Preface

RATIONALE

In the last twenty five years, design technology, and the EDA industry in particular, have been very successful, enjoying an exceptional growth that has been paralleled only by advances in semiconductor fabrication. Since the design problems at the lower levels of abstraction became humanly intractable and time consuming earlier then those at higher abstraction levels, researchers and the industry alike were forced to devote their attention first to problems such as circuit simulation, placement, routing and floorplanning. As these problems become more manageable, CAD tools for logic simulation and synthesis were developed successfully and introduced into the design process. As design complexities have grown and time-to-market have shrunk drastically, both industry and academia have begun to focus on levels of design that are even higher then layout and logic. Since higher levels of abstraction reduce by an order of magnitude the number of objects that a designer needs to consider, they have allowed industry to design and manufacture complex application-oriented integrated circuits in shorter periods of time.

Following in the footsteps of logic synthesis, register-transfer and high-level synthesis have contributed to raising abstraction levels in the design methodology to the processor level. However, they are used for the design of a single custom processor, an application-specific or communication component or an interface component. These components, along with standard processors and memories, are used as components in systems whose design methodology requires even higher levels of abstraction: system level. A system-level design focuses on the specification of the systems in terms of some models of computations using some abstract data types, as well as the transformation or refinement of that specification into a system platform consisting of a set of processor-level components, including generation of custom software and hardware components. To this point, however, in spite of the fact that sys-

tems have been manufactured for years, industry and academia have not been sufficiently focused on developing and formalizing a system-level design technology and methodology, even though there was a clear need for it. This need has been magnified by appearance of embedded systems, which can be used anywhere and everywhere, in plains, trains, houses, humans, environment, and manufacturing and in any possible infrastructure. They are application specific and tightly constrained by different requirements emanating from the environment they operate in. Together with ever increasing complexities and market pressures, this makes their design a tremendous challenge and the development of a clear and well-defined system-level design technology unavoidable.

There are two reasons for emphasizing more abstract, system-level methodologies. The first is the fact that high-level abstractions are closer to a designer's usual way of reasoning. It would be difficult to imagine, for example, how a designer could specify, model and communicate a system design by means of a schematic or hundred thousand lines of VHDL or Verilog code. The more complex the design, the more difficult it is for the designer to comprehend its functionality when it is specified on register-transfer level of abstraction. On the other hand, when a system is described with an application-oriented model of computation as a set of processes that operate on abstract data types and communicate results through abstract channels, the designer will find it much easier to specify and verify proper functionality and to evaluate various implementations using different technologies. The second reason is that embedded system are usually defined by the experts in application domain who understand application very well, but have only basic knowledge of design technology and practice. System-level design technology allows them to specify, explore and verify their embedded system products without expert knowledge of system engineering and manufacturing.

It must be acknowledged that research on system design did start many years ago; at the time, however, it remained rather focused to specific domains and communities. For example, the computer architecture community has considered ways of partitioning and mapping computations to different architectures, such as hypercubes, multiprocessors, massively parallel or heterogeneous processors. The software engineering community has been developing methods for specifying and generating software code. The CAD community has focused on system issues such as specification capture, languages, and modeling. However, simulation languages and models are not synthesizable or verifiable for lack of proper design meaning and formalism. That resulted in proliferation of models and modeling styles that are not useful beyond the modeler's team. By introduction of well-defined model semantics, and corresponding model transformations for different design decision, it is possible to generate models automatically. Such models are also synthesizable and verifiable. Furthermore, model automation relieves designers from error-prone model coding and even

learning the modeling language. This approach is appealing to application experts since they need to know only the application and experiment with a set of design decisions. Unfortunately, a universally accepted theoretical framework and CAD environments that support system design methodologies based on these concepts are not commercially available yet, although some experimental versions demonstrated several orders of magnitude productivity gain. On the other hand, embedded-system design-technology based on these concepts has matured to the point that a book summarizing the basic ideas and results developed so far will help students and practitioners in embedded system design.

In this book, we have tried to include ideas and results from a wide variety of sources and research projects. However, due to the relative youth of this field, we may have overlooked certain interesting and useful projects; for this we apologize in advance, and hope to hear about those projects so they may be incorporated into future editions. Also, there are several important system-level topics that, for various reasons, we have not been able to cover in detail here, such as testing and design for test. Nevertheless, we believe that a book on embedded system techniques and technology will help upgrade computer science and engineering education toward system-level and toward application oriented embedded systems, stimulate design automation community to move beyond system level simulation and develop system-level synthesis and verification tools and support the new emerging embedded application community to become more innovative and self-sustaining.

AUDIENCE

This book is intended for four different groups within the embedded system community. First, it should be an introductory book for application-product designers and engineers in the field of mechanical, civil, bio-medical, electrical, and environmental, energy, communication, entertainment and other application fields. This book may help them understand and design embedded systems in their application domain without an expert knowledge of system design methods bellow system-level. Second, this book should also appeal to system designers and system managers, who may be interested in embedded system methodology, software-hardware co-design and design process management. They may use this book to create a new system level methodology or to upgrade one existing in their company. Third, this book can also be used by CAD-tool developers, who may want to use some of its concepts in existing or future tools for specification capture, design exploration and system modeling, synthesis and verification. Finally, since the book surveys the basic concepts and principles of system-design techniques and methodologies, including software and hardware, it could be valuable to advanced teachers and academic

programs that want to teach software and hardware concepts together instead of in non-related courses. That is particularly needed in today's embedded systems where software and hardware are interchangeable. From this point, the book would also be valuable for an advanced undergraduate or graduate course targeting students who want to specialize in embedded system, design automation and system design and engineering. Since the book covers multiple aspects of system design, it would be very useful reference for any senior project course in which students design a real prototype or for graduate project for system-level tool development.

ORGANIZATION

This book has been organized into eight chapters that can be divided into four parts. Chapter 1 and 2 present the basic issues in embedded system design and discuss various system-design methodologies that can be used in capturing system behavior and refining it into system implementation. Chapter 3 and 4 deal with different models of computations and system modeling at different levels of abstraction as well as system synthesis from those models. Chapter 5, 6, and 7 deal with issues and possible solutions in synthesis and verification of software and hardware component needed in a embedded system platform. Finally, Chapter 8 reviews the key developments and selected current academic and commercial tools in the field of system design, system software and system hardware as well as case study of embedded system environments.

Given an understanding of the basic concepts defined in Chapter 1 and 2, each chapter should be self-contained and can be read independently. We have used the same writing style and organization in each chapter of the book. A typical chapter includes an introductory example, defines the basic concepts, it describes the main problems to be solved. It contains a description of several possible solutions, methods or algorithms to the problems that have been posed, and explains the advantages and disadvantages of each approach. Each chapter also includes relationship to previously published work in the field and discusses some open problems in each topic.

This book could be used in several different courses. One course would be for application experts with only a basic knowledge of computers engineering. It would emphasize application issues, system specification in application oriented models of computation, system modeling and exploration as presented in Chapter 1 - 4. The second course for embedded system designers would emphasize system languages, specification capture, system synthesis and verification with emphasis on Chapter 3, Chapter 4, and Chapter 7. The third course may emphasize system development with component synthesis and tools as described in Chapter 5 - Chapter 8. In which ever it is used, though, we feel that

http://www.springer.com/978-1-4419-0503-1

Embedded System Design
Modeling, Synthesis and Verification
Gajski, D.D.; Abdi, S.; Gerstlauer, A.; Schirner, G.
2009, XXV, 352 p., Hardcover
ISBN: 978-1-4419-0503-1