design automation of digital systems

VOLUME 1

THEORY AND TECHNIQUES

Edited by MELVIN A. BREUER

PREFACE

The design and development of digital systems can be partially automated by using digital computers as design tools. To accomplish this, effective techniques must be established for each of the various steps of design, evaluation, manufacture, and maintenance. In the process of automated development of such systems, digital computers have been utilized both as controlling devices and ancillary aids.

The rapid growth of design automation has been evidenced by the investments of large digital system manufacturers, as well as by the relatively significant commitments made by small companies and some software organizations.

Unfortunately, the growth of the automation technology has not been matched by the existence of adequate and easily obtainable literature. This difficulty has been compounded by the fact that until recently, few techniques related to design automation were taught at the university level. As a result, there is considerable ignorance of many existing techniques, even among practitioners. The user can find only sparse justification to permit him to evaluate, with confidence, the adequacy of various techniques. Thus, work is frequently duplicated and, as new people enter the field, they often "re-invent the wheel."

Recognizing this problem, the Executive Committee of the IEEE Computer Group created an ad hoc committee of the Design Automation Committee to write a substantive book on design automation which would provide: 1) a definitive introduction to the technical aspects of digital system design and development for individuals with either hardware or software oriented backgrounds, 2) a guide to the selection of techniques for inclusion in a design automation system, and 3) a foundation from which others could develop new and better design automation techniques.

This volume and a subsequent one are the results of this effort which first began late in 1968. This volume deals with four aspects of the logic of a system; namely, synthesis, simulation, testing, and physical implementation. The latter area includes partitioning, placement, and routing. The subsequent volume will deal with system level simulation, simulation and synthesis at the register transfer level, file maintenance, and interactive systems.

These two volumes collectively describe practical techniques which have either been successfully employed, or have been considered to be useful alternatives in a design automation system. The constraint of practicality often eliminates both total enumerative and elegant theoretical techniques. Enumerative techniques, though effective since all digital systems are finite, are usually computationally infeasible due to high costs. On the other hand, the implied benefits of many theoretical solutions may not be achievable in actual practice.

We realize that what is and is not practical varies with the user, the number of parameters in the problem, and time. Due to the rapid technological advances, solutions to problems posed today may not be acceptable solutions for these same problems tomorrow. Here we discuss techniques in which the cost of developing and executing the design automation system are reasonable, and in which acceptable results are obtainable.

This work is suitable not only for those who are familiar with some aspects of design automation, either from a hardware or software viewpoint, and wish to learn about other areas, but also for those just entering the design automation field. It will be of particular interest to students who desire to learn aspects of computer design which are different from those taught in the traditional courses on logic design and switching theory. The reader need only have a basic knowledge of digital computers and logical design.

We believe this work will also have strong appeal to the applied combinatorial analyst, since we cover aspects of many key subjects in this field such as discrete optimization theory, branch and bound techniques, integer programming, graph theory, the covering problem, the traveling salesman problem, the assignment problem, and the quadratic assignment problem.

The subject matter has been divided into two volumes, the first being somewhat hardware oriented, the second software oriented.

Volume 1 deals with the problems of logic synthesis, simulation, testing, and physical implementation.

In chapter one we introduce the general area of computer aided design of digital systems, discuss the motivation for and goals of computer aided design, and outline the historic development of this growing area. Finally, the chronological steps involved in the realization of a new digital system are outlined, and we indicate where and how design automation can interact in each step of the design and manufacturing process.

The classical problem of logic synthesis is discussed in chapter two, where techniques are presented for logic simplifications, factorization, and conversion between different families of logic elements. Practical restrictions related to fan-in and fan-out constraints are considered.

Chapter three deals with the simulation of a digital system at the logic or gate level. Here, the goals of logic simulation, simulation languages, techniques, and systems are discussed.

The next three chapters deal with three major problems related to the physical construction of a digital system.

Chapter four relates to various partitioning and assignment problems associated with component layout, as well as to problems of evaluating the effectiveness of standard circuit modules.

Chapter five presents a detailed analysis of various placement algorithms dealing with the problems of initial placement and placement improvement. These techniques are applicable to a wide range of backplane functional units, e.g., components, modules, and boards.

The interconnection or routing problem is treated in chapter six. The discussion covers both discrete and etched multi-layer connection techniques, including the related problems of ordering of interconnections, path selection, pin selection, and layer selection.

Finally, in chapter seven we cover the problem of generating fault diagnostic and detection tests for combinational and sequential logic circuits.

As programming languages have evolved from machine code to assembly languages to machine independent languages, so digital design languages have evolved from circuit descriptions to Boolean expressions to register transfer languages and finally to system level languages. Volume II will deal primarily with new design techniques at the higher levels of description as well as associated support systems. More specifically, this work will cover material on the following subjects:

1. Descriptive languages for digital devices at the system or functional level, and related simulation techniques used in evaluating system performance.
2. Register transfer languages.
3. Synthesis of digital devices from their register transfer level description.
4. Simulation of systems at their register transfer level.
5. Data-base systems for use in an integrated design automation system.
6. Interactive graphic systems and their role in computer-aided design.

As a final note I would like to thank my co-authors who have contributed so much of their time and effort in the preparation of this book.

MELVIN A. BREUER

CONTENTS

chapter one

INTRODUCTION

RALPH J. PREISS

IBM Systems Development Division
Poughkeepsie, New York

This chapter presents an overview of the purpose and evolution of design automation and its application to the computer design process. As an overview of a field involving many manufacturers, universities, and research foundations, the reader should not assume that any process described is practiced by any particular organization. Rather, the reader should realize that this overview is a synthesis of the many practices that the author has observed, read about, or discussed with other design automation practitioners.

The chapter is divided into three sections. Section 1.1 reviews the objectives of design automation and the attempts that have been made to achieve them. Section 1.2 covers the history of design automation as an evolutionary discipline. Section 1.3 provides a summary of the steps involved in the design of a digital system, and it emphasizes the designer's responsibilities and interplay with a design automation system.

1.1 DESIGN AUTOMATION OBJECTIVES AND USAGE

Design automation, as used here, is a goal rather than an immediately realizable objective. It is the art of utilizing digital computers to help generate, check, and record the data and the documents that constitute the design of a digital system [1].

The objectives of design automation are cost- and time-reduction between start of design and completion of fabrication. These objectives are accomplished by relieving design engineers of repetitive, time-consuming manual tasks such as:

1. generating detailed design information and documenting it on what is here called *systems logic pages*;
2. controlling changes to systems logic pages;
3. checking systems design for electrical, logical, and physical compatibility;
4. preparing wiring lists, cable lists, location charts, and other manufacturing data, such as the bills of material, and tests.

A wide variety of topics are covered by the design automation umbrella. The only unifying theme is the elimination of repetitive manual operations or computations in the design of digital systems. Excluded from consideration are the computations for the design of components, circuits, mechanical structures, and cooling. Design automation, therefore, limits itself to filling the gap between the systems specifications and the manufacturing data. It is involved with converting the systems specifications into logic hardware, packaging the hardware into mechanical structures, and describing this process for fabrication.

If the design process could be fully automated, the conceptual data that describe a proposed digital system in a higher-level language could be converted directly into the mass of detailed data, i.e., part numbers, assembly sequences, cabling information, etc., necessary to manufacture the machine. Failing complete automation, design automation can be used as a data base from which to extract design information on demand, calculate (with specialized design mechanization aids) additional details, and then return the new data to the data base. This permits creative manual intervention and allows additions and deletions to be made as specifications change.

The data base in which the design is described may contain computer-generated as well as hand-generated data. This includes, for each logic function, the part number code implementing it, its physical placement in the machine frame, test points, cabling and internal wiring information, cross-referencing data, notes, and design change activity information. Supplementary information, such as the exact location and types of bends or the lengths of cables and interconnecting wires, may be omitted from the main documents and relegated to secondary documents. Similarly, the detailed description of subassemblies, such as discrete components mounted on cards, may be relegated to secondary documents. However, each of these secondary documents, once entered into the design automation system data base in computer-readable form, is seldom retranscribed but is utilized by the system in subsequent calculations.

While the basic goals of design automation tend to be the same from one manufacturer to another, the automating philosophies tend to be quite different. On the one extreme, a manufacturer considers the accuracy of his field documents paramount and therefore uses the machine-readable data base and the same documents in design and manufacture. On the other extreme, a manufacturer may permit differences between the field and the design and the manufacturing documents. The latter approach could best be described as having multiple data bases. It is characterized by multiple transcriptions to or from computer-readable form to take advantage of the computer for certain calculations, such as logic simulation (requiring logic data), wiring (requiring interconnection data), or load checking (requiring electrical data). Errors that crop up through transcription mistakes are detected by procedural means. The field documents are usually draftsman-produced, and they may or may not fully reflect the latest design changes.

Design automation programs are generally not shared among manufacturers, since the programs are quite heavily dependent upon design philosophies, circuit technology used, and the computer available to the design engineer to help him with his work. (The use of higher-level languages usable with different computers has been quite discouraging. These tended to be inefficient: fine for experimental purposes, but too expensive for production use.) Furthermore, design automation is especially advantageous if standardized packaging and logic are used and if a standard control system, unique to individual manufacturers, is set up. These aspects are usually proprietary since they are tied into providing an economic advantage of one design/manufacturing process over another.

Besides automating design checking and record-keeping, a design automation system serves another important function: effective change control. By reducing design and engineering-change time, it permits logic changes to be made late in the development process, assures accurate manufacturing instructions and test procedures, and instills a confidence that manual procedures could not hope to attain.

1.2 EVOLUTION

In the mid-1950s, the first generation (tube generation) of stored-program computers was in its prime. At the same time, transistor technology was getting to be understood, and uniform circuit characteristics could be assured in large batches of manufactured components. It was obvious that the new (second generation) computers being designed would utilize this new technology.

While transistor circuits were smaller, faster, and cheaper than their tube counterparts, their use implied that the new generation of computers would have to be redesigned from the frame up. In addition, the small size of the new components meant the introduction of new problems in packaging and interconnection. Where a single 7-tube circuit might require some 100 cubic inches of space, perhaps ten transistor circuits might require no more space. The smaller components permitted an increase in circuit speeds, but inductive noise was increased at the same time because of the closeness of the interconnecting wires.

Also, at this time, computer manufacturers were faced with the prospect of meeting contractual deadlines for military computers using transistors. These deadlines forced computer manufacturers to look for new ways to design computers in order to reduce their risks.

1.2.1. Efforts to Automate Design Procedures

Around 1955, without a single bit of literature appearing in any of the journals, the idea of design automation caught on, and the major computer manufacturers were all in the computer-aids-to-design, or design-mechanization, or design-automation business.

In a typical company, study of the design procedures showed that a logic designer sketched out his ideas in rough form and passed them on to draftsmen who put them into more readable form. Additional information would be added to a copy of the draftsman's drawings until enough information was contained so that the wiring and cabling layout could be developed. This process in itself would add additional information to the master diagram. Each time an addition or a change was made, the history would be recorded, and the appropriate designer would initial the drawings after review.

With the advent of transistor circuits, the wiring layout designer's job became more difficult. He had to make sure that two adjacent wires didn't travel too many inches in parallel because of "noise buildup." He constituted a vital link and he needed assistance in the design and build process. The computer could be used to lay out the wiring grids, take into account capacitive loading and interwire noise, and print the diagrams of these wires. This approach could provide a more direct involvement of the designer in the final hardware. It could also aid in the manufacturing process by calculating the path and the sequence that the wire-wrap people should follow. Furthermore, these calculations could save days in the schedule.

The computer designers also realized that they could substantially improve design turnaround if the logic diagrams were maintained on magnetic tape files and the computer could print these diagrams for every engineering change. It would only be necessary to keypunch a few cards describing a change, update the master file, and a printout of the modified design could be obtained in a few minutes.

1.2.2. First Uses of Design Automation

One of the first design automation systems described in the literature was presented at the 1956 Western Joint Computer Conference by Cray and Kisch [2]. They described a three-phase program which started with the checking of logic equations for logical, clerical, and timing errors. Further checking could be done by counting the number of inputs and driven outputs per circuit (i.e., fan-in/fan-out checking). The Boolean equations could also be sorted into various categories, and printed outputs could be provided to those who needed them. In the second phase, the simulation phase, the designer could apply input values to his Boolean equations from a simulated switch panel and watch the results as though he had built the machine and were testing it out. Finally, in the third phase, he could package his design into assemblies, compute the interconnections between chassis, provide lists of the origin and destination points, and designate the color and the lengths of the wires. One interesting observation which also might indicate the pioneering nature of this article is that it cited no references. Another early article by Kloomok, Case, and Graff, describing a similar but more elaborate system, was presented at the Eastern Joint Computer Conference in 1958 [3]. After 1960, many manufacturers were busy discussing their systems [4], [5], [6], [7], [8], [9]. Figure 1-1 depicts the major problems facing computer designers and design automation. What follows here is a narrative account of the design-automation evolution.

Let us return to the development of design automation systems. Initially, the input for the wiring layout was from system logic blueprints. From these, the coordinates of the series of points which were to be connected by wires were keypunched. The computer program consisted of the wiring rules and legal wiring channels. After reading the keypunched coordinates, the computer produced a listing that indicated the length of each wire, the sequence in which each should be installed, and the route each wire should take (Fig. 1-1, items 1 and 2).

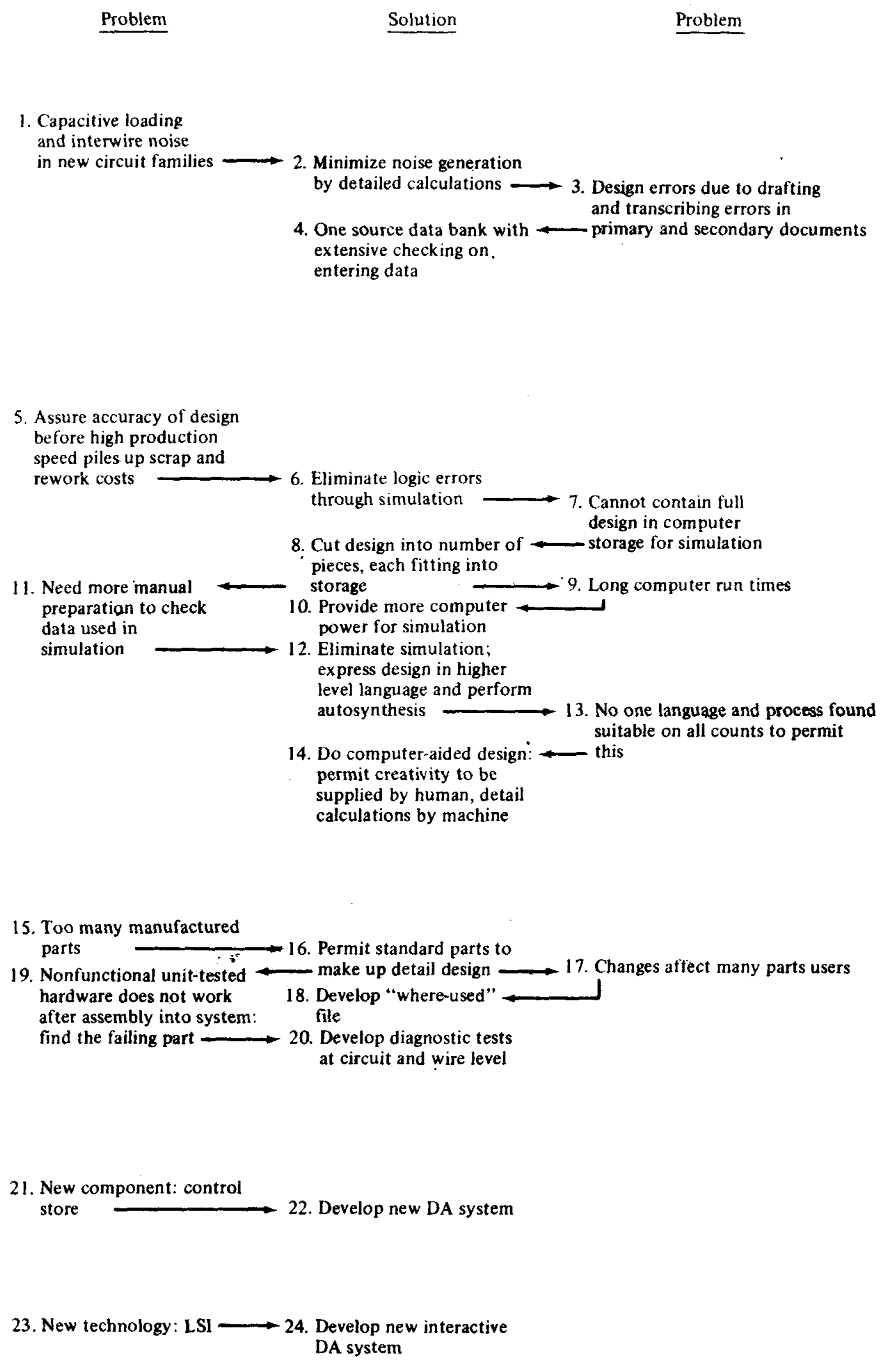
Problem
Solution
Problem
1. Capacitive loading and interwire noise in new circuit families
2. Minimize noise generation by detailed calculations
3. Design errors due to drafting and transcribing errors in primary and secondary documents
4. One source data bank with extensive checking on entering data
5. Assure accuracy of design before high production speed piles up scrap and rework costs
6. Eliminate logic errors through simulation
7. Cannot contain full design in computer storage for simulation
8. Cut design into number of pieces, each fitting into storage
9. Long computer run times
10. Provide more computer power for simulation
11. Need more manual preparation to check data used in simulation
12. Eliminate simulation; express design in higher level language and perform autosynthesis
13. No one language and process found suitable on all counts to permit this
14. Do computer-aided design: permit creativity to be supplied by human, detail calculations by machine
15. Too many manufactured parts
16. Permit standard parts to make up detail design
17. Changes affect many parts users
18. Develop "where-used" file
19. Nonfunctional unit-tested hardware does not work after assembly into system: find the failing part
20. Develop diagnostic tests at circuit and wire level
21. New component: control store
22. Develop new DA system
23. New technology: LSI
24. Develop new interactive DA system

Fig. 1-1. Design automation evolution.

Meanwhile, manufacturing engineers considered automation further. Why should people wire panels? Why not have a computer produce punched cards with instructions that a wiring machine could follow? Couldn't the data from the engineering listings be keypunched in the proper format for the wiring machine?

The Gardner-Denver Tool Company developed a card-controlled wiring machine which came into extensive use in the late 1950s [10]. The programs that routed the wires were modified to control this machine. With its use, not only was wiring speed increased, but also fewer errors occurred. The backpanel wiring technicians could now multiply their productivity tremendously. But all could not be automated. Since the engineers were constantly improving their designs through engineering changes, manual rewiring was still necessary.

Concurrently, schemes to accumulate changing engineering information on a logic master tape were developed. This master tape could then be printed out to indicate the design status, and the printouts could be marked with colored pencil to indicate more changes. Then, only the changes would have to be keypunched to update the master tape. The engineer who provided the original sketch would have to proofread only the marked-up sections, thus saving himself a considerable amount of time.

However, the logic design had earlier depended upon draftsmen to design the packaging, do the design checking, and sometimes even add carelessly left-out details. With the draftsmen removed from the design process, the design engineer had more details to contend with. For example, a simple keypunch error could hurt the design schedule. (Fig. 1-1, item 3). To aid the designer in following details, checking programs were developed to perform a reasonableness check on the data fields of the master tape: alphabetic versus numeric formats of known functions, physical layout (so that two packages didn't occupy the same slot), and circuit fan-out (overloading). Some of these check the keypunch operators; others check the logic designers' application of the rules of design (Fig. 1-1, item 4).

One might ask if this was design automation. The answer might be that the drudgery of drawing lines, of copying numbers, and of cutting, pasting, erasing, and redrawing had been removed. The designer had more direct control over the implementation and accuracy of his design.

1.2.3. Use of Simulation

Meanwhile, an attempt was being made to reduce engineering change activity. With the high production rate of the automatic wiring machine, errors could be reproduced rapidly. Slow response could cause a lot of scrap and rework, and many dollars would be wasted. If the logic were simulated, proven once and for all to work the way it was intended to work, and checked to ensure that nothing was left out inadvertently, fewer designers would be needed, and scrap and rework would be reduced, if not eliminated (Fig. 1-1, items 5 and 6).

The engineers quickly found that in a small (4000 word model) computer they could not contain the entire design that they wanted to simulate (Fig. 1-1, item 7). But even 8000 words sometimes were not enough. Only a few small pieces of the design could be simulated in each computer run. The value of the time needed to keypunch the data and evaluate the results was often more than the value of the results themselves. But programming helped solve the problem by putting the complete design on a drum or tape storage and swapping small segments with high speed memory as required. This technique tended to use up much computer time (Fig. 1-1, items 9 and 10).

Simulation had run into trouble. The amount of data generated and printed out was unwieldy. The designer disliked having to generate data for the unnatural partitions into which his logic had to be subdivided in order to fit into the computer (Fig. 1-1, items 8 and 11). Why not validate a design before it gets into the detail level? In one sense, the true creative person in the design of a computer is the computer architect rather than the designer. If the architect could express his design in some language that could be synthesized directly into logic (Fig. 1-1, item 12), the logic then could be implemented in hardware, assigned to panels and frames, wired, and cabled automatically. It could be tested at the higher design language level by another program to prove that the architect's design, as conceived, was indeed implemented. If simulation as a means of testing a manually produced detail design did not at first succeed, maybe it should be approached differently and bypass the designer by going directly into an automatic synthesis routine. Of course, the utopia we speak of has not yet been realized. It has literally been lost in a Tower of Babel (Fig. 1-1, item 13). The languages for expressing the architect's dream are many. A discussion of higher level design languages is planned for Volume II of this series. Suffice it to say here that the search is still on for a simple, yet effective, man-machine system for computer-aided design (Fig. 1-1, item 14).

In the meantime, especially with the advent of large-scale memories, simulation was integrated into design automation systems so that the designers could make use of the already-digitized data base. Simulation thus became a practical and widely used tool.

1.2.4. The Advent of Standardization

With some success at automation behind them, manufacturing engineers introduced additional cost-saving changes by sharing fabrication techniques between models and by establishing a standard interface between design-automation-produced data and manufacturing-usable data (Fig. 1-1,