Preface

Nowadays, embedded control systems are widely used in many industrial sectors such as consumer electronics like personal digital assistants, mp3 and DVD players, videogame consoles, digital cameras, fax and printers, GPS receivers, and mobile phones. In addition, many household devices such as washing machines, intelligent alarms, dishwashers, microwave ovens, new televisions are based on embedded control systems for control and surveillance. In Transport, systems from flight, maritime to automobiles, motorcycles and bikes increasingly use embedded systems especially for safety. Future electric vehicles are increasingly using embedded software to maximize efficiency and reduce pollution. Telecommunication systems employ also embedded systems from telephone switches to mobile phones at the end-user. Moreover, ventilating, heating and air conditioning systems use networked thermostats to control temperature that can change by time of day and season. Many automated systems use wired- and wireless-networking that can be used to control lights, security, surveillance, climate etc., all of which use embedded control devices for sensing and controlling. Medical equipments are increasingly using today more and more embedded control systems for vital signs monitoring, electronic stethoscopes for amplifying sounds, and various medical imaging for non-invasive internal inspections.

Although embedded control systems are widely used anywhere, their development process is not easy in several cases because a failure can be critical for the safety of human beings (e.g. air and railway traffic control, nuclear plant control, aircraft and car control). They classically should satisfy according to user requirements, functional and temporal properties, but their time to market should be shorter and shorter than ever. A new generation of these systems is addressing new criteria as flexibility and agility. To reduce their cost, these systems should be changed and adapted to their environment without disturbances. Several interesting academic and industrial research works have been made last years to develop reconfigurable embedded control systems. We distinguish in these works two reconfiguration policies: static and dynamic reconfigurations such that static reconfigurations are applied off-line to apply changes before the system cold start, whereas dynamic reconfigurations are dynamically applied at run-time. Two cases exist in the last policy: manual reconfigurations applied by users and automatic reconfigurations applied by Intelligent Agents. The system is modeled therefore in the functional level by different networks of software components such that only one network should be executed when a well-defined reconfiguration scenario is manually-automatically or statically applied. In the operational level, each network is assumed as a set of OS tasks under real-time constraints in some cases. We are interested in this book in the development of reconfigurable embedded control systems: from modeling to final deployment.

Chapter 1 deals with reconfigurable embedded control systems following component-based technologies and/or Architecture Description Languages. The author defines Control Components as software

units to support control tasks of the system which is assumed to be a network of components with precedence constraints. An agent-based architecture is proposed to handle automatic reconfigurations under well-defined conditions by creating, deleting or updating components to bring the whole system into safe and optimal behaviors. To cover all reconfiguration forms, the agent is modeled by nested state machines such that states correspond to other state machines. Several complex networks can implement the system where each one is executed at a given time when a corresponding reconfiguration scenario is automatically applied by the agent. To check the correctness of each one of them, a refinement-based approach is defined to automatically specify feasible Control Components in several steps according to NCES. The model checker SESA is automatically applied in each step to verify deadlock properties of new generated components, and is manually used to verify CTL-based properties according to user requirements. The chapter implements the reconfiguration agent by three modules that allow interpretations of environment evolutions, decisions of useful reconfiguration scenarios and finally their applications.

Embedded and Real-Time Systems (ERTS) have continued to expand at a vigorous rate. Designers of ERTS systems are continually challenged to provide new capabilities that can meet the expanding requirements and increased computational needs of each new proposed application, but at a decreasing price/performance ratio. Conventional solutions using general purpose processors or custom ASICs are less and less able to satisfy the contradictory requirements in performance, flexibility, power, development time, and cost. Chapter 2 introduces an approach to generate semi-custom platforms driven from a traditional multithreaded programming model of an embedded real-time system. This approach offers the advantage of achieving productivity levels close to those associated with software by using an established programming model, but with a performance level close to custom hardware through the use of a flexible hardware platform capable of adapting to specialized application requirements. The authors discuss the underlying concepts, requirements and advantages of multithreading in the context of reconfigurable hardware, and present two approaches which provide multithreading support to hardware and software components at the operating system level.

Future manufacturing is envisioned to be highly flexible and adaptable. New technologies for efficient engineering of reconfigurable systems and their adaptations are preconditions for this vision. Without such solutions, engineering adaptations of Industrial Process Measurement and Control Systems (IPMCS) will exceed the costs of engineered systems by far and the reuse of equipment will become inefficient. Especially the reconfiguration of control applications is not sufficiently solved by state-of-the-art technology. Chapter 3 gives an overview of the use of reconfiguration applications for zero-downtime system reconfiguration of control applications on basis of the standard IEC 61499 which provides a reference model for distributed and reconfigurable control systems. A new approach for the reconfiguration of IEC 61499 based control application and the corresponding modeling is discussed. This new method significantly increases engineering efficiency and reuse in component-based IPMCS.

Numerous research efforts in reconfigurable embedded processors have shown that augmenting a CPU core with a coarse-grained reconfigurable array for application-specific hardware acceleration can greatly increase performance and energy-efficiency. The traditional execution model for such reconfigurable co-processors however requires the accelerated function to fit onto the reconfigurable array as a whole, which restricts the applicability to rather small functions. Chapter 4 studies hardware virtualization approaches that overcome this restriction by leveraging dynamic reconfiguration. It presents two different hardware virtualization methods, virtualized execution and temporal partitioning, and introduces the Zippy reconfigurable processor architecture that has been designed with specific hardware virtualiza-

tion support. Further, the authors outline the corresponding hardware and software tool flows. Finally, they demonstrate the potential provided by hardware virtualization with two case studies and discuss directions for future research.

Reconfigurable embedded computing opens many interesting possibilities for efficiency and reliability improvement. Still, it is necessary to verify whether the reconfigurable embedded computer system fulfills its timing requirements. Purely *static worst-case execution time* (WCET) analysis is not the best choice for verifying the timing behavior of reconfigurable embedded computer systems, as frequent remodeling of different hardware configurations is not practicable. Chapter 5 describes *measurement-based timing analysis* as an approach that is better applicable for reconfigurable applications. Measurement-based timing analysis is a hybrid approach that combines static program analysis techniques, borrowed from static WCET analysis, with empirical learning of the application's timing behavior by performing systematic measurements. But even the MBTA approach is challenged by the reconfigurable computing paradigm by the need for adequate coverage of the reconfigurable fabric.

Chapter 6 presents reconfigurable embedded systems by looking closely at three different but interrelated aspects: design tools, methodologies and architectures, paying special attention at reconfigurable interconnections. The authors start by having a closer look at the evolution of the latest design strategies, tools and methodologies facing challenges and user requirements. Reconfigurable interconnections will be analyzed, examining topologies, drawbacks, and capabilities, specially focusing on the reconfiguration potential. From the application point of view, the authors resume with a case study regarding embedded systems, a Software-Defined Radio application highlighting the most significant technical features and design choices.

Chapter 7 deals with the problem of reconfiguring embedded real-time systems. Such reconfiguration can be decided either off-line to determine if a given application can be run on a different platform, while preserving the timeliness constraints imposed by the application, or on-line, where a reconfiguration should be done to adapt the system to the context of execution or to handle hardware or software faults. The task model considered in this chapter is the classical sporadic task model defined by a Worst Case Execution Time (WCET), a minimum inter-arrival time (also denoted the minimum Period) and a late termination deadline. The authors consider two preemptive scheduling strategies: Fixed Priority highest priority first (FP) and Earliest Deadline First (EDF). They propose a sensitivity analysis to handle reconfiguration issues. Sensitivity analysis aims at determining acceptable deviations from the specifications of a problem due to evolutions in system characteristics (reconfiguration or performance tuning). The chapter presents a state of the art for sensitivity analysis in the case of WCETs, Periods and Deadlines reconfigurations and study to what extent sensitivity analysis can be used to decide on the possibility of reconfiguring a system.

The continuous increase in the degree of design complexity in the design of modern digital hardware systems, come into being due to the increasing demand of more and more functionality under strict design constraints, has led to designers to try to alleviate this complexity by increasing the level of abstraction when describing the functionality of a system. Recent proposals in the field of Electronic Design Automation intend to use common programming languages, like C, C++, and Java, or dialects derived from them, to describe the behavior of a digital hardware system and then generate a lower-level representation, closer to the hardware implementation platform, from such description. This phenomenon led us to firmly believe that the process of describing the functionality of a digital circuit resembles more and more the process of developing software; and, thus, it is possible to experiment with the application of the latest trends in software engineering, like the Model-Driven Engineering (MDE) paradigm, to

design digital hardware systems. Chapter 8 describes the basic principles of MDE, and provides some hints about the kind of languages and transformation tools needed to design algorithms in the domain of digital control that could be transformed into a digital circuit. The authors intend to open doors and encourage the research on the design of digital control systems at higher levels of abstraction and their implementations in different kinds of hardware platforms, including reconfigurable devices.

Massive parallel processing systems, particularly Single Instruction Multiple Data Architectures, play a crucial role in the field of data intensive parallel applications. One of the primary goals in using these systems is their scalability and their linear increase in processing power by increasing the number of processing units. However, communication networks are the big challenging issue facing researchers. One of the most important networks on chip for parallel systems are the multistage interconnection networks. Chapter 9 proposes a design methodology of multistage interconnection networks for massively parallel systems on chip. The framework covers the design step from algorithm level to RTL. The authors first develop a functional formalization of MIN-based on-chip network at a high level of abstraction. The specification and the validation of the model have been defined in the logic of ACL2 proving system. The main objective in this step is to provide a formal description of the network that integrates architectural parameters which have a huge impact on design costs. After validating the functional model, step 2 consists in the design and the implementation of the Delta multistage networks on chip dedicated to parallel multi-cores architectures on reconfigurable platforms FPGA. In the last step, an evaluation methodology based on performance and cost metrics is proposed to evaluate different topologies of dynamic network through data parallel applications with different number of cores. The authors also show in the proposed framework that multistage interconnection networks are cost-effective high performance networks for parallel SOCs.

Chapter 10 deals with distributed multi-agent reconfigurable embedded control systems following the component-based International Industrial Standard IEC61499 in which a Function Block (abbreviated by FB) is an event-triggered software component owning data and a control application is a distributed network of Function Blocks that have classically to satisfy functional and to meet temporal properties described in user requirements. The authors define a new reconfiguration semantic where a crucial criterion to consider is the automatic improvement of the system's performance at run-time, in addition to its protection when hardware faults occur. To handle all possible cases in industry, the authors classify thereafter the reconfiguration scenarios into three forms before they define an architecture of reconfigurable multi-agent systems where a Reconfiguration Agent is affected to each device of the execution environment to apply local reconfigurations, and a Coordination Agent is proposed for any coordination between devices in order to guarantee safe and adequate distributed reconfigurations. A Communication Protocol is proposed to handle any coordination between agents by using well-defined Coordination Matrices. The authors specify both the reconfiguration agents to be modeled by nested state machines, and the Coordination Agent according to the formalism Net Condition/Event Systems (Abbreviated by NCES) which is an extension of Petri nets. To verify the whole architecture, the model checker SESA is applied in each device to verify functional and temporal properties described in the temporal logic"Computation Tree Logic", but any coordination between devices should also be checked by verifying that whenever a reconfiguration is applied in a device, the Coordination Agent and other concerned devices should react as described in user requirements.

The performances of System on Chip (SoC) and the Field Programmable Gate Array (FPGA) particularly, are increasing continually. Due to the growing complexity of modern embedded control systems,

the need of more performance digital devices is evident. Recent FPGA technology makes it possible to include processor cores into the FPGA chip, which ensures more flexibility for digital controllers. Indeed, greater functionality of hardware and system software, Real-Time (RT) platforms and distributed subsystems are demanded. In Chapter 11, design concept of FPGA based controller with Hardware/Software (Hw/Sw) co-design is proposed. It is applied for electrical machine drives. There are discussed different MultiProcessor SoC (MPSoC) architectures with Hw peripherals for the implementation on FPGA-based embedded processor cores. Hw accelerators are considered in the design to enhance the controller speed performance and reduce power consumption. Test and validation of this control system are performed on RT motor emulator implemented on the same FPGA. Experimental results, carried on a real prototyping platform, are given in order to analyze the performance and efficiency of discussed architecture designs helping to support hard RT constraints.

The recent and astonishing advances in Molecular Biology, which led to the sequencing of an unprecedented number of genomes, including the human, would not have been possible without the help of Bioinformatics. Bioinformatics can be defined as a research area where computational tools and algorithms are developed to help biologists in the task of understanding the organisms. Some Bioinformatics applications, such as pairwise and sequence-profile comparison, require a huge amount of computing power and, therefore, are excellent candidates to run in FPGA platforms. Chapter 12 discusses in detail several recent proposals on FPGA-based accelerators for these two Bioinformatics applications, highlighting the similarities and differences among them. At the end of the chapter, research tendencies and open questions are presented.

Real-time systems usually involve a subtle interaction of a number of distributed components and have a high degree of parallelism, which makes their performance analysis quite complex. Thus, traditional techniques, such as simulation, or state-based formal methods usually fail to produce reasonable results. The main limitation of these approaches may be overcome by conducting the performance analysis of real-time systems using higher-order-logic theorem proving. Chapter 13 is mainly oriented towards this emerging trend and provides the details about analyzing both functional and performance related properties of real-time systems using a higher-order-logic theorem prover (HOL). For illustration purposes, the Stop-and-Wait protocol, which is a classical example of real-time systems, has been considered as a case-study.

Chapter 14 deals with the use of two verification approaches: theorem proving and model checking. The authors focus on the Event-B method by using its associated theorem proving tool (Click_n_Prove), and on the language TLA+ by using its model checker TLC. By considering the limitation of the Event-B method to invariance properties, the authors propose to apply the language TLA+ to verify liveness properties on a software behavior. They extend first the expressivity and the semantics of a B model (called temporal B model) to deal with the specification of fairness and eventuality properties. Second, they give transformation rules from a temporal B model into a TLA+ module. The chapter presents in particular, a prototype system called B2TLA+ that supports this transformation; then these properties can be verified thanks to the model checker TLC on finite state systems. For the verification of infinite-state systems, the authors propose the use of the predicate diagrams.

In flexible manufacturing systems, deadlocks usually occur due to limited resources. To cope with deadlock problems, Petri nets are widely used to model these systems. The chapter 15 focuses on deadlock prevention for flexible manufacturing systems that are modeled with S^4R nets, a subclass of generalized Petri nets. The analysis of S^4R leads us to derive an iterative deadlock prevention approach. At each iteration step, a non-max-controlled siphon is derived by solving a mixed integer linear programming.

A monitor is constructed for the siphon such that it is max-controlled. Finally, a liveness-enforcing Petri net supervisor can be derived without enumerating all the strict minimal siphons.

Chapter 16 provides a comparative study between recent operating systems, designed for embedded systems. The study focuses, in particular, on systems designed for Multiprocessors implementations called MPSoC. An OS can be seen as abstract layer or an interface between the embedded application and the underlying hardware. In this chapter, the authors give a comparative study of main operating systems used in embedded systems. The originality of this chapter is that it specially focuses on the OS ability to be optimized to support and manage a multiprocessor architecture. A multiprocessor system-on-chip is software driven and mastering the development complexity of the software part of MPSoC, is the key to reduce developing time factor. This opportunity could be reached through the use of a document giving a detailed description and analysis for criteria related to MPSoC. The wide diversity of existing operating systems, the huge complexity to develop an application specific or a general purpose, and the aggressive evolution of embedded systems makes the development of such a system a so difficult task. These considerations lead to the realization that a work that provides guidance for the MPSoC designers will be very beneficial for these communities.

Chapter 17 presents a specification technique borrowing features from two classes of specification methods, formal and semi-formal ones. Each of the above methods have been proved to be useful in the development of real-time and critical systems and widely reported in different papers. Formal methods are based on mathematical notations and axiomatic which induce verification and validation. Semi-formal methods are, in the other hand, graphic, structural and uer-friendly. Each method is applied on a suitable case study, that the author regrets some missing features she could find in the other class. This remark has motivated the work. The chapter is interested in the integration of formal and semi-formal methods in order to lay out a specification approach which combines the advantages of theses two classes of methods. The proposed technique is based on the integration of the semi-formal method STATEMATE and the temporal logic FNLOG. This choice is justified by the fact that FNLOG is formal, deals with quantitative temporal properties and that these two approaches have a compatibility which simplifies their integration. The proposed integration approach uses the notations of STATEMATE and FNLOG, defines various transformation rules of a STATEMATE specification towards FNLOG and extends the axiomatics of the temporal logic FNLOG by new lemmas to deal with duration properties. The chapter presents the various steps of the integration approach, the proposed extensions and illustrates it over a case of critical real-time systems : the gas burner system.

The research presented in Chapter 18 deals with the design and implementation of Real-Time (RT) control systems applying advanced Field Programmable Gate Array (FPGAs). The chapter proposes a promising flexible architecture that uses RT Operating System (RTOS) and ready-to-use Intellectual Properties (IPs). The authors detail an approach that uses software closed control loop function blocks (FB), running on embedded processor cores. These FBs implement the different control drive sub-modules into RTOS tasks of the execution environment, where each task has to be executed under well defined conditions. Two RTOSes are evaluated: µC-OS/II and Xilkernel. The FPGA embedded processor cores are combined with reconfigurable logic and dedicated resources on the FPGA. This System-on-Chip (SoC) has been applied to electric motors drive. A comparative analysis, in terms of speed and cost, is carried-out between various hardware/software FPGA-based architectures, in order to enhance flexibility without sacrificing performance and increasing cost. Case studies results validate successfully the feasibility and the efficiency of the flexible approach for new and more complex control algorithms.

The performance and flexibility of FPGA-based motor controllers are enhanced with the reliability and modularity of the introduced RTOS support.

Aircraft manufacturers have been moving toward the Integrated Modular Avionics (IMA) approach to reduce the number of dedicated boxes in the aircraft. Standards such as as DO178B or ARINC 653 must be followed during design, configuration or certification of IMA systems. Productivity and costs must also be improved while preserving conformance to standards. For instance, development process of avionics systems involves several system representations and representation transformations are done manually. Moreover, the complexity of new generation of safety-critical systems has also increased the complexity of their development. Chapter 19 presents a component-based approach which relies on an appropriate modeling language (AADL) combined with modeling patterns to represent, configure and deploy an IMA system. It reduces costs by detecting errors earlier and prevents specifications revisions. The proposed code generator reduces memory footprint and improves code coverage. One last benefit is a possible automatic certification.

Finally, developing an embedded software solution can be time consuming and challenging especially for non-software trained engineers. This is because traditionally, embedded software is programmed manually in proprietary computer languages such as C, C++, Java and assembly languages, meaning that the developers have to be familiar with at least one of these languages. In addition, most of the embedded software design environments do not cater for both microprocessors-based and Field Programmable Gate Array (FPGA) based embedded computing environments, making the development process even more difficult without the assistance of a common method. The chapter 20 proposes a design of a new embedded system code generator framework which is based on the International Electrotechnical Commission (IEC) 61499 Function Block, XML and EBNF. Along with this code generator, an Iterative Knowledge Based Code Generator (IKBCG) is presented to improve the accuracy of the target codes.

These different chapters, prepared in different known research laboratories, address many interesting topics that can be considered in Industry. We hope that the scientific and technical contributions of the book will satisfy researchers, and will be useful for new generations of embedded control systems.

Mohamed Khalgui
Xidian University, China

Hans-Michael Hanisch
Martin Luther University, Germany