1 ESSENTIAL ISSUES IN CODESIGN

Daniel D. Gajski,
Jianwen Zhu,
and Rainer Dömer

Department of Information and Computer Science
University of California, Irvine
Irvine, California, USA

1.1 MODELS

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 of abstraction became humanly intractable earlier than those at higher abstraction levels, researchers and the industry alike were forced to devote their attention first to lower-level problems such as physical and logic design. 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 system levels of design since they reduce the number of objects that a designer needs to consider by an order of magnitude and thus allow the design and manufacturing of complex application specific integrated circuits (ASIC) quickly.

This chapter introduces key concepts in codesign: models of computations, hardware architectures, required language features for describing embedded systems and a generic methodology for codesign. The other chapters in this book give a more detailed survey and explore models (Chapter 6), languages (Chapter 7), architectures (Chapter 4), synthesis algorithms (Chapter 2), compilers (Chapter 5), and prototyping and emulation of embedded systems (Chapter 3). The last three chapters provide details of three research systems that demonstrate several aspects of codesign methodology.

1.1.1 Models and Architectures

System design is the process of implementing a desired functionality using a set of physical components. Clearly, the whole process of system design must begin with specifying the desired functionality. This is not, however, an easy task.

The most common way to achieve the level of precision we need in specification is to think of the system as a collection of simpler subsystems, or pieces, and the method or the rules for composing these pieces to create system functionality. We call such a method a model. To be useful, a model should possess certain qualities. First, it should be formal so that it contains no ambiguity. It should also be complete, so that it can describe the entire system. In addition, it should be comprehensible to the designers who need to use it, as well as being easy to modify, since it is inevitable that, at some point, they will wish to change the system’s functionality. Finally, a model should be natural enough to aid, rather than impede, the designer’s understanding of the system.

It is important to note that a model is a formal system consisting of objects and composition rules, and is used for describing a system’s characteristics. Typically, we would use a particular model to decompose a system into pieces, and then generate a specification by describing these pieces in a particular language. A language can capture many different models, and a model can be captured in many different languages (see Chapter 7).

Designers choose different models in different phases of the design process, in order to emphasize those aspects of the system that are of interest to them at that particular time. For example, in the specification phase, the designer knows nothing beyond the functionality of the system, so he will tend to use a model that does not reflect any implementation information. In the implementation phase, however, when information about the system's components