

## ***Embedded Systems Hardware For Software Engineers***

Current practice dictates the separation of the hardware and software development paths early in the design cycle. These paths remain independent with very little interaction occurring between them until system integration. In particular, hardware is often specified without fully appreciating the computational requirements of the software. Also, software development does not influence hardware development and does not track changes made during the hardware design phase. Thus, the ability to explore hardware/software tradeoffs is restricted, such as the movement of functionality from the software domain to the hardware domain (and vice-versa) or the modification of the hardware/software interface. As a result, problems that are encountered during system integration may require modification of the software and/or hardware, resulting in potentially significant cost increases and schedule overruns. To address the problems described above, a cooperative design approach, one that utilizes a unified view of hardware and software, is

described. This approach is called hardware/software codesign. The Codesign of Embedded Systems develops several fundamental hardware/software codesign concepts and a methodology that supports them. A unified representation, referred to as a decomposition graph, is presented which can be used to describe hardware or software using either functional abstractions or data abstractions. Using a unified representation based on functional abstractions, an abstract hardware/software model has been implemented in a common simulation environment called ADEPT (Advanced Design Environment Prototyping Tool). This model permits early hardware/software evaluation and tradeoff exploration. Techniques have been developed which support the identification of software bottlenecks and the evaluation of design alternatives with respect to multiple metrics. The application of the model is demonstrated on several examples. A unified representation based on data abstractions is also explored. This work leads to investigations regarding the application of object-oriented techniques to hardware design. The Codesign of Embedded Systems: A Unified Hardware/Software Representation describes a novel approach to a topic of immense importance to CAD researchers and designers alike.

An introduction to the engineering principles of embedded systems, with a focus on modeling, design, and analysis of cyber-physical systems. The most visible use of computers and software is processing information for human consumption. The vast majority of computers in use, however, are much less visible. They run the engine, brakes, seatbelts, airbag, and audio system in your car. They digitally encode your voice and construct a radio signal to send it from your cell phone to a base station. They command robots on a factory floor, power generation in a power plant, processes in a chemical plant, and traffic lights in a city. These less visible computers are called embedded systems, and the software they run is called embedded software. The principal challenges in designing and analyzing embedded systems stem from their interaction with physical processes. This book takes a cyber-physical approach to embedded systems, introducing the engineering concepts underlying embedded systems as a technology and as a subject of study. The focus is on modeling, design, and analysis of cyber-physical systems, which integrate computation, networking, and physical processes. The second edition offers two new chapters, several new exercises, and other improvements. The book can be used as a textbook at the advanced

undergraduate or introductory graduate level and as a professional reference for practicing engineers and computer scientists. Readers should have some familiarity with machine structures, computer programming, basic discrete mathematics and algorithms, and signals and systems.

A PRACTICAL GUIDE TO HARDWARE FUNDAMENTALS Embedded Systems Hardware for Software Engineers describes the electrical and electronic circuits that are used in embedded systems, their functions, and how they can be interfaced to other devices. Basic computer architecture topics, memory, address decoding techniques, ROM, RAM, DRAM, DDR, cache memory, and memory hierarchy are discussed. The book covers key architectural features of widely used microcontrollers and microprocessors, including Microchip's PIC32, ATMEL's AVR32, and Freescale's MC68000. Interfacing to an embedded system is then described. Data acquisition system level design considerations and a design example are presented with real-world parameters and characteristics. Serial interfaces such as RS-232, RS-485, PC, and USB are addressed and printed circuit boards and high-speed signal propagation over transmission lines are covered with a minimum of math.

A brief survey of logic families of integrated circuits and programmable logic devices is also contained in this in-depth resource. **COVERAGE INCLUDES:** Architecture examples Memory Memory address decoding Read-only memory and other related devices Input and output ports Analog-to-digital and digital-to-analog converters Interfacing to external devices Transmission lines Logic families of integrated circuits and their signaling characteristics The printed circuit board Programmable logic devices Test equipment: oscilloscopes and logic analyzers

**Embedded Systems: A Contemporary Design Tool, Second Edition**

Embedded systems are one of the foundational elements of today ' s evolving and growing computer technology. From operating our cars, managing our smart phones, cleaning our homes, or cooking our meals, the special computers we call embedded systems are quietly and unobtrusively making our lives easier, safer, and more connected. While working in increasingly challenging environments, embedded systems give us the ability to put increasing amounts of capability into ever-smaller and more powerful devices. **Embedded Systems: A Contemporary Design Tool, Second Edition** introduces you to the theoretical hardware and software foundations of these systems and

expands into the areas of signal integrity, system security, low power, and hardware-software co-design. The text builds upon earlier material to show you how to apply reliable, robust solutions to a wide range of applications operating in today ' s often challenging environments. Taking the user ' s problem and needs as your starting point, you will explore each of the key theoretical and practical issues to consider when designing an application in today ' s world. Author James Peckol walks you through the formal hardware and software development process covering: Breaking the problem down into major functional blocks; Planning the digital and software architecture of the system; Utilizing the hardware and software co-design process; Designing the physical world interface to external analog and digital signals; Addressing security issues as an integral part of the design process; Managing signal integrity problems and reducing power demands in contemporary systems; Debugging and testing throughout the design and development cycle; Improving performance. Stressing the importance of security, safety, and reliability in the design and development of embedded systems and providing a balanced treatment of both the hardware and the software aspects, Embedded Systems: A Contemporary Design Tool,

Second Edition gives you the tools for creating embedded designs that solve contemporary real-world challenges.

Best Practices for Improving Embedded Systems Development

A Cyber-Physical Systems Approach

Programmable Hardware

Embedded Software Development

Embedded Controller Hardware Design

Software and Compilers for Embedded Systems

Co-Synthesis of Hardware and Software for Digital Embedded Systems, with a Foreword written by Giovanni De Micheli, presents techniques that are useful in building complex embedded systems. These techniques provide a competitive advantage over purely hardware or software implementations of time-constrained embedded systems. Recent advances in chip-level synthesis have made it possible to synthesize application-specific circuits under strict timing constraints. This work advances the state of the art by formulating the problem of system synthesis using both application-specific as well as reprogrammable components, such as off-the-shelf processors. Timing constraints are used to determine what part of the system functionality must be delegated to dedicated application-specific hardware while the rest is delegated to software that runs on the processor. This co-synthesis of hardware and software from behavioral specifications makes it possible to realize real-time embedded systems using off-the-shelf parts and a relatively small amount

of application-specific circuitry that can be mapped to semi-custom VLSI such as gate arrays. The ability to perform detailed analysis of timing performance provides the opportunity of improving the system definition by creating better prototypes. Co-Synthesis of Hardware and Software for Digital Embedded Systems is of interest to CAD researchers and developers who want to branch off into the expanding field of hardware/software co-design, as well as to digital system designers who are interested in the present power and limitations of CAD techniques and their likely evolution.

This textbook introduces the concept of embedded systems with exercises using Arduino Uno. It is intended for advanced undergraduate and graduate students in computer science, computer engineering, and electrical engineering programs. It contains a balanced discussion on both hardware and software related to embedded systems, with a focus on co-design aspects. Embedded systems have applications in Internet-of-Things (IoT), wearables, self-driving cars, smart devices, cyberphysical systems, drones, and robotics. The hardware chapter discusses various microcontrollers (including popular microcontroller hardware examples), sensors, amplifiers, filters, actuators, wired and wireless communication topologies, schematic and PCB designs, and much more. The software chapter describes OS-less programming, bitmath, polling, interrupt, timer, sleep modes, direct memory access, shared memory, mutex, and smart algorithms, with lots of C-code examples for Arduino Uno. Other topics discussed are prototyping, testing, verification, reliability, optimization, and regulations. Appropriate for courses on embedded systems, microcontrollers, and instrumentation, this textbook teaches budding embedded system programmers practical skills with fun projects to prepare them for industry products. Introduces embedded systems

## Online Library Embedded Systems Hardware For Software Engineers

for wearables, Internet-of-Things (IoT), robotics, and other smart devices; Offers a balanced focus on both hardware and software co-design of embedded systems; Includes exercises, tutorials, and assignments.

In this new edition the latest ARM processors and other hardware developments are fully covered along with new sections on Embedded Linux and the new freeware operating system eCOS. The hot topic of embedded systems and the internet is also introduced. In addition a fascinating new case study explores how embedded systems can be developed and experimented with using nothing more than a standard PC. \* A practical introduction to the hottest topic in modern electronics design \* Covers hardware, interfacing and programming in one book \* New material on Embedded Linux for embedded internet systems

When planning the development of modern embedded systems, hardware and software cannot be considered independently. Over the last two decades chip and system complexity has seen an enormous amount of growth, while more and more system functionality has moved from dedicated hardware implementation into software executing on general-purposed embedded processors. By 2010 the development effort for software had outgrown the development efforts for hardware, and the complexity trend continues in favor of software. Traditional design techniques such as independent hardware and software design are being challenged due to heterogeneous models and applications being integrated to create a complex system on chip. Using proper techniques of hardware-software codesign, designers consider the trade-offs in the way hardware and software components of a system work together to exhibit a specified behavior, given a set of performance goals and

technology. This chapter will cover these topics.

Computing in Harsh Environments

Unleash the Power of Arduino!

Hardware-Software Co-Design of Embedded Systems

Embedded Systems Security

Embedded Systems Hardware for Software Engineers

The POLIS Approach

This introduction to the design of embedded systems provides for hardware and software engineers the methodology, base of knowledge, and common problems in the field of embedded design. Included are discussions of device architecture, memory, I/O and development techniques. 5 photos, 95 line drawings, 12 tables.

Modern embedded systems require high performance, low cost and low power consumption. Such systems typically consist of a heterogeneous collection of processors, specialized memory subsystems, and partially programmable or fixed-function components. This heterogeneity, coupled with issues such as hardware/software partitioning, mapping, scheduling, etc., leads to a large number of design possibilities, making performance debugging and validation of such systems a difficult problem. Embedded systems are used to control safety critical applications such as flight control, automotive electronics and healthcare monitoring. Clearly, developing reliable software/systems for such applications is of utmost importance. This book describes a

host of debugging and verification methods which can help to achieve this goal. Covers the major abstraction levels of embedded systems design, starting from software analysis and micro-architectural modeling, to modeling of resource sharing and communication at the system level Integrates formal techniques of validation for hardware/software with debugging and validation of embedded system design flows Includes practical case studies to answer the questions: does a design meet its requirements, if not, then which parts of the system are responsible for the violation, and once they are identified, then how should the design be suitably modified?

Develop the software and hardware you never think about. We're talking about the nitty-gritty behind the buttons on your microwave, inside your thermostat, inside the keyboard used to type this description, and even running the monitor on which you are reading it now. Such stuff is termed embedded systems, and this book shows how to design and develop embedded systems at a professional level. Because yes, many people quietly make a successful career doing just that. Building embedded systems can be both fun and intimidating. Putting together an embedded system requires skill sets from multiple engineering disciplines, from software and hardware in particular. Building Embedded Systems is a book about helping you do things in the right way from the beginning of your first project: Programmers who know software will learn what they need to know about hardware. Engineers with hardware knowledge likewise will learn about the

software side. Whatever your background is, *Building Embedded Systems* is the perfect book to fill in any knowledge gaps and get you started in a career programming for everyday devices. Author Changyi Gu brings more than fifteen years of experience in working his way up the ladder in the field of embedded systems. He brings knowledge of numerous approaches to embedded systems design, including the System on Programmable Chips (SOPC) approach that is currently growing to dominate the field. His knowledge and experience make *Building Embedded Systems* an excellent book for anyone wanting to enter the field, or even just to do some embedded programming as a side project.

**What You Will Learn**

- Program embedded systems at the hardware level
- Learn current industry practices in firmware development
- Develop practical knowledge of embedded hardware options
- Create tight integration between software and hardware
- Practice a work flow leading to successful outcomes
- Build from transistor level to the system level
- Make sound choices between performance and cost

**Who This Book Is For**

Embedded-system engineers and intermediate electronics enthusiasts who are seeking tighter integration between software and hardware. Those who favor the System on a Programmable Chip (SOPC) approach will in particular benefit from this book. Students in both Electrical Engineering and Computer Science can also benefit from this book and the real-life industry practice it provides.

Interested in developing embedded systems? Since they don't tolerate inefficiency, these

systems require a disciplined approach to programming. This easy-to-read guide helps you cultivate a host of good development practices, based on classic software design patterns and new patterns unique to embedded programming. Learn how to build system architecture for processors, not operating systems, and discover specific techniques for dealing with hardware difficulties and manufacturing requirements. Written by an expert who's created embedded systems ranging from urban surveillance and DNA scanners to children's toys, this book is ideal for intermediate and experienced programmers, no matter what platform you use. Optimize your system to reduce cost and increase performance Develop an architecture that makes your software robust in resource-constrained environments Explore sensors, motors, and other I/O devices Do more with less: reduce RAM consumption, code space, processor cycles, and power consumption Learn how to update embedded code directly in the processor Discover how to implement complex mathematics on small processors Understand what interviewers look for when you apply for an embedded systems job "Making Embedded Systems is the book for a C programmer who wants to enter the fun (and lucrative) world of embedded systems. It's very well written—entertaining, even—and filled with clear illustrations." —Jack Ganssle, author and embedded system expert.

Co-Synthesis of Hardware and Software for Digital Embedded Systems  
Embedded Systems

Rugged Embedded Systems

Design of Hardware/Software Embedded Systems

Embedded and Networking Systems

Introduction to Embedded Systems, Second Edition

Why care about hardware/firmware interaction? These interfaces are critical, a solid hardware design married with adaptive firmware can access all the capabilities of an application and overcome limitations caused by poor communication. For the first time, a book has come along that will help hardware engineers and firmware engineers work together to mitigate or eliminate problems that occur when hardware and firmware are not optimally compatible. Solving these issues will save time and money, getting products to market sooner to create more revenue. The principles and best practices presented in this book will prove to be a valuable resource for both hardware and firmware engineers. Topics include register layout, interrupts, timing and performance, aborts, and errors. Real world cases studies will help to solidify the principles and best practices with an aim towards cleaner designs, shorter schedules, and better implementation! Reduce product development delays with the best practices in this book Concepts apply to ASICs, ASSPs, SoCs, and FPGAs Real-world examples and case studies highlight the good and bad of design processes

Front Cover; Dedication; Embedded Systems Security: Practical Methods for Safe and Secure Software and Systems Development; Copyright; Contents; Foreword; Preface;

About this Book; Audience; Organization; Approach; Acknowledgements; Chapter 1 -- Introduction to Embedded Systems Security; 1.1What is Security?; 1.2What is an Embedded System?; 1.3Embedded Security Trends; 1.4Security Policies; 1.5Security Threats; 1.6Wrap-up; 1.7Key Points; 1.8 Bibliography and Notes; Chapter 2 -- Systems Software Considerations; 2.1The Role of the Operating System; 2.2Multiple Independent Levels of Security.

Current multimedia and telecom applications require complex, heterogeneous multiprocessor system on chip (MPSoC) architectures with specific communication infrastructure in order to achieve the required performance. Heterogeneous MPSoC includes different types of processing units (DSP, microcontroller, ASIP) and different communication schemes (fast links, non standard memory organization and access). Programming an MPSoC requires the generation of efficient software running on MPSoC from a high level environment, by using the characteristics of the architecture. This task is known to be tedious and error prone, because it requires a combination of high level programming environments with low level software design. This book gives an overview of concepts related to embedded software design for MPSoC. It details a full software design approach, allowing systematic, high-level mapping of software applications on heterogeneous MPSoC. This approach is based on gradual refinement of hardware/software interfaces and simulation models allowing to validate the software at different abstraction levels. This book combines Simulink for high level programming

and SystemC for the low level software development. This approach is illustrated with multiple examples of application software and MPSoC architectures that can be used for deep understanding of software design for MPSoC.

Embedded and Networking Systems: Design, Software, and Implementation explores issues related to the design and synthesis of high-performance embedded computer systems and networks. The emphasis is on the fundamental concepts and analytical techniques that are applicable to a range of embedded and networking applications, rather than on specific embedded architectures, software development, or system-level integration. This system point of view guides designers in dealing with the trade-offs to optimize performance, power, cost, and other system-level non-functional requirements. The book brings together contributions by researchers and experts from around the world, offering a global view of the latest research and development in embedded and networking systems. Chapters highlight the evolution and trends in the field and supply a fundamental and analytical understanding of some underlying technologies. Topics include the co-design of embedded systems, code optimization for a variety of applications, power and performance trade-offs, benchmarks for evaluating embedded systems and their components, and mobile sensor network systems. The book also looks at novel applications such as mobile sensor systems and video networks. A comprehensive review of groundbreaking technology and applications, this book is a timely resource for system designers, researchers, and students interested in

the possibilities of embedded and networking systems. It gives readers a better understanding of an emerging technology evolution that is helping drive telecommunications into the next decade.

Embedded Systems Design

Simulink and System C Case Studies

Readings in Hardware/software Co-design

Embedded Systems Foundations of Cyber-Physical Systems

Software Engineering for Embedded Systems

Embedded Software Development with C

Este libro presenta los desafíos planteados por las nuevas y sumamente poderosas tecnologías de integración de sistemas electrónicos, que están en la base de los cambios sociales hacia lo que llaman la Sociedad de la Información; en la que los dispositivos electrónicos se harán una parte incorporada de la vida diaria, encajados en casi cada producto. Es necesario un conocimiento cuidadoso de los desafíos para aprovechar la amplia gama de ocasiones ofrecidas por tales capacidades de integración y las correspondientes posibilidades de diseño de sistemas electrónicos.

Embedded Systems – A Hardware-Software Co-Design Approach  
Unleash the Power of Arduino! Springer Nature

Rugged Embedded Systems: Computing in Harsh Environments describes how to design reliable embedded systems for harsh environments, including architectural

approaches, cross-stack hardware/software techniques, and emerging challenges and opportunities. A "harsh environment" presents inherent characteristics, such as extreme temperature and radiation levels, very low power and energy budgets, strict fault tolerance and security constraints, etc. that challenge the computer system in its design and operation. To guarantee proper execution (correct, safe, and low-power) in such scenarios, this contributed work discusses multiple layers that involve firmware, operating systems, and applications, as well as power management units and communication interfaces. This book also incorporates use cases in the domains of unmanned vehicles (advanced cars and micro aerial robots) and space exploration as examples of computing designs for harsh environments. Provides a deep understanding of embedded systems for harsh environments by experts involved in state-of-the-art autonomous vehicle-related projects Covers the most important challenges (fault tolerance, power efficiency, and cost effectiveness) faced when developing rugged embedded systems Includes case studies exploring embedded computing for autonomous vehicle systems (advanced cars and micro aerial robots) and space exploration

Famed author Jack Ganssle has selected the very best embedded systems design material from the Newnes portfolio and compiled into this volume. The result is a book covering the gamut of embedded design—from hardware to software to integrated embedded systems—with a strong pragmatic emphasis. In addition to specific design

techniques and practices, this book also discusses various approaches to solving embedded design problems and how to successfully apply theory to actual design tasks. The material has been selected for its timelessness as well as for its relevance to contemporary embedded design issues. This book will be an essential working reference for anyone involved in embedded system design! Table of Contents: Chapter 1. Motors - Stuart Ball Chapter 2. Testing – Arnold S. Berger Chapter 3. System-Level Design – Keith E. Curtis Chapter 4. Some Example Sensor, Actuator and Control Applications and Circuits (Hard Tasks) – Lewin ARW Edwards Chapter 5. Installing and Using a Version Control System – Chris Keydel and Olaf Meding Chapter 6. Embedded State Machine Implementation - Martin Gomez Chapter 7. Firmware Musings – Jack Ganssle Chapter 8. Hardware Musings – Jack Ganssle Chapter 9. Closed Loop Controls, Rabbits, and Hounds - John M. Holland Chapter 10. Application Examples David J. Katz and Rick Gentile Chapter 11. Analog I/Os – Jean LaBrosse Chapter 12. Optimizing DSP Software – Robert Oshana Chapter 13. Embedded Processors – Peter Wilson \*Hand-picked content selected by embedded systems luminary Jack Ganssle \*Real-world best design practices including chapters on FPGAs, DSPs, and microcontrollers \*Covers both hardware and software aspects of embedded systems Design Patterns for Great Software With C and GNU Development Tools Building Embedded Systems

Designing Embedded Hardware  
Using Microcontrollers and the MSP430  
Hardware/Firmware Interface Design

*Embedded Software Development With C offers both an effectual reference for professionals and researchers, and a valuable learning tool for students by laying the groundwork for a solid foundation in the hardware and software aspects of embedded systems development. Key features include a resource for the fundamentals of embedded systems design and development with an emphasis on software, an exploration of the 8051 microcontroller as it pertains to embedded systems, comprehensive tutorial materials for instructors to provide students with labs of varying lengths and levels of difficulty, and supporting website including all sample codes, software tools and links to additional online references.*

*Embedded computer systems use both off-the-shelf microprocessors and application-specific integrated circuits (ASICs) to implement specialized system functions. Examples include the electronic systems inside laser printers, cellular phones, microwave ovens, and an automobile anti-lock brake controller. Embedded computing is unique because it is a co-design problem - the hardware engine and application software architecture must be designed*

*simultaneously. Hardware-Software Co-Synthesis of Distributed Embedded Systems proposes new techniques such as fixed-point iterations, phase adjustment, and separation analysis to efficiently estimate tight bounds on the delay required for a set of multi-rate processes preemptively scheduled on a real-time reactive distributed system. Based on the delay bounds, a gradient-search co-synthesis algorithm with new techniques such as sensitivity analysis, priority prediction, and idle- processing elements elimination are developed to select the number and types of processing elements in a distributed engine, and determine the allocation and scheduling of processes to processing elements. New communication modeling is also presented to analyze communication delay under interaction of computation and communication, allocate interprocessor communication links, and schedule communication. Hardware-Software Co-Synthesis of Distributed Embedded Systems is the first book to describe techniques for the design of distributed embedded systems, which have arbitrary hardware and software topologies. The book will be of interest to: academic researchers for personal libraries and advanced-topics courses in co-design as well as industrial designers who are building high-performance, real-time embedded systems with multiple processors. This is a book about developing the software and hardware you never think*

*about. We're talking about the nitty-gritty behind the buttons on your microwave, inside your thermostat, inside the keyboard used to type this description, and even running the monitor on which you are reading it now. Such stuff is termed embedded systems, and this book shows how to design and develop embedded systems at a professional level. Because yes, many people quietly make a successful career doing just that. Building embedded systems can be both fun and intimidating. Putting together an embedded system requires skill sets from multiple engineering disciplines, from software and hardware in particular. Building Embedded Systems is a book about helping you do things in the right way from the beginning of your first project: Programmers who know software will learn what they need to know about hardware. Engineers with hardware knowledge likewise will learn about the software side. Whatever your background is, Building Embedded Systems is the perfect book to fill in any knowledge gaps and get you started in a career programming for everyday devices. Author Changyi Gu brings more than fifteen years of experience in working his way up the ladder in the field of embedded systems. He brings knowledge of numerous approaches to embedded systems design, including the System on Programmable Chips (SOPC) approach that is currently growing to dominate the field. His knowledge*

*and experience make Building Embedded Systems an excellent book for anyone wanting to enter the field, or even just to do some embedded programming as a side project. What You Will Learn Program embedded systems at the hardware level Learn current industry practices in firmware development Develop practical knowledge of embedded hardware options Create tight integration between software and hardware Practice a work flow leading to successful outcomes Build from transistor level to the system level Make sound choices between performance and cost Who This Book Is For Building Embedded Systems: Programmable Hardware is for embedded-system engineers and intermediate electronics enthusiasts who are seeking tighter integration between software and hardware. Those who favor the System on a Programmable Chip (SOPC) approach will in particular benefit from this book. Students in both Electrical Engineering and Computer Science can also benefit from this book and the real-life industry practice it provides. This textbook serves as an introduction to the subject of embedded systems design, using microcontrollers as core components. It develops concepts from the ground up, covering the development of embedded systems technology, architectural and organizational aspects of controllers and systems, processor models, and peripheral devices. Since microprocessor-based embedded*

*systems tightly blend hardware and software components in a single application, the book also introduces the subjects of data representation formats, data operations, and programming styles. The practical component of the book is tailored around the architecture of a widely used Texas Instrument's microcontroller, the MSP430 and a companion web site offers for download an experimenter's kit and lab manual, along with Powerpoint slides and solutions for instructors.*

*Programming Embedded Systems*

*The Codesign of Embedded Systems: A Unified Hardware/Software Representation*

*Embedded Software Design and Programming of Multiprocessor System-on-Chip*

*Embedded Systems - A Hardware-Software Co-Design Approach  
Methods, Practical Techniques, and Applications*

**This title serves as an introduction and reference for the field, with the papers that have shaped the hardware/software co-design since its inception in the early 90s.**

**This book constitutes the refereed proceedings of the 8th International**

**Workshop on Software and Compilers for Embedded Systems, SCOPES 2004, held in Amsterdam, The Netherlands, in September 2004. The 17 revised full papers presented were carefully reviewed and selected from close to 50 submissions. The papers are organized in topical sections on application synthesis, data flow analysis, data partitioning, task scheduling, and code generation.**

**Appropriate for use as a graduate text or a professional reference, Languages for Digital Embedded Systems is the first detailed, broad survey of hardware and software description languages for embedded system design. Instead of promoting the one language that will solve all design problems (which does not and will not ever exist), this book takes the view that different problems demand different languages, and a designer who knows the spectrum of available languages has the advantage over one who is trapped using the wrong language. Languages for Digital Embedded Systems concentrates on successful, widely-used design languages, with a secondary emphasis on those with significant theoretical value. The syntax, semantics, and implementation of each language is discussed, since although hardware synthesis and software compilation technology have steadily improved, coding style still matters, and a thorough understanding of how a language is synthesized or compiled is generally necessary to take full advantage of a language. Practicing designers, graduate students, and advanced undergraduates**

**will all benefit from this book. It assumes familiarity with some hardware or software languages, but takes a practical, descriptive view that avoids formalism.**

**Covers the significant embedded computing technologies—highlighting their applications in wireless communication and computing power An embedded system is a computer system designed for specific control functions within a larger system—often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. Presented in three parts, Embedded Systems: Hardware, Design, and Implementation provides readers with an immersive introduction to this rapidly growing segment of the computer industry. Acknowledging the fact that embedded systems control many of today's most common devices such as smart phones, PC tablets, as well as hardware embedded in cars, TVs, and even refrigerators and heating systems, the book starts with a basic introduction to embedded computing systems. It hones in on system-on-a-chip (SoC), multiprocessor system-on-chip (MPSoC), and network-on-chip (NoC). It then covers on-chip integration of software and custom hardware accelerators, as well as fabric flexibility, custom architectures, and the multiple I/O standards that facilitate PCB integration. Next, it focuses on the technologies associated with embedded computing systems, going over the basics of field-programmable gate array (FPGA), digital signal processing (DSP)**

**and application-specific integrated circuit (ASIC) technology, architectural support for on-chip integration of custom accelerators with processors, and O/S support for these systems. Finally, it offers full details on architecture, testability, and computer-aided design (CAD) support for embedded systems, soft processors, heterogeneous resources, and on-chip storage before concluding with coverage of software support—in particular, O/S Linux. Embedded Systems: Hardware, Design, and Implementation is an ideal book for design engineers looking to optimize and reduce the size and cost of embedded system products and increase their reliability and performance.**

**A Contemporary Design Tool**

**A Unified Hardware/Software Representation**

**Design, Software, and Implementation**

**Co-verification of Hardware and Software for ARM SoC Design**

**Hardware-Software Co-Synthesis of Distributed Embedded Systems**

**Chapter 1. Software Engineering of Embedded and Real-Time Systems**

An embedded system is a computer system designed for a specific function within a larger system, and often has one or more real-time computing constraints. It is embedded as part of a larger device which can include hardware and mechanical parts. This is in stark contrast to a general-purpose computer, which is designed to be flexible and meet a wide range of end-user needs. The methods, techniques, and tools for developing software systems that were successfully applied to general purpose computing are not as readily applicable to embedded computing. Software systems running on networks of mobile,

embedded devices must exhibit properties that are not always required of more traditional systems such as near-optimal performance, robustness, distribution, dynamism, and mobility. This chapter will examine the key properties of software systems in the embedded, resource-constrained, mobile, and highly distributed world. The applicability of mainstream software engineering methods is assessed and techniques (e.g., software design, component-based development, software architecture, system integration and test) are also discussed in the context of this domain. This chapter will overview embedded and real-time systems.

This book presents techniques for energy reduction in adaptive embedded multimedia systems, based on dynamically reconfigurable processors. The approach described will enable designers to meet performance/area constraints, while minimizing video quality degradation, under various, run-time scenarios. Emphasis is placed on implementing power/energy reduction at various abstraction levels. To enable this, novel techniques for adaptive energy management at both processor architecture and application architecture levels are presented, such that both hardware and software adapt together, minimizing overall energy consumption under unpredictable, design-/compile-time scenarios.

This book introduces a modern approach to embedded system design, presenting software design and hardware design in a unified manner. It covers trends and challenges, introduces the design and use of single-purpose processors ("hardware") and general-purpose processors ("software"), describes memories and buses, illustrates hardware/software tradeoffs using a digital camera example, and discusses advanced computation models, controls systems, chip technologies, and modern design tools. For courses found in EE, CS and other engineering departments.

**Embedded Software Development: The Open-Source Approach** delivers a practical introduction to embedded software development, with a focus on open-source components. This programmer-centric

book is written in a way that enables even novice practitioners to grasp the development process as a whole. Incorporating real code fragments and explicit, real-world open-source operating system references (in particular, FreeRTOS) throughout, the text: Defines the role and purpose of embedded systems, describing their internal structure and interfacing with software development tools Examines the inner workings of the GNU compiler collection (GCC)-based software development system or, in other words, toolchain Presents software execution models that can be adopted profitably to model and express concurrency Addresses the basic nomenclature, models, and concepts related to task-based scheduling algorithms Shows how an open-source protocol stack can be integrated in an embedded system and interfaced with other software components Analyzes the main components of the FreeRTOS Application Programming Interface (API), detailing the implementation of key operating system concepts Discusses advanced topics such as formal verification, model checking, runtime checks, memory corruption, security, and dependability Embedded Software Development: The Open-Source Approach capitalizes on the authors' extensive research on real-time operating systems and communications used in embedded applications, often carried out in strict cooperation with industry. Thus, the book serves as a springboard for further research.

Making Embedded Systems

Hardware, Design and Implementation

Embedded Systems and Software Validation

Hardware/ Software Co-Design for Data Flow Dominated Embedded Systems

Introduction to Embedded Systems

Hardware/ Software Architectures for Low-Power Embedded Multimedia Systems

A presentation of developments in microcontroller technology, providing lucid

instructions on its many and varied applications. It focuses on the popular eight-bit microcontroller, the 8051, and the 83C552. The text outlines a systematic methodology for small-scale, control-dominated embedded systems, and is accompanied by a disk of all the example problems included in the book.

This Expert Guide gives you the techniques and technologies in software engineering to optimally design and implement your embedded system. Written by experts with a solutions focus, this encyclopedic reference gives you an indispensable aid to tackling the day-to-day problems when using software engineering methods to develop your embedded systems. With this book you will learn:

- The principles of good architecture for an embedded system
- Design practices to help make your embedded project successful
- Details on principles that are often a part of embedded systems, including digital signal processing, safety-critical principles, and development processes
- Techniques for setting up a performance engineering strategy for your embedded system software
- How to develop user interfaces for embedded systems
- Strategies for testing and deploying your embedded system, and ensuring quality development processes
- Practical techniques for optimizing embedded software for performance, memory, and power
- Advanced guidelines for developing multicore software for embedded systems
- How to develop embedded software for networking, storage, and automotive segments
- How to manage the embedded development process

Includes contributions from: Frank Schirrmeister, Shelly Gretlein, Bruce Douglass, Erich Styger,

Gary Stringham, Jean Labrosse, Jim Trudeau, Mike Brogioli, Mark Pitchford, Catalin Dan Udma, Markus Levy, Pete Wilson, Whit Waldo, Inga Harris, Xinxin Yang, Srinivasa Addepalli, Andrew McKay, Mark Kraeling and Robert Oshana. Road map of key problems/issues and references to their solution in the text Review of core methods in the context of how to apply them Examples demonstrating timeless implementation details Short and to- the- point case studies show how key ideas can be implemented, the rationale for choices made, and design guidelines and trade-offs Hardware/software co-verification is how to make sure that embedded system software works correctly with the hardware, and that the hardware has been properly designed to run the software successfully -before large sums are spent on prototypes or manufacturing. This is the first book to apply this verification technique to the rapidly growing field of embedded systems-on-a-chip(SoC). As traditional embedded system design evolves into single-chip design, embedded engineers must be armed with the necessary information to make educated decisions about which tools and methodology to deploy. SoC verification requires a mix of expertise from the disciplines of microprocessor and computer architecture, logic design and simulation, and C and Assembly language embedded software. Until now, the relevant information on how it all fits together has not been available. Andrews, a recognized expert, provides in-depth information about how co-verification really works, how to be successful using it, and pitfalls to avoid. He illustrates these concepts using concrete

examples with the ARM core - a technology that has the dominant market share in embedded system product design. The companion CD-ROM contains all source code used in the design examples, a searchable e-book version, and useful design tools. \* The only book on verification for systems-on-a-chip (SoC) on the market \* Will save engineers and their companies time and money by showing them how to speed up the testing process, while still avoiding costly mistakes \* Design examples use the ARM core, the dominant technology in SoC, and all the source code is included on the accompanying CD-Rom, so engineers can easily use it in their own designs

Embedded systems are informally defined as a collection of programmable parts surrounded by ASICs and other standard components, that interact continuously with an environment through sensors and actuators. The programmable parts include micro-controllers and Digital Signal Processors (DSPs). Embedded systems are often used in life-critical situations, where reliability and safety are more important criteria than performance. Today, embedded systems are designed with an ad hoc approach that is heavily based on earlier experience with similar products and on manual design. Use of higher-level languages such as C helps structure the design somewhat, but with increasing complexity it is not sufficient. Formal verification and automatic synthesis of implementations are the surest ways to guarantee safety. Thus, the POLIS system which is a co-design environment for embedded systems is based on a formal model of computation. POLIS was initiated in 1988 as a research project at the

University of California at Berkeley and, over the years, grew into a full design methodology with a software system supporting it. Hardware-Software Co-Design of Embedded Systems: The POLIS Approach is intended to give a complete overview of the POLIS system including its formal and algorithmic aspects. Hardware-Software Co-Design of Embedded Systems: The POLIS Approach will be of interest to embedded system designers (automotive electronics, consumer electronics and telecommunications), micro-controller designers, CAD developers and students. 8th International Workshop, SCOPES 2004, Amsterdam, The Netherlands, September 2-3, 2004, Proceedings

The Open-Source Approach

Chapter 2. Embedded Systems Hardware/Software Co-Development

Embedded System Design

A Unified Hardware/Software Introduction

Practical Methods for Safe and Secure Software and Systems Development

Intelligent readers who want to build their own embedded computer systems-- installed in everything from cell phones to cars to handheld organizers to refrigerators-- will find this book to be the most in-depth, practical, and up-to-date guide on the market. Designing Embedded Hardware carefully steers between the practical and philosophical aspects, so developers can both create their own devices and gadgets and customize and extend off-the-shelf systems. There are hundreds of books to choose from if you need to learn

programming, but only a few are available if you want to learn to create hardware. Designing Embedded Hardware provides software and hardware engineers with no prior experience in embedded systems with the necessary conceptual and design building blocks to understand the architectures of embedded systems. Written to provide the depth of coverage and real-world examples developers need, Designing Embedded Hardware also provides a road-map to the pitfalls and traps to avoid in designing embedded systems. Designing Embedded Hardware covers such essential topics as: The principles of developing computer hardware Core hardware designs Assembly language concepts Parallel I/O Analog-digital conversion Timers (internal and external) UART Serial Peripheral Interface Inter-Integrated Circuit Bus Controller Area Network (CAN) Data Converter Interface (DCI) Low-power operation This invaluable and eminently useful book gives you the practical tools and skills to develop, build, and program your own application-specific computers.

Introduces different tasks of hardware/software co-design, including system specification, hardware/software partitioning, co-synthesis, and co-simulation. Summarizes and classifies co-design tools and methods for these tasks, and presents the co-design tool COOL, useful for solving co-design tasks for the class of data-flow dominated embedded systems. Primary emphasis is on hardware/software partitioning and the co-synthesis phase and their coupling. A mathematical formulation of the hardware/software partitioning problem is given, and several novel approaches are presented and compared for solving the partitioning problem.

Annotation copyrighted by Book News, Inc., Portland, OR

Until the late 1980s, information processing was associated with large mainframe computers and huge tape drives. During the 1990s, this trend shifted toward information processing with personal computers, or PCs. The trend toward miniaturization continues and in the future the majority of information processing systems will be small mobile computers, many of which will be embedded into larger products and interfaced to the physical environment. Hence, these kinds of systems are called embedded systems. Embedded systems together with their physical environment are called cyber-physical systems. Examples include systems such as transportation and fabrication equipment. It is expected that the total market volume of embedded systems will be significantly larger than that of traditional information processing systems such as PCs and mainframes. Embedded systems share a number of common characteristics. For example, they must be dependable, efficient, meet real-time constraints and require customized user interfaces (instead of generic keyboard and mouse interfaces). Therefore, it makes sense to consider common principles of embedded system design. Embedded System Design starts with an introduction into the area and a survey of specification models and languages for embedded and cyber-physical systems. It provides a brief overview of hardware devices used for such systems and presents the essentials of system software for embedded systems, like real-time operating systems. The book also discusses evaluation and validation techniques for embedded systems. Furthermore, the book presents

an overview of techniques for mapping applications to execution platforms. Due to the importance of resource efficiency, the book also contains a selected set of optimization techniques for embedded systems, including special compilation techniques. The book closes with a brief survey on testing. Embedded System Design can be used as a text book for courses on embedded systems and as a source which provides pointers to relevant material in the area for PhD students and teachers. It assumes a basic knowledge of information processing hardware and software. Courseware related to this book is available at <http://ls12-www.cs.tu-dortmund.de/~marwedel>.

Authored by two of the leading authorities in the field, this guide offers readers the knowledge and skills needed to achieve proficiency with embedded software.

Embedded Systems Design with 8051 Microcontrollers

Languages for Digital Embedded Systems

Hardware and Software