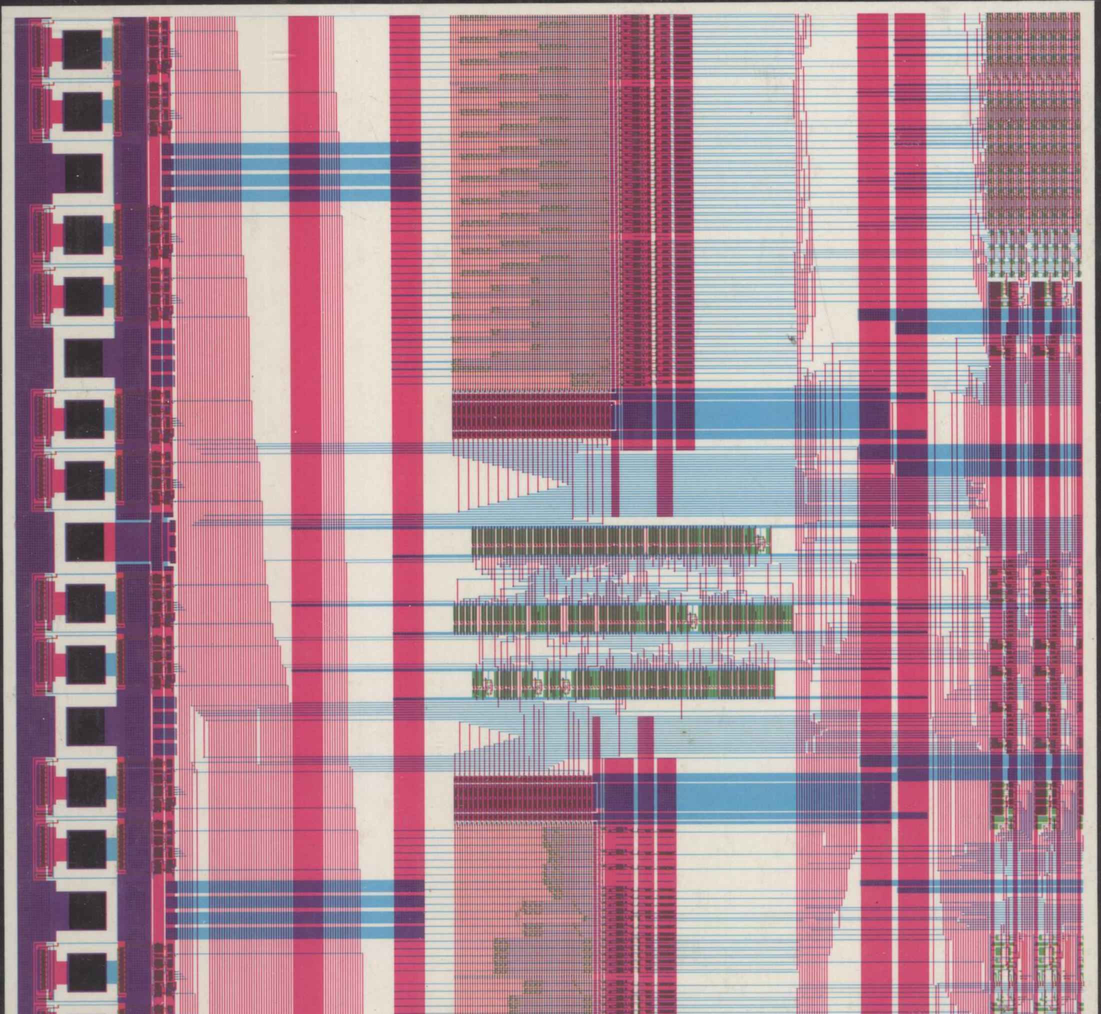
Edited by
Bryan Preas
and
Michael Lorenzetti

Physical Design Automation of VLSI Systems



TN47
P578

Physical Design Automation of VLSI Systems

Edited by Bryan T. Preas and Michael J. Lorenzetti

Bryan D. Ackland
AT&T Bell Laboratories

D. Scott Baeder
Digital Equipment Corporation

William E. Donath
IBM T. J. Watson Research Center

Alfred E. Dunlop
AT&T Bell Laboratories

Daniel D. Gajski
University of California, Irvine

Donald L. Hanson
Integrated CMOS Systems

Patrick G. Karger
Tektronix/CAE Systems Division

Thomas Lengauer
University of Paderborn

Y-L. Steve Lin
Tsing Hua University

Michael J. Lorenzetti
Microelectronics Center of North Carolina

Bryan T. Preas
Xerox Palo Alto Research Center

Thomas G. Szymanski
AT&T Bell Laboratories

Christopher J. Van Wyk
AT&T Bell Laboratories

Wayne H. Wolf
AT&T Bell Laboratories

jamin/Cummings [illegible] pany, In
Hill Road
Californ

Sponsoring Editor: **Alan Apt**
Production Supervisor: **Karen Gulliver**
Production Coordinator: **Sharon Montooth**

LIBRARY OF CONGRESS
Library of Congress Cataloging-in-Publication Data

```
Physical design automation of electronic systems / edited by Bryan T.
  Preas and Michael J. Lorenzetti ; Bryan D. Ackland ... [et al.].
       p.   cm.
    Includes index.
    ISBN (invalid) 0-8053-0412-9
    1. Integrated circuits--Design and construction--Data processing.
  2. Computer-aided design.   I. Preas, Bryan T.   II. Lorenzetti,
  Michael J.  III. Ackland, Bryan D.
  TK7874.P47 1988
  621.381'73--dc19                                            88-14479
                                                                  CIP
```

ISBN 0-8053-0142-9

BCDEFGHIJ-HA-898

The Benjamin/Cummings Publishing Company, Inc.
2727 Sand Hill Road
Menlo Park, California 94025

Preface

The field of design automation has received increasing attention and commercialization in the last two decades but has produced relatively few books. A body of knowledge exists in journals and conference proceedings, but only a few comprehensive (and now out-of-date) books have been produced. As a result, it is difficult for people new to the field to know what has happened before and to avoid reinventing old ideas and duplicating old results. Several universities offer graduate courses in design automation (DA) for VLSI, but until now no suitable textbook for physical design existed. This book fills that void.

Goals of the Book

This book is intended as a textbook for an upper division or first-year graduate course in VLSI physical design automation or as a reference book for the practicing professional. The important aspects are covered: synthesis, analysis and verification, knowledge-based techniques, and combinatorial complexity. We provide a review of existing techniques, a guide to related literature, and an in-depth discussion of the state of the art. Each chapter covers a particular aspect of physical design automation. The individual chapters can be studied independently or as part of a larger, more comprehensive course that includes all of physical design automation. Advances are put into historical and taxonomic perspective. We have collected and described algorithms to give the reader a perspective on the state of the art in each area. We have avoided detailed descriptions, data structures, and implementation details in favor of concepts and algorithms. Each chapter discusses applications of algorithms; the examples are drawn from MOS ICs.

Taken as a whole, this book represents a survey of electronic system, physical design methods for a wide range of design styles. To the greatest possible extent, a technology-independent presentation has been maintained so that the algorithms and techniques are applicable to a wide range of fabrication processes.

The area of physical design automation is too large to be covered by a single author to the depth necessary for a book such as this. Unifying a book by many authors is always a challenge, and we are indebted to the authors for the many changes necessary to produce a consistent presentation. The authors are recognized experts in their individual fields. Their opinions are sometimes subjective but well informed.

We cover *physical* design automation in detail but do not address other aspects of VLSI design automation such as design capture, simulation and test. This is not a book about "advances" but rather a review of the current state of the art. Neither is this a cookbook. This book is not an implementation guide for layout systems; rather, it is a means to understanding existing techniques and systems. The many references will permit readers to use this book as a guide to and review of the existing literature.

Use of the Book and the Exercises

When used as a textbook, we suggest that students be familiar with digital design and have a strong computer science background. The course, as we envision it, would be tutorial in nature. This text should be supplemented with some of the references when a more in-depth study of a particular topic is desired. For such a course, this book provides definitions of problems, a taxonomy of the solution space, and a perspective on previous solutions.

Some of the exercises at the end of each chapter test understanding while others extend the concepts presented in the chapters. The exercises are divided into three levels of difficulty.

- Questions to test basic understanding of the material covered in the chapter. These exercises have no special marking.
- Problems of intermediate difficulty. These problems are preceded by a dagger (†).
- Suggested projects which take a week or more to complete. These projects are preceded by a double dagger (‡).

When used as a reference by the practicing professional, this book provides a survey of the state of the art with editorial comments and an ample list of references for further research into individual topics.

Acknowledgements

The editors wish to acknowledge the Xerox Palo Alto Research Center (PARC) and the Microelectronics Center of North Carolina (MCNC) for the use of their equipment in the preparation of this book and for allowing us the time required to complete it. Without their continued support, this book could not have been written. In particular, Bob Ritchie provided enthusiastic encouragement and both tangible and intangible support.

The Association for Computing Machinery Special Interest Group on Design Automation (ACM SIGDA) recognizes the need for up-to-date texts and references in design automation and provided funds to improve the quality of this book. We used the funds for editing, formatting, checking references, and proofreading the manuscript. Chuck Shaw's support is appreciated.

Wayne Wolf and Lisa Kelly did an excellent job of unifying the text and presentation across the chapters. Special thanks go to Lorna Fear of Xerox PARC for her tireless efforts and long hours in managing the production of the book, as well as providing camera-ready copy. Additional thanks go to Jeri Williams of MCNC and Kathy Preas for their time and effort in editing and reformatting the individual chapters. Jeri Williams also checked each of the references. We are also grateful to Giordano Beretta for producing the color plates and the cover art, to Maureen Stone for color consultation, to Peter Kessler for help with the typesetting programs, to Tim Diebert for help with the printers and typesetters, and to Bridget Scamporrino for the typesetting.

Many of our colleagues have assisted us by reviewing and commenting on early drafts; we are especially indebted to William Dees of Automated Design Enterprises and Jim Cohoon of the University of Virginia for the time that they devoted to the project. Jim Cohoon did an excellent job of unifying the exercises. Lissy Bland, Ed Dorsey, Louis Monier and Wayne Stewart all made helpful comments.

Bryan T. Preas
Palo Alto, California

Michael J. Lorenzetti
Raleigh, North Carolina

Contents

Chapter 1

Introduction to Physical Design Automation

Bryan T. Preas

Xerox Palo Alto Research Center
Palo Alto, California

1.1 Introduction

This book focuses on the automation of the physical design process for electronic circuits and describes physical design methods for a wide range of design styles. In this chapter, physical design automation is introduced by defining automation and the physical design process within the context of electronic system design. This establishes a framework for the remaining chapters in the book.

Section 1.2 provides an overview of electronic system design by describing the design process and discussing broad categories of design automation tools. Section 1.3 further places physical design within the context of electronic system design by providing a prescription for the design process. Section 1.4 focuses on physical design through a brief introduction to fabrication methods and provides an overview of the synthesis, analysis, and verification methods for physical design. Design styles, the focus of Section 1.5, are a restriction to a particular class of circuit structures and are defined by a combination of the fabrication methods and physical design methods described in Section 1.4. The physical design methods are further illustrated through an example circuit produced by a modern design automation system; this example is presented in Section 1.6. The organization of this book is the subject of Section 1.7.

1.2 Automating the Design Process

The creation of large, complex electronic systems has grown beyond the capabilities of any number of people without computer support; successful completion of large design projects requires that computers be used in virtually all aspects of the design. This trend toward automation will accelerate as improved circuit fabrication technologies permit higher levels of integration and as more powerful computers allow more sophisticated tools.

While we cannot yet specify a large, high-performance computer system and automatically produce its design, more restricted goals are well within the state of the art. Computer-based, design automation (DA) tools enable designs that are too large or complex to undertake otherwise, shorten design time, improve product quality (performance and reliability), and reduce product costs. To understand the impact of these tools, let's first look at the elements in the design process.

The design process can be viewed as a sequence of transformations on behavioral, structural and physical design representations, at various levels of abstractions. For example, *functional design* defines the behavior of system outputs as a function of the inputs; *logic design* manipulates structural descriptions (schematic or logic diagrams) while preserving functionality. *Physical design* involves either transformations on or transformations into information used in the manufacture of physical systems.

The DA tools used in each of these transformations can be placed into the following general categories: management of design information and flow; synthesis; analysis; or verification and validation. The first category, *management of design information and flow*, provides the foundation on which DA systems are built. This foundation must be flexible and adaptable to new tools, design styles, and fabrication technologies. A design management system and paradigm for accessing design data and maintaining consistency are described in [Kat85]. The latter three categories transform, analyze or verify the design representations.

Synthesis creates new representations, or provides refinements to existing representations, for objects being designed. For example, automatic placement and routing systems create new representations of layouts from structural representations such as lists of components that comprise the design and lists of pins on the components to be connected. As an alternative to creating new representations, refinement operations involve optimizations to improve quality or reduce cost. These optimizations may be either *combinatorial* (for example, cell placement improvement or wiring compaction) or *parametric* (for example, sizing transistors to improve performance).

Analysis operations evaluate the consistency or correctness of design representations. An example of an analysis tool is a design rule checker; it checks physical layout representations for geometric rule violations.

Verification operations provide a formal process for demonstrating the equivalence of two design representations under all specified conditions, while validation is an informal, less rigorous correctness check.

1.3 Electronic System Design

This section places physical design within the larger context of system design by providing an overview of the design process for electronic systems. Automation of the design process is enabled by extensive use of computer design tools. Effective use of these programs requires a structured approach; otherwise, the synthesis, analysis and verification tools would be unmanageable. From the perspective of physical design, the important aspects of electronic design are the following:

- Design representations.
- The design process — to define the limits and interfaces of physical design.
- The degree of automation in the design process.
- A structured design method.

Each of these elements is described in the remainder of this section.

Design Representations

During the design process several different *representations*, or *views*, are used to show different aspects of the system under design. These views are classified as behavioral, structural and physical, and represent various levels of abstraction [New81, Gaj83 and Chapter 7].

Behavioral representations describe a circuit's function. Procedural descriptions (for example, **if** *clock* = *high* **then** *counter* := *counter* + 1) and Boolean expressions (for example, *out* := *a*b* + *c*) are behavioral representations; they say nothing about implementation. Some hardware description languages are purely behavioral representations while others include several views.

Structural representations describe the composition of circuits in terms of *cell*s (abstractions of circuit element definitions) and *component*s (abstractions of instances of circuit elements) and interconnections among the components. Such descriptions are usually hierarchical; a component at one level may be decomposed into constituent components until elementary (primitive or leaf) components are reached. Examples of structural descriptions are block diagrams, schematic drawings, and net lists of logic gates. Structural representations say nothing about functionality of a circuit except what can be inferred from the behavior of the components given the structure.

Physical representations are characterized by information used in the manufacture or fabrication of physical systems. These representations only implicitly describe how a circuit behaves. Physical information may be either a geometric layout (for example, location of transistors or wiring on a silicon surface) or a topological constraint (for example, a higher order cell of a counter should be placed to the left of a lower order cell).

The Electronic Design Process

Electronic design is carried out in many ways by various designers for a variety of purposes; it is impossible to describe one methodology that applies in all cases. Instead of attempting to provide a comprehensive description, this section provides a prescription for the design process that places physical design in perspective and defines the interfaces of physical design with other design phases. This general method is adopted with variations by different designers.

A top-down design methodology [Lat81] divides the design process into phases. As shown in Figure 1.1, the phases are design specification, functional design, logic design, circuit design, and physical design. Each design phase is further divided into three steps consisting of synthesis, analysis, and verification. The generic design steps within the phases are discussed next; this is followed by a discussion of the design phases.

Steps Within the Design Phases. Each design phase is characterized by synthesis, analysis and verification steps as shown in Figure 1.2. *Synthesis* derives a new or improved representation based on the representation derived in the previous phase. At lower levels of abstraction, synthesis is typically automatic; at higher levels of abstraction, synthesis is a topic of intense research, but computer-aided synthesis is still the order of the day.

Analysis follows synthesis in each design phase and generally takes two forms. First, a design representation must be evaluated against its requirements. For very large-scale integrated (VLSI) circuits the requirements are usually specified in terms of die size, performance, and power consumption. A design must also be evaluated for behavioral, structural and physical correctness and completeness. Demonstrating that a design representation meets these criteria can be very difficult. The criteria are analyzed many times and the process is often forced to backtrack when constraints cannot be met.

Verification, the final step within a design phase, demonstrates that the synthesized representation is equivalent under all conditions of interest to another representation. An example of verification is the exhaustive comparison of two representations such as a schematic diagram and the physical layout synthesized from the schematic. In some cases verification is not possible, and validation, a semiformal process, is substituted. *Validation* demonstrates the equivalence of representations, in behavior or structure, under restricted conditions. For example, a logic model is validated against a functional model if both produce the same sequence of outputs for the same sequence of inputs. Validation may also involve test cases or worst case models that stress a design representation to its worst case limits.

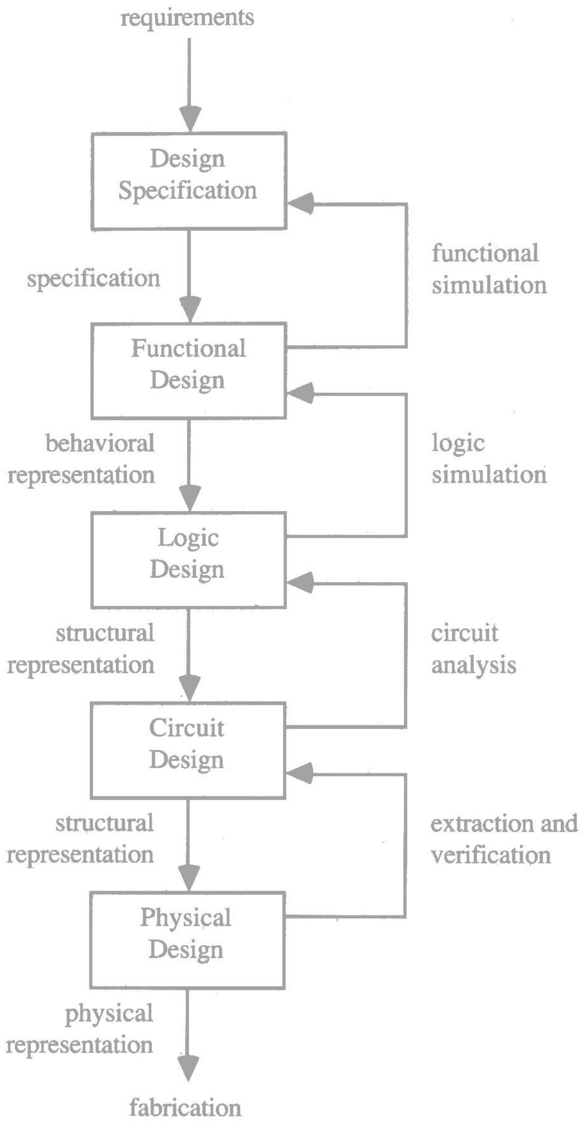


Figure 1.1 The phases of electronic system design for a top-down design methodology. This prescription provides a structure for, and defines the boundaries of, physical design automation.

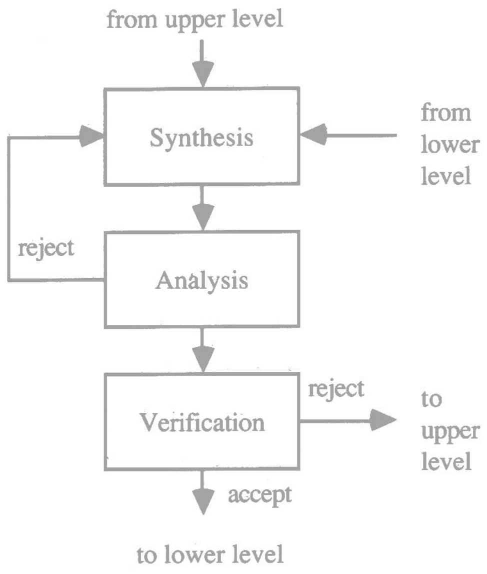


Figure 1.2 Each of the design phases of Figure 1.1 is composed of synthesis, analysis and verification steps. Sometimes verification is too difficult and validation is substituted.

The Phases of Electronic Design. Now that the steps within the design phases have been described, the design phases of the top-down process are described. The first design phase, *specification*, is a time-consuming, normally manual step. Important factors to be considered are the application of the system, the performance required to meet the application, the architecture of the system, the external interfaces and protocols, the competitive market, the product cost targets, the choice of manufacturing technology, and the design tools that are available. Particular attention must be paid to the design methodology, its impact on the cost of the design, and the time required to complete the design.

Functional design follows specification. In this phase, a functional behavior is synthesized to meet the specifications. The result may be a purely behavioral representation (for example, an instruction set description or a timing diagram), or it may include structural aspects by partitioning functionality into components. Behavioral simulation is the normal method of analysis.

Logic design, the next phase, concerns the logic structure that implements the functional design. The design representations may be either a textual, register transfer level (RTL) description or a graphic, schematic description. For analysis, these representations are simulated at the transistor, gate, or register level. The logic design is validated by comparing the results from the logic level and behavioral level simulations.

The *circuit design* phase concerns the electrical laws that govern the detailed behavior of the basic circuit elements such as transistors, resistors, capacitors and inductors. In this phase, transistors are sized to meet signal delay requirements. Automatic synthesis tools are available to determine the transistor sizes. Analysis is performed using circuit and timing simulations. Timing verification tools can determine if signal delay specifications are met.

In the *physical design* phase, the behavioral or structural representations from the previous phases are transformed into the geometric shapes that are used in the fabrication of the system. Automatic synthesis, analysis and verification tools for this phase are widely available. Physical design is discussed in greater depth in Section 1.4, and on a larger scale, it is the subject of this book.

Automation in the Design Process

Computers play a major role in electronic design, but the degree of automation varies widely. The extremes in automation are at one end of the spectrum, handcrafted design (where the designer supplies the intellectual content or inspiration of the design process), and at the other end of the spectrum, completely automated design (where a computer supplies the inspiration as well as the perspiration of the design process). Computer-aided design (CAD) falls between these extremes. The following descriptions do not represent discrete classifications but, instead, represent points on a continuous spectrum.

Handcrafted Design. Handcrafted design describes a less constrained method of design. This is the oldest method and is rarely used today to create large designs; its use is relegated to creating small cells or to finishing any incomplete wiring left by automatic routing. Even though the term implies manual operation, it also applies to the intellectual content of the design process. Generally speaking, the term is applied to the design method where a computer is used as a high-speed drafting machine and the designer provides all the creativity.

Computer-Aided Design. Computer-aided design systems provide tools to synthesize, analyze and verify portions of the design but require active participation of the designer. The designer directs and sequences the tools to complete the design. He or she must supply a degree of creativity, but the CAD system frees him or her from tedious tasks.

Design Automation. Given an abstract specification of an object to be designed, a design automation system generates the physical design automatically and verifies that the design conforms to the specification. Currently, most of the individual areas of physical design are understood, but no complete design automation system exists. Even though much research remains, promising results are available. Design automation also encompasses correct-by-construction concepts or verification of correctness of the design. Design automation and computer-aided design differ in the degree to which the designer supplies the intellectual content of the design process.

A Structured Design Method

A structured design method is of major importance for automating the design process. The design method discussed in this section permits the design to evolve rationally as a modular system. Describing the design as a formal hierarchy leads to well-defined interactions among the components of the design system, and produces a simplified design. From the perspective of physical design, hierarchical design provides the framework that makes automation possible and enables design styles, which are the subject of Section 1.5.

A structured design method enables consistent descriptions in the three design representations (behavioral, structural and physical) at all relevant levels of abstraction. Consistent descriptions are imperative for design automation systems to work. Furthermore, because the design of large electronic systems is so complicated, a structured design method provides the simplification necessary to complete the design activity. Good methods for simplification are abstraction, regularity and standardization. These simplification methods are described below. Through standardization, developers of design automation systems can create efficient tools; abstraction and regularity allow system designers to reduce complexity by working with simpler objects. More comprehensive discussions of managing the complexity of large electronic system designs are available [Seq83, Nie83].

Abstraction. Abstraction replaces an object with a simplified model that defines the interaction of the object with its environment. Any details of internal organization or implementation are deleted. Abstraction reduces the amount of data needed to describe an object. However, for large systems, one level of abstraction is insufficient to reduce objects to manageable sizes. Thus, hierarchical abstraction or *hierarchy* is required. Hierarchy involves decomposing a system into a set of components. These components are recursively decomposed into subcomponents until all components are small enough to manipulate. In order for a structural, hierarchical decomposition to be of greatest value, the abstraction process must follow certain rules [Seq83]. Two of the most important of these rules are modularity and locality.