how the topics can be sorted and linked. As result, a structure for embedded system education is suggested.

2. RELATED WORK

In our approach to teaching embedded system design, we want to focus on concepts and methods instead of particular tools. A key reason is that concepts and methods remain valid for a number of years, whereas tools may already be obsolete by the time the student graduates. Nevertheless, we demonstrate concepts and methods using exemplary tools. Many of the books on embedded system design cover the programming of small systems. They cover subjects such as programming with interrupts and memory maps. A treatment of the concepts of embedded system design is frequently lacking. Such books cannot be used for our teaching and we do not include them in this list of related work.

Concepts include computational models, real-time systems, fundamentals of hardware components, etc. Computational models are presented by S. Edwards [5]. A detailed in-depth coverage of models of computation (MoCs) is provided by A. Jantsch [8]. We found that this coverage is too advanced to be used in our undergraduate courses.

A book written by F. Vahid focuses on the implementation of finite state machines (FSMs) [23]. FSMs are also covered in a recent text book by D. Gajski [6].

Real-time systems are the subject of a well-known book by H. Kopetz [9] and its recent update [10]. G. Buttazzo published the standard reference book on real-time scheduling [3]. Also, Krishna and Shin [11] and Lui [14] published books in the area.

Guidelines concerning graduate embedded system education are listed in the Artist recommendations for embedded system curricula [4]. In general, the proceedings of the workshop on embedded system education (WESE) contain contributions toward embedded system education [18]. A description of the educational approach for embedded software in Taiwan is one of the outstanding contributions resulting

from the workshop [22]. Various top-level contributions to the area are included in a book edited by R. Zurawski [26].

Fundamentals of cyber-physical system design are covered in a recent book by Lee and Seshia [13]. In contrast to other books, it also comprises information on the physics of robots.

For our education of senior undergraduate students, the books listed above do not meet our goals. Consistent with the Artist guidelines, we want to provide a broad foundation for more detailed courses. A focus on real-time systems or finite state machines would be too narrow for our purpose. Unfortunately, requiring students to attend more than two or three courses related to embedded systems is not an option in our undergraduate education and we believe that this is a rather common situation for many CS and possibly also CE curricula. Considering this constraint, our approach is to offer one introductory course with a broad scope, optionally followed by one or more specialized courses, one of which could train practical skills [15]. How could a text book for such a broad scope look like? In the following, we will demonstrate how the topics covered in the first edition evolved over time.

3. CHANGES TO THE FIRST EDITION

3.1 Scope

In the first edition, embedded systems (ES) were defined as "*information processing systems embedded into enclosing products such as cars, telecommunication or fabrication equipment.*". This definition of embedded systems still applies. However, E. Lee wrote "*Embedded software is software integrated with physical processes. The technical problem is managing time and concurrency in computational systems*" [12]. Lee's definition places more emphasis on the integration with the physical environment and on the modeling of time. Also, the term "*cyber-physical systems*" has been introduced. Models of cyber-physical systems include a model of the physical environment as well as a model of information processing embedded into the environment:

Cyber-physical system (CPS) =
Information processing (ES) + physical environment

The description of embedded systems in the first edition still is a relevant description of information processing in cyber-physical systems. In this way, a curriculum on embedded system design covers all information processing aspects of cyber-physical systems and will provide the foundations for modeling such systems.

In 2003, no major distinction was made between mobile phones and other embedded systems. In the mean-time, mobile phones have evolved into smart phones and more general computing platforms. The distinction between advanced mobile phones and small laptops, such as tablet PCs, is blurring. Just like PCs, they do typically allow a connection to the internet. The performance of mobile platforms has also been increased. Even Gigahertz clock rates are now feasible for smart phones. In this way, phones have become more similar to general computing. However, consequences of the lack of high capacity batteries have become more drastic, which means that one of the constraints of embedded systems still applies. Also, mobile platforms have become much more important. Accordingly, we are keeping smart phones within the scope of the curriculum, but do distinguish more clearly between smart phones and other embedded platforms.

3.2 Structure of the second edition

Over time, the structure of the first edition became inappropriate. There was too much emphasis on particular languages and not enough on models, validation was lacking, as was the mapping of applications to platforms.

For the redesign, major attention was paid to improve the structure of the material to be presented. We decided to structure the topics according to a generic design flow. This design flow includes specifications, hardware components, system software, evaluation and validation, mapping of applications to platforms, optimizations and test as components as shown in fig. 1.

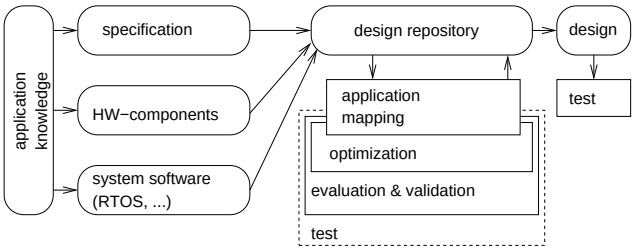

Figure 1: Simplified design flow

In this figure we assume that evaluation, validation, mapping to platforms and optimization are interleaved. In the general case, no strict sequence of these steps can be selected. The design flow is very general and allows specific design flows to be derived. Figure 2 demonstrates the new overall structure of the book and how this structure relates to the first edition.

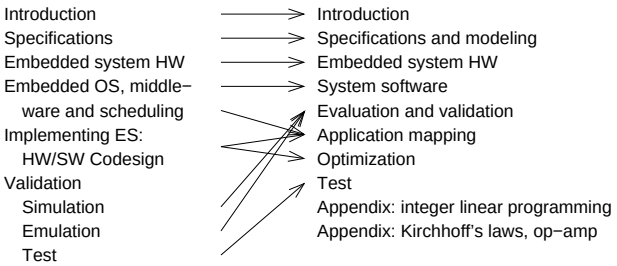

Figure 2: Overall structure of the book and the corresponding curriculum

Obviously, we will always start with an introduction. Next, the new edition contains a chapter on specifications and modeling, instead of just on specifications. This was done as models are required throughout the design, and not just at the very beginning.

The chapter on embedded hardware was kept (but extended, as will be explained below).

The next chapter is on embedded system software, not just on operating systems and middleware. This change reflects the fact that there may be system software other than just operating systems and middleware. In fact, even the first edition briefly covered real-time data bases.

A new chapter on evaluation and validation was introduced. This was done due to the increased body of knowl-

edge on the evaluation of real-time systems, which we considered to be very essential. Evaluation and validation are covered in the same chapter, since the two are frequently using similar techniques, such as simulation or emulation. Coverage of these two techniques was moved into the new chapter.

Also, a new chapter on the mapping of applications to execution platforms was added. We believe that this is a very important area and it deserves its own chapter. This chapter now comprises information about standard real-time scheduling techniques, mapping techniques from high-level synthesis, hardware-software partitioning techniques and specific new techniques. We believe that these techniques are very essential and should be included in all courses on the fundamentals of embedded system design.

Optimizations were moved into a separate chapter. This was done due to the thousands of optimization techniques for embedded systems. There is no hope that they can all be presented. Only an exemplary set of methods can be included. We consider this chapter as a "nice-to-have" feature. No serious harm is caused if this chapter cannot be included in a course. However, this chapter allows us to include some of our own research results in the text book.

Test has been moved into its own chapter in order not to overload the chapter on evaluation and validation. We consider this chapter as a "nice-to-have" feature as well and, again, no serious harm is caused if this chapter cannot be included in an embedded system course.

Next, let us look more closely at each of the chapters.

3.3 Specifications, modeling and languages

In chapter 2, the first edition contained a set of specification languages which we thought of being important for embedded system design. Models of computation (MoCs) were covered only briefly.

This was turned around in the second edition: models of computation are now the key element for structuring the chapter. Models based on finite state machines, on data flow, on discrete event systems and on von-Neumann languages are now covered as such. Specific languages are included as examples detailing the description of models of computation. Figure 3 demonstrates how this chapter has been reorganized.

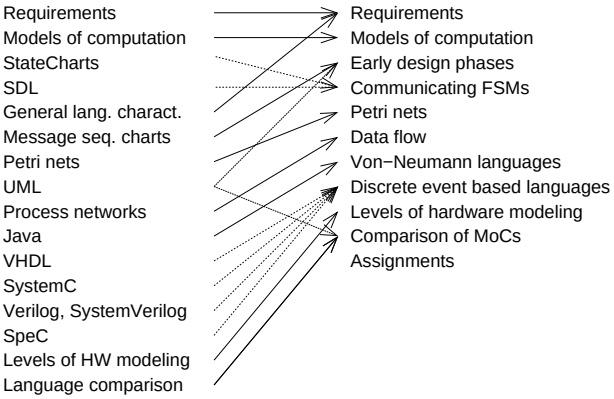

Figure 3: Transition from a focus on languages to a focus on MoCs

StateCharts and SDL became examples of languages based on finite state machines. Process networks were moved into a section on data flow models. VHDL, Verilog, SystemC, and SpeC are now covered in a section on discrete event based languages. Java (as well as CSP and Ada) became part of a section on von-Neumann languages. Message sequence charts were moved into a section on early design phases. UML is now covered in two ways. First, it provides examples of languages useful during early phases. Second, in a comparison of the different MoCs, we analyze the MoCs available in UML. This approach simplifies understanding the capabilities and limitations of UML. We believe that, with the new approach, it became easier to understand the similarities of languages such as VHDL, Verilog, SystemC, and SpecC. The same applies to imperative languages such as Java, CSP and Ada.

A table was been introduced in order to structure the presentation of models of computation (see table 1).

Communic./ Organiz. of components	Shared memory	Message passing	
		synchronous	asynchronous
Undefined components	Plain text or graphics, use cases	(Message) sequence charts	
Communicating finite state machines	StateCharts		SDL
Data flow	(not useful)		Kahn process networks, SDF
Petri nets		C/E nets, P/T nets, ...	
Discrete event (DE) model[1]	VHDL, Verilog SystemC	(Only experimental systems) (Distributed DE in Ptolemy)	
Von-Neumann model	C, C++, Java	C, C++, Java with libraries CSP, ADA	

Table 1: Overview of MoCs and languages considered

Furthermore, we took into account that specification languages are required beyond the initial specification. Therefore, the chapter now also refers to *modeling*. The idea is that any type of modeling, regardless of where it is used in the design process, is covered in the chapter.

3.4 Extension of the chapter on embedded system hardware

Embedded hardware was already covered in the first edition of the book. However, we found that key knowledge was frequently lacking among students, especially computer science students. Therefore, this chapter was significantly extended in the new edition.

First of all, the impact of timing predictability and energy-efficient designs is now given more attention in general. There-

[1]The classification of VHDL, Verilog and SystemC is based on the implementation of these languages in simulators. Message passing can be modeled in these languages "on top" of the simulation kernel.

fore, the impact of the need for power-efficient and timing-predictable designs is now frequently referred to in the text.

Another extension concerns the modeling of signals. Computer science students are typically unaware of digital signal processing. Therefore, the basic terms and definitions are now included in the book. Signals are defined as mappings from a time domain to a value domain.

Fig. 4 contains the hardware-related content which we consider being an essential part of any embedded system curriculum.

- Introduction
- Input
  - Sensors
  - Discretization of time
  - Discretization of values
- Processing units
  - ASICs
  - Processors
  - Reconfigurable logic
- Memories
- Communication
  - ...
  - Guaranteeing real−time behaviour
- Output
  - D/A−converters
  - Sampling theorem
  - Actuators
- Secure hardware
- Assignments

Figure 4: Hardware-related content

Our hardware-related content now includes a more detailed discussion of the conversion between analog and digital signals, partially due to the reference to cyber-physical systems. The first set of extensions concerns discretization and the sampling theorem (including A/D- and D/A-converters). It turned out that students had problems understanding simple circuits. Many of the discussions evolved along attempts to explain the behavior of circuits. We found that it became necessary to "remind" CS students of Kirchhoff's laws. A special appendix was added for this purpose. Also, understanding D/A-converters requires an understanding of op-amps. A short description (2.5 pages) of the relevant properties of op-amps was also added to the appendix. In this way, we used the appendix to ensure that students are aware of prerequisites which we found frequently lacking. We found that the level of detail that was added is absolutely sufficient.

Also, we found it very important that students understand limitations of the conversion from the digital domain back into the analog domain. However, a comprehensive presentation of this topic would require a detailed explanation of sampling theory, using Fourier transforms. This was clearly beyond the material which we wanted to include in a general embedded systems course. So, we were faced with the problem of developing a "short" introduction to sampling theory. We decided to use a special approach in which Fourier decomposition is explained with examples. Then, the restriction to linear systems is employed to motivate the independent consideration of each of the wave forms. Finally, it is explained how analog signals can be recovered from the digital samples and which limitations exist for such a recovery. Fig. 5 demonstrates the resulting difference between the original signal and the analog signal reconstructed from digital sequence values.

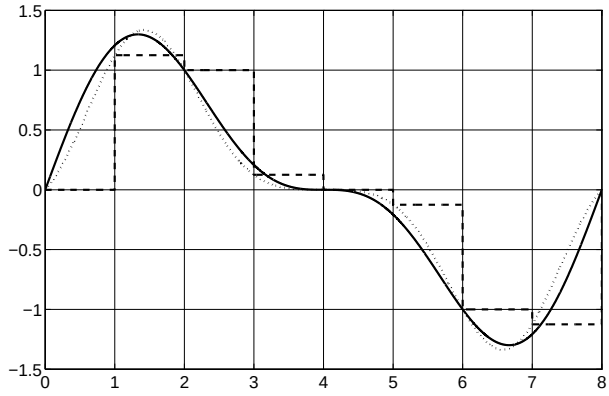

Figure 5: original (solid), at D/A output (dashed), reconstructed analog (dotted)

The essential limitations such as quantization noise, imperfect filters, and knowledge of only a finite segment of the signal, can be easily motivated with this approach. For example, fig. 5 assumes sampling at integer times. Due to quantization noise, original and reconstructed signal differ at times t =1, 3, 5, and 7, whereas the original signal can be exactly represented at times t =0, 2, 4, 6, and 8.

For communication, emphasis is on the real-time behavior. We believe that the need to guarantee real-time behavior is a key difference between general and embedded computing.

Our content includes a brief section on secure hardware. This section allows a more detailed coverage of the topic to be linked to our structure.

3.5 Redesign of the chapter on operating systems

The first edition of the book contained a chapter on embedded and real-time operating systems, real-time scheduling and a brief subsection on real-time data bases. Since the writing of the first edition, middleware gained increased interest. Moreover, we saw scheduling as a special case of mapping applications to execution platforms.

Therefore, this chapter was completely reorganized. The chapter was renamed into a chapter on system software. Middleware and real-time data bases are now very natural elements in the reorganized chapter. Resource access protocols such as the priority inheritance protocol remain an appropriate topic to be discussed in the context of operating systems. However, general real-time scheduling was moved into the chapter on the mapping of applications to execution platforms (see below). Fig. 6 describes the suggested system software related content.

The need for customizing embedded operating systems was emphasized and a brief subsection on virtualization was added.

The section on hardware abstraction layers is new and

- Embedded operating systems
  - General requirements (e.g. configurability)
  - Real−time operating systems
  - Virtual machines
  - Resource access protocols
- Example of an embedded operating system
- Hardware abstraction layers
- Middleware
- Real−time data bases
- Assignments

Figure 6: Operating system related content

the description of middleware has been extended, due to the increasing importance of communication middleware.

The section on real-time data bases is short. Its main purpose is to link a possible more detailed coverage of the topic to the suggested standard curriculum.

3.6 New chapter on evaluation and validation

In the original book, evaluation and validation were a minor topic. One of the reasons was the fact that the state of the art concerning the evaluation of embedded systems was still immature. In the mean-time, several evaluation techniques have been proposed. These techniques are considered to belong to the essential curriculum for embedded system design. Therefore, they are now included. The contents of this chapter are listed in fig. 7.

- Introduction
  - Scope
  - Multi−objective optimization
  - Relevant objectives
- Performance evaluation
  - Early phases
  - WCET estimation
  - Real−time calculus
- Energy and power models
- Thermal models
- Risk− and dependability analysis
- Simulation
- Rapid prototyping and emulation
- Formal verification
- Assignments

Figure 7: Contents concerning evaluation and validation

The need of considering several objectives for an evaluation of embedded systems has become more obvious. Therefore, the fundamentals of multi-objective design and Pareto-optimality are now given more room.

Regarding particular objectives, worst-case execution times (WCETs) are now given more emphasis. Their safe computation has become more mature. The technique proposed by R. Wilhelm et al. [24] as incorporated into the aiT timing analyzer is a standard reference. Implicit path enumeration evolved as a technique capable of avoiding the complexity of complete path enumeration.

Also considering worst-case delays, L. Thiele's real-time calculus (see, for example, [20]) is now regarded as a kind of basic calculus for evaluating real-time designs. The essence of this calculus should be known by everyone working in the area. It is therefore now included.

Reliability is another objective which is now covered in more detail in this chapter. This is motivated by the trend toward circuits which are inherently including faults, as predicted by the ITRS road map [7].

Brief sections on energy, power and thermal modeling have been added as well. We assume that these sections will be extended in future editions, as these issues are of increasing importance.

Formal verification is covered only briefly. This section puts formal verification into perspective and allows presenters to link this topic to the other topics.

3.7 New chapter on the mapping of applications to execution platforms

In embedded system design, the use of more or less standardized execution platforms is becoming more widespread. The increasing cost of semiconductor fabrication is one of the reasons for this. Therefore we came to the conclusion, that the mapping of applications to execution platforms should be given more emphasis. This is also in-line with the introduction of the new series of workshops on the mapping of applications to MPSoCs (see [16] for the most recent workshop). In the first edition of the book, the mapping of applications was included in a general chapter of implementing embedded system designs. We decided to distinguish between general mapping techniques and examples of optimization techniques. General mapping techniques are included in the content listed in fig. 8.

- Scheduling in real−time systems
  - Aperiodic and periodic scheduling, without and with precedence constraints
- Hardware/software partitioning
- Mapping to heterogeneous multi−processors
- Assignments

Figure 8: Mapping-related content

We consider standard scheduling techniques to be a special case of mapping techniques. They are mostly designed for single-processor or homogeneous multi-processor systems. For tasks with precedence constraints, we include techniques from high-level synthesis. Hardware/software partitioning algorithms consider heterogeneous systems, but usually they do not exploit global knowledge about schedulability. New, dedicated algorithms have been developed for the mapping to heterogeneous processors (e.g., DOL [21]). A survey of such algorithms is now included.

3.8 New chapter on optimizations

Optimization techniques were moved into a separate chapter further towards the end of the book. This way, their coverage becomes somewhat more optional and can be skipped if not enough time is available. This positioning of this material is motivated by the observation that we are seeing an increasing number of optimizations. Therefore, only examples can be covered. A comprehensive overview of optimizations for embedded systems design is neither feasible nor appro-

priate. Fig. 9 includes the list of exemplary optimizations which we have included in our book.

Task level concurrency management
High−level optimizations
Compilers for embedded systems
Power management and thermal management
Assignments

Figure 9: Exemplary coverage of optimizations

The new chapter is supplemented by a new appendix on integer linear programming. We found that many students are unaware of this technique and that a minimum knowledge regarding this technique is required for many optimizations and also for implicit path enumeration as covered in the discussion on computing worst case execution times.

3.9 Separate chapter on testing

Testing was moved out of the chapter on validation, since validation is now covered together with evaluation. A separate chapter on testing reflects the fact that testing is typically done as a separate step after the design has been completed. However, it would also make sense to include testability evaluation in the chapter on evaluation and validation. Torn between an inclusion in that chapter and making the topic a separate chapter, we have finally chosen the second approach. Reasons include the risk of overloading the chapter on evaluation and validation too much, the need to skip this topic in shorter courses and the lack of integrated testing in industrial practices. Fig. 10 structures those topics which we would like to include in a standard coverage of embedded system fundamentals.

Test procedures
  Test pattern generation
  Self−test programs
Evaluation of test patterns and system robustness
  Fault coverage
  Fault simulation
  Fault injection
Design for testability
  Scan design
  Signature analysis
  Pseudo−random test pattern generation
  The built−in logic block observer (BILBO)
Assignments

Figure 10: Test-related content

We observed that the topics covered in this chapter are frequently referred to in conference talks and that students working on embedded systems should therefore be aware of them. Briefly covering these topics seems to be sufficient. A more detailed discussion would be nice to have, but computer science curricula as well as other curricula frequently do not include a standard course on the topic.

3.10 Other extensions

The design of embedded systems involves many dynamic behaviors. Such dynamic behaviors are sometimes difficult to understand without any visualization. Therefore, the LEVI simulation software was developed. LEVI allows the visualization of time-distance charts, Kahn process networks, the FlexRay^{TM} protocol, and real-time scheduling. References to LEVI have been integrated into the book where appropriate. As an example, fig. 11 shows a screen shot of the FlexRay^{TM} simulation tool[2]. Boxes in the lower part of the window demonstrate how messages are transmitted over the two buses.

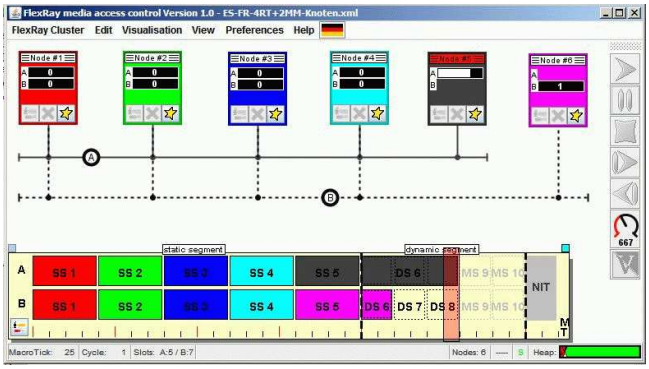

Figure 11: Screen shot from FlexRay^{TM} simulation tool

The first edition did not include any assignments. We believed that the situation would be too dynamic and that assignments available in the internet would be more appropriate. In the mean-time, the situation became more stable and it is therefore possible to include assignments in the book itself.

Due to the increased level of maturity, it became now possible to tape the lectures. Recorded lectures are now available on the internet [17]. Fig. 12 demonstrates the approach taken for the videos[3].

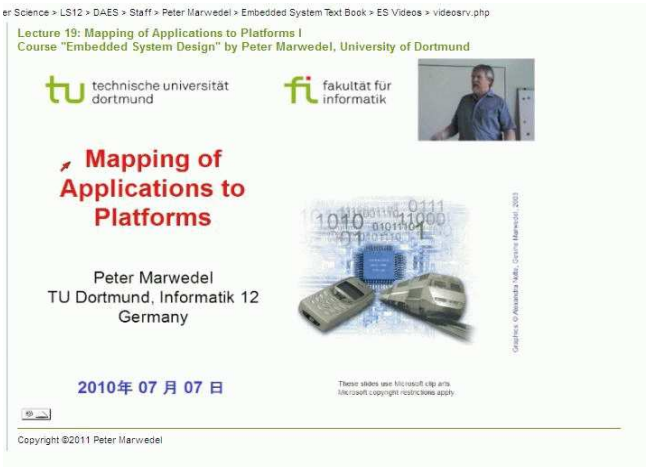

Figure 12: Screen shot from recorded lecture video

The focus is on a high resolution capture of the entire video stream available at the presentation laptop. All animations, simulations, and cursor movements are recorded.

[2]Will be shown in the presentation
[3]Will be shown in the presentation

The presenter is just shown in a small window which is kept empty on the slides.

3.11 Topics which have been dropped

Specific tool flows included in the first edition have been dropped, since we came to the conclusion that they do not need to be included in a standard curriculum. Their application range is too limited and they may have become obsolete. In particular, the descriptions of the Cosyma design flow [19] and the Octopus design flow [1] have been dropped. The SpecC design flow is listed only as an example of a flow which is compatible with our generic flow.

4. EVALUATION

The book became available during the course that was held in German during the winter of 2010/2011. Many of the students preferred to follow the German translation of the first edition instead of switching to the English second edition. Once again, this demonstrates that there is a market for local language translations, despite the wide-spread use of English for publications and for course slides. A German translation of the second edition will become available during the fall of 2011.

The first English course using the new edition was held in the summer of 2011. 26 students participated in the course. Unfortunately, only two of them responded to the request for comments on the book. Both suggested to include more material on the programming of embedded systems, in particular on programming robots. It is difficult to follow this recommendation. Only a very limited number of lab hours is available (about 12 weeks of 90 mins. each). These labs are intended to be used for discussions of concepts and to teach practical skills. Additional emphasis on programming would reduce the time available for concepts. Unfortunately, increasing the weight of the course in the CS curriculum is not an option.

Other comments have been received by colleagues. In one case, the book was considered to be very useful, but the inclusion of an introduction to evolutionary algorithms was suggested. Initially, the plan was to include such an introduction as an appendix, next to the introduction to integer linear programming. However, we came to the conclusion that a serious introduction would require quite some space and it would be essentially identical to one of the standard introductions to evolutionary algorithms. This plan was therefore dropped.

In another case, a colleague is recommending the book for his course on embedded systems. This colleague reported that he is always tempted to go more into details about the material that is covered and to provide more background. Indeed, it would be nice if the material could be covered in more detail. However, the material currently included in the book fills our course up the maximum. We have to accept that a course on the fundamentals of embedded system design cannot cover more that what is currently included.

A quick walk through the book has also been selected as the first part of a summer school on embedded systems [2]. At the summer school, advanced topics are linked to this first part (see fig. 13). This selection confirms the view of the book as a source for the fundamental knowledge required for any advanced work on embedded systems.

Three comments about the first edition of the book apply to the second edition as well: there seem to be several

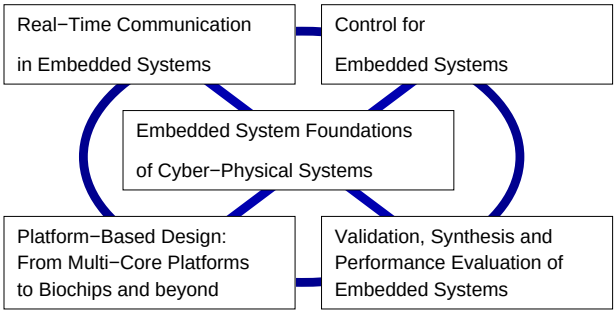

Figure 13: Linking advanced topics to the contents of our embedded systems book

universities, for which the basic education in the computer science program is insufficient for students to understand the book. In such cases, an additional course on computer organization (or similar) needs to be introduced in order to enable the comprehension of the text book.

E. Lee mentions the second edition as one of the few books covering the principles of embedded system design [13]. In still another case, a strong demand for copies of the book was reported. Obviously, plans are under way to include the topics of the book in the standard curriculum of several universities. Plans for translation into various languages such as German, Greek and Portuguese can be seen as an indication of meeting the needs of universities throughout the world for a text book in the topic.

5. CONCLUSION

From our point of view, the second edition of the text book "Embedded systems design" is a big step forward, enabling the easy integration of a course on the subject into computer science and computer engineering curricula.

The authors acknowledge the partial support of this work via the "Ruhr Campus Online" initiative as well as via the funding of the ArtistDesign network of excellence by the Commission of the European Communities.

6. REFERENCES

[1] M. Awad, J. Kuusela, and J. Ziegler. *Object-Oriented Technology for Real-Time Systems*. Prentice Hall, 1996.

[2] B. Bouyssounouse. Schedule of the ARTIST summer school in china. *http://www.artist-embedded.org/artist/Schedule,2321.html*, 2011.

[3] G.C. Buttazzo. *Hard Real-time computing systems*. Kluwer Academic Publishers, 4th printing, 2002.

[4] P. Caspi, A. Sangiovanni-Vincentelli, L. Almeida, A. Benviste, B. Bouyssounouse, G. Buttazzo, I. Crnkovic, W. Damm, J. Engblom, G. Fohler, M. Garcia-Valls, H. Kopetz, Y. Lakhnech, F. Laroussinie, L. Lavagno, G. Lipari, F. Maraninchi, Ph. Peti, J. de la Puente, N. Scaife, J. Sifakis, R. De Simone, P. Verissimo M. Torngren and, A. J. Wellings, R. Wilhelm, T. Willemse, and W. Yi. Guidelines for a graduate curriculum on embedded software and systems. *ACM Transactions on Embedded Computing Systems (TECS)*, pages 587–611, 2005.

[5] Stephen A Edwards. *Languages for digital systems*. Kluwer Academic Publishers, 2000.

[6] Daniel. D Gajski, Samar Abdi, Andreas Gerstlauer, and Gunar Schirner. *Embedded System Design*. Springer, Heidelberg, 2009.

[7] ITRS Organization. International technology roadmap for semiconductors (ITRS). *http://public.itrs.net*, 2009.

[8] A. Jantsch. *Modeling Embedded Systems and SoC's: Concurrency and Time in Models of Computation*. Morgan Kaufmann, 2004.

[9] H. Kopetz. *Real-Time Systems –Design Principles for Distributed Embedded Applications–*. Kluwer Academic Publishers, 1997.

[10] H. Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Springer, 2011.

[11] C.M. Krishna and K. G. Shin. *Real-Time Systems*. McGraw-Hill, Computer Science Series, 1997.

[12] Edward A. Lee. The future of embedded software. *ARTEMIS Conference, Graz, http://ptolemy.eecs.berkeley.edu/presentations/06/FutureOfEmbeddedSoftware_Lee_Graz.ppt*, 2006.

[13] Edward A. Lee and Sanjit A. Seshia. Introduction to embedded systems, a cyber-physical systems approach. *http://LeeSeshia.org, ISBN 978-0-557-70857-4*, 2011.

[14] Jane W.S. Liu. *Real-Time Systems*. Prentice Hall, 2000.

[15] P. Marwedel. Towards laying common grounds for embedded system design education. *ACM SIGBED Review*, pages 25–28, 2005.

[16] P. Marwedel. 4th workshop on mapping of applications to MPSoCs. *http://www.artist-embedded.org/artist/Program,2298.html*, 2011.

[17] P. Marwedel. Home page for the book "embedded system design". *http://ls12-www.cs.tu-dortmund.de/~marwedel/es-book/*, 2011.

[18] P. Marwedel, J. Jackson, and K. Ricks. Workshop on embedded system education. *http://www.artist-embedded.org/artist/WESE-10.html*, 2010.

[19] Achim Österling, Thomas Benner, Rolf Ernst, Dirk Herrmann, Thomas Scholz, and Wei Ye. The COSYMA system. *http://www.ida.ing.tu-bs.de/research/projects/cosyma/overview/node1.html*, 1997.

[20] L. Thiele. Performance analysis of distributed embedded systems. *In: R. Zurawski (Ed.): Embedded Systems Handbook, CRC Press*, 2006.

[21] Thiele, L. et al. SHAPES TIK. *http://www.tik.ee.ethz.ch/~shapes/dol.html*, 2009.

[22] Shiao-Li Tsao, Tai-Yi Huang, and Chung-Ta King. The development and deployment of embedded software curricula in taiwan. *SIGBED Rev.*, 4:64–72, January 2007.

[23] F. Vahid. *Embedded System Design*. John Wiley& Sons, 2002.

[24] R. Wilhelm. Determining bounds on execution times. *In: R. Zurawski (Ed.): Embedded Systems Handbook, CRC Press*, 2006.

[25] W. Wolf. *Computers as Components*. Morgan Kaufmann Publishers, 2001.

[26] R. Zurawski, editor. *Embedded Systems Handbook*. CRC Press, 2006.