SPECIFICATION AND DESIGN OF EMBEDDED SYSTEMS

by

Daniel D. Gajski
Frank Vahid
Sanjiv Narayan
Jie Gong

University of California at Irvine

1994

Contents

Preface

Rationale

In the last ten years, VLSI design technology, and the CAD industry in particular, have been very successful, enjoying an exceptional growth that has been paralleled only by the advances in IC fabrication. Since the design problems at the lower levels became humanly intractable and time consuming earlier than 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 became 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 requirements have shrunk drastically, both industry and academia have begun to focus on levels of design that are even higher than logic and layout. Since these 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 specific integrated circuits (ASICs) in shorter periods of time.

Following in the footsteps of logic synthesis, behavioral synthesis has contributed to raising the abstraction levels in the design methodology. Behavioral synthesis, however, is used for the design of single ASICs. These ASICs, along with standard processors and memories, are used as components in systems whose design methodology requires even higher levels of abstraction. A system-level methodology focuses on the specification of systems in terms of computations to be executed on abstract data types, as well as the transformation or refinement of that specification into a set of connected components, including compiling software for standard processors and synthesizing hardware for custom components. To this point, however, in spite of the fact that systems have been manufactured for years, industry and academia have not been sufficiently focused on developing and formalizing a system-level methodology, even though there is a clear need for it. In order to manage complexity and shorten design cycles, industry has recently focused on developing a coherent system-level design methodology.

The main reason for emphasizing more abstract, system-level methodology is the fact that high-level abstractions are closer to a designer's usual way of thinking. It would be difficult to imagine, for example, how a designer could specify, document and communicate a system design by means of a circuit schematic with 100,000 gates, or a logic description with 100,000 Boolean expressions. The more complex the design, the more difficult it is for the designer to comprehend its functionality when it is specified with circuit, logic or register-level schematics. On the other hand, when a system is described as a series of complex computations 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.

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 mapping computations and algorithms to different architectures, such as systolic arrays, hypercubes, multiprocessors, and massively parallel processors. The software engineering community has been developing methods for specifying and engineering software code. The CAD community has focused on system issues such as interface synthesis, memory management, specification capture and design exploration. However, many problems still remain open, the most important of which are the lack of a universally accepted theoretical framework and the lack of CAD environments that support system design methodologies. In spite of these open problems, system design technology has matured to the point that a book summarizing the basic concepts and results developed so far will help students and practitioners in system design. In this book, we have tried to include ideas and results from a wide variety of 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, including formal verification, design for test, and cosimulation. Nevertheless, we believe that a book on system specification and design will help the electronic system design automation (ESDA) community to grow and prosper in the future.

Audience

This book is intended for three different groups within the computer science and engineering communities. First, it should appeal to system designers and engineering managers, who may be interested in ASIC and system design methodology, software-hardware codesign and design process management. Second, 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 and refinement. Finally, since this book surveys the basic concepts in system design and presents the principles of system-design methodologies, including software and hardware, it could also be valuable for an advanced undergraduate or graduate course targeting students who want to specialize in computer architecture, design automation and/or software engineering.

Textbook Organization

This book has been organized into nine chapters that can be divided into four parts. Chapters 1 and 2 present the basic issues in system design and discuss various conceptual models that can be used in capturing system behavior and its implementation. Chapters 3, 4, and 5 deal with the languages used for specifying system functionality, as well as with the different issues involved in verifying a system's functionality through simulation. Chapters 6, 7, and 8 provide a survey of algorithms and techniques for system partitioning, estimation and model refinement, and Chapter 9 combines all of these topics into a consistent design methodology, including a discussion of the general environments for system design.

Given an understanding of the basic concepts defined in Chapters 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 and describes the main problems to be solved. It contains a description of several well-known algorithms or solutions to the problems that have been posed, and explains the advantages and disadvantages of each approach. Each chapter also includes a short survey of other work in the field and some open problems.

At the end of each chapter we have included several exercises, which are divided into three categories: homework problems, project problems and thesis problems. The homework problems are designed to test the reader's understanding of the basic material in the chapter. To solve the project problems, indicated by an asterisk, the reader will need a more thorough understanding of the topic based on some literature research; these problems may require several weeks of student work. The thesis problems, indicated by a double asterisk, are open problems that could result in an M.S. or even a Ph.D. thesis if researched thoroughly.

This book could be used in two different courses. One course, for example, could concentrate on system specification, documentation, and verification, omitting the algorithms in Chapter 6, 7, and 8. A second course could emphasize design methodology and design-exploration techniques, omitting the material on languages and simulation. In which ever way it is used, though, we feel that this book will help to fill the vacuum in computer science and engineering curricula where we should be teaching system design techniques in addition to covering material on circuit and logic design, and computer architecture.

We hope that the material selection and the writing style will approach your expectations; we welcome your suggestions and comments.

Daniel Gajski, Frank Vahid, Sanjiv Narayan, Jie Gong
Irvine, California
1994