Computer-Aided Logic Design

Robert M. McDermott

Computer-Aided Logic Design

Robert M. McDermott

Howard W. Sams & Co., Inc.
A Subsidiary of Macmillan, Inc.
4300 West 62nd Street, Indianapolis, Indiana 46268 U.S.A.

FIRST EDITION
FIRST PRINTING-1985

International Standard Book Number: 0-672-22436-4
Library of Congress Catalog Card Number: 85-72105

Production supervised by White River Press, Inc.

Printed in the United States of America

Preface

The environment for electronics design is rapidly changing. In years past, electronics design was characterized by large lab areas for experimenting, prototyping, and testing, and by large drafting areas for schematic drawing and documentation. In modern electronic design facilities, however, the scene is one of engineers and technicians using computers to perform experimentation, prototyping, testing, and other tasks with the aid of *computer-aided design* (CAD) tools and systems.

Until recently, texts and courses on digital design were accompanied by lab courses in which students would experiment and verify their logic concepts by building models and prototypes. They would then spend additional hours describing and documenting their accomplishments. In today's universities, colleges, and technical schools, the trend is toward having students prototype and verify their designs using computer-based analysis programs, with hard-copy printouts to document their results.

Few schools, however, can afford the major capital investment (in the hundreds of thousands of dollars) required to provide a computer-aided design facility. Also, few texts are available to couple formal training to hands-on experience. Self-educated technicians or hobbyists are unable to afford commercial CAD systems, and even the relatively expensive and sophisticated correspondence courses in electronics or computer technology rely on the age-old approach of hardware breadboarding. Such schools or training programs run the risk of producing engineers and technicians whose skills are outdated even before they graduate.

This book is intended to couple the theory and techniques of digital design with the application and use of CAD tools (specifically, *logic simulation* and *logic minimization*). To provide hands-on experience with such CAD tools at a minimal cost, this book also includes logic simulation and logic minimization programs which can be run on a low-cost personal computer. These programs will not eliminate the need for hardware breadboarding and hands-on prototyping and verification, any more than commercial programs eliminate the need for first prototyping in professional design centers. But, as is the case in professional design centers, the programs

will allow you to perform the first levels of design analysis and verification (often the most difficult task when using the hardware breadboarding concept). They will also provide the exposure to CAD that is increasingly necessary for professional employment and career growth.

Robert M. McDermott

Contents

1

COMPUTER-AIDED DESIGN

1.1 CHAPTER OVERVIEW

Computer-Aided Design (CAD) has become a reality in all areas of electronics design work. Computers are used at each stage of the design process from schematic drawing to logic and circuit analysis to automated layout of complex electronic systems. This chapter will provide a brief overview of some of the CAD tools for the electronic design of printed circuit boards (PCBs) and very large scale integrated (VLSI) circuits.

1.2 THE DESIGN PROCESS

Design engineering embodies all the tasks required to solve a problem subject to various constraints. Recognition of a problem may be as broad as the observation that the crime rate in the U.S. is increasing. From such a problem statement, the need for a potential engineered solution to the problem (or a subset of the problem) is identified: for example, a system to protect one's property.

If this process takes place within a "for profit" organization, a market study is conducted to determine the need for a solution, the specific functions required, and the feasible market price. The conclusions of the study might indicate that property owners acknowledge the need for a burglar alarm, and would be willing to pay a given price for a system with certain features. On an individual scale, the same steps are performed, but not necessarily in such a formal manner.

At this point, the actual "design process" commences. This is the process of determining a feasible solution that meets the requirements of the problem. The requirements typically include both *explicitly* stated functions and specifications and *implied* functions and capabilities. Assumed characteristics based on accepted engineering standards will also normally be part of the

requirements. These requirements place specific constraints on the potential solution, such as cost, size, power, or environmenal considerations. The state of technology also imposes constraints, in terms of practical solutions. The engineering task involves trading off among various alternatives until a solution is identified which fits within the various constraints and still satisfies the functional requirement. Typical "trade-offs" are cost vs. speed and speed vs. power. (A personal computer which costs only $50 but takes days to run a simple program would not be practical; neither would a high-speed computer which ran up enormous electricity bills.)

Once a practical solution is identified, the task becomes one of actually designing a circuit to perform the required function. The design must then be analyzed to determine that the circuit does, in fact, produce the required results. Such analysis is vital before committing resources to actually produce the design.

The design is initially drawn as a functional block diagram. It is partitioned into managable sub-blocks, with particular attention given to the inter-block communications (interfaces). The key to effective design is partitioning it into sub-blocks with clearly defined, independent functions. This partitioning of function also introduces a partitioning of constraints: speed, cost, power, etc. are allocated to each functional block.

The detailed design of each functional block should focus on "robustness," to assure that when these independent blocks are interconnected, they will perform the overall function. If each block has only a small margin of performance, the likelihood of the composite design performing reliably would be extremely low. If, on the other hand, the speed of operation of the individual blocks is relatively independent of the speed of the specific devices used, the margin of performance will increase, as will the likelihood of proper system performance.

1.3 DESIGN FLOW

The conceptual design flow, from requirement to product, is shown in Fig. 1-1. In this context, the functional design is the initial process of deriving a potential and realizable solution to the input design requirements. This is sometimes referred to as architectural analysis and design, and includes such activities as hardware/software tradeoffs or speed/power tradeoffs. With a firm functional design, analysis is then performed to determine the best way to implement the design, subject to the design constraints (technology, size, power, cost, etc). A schematic is then drawn to show the proposed interconnections of available parts.

This proposed implementation is analyzed for proper functioning by applying a test sequence that emulates a subset of the conditions to be expected in real use. Once the designer is confident that the design will

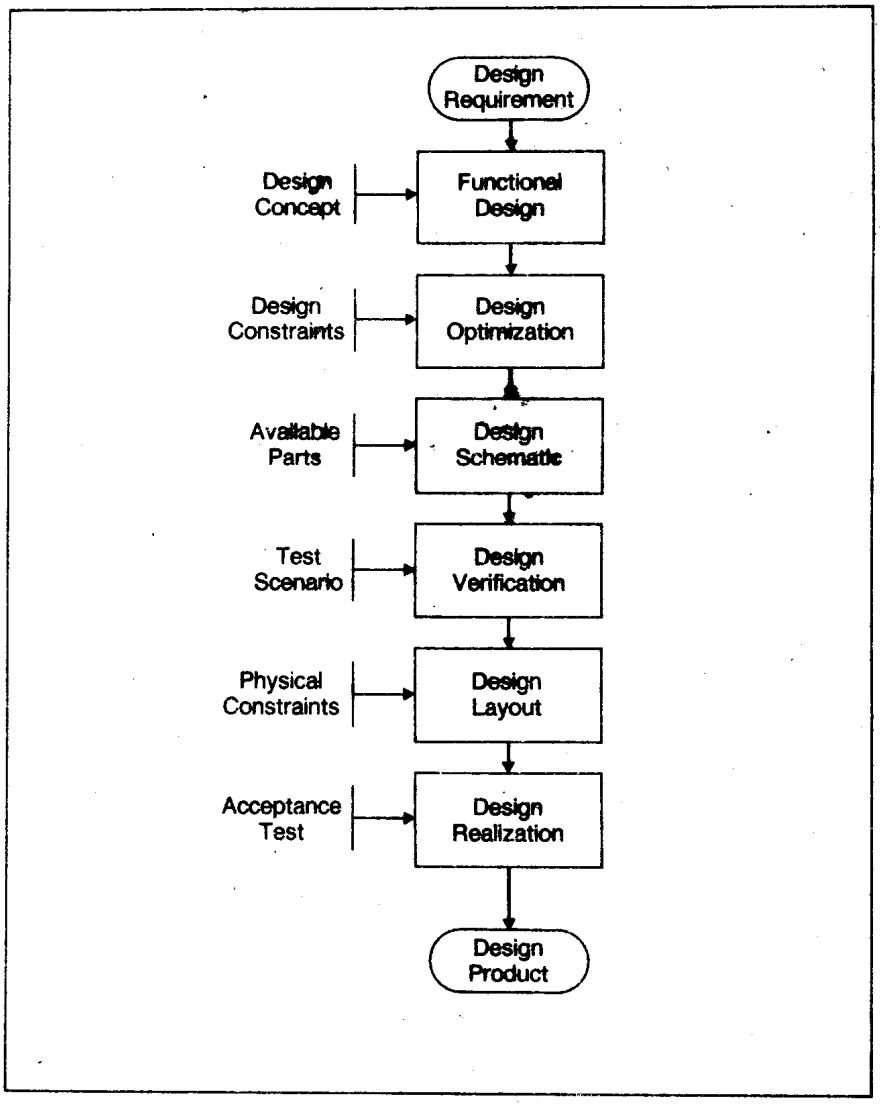

Fig. 1-1
The design process.

meet functional requirements, the physical layout for the actual interconnection of devices is formulated. This layout is subject to available physical technology, such as types of packages and levels of interconnection. Finally, the design is realized and tested for release as a product.

It should be obvious that this design flow is sequential only in the ideal, conceptual case. During the optimization (functional analysis) phase, alternative functional designs may be developed, or new schematics may be drafted. During design verification, flaws in the functional design or schematic may be uncovered, or preliminary layouts may be drafted. During

acceptance tests, flaws in functional design may be uncovered, making a major redesign necessary. The key to effective design is to strive for a sequential flow, keeping any backtracking/recycling limited to one or two stages in the design process. The purpose of computer-aided design is to assist the designer through each stage, minimizing backtracking or major recycles. CAD capabilities range from analysis and verification tools to totally automated processes, such as logic minimization and automatic layout.

While this conceptual design flow is applicable to most electronic design tasks, later sections of this book will primarily address digital logic design and those computer-aided design tools which support digital logic design.

1.4 DIGITAL DESIGN

The term "digital" design applies to the field of electronics involving the use of Boolean/binary logic. Designing a circuit which will detect the *presence* or *absence* of a voltage or current is a fairly straightforward task, but discerning various *levels* of voltage and current is significantly harder.

Consider, for example, the addition of two numbers electronically: ideally, 1 V plus 1 V would produce 2 V. A linear circuit could perform this task, but factors such as line resistance or temperature could convert the 1 V to a smaller value, such as .99 V. The resultant output would no longer be correct (i.e., 1+1 = 1.98?).

By "clipping" or "rounding" the voltage to a predefined level, a correct ("accurate") result could be obtained, but at the cost of precision. Taken to its extreme, any voltage could be rounded to a binary ON/OFF state, so that the effects of line loss, noise, or temperature are minimized. Again, this would be done at the cost of precision (ON + ON = ON?). Precision can be maintained, however, by enlarging the "width" of the value being measured or produced, as is done in conventional arithmetic. When the one-digit numbers 6 and 7 are added, the number 1 is carried to the next column (thus increasing the "width" of the number), to produce the two-digit number 13 as a result. When 7 is divided by 2 (producing 3) the result is widened to add a column for the .5. This produces a more precise result of 3.5. In binary logic, we provide precision by widening the terms used, so that ON + ON = ON OFF (or 1+1 = 10).

In digital design, each signal is used to represent a single *B*inary dig*IT* (BIT), and can assume one of two possible values: ON or OFF. These signals, or groups of signals, are used for computation, communication, and/or control in complex electronic designs. The use of these signals provides a noise-immunity and precision that cannot be achieved with analog (linear) design. As technology advances, more and more classically analog functions are being replaced by digital logic to take advantage of these characteristics. In 1983, for example, ITT announced the design of a set of VLSI (Very Large

Scale Integrated circuit) chips to produce digital television, with characteristics and capabilities impossible to achieve with traditional analog circuitry. Digital filters, digital communication links, and digital switches are commonplace today in the telecommunications industry.

The circuitry needed to make use of logic gates (the basic building blocks of digital design) is relatively simple to design. The design of complex systems using these basic logic gates, however, requires careful planning, design, analysis, and testing to assure that the design will perform the intended function.

1.5 FUNCTIONAL DESIGN

Quite often, the functional requirements for a design are ambiguous or nebulous (particularly for complex electronic designs). One of the primary initial tasks in the design process is to "pin down," or clarify, these requirements. In the most primitive sense, computers are sometimes used to verify the requirements by modeling the function as a computer program. Standard programming languages, such as FORTRAN, PASCAL, or even BASIC, are used for a program that will verify the intended "transformation" from input to output.

This programming approach can be extended by using a language tailored to electronic design modeling: a *Hardware Description Language* (HDL). Such languages support constructs such as Register Transfer (RTL), for modeling the transfer of data among registers in a design, or Instruction Set Processing (ISPS), for modeling the control of operations based on a defined instruction set. These languages may be stand-alone programming languages which are compiled or interpreted for direct execution on the computer, languages intended to operate in conjunction with a simulator, or a combination of the two. In the stand-alone approach, the user must include the time sequencing, input and output processing operations, etc., while such operations are *included* in the simulator.

In some cases, *animation* tools are available to model functional operations. They typically use a graphical input language coupled with a background simulation program, for modeling sequential flow or control systems such as finite state machines.

The designer would typically operate at this functional design level, using the high-level design languages to partition the design into lower-level, well-defined and verified functional blocks such as processors, controllers, and memory. The CAD tools would be used to simulate the operation of the design at this level, before any physical building is done, to verify that the design concept and partitioning satisfy the functional requirements. Without CAD, the designer would typically analyze the design manually and mathematically at this stage.

1.6 DESIGN OPTIMIZATION

As part of the analysis process, the designer typically considers alternative overall strategies to achieve the same function at reduced cost, reduced power, reduced size, etc. The same consideration is given to each functional block of the selected design approach as well.

One such technique is *logic minimization*: achieving the functional block's requirements using the fewest number of gates with the fastest possible speed. Logic minimization will be discussed extensively in later chapters. Other techniques, such as *state assignment optimization* and *code optimization* are also employed to minimize costs. Analysis is also performed to allocate the design constraints, such as speed and power, among the various functional blocks. This is done to avoid making any particular block unfeasible and/or uneconomical to implement.

Unfortunately, a truly *optimal* design that satisfies all design constraints and optimization objectives is seldom possible to achieve (even if it is achieved, it is virtually impossible to "prove" that this design is better than every other possible alternative). The major intent at the optimization phase is to eliminate any obviously wasteful approaches in the overall design or in the design of each functional block and to identify any overconstrained requirements imposed by the architecture selected.

1.7 DESIGN SCHEMATIC

Schematics are a pictorial representation of a design, showing (in symbol form) the parts used and the ways those parts are connected. A schematic may be at the block level, functional level, gate level, or transistor level. Typically, higher-level blocks refer to more detailed lower-level schematics in a hierarchical fashion.

In the 1960s and 1970s, graphics terminals connected to a computer were used for drawing schematics, eliminating the need for manual drafting. Schematics drawn on a graphics terminal resulted in high-resolution drawings that could be easily changed. Usually, the schematics were first hand-drawn by the designer, then turned over to a professional drafting department for computer entry. In this system, design changes entered a queue, often resulting in a lag between the design change and its formal documentation on the schematic. In the 1980s, engineering workstations (high-resolution computer systems for individual use) were introduced, allowing the concept of *schematic capture* by the designer to become a reality. Schematic capture systems, which allow the designer to draw a schematic and "capture" its information content, are now available for use on personal computers (Figs. 1-2, 1-3).

Fig. 1-2
FutureNet DASH schematic design system.

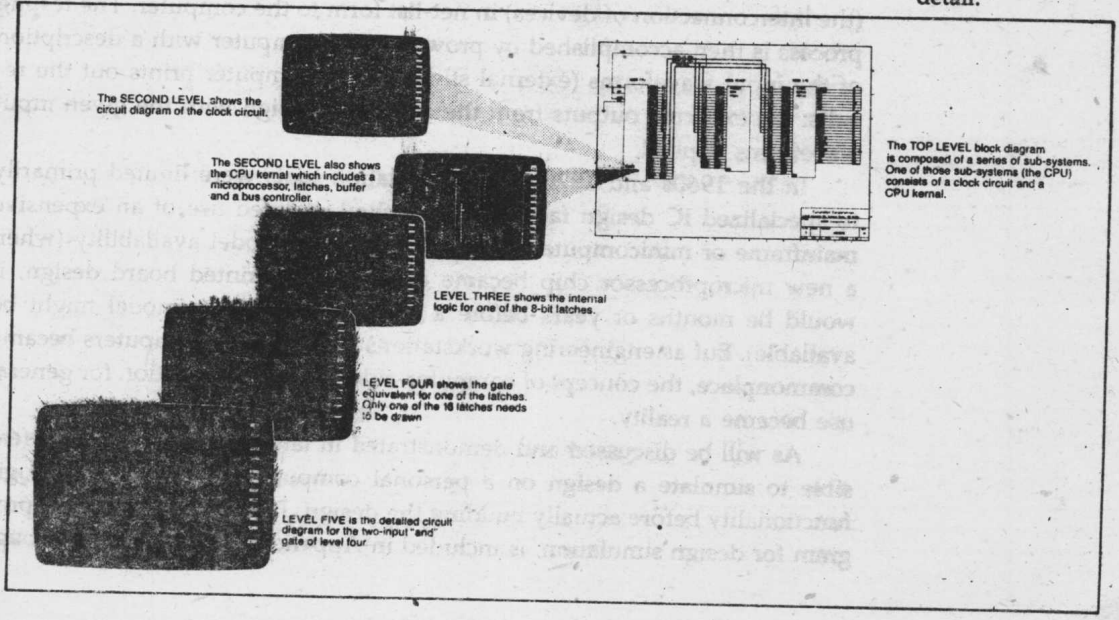

Fig. 1-3
The Structured Interactive Design System option of FutureNet displays up to 99 levels of circuit detail.

Today's typical high technology design environment gives the designer the ability to enter a schematic on the computer directly, with an option to automatically produce a *net-list* (a list of devices and their interconnections). The schematic can then be processed by computer-based analysis and verification tools, or used in manual or automatic construction of the design from the net-list.

1.8 DESIGN VERIFICATION

Before a design goes into production, the designer performs extensive testing and verification to make sure that it will perform as intended or specified.

In the past, such testing was typically performed on the designer's workbench, using prototype (*breadboard*) models of the schematic design. Building and testing the model itself was often a time-consuming process, since each part and wire had to be verified as identical to that shown in the schematic. Once the breadboard and schematic were verified as identical, design changes were difficult, sometimes introducing wiring errors in previously verified sections. The test setup and the testing process itself was tedious, and manual data collection was the norm for documenting results.

The development of the integrated circuit (IC) made necessary the use of computer-based "software breadboarding" (*circuit and logic simulation*) approaches to design verification. Hardware breadboarding of the thousands (now hundreds of thousands) of transistors in an integrated circuit became both impractical and inaccurate, so computer models for transistors and logic gates were formulated and incorporated into simulation systems.

Rather than building a breadboard, today's designer describes the design (the interconnection of devices) in net-list form to the computer. The testing process is then accomplished by providing the computer with a description of the input waveforms (external stimuli). The computer prints out the results: the expected outputs from the described design with the given input waveforms applied.

In the 1960s and 1970s, such simulation tools were limited primarily to specialized IC design facilities, since they required use of an expensive mainframe or minicomputer and were limited in model availability (when a new microprocessor chip became available for printed board design, it would be months or years before a software simulation model might be available). But as engineering workstations and personal computers became commonplace, the concept of computer-aided design verification for general use became a reality.

As will be discussed and demonstrated in later chapters, it is now feasible to simulate a design on a personal computer to verify proper logic functionality before actually building the design. PROTOSIM, a BASIC program for design simulation, is included in Appendix A. Testing and debug-