

600 Community Drive, Manhasset, New York 11030

1988

\$50.00

Bulk Rate  
U.S.  
Postage  
PAID  
CMP  
Publications  
Inc.

# VLSI

SYSTEMS DESIGN'S

# USER'S GUIDE TO DESIGN AUTOMATION



**The Information Source For Users  
Of Electronic Design Automation Tools**

A CMP PUBLICATION

**“ASICs CREATE A  
SET OF TEST PROBLEMS.  
WE NEED A WHOLE NEW**



# WHOLE NEW DOESN'T THAT MEAN TEST STRATEGY?"

## IT SURE DOES.

It's easy to see that the tremendous potential of ASICs has only just begun to be tapped. What's not so evident is the fact that developing these unique ASIC devices carries with it some unprecedented test problems. Problems that traditional test approaches and traditional ATE simply are not equipped to handle.

At ASIX Systems our focus has always been exclusively on ASICs. From the start we recognized the unique ASIC test problems. That's why we took an entirely different approach to solving these problems. For instance, we saw that adapting existing ATE to fit the needs of ASICs didn't make sense. Designing a totally new, focused ASIC test system did. Test programs needed to be automated, developed from the design data base, and simple to change. The test system itself needed to be easy to use, designed for its particular environment, and a cost-effective alternative to the huge, expensive, complicated ATE.

## TEST SOLUTIONS FOR THE WHOLE ASIC COMMUNITY.

Our unique perspective allowed us to understand that the ASIC world is not Design Engineers, Test Engineers and Quality Engineers performing separate functions. It's actually a "community" of specialists whose tasks are intrinsically linked. So we made sure that we could provide another crucial element. Communication. In order to capture the vital time-to-market edge, what ASIC designers and vendors really need is the opportunity to use the same test programs and the same tester. Because when both environments are working from a common frame of reference there can be some real communication about test results. That's a whole new way of looking at ASIC testing. That's the ASIX-1 family of test systems.

## ASIX-1: ASIC TEST SYSTEMS THAT MAKE SENSE.

This isn't the place to tell you everything the ASIX-1 family has to offer. But here are a few things to think about: automatic, menu-guided programming; data base management; ATE architecture and flexibility at an affordable cost; 256 true I/O pins; "zero footprint"; fully integrated PMU; automatic calibration; simple fixturing; no cabling; high MTBF. Enough. You get the point. You really ought to see the ASIX-1 for yourself. And the sooner the better. ASIX Systems Corporation • 47338 Fremont Blvd • Fremont, CA 94538.



**CALL: 1-800-FOR-ASIX**



**REAL ASIC TEST SOLUTIONS.  
FROM THE REAL ASIC TEST COMPANY.  
CIRCLE NUMBER 1**





**Editorial Director**  
Girish Mhatre

**Special Projects Editor**  
Mike Robinson

**Editor Emeritus**  
Roderic Beresford

**Editor-At-Large**  
Stan Baker

**Editors**  
David Smith (Western Region)  
Ernest L. Meyer

**Editorial Assistant**  
Michelle A. Losquadro

**Technical Advisers**  
John A. Darringer Jeffrey T. Deutsch Edward J. McCluskey Alan F. Podell Daniel G. Schweikert Susan L. Taylor

**Editorial Production**  
Patricia L. Gaynor, Sr. Production Editor  
Ingrid Atkinson, Production Editor

**Cover Design**  
Laurie Kaufman

**Editorial Art**  
Marie D'ippolito, Design Director  
Christine Catalano, Virginia Bassett, Layout

**Manufacturing**  
David Getlen, Dir. of Manufacturing  
Marie Myers, Production Manager  
James Pizzo, Production Supervisor  
Vance Hicks, Coordinator

**Publisher**  
Norm Rosen

**Ad Coordinator**  
Tracy Renk

VLSI SYSTEMS DESIGN (ISSN 0279-2834) is published monthly with an extra issue in May by CMP Publications, Inc., 600 Community Drive, Manhasset, NY 11030. (516) 365-4600. VLSI SYSTEMS DESIGN is free to qualified subscribers. Subscriptions to others in the US: one year \$60.00, two years \$95.00; Canada and Mexico: one year \$90.00, two years \$165.00; Europe, Central and South America: one year \$120.00, two years \$225.00. Asia, Australia, Israel and Africa: one year \$150.00, two years \$285.00. Second-class (Requester) postage paid at Manhasset, NY and additional mailing offices. POSTMASTER: Send address changes to VLSI SYSTEMS DESIGN, 600 Community Drive, Manhasset, NY 11030. Copyright 1988, CMP Publications, Inc. All rights reserved.

40,000 copies of this issue printed.

**CMP ELECTRONICS GROUP**  
Electronic Buyers' News  
Electronic Engineering Times  
VLSI Systems Design  
Ken Cron, Vice President/Publishing  
Frank J. Burge, Vice President/Group Publisher  
Norm Rosen, Group Marketing Manager

CMP Publications, Inc.  
Gerard G. Leeds, President  
L.J. Leeds, Senior Vice President  
Michael S. Leeds, Vice President  
Pearl Turner, Vice President/Treasurer

## I INTRODUCTION 5

### INTRODUCTION TO THE GUIDE 8

Mike Robinson

### EDA PUSHES TOWARD LOGIC SYNTHESIS 10

George Bouhasin

### BUILD BETTER HARDWARE BY FOCUSING ON SOFTWARE 18

Cindy Thames and Andrew S. Rappaport

## II LOGIC DESIGN 29

### BENCHMARKING SCHEMATIC ENTRY SYSTEMS 30

Paul K. Filseth

### A FIRST COURSE IN VHDL 40

David L. Barton

### SOFTWARE VS. HARDWARE MODELS FOR SYSTEM SIMULATION 50

Pete Johnson

## III PHYSICAL DESIGN 63

### LEAF CELL DESIGN 64

M.Y. Tsai and Stephen Wu

### GRAPHICAL FLOORPLAN DESIGN OF CELL-BASED ICS 72

Edmond Macaluso

### A RULES-DRIVEN APPROACH TO CIRCUIT BOARD DESIGN 80

Joseph Prang and Katherine Gambino

---

## **IV DIRECTORIES 95**

**DIRECTORY OF CAE SYSTEMS 96**

**DIRECTORY OF IC LAYOUT SYSTEMS 110**

**DIRECTORY OF PRINTED CIRCUIT BOARD LAYOUT SYSTEMS 118**

---

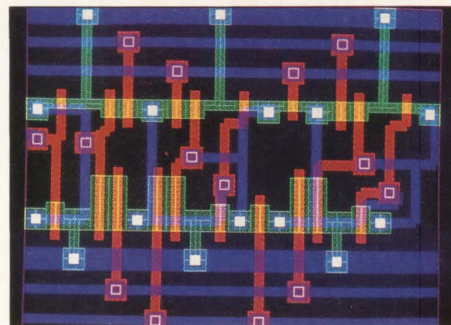
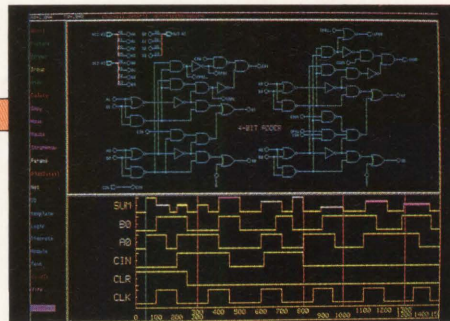
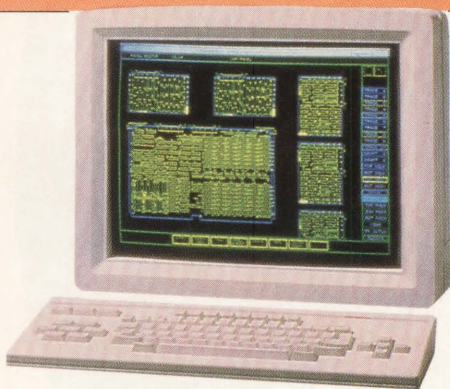
## **V REFERENCES 129**

**GUIDE TO THE LITERATURE 130**

**VLSI SYSTEMS DESIGN SUBJECT INDEX,  
1986-1987 133**

**ADVERTISERS' INDEX 138**

---



# 2

## Tektronix Delivers Integrated PCB Design Capture, Simulation, Layout and Manufacture.

IN A SERIES

**Tektronix  
Aided  
Engineering**

Choosing the right PCB design solution can be a challenge.

Especially when you consider that today's dense, multilayer designs combine digital and analog technologies with both through-hole and

surface-mount packages.

The one sure way to meet your design requirements is with the PCB WorkSystem™.

Developed by Tektronix as part of Tektronix Aided Engineering, the PCB WorkSystem combines design capture, verification, documentation and PCB layout into one powerful solution.

Which means you get Designer's Database Schematic Capture, industry-standard HILO-3® logic simulation and MERLYN-P™ layout. All in the same PCB design environment.

So you can capture your schematic, simulate the circuit, fully place and route your design, and then transfer CAM output data to manufacturing.

The PCB WorkSystem also lets you manage Engineering Change Order iterations more efficiently. The system's automatic forward and backward annotation tools ensure that your schematic always matches your layout.

What's more, the system's router completely enforces flexible, user-defined design rules, resulting in fully routed designs that meet your manufacturing requirements.

When you're ready to fabricate your boards, the Tektronix PCB division provides you with automated manufacturing, on-time delivery and consistent quality control.

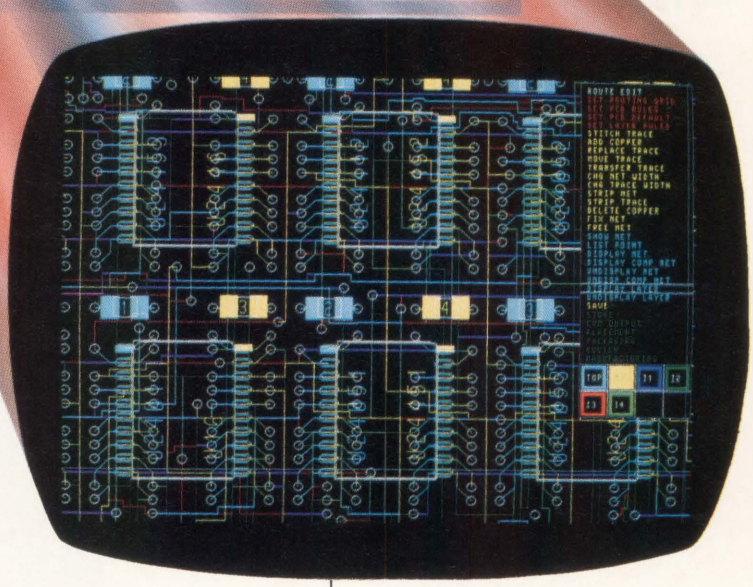
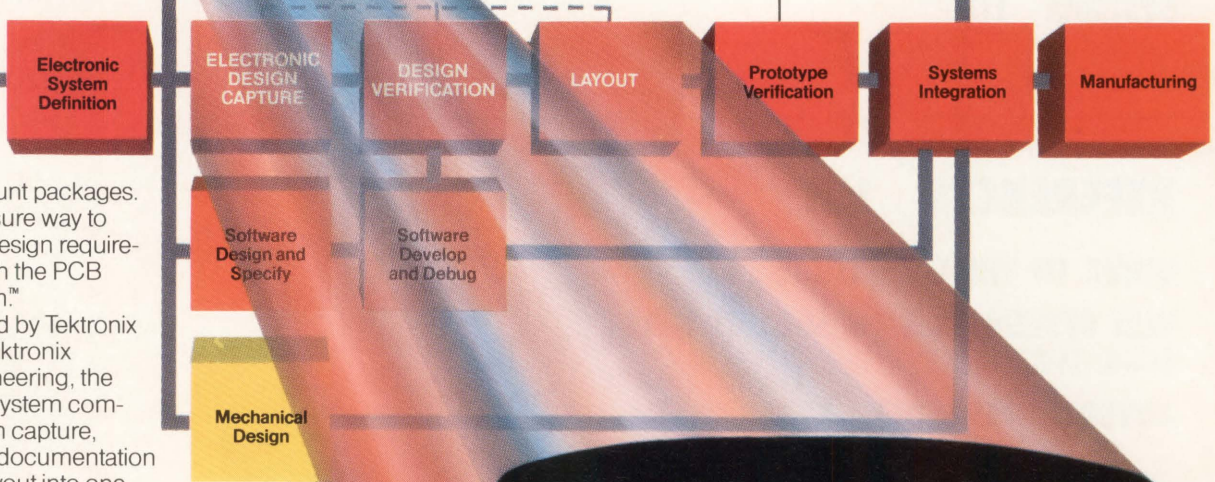
It's all part of Tektronix Aided Engineering. A family of integrated WorkSystems addressing each area of your electronic design cycle. From design capture, verification and documentation to IC and PCB layout. All running on industry-

standard platforms from Apollo® and DEC.™

Best of all, it's from Tektronix. The name you've always trusted to get the engineering job done. So you're assured of worldwide service, support and training.

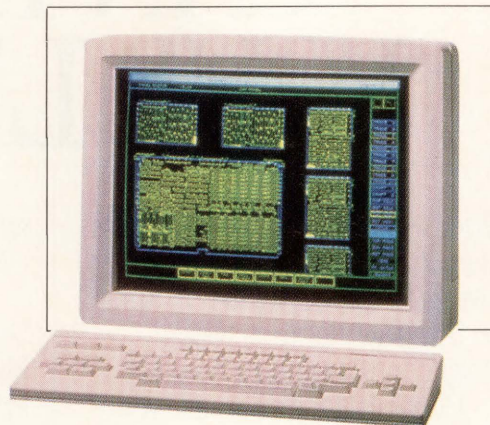
To take advantage of Tektronix Aided Engineering, contact your local Tektronix, CAE Systems Divisions, sales office, Or call 800/547-1512. Tektronix, CAE Systems Division, P.O. Box 4600, Beaverton, OR 97076.

**PRODUCT  
CONCEPT**



WorkSystem, DDSC, and MERLYN-P are trademarks of Tektronix, Inc. HILO is a registered trademark of GenRad, Inc. Apollo is a registered trademark of Apollo Computer, Inc. DEC is a trademark of Digital Equipment Corp.

# I INTRODUCTION



## 8 INTRODUCTION TO THE GUIDE

Mike Robinson

## 10 EDA PUSHES TOWARD LOGIC SYNTHESIS

George Bouhasin, Mentor Graphics Corp.

When logic synthesis appears in electronic design automation systems, it will let designers enter designs in the high-level format of their choice. The system will break down these descriptions into a network of primitives and optimize the design according to the designer's specifications and goals.

## 18 BUILD BETTER HARDWARE BY FOCUSING ON SOFTWARE

Cindy Thames and Andrew S. Rappaport, The Technology Research Group

The interdependence of the hardware and software portions of increasingly complex systems demands greater cooperation between hardware and software designers. A further quantum step in hardware development will come about only when design-tool suppliers address the challenges of concurrent hardware and software design.

# NOW YOUR RIGHT HAND CAN KNOW WHAT YOUR LEFT IS DOING.

Ever seem like your CAE and CAD people are playing for different teams? Especially when it's time to turn that hot new system design into a working board?

Chances are it's because your design systems can't communicate critical information from the engineer to the layout designer. So instead of a smooth handoff, you get hand-to-hand combat.

But now there's a system that

streamlines the way CAE and CAD teams work together.

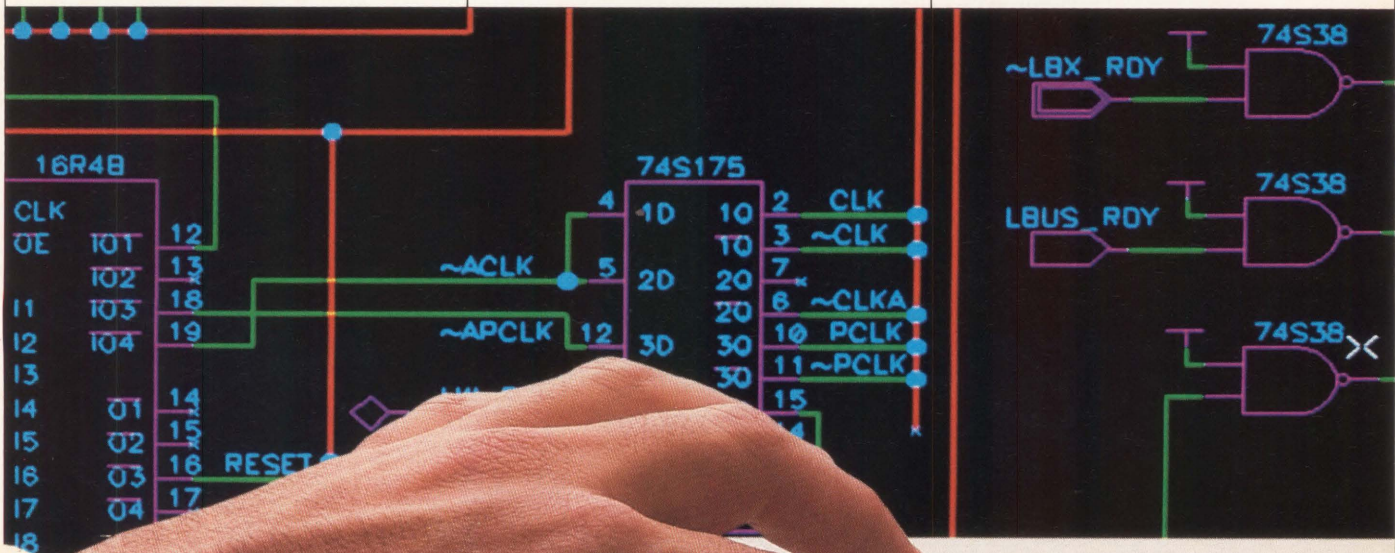
It's Daisy's BOARDMASTER™. The first automated system that plays by the rules of real-world system design.

## Rules-driven PCB design puts CAE and CAD on the same team.

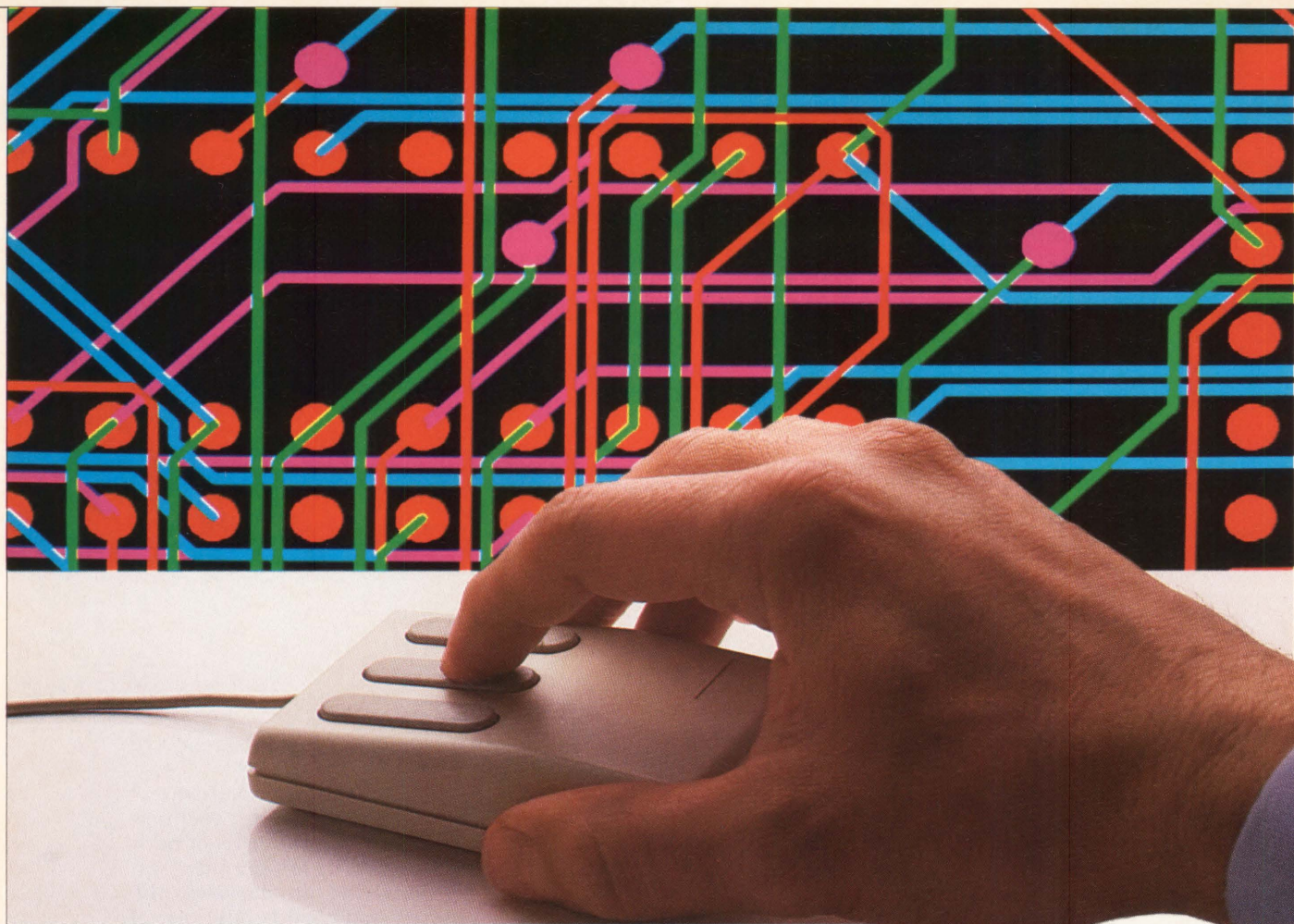
With its rules-driven PCB design environment, BOARDMASTER gives engineers the flexibility to

specify key design rules in the schematic. Rules for signal priority. Ordering and termination for ECL nets. Package types and power definition. Pre-packaging and pre-placement priorities. Pin and gate swapping. And many other important design considerations.

This critical information becomes part of the design database and is passed directly to BOARDMASTER's powerful set







of PCB layout tools.

With these rules guiding the process, layout designers can concentrate on maximizing the quality and manufacturability of the layout without having to second-guess the engineer's real intentions.

**The most advanced tools for today's complex designs.**

BOARDMASTER's rules-driven methodology guides the most advanced set of layout tools available anywhere. Like 100% autorouting, with separate rip-up/reroute and manufacturing passes to increase board yields and reduce per unit costs.

There's full support for advanced technologies like SMD, ECL, analog and ultra fine line designs. Plus a variety of interfaces to photo-plotters, N/C drill machines

and other manufacturing equipment. There's even a Sun-4™-based routing accelerator, so your team can spend less time routing and more time exploring design alternatives.

BOARDMASTER even takes the frustration out of design changes. Because its incremental update capability processes only the parts of the database that need changing. Which keeps ECO from becoming a four-letter word.

**Get your hands on BOARDMASTER and see for yourself.**

So if you'd like to get a grip on better board design, put BOARDMASTER to the test in your next project. And give your entire team a hand.

For a demonstration or more information, call Daisy at: 1 (800) 556-1234, Ext. 32. In California: 1 (800) 441-2345, Ext. 32.

European Headquarters:  
Paris, France (1) 45 37 00 12.  
Regional Offices:  
England (256) 464061;  
West Germany (89) 92-69060;  
Italy (39) 637251.



# INTRODUCTION TO THE GUIDE

Mike Robinson, *VLSI Systems Design*

The year since our first *User's Guide to Design Automation* has seen a number of changes, both economic and technical. Although design automation tools have been adopted by only a small fraction of the design engineering community—indeed, most designers have yet to move beyond schematic capture and logic simulation—the existing technologies are now well established among, and relatively well understood by, those advanced users. This status reflects a phase in the evolution of design automation: The industry seems to have reached a first plateau of maturation. At such a stage, the question, Just how many vendors can design automation support? pushes itself to the fore. Consequently, on the economic front, a number of mergers have taken place in an already crowded field, and more are likely to occur.

Meanwhile, two major directions in technology are already clear. Logic synthesis, which converts a high-level description of a logic function into a structural description, has started to blossom. Initially confined to programmable logic devices, logic synthesis has begun to appear for gate arrays, and several products of this type will likely sprout for both gate arrays and standard-cell ICs in 1988 or 1989. Further behind is the area of computer-aided software engineering, or CASE. Although everyone recognizes that software development is now the major bottleneck in system design, and CASE has been the subject of much talk for the last couple of years, the technology is still in its infancy and is characterized by scattered, essentially preliminary tools geared to general software development, rather than true system-level design and integration. So it seemed appropriate that our introductory section be devoted to these two topics.

In "EDA Pushes toward Logic Synthesis," Mentor Graphics' George Bouhasin looks at the needs of logic synthesis, such as the incorporation of the rules of the various design methodologies (gate arrays, standard cells, PLDS) and processes (CMOS, ECL, and so on), the inclusion of accurate and complete component libraries, and the ability to consider both marketing goals and

technical specifications. In doing so he paints a picture of what an actual product may well look like.

In "Build Better Hardware by Focusing on Software," Cindy Thames and Andrew Rappaport of the Technology Research Group assess the present status of both hardware and software development for complex systems and argue that several trends, among them the increasing system complexity spawned by the use of VLSI and design automation tools, are forcing hardware designers to take software development into consideration as well. They examine the needs of software development, especially for embedded software designed for custom hardware, from a system perspective and look at the emerging solutions.

## THE FIRST STEPS

Our second section is devoted to logic design. Appropriately enough, it starts off with an article on the most widely used tool, schematic entry programs, which today typically represent the first step in CAE/CAD. Paula Filseth, in "Benchmarking Schematic Entry Systems," chooses four popular programs—three from CAE/CAD vendors and one from a major ASIC company—and examines the features of each from a user's point of view, concentrating on those that make for ease of use. She then gives the time required to enter a benchmark LSI circuit with each system and presents an overall evaluation of each.

Hardware description languages provide an alternative or supplement to schematic entry, giving a designer the ability to describe and design very complex components and systems using a high-level language geared to that purpose. In the second article under "Logic Design," "A First Course in VHDL," David Barton of Intermetrics offers an introduction to the federally sponsored VHASIC Hardware Description Language, which seeks to aid designs done under the government's Very High Speed Integrated Circuits program.

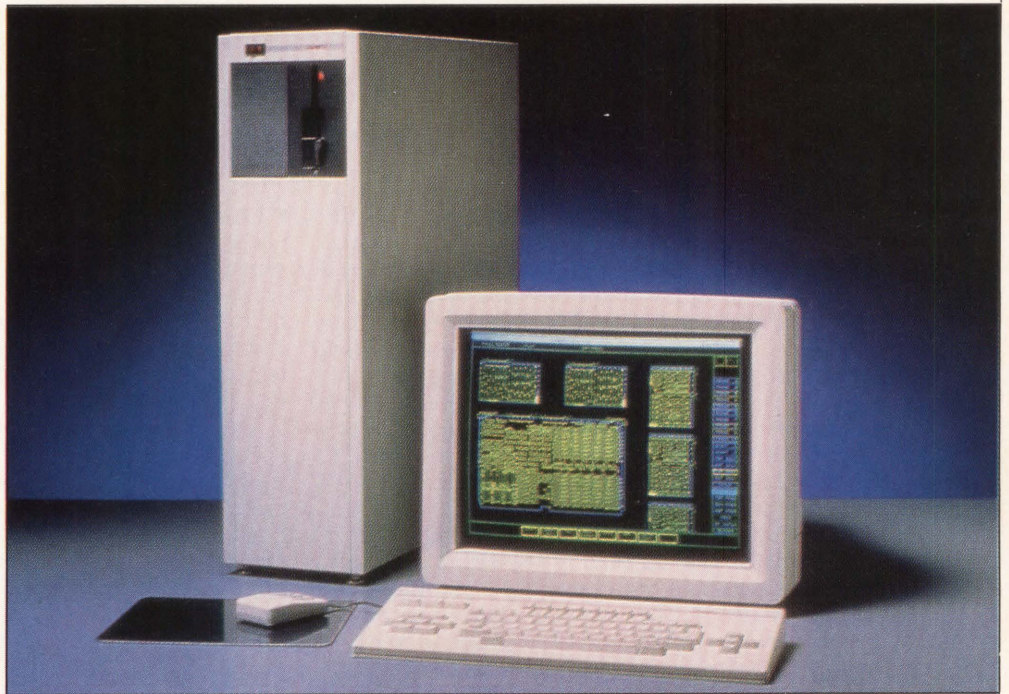
Here Barton emphasizes the areas of register-transfer descriptions of behavior and structural descriptions of circuits.

Ideally, after a design has been simulated, an engineer will want to see how it works in its intended system. For many systems that means simulating a 32-bit microprocessor or other LSI or VLSI standard parts, or both, and that poses the problem of modeling these complex devices. Gateway Design Automation's Pete Johnson evaluates the two techniques for modeling complex components in "Software vs. Hardware Models for System Simulation." Describing the trade-offs involved, Johnson shows that the conventional wisdom, that software models are slower than hardware models and are therefore most appropriate for simulating devices not yet available in silicon, is not always true.

## IMPLEMENTATION

Once a logic design is completed, it must be turned into a physical design that is both correct and compact—be it a chip or a printed circuit board (and of course, if it is a chip, the circuit board it goes on will have to be laid out and routed as well). Some of the tasks involved in the physical design of chips and boards are considered in Section III.

In virtually every methodology, leaf cells are the basic units for building up an IC design. M.Y. Tsai and Stephen Wu of ECAD present a tutorial on leaf cell design, entitled simply "Leaf Cell Design," and give guidelines for creating dense cells. They also describe how a sym-



bolic layout tool speeds the design process and makes possible process independence, so that the leaf cell library can easily adapt to changes in technology.

Standard cells, blocks, and macros are built up from leaf cells. Typically, a floorplan is the first step in a semicustom chip design, serving as a general guide for the layout of these larger units. In "Graphical Floorplan Design of Cell-Based ICs," Tektronix's Edmond Macaluso discusses the basics of floorplanning and describes a program that handles the various tasks involved.

Our last article takes up the subject of printed circuit board design. In "A Rules-Driven Approach to Circuit Board Design," Joseph Prang and Katherine Gambino, form Valid Logic Systems, detail an expert-system layout tool that enables the CAE engineer to specify implementation rules to the PCB designer, thereby integrating electrical and physical specifications. A design example helps clarify the methodology.

Concluding the guide are a directories section, providing detailed listings of systems for CAE, IC layout, and printed circuit board layout, and a references section, presenting a select guide to the literature and a subject index to *VLSI Systems Design* for 1986–1987. □

# EDA PUSHES TOWARD LOGIC SYNTHESIS

George Bouhasin, Advanced Development, Mentor Graphics Corp., Beaverton, OR

**W**hen a new design is first conceived, it is described using high-level component blocks and connectivity information. When it is first entered into an electronic design automation (EDA) system, it is defined as a network of low-level primitive design elements. Today, the process of converting the high-level specification into the low-level structural representation requires that the system architect delve inside each block and partition elements manually until each block is described at the primitive level. As shown in Figure 1, the conversion consumes the majority of man-hours on any typical design project.

Years ago, structural schematics were made up of transistor primitives exclusively. In the era of SSI chips, logic gates and registers were standard primitive elements. Primitive design files now contain a host of MSI and LSI parts. However, while primitives keep increasing in complexity, system design structures also are becoming larger and more complex. As a result, the task of decomposing a high-level description into a primitive network is not getting any easier. In fact, now that EDA systems have streamlined design analysis and automated physical layout, the front-end task of creating the design file often consumes more design time than any other phase of development.

There is widespread interest in EDA tools that can automate design file creation, especially among system designers who are more comfortable working at higher levels of design description. The design methodology that is slated to answer this demand is known as logic synthesis. Today, logic synthesis tools are commercially available for PLDs and are under development for gate arrays and standard-cell ICs. To deliver synthesis tools that answer the pent-up demand of the entire IC and board design market will require an integrated solution, one which results from close cooperation among major EDA system suppliers, third-party software vendors, and semiconductor manufacturers.

## SYNTHESIS BUILDING BLOCKS

The principal requirement of a logic synthesis system is the ability to take a high-level description and produce

a structural representation with little help from the user. However, this ability is not in itself enough. The synthesized schematic must also meet design goals other than the functional requirements outlined by the high-level design description. Toward this end, the synthesizer must be capable of making trade-offs among factors such as cost, power requirements, and space utilization.

Also, when selecting parts to include in the structural schematic, the synthesizer should have access to an expansive collection of parts libraries so that it can select existing components rather than promote the need for custom parts. Lastly, all major silicon technologies should be supported by the synthesizer so that the most appropriate process can be used to implement the design.

Many of the components needed to build an integrated design system with logic synthesis are already in place. For example, system architects can already define their high-level system components in one of a choice of formats that is most natural to each particular block. Schematic capture editors, now allow designs to be described as high-level behavioral models, as well as entered as schematic information.

Eventually, capture editors will be able to accept any number of input description forms, including hardware description language (HDL) models that define only function and connectivity, truth tables, Boolean equations, state assignment tables, and algorithms (see Figure 2). It is likely that some graphic input formats unlike those now in use also will emerge. Additionally, it would be advantageous to have graphic output, in the form of a schematic display, for each component block at every level in the hierarchy.

## TECHNOLOGY TARGETING AND LIBRARIES

To automate the component selection process, logic synthesizers must be able to target both the methodology—PLD, gate array, standard-cell, full-custom, or off-the-shelf standard components—and the intended process—CMOS, ECL, gallium arsenide, and so forth. Each methodology and process has its own idiosyncrasies,

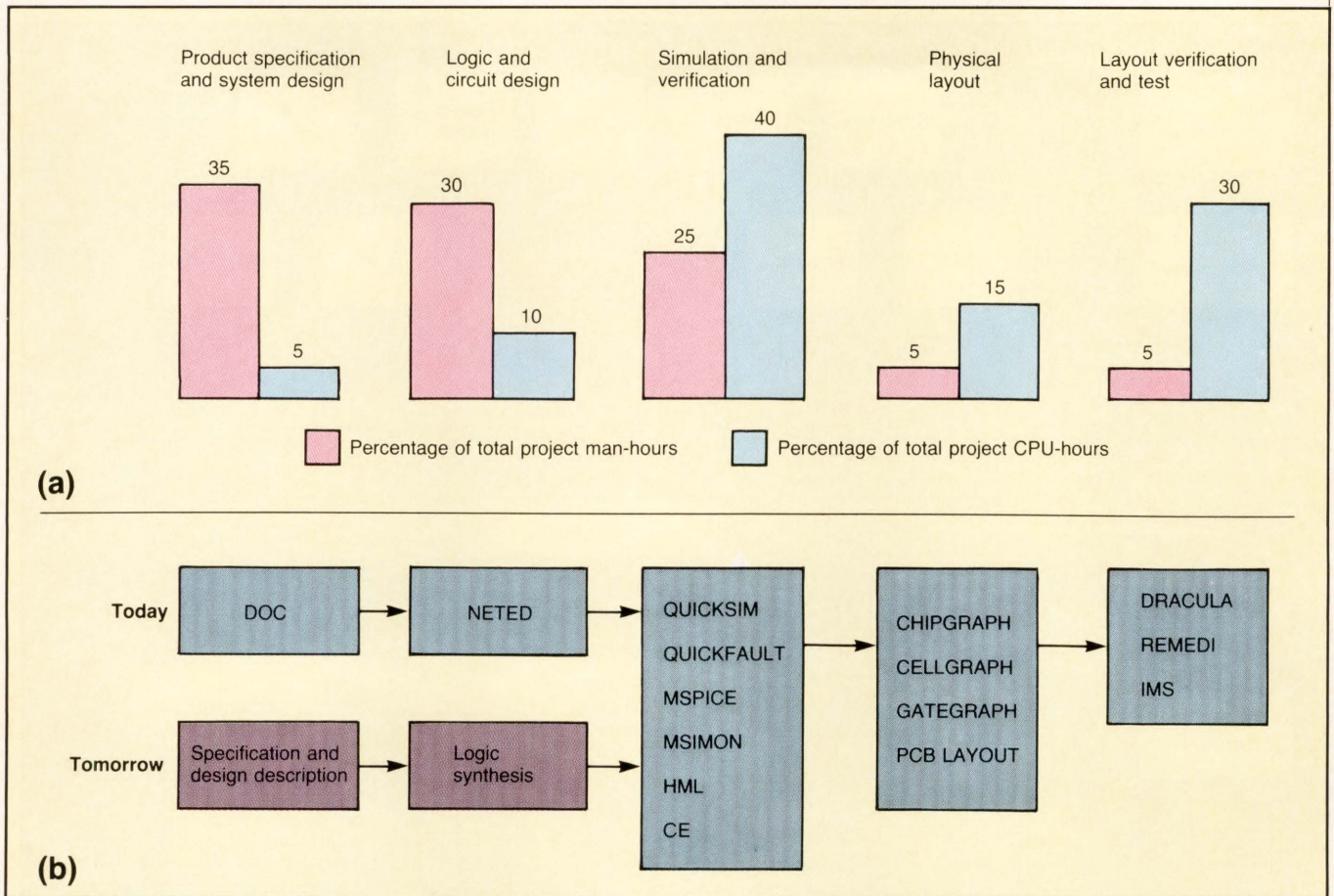


FIGURE 1. The system design process in terms of man-hours and CPU-hours (a) and today's and tomorrow's tools at Mentor Graphics (b).

and a designer can work much more efficiently if they are well understood. The same is true for a synthesis system. Thus logic synthesizers are likely to include a *technology discriminator*, a tool that selects among available choices for methodology and process.

For this purpose, a logic synthesizer will include a *knowledge base* that describes the rules of each methodology and process technology that it supports. For example, when synthesizing a design for a CMOS technology, the system should understand the common rule of thumb that logic can be minimized through the use of inverted signals and complex gates. There are many such rules that can be enforced using expert system techniques. In fact, Trimeter Technologies, a Pittsburgh-based company, has already commercialized a knowledge-capture tool for gate arrays. It also offers knowledge bases for ASIC product lines from such companies as LSI Logic Corp.

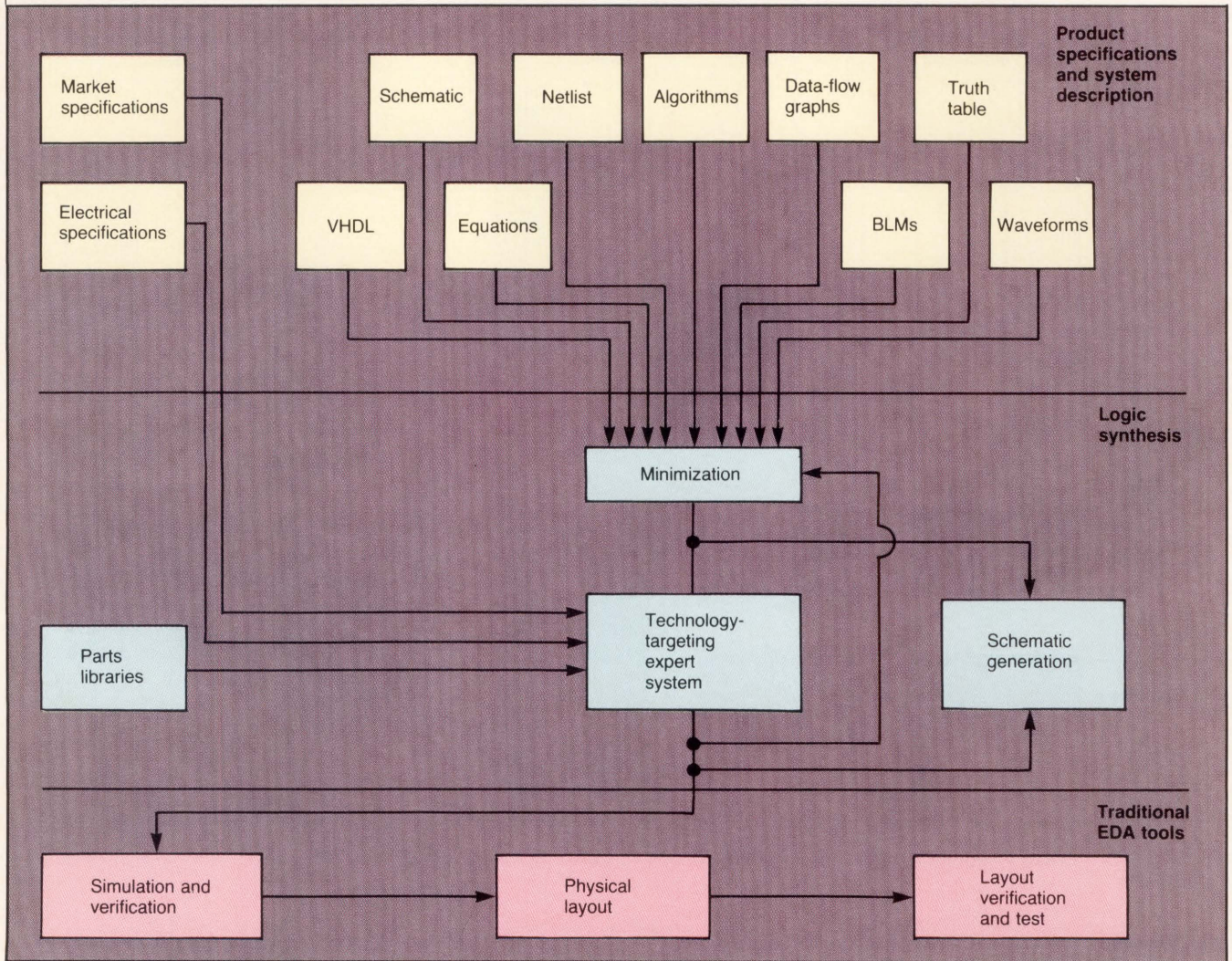
Of course, without accurate, complete component libraries, the synthesis process has nothing to target. Fortunately, semicustom and standard-parts libraries now provide a wealth of components that are supported by semiconductor manufacturers and qualified by EDA vendors. System architects can now pull models of stan-

dard parts, offered by a range of manufacturers, from their network libraries to include in design schematics and simulations.

Microprocessors and other more complex parts can be represented with behavioral language models supplied by the EDA system or written by third-party software vendors such as Logic Automation and Quadtree. If the VLSI parts are available in hardware, hardware modeling libraries, such as Mentor Graphics' HML, can be used to generate a model for simulation quickly.

ASIC designers now also have a range of library options. Many ASIC foundries have ported their cell libraries to the leading EDA systems; consequently, EDA vendors can offer to designers a selection of gate array and standard-cell libraries. It is important that synthesis tools have access to libraries for all available design methodologies, so that feasibility analyses can evaluate all possible implementations to find the best method for a particular design.

Feasibility analyses will be performed by expert-system-based tools that consider marketing goals (production, volume, time-to-first-prototype, price, and the like) and technical specifications (power, performance, temperature, and the like). These analyses can



**FIGURE 2. An EDA system of the future with logic synthesis.**

be “rule-of-thumb” analyses, giving recommendations based on knowledge of good design practice and previous successful designs; alternatively, they may use analysis tools in the tool suite to make more exact comparisons.

For the expert systems to evaluate all those factors, the type and breadth of information in component libraries must be expanded. With SSI- and MSI-level catalog components, for example, only the barest timing and power specifications (usually worst-case numbers) are included in timing models; more precise information would be needed for logic synthesis. More complex chips may have more timing data, but manufacturing data like volume pricing, delivery schedules, and package types could be used by an expert system to evaluate the impact of employing the chips in production systems. The inclusion of manufacturing data, and the development of the tools to evaluate design trade-offs in terms of manufacturability, is just beginning.

In the future, major semiconductor vendors will offer ASIC library cells that are similar to today’s standard components. Vendors such as Intel and Motorola will provide microprocessor core cells; chip makers in the scientific-processing market will provide floating-point processing cells; others may sell graphics controller cells. System design would be more efficient if these commercial cells were predesigned, modeled, and characterized, and if they had the potential for multiple sourcing at the physical layout stage. It is likely that they will be offered alongside standard packaged parts, so that system designers will always have the option of designing either a traditional printed circuit board or, in effect, a “silicon PCB.”

Another significant change for library users may be the advent of huge dial-up databases for both cells and standard parts. This type of service has just begun to supplement the standard-component data books that have lined designers’ bookshelves for decades. These

part databases contain cross-references and some component attributes like timing. Tomorrow's databases will have many more attributes, including size, cost, and listings of supported simulation model libraries. Therefore they may serve more as a reference than as a source.

When logic synthesis systems with their technology-targeting capabilities are in place, designers will have an effective means of querying these databases and taking advantage of such detailed component information. Because the databases will be used as a reference, they will not be as tightly linked to the design system as the component libraries on the design system's network. Dial-up links will probably suffice.

The semiconductor foundries must take responsibility for the component libraries. They can either develop and support the models themselves or contract with outside vendors to develop them. Competitiveness will go to those foundries that make the most information about their parts available for the least cost. In addition, the foundries have to broaden the types of information that they provide, for the expert-system programs will need detailed component and manufacturing data. Pricing, reliability, availability, packaging, full temperature and performance curves, as just a few examples, all may be needed.

## SYNTHESIZED DESIGN METHODOLOGY

In the era of logic synthesis, designers are likely to have great flexibility in describing their systems. The input description method will vary depending upon the type of logic to be entered and the amount of detail the user wants to provide. For example, it will be possible to enter combinational logic blocks as a schematic, a Boolean equation, or a truth table; in the same system, a complex counter block may be entered as a state assignment or a functional description.

Synthesis tools could then be used to translate these descriptions into some number of high-level component blocks that define system behavior. Designers may prefer that the blocks be displayed in a schematic so that they can verify the logic.

If simulation models were available for selected components at each level, a high-level simulation of the whole system could be performed early in the process. Simulation could continue for subcircuits in the design after the design has been partitioned into lower-level primitives.

Synthesis should also afford the opportunity to capture the *user's design goals*, which then would become objectives that guide the synthesis process. For example, if the overriding goal were to complete the design in the shortest amount of time, the designer would want to avoid the need to customize components. In this case, the synthesizer could come closer to the ideal implementation if it could give preference to using existing parts—even if they provide some superfluous logic—and treat the classic design objective of minimizing the number of logic elements as a secondary goal. Likewise, if performance were the most critical design factor, the synthesizer should allow extra logic to provide redundancy in

critical control paths (for instance, "look-ahead-carry" circuitry).

Goal-capture tools should also support *decision making*. Designers will need a means of collecting and organizing data so that they can run feasibility analyses on prospective architectures and primitives. The amount of information might otherwise be overwhelming; for example, they may want to gauge the effect of processing and design methodology options on not only the size, power, and performance of their designs but also their cost and time to market. For the same reason, designers are likely to want synthesis tools to include *expert-system building capabilities*, so that they can create knowledge bases that contain their own design and manufacturing experience.

Once a designer has selected a methodology and process by working through his own feasibility analyses, or using tools such as the technology discriminator, the information will be available to synthesis tools for technology targeting. That is, the synthesizer can begin to query part databases to retrieve components that are based on the appropriate technology and that match the user's design goals.

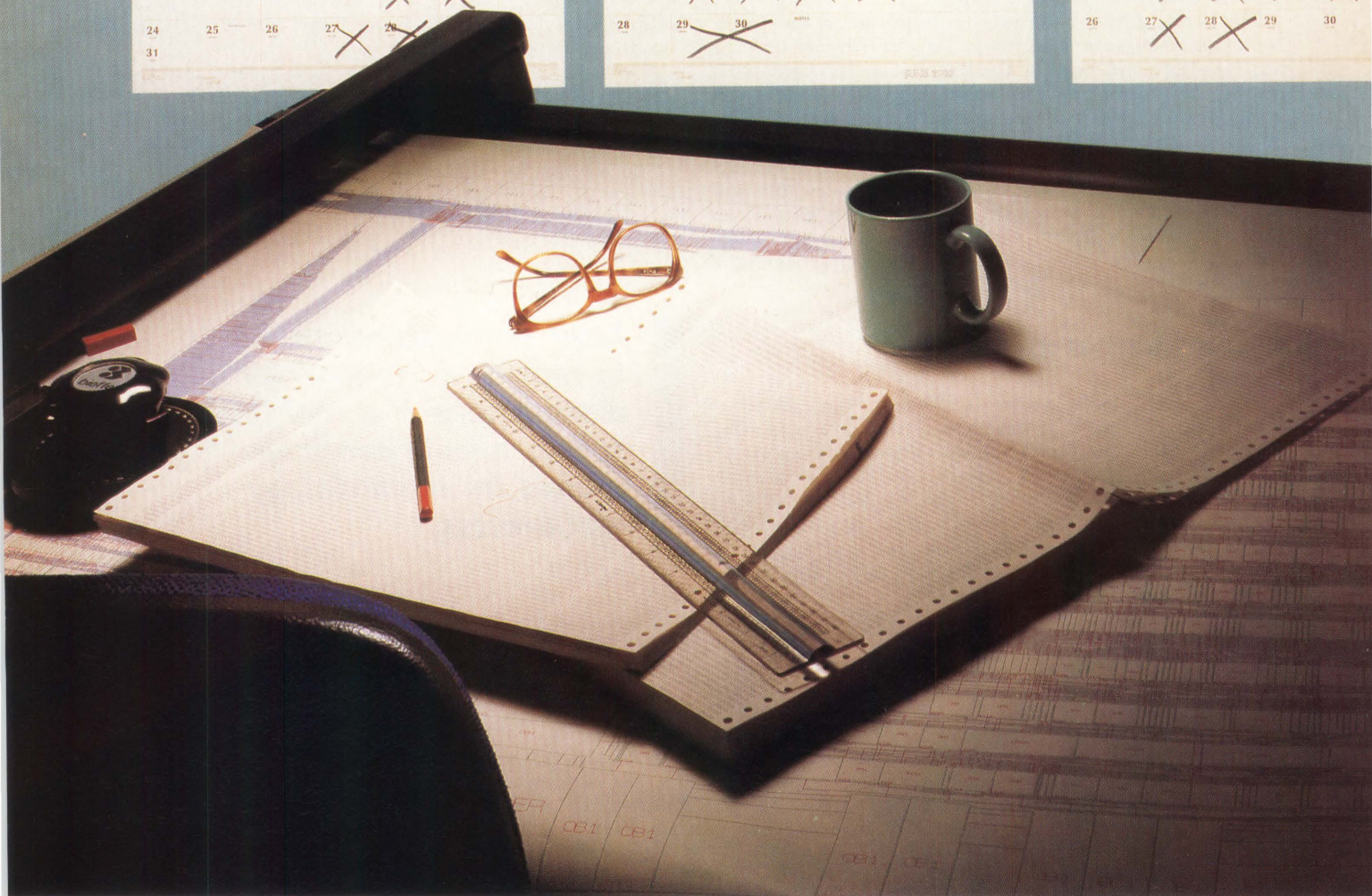
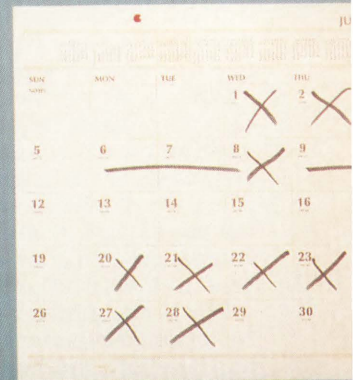
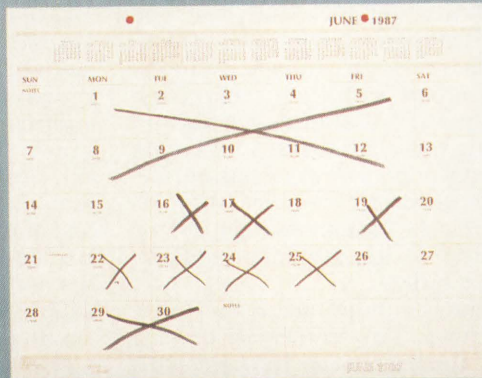
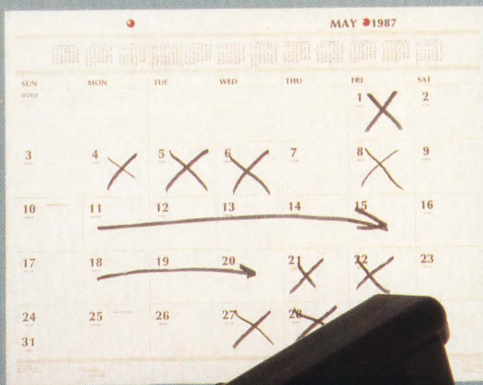
**Logic synthesizers are likely to include a "technology discriminator," a tool that selects among available choices for methodology and process.**

Block by block, the synthesizer can then replace lower-level schematics with higher-level components. The design goals specified by the user can be applied either component by component or at later stages on completed portions of the design. If the schematic components that replace a higher-level logic block still present more functionality than desired by the user, this level can be further decomposed. The process will continue until appropriate primitives are determined for the complete design file.

Also, as each block is synthesized, the system might build in resettable scan-test registers to ensure testability. When such design-for-test architectures are implemented, component test patterns for automatic test pattern generation will have to be made available. Ideally, the semiconductor manufacturers that offer components and models will also offer the test vectors.

As schematic information becomes more detailed, new simulations and new, more accurate analyses of cost, performance, power, and other design goals should be run. After the logic synthesis step, the design can proceed to physical layout, where traditional structured design methods are applied. Although the analyses may seem to require enormous CPU power, the new breed of

# Three months



## Compiled Cell Custom from S-MOS saves you valuable ASIC design time.

With S-MOS' new Compiled Cell Custom program, you can cut up to three months from your custom IC design cycle.

Simply by pressing three buttons.

C.C. Custom automates the design process by combining S-MOS' advanced LADS software with Tangent's Tancell.<sup>®</sup> The program allows you to create new designs based on standard

cells, our megacells, your megacells or any combination.

You can mix and match more than 300 standard macrocells simply by doing the schematic capture and simulation steps that you would do for a gate array.

The circuit is then laid out and routed automatically at our design center. The system is timing driven, so

new user-defined macrocells will match their simulations. The first time and every time.

With a full custom design, you typically have to wait half a year for the first chip samples. C.C. Custom can do it in 12-14 weeks with NRE charges normally associated with standard cells. Or about half what a full custom would cost.

So you get your products to market faster and more economically.

Since we produce our chips on the 1.5 micron line of our affiliate,



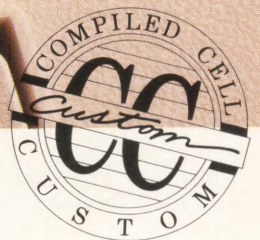
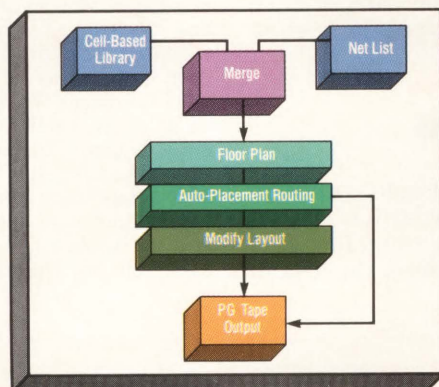
# or three buttons.



Seiko Epson, you will also benefit from higher-speed operation and virtually nonexistent failure rates. Seiko Epson is one of the world's most automated and reliable CMOS manufacturers.

If you'd rather press three buttons than spend three additional months designing circuits, call us. (408) 922-0200.

S-MOS Systems  
2460 North First Street  
San Jose, CA 95131-1002



**S-MOS**  
**SYSTEMS**  
*A Seiko Epson Venture*

Tancell is a registered trademark of Tangent Systems.

CIRCLE NUMBER 7

## **An expert system could use manufacturing data on standard parts, like volume pricing, delivery schedules, and package types, to evaluate the impact of employing the chips in production systems.**

multiple-MIPS workstations seems to provide enough processing power for the software requirements.

To aid the development of such design systems, third-party software developers will probably help develop the knowledge bases for particular applications. They should, in the long run, continue to provide specialized algorithms and translators to fill the niches in the spectrum of design applications.

### **TRUE SILICON COMPILATION**

Logic synthesis is actually a subset of a more comprehensive future methodology, a methodology that reduces the development process to a matter of entering a design concept as a high-level description and receiving a physical layout in return. The next step in the evolution toward such "true silicon compilation" would be to use logic synthesis tools at the front end and structural synthesis through semicustom design methodologies at the back end.

Although today's silicon compilers automate much of the low-level decision making necessary to translate a schematic into silicon, they still require that a design be described mostly at the structural level. Thus, from a front-end standpoint, they are only somewhat more automatic than other layout tools. Without logic synthesis, designers must still demonstrate a high degree of "silicon literacy" to build an IC. In contrast, once logic synthesizers are available, all interaction with the EDA system can be handled at a relatively high level. Then tools that operate like current silicon compilers will be a more attractive option for those designers looking for an automatic solution to IC design.

For the automatic solutions, the EDA system vendors will be responsible for the design environment and analysis tools. They must also provide the interfaces to sources of data that designer will need access to, such as dial-up databases.

### **THE LOGIC SYNTHESIS PROVING GROUND**

The major commercial market for design synthesis now is in PLD design. Synthesis technology has evolved more rapidly here because PLD combinational logic is relatively simple and well constrained. Also, the task of

translating a conceptual design description into a PLD format was a natural focus for automation, because it took so long compared with the minimal time spent programming the prototype once the logic was defined.

Most of the PLD minimization programs now available, including the popular ABEL from Data I/O, are based on Espresso, a public-domain program developed at the University of California at Berkeley. Espresso uses algebraic methods to automatically synthesize and optimize combinational circuits. ABEL automates the PLD design process by combining Espresso capabilities with advanced design capture methods that allow input descriptions in many forms, including Boolean equations, truth table, or state assignment table.

The next likely application area for commercial synthesis tools will be gate array design. Today, there are tools under development that attempt to synthesize the multilevel logic commonly used in gate arrays (Brayton, 1986; Brayton et al., 1986). Also, in the near term, state machine tools that synthesize some sequential logic will be applied to PLD and standard-cell designs. This work will eventually be commercialized by vendors developing logic synthesis systems.

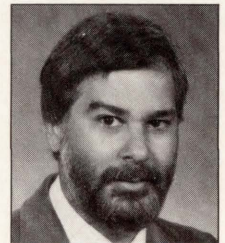
Thus the pieces of the synthesis puzzle are beginning to fall into place. According to the Technology Research Group (1987), "the worldwide market for logic synthesis will grow from virtually nothing in 1986 to roughly \$200 million in 1990." However, it will take a concerted effort by EDA tool vendors, chip makers, and third-party software suppliers to provide the integrated solution that is needed. □

### **REFERENCES**

- Brayton, R.K. 1986. "Algorithms for Multi-Level Logic Synthesis and Optimization," IBM Corp., Thomas J. Watson Research Center, Yorktown Heights, NY (paper submitted to the NATO Advanced Study Institute).
- Brayton, R.K., et al. 1986. "Multiple-Level Logic Optimization System," *IEEE International Conference on Computer-Aided Design*, Santa Clara, CA.
- Technology Research Group. February 1987. "Logic Synthesis Is New Opportunity in Design," *The Technology Research Group Letter*, Boston, MA.

### **ABOUT THE AUTHOR**

**George Bouhasin**, the advanced development manager at Mentor Graphics, has 15 years of semiconductor experience in design, processing, test, production, and marketing. In 1979 he helped start California Devices Inc. of San Jose, CA, where he developed the first no-channel gate array. In 1984, he cofounded a silicon compilation company that was later acquired by Mentor Graphics. Now, in advanced development, his interests include logic synthesis and the application of expert systems to CAE. He received his BSEE from the University of Cincinnati.



# Chip, Chip, Array!

## For High Speed with Low Power.

AMCC has the chips worth cheering about. When you need the versatility of high speed with low power in a bipolar array, our Q5000 Series Logic Arrays are the answer. They're designed for logic applications requiring speed/power efficiency. And they deliver.

Today's hi-rel commercial and military semicustom applications need high performance and proven reliability. And, our Q5000 Series gives you both—without paying the power penalty.

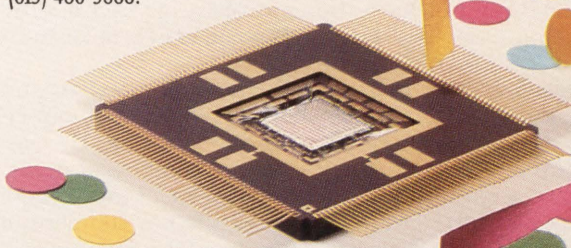
Our newest bipolar series is comprised of five arrays. All feature 4 levels of speed/power programmable macros and over 600 MHz I/O capability. One comes with 1280 bits of configurable RAM.

### Q5000 Series Key Features

Equivalent Gate Delay:	210-545ps
Flip/Flop Frequency:	>600 MHz
Power Per Gate:	1mW
Speed/Power Product:	0.5pj
Equivalent Gates:	1300-5000
I/O Pads:	76-160
Operating Temperature Range:	-55°C to +125°C

AMCC Bipolar Logic Arrays have been designed with other flexible performance features in mind, too. Mixed ECL/TTL I/O compatibility. Your choice of packaging. Full military screening. AMCC's MacroMatrix® design tools. And, unrivaled customer support.

To talk with an applications engineer about your specific needs, in the U.S., call toll free (800) 262-8830. In Europe, call AMCC (U.K.) 44-256-468186. Or write, Applied MicroCircuits Corporation, 6195 Lusk Blvd., San Diego, CA 92121. (619) 450-9333.



A Better Bipolar Array is Here.

**AMCC**

# BUILD BETTER HARDWARE BY FOCUSING ON SOFTWARE

Cindy Thames and Andrew S. Rappaport, The Technology Research Group Inc., Boston, MA

In most electronics system design projects, hardware designers outdo each other to escape being stuck with programming the embedded software. Indeed, when dedicated programmers handle the embedded software portion of a project, the hardware designers tend to consider themselves relieved of any concern about software. Software, however, is becoming a more critical, complex, and time-consuming aspect of system development, adding to the development difficulties brought about by ever increasing hardware complexities. Indeed, it has become the bottleneck in an increasing number of system designs. Yet little has been done so far to merge hardware and software design.

Designing the software portion of a hardware system is tedious and error-prone, as well. Software programmers have to make do with the equivalent of slide rules and drafting paper, while an array of sophisticated tools running on powerful computers supports the development of hardware. High-level languages simplify software development, but going from assembly to a high-level language is like going from transistor-level design to TTL. Programmers involved in high-performance system design do not have the equivalent of VLSI building blocks that represent thousands instead of tens or hundreds of primitive elements.

As designers use more VLSI for hardware design, and design automation tools for hardware become more effective, the problems of programmers worsen. By contributing to the development of more complex systems, the growing use of VLSI increases the need for much more complex software; and by speeding the development of hardware, hardware-design tools increase the percentage of the total development effort required by software. Lastly, the increasing use of custom VLSI made possible by improving hardware-design tools recasts the hardware-software development process into a much more

interactive process than before.

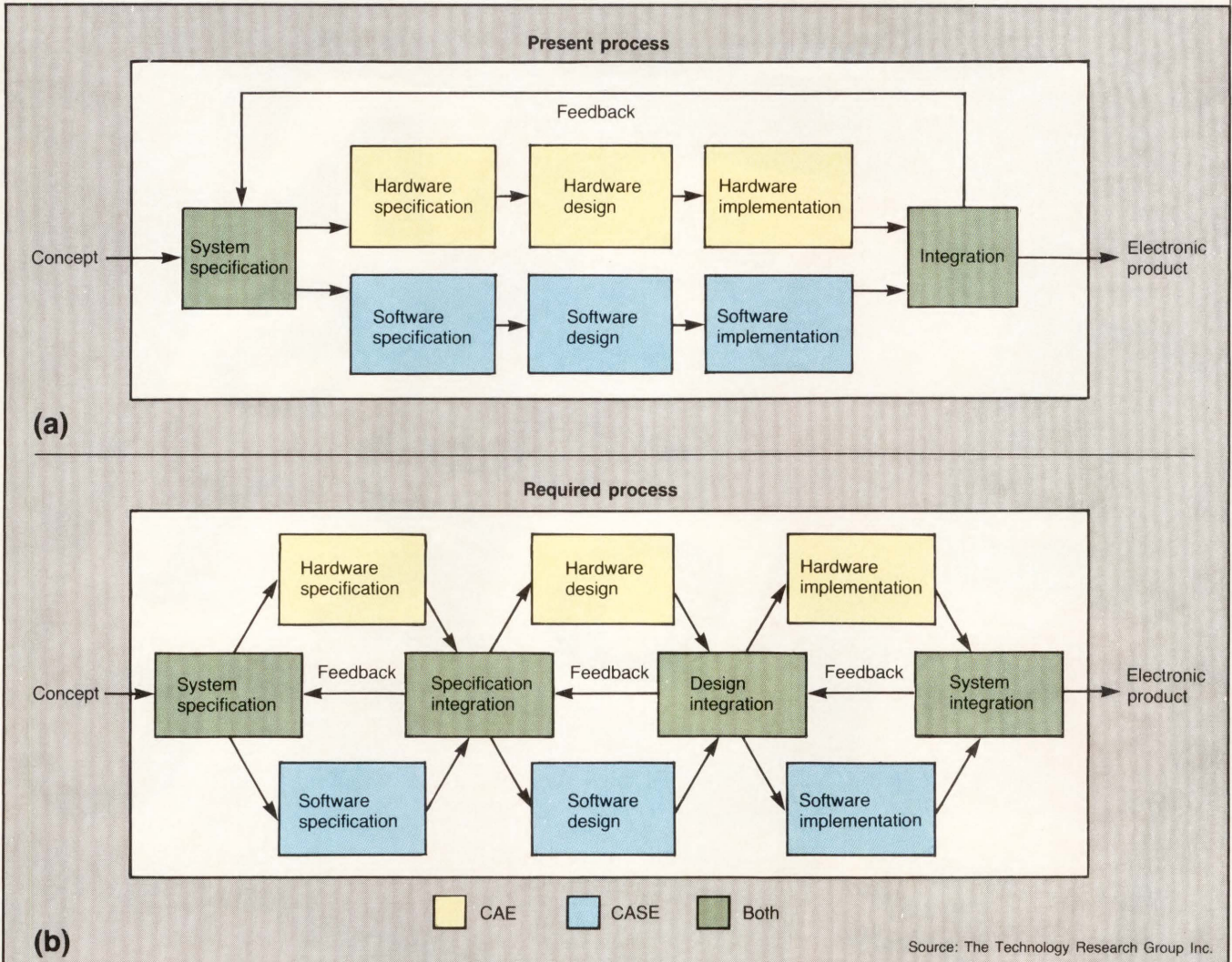
Consequently, trends in the development of electronic systems are forcing more hardware designers to become involved with software development, either directly, as programmers, or indirectly. That is, the interdependence of hardware and software portions of increasingly complex systems demands greater cooperation between hardware and software designers.

Interdependence also demands better tools. The lack of automation for embedded-software development holds up total system debugging and delays the time to market for new electronics systems. The lack of a design methodology that merges distinct hardware and software design processes into an integrated whole (see Figure 1) detracts from the overall benefit of existing tools for hardware design.

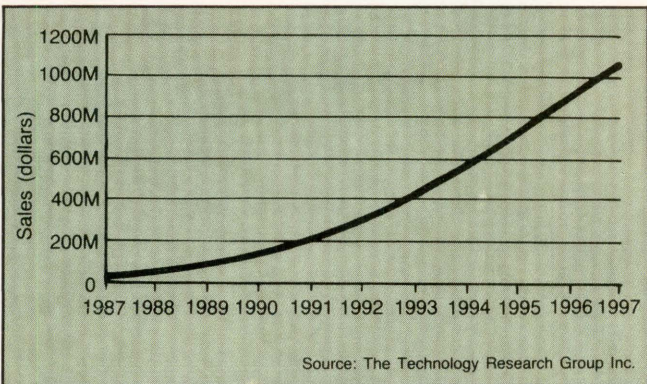
The market for embedded-software development tools will be integral to that for hardware-design tools. Hardware-design automation suppliers must make every effort to draw in software-development aids—not only at the point of system integration, which they are doing now, but also as far back as system analysis. We project that the market for electronic-design automation software will reach \$2 billion by 1990. The potential for tools for creating software for these systems is at least as great, although the market is developing much more slowly. It will be, we believe, the next major growth market in design automation, exceeding \$200 million by 1991 and crossing the billion-dollar mark by 1997 (see Figure 2).

## A MOVING BOTTLENECK

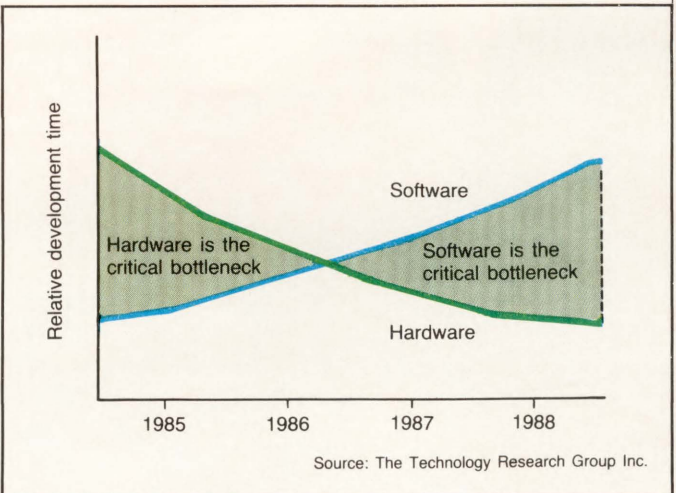
Despite the limitations of some existing tools and the immaturity of some recent ones, suppliers of hardware-



**FIGURE 1.** In the present design process (a), hardware and software design are essentially separate. With software development becoming increasingly complex and time-consuming, what is needed is an integrated design process (b).

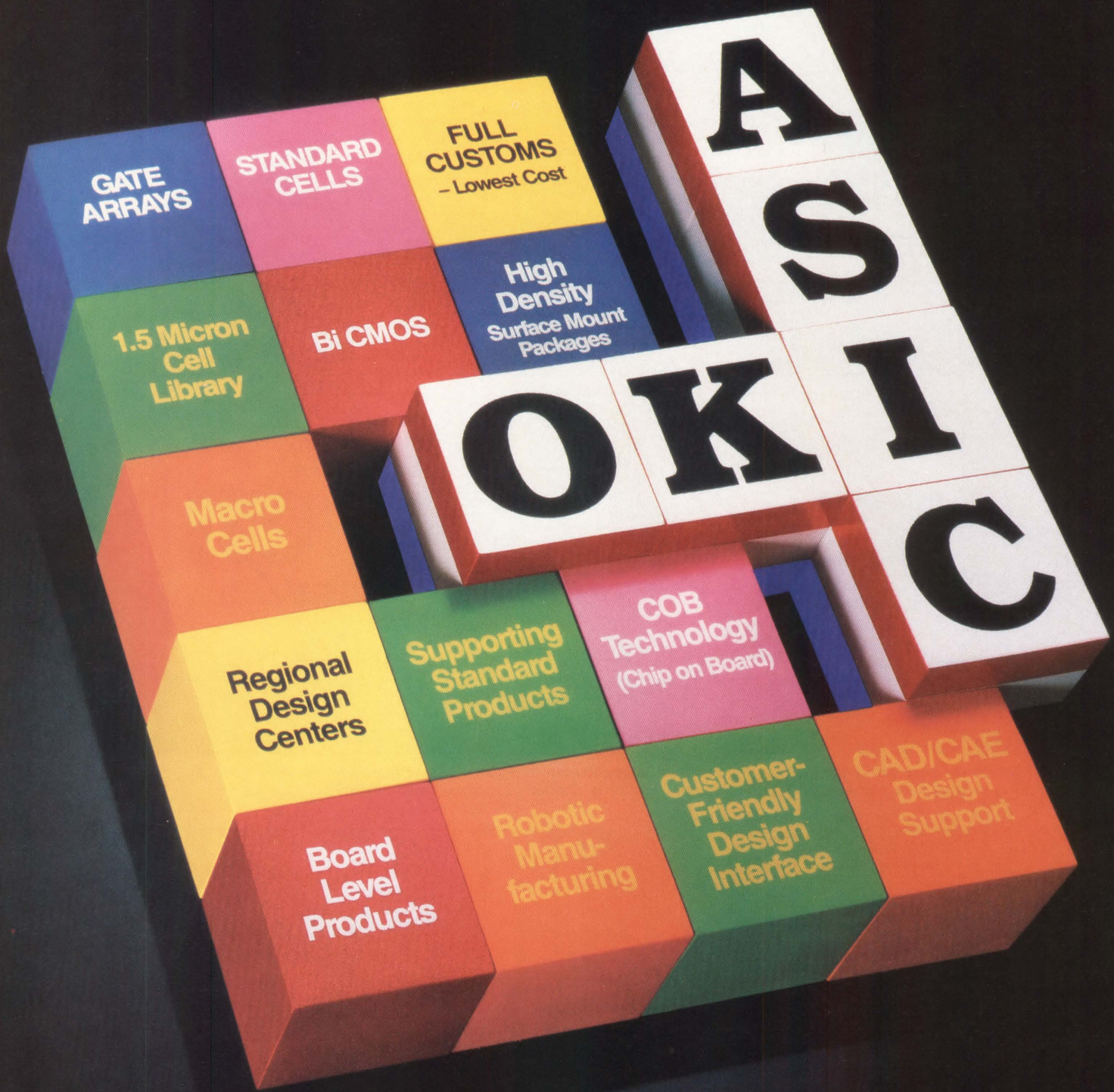


**FIGURE 2.** The projected market for software tools to aid in the development of software for custom hardware.



**FIGURE 3 (right).** The moving bottleneck in electronic-system development.

# Easy ASIC



# Only from OKI:

- Most **complete** ASIC building blocks.
- Most **versatile** design/package options.
- Most **experienced** ASIC technology.

Nobody but nobody puts ASIC technology together like OKI Semiconductor can.

Ease into ASIC with OKI as your close working partner—and you instantly support your VLSI application with the most comprehensive ASIC capabilities on the world market today. Bar none.

From gate array, standard cell and full custom chips to standard components to integration to advanced board level products, OKI alone puts you on the leading edge of ASIC technology and its complete implementation.

## OKI: the totally logical choice.

Opt for OKI ASIC, and you open up your options across the board. Only OKI now offers the system designer the unique security and entry ease that only a proven track record in CMOS ASIC problem-solving can provide. This history of performance built up since 1977 has produced the widest range of solid building blocks yet: advanced ASIC products and packaging including surface-mount, backed up with the most flexible cell libraries, CAD/CAE design tools and development aids.

As your working partner, OKI ASIC expertise is available to you at any stage of the development process. We'll help you define system requirements, determine the most cost-effective product solutions and supply complete design software—accessible at your own workstation or through our regional design centers. And then we take it from there: with high

volume fabrication, assembly and testing completed in one of the world's most highly robotized manufacturing facilities.

Compare Total ASIC Capabilities	OKI ASIC	Source "B"	Source "C"
Gate Arrays to 10K Gates	●	●	
Standard Cells to 30K Gates	●	●	
Full Customs – Lowest Cost	●		
1.5 Micron Cell Library	●	●	●
Macro Cells	●	●	
Bi CMOS	●		
High Density Surface Mount Packages	●		
Board Level Products	●		
Supporting Standard Products	●		
COB Technology (Chip on Board)	●		
CAD/CAE Design Support	●	●	●
Customer-Friendly Design Interface	●	●	●
Regional Design Centers	●	●	●
Robotic Manufacturing	●		

ASIC Solutions from **OKI:**  
**You can't beat the logic!**

## Check out OKI ASIC data:

VLSI 12/87

Please rush complete technical data/specs on OKI capabilities in:

- ( ) Gate Arrays
- ( ) Standard Cells
- ( ) Full Customs
- ( ) Please call: we have immediate requirements:

Name/Title \_\_\_\_\_

Company \_\_\_\_\_

Attach coupon to business card or letterhead and return to: ASIC Customer Service, **OKI Semiconductor**, 650 North Mary Avenue, Sunnyvale, CA 94086. Tel: (408) 720-1900.



CIRCLE NUMBER 8

design automation tools have done a generally good job addressing the critical bottlenecks that inspired the current generation of electronic-design automation tools. As a result, the bottlenecks have moved (see Figure 3). A further quantum step in hardware development will come about only when design tool suppliers meet the challenges of concurrent hardware and software design. In the meantime, hardware designers can begin to use existing tools to bring software cognizance to the hardware-development process.

The need for automation is more acute for the development of embedded software than for any other type of

**System optimization is the practical, obvious, immediate reason for hardware designers to care about software development. At a higher level, the efficient production of complex electronics products requires balancing the needs of hardware and software development.**

software development. Unlike commercial software products and management information system (MIS) code, embedded software is often designed for hardware with performance and functional characteristics not fully defined or definable at the outset of software development. This indefiniteness complicates software development, putting hardware development, prototyping, and debugging in the critical path—and with hardware design in the critical path of software, and software in the critical path of hardware, it is surprising that systems are ever completed.

Without concerted attention, the problem will get much worse. The complexity of embedded software is exploding. Respondents to a recent survey of software programmers conducted by the Technology Research Group in conjunction with L.F. Rothschild and Co. (Technology Research Group, 1987) indicated that the median length of an embedded software program was 10,644 lines of code in 1985 (see Figure 4). The median length expected in 1989 is three times that: 30,394 lines. During the same period, the median length of commercial applications code will grow from 11,087 lines to 18,069, according to the respondents. The median length of MIS code will increase by only a few percent, from 4,230 lines in 1985 to 4,380 in 1989.

Tools for software development—so-called computer-

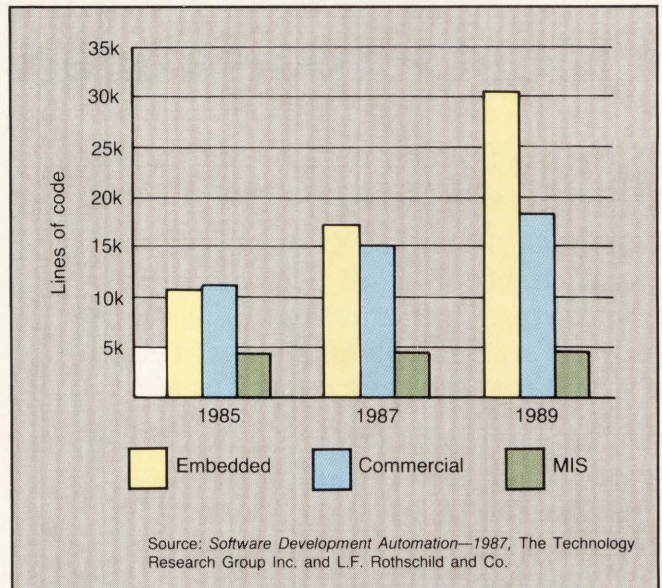


FIGURE 4. The median length of software programs.

aided software engineering (CASE) tools—are beginning to constitute a new industry, modeled in large part on the electronic-design automation industry. Yet most of these tools are structured for general software development or, if targeted at applications involving custom hardware, adapted from methodologies developed for MIS or commercial development. For the most part, suppliers of CASE tools have not differentiated between products for embedded software for custom hardware and those for software designed independent of hardware.

The complexity of designing real-time systems, the difficulties of developing hardware and software concurrently, and the demand for reliability all compound the difficulty of creating embedded software. This difficulty is the hardware designer's challenge, as well as the software developer's job.

## BLAME CUSTOM SILICON

The trend toward custom CPUs, brought about in part by better access to custom and semicustom silicon, is breaking down the former division of labor for embedded-software development. When hardware designers used only standard CPUs, they needed to concern themselves only with the embedded software programs written in a standard CPU's instruction set. Only CPU designers or developers of specialized processing systems needed to develop microcode. That has been a small group.

As custom and semicustom IC technology and better hardware-design tools make custom processor design easier and more economical, they put the ability to design CPUs into the hands of more hardware engineers and make custom processor design appropriate for a broader range of systems. That means that more hardware engineers are facing the necessity of developing microcode. The penetration of semicustom IC technology



has expanded the purview of the system-level designer—for both hardware and software. Just as semicustom is turning system designers into part-time IC designers, it will turn them into part-time programmers.

Already, 18% of all CMOS gate array designs are used in custom processor architectures. By 1990, that percentage will increase to 21% (Rappaport, 1987). Chips that integrate several large LSI blocks will represent 34% of CMOS array designs and more than 50% of cell-based semicustom by 1990. Many of those will include core microprocessors, often with customized instruction sets or architectures. All will include software development and verification using software as integral parts of the chip design and verification process. Abundant automation tools will help them design the chip, but paltry ones will help them design the software.

Even for designers not directly charged with microcode or software development, increasing freedom to create custom hardware architectures affects and is affected by software development. Bit-slice systems are a good example. Designers who use standard bit-slice components have limited ability to optimize architectures at a low level. The coarse granularity of standard component design approaches imposes those limits. They can optimize systems only within the resolution of standard building blocks. VLSI components cut the cost

**Without up-front analysis, optimizing a system becomes a lost cause: In most projects, once hardware and software designs progress to a level of detail sufficient for combined simulation or prototype verification, the re-engineering required for optimization is too costly or time-consuming.**

of hardware and facilitate the development of highly complex systems, but because they are large standard-function blocks that designers cannot alter, they reduce the ability to optimize architectures.

Custom and semicustom VLSI increases the granularity of design and optimization. Custom technologies free designers to optimize architectures at a very low level, enabling them to trade off hardware modifications to simplify software or trade off software complexity to speed hardware. A few years ago, altering software to suit available hardware options was simpler and

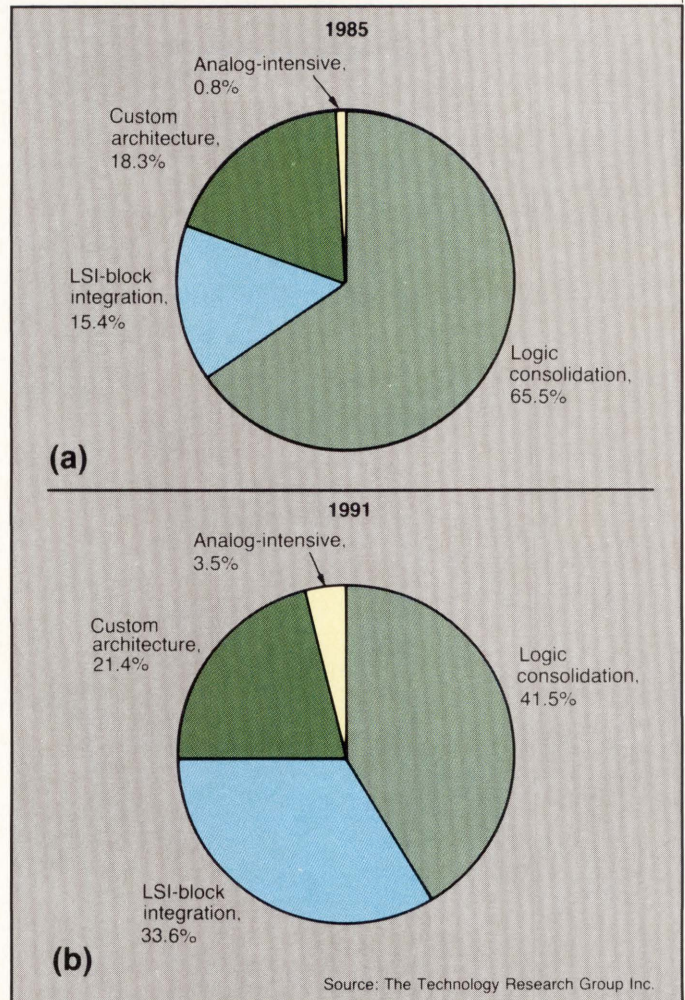


FIGURE 5. Distribution of CMOS gate array and cell-based designs, 1985 (a) and 1991 (b).

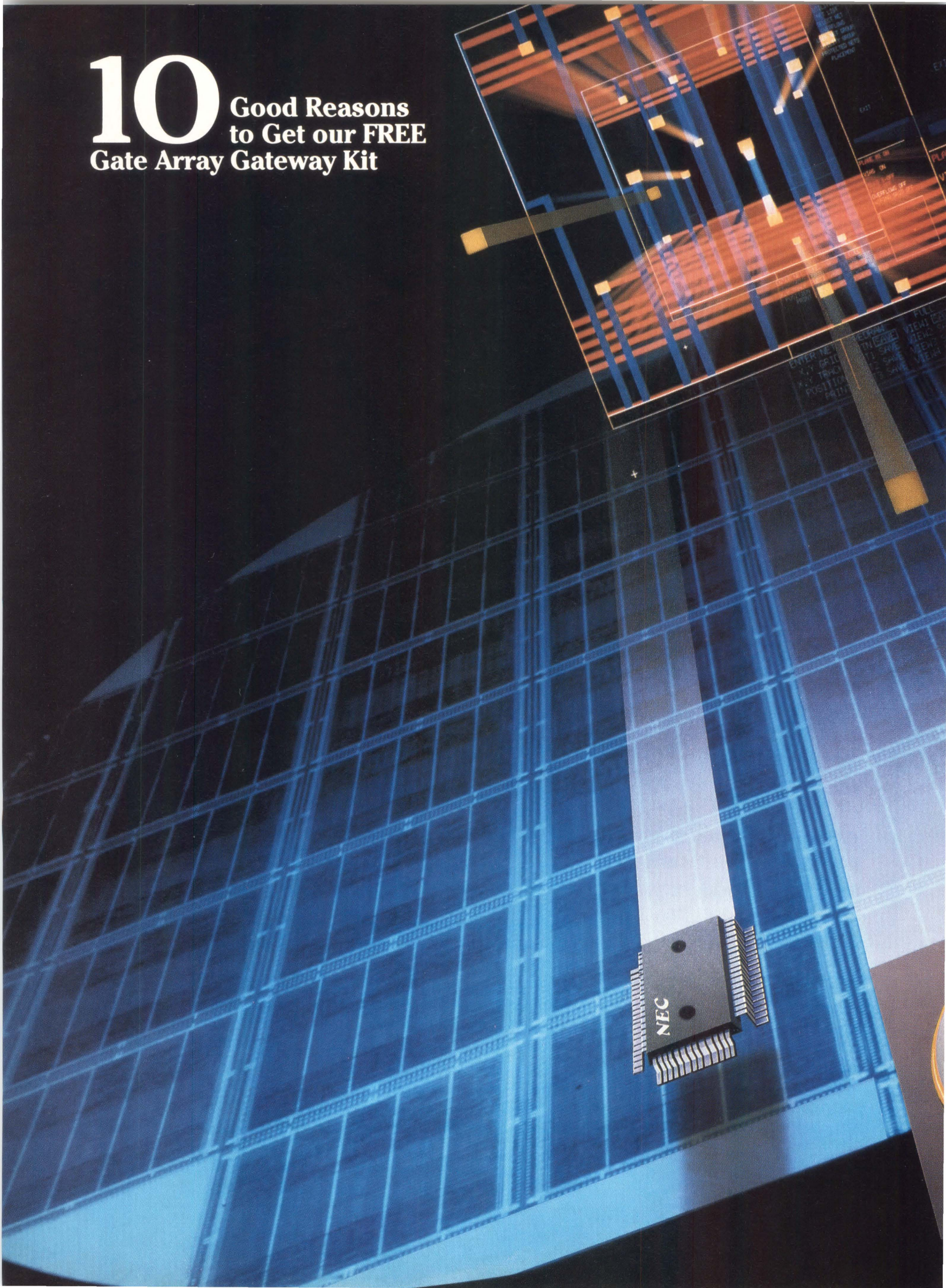
cheaper than the opposite. Now, altering hardware to suit programming considerations is often easier, less expensive, and more effective.

An efficient design process involves complex interactions between high-level hardware and software design to optimize trade-offs. Yet the present distinction between hardware and software design prevents such joint analysis and design, for both technical and organizational reasons.

System optimization is the practical, obvious, immediate reason for hardware designers to care about software development. At a higher level, the efficient production of complex electronics products requires balancing the needs of hardware and software development. Electronic-design automation tools for hardware are giving engineers the ability to experiment with circuit design. That flexibility needs to be extended to embedded-software design without creating havoc in the process of designing the system.

The first step in optimizing either a system under development or the development process is the effective partitioning of system elements between hardware

# 10 Good Reasons to Get our FREE Gate Array Gateway Kit

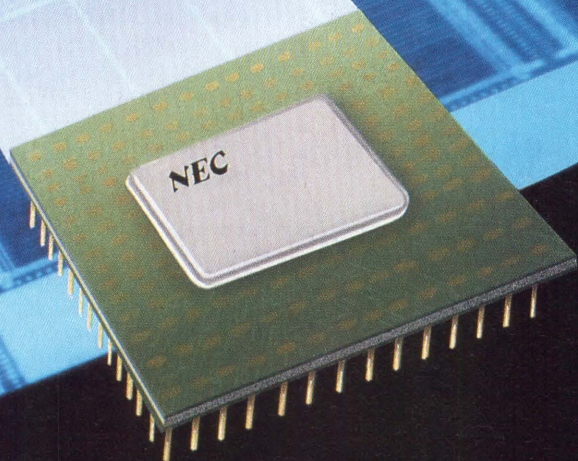


**NEC's FREE Design Kit is Your Gate Array Gateway to**

1. CMOS, BiCMOS, and ECL technologies.
2. Valid, Sun, Mentor, Daisy, LOGICAN, Tektronix/CAE, HP 9000, FutureNet, MILA, Calma TEGAS V, and HILO compatibility.
3. A powerful block library with more than 170 macros.
4. Cell utilization rates of 95%.
5. Better than 95% first-time success with engineering prototypes.
6. The most experienced manufacturing capability.
7. The widest choice of through-hole and surface-mount packages.
8. The most effective way to lower your system costs.
9. Local design support with local FAEs and design centers backed by a satellite network of computers for worldwide communications services.
10. The easiest translation of your ideas into silicon.

**NEC Electronics Inc.**

401 Ellis Street, P.O. Box 7241  
Mountain View, CA 94039



Our gate arrays are your gateway to higher system performance. Call **800-632-3531** to browse through our block library. **FREE!**

and software. Typically, partitioning of major systems is done in an ad hoc way. Without tools for representing and simulating hardware and software elements together, developing an optimized specification for hardware and software elements is impossible for systems of any great complexity. In many systems, decisions about which functions to implement in software and which to build in hardware affect system cost and performance as much as, or more than, the quality of implementation. Indeed, without up-front analysis, optimizing a system becomes a lost cause: In most projects, once hardware and software designs progress to a level of detail sufficient for combined simulation or prototype verification, the re-engineering required for optimization is too costly or time-consuming. As a result, many of the improvements suggested by performance analyses done at the implementation level are useful only as they influence thinking about the next product. By the implementation phase, it is too late to use performance and other analyses to streamline the development process.

Increasing system speeds complicate even the problem of prototype verification. In-circuit emulation techniques are proving difficult to implement for 32-bit processors running at clock rates of 25 MHz or more. Those methods are being replaced by approaches that combine simulation with observation of activity at device pins. Logic analysis is used to gather signal data from prototypes; simulation is used to investigate inter-

***Ideally, tools that furnish a common starting point for hardware and software development also should provide a framework for incremental hardware-software integration at various levels of circuit and code abstraction throughout the development process.***

nal device operation and perform many of the debugging operations traditionally done in hardware. This methodology demands integration of hardware and software development and debugging tools.

## **THE BEGINNINGS OF SOLUTIONS**

Coping with the encroachment of software on the hardware-development process requires several types of tools. The first class of tools encompasses those that furnish a common starting point for hardware and

software development, thereby allowing up-front analysis and intelligent partitioning. At the very least, such tools formalize hardware and software specifications, minimizing the potential for discovering conceptual errors late in the design process. Ideally, these tools also should provide a framework for incremental hardware-software *integration* at various levels of circuit and code abstraction throughout the development process.

Structured design and analysis tools, the flagships of major suppliers of CASE tools, have the potential of providing the framework for planning and partitioning. They help software developers analyze the goals of systems under development and the architecture of the software to implement those systems. These tools use graphics to capture system and software specifications at an abstract level and generate templates for designing code blocks. However, as implemented now, these tools do not feed software—let alone hardware—simulations. They formalize software specifications but do little to aid software development.

Similarly, some high-level hardware simulators allow behavioral definition of entire hardware systems prior to implementation, but they do not link well to either software-development or hardware-implementation processes. Ideally, structured analysis and design systems should feed compatible hardware and software simulators, making possible interactive analysis of system specifications, designs, and partitioning strategies. The greatest promise so far for the development of such tools appears to be the extension of high-level software design and simulation tools to include hooks for hardware description and programs for hardware simulation.

For the most part, structured software-design tools available are not yet up to the task. Not only do most not yet support simulation, but they incorporate methodologies and support languages not suited to the design of systems based on custom hardware. Analysis and design tools for embedded code need specification methodologies different from those of MIS and commercial tools. Structured techniques, even when enhanced for real-time development, fall short of the requirements for the complete, unambiguous, and concise representation of complex real-time systems. Systems not designed for developing real-time programs do not execute subsystems concurrently or prioritize system reaction. Many systems that do not require real-time extensions demand assembly-language programming using instruction sets that are custom or, even worse, that change during the development process. High-level design systems for these projects need languages much more flexible than those required for other types of programming. The most appropriate tools for overall system development will be those targeted specifically to concurrent hardware-software design.

The second class of tools addresses the problem of verifying hardware and software designs together at various stages of design before hardware prototypes and production code are complete. These tools include facilities for downloading code into hardware simulation environments and for linking high-level models of incom-

plete hardware and/or software elements to implementation-level models of completed system elements. Several programs exist for mixed-level modeling of hardware, but little work has been done to link these to mixed-level software models. The best done so far in commercial tools is the development of virtual microprocessor or microcode development systems tied to low-level simulators.

The problem with that approach is simulation speed. Simulating a typical 100,000-gate processor that executes one instruction every four clock cycles, a simulation accelerator performing 1 million events per second

**Several programs exist for mixed-level modeling of hardware, but little work has been done to link them to mixed-level software models. The best done so far in commercial tools is the development of virtual microprocessor or microcode development systems tied to low-level simulators.**

would evaluate 25 instructions per second. At that rate, simulating one second of operation for a 1-MIPS processor would take more than 10 hours. In systems employing standard microprocessors for which behavioral models are available, this speed can be improved, but the amount of real software simulation now practical is still limited. Consequently, high-level system simulation and effective, verifiable links from high-level design to low-level implementation are more important. In the absence of virtual development systems, designers could make good use of a microprocessor development system that can plug into hardware modelers.

The final class of tools includes those for synthesis and optimization. Ultimately, high-level descriptions will drive both hardware and software development. Tools for hardware synthesis have already been demonstrated, but they address microcode synthesis only to varying degrees. Similarly, a few compilers exist for compiling microcode from high-level descriptions, but they do not address hardware synthesis at all. So simultaneous—or even automated iterative—hardware-software optimization is not likely to be possible any time soon.

The absence of synthesis tools creates an acute need for programming aids for hardware developers. Hardware designers developing microcode have different re-

quirements from mainstream programmers. Ideally, they should be able to generate microcode and a high-level description of desired instruction execution automatically from a hardware design. Short of that, microcoding aids should help optimize performance, storage, and resources. Even with microcode synthesis tools, designers would still need code analysis tools that assume that hardware architectures are changeable and thus can be altered to optimize microcode, just as code is typically altered to optimize hardware.

Suppliers dedicated to automating portions of the embedded-software development task are just beginning to emerge. The technology and the market now are comparable to those for hardware-design automation tools several years ago, when hardware designers used a hodgepodge of instruments, programs, and methods to assess and correct their designs at discrete points in the process. Smart hardware designers will form the vanguard of the move to automated, integrated code development. □

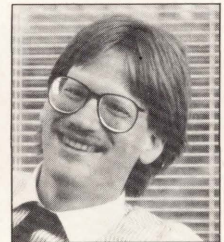
#### REFERENCES

- Rappaport, A.S. 1987. "The Pending Fragmentation of the Semicustom IC Market," *VLSI Systems Design's Semicustom Design Guide*.
- The Technology Research Group Inc. 1987. *Software Development Automation*, Boston, MA.

#### ABOUT THE AUTHORS

**Cindy Thames** is the vice president and cofounder of the Technology Research Group. She is responsible for the company's publications, which include reports on studies of users and potential users of advanced technologies for developing electronic systems and components, as well as periodic reports on the size and projected growth of markets for such technologies. Cindy was previously a senior editor at *Electronic Business* magazine, specializing in product development issues.

**Andrew S. Rappaport** is the president of the Technology Research Group, where he oversees consulting and market research into new electronic technologies. Since co-founding the organization in 1984, he has consulted extensively on business development and strategic planning issues for companies commercializing such new technologies as hardware and software design automation, custom and semicustom ICs, and specialized test equipment and instrumentation. He has six years' electrical engineering and management experience and spent two years as a senior editor at *EDN* magazine. Andy studied electrical engineering and political science at Princeton University and holds one patent.



# 53.6 Reasons to Choose P-CAD for CAE and PCB design.



• End-to-end PCB design • Workstation performance • 53.6% market share\* • New! SMT support



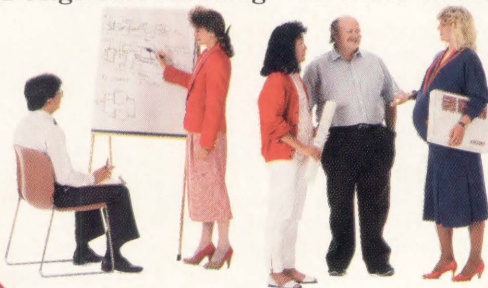
• Low cost schematic design • Auto place & route (45°) • Large board capacity • ASIC design kits



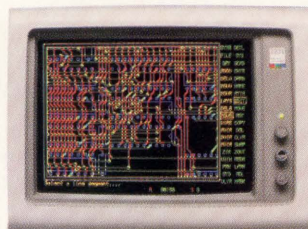
• Operates on standard hardware • Full range of system interfaces • 3000+ component library



• Design rule checking • Absolute data security • 3rd party software & services • 24-hour on-line support



• Local sales & training



To find out why 53.6% of engineers using PC-based CAD systems choose P-CAD® for workstation level performance, call toll-free:

**800-523-5207 U.S.**  
**800-628-8748 California**

**p-cad**  
PERSONAL CAD SYSTEMS INC.

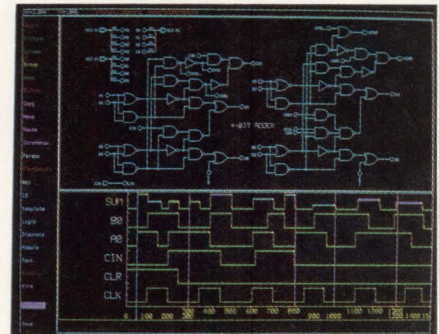
Personal CAD Systems, Inc.  
1290 Parkmoor Avenue  
San Jose, California 95126 USA  
Telex: 371-7199 FAX: 408-279-3752

\*Source: Dataquest, Inc.  
P-CAD is a registered trademark of Personal CAD Systems, Inc.  
Generation 2.0 is a trademark of Personal CAD Systems, Inc.

CIRCLE NUMBER 9

New! State-of-the-art  
Generation 2.0™

## II LOGIC DESIGN



### 30 BENCHMARKING SCHEMATIC ENTRY SYSTEMS

Paula K. Filseth

This article offers an unusual approach to evaluating different schematic capture systems: Besides presenting the functional characteristics of four major schematic capture systems, it benchmarks the systems' efficiency in performing certain operations in terms of keystroke counts, plus their speed in entering an LSI circuit.

### 40 A FIRST COURSE IN VHDL

David L. Barton, Intermetrics Inc.

The VHSIC Hardware Description Language is rich in constructs for various levels of hardware description. This introduction to the language focuses on register-transfer descriptions of behavior and structural descriptions of circuits.

### 50 SOFTWARE VS. HARDWARE MODELING FOR SYSTEM SIMULATION

Pete Johnson, Gateway Design Automation Corp.

Behavioral and physical modeling each can partially solve the modeling problem posed by system-level simulation. This article discusses the advantages and disadvantages of both approaches and considers when each is appropriate.

# BENCHMARKING SCHEMATIC ENTRY SYSTEMS

Paula K. Filseth, Fremont, CA

A schematic editor is the computer program used by circuit design engineers to "build" and modify circuit diagrams on the terminal screen. Although the details of an editor's makeup vary with the program, all schematic editors execute three basic tasks. First, they provide the circuit designer with symbols and commands to be used for entering, creating, and modifying circuit diagrams. Second, they save the finished diagram by copying it from memory onto a disk so that it can be retrieved later. Finally, the editors "netlist" the finished design so the simulator can exercise the circuit.

For the purpose of this study, four popular editors were selected for comparison. Three were general-purpose editors: NETED/SYMED, from Mentor Graphics Corp.; ACE, from Daisy Systems Corp.; and VALIDGED, from Valid Logic Systems Inc. The fourth was a vendor-specific schematic entry system: LSED, from LSI Logic Corp.

The study was conducted by interviewing design engineers; by gathering information from specification sheets, user's manuals, and tutorial publications; and by actually using the editors.

## CRITERIA

We judge the most important criterion of an editor to be its ease of use. Several factors determine how easy an editor is to use and consequently how quickly designs can be entered and modified. These factors include method of command entry, techniques for general editing, group operations, and movement within a design. Special features also are considered part of the ease-of-use criterion.

Other important criteria are performance, predictability, generality, and hardware platforms. The aesthetic quality of schematics produced by the different editors, though somewhat subjective, should also be considered. Table 1 summarizes (somewhat subjectively) the features of the four editors.

## EASE OF USE

For command entry, all of the editors use a variety of command styles, including text commands, menu selec-

tion, and single-key commands, but the importance of each style varies from editor to editor. LSED primarily uses single-key commands, with the most common commands assigned to mouse keys. NETED/SYMED, VALIDGED, and ACE all use menus to select most operations. NETED/SYMED uses hierarchical menus, whereas VALIDGED and ACE have fixed menus that list only the most frequently used commands. For less commonly used commands, VALIDGED requires text commands; ACE provides pop-up menus and forms.

## General Editing

Circuit schematics consist primarily of component symbols, lines that represent wires, and text. The main commands are for adding, moving, copying, and deleting these objects.

*Adding symbols.* The benchmarked editors all let the user add a pre-existing symbol by typing a text command in which the name of the symbol is specified. They all also allow the user to copy a symbol immediately after adding it without having to select a copy option, thus reducing the time required for copying subcomponents.

LSED, VALIDGED, and ACE permit the engineer to "draw" new symbols freehand. NETED/SYMED also lets the user draw new symbols; however, he must exit the network editor (NETED) and start up the symbol editor (SYMED) before he can do so—an awkward arrangement that costs the designer valuable time.

*Adding wires.* In some cases, the schematic editor must be given some information about what path the wire should take. This information can range from the location of one or more intermediate points to a complete tracing of the path.

*Adding lines to symbols.* Each editor allows diagonal lines in symbols. They also allow circular arcs. VALIDGED requires that the center of the circle on which the arc lies be specified. This requirement slows down arc drawing considerably, since the point must be found by trial and error. The other editors let the user specify both end points and an intermediate point anywhere on the arc.

*Adding text.* All the editors use text to specify the names of parts and wires, to specify their properties, and for notes to clarify the schematic for readers.



*Moving objects.* All of the editors automatically move wires attached to moved objects. Wires attached to components in VALIDGED, LSED, and ACE not only will move with the component, but also will retain their orthogonal structure. The resulting wire looks fine if the component has been moved only a short distance; a longer move, however, may result in the new wire crossing over intervening components. Although NETED does reconnect wires, it usually replaces horizontal and vertical wires with diagonal ones.

*Copying objects.* The user may copy any object on the schematic by specifying which object is to be copied and where the copy is to be placed. All four editors allow an object to be copied several times in a row without requiring that the copy operation be respecified each time.

*Deleting objects.* All four editors let the user perform "group operations"; that is, the user may specify a group of objects on a schematic and move, copy, or delete them all at once. LSED and VALIDGED permit the user to select groups by drawing arbitrary polygons around the objects in the group. These arbitrary polygons permit a great deal of flexibility. NETED and ACE, on the other hand, only let the user delete rectangular areas, designated by the specification of two opposite corners. When an object is deleted on the LSED screen, any wires attached to it also are automatically deleted; VALIDGED and ACE will not delete such wires. NETED permits the user to specify whether or not these wires are to be deleted. Finally, VALIDGED assigns the group a name; for future operations on the group, the user may select the group by name instead of having to redraw the polygon.

### Moving Around in a Design

When editing a schematic, the user must frequently bring a different portion of the design onto the screen. Five types of operations enable him to do so: diving into a part, popping out of a part, zooming in, zooming out, and panning (which centers the screen round a different part in the schematic). In all the surveyed editors, the part to edit or area to examine may be specified either by description or by placing the cursor on an instance of the desired part.

### Special Features

Additional features include automatic wire routing, a recovery (or "undo") mechanism, buses, and windows.

*Autorouting.* LSED needs the fewest keystrokes for wiring, because it inserts as many bends as necessary between the source and destination of a route. LSED also allows the engineer to route wires manually; that is, the engineer may force the wire to take a certain path by specifying points that it must pass through.

VALIDGED and ACE automatically provide L-shaped routes (routes with one bend, or "jog"). Although these programs require the engineer to put in intermediate points, they will route to those points, putting one jog in the wire if necessary. Furthermore, the engineer may flip, or "toggle," the wire if he wants it to bend in the opposite direction.

	ACE (Daisy)	LSED (LSI Logic)	NETED/ SYMED (Mentor)	VALIDGED (Valid)
Primary method of command entry	Menu/form	Single-key	Menu	Menu
Ease of making symbols	Good	Good	Poor	Good
Connectivity maintenance after moves	Fair	Good	Poor	Fair
Group shape	Box	Polygon	Box	Polygon
Autorouting	No	Yes	No	No
Undo	None	Slow	Fast	Fast
Buses				
Wires	Yes	Yes	Yes	Yes
Parts	No	Yes	Yes	Yes
Structure	Yes	Yes	No	No
Operations between windows	Copying, wiring, moving	Copying, wiring, moving	Copying	Clumsy copying

TABLE 1. Features of the four editors surveyed.

Wiring with NETED/SYMED is slow because the engineer must trace precisely the path he wants the wire to follow.

*Undo.* The undo operation is a recovery mechanism. NETED will allow the user to undo the most recent modification of the circuit; SYMED, however, will not. VALIDGED and LSED both allow an arbitrary number of undo operations; however, VALIDGED's UNDO command is very fast, whereas LSED's is extremely slow. ACE does not offer an undo operation.

*Buses.* All the editors allow the user to attach a piece of text to a wire to indicate that the wire on the schematic represents several wires in the circuit. All except ACE permit the same thing to be done to a subcomponent. ACE-generated schematics tend to be cluttered because replicated parts must be drawn out in full.

LSED and ACE permit structured buses, which are buses composed of subbuses. This feature enables the user to treat several related buses and wires as a single bus. Each subbus of a structured bus has a name that can be used to make connections to that subbus.

*Windows.* All the editors let the user specify windows, select a portion of a circuit in one window, and copy it into another. With VALIDGED, this operation is somewhat clumsy, involving writing the selected portion of the circuit out to a disk file and reading it back into the other circuit. LSED and ACE allow connections to be made from an object in one window to an object in another.

# 'REAL-TIME' SOLUTION TO ASIC VERIFICATION

**Tests Full Speed  
At Up to 50 MHz  
Across Entire Cycle.**

**F**or the first time, you can test your VLSI prototype design at real world operating speeds. Thoroughly and easily. Across the entire cycle. Without compromise.

Topaz is a totally-integrated ASIC verification system that reduces prototype characterization and fault analysis time, while offering these exclusive advantages:

- **Full Data Formatting to 50 MHz**—for quick measurement of set-up times and propagation delays.
- **256 I/O Channels at Speed, Without Multiplexing**—for maximum performance and flexibility.
- **Programmable Pattern Generation to 50 MHz**—for initiation of loops, branching and data control.

ASIC design requires painstaking accuracy. Verifying that design has been neither fast nor easy. The time available to get today's increasingly-complex ASICs to market continues to contract, and the price of an undetected error can be incredibly costly.

With Topaz, you'll know your design is right, and you'll know it *faster*. CAE-LINK™ software permits easy translation of simulator vectors into ready-to-use test vectors. And, our exclusive Meta-Shmoo™ software allows you to quickly sweep voltages and times *at 500ps increments across an entire cycle*, without programming.

It acquires data with a minimum of effort; and its ability to do graphic error-bit mapping and multi-level triggering gives it unequalled performance in failure analysis.

Topaz is a cost-effective solution to today's high speed ASIC verification needs, and the even higher speeds you'll require tomorrow. Call for complete details or your personal demonstration.

**HILEVEL**  
TECHNOLOGY, INC.

18902 Bardeen, Irvine, CA 92715

Phone: (714) 752-5215

**DIAL TOLL FREE 1-800-HILEVEL  
(In California 1-800-752-5215)**

CIRCLE NUMBER 5 SEND LITERATURE  
CIRCLE NUMBER 6 NEED DEMO



	ACE	LSED	NETED/ SYMED	VALIDGED
<b>General editing</b>				
Adding parts	10	9	12	8
Wiring	12	4	12	12
Moving	6	4	8	6
Copying	6	5	8	6
Deleting	4	3	7	4
Text commands				
Naming	18/14	14	9/21*	12/8†
Properties	27	14	21	12
Notes	10	9	21	12
<b>Group operations</b>				
Moving	14	11	18	22/12‡
Copying	14	11	18	22/12‡
Deleting	12	10	17	19
<b>Moving around in a design</b>				
Diving in	6	3	8	4
Popping out	6	2	8	2
Zooming in	6	3	6	2
Zooming out	6	2	6	2
Panning	4	3	2	3
<b>Total</b>	<b>159</b>	<b>107</b>	<b>187</b>	<b>136</b>
<b>Ratio</b>	<b>1.49</b>	<b>1.00</b>	<b>1.75</b>	<b>1.27</b>
* On Mentor, attaching names to subparts is much simpler than attaching names to other objects.				
† On Valid, naming several objects in a row takes fewer keystrokes per object than naming just one.				
‡ On Valid, group moves and copies are simpler for a second operation on the same group.				

TABLE 2. Keystroke counts for common editing operations.

## EASE-OF-USE BENCHMARK

In an effort to quantify subjective properties, a keystroke criterion was employed in our benchmark. Clearly, the number of keystrokes or mouse commands required to perform a function is indicative of the time required to learn and use a schematic entry system.

All editing operations consisted of four types of movement or keystrokes: cursor positioning, typing, single-key commands, and depression of mouse buttons.

A short experiment indicated that moving the mouse and pressing its buttons took about the same amount of time in all four systems. Pressing keyboard keys took about twice as long as pressing mouse keys, because it

took time to find the desired key. Typing a piece of text averaged about six times as long as pressing a mouse key. Since each system uses a different set of movements, the number of "equivalent mouse keystrokes" required for each common operation was computed, for comparison purposes, by analyzing the procedures that the manuals specified for performing that operation. In some cases, proficient users reported faster procedures for performing certain operations, so those procedures were used instead. Table 2 summarizes the number of equivalent keystrokes for the four systems: LSED was a clear winner, followed by VALIDGED, ACE, and NETED/SYMED, in that order.

## PERFORMANCE BENCHMARKS

The time required to enter a design on a particular schematic entry system can be used to measure ease of use and performance (execution time of various operations). Variables include the expertise of the operator and the type of host.

The circuit in Figure 1, a simple 600-gate synchronous design, was selected. It uses a 16-bit datapath and a random-logic finite-state machine. The entries were performed using release 5.02.02 of ACE running on a Logician-V, release 8 of VALIDGED on an S-32, release 6.2 of LSED on a Sun-3/75, and release 5.1 of NETED/SYMED on an Apollo DN420.

To obtain the performance benchmarks shown in Table 3, an engineer proficient in the use of each system entered the design. LSED again won the race, followed by VALIDGED and NETED/SYMED. ACE came in last, taking over four times longer than LSED. Interestingly, ACE required substantially more time to enter the benchmark design than did NETED/SYMED, even though ACE's overall keystroke count was lower. ACE's slow response to commands contributed heavily to its poor overall time in entering the benchmark design.

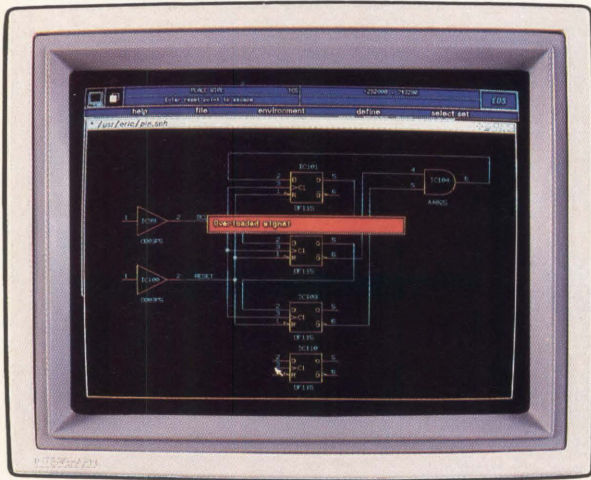
## PREDICTABILITY

All four schematic editors gave users unpleasant surprises at times.

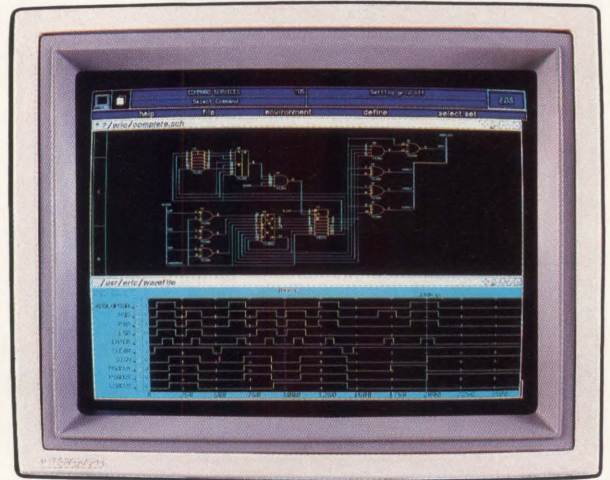
ACE was described by users as a fairly unpredictable program. The most serious problem is that, under certain circumstances, the wires in a bus are connected to the wires in a bus pin of a submodule in a highly counterintuitive order. Further, group operations yield particularly unpleasant surprises and take much longer than might be expected. For instance, copying a 14-part group took ACE 60 seconds, as opposed to an average of only 7 seconds for the other editors. But even worse, at one point a copy never materialized at all, and the copy command had to be re-entered. In another case, it appeared with diagonal wires, one of which proved impossible to delete.

The only predictability problem VALIDGED users reported was that when text is entered, it is occasionally attached to the wrong object. VALIDGED automatically

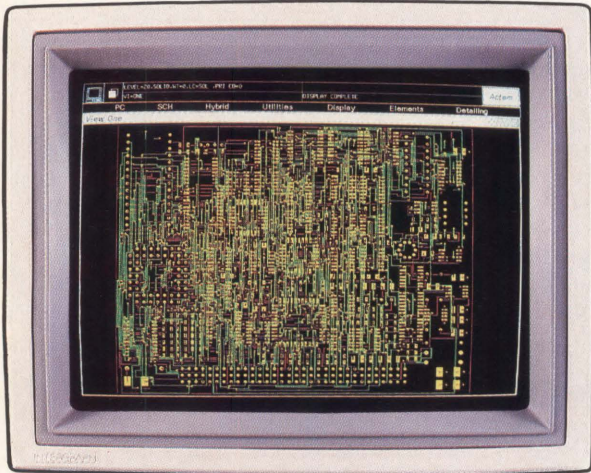
# When you need more



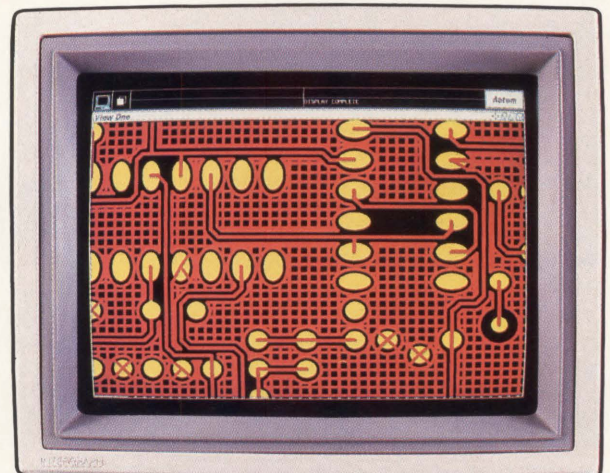
*Create schematics faster with correct-by-construction design and on-line Electrical Rules Checking*



*Logic analyzer type display for interactive digital analysis using HILO-3<sup>®</sup>*



*Begin PCB design with a feasibility check, then rely on a full suite of tools for autoplacement, high-completion autorouting, and design rule checking*

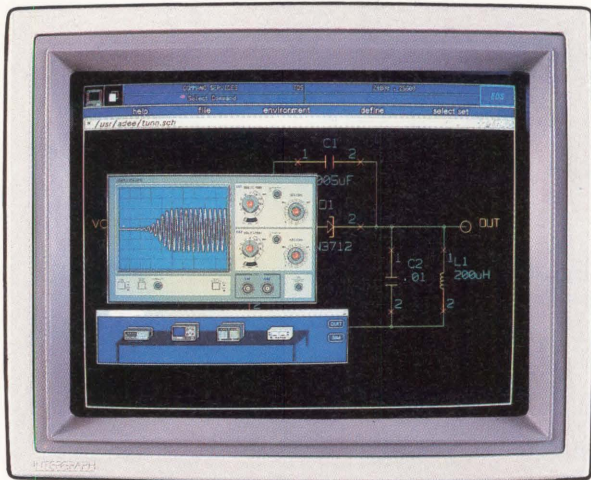


*Power and ground planes are generated automatically*

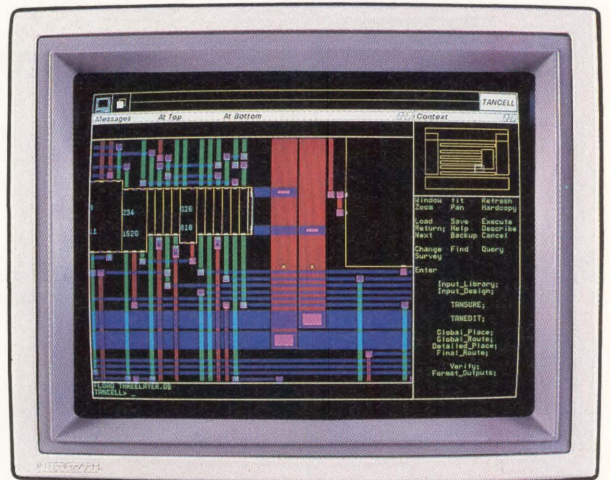
# Think Intergraph.

For more information or a demonstration, call your nearest Intergraph representative. Call toll-free 1-800-826-3515 or (205) 772-2700. Call (31) 2503-66333 in Europe.

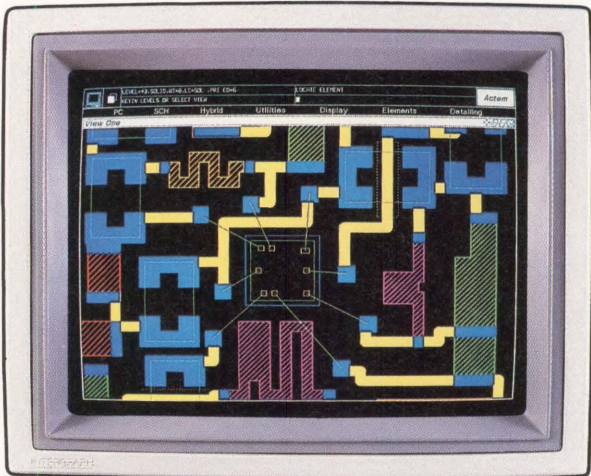
# than a niche solution



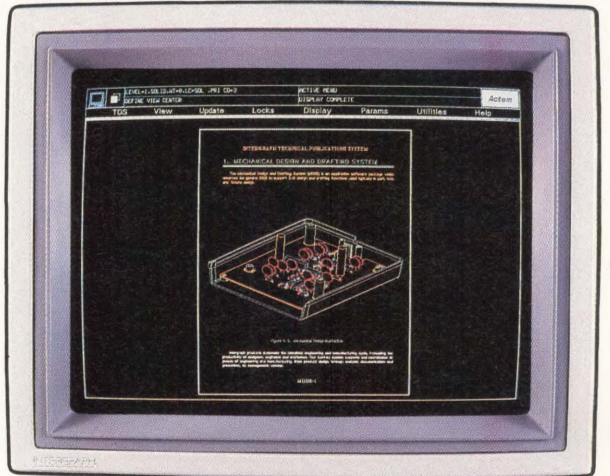
*Easy-to-use virtual instruments provide a familiar interface for analog analysis using Intergraph's Cspice or ACS*



*Reduce ASIC die size and improve yield with Tangent Systems' timing-driven layout, automatic design for testability, and test pattern generation*



*A dedicated package for thick and thin film hybrid design features automatic resistor generation and automatic routing of chip and wire parts and SMDs*



*64-bit mechanical design packages and electronic publishing software round out a system for total product development*

## INTERGRAPH

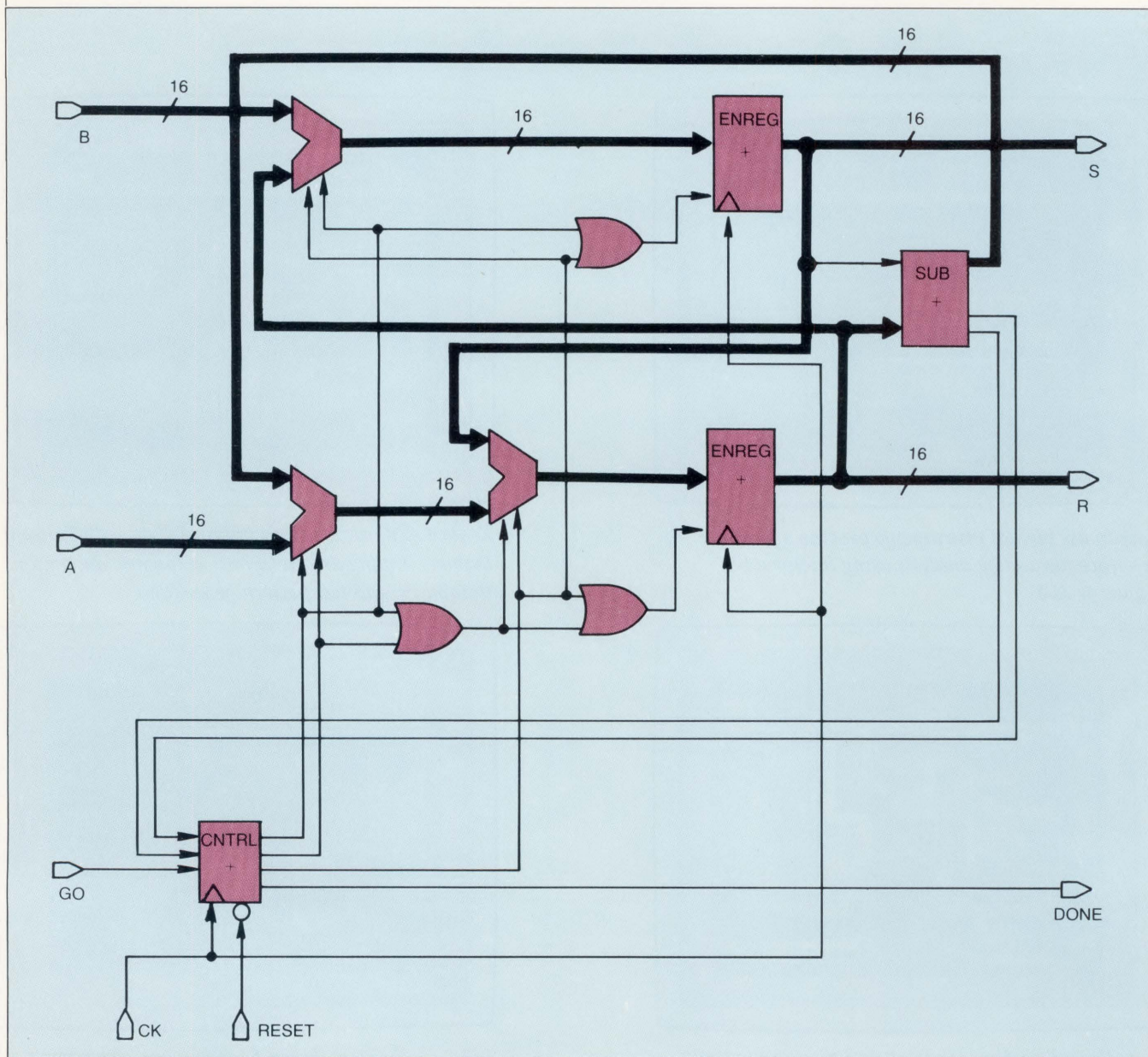


FIGURE 1. The benchmark design (entered using LSED).

attaches text to the closest object rather than requiring the user to specify which object it is related to; in some cases, that object is not what the user intends. Commands to show text connections and to reattach text to different objects do alleviate this problem, however.

The primary predictability problem with NETED/SYMED is that SYMED does not work like NETED. For example, SYMED cannot copy part of a symbol into another symbol, it cannot rotate or flip objects, it cannot stretch lines, and it cannot undo a deletion. These differences frequently take new SYMED users by surprise.

Another SYMED problem is the existence of two distinct grids, one finer than the other. Objects may be placed

anywhere on the fine grid but can be moved only in increments equal to the coarse-grid spacing.

The most serious complaint about NETED was that when wire segments and pins are placed on top of other wires, they may be shorted together. Shorting is particularly common when a part with wires connected to it is rotated. The original connectivity of the circuit is changed and must be corrected by hand. Users also reported difficulty in creating small jogs in wires. The wiring command tries to eliminate the bends completely or to make them diagonal. Finally, the rule-checking command gives many spurious warning messages, making it difficult to find the real problems in the circuit.

	ACE	LSED	NETED/ SYMED	VALIDGED
Minutes	259:45	63:05	137:55	103:55
Ratio	4.12	1.00	2.19	1.65

TABLE 3. Time required to enter the benchmark design.

	ACE	LSED	NETED/ SYMED	VALIDGED
Ease of use	Poor	Good	Poor	Fair
Performance	Fourth	First	Third	Second
Predictability	Poor	Poor	Poor	Good
Machines	Logician, IBM PC AT	Sun	Apollo	Valid- Station, IBM PC AT, MicroVAX
Aesthetics	Poor	Good	Poor	Good

TABLE 4. General conclusions.

The worst surprise in store for LSED users is that the program crashes on rare occasions. LSED has the ability to undo a crash, so that crashes are not a disaster, but they are a major irritation, since the undo operation can take as long as 15 minutes to run. A much more common complaint was that since the DIVE INTO A PART command is the same key as the MODIFY TEXT command, the two are confused if the cursor is too close to a piece of text when the user tries to dive into a part. If the user is looking at the keyboard instead of the screen, he may not notice what is happening until he has typed several commands onto the end of the piece of text. The SELECT and CONNECT commands also are easy to confuse.

LSED users also complained that text is not clipped at the screen borders. Instead, if a piece of text does not entirely fit on the screen, it is not displayed at all.

### GENERALITY, HARDWARE PLATFORMS, AESTHETICS

In addition to ease of use, the generality of a particular editor must be considered. General-purpose CAE workstations support a variety of ASIC vendors, as well as standard families like the 7400 logic and 2900 bit-slice families. Vendor-specific programs, such as provided by LSI Logic, support only their (and their licensees') ASIC libraries. Clearly, all else being equal, a general-purpose solution is preferable.

Currently, NETED/SYMED is hosted only on Apollo, and LSED only on Sun workstations. ACE is available on Daisy Systems' own hardware, the Logician, and on a PC. VALIDGED is available on four machines: the MicroVAX; the PC AT; the Sun; and Valid's own host machine, the

ValidStation.

The speed or ease of use of each editor was considered most important; however, since a speedy editor that turns out poorly executed or unreadable diagrams is no asset to its user, consideration was also given to the quality of the finished design. A panel of logic designers reviewed the logic diagrams that each schematic editor produced for the 600-gate benchmark.

ACE's designs were considered aesthetically poor. ACE does not support bused parts. For instance, to invert a 16-bit bus, 16 inverter symbols are required; the other editors will allow the engineer to use a single inverter symbol and specify that it stands for 16 inverters. This lack tends to make schematics generated on ACE look cluttered and sometimes forces the designer to add extra hierarchical levels to a design.

NETED/SYMED's designs also were rated as poor. Groups of parts are replicated by placing them in a rectangle called a "frame." Parts that have been replicated in this manner cannot be wired to parts outside the frame; instead, wires inside and outside the frame are connected by being given the same name. It can therefore be difficult to tell whether parts are connected.

Both VALIDGED and LSED received high ratings for aesthetic quality.

### SUMMARY

Of the four systems analyzed, LSI Logic's LSED emerged as a clear winner—not surprising, as it is three or four years newer than NETED/SYMED and VALIDGED and includes many of the best features of its competition. Unfortunately, it is available only with LSI Logic libraries.

VALIDGED came in second, NETED/SYMED third, and ACE a distant fourth. All three have libraries available from most of the ASIC vendors, including LSI Logic.

Finally, some cautionary notes. First, software is in a constant flux, and the results shown here are probably already out of date. Second, there are dozens of schematic entry systems; this article addresses only four. And last, schematic entry is only a portion of a total CAE system. In the end, a designer will select an integrated package that will solve his problem, sometimes at the expense of being "sole-sourced" and tied to one ASIC vendor's silicon products. □

### ABOUT THE AUTHOR

**Paula Filseth** is currently a technical writer at Gould Inc.'s Imaging and Graphics Division in Fremont, CA. At 18, Ms. Filseth completed the requirements for a bachelor's degree in communications from Stanford University. Now, two years later, she has almost managed to figure out the repayment schedule on her various student loans. A fiction lover beginning to despair of ever finding the time to write the Great American Novel, Ms. Filseth has vowed at least to produce the Great American Technical Manual.





# Only Mentor Graphics has brought a billion gates to light.

more products than any other electronic design automation vendor.

A claim only Mentor Graphics can make.

Along the way, we've pioneered schematic capture and simulation tools that are now industry standard. Like hierarchical design entry, which allows efficient management of even the largest designs. And MSPICE,<sup>™</sup> which brings real interactivity and a graphics-oriented interface to analog simulation.

In just 5 years, over a billion gates have flowed through our IDEA Series<sup>™</sup> design automation systems. And that's a very conservative estimate.

Which makes it seem all the more incredible that, before we came along, almost all electronic circuits were drafted and breadboarded by hand.

Since then, our schematic capture and simulation tools have produced more circuits for

At the same time, we've provided the depth and power required to work with very large designs. A macro language allows you to build a highly customized interface, one suited specifi-



cally to your particular productivity needs. And "case frames" allow very complex circuit patterns to be expressed in just a few keystrokes.

For simulation, our QUICKSIM™ family brings you logic, timing, and fault simulation in a single, integrated package. Plus the ability to use a mixture of modeling techniques, including chip-based modeling with our Hardware Modeling Library.™ And you can call upon our Compute Engine™ general-purpose accelerator to enhance standard workstation performance.

Once your design is complete and verified, our IDEA Series lets you express it in any standard physical form: PCB, full-custom or semicustom. We have a full set of layout tools for each. All fully compatible with our front-end tools.

As we head toward our next billion gates, we'd

like to make some of them yours. It's all part of a vision unique to Mentor Graphics, the leader in electronic design automation. Let us show you where this vision can take you.

Call us toll free for an overview brochure and the number of your nearest sales office.

**Phone 1-800-547-7390**  
**(in Oregon call 284-7357).**

Sydney, Australia; Phone 02-959-5488 Mississauga, Ontario; Phone 416-279-9060 Nepean, Ontario; Phone 613-828-7527 Paris, France; Phone 0145-60-5151 Munich, West Germany; Phone 089-57096-0 Wiesbaden, West Germany; Phone 06121-371021 Hong Kong; Phone 0566-5113 Givatayim 53583, Israel; Phone 03-777-719 Milan, Italy; Phone 02-824-4161 Asia-Pacific Headquarters, Tokyo, Japan; Phone 03-505-4800 Tokyo, Japan; Phone 03-589-2820 Osaka, Japan; Phone 06-308-3731 Seoul, Korea; Phone 02-548-6333 Spanga, Sweden; Phone 08-750-5540 Zurich, Switzerland; Phone 01-302-64-00 Taipei, Taiwan; Phone 02-7762032 or 02-7762033 Halfweg, Netherlands; Phone 02907-7115 Singapore; Phone 0779-1111 Bracknell, England; Phone 0344-482828 Livingston, Scotland; Phone 0506-41222 Middle East, Far East, Asia, South America; Phone 503-626-7000

**Mentor  
Graphics®**

# A FIRST COURSE IN VHDL

David L. Barton, Intermetrics Inc., Bethesda, MD

The VHSIC Hardware Description Language (VHDL) was designed at the request of the VHSIC Program Office to provide a notation capable of the design and description of very complex components and systems. The final language, a result of efforts under both an original government contract and a later standardization drive by the IEEE, is a rich collection of constructs for various levels of hardware description.

This article is an introduction to VHDL. It considers two specific problem domains within the overall field of hardware design and description. The first is register-transfer descriptions of behavior. The second is structural descriptions of circuits, which we introduce with a section on entities and architectures, the mechanisms in VHDL for decomposing large descriptions. A third important problem domain is behavioral descriptions and the mechanisms for describing relationships between behaviors. This topic, along with other advanced language features, will be covered in a future article.

Two subjects recur throughout the article in different forms. Rather than try to explain them completely in isolation, their importance will become apparent as their effects on each problem domain emerge. The first is the basic simulation cycle of VHDL. It is this cycle that controls the resolution of signal values and the execution of the components of a hardware description. The second is the idea of a "view" of the hardware description. A view of the description is a way of decomposing the hardware description into parts, which may in turn be further decomposed into parts. Several different views are inherent in the definition of VHDL, and they will be described as their roles in hardware design and description become clear to the designer.

All of the examples and the information in this article reflect the 1076/B version of VHDL, as published in the May 1987 Language Reference Manual. This version is the subject of the standardization ballot in the IEEE. As a normal part of the standardization process, changes are being made in response to comments.

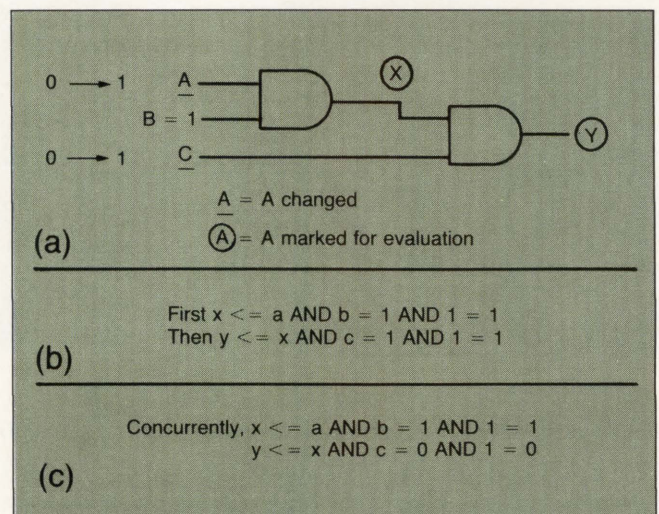


FIGURE 1. Situation depicted in (a) would be evaluated sequentially as in (b) or concurrently, as in VHDL, as given in (c).

## REGISTER-TRANSFER DESCRIPTIONS

A register-transfer description of hardware consists of a series of Boolean logic expressions. Each operator represents a gate or a series of gates in a hardware realization. Such expressions are a convenient mechanism for portraying the behavior of a hardware component. VHDL supports simple register-transfer statements, as well as timed, conditional, and guarded-assignment statements.

The objects in the expressions are called signals, and the expressions themselves are called signal assignment statements. Names are given to the signals in the signal declarations. An example describing a full adder is:

```
signal x, y, cin, cout, sum: BIT;
```

```
sum <= x xor y xor cin;  
cout <= (x and y) or (x and cin) or (y and cin);
```

The first statement identifies the objects that will appear in the expressions and the following two statements define the values of the outputs in terms of the inputs.

Such expressions also may operate upon bit vectors. The following example implements a byte adder using bit vectors:

```
signal x, y, cin, cout, sum: BIT_VECTOR (0 to 7);  
signal byte_cin, byte_cout: BIT;
```

```
cin (0) <= byte_cin;  
sum <= x xor y xor cin;  
cout <= (x and y) or (x and cin) or (y and cin);  
cin (1 to 7) <= cout (0 to 6);  
byte_cout <= cout (7);
```

The carry in and carry out bits for the entire byte adder are represented by the signals "byte\_cin" and "byte\_cout." The first line sets the first element of the carry in array to the carry in of the entire byte adder. The second two statements are just as in the previous example, except that they operate upon entire arrays rather than bits. The fourth line propagates the carry out values to the proper carry in values. The last line sets the carry out bit of the entire byte adder.

Each register-transfer expression is evaluated whenever one of the elements of the expression changes value. The definition of the simulation cycle implies that this evaluation proceeds in three steps: First, the expressions are marked for evaluation; then the expressions themselves are evaluated; and after all expressions are evaluated, the values are reflected in the actual objects. Thus, if two expressions are evaluated and the target of one appears in the expression of the other, both expressions are evaluated before either of the targets are updated. The order of evaluation has no effect on the final values of the expressions (see Figure 1).

Signal assignment statements that share signals may be said to be connected. In particular, when a signal assignment statement's target signal name (that is, the name on the left side of the "<=" sign) appears in the expression part (the right side of the assignment) of another signal assignment statement, the first may be said to "trigger" or "kick" the second. Data flow from signal names occurring on the left-hand side of signal assignment statements to signal names occurring in expressions on the right-hand side of signal assignment statements. Thus, in the example given above, the carry information may be seen flowing through the first statement, then from left to right (low to high elements of the bit array) through the third and fourth statements, and finally to "byte\_out" in the fifth statement.

The expressions above are very similar to register transfer expressions in other notations, although unlike many notations, VHDL requires a separate signal declaration statement. A separate declaration is an important factor in eliminating elusive spelling errors during the development process. The additional effort spent in declaring objects used in the expressions is well repaid in reduction in debugging and correction efforts.

## Timed Signal Assignment Statements

Real hardware does not evaluate expressions instantaneously. The designer should be able to easily and conveniently express the amount of time an expression will take to be evaluated and have those times reflected in the description.

Each expression in a series of signal assignment statements may be followed by the key word *after*, followed by a time. Adding timing information to the examples above could yield the following expressions:

```
signal x, y, cin, cout, sum, strobe: BIT;  
  
sum <= x xor y xor cin after 10ns;  
cout <= (x and y) or (x and cin) or (y and cin)  
after 15ns;
```

The timing information above states that the sum will be available 10 ns after any of the inputs change and the carry out 15 ns after any of the inputs change.

Given time delays, the order of actions in the execution of a signal assignment statement at a given time is as follows: First, the values of all declared objects for that time are determined; second, signal assignment statements that contain objects whose values have changed are marked for execution; and third, the expressions are evaluated (in any order).

If no timing clause appears in the signal assignment statement, then a time of 0 ns is assumed. However, even if 0 ns is in the timing clause, all the statements marked for execution are executed before the values of any objects are updated. The notion here is that some infinitesimal amount of time has passed, even if the actual simulation time has not advanced. Such a simulation cycle that does not cause the simulation time to be advanced is called a delta cycle.

This model of execution allows the designer to write signal assignment statements without regard to the order of their execution. The simplicity of this situation removes a great burden from the user of these statements. No concern need be taken for the exact order of the statements, but only for the values they produce.

## Conditional and Selected Signal Assignment Statements

Hardware behavior is not always a direct Boolean function of several variables. Although conditional situations may always be expressed as complex Boolean

```
signal d0, d1, d2, d3, d4, d5, d6, d7, a, b, c, w: BIT;
```

```
with (a & b & c) select  
w <= not d0 when "000",  
not d1 when "001",  
not d2 when "010",  
not d3 when "011",  
not d4 when "100",  
not d5 when "101",  
not d6 when "110",  
not d7 when others;
```

FIGURE 2. Example of selected signal assignment statement.

functions, such phrasing is often tedious. A more flexible notation for conditional situations is useful.

Two forms of signal assignment statement are designed to provide a convenient notation in these situations. The first is called the conditional assignment statement. It allows the use of conditional expressions to filter the expressions that appear in the actual signal

expression itself; the key word *guarded* identifies those expressions that will not change value unless the guard is true ("CLK" is high). Guards need not be only level-sensitive; edge-sensitive guards are possible as well.

The block statement itself is the primary method of grouping different parts of a description in VHDL. With a few technical exceptions, all the ways of decomposing descriptions of behavior into parts are equivalent to different block statements.

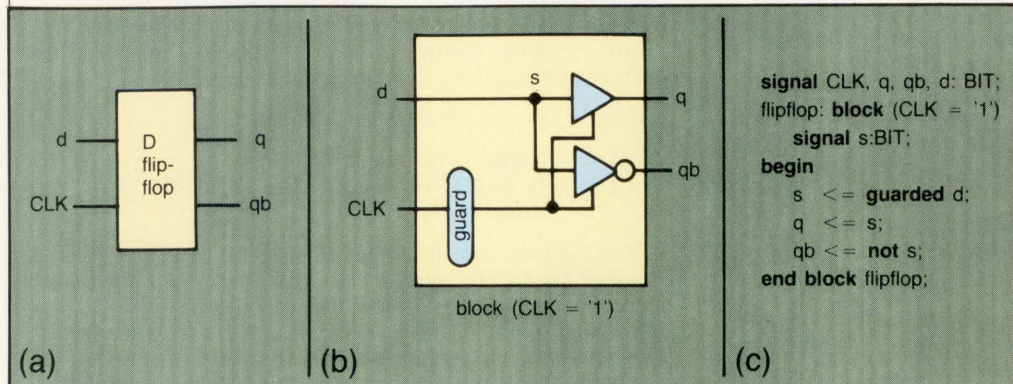


FIGURE 3. D flip-flop (a) represented with "guard" concept (b); VHDL code with block statement (c).

assignment statement. An example is:

```

signal q, r, s: BIT;
q <= q when (r = '0') and (s = '0') else
  '1' when (r = '1') and (s = '0') else
  '0' when (r = '0') and (s = '1') else
  UNDEFINED;
  
```

where "UNDEFINED" is a user-defined function. This statement reflects the action of an RS latch.

The second language structure is the selected signal assignment statement, which allows the results of the expression to be selected by a series of conditions based upon a single value. An example is given in Figure 2, where a single expression embodies the function of an eight-input inverted multiplexer.

### Guarded Signal Assignment Statements

The various forms of the signal assignment statement make the writing of Boolean expressions quick and easy. However, all of the preceding examples apply to asynchronous circuits and situations. There are two aspects to the expression of synchronous behavior: specification of the "clock" or enable circuit, and the grouping of the circuits that are affected by this enable. Both of these problems are solved by the VHDL block statement. A block statement groups one or more signal assignment statements into a related unit. It also may contain an optional *guard* expression. This guard is the specification of the enable circuit that controls the operation of the synchronous circuit.

An example is given in Figure 3. This D flip-flop (encapsulated in the block statement labeled "flipflop") will reflect the value of "d" whenever the clock is high (equal to 1). There is no need to place the clock directly in the

what makes VHDL different from many other hardware description languages.

The basic unit of design description is called a design entity. Any one entity may be reused many times within the overall description. VHDL provides features that allow the design entity to alter its behavior in its different incarnations. Moreover, different implementations for a design entity, corresponding to alternative physical realizations of a given function, may be used in different portions of the description.

```

entity full_adder is
  generic (sum_delay, carry_delay: TIME := 10ns);
  port (x, y, cin: in BIT; sum, cout, strobe: out BIT);
end full_adder;

architecture data_flow of full_adder is
  begin
    sum <= x xor y xor cin after sum_delay;
    cout <= (x and y) or (x and cin) or (y and cin) after carry_delay;
    strobe <= cout after 5ns, '0' after sum_delay;
  end data_flow;
  
```

FIGURE 4. The use of generics to identify delay values.

The first step in setting up a design entity that will contain a number of signal assignment statements is declaring what the entity will look like. An entity declaration fixes the name of the entity and the names by which other design entities will refer to the connections of the entity. An example follows:

```

entity rs_flipflop is
  port (r, s: in BIT; q: out BIT);
end rs_flipflop;
  
```

The name of this entity is "rs\_flipflop" and its ports

are called "r," "s," and "q." Each port has an associated mode, which identifies inputs and outputs. Not shown is "inout," which signifies a port that is both read and written by the entity.

We must specify what the entity does in an *architecture* description. A single entity may have several architectures describing different ways of realizing the entity. An example architecture of the "rs\_flipflop" entity is:

```
architecture implementation_a of rs_flipflop is
begin
  q <= q when (r = '0') and (s = '0') else
    '1' when (r = '1') and (s = '0') else
    '0' when (r = '0') and (s = '1') else
    UNDEFINED;
end implementation_a;
```

Note that there is no signal declaration for "q," "r," and "s."

The port clause in the entity declaration takes the place of this signal declaration. Any statement may appear in an architecture, including a block statement; however, entities and architectures may not be declared inside other entities and architectures.

The design entity should be thought of as a separate conceptual unit on its own. In many cases, this conceptual unit will correspond to an actual part in an assembly. The pins of the part are the ports. We will examine the method of connection in the next section; for now, it is enough to think of the "level of abstraction" as being about the same as a part in a data book.

The design entity as given above allows the decomposition of any hardware design into parts. A design entity has the effect of encapsulating part of the description and making it "general," in the sense that the entity may be used whenever the function described by this code is needed. Thus a design entity is somewhat analogous to a subroutine in a software language.

### Generics

Given that we wish to reuse a design entity whenever practical, we need some means of providing information to the design entity concerning parameters that may change in the various uses of the entity, parameters like delay and capacitance. The means of providing this information to the design entity are called generic constants, or just "generics." Generics fulfill much the same

role in design entities that parameters fulfill in subroutines in software languages. Generic formals, which appear in the entity declaration, may take on different values in the different uses of the entity.

Consider the full-adder entity and architecture shown in Figure 4. The names "sum\_delay" and "carry\_delay" represent delay times that may be changed each time the design entity is used. If no values are given by the user, the default values in the expression inside the generic clause will be used.

The existence of default values on the generic declarations means that the user of the design entity need only provide values for the formal generics if he wishes to change the "normal" behavior of the design entity. Generics can be extremely powerful. For example, if a formal generic appears in a conditional signal assignment statement, the entire behavior of a design entity may be changed by changing the value of a generic.

### STRUCTURAL DESCRIPTIONS

After dividing a large hardware description into parts represented by design entities, the question of instantiating these design entities arises. The user has considerable power to change the behavior of an entity using generics. The method of structural description must allow the user to exercise this power. Indeed, it would be nice if a user could give merely a general description of what a part "looks like." Any entity that satisfies this general description could later be identified with the actual part to be used in a spe-

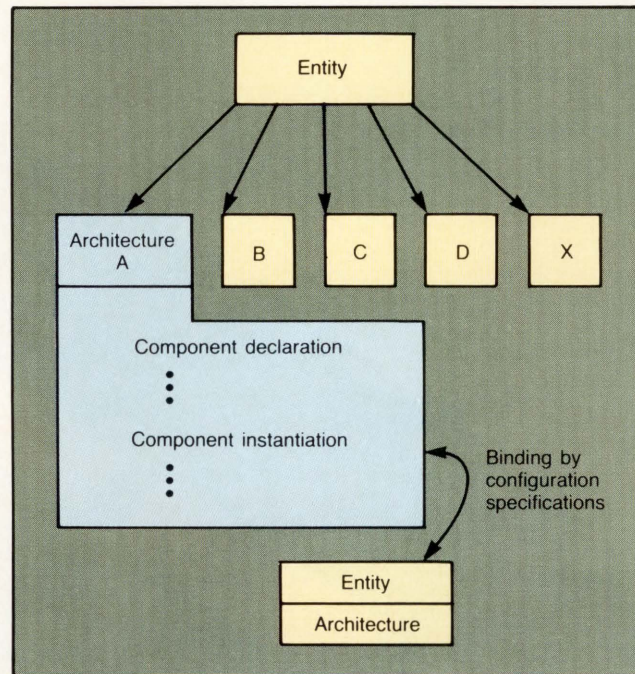


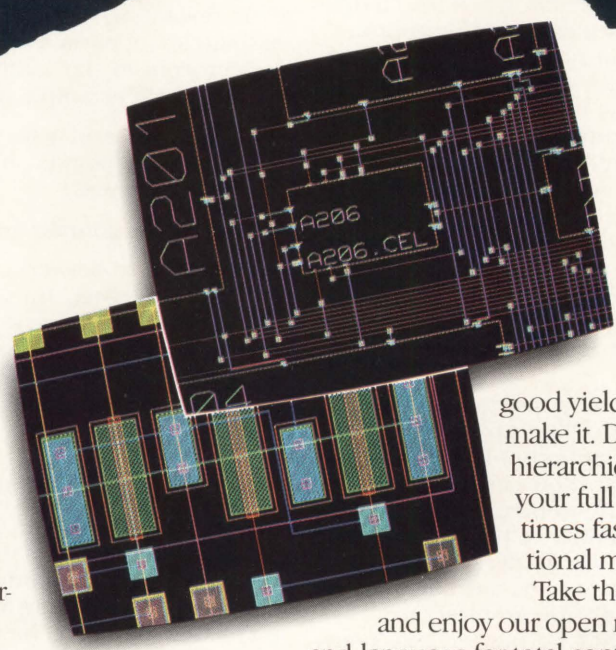
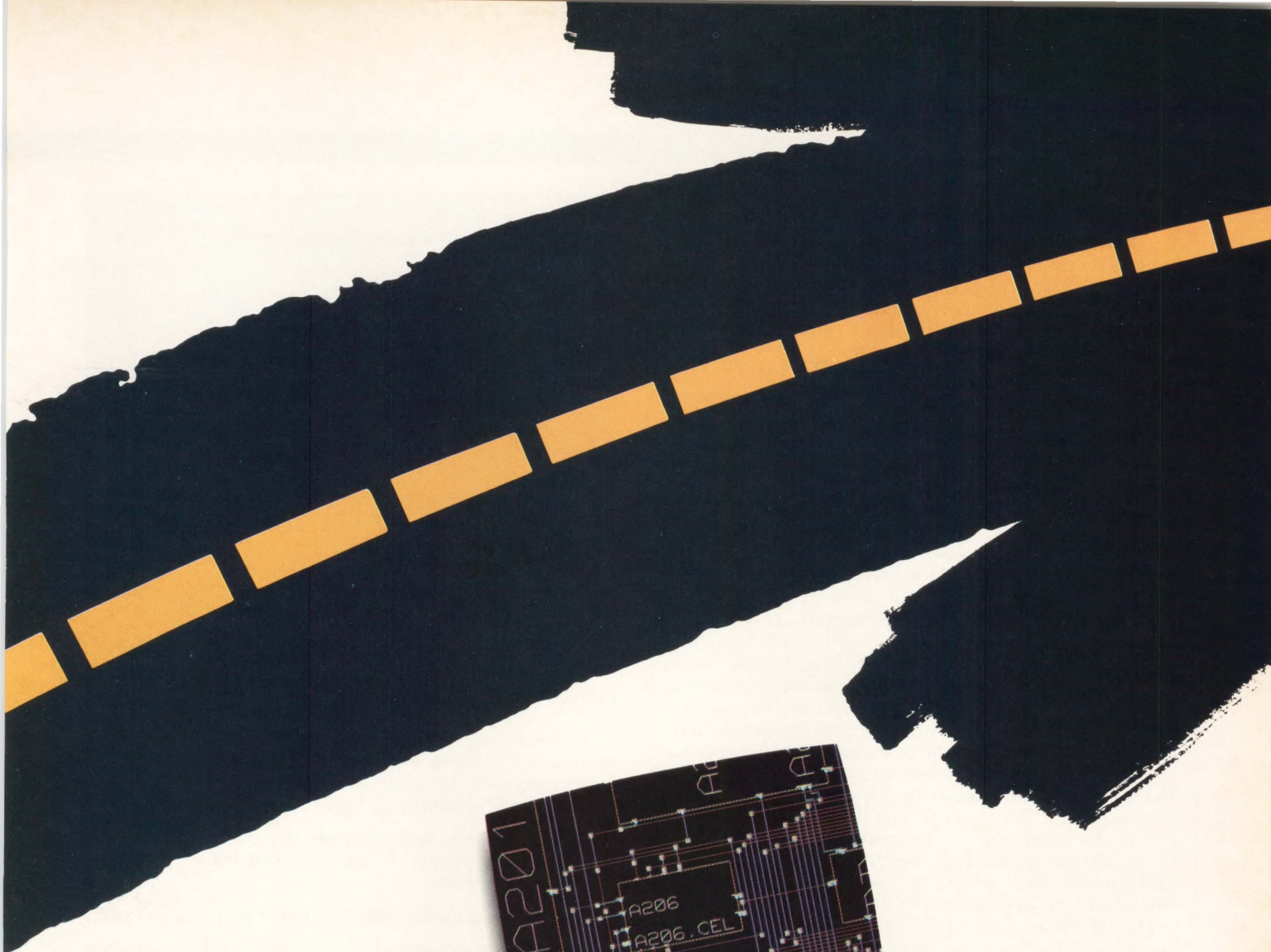
FIGURE 5. Scheme for structural descriptions in VHDL: component instances are bound to entity/architecture descriptions.

cific hardware design.

These goals have been considered in the design of the language features that allow structural description. The concept of a component is introduced as the basic unit of design implementation. The language is organized to permit components to be declared and instantiated within architecture descriptions. Then, a configuration specification binds component instances to the design entities and architectures that describe the desired parts (see Figure 5).

At first glance, it might appear that a component declaration serves the same purpose as an entity declaration. However, there are good reasons to separate the two. With local component declarations, the designer can work from the top down—a component can be used before it is actually in the library.

Our  
open road  
gets you  
to market  
faster.



Our open road architecture means you can choose the IC design tools and hardware you want, when you want them, knowing that each addition contributes synergistically to the power of your IC design system.

Our SYMBAD™ family of integrated layout tools can take you from netlist to maximum density layout from five to ten times faster than with conventional tools. Design rule independent results mean that creation of new layouts for changed processes is almost instantaneous. A new layout language captures your methodologies, converting design expenses into assets. Our industry standard DRACULA™ family, interactively integrated with SYMBAD, ensures that your silicon will work with

good yield, the first time you make it. DRACULA's established hierarchical techniques verify your full layout from five to ten times faster than with conventional methods.

Take the fast lane to market and enjoy our open road: an open database and language for total control, an open chance to use multiple artwork and netlist sources, the open opportunity to work with all the popular hardware choices (DEC, Apollo, Sun, IBM), and the peace of mind of dealing with an industry leader. Contact us today: ECAD, 2455 Augustine Drive, Santa Clara, CA 95054. (408) 727-0264.

Western U.S. (408) 727-0264; Southwestern U.S. (714) 752-8451; Mid-America (214) 869-0033; Eastern U.S. (201) 494-8881; Boston (617) 448-3543; United Kingdom 44-932-568544; Japan 81-3-989-7881; France 33-1-64460531; West Germany 49-89-1782047; Asia 852-5-660136.



Optimizing time and space  
CIRCLE NUMBER 12

## Component Declarations and Instantiations

Given a number of components, eventually to be described by design entities, we need to hook them up to form a circuit. A component declaration describes the component to be hooked up locally, inside a given design unit. A component declaration of "and\_gate" is:

```
component and_gate port (a, b: in BIT; c out BIT);
```

The actual use of a component is identified by a component instantiation statement. A specific use (instance) of the component "and\_gate" in a component instantiation statement is:

```
signal x, y, s1: BIT;  
A1: and_gate port map (x, y, s1);
```

The component instantiation statement states that there is a specific instance of "and\_gate" and that signals "x," "y," and "s1" are connected to ports "a," "b," and "c."

A full structural design entity is given in Figure 6. The design entity in this example is the same as that in earlier examples of the full adder. It is the architecture that is different. Both architectures may be associated with the same design entity. The exact architecture to be used may be selected elsewhere, either when the component is instantiated or by a separate configuration.

There are three lists used to hook up a component. The ports in the design entity declaration are called formal ports. The ports in the component declaration are called local ports. The names in the component instantiation statement, including both local signals and formal ports of the design entity containing the instantiation, are called actual ports.

This three-level hierarchy allows the separation of the component declarations from the design entity declarations. The connection between local ports and actual ports is provided by the component instantiation statement (in the clause preceded by the words *port map*, called the port map aspect).

The separation of the component declaration from the design entity declaration means that any design entity that matches the overall description of the component given in the component declaration can be used. The exact specification of the entity to be used in the final description is deferred. The ability to put off this decision allows the user to make overall decisions without worrying about the specific chips to be used; he may try several chips without changing the architecture.

It is worth remembering that no matter how many "blocks within blocks within blocks" are created by component instantiation statements, it is the signal assignment statements that actually specify the actions. The final view consists of signal assignment statements, encapsulated in design entities, connected by signals or connected lines of signals and ports. Signal assignment statements are the units of action and component instantiation statements specify how the design entities containing them are connected.

### Generic Value Association

The previous section described how signal connections are made through ports. We also need to specify what values will be associated with generic constants.

```
entity full_adder is  
  port (x, y, cin: in BIT; sum, cout: out BIT);  
end full_adder;  
  
architecture structural of full_adder is  
  
  component and_gate port (a, b: in BIT; c out BIT);  
  end component;  
  component xor_gate port (a, b: in BIT; c out BIT);  
  end component;  
  component or_gate port (a, b: in BIT; c out BIT);  
  end component;  
  
  signal s1, s2, s3: BIT;  
  
  begin  
  
    X1: xor_gate port map (x, y, s1);  
    X2: xor_gate port map (s1, cin, sum);  
    A1: and_gate port map (cin, s1, s2);  
    A2: and_gate port map (x, y, s3);  
    O1: or_gate port map (s2, s3, cout);  
  
  end structural;
```

FIGURE 6. A full structural design entity.

This resolution of generic values is very similar to the association of ports with signals. Consider the following refinements to the "and\_gate" declaration and instantiation of the example in Figure 6:

```
component and_gate generic (delay: TIME);  
  port (a, b: in BIT; c out BIT);  
end component;
```

```
A1: and_gate generic map (5ns) port map (cin, s1, s2);
```

The similarity between the port map and the generic map, and between their functions, is obvious.

Once again, actions are defined by signal assignment statements. The generic values in the component instantiation statement must eventually be reflected in a signal assignment statement in order to have any effect on the simulation.

### Configuration Specifications

The binding between a component instance and a design entity is accomplished by a configuration specification, which identifies the entity and the architecture body of the entity to be used. There also are means for connecting any ports that have not been connected by the component instantiation statement and for the resolution of any unresolved generic values. Building on the example above, a configuration specification might appear as shown in the following example. Here the configuration specification begins with the key word *for* and ends with a semicolon:

```
component and_gate generic (delay: TIME);  
  port (a, b: in BIT; c out BIT);  
end component;  
  
for A1: and_gate use entity TTL_and (data_flow_and)  
  generic map (5ns);
```

```
A1: and_gate port map (cin, s1, s2);
```

The configuration specification states that, for the component instantiation statement with label "A1" of the



component "and\_gate," the design entity "TTL\_and" will be the actual design entity and the architecture "data\_flow\_and" will be used as the implementation. The generic value, which is not resolved by the component instantiation statement, will be 5 ns.

In place of the label "A1," the key words *all* and *others* are allowed. The key word *all* specifies all component instantiation statements of a given component declaration (specified by the name following the colon). The *others* key word is the last in a series of configuration specifications and denotes all component instantiation statements not specifically appearing in another configuration specification. In the case shown, the configuration specification applies only to the component instantiation statement with label "A1." The other instance of "and\_gate," as well as all the other statements, are not affected. These may be bound elsewhere, even outside of the given architecture description.

Information in a configuration specification may also be given in a completely separate design unit, called a configuration. Each component instantiation statement in the entire design may be configured in a single configuration. Each architecture and each block within an architecture is identified by name in a hierarchical structure called a block configuration. For each configuration specification that would appear in an architecture, a similar structure called a component configuration appears in the configuration.

However, neither configuration specifications nor configurations are necessary in order to simulate a design. In the absence of a configuration or a configuration specification, the language assumes that the name of the entity is the same as the name found in the corresponding component declaration in the architecture. The names of the ports must match, as must the names of the generics. If all the names match, then no configuration information is needed.

## GENERAL BEHAVIORAL DESCRIPTIONS

When modeling extremely complex circuits behaviorally, signal assignment and component instantiation statements can be cumbersome. A general method of describing behavior is needed. The center of this method is the process statement. The process statement is the true unit of action in VHDL; a signal assignment statement is just a special case of a process statement. With the VHDL process statement, the facilities of a general-purpose programming language are available to the hardware designer.

In addition to most of the general-purpose structures found in Ada, Pascal, or any other programming language, VHDL includes mechanisms specialized for hardware description. One example is the *wait* statement that specifies that a process should suspend execution until a simulation cycle when specified conditions are met. Another example is the behavior of a multisourced signal (called a bus in VHDL) which is defined by a function that is specified in the declaration of the signal. These and other advanced features of the language are beyond the scope of this article. However, they will be taken up in a future article.

## CONCLUSION

This article has set forth some of the basic structures of VHDL. It is an extremely rich language, with a variety of language features for a variety of situations. Specific features are designed to facilitate the description of hardware behavior by means of Boolean expressions, structural description, and algorithmic definition.

The presence of design entities allows a given installation to establish standard design practices. A given installation may use a set of standard library units that correspond to available hardware components. The majority of designers on a project will usually use a small portion of the language features, while a small set of technicians may write and assemble these standard library units.

The language features governing design decomposition allow a large team to work on portions of a complex design in isolation. Different parts of the design may be connected just as any other component or set of components would be connected. The configuration allows the final measure of control, appropriately connecting unconnected ports, resolving unresolved generic values, and selecting the actual architectures to be used for the various portions of the design.

The richness of VHDL entails some complexity. This complexity can be controlled by selecting design practices within an installation that limit the number of language features that need to be learned by the majority of designers. In this manner the complexity of the actual design can be controlled as well as the complexity of learning a rich hardware description language. VHDL can indeed be an asset in the hardware design process, regardless of the size or nature of the hardware. □

## ACKNOWLEDGMENTS

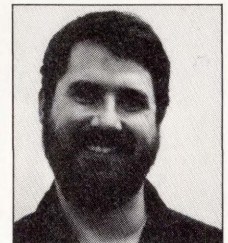
The original suggestion for the overall format of this article came from Dr. Al Gilman. Craig Brown, Rachael Rusting, Al Gilman, and Roger Lipsett all have reviewed earlier versions of this article and made valuable suggestions. The author gratefully acknowledges his debt to these people.

## REFERENCE

VHDL Language Reference Manual, Draft Standard 1076/B, May 1987. IEEE Computer Society Publications Department, Los Angeles, CA.

## ABOUT THE AUTHOR

**David L. Barton** is currently a principal language consultant at Intermetrics. He is also currently enrolled in the PhD program at the University of Maryland. Barton is the designer of PLCB, a programming language for use with a shared-memory multiprocessor. His current research interests include the formal specification and verification of distributed programs. He received the MS degree in computer science from the University of Maryland in 1984.



# Macrochip



**Unleashing the Power  
of Analog ASICs**

# ...It's about time!

## Macrochip—The analog ASIC solution

Leading-edge system designs require powerful IC solutions. As an analog circuit designer, you are challenged to meet system requirements—performance, miniaturization, reliability and cost—on time and within budget. The solution—Macrochip.

## The benefits of true semicustom analog

Ferranti Interdesign brings all the benefits of true semi-custom IC technology to analog circuit designers, because the Macrochip, like Gate Arrays, require customization of only the metal mask to implement even the most sophisticated analog design. Your benefits—Assured performance, design flexibility and low development costs.

## Semicustom analog means powerful reductions in both cost and turn-around time

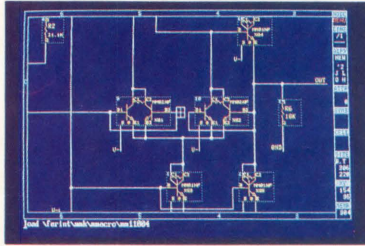
Integration of a typical analog Macrochip, from completed layout to prototypes, starts at \$1,800.00 with a turn-around time of 4 weeks. Compare that to standard cell or full custom!

Our broad selection of Macrochip arrays will let you choose the chip that matches your circuit complexity without having to pay for unused silicon! The Result—A cost effective solution from development through production.

## Macrochip—A growing family

The Macrochip series now offers the designer a family of gridded arrays with various component counts and a repeated cell structure common to all arrays, providing optimum array sizes for any analog design, while permitting the transfer of designs between arrays.

The present Macrochip family of nine 20V bipolar arrays with  $f_T$  of 400MHz (small NPNs) is expanding to include a family of 40V bipolar arrays with a selection of MOS capacitors and high value implant resistors. This allows you to use the Macrochip in a broad range

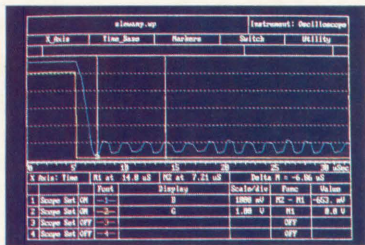


Schematic Capture using FutureNet with Ferranti Interdesign Symbol libraries

of standard, as well as high-voltage, low-current linear applications. Further enhancements to the Macrochip family will soon be available from Ferranti Interdesign for even higher-performance linear applications.

## Complete CAE support for Industry-Standard workstations.

Macrochip is fully supported for schematic capture, simulation and layout with Ferranti Interdesign libraries, using FutureNet®/Pspice™, Analog Workbench™, PC Workbench™ or Mentor™ software. Ferranti Interdesign libraries consist of extensive Macrochip device symbols and Macro subcircuit libraries, models for the various versions of Spice and layout



Circuit Simulation using PC Workbench with Ferranti Interdesign Circuit Models

databases—all provided to our customers free of charge.

## A continually expanding analog cell library.

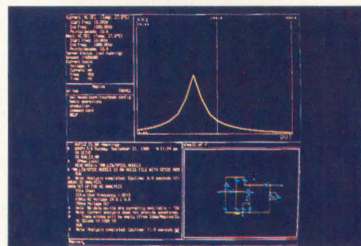
Another powerful feature of Macrochip is our continually expanding Macro cell

library. Predesigned analog circuit functions such as op amps, comparators, sample and hold circuits and timers give you the flexibility to design hierarchically from sub-system level to transistor level—whichever suits your circuit requirements.

## The power of experience.

Ferranti Interdesign brings the experience of over 2,500 analog integrations to meet your analog ASIC requirements. That's more semi-custom analog integrations than anyone in the industry!

Let Macrochip unleash the power of your system design today by contacting a local sales representative, an Authorized



Analog Simulation using MSpice on Mentor with Ferranti Interdesign Circuit Models

Design Center, or call Ferranti Interdesign direct at (408) 438-2900.

### FREE OFFER Macrochip Demo Diskette\*

For a visual demonstration illustrating the ease of analog semicustom using Macrochip, fill in the coupon below, and mail it to:

Ferranti Interdesign Inc.  
1500 Green Hills Road  
Scotts Valley, CA 95066

Name \_\_\_\_\_  
Position \_\_\_\_\_  
Company \_\_\_\_\_  
Address \_\_\_\_\_  
City/State/Zip \_\_\_\_\_  
Country/Phone \_\_\_\_\_

\*5 1/4" floppy disk—requires IBM PC or compatible with EGA or better.

Solutions  
in  
Silicon

**FERRANTI**  
Ferranti Interdesign Inc

FutureNet is a registered trademark of the FutureNet Corp. PSpice is a trademark of the MicroSim Corp. Analog Workbench and PC Workbench are trademarks of Analog Design Tools Inc. Mentor is a trademark of the Mentor Graphics Corp.

# SOFTWARE VS. HARDWARE MODELS FOR SYSTEM SIMULATION

Pete Johnson, Gateway Design Automation Corp., Westford, MA

Designers of VLSI circuits have depended upon simulation techniques for years, owing to the fact that it is very difficult to breadboard an IC. Additionally, the circuit building blocks that IC designers have used have been of relatively low complexity—usually a few logic gates or switches. This simplicity enabled simulation models to be obtained easily, since gates and switches are the traditional primitives of most logic simulators.

The system designer using these ICs at the printed circuit board level, however, has not had such an easy time. Models for a single IC can take months to generate. For a board with a dozen or more complex chips, model development requires a tremendous amount of effort. As a result, most system designers either have not embraced system simulation or have attempted to simulate "around the holes" by mimicking with stimuli the actions of the ICs that had no simulation models.

Clearly, generating and maintaining models for the thousands of complex ICs that systems designers can choose from is a huge undertaking. Currently, two primary modeling techniques are being used: behavioral and hardware modeling.

Behavioral modeling is a software representation of the functions that a given design performs (or should perform). It is used by IC designers in developing and trying out new design ideas. For off-the-shelf components, however, behavioral modeling is used to accurately represent the operation of an existing component.

Generally, behavioral models are written in a high-level programming language (like C or Pascal) or in a hardware description language (HDL) that has been developed exclusively to model hardware designs. HDLs differ from general-purpose programming languages in that they offer additional constructs to directly support the description of hardware. For example, HDLs typically offer additional data types such as *register* and *wire* that can hold electronic values or key words to describe asynchronous behavior like a signal changing. Figure 1 shows a small behavioral model of a 32-bit ALU that demonstrates some of the features of a HDL.

In a system simulation, behavioral models would be

```
module alu(a__bus, b__bus, c__bus, control, clock);

input clock;
input [1:0] control;
input [31:0] a__bus, b__bus;

output [31:0] c__bus;

reg [32:0] c__reg;
reg [31:0] a__reg, b__reg, c__bus;

always @ (posedge clock)
begin
  a__reg = a__bus;
  b__reg = b__bus;
  case (control [1:0])
    2'b 00: c__reg = a__reg + b__reg;
    2'b 01: c__reg = a__reg - b__reg;
    2'b 10: c__reg = a__reg & b__reg;
    2'b 11: c__reg = a__reg | b__reg;
  endcase
  c__bus [31:0] = c__reg [31:0];
end
endmodule;
```

FIGURE 1. Example of a behavioral model written in an HDL: a 32-bit ALU.

mixed with other models of the rest of the components to be simulated. If these additional components are described at various levels of abstraction (such as gate or switch representations of SSI or MSI components), the resulting simulation is called *mixed-level*.

Many large companies have developed behavioral models of components that they work with often. However, this development requires a dedicated set of engineers and is very expensive. Lately, third-party companies have emerged that generate behavioral models of many popular components. These companies can usually offer a lower-priced model than could be developed internally, since they can amortize the development cost over many customers and simulators. Typically, prices range from \$500 to \$6000.

An alternative to behavioral modeling is hardware

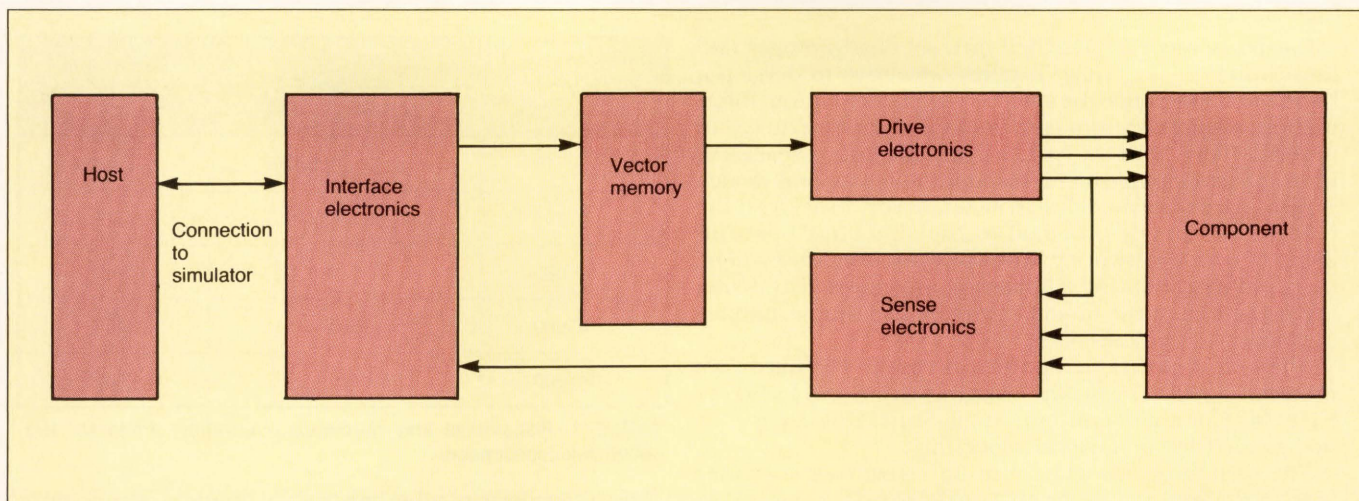


FIGURE 2. Typical structure of a physical modeling system.

modeling. Also called physical modeling, this approach uses the component itself to model its functionality in a simulation. Typically, a hardware modeling system consists of a set of dedicated hardware that a target chip can be plugged into, as shown in Figure 2, plus the necessary software. The simulator sends input data that is to be evaluated to the modeling system, which applies that data to the actual component and records the chip's response; the response is then sent back to the simulator. Note that the hardware model usually supplies only the functional information of the component's response; software is required to supply timing information.

Hardware modelers are usually stand-alone systems and are connected to a network, like Ethernet, or attached directly to a workstation or hardware accelerator.

## TRADE-OFFS

Neither behavioral models nor hardware models are the ideal solution to every system design situation. However, understanding the advantages and disadvantages of each type can help the designer determine which one (or a combination of the two) best suits the design requirements.

The advantages of behavioral models fall into three main categories: flexibility, availability, and cost.

A behavioral model is extremely flexible. Since it is entirely represented in software, it can be modified simply. Modifications that a designer may want to make include adding more timing information (like minimum and maximum delays or timing checks) or changing operating conditions of the component (for example, making a 12-MHz component a 16-MHz part).

Behavioral models are available as soon as they can be encoded and tested. Usually, all that is needed to begin development of a behavioral model is a standard data book description of the component. There is no need to actually have silicon for the component to develop the model.

Finally, there are no hardware costs in developing or using a software model. Since they are developed using

standard languages, they will run on the same computer that the simulator is being run on.

The cost to develop these models, however, can be large. Since the model is usually developed by someone other than the IC designer, it takes time to understand the complete operations of the component. Even once the part is completely understood, the time required to encode and test the model can be many months.

Also, once developed, behavioral models are difficult to maintain. As the IC vendor releases more formal data books or introduces new versions of the component, the model must be updated.

One final disadvantage of behavioral models is their speed of evaluation. Although they are much faster than

**Hardware models can evaluate very quickly. If the modeler is directly attached to the simulator host, little software overhead will be incurred in evaluating a hardware model.**

a corresponding gate-level model, the evaluation time for a single input change can be many milliseconds, even on mainframe computers. For a design with a lot of components and a large number of input changes, simulation can take a long time.

Purchasing models from a third party can remove some of these obstacles. A third party may already have a particular component available or be in the process of developing it. In that case, the designer may want to do a make-versus-buy analysis. For complex components, the purchase price will usually be less than the cost to develop the model internally.

Hardware models have their own set of advantages and disadvantages. Typically, it takes only days to develop a hardware model once the silicon is available. Timing information is obtained from data books and can be entered in advance. Thus both the cost and time are dramatically less than for developing the corresponding behavioral model.

Also, hardware models can evaluate very quickly. If the modeler is directly attached to the simulator host (instead of over a network), little software overhead will be incurred in evaluating a hardware model. Once the data are presented to the model, the response time is usually measured in nanoseconds.

There is, however, a significant cost to the hardware modeler itself. Typically ranging from \$35,000 to \$100,000 for a well-configured machine, this expense is not needed for behavioral modeling.

The design group will try to amortize this cost over multiple users and place the modeler on a network so that multiple simulations can take advantage of it. Sharing a hardware modeler reduces its performance advantage over behavioral modeling, since networking software must be executed and other network traffic must be contended with for each evaluation of the component.

Additionally, hardware models limit the length of simulation for most of today's popular VLSI components, owing to the dynamic nature of these chips. This same property also dramatically increases the evaluation time as the length of the simulation increases.

***The evaluation speed at which of a behavioral model evaluates depends on many factors: the speed of the simulator and the host computer, the efficiency of the modeling language, and even how well the model is written.***

Since simulation performance is a critical issue in evaluating a design methodology, the next section explores in more detail the issue of speed of evaluation of behavioral versus hardware models.

## **SPEED ISSUES**

The speed at which a behavioral model evaluates depends on many factors: the speed of the simulator and the host computer, the efficiency of the modeling language, and even how well the model is written. However, the evaluation time will vary linearly with the length of the simulation. A similar evaluation of a given part will always take approximately the same amount of

Number of instructions	Hardware modeling time (s)	Behavioral modeling time (s)
10	0.00059	1.4
100	0.059	14
1,000	5.9	140
10,000	590	1,400
100,000	59,000	14,000

**TABLE 1. Behavioral and hardware evaluation times for the 68000 microprocessor.**

time, regardless of whether that evaluation occurs early or late in the simulation.

The same cannot be said for most hardware models. These components, which are typically referred to as "dynamic," require a minimum clock frequency to maintain their internal state. Since the simulator cannot guarantee a minimum evaluation frequency, the hardware modeler handles this problem by maintaining the complete set of input evaluations. Each time the simulator requests a new evaluation, the modeler replays the entire evaluation history at a speed sufficient for the component to maintain its internal state. Consequently, each evaluation takes increasingly more time, as the length of evaluations to be played back increases. This increase is quadratic in nature, proportional to the square of the number of evaluations.

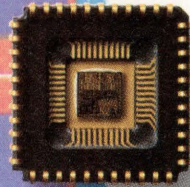
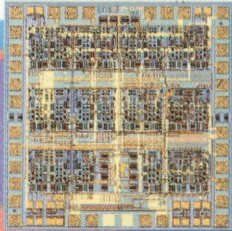
A simple look at the preceding information could result in the following assessment being made: For short simulations, hardware models are faster than behavioral models; whereas for longer simulations, behavioral models will be faster. Though this statement is generally true, there are many other factors that affect this relationship.

Although many of today's popular components are dynamic in nature, some are not. A nondynamic ("static") component does not need the replay of the entire evaluation history before each new evaluation, since it does not lose its internal state. Consequently, as with behavioral models, the evaluation time for these components changes linearly with simulation time and in long simulations can be dramatically less than for dynamic components.

The speed of the simulator host computer is also important. Many designers use workstations for simulation. The dramatic increase in their performance over the last few years directly affects the evaluation performance of behavioral models. Mainframe computers can further boost the simulation performance. In contrast, hardware modelers can be speeded up only to the maximum operating frequency of the component to be modeled. Although this value, too, is rising, it is not increasing as quickly as workstation CPU performance.

New behavioral modeling techniques are also improving the speed at which model evaluation is performed. One of these techniques, called "bus-functional" modeling, reduces some of the accuracy and functionality of the model, but it gives significantly higher performance.

# Custom performance, QuickChip™ turnaround.



## Imagine.

The performance, speed and reliability of Tektronix bipolar ICs. Plus technical expertise in analog design second to none. Together they cut a direct path to market without cutting quality for your new products.

## That's Tek's QuickCustom™ approach.

It begins with a family of seven QuickChip arrays to start the design process. Within weeks you have analog or analog/digital ASIC's in hand that meet your requirements.

Tek delivers the training, graphic layout and simulation tools, plus access to an experienced Tektronix IC Design Engineer. You get the performance you need at QuickCustom prices.

**So call (800) 835-9433  
ext. 100.**

Get your hands on the semi-custom, analog and analog/digital IC development resources you'll need. Full custom ICs are also available.

### WHAT ARE QUICKCHIPS?

#### Analog QuickChip family:

- 150—524 NPN Transistors
- $f_T$  typical to 6.5 GHz at 15V or 2.5 GHz at 65V.

#### Analog/Digital QuickChips:

- Gate propagation delay: 400 ps
- Digital function library

**Tektronix®**  
COMMITTED TO EXCELLENCE

Thus these models can offer a fast, inexpensive way to remove many design errors.

If multiple uses of a single component are required within a design, using a hardware modeler can slow down the simulation, reduce the length of simulation, or increase the cost of the hardware modeler. The reason is that the multiple instances of the component must either share the same physical memory of the hardware modeler or else spend time in having to swap patterns in and out of this memory, or multiple copies of the chip must be used. In general, using multiple instances of a behavioral model does not impose any of those penalties.

In either case, hardware accelerators cannot help VLSI model evaluation performance. Since the accelerator handles only gate- and switch-level evaluations, both behavioral and physical models must be evaluated separately (the behavioral models on an attached host, the physical models on an attached hardware modeler). If the amount of evaluation taking place in these models is significant, then the accelerator is spending most of its time waiting for evaluations. However, hardware modeling may be preferable when an accelerator is used, since the user may be willing to dedicate a modeler and tightly couple it to the accelerator.

### DETERMINING THE OPTIMAL MODELING SOLUTION

Given the myriad of complexities, it is still possible to determine which modeling technique offers better performance under a particular set of constraints. A designer can evaluate the set of components to be modeled and the available simulation tools to get a good idea which technique(s) are preferable.

The main issues that need to be considered in this evaluation are:

- Device complexity
- Modeling system performance
- Modeling system structure
- Logic simulator and host computer performance

Device complexity will determine the amount of evaluation time required for a behavioral model; a model for an 8085 microprocessor, for instance, will evaluate much quicker than a model for a 68020. Complexity has much less of an effect for hardware models.

The modeling system performance may limit the speed at which the hardware model is evaluated. Many of today's popular components can operate faster than the typical 16-MHz frequency of most of the available hardware modelers.

The structure of the modeling system also will affect evaluation performance. Some modeling systems save evaluations from the simulator until a change is noted in any of the specified strobe pins. For example, if a single data pin has changed, the evaluation would not be presented to the component. Once a strobe pin has changed (such as a clock pin rising), the entire input set is presented to the component for evaluation. This method reduces the number of total evaluations performed and so can reduce simulation times or lengthen the simulation (since the simulation must end once the

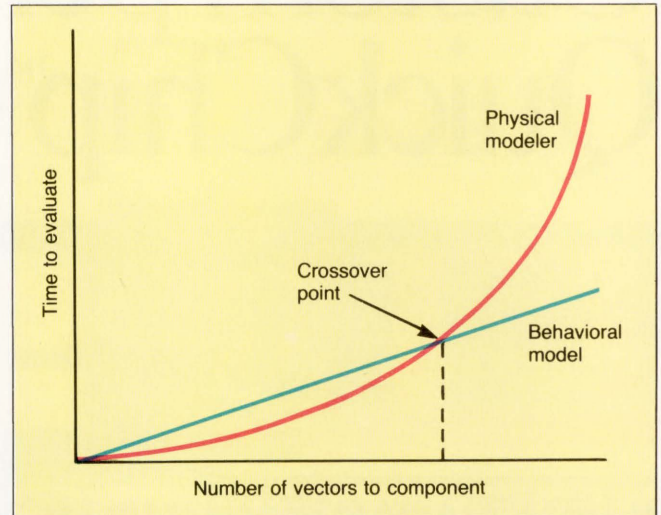


FIGURE 3. Relative performance of behavioral and hardware modeling.

available memory for evaluations has been filled).

A networked system will significantly slow down the evaluation time, as will a system that allows a single physical component to be shared by many users.

Finally, the speed of both the logic simulator and the host is important. The simulator's speed is directly related to the evaluation speed of the behavioral model and influences the evaluation of hardware models, since an interface must be implemented. Note that in some logic simulators a similar interface must be available for behavioral evaluation as well. The speed of the host computer that the simulator is running on is likewise important, in that faster CPU performance will decrease behavioral evaluation times.

### DESIGN EXAMPLE: SIMULATING THE 68000

The following design example concerns the simulation of a Motorola 68000 microprocessor. This example uses just the 68000, but evaluation times should be proportional when the part is used in a complete system design. A behavioral model of the 68000 was generated in an HDL and simulated with a mixed-level simulator on an engineering workstation. As for the hardware modeler, two different types were evaluated, making some assumptions in an attempt to compare simulation times.

Simulating the 68000 with a behavioral model resulted in a performance of approximately 7 instructions per second on a 2-MIPS workstation. Since this performance is linear, it can be extrapolated to any length of simulation desired to compare it with that of a physical modeling system.

The first hardware modeling system analyzed is one that presents each evaluation as it occurs. Thus a clock edge or other strobing signal is not treated differently from any other signal change. Typically, this type of system produces about 2.3 evaluations per clock cycle of the microprocessor. Assuming about 6 machine cycles per 68000 instruction, that figure translates into 13.8

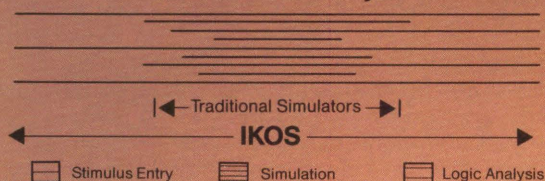


# Without the IKOS Simulation System, your IC design cycle is spent like this.

## There's only one way to see 8K gates, simulated with 20K vectors in 15 seconds.

The IKOS Simulation System. IKOS isn't just a piece of hardware designed to "speed up" somebody else's simulation software. Nor is IKOS just software meant to run simulations only. IKOS is a comprehensive, integrated system incorporating all three steps of logic validation: stimulus entry, simulation *and* logic analysis. If you're working with any other tool, you're working in the dark.

### Circuit Simulation Cycle



## Command a 100:1 Performance Advantage in Stimulus Entry

IKOS lets you enter stimulus the way you think about it, by drawing waveforms. Never again will you spend time debugging arcane stimulus programs, or laboriously typing in ones and zeroes. You'll gain a 100:1 performance advantage over traditional techniques.

## Simulate Seconds, Even Minutes of Real Time

IKOS substantially lowers the risk of IC failure in the target system.

Since there's *no limit* on the number of test cases you can generate and analyze with IKOS, you'll be able to simulate seconds, or *even minutes* of real time system operation. You'll get a full, detailed picture of how well your circuit is performing instead of just a millisecond snapshot.

## Real Time Logic Analyzer Dramatically Speeds Analysis

IKOS incorporates a real time logic analyzer that runs as the simulation runs, and instantly finds and displays the logic condition you're interested in. Gone are the hours spent pouring over output listings or running off-line comparison programs.

There's more, much more to look over in the IKOS Simulation System—like its ability to accept netlists from all major workstation vendors, accurate timing modeling, semi-custom library support, fault simulation, *plus* a low price compared to alternatives. Let us show you eye-opening performance and accuracy gains for every step of the logic validation process.

Give us a call.



IKOS

Simulators for the  
Real World.  
408-245-1900

IKOS Systems, Inc.

145 N. Wolfe Rd.  
Sunnyvale, CA 94086

CIRCLE NUMBER 10

**Designers can determine which modeling technique offers better performance under a particular set of constraints. The main issues to be considered are device complexity, modeling system performance, modeling system structure, and logic simulator and host computer performance.**

hardware evaluations per instruction. This modeler is capable of applying 16 million evaluations per second. Assuming no overhead in networking or other software, the total evaluation time can be calculated for a given number of instructions and compared with the time needed for the behavioral modeler. Table 1 gives some calculated simulation times for this hardware modeling system and behavioral models executing on the 2-MIPS workstation.

Note that the hardware evaluation time increases by the square of the number of instructions ( $100 \times$  increase for each  $10 \times$  increase in instructions), whereas the behavioral evaluation grows linearly. Note also that at some point between 10,000 and 100,000 instructions, the behavioral evaluation time begins to be less than the hardware evaluation time.

It can be shown that this crossover point—that is, the number of instructions to be executed before the total evaluation time for behavioral models is equal to the total evaluation time for hardware models (see Figure 3)—is about 24,000 instructions (3400 seconds, or 57 minutes, of evaluation time). Thus simulations involving less than this amount will be faster using hardware models, and simulations using more will be faster using software models.

A modeler that applies one vector per machine cycle would move this crossover point higher, since fewer evaluations are being performed. In this case, the point moves to 127,000 instructions. (This difference is about 5.3 times, which not coincidentally is the square of 2.3, the ratio of the number of evaluations for the two hardware modeling systems.)

Note that both of these are very long simulations. Even 24,000 instructions require about one hour of simulation, and that assumes that there are no other models in the design. For most simulations, therefore, the hardware modeler will be faster.

Of course, if the modeler is networked, the crossover point will move in the other direction. Even with the faster modeler, adding in a network response time of 40 ms for each evaluation (20 ms in each direction), the

crossover point moves from 127,000 instructions to about 91,000 instructions. For the slower modeler, the point moves from 24,000 instructions to about 17,000 instructions.

The other way in which the point can be significantly moved is by making the behavioral model run faster. The same simulation running on a 10-MIPS workstation or mainframe would move the two crossover points to about 25,000 and 5,000 instructions for the faster and slower modeler, respectively. Adding in the previously defined networking overhead, the crossover points for both systems become negative, meaning that even a single instruction is faster behaviorally than with the hardware modeler.

## NONPERFORMANCE ISSUES

We have focused mostly on the performance issues between hardware and behavioral models. There are, however, two other key issues that cannot be ignored in deciding between these modeling techniques. The first is simulation length. For most components (those that are dynamic), simulation with a hardware modeler must end once the available memory for evaluations has been filled, as previously mentioned. For most hardware modelers, this limitation ranges from 16k to 256k evaluations. This length is probably sufficient for most simulations, but if the user is planning on running very long simulations, such as batch regression tests, the limit may be reached. For behavioral modeling, there are no such limits.

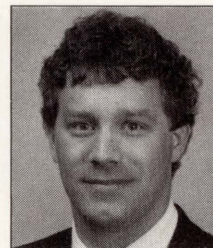
The other major concern is the availability of models. Assuming that the design group is unable to develop its own internal models of complex components, the availability of models is dependent on outside resources. In most cases, silicon will be available before a third-party behavioral model. However, recently third-party vendors have been striking up relationships with IC manufacturers to deliver behavioral models before first silicon is available.

## SUMMARY

Both behavioral and physical modeling can partially answer the system simulation modeling problem. However, neither technique is a solution by itself in all cases. A user must carefully consider the alternatives in terms of costs, performance, and length of simulation. Often, however, availability will be the determining factor. □

## ABOUT THE AUTHOR

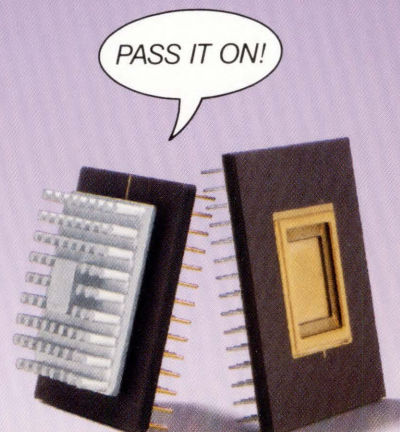
**Pete Johnson**, product marketing manager, came to Gateway from Daisy Systems, where he was a marketing manager for simulation and test tools in the digital design automation division. He spent six years at IBM as a designer and a design manager. Pete received his BSEE from Clarkson University in Potsdam, NY.



ASIC Design Kits

**VALID**

# The Best Kept Secret In CAE!



# The Word Is Out!

## The word is out on who has the best ASIC solution.

If you're looking for a complete and easy solution for ASIC designs, you've come to the right company.

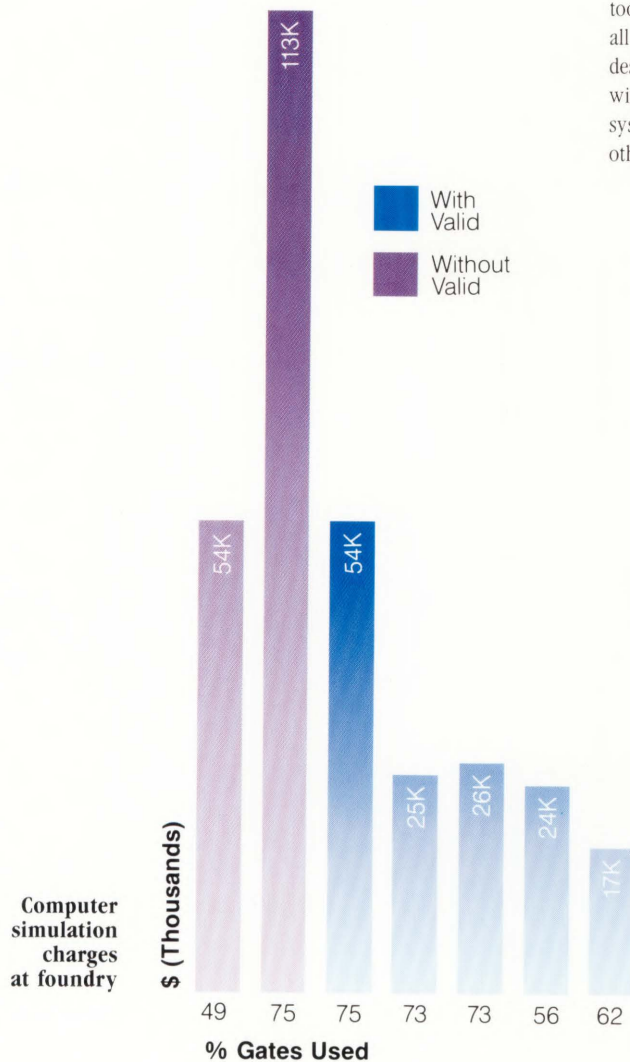
When you're designing semi-custom IC's, your biggest challenge is to design them quickly and accurately. To achieve this, you need two things: The best CAE design tools. And complete ASIC Design Kits from the leading vendors. Valid and its ASIC vendors can provide you with both.

## Valid makes it easy.

Valid's ASIC solutions allow you to reap the benefits of designing with ASIC devices while dramatically reducing non-recurring engineering (NRE) costs and long development cycles.

And you can ease into our solution. You use the same CAE tools you would use to design with off-the-shelf components. All Valid tools, from schematic capture through simulation to net list generation, operate as easily on ASIC designs as on designs that include only standard components.

What's more, not only do all Valid tools support ASIC design, but all Valid platforms support ASIC design, too. So you can design with ASIC devices on the same system you use for all your other designs.



## Valid offers plenty of ASIC design support.

More than 97 design kits have been developed by the leading ASIC vendors to support our CAE solutions. Each ASIC Design Kit is fully qualified by the ASIC vendor to ensure that the verification you perform on your Valid workstation will meet the requirements of the vendor.

Moreover, Valid offers ASIC-specific tools. For example, if your ASIC needs include programmable device support, then use ValidPLD™

**APPROVED!**

It provides support for the full spectrum of programmable devices. ValidPLD is fully integrated with industry-standard PLD tools, such as ABEL™, CUPL™ and PALASM™. And you enter the program for your device only once. In the format of your choice. ValidPLD then automatically generates a graphical representation and validation models to be used with ValidGED™

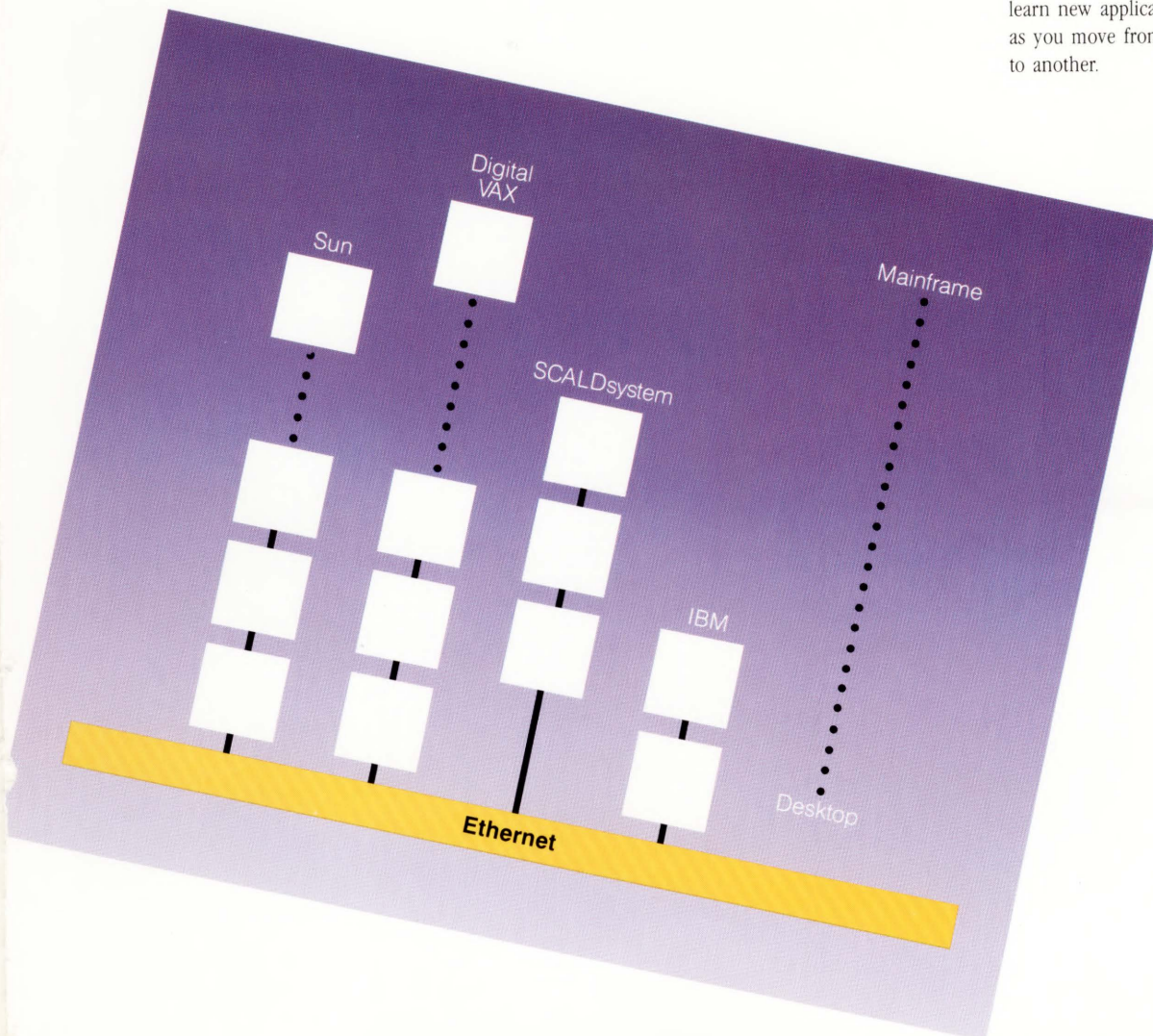
Graphics Editor, ValidSIM™ Interactive Logic Simulator, and ValidTIME™ Timing Verifier.

If your ASIC needs run to silicon compilation, Valid's silicon design solution allows you to use silicon compilation technologies from within the same Valid environment. Our menu-driven silicon compiler not only automatically generates layout for you, but it also simultaneously generates schematic symbols and validation models to be used with ValidGED, ValidSIM and ValidTIME.

## Design your ASIC devices on industry-standard platforms.

Valid's design validation software features ValidGED and ValidSIM, a simulator that can distinguish between ASIC and PC board validation needs. The software runs on the VAXstation II™, Sun Workstations®, IBM PC AT™ or on Valid's own SCALDsystem®. All of these systems can be networked together, using the industry-standard Ethernet™ protocol TCP/IP.

You can configure the network of your choice, mixing and matching platforms to meet your company's needs. The same software runs on all platforms, so you're not locked into one. Nor do you have to learn new application software as you move from one platform to another.





**Valid Logic Systems**

2820 Orchard Parkway  
San Jose, CA 95134  
(408) 432-9400  
Telex 371 9004

**Valid International**

Valid House  
39 Windsor Road  
Slough, Berkshire SL1 2EE  
United Kingdom  
44 753 820101  
Telex 847 318

**Nihon Valid**

Tokyu Building  
2-16-8 Minami-Ikebukuro  
Toshima-Ku, Tokyo 171  
Japan  
81 3 980 6421  
Fax 981 8775

ValidPLD, ValidGED, ValidSIM, ValidTIME, SCALDSYSTEM,  
Realchip, Realmodel are trademarks of Valid Logic Systems  
Incorporated. ABEL is a trademark of Data I/O Corporation.  
CUIPL is a trademark of Assisted Technology Incorporated.  
PALASM is a trademark of Monolithic Memories Incorporated.  
UNIX is a trademark of A.T.&T. Bell Laboratories. IBM PC AT is a  
trademark of International Business Machines Corporation.  
VAXstation II is a trademark of Digital Equipment Corporation.  
Ethernet is a trademark of Xerox Corporation. Sun Workstation  
is a registered trademark of Sun Microsystems, Incorporated.

## Valid helps you integrate ASIC devices into system-level designs.

Designing your ASIC device is only one step in the design cycle. Equally important is being able to easily integrate the device into your system level design and verification. Valid's ASIC solution handles every step of the system integration process.

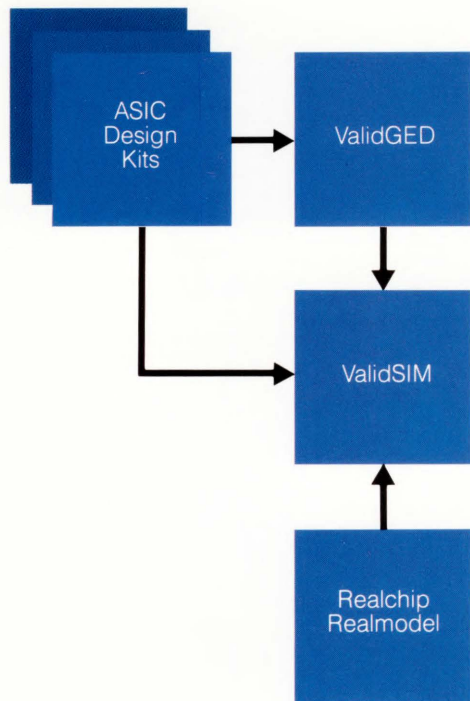
For example, after designing your device, but before laying it out, you need to ensure that the device works within your target system design. ValidSIM's estimated wire delay handles this verification by letting you estimate the effect of wire delays before the chip is actually laid out. And, since ValidSIM includes features for both PC board and ASIC simulation, you don't have to switch simulators when you do a system level simulation with your ASIC device.

Then, after your device has been laid out, but before fabrication, ValidSIM lets you simulate your

design with backannotated wire delays. So you know the fabricated device will work within your target system.

When you receive your first fabricated ASIC devices, you can use ValidSIM, augmented by either of Valid's hardware modeling systems\*, Realchip™ or Realmodel™ to test the devices. Realchip or Realmodel enables ValidSIM to use the real physical IC chip to model the function of the device while it obtains its timing information from a user-specified file. So you don't have to wait for full-speed parts to use the real devices in your simulation.

Finally, after verifying the functions of your devices, you can continue to use it with Realchip or Realmodel as the simulation models for the remainder of your system-level simulation.



\* U.S. Patent No. 4,590,581

**Now that you're in on it, we'll give you the whole story.**

**YES!** I'd like more information about Valid's ASIC solution.

Please have a Valid Sales Representative call me.

I'd like a demonstration of Valid's ASIC solution.

Please send me more information about Valid. My specific ASIC needs are \_\_\_\_\_.

Name \_\_\_\_\_

Title \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_

State \_\_\_\_\_ Zip \_\_\_\_\_

Phone (\_\_\_\_) \_\_\_\_\_

**Now that you're in on it, we'll give you the whole story.**

**YES!** I'd like more information about Valid's ASIC solution.

Please have a Valid Sales Representative call me.

I'd like a demonstration of Valid's ASIC solution.

Please send me more information about Valid. My specific ASIC needs are \_\_\_\_\_.

Name \_\_\_\_\_

Title \_\_\_\_\_

Company \_\_\_\_\_

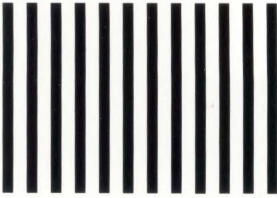
Address \_\_\_\_\_

City \_\_\_\_\_

State \_\_\_\_\_ Zip \_\_\_\_\_

Phone (\_\_\_\_) \_\_\_\_\_

NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES



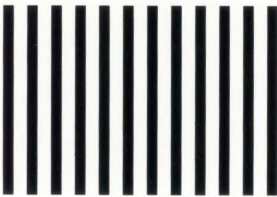
**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT NO. 7656 SAN JOSE, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Valid Logic Systems  
ATTENTION: P. Clark  
2820 Orchard Parkway  
San Jose, CA 95134

NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES



**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT NO. 7656 SAN JOSE, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Valid Logic Systems  
ATTENTION: P. Clark  
2820 Orchard Parkway  
San Jose, CA 95134

**Select your ASIC  
technology from the  
leading ASIC vendors.**

Today, ASIC Design Kits are available for the following ASIC vendors:

AMCC	Mitsubishi
AMD	Motorola
AMI	National
ASEA HAFO	NCR
AT&T	NEC
CDC	OKI
Fairchild	Phillips
Ferranti	Plessey
Fujitsu	RCA
Harris	Ricoh
Hitachi	SAGEM
Hughes	SGS
IMI	Signetics
IMP	SMC
IMSC	Thomson-Mostek
LSI Logic	TI
Matra Harris	Toshiba
MEDL	VTI
Mietec	

And we're adding more all the time. Because in our effort to offer the easiest, most complete solution for ASIC design, we're constantly working with ASIC vendors to develop and qualify their Design Kits for Valid.

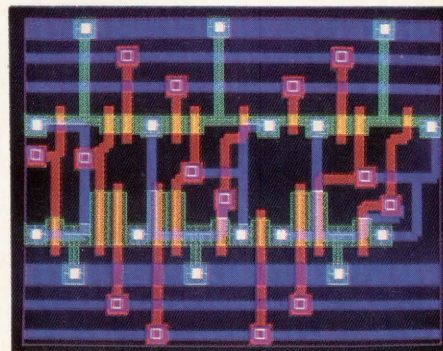
Contact your Valid sales office for the most up-to-date list.

It's no longer a secret. Valid offers the best ASIC solution in the industry. To get the whole story for your company, complete and mail the attached reply card today.





# III PHYSICAL DESIGN



## 64 LEAF CELL DESIGN

M.Y. Tsai and Stephen Wu, ECAD Corp.

Leaf cells are built better by experienced designers following a set of guidelines than by design automation tools, because the range of possible solutions is too great for these tools. Such tools should aid the designer, not replace him, by preserving his designs and design practices through symbolic design and a procedural layout design language.

## 72 GRAPHICAL FLOORPLAN DESIGN OF CELL-BASED ICS

Edmond Macaluso, Tektronix CAE Systems Division

This article describes the range of tasks necessary for floorplan design—editing device specifications and placement; estimating area; creating and evaluating assemblies; and performing placement, channel editing, and layout evaluations—and an interactive graphical floorplanning system that encompasses them.

## 80 A RULES-DRIVEN APPROACH TO CIRCUIT BOARD DESIGN

Joseph Prang and Katherine Gambino, Valid Logic Systems Inc.

Printed circuit board layout is conventionally driven by physical design parameters. This article shows how an expert system can integrate the electrical specifications with the physical requirements in accordance with rules specified by the CAE designer during schematic capture.

# LEAF CELL DESIGN

M.Y. Tsai and Stephen Wu, ECAD Corp., Santa Clara, CA

**T**hanks to advances in silicon technology, it is now feasible to put millions of devices on a single chip. Such capability presents designers with some challenges, the complexity of which can be controlled through some simplifications. The best way to simplify the problem is the "divide and conquer" method (Mead and Conway, 1980). Hierarchical design or other approaches based on regularity all build upon the same basic building units, called *leaf cells*.

The composition of these cells and their relationships to higher-level blocks has been the focus of numerous papers and software tools (Hu and Kuh, 1985). However, basic leaf cell design has for the most part been left in the hands of layout and circuit designers. Till now almost all leaf cell designs have been done through manual digitization. Because of the intricacy involved (too many possible solutions), leaf cell design is the most time-consuming task in the design process, regardless of the design methodology used—gate array, standard-cell, structured custom, or silicon compilation. Improve the productivity in leaf cell design and you will have a major impact on the difficulty of VLSI design.

Design rule independence is becoming almost as important as controlling complexity. ASIC designers do not like being restricted to a single foundry, and they would like to be able to "port" their designs from one fabrication process to another. Even within one fabrication line, the rapid advance of silicon technology makes design rules obsolete within one or two years and invalidates existing successful designs.

Preservation of designs with design rule changes is a key issue in the semiconductor industry. In the past it was a problem without a solution. Redesign and layout were the only methods available to implement design rule changes. Recent changes in CAD, however, make it possible to build a design-rule-independent leaf cell library that can easily adapt to changes in technology.

Because leaf cell design is the foundation of VLSI design, this tutorial will discuss some of the methodolo-

gies and tools that can help speed up the design process and establish process independence.

## DESIGN METHODOLOGIES FOR LEAF CELLS

Leaf cell design is basically a problem of placement and routing of devices and polygons. It differs significantly from cell and block placement in a few aspects.

First off, the placement of devices inside leaf cells is very different. Typically, there are no fixed-sized objects like those found in gate arrays or standard-height cells. Instead, designers have to design with objects of arbitrary shapes. Placement methods for cells or blocks will not apply with leaf cells.

Second, the routing of elements within a leaf cell is not subject to the same constraints as the routing of cells or blocks. There are no restrictions for routing over objects as long as the routing layers have no conflicts. Most wires are not restricted by a preferred wiring direction. Also, the connections to the devices in the leaf cells can be made in many places. In cell or block layout, in contrast, wires avoid device areas and connect only to designated terminals. The lack of design constraints complicates the layout of leaf cells, but it also makes the leaf cells compact.

Finally, the placement of devices within a leaf cell is subject not only to geometric design rules but also to electrical rules such as maximum resistance and capacitance and CMOS latch-up protection. Because it is a multiple-constraint problem, leaf cell layout is best left to a designer's experience. It is generally true that experienced layout designers can produce designs manually that are better than automatically generated designs.

Because leaf-cell design depends on the skill of the designer, it is important to study typical design methodologies. This discussion is divided into three parts: design rules, electrical rule guidelines, and layout algorithms.

## DESIGN RULES

When placing and routing devices inside a leaf cell, a designer must follow the design rules absolutely. Any violation of the rules voids the whole design. A typical design rule set for all layers in a process can contain 100 rules that specify exactly how closely polygons can be placed within and next to one another. The designer memorizes all the rules and tries to achieve optimal packing density without breaking any of them. A small violation can take a whole day to correct if the correction requires moving many polygons.

As technology advances, more layers and structures are defined for processes, thus increasing the size of the design rule set and the complexity of leaf cell design. The adherence to design rules is an area more addressed by advanced design rules, as discussed in the section below on CAD.

## ELECTRICAL RULE GUIDELINES

In addition to design rules, designers must typically follow electrical rules that ensure good electrical behavior. Even when the guidelines are clearly specified, the output of layout designers can vary greatly in area and quality. Though these designs may satisfy all design rules, the layout may still have problems with noise, latch-up, or electromigration. These problems contribute to the need for redesign of leaf cells.

Electrical rule guidelines are not absolute and may vary from design to design. Typically, each company or project has its own guidelines for designers. Electrical rule checking programs can be constructed to determine compliance with the rules. To understand these guidelines, consider the following example of a typical electrical rule set for CMOS technology:

1. All the cells should have both p and n wells to allow a choice of p- or n-well processing. Depending on the foundry, one of the wells may need to have a guard ring or may have to be eliminated.
2. Substrate and well contacts should be placed at every contact to  $V_{CC}$  or  $V_{SS}$ . The only exceptions are for memory arrays. The maximum spacing between two well contacts is 100  $\mu\text{m}$ . In almost all cases, however, the spacing should be less than 100  $\mu\text{m}$ .
3. Stacked devices in complex CMOS gates may cause latch-up because the floating diffusion can be coupled to high or low voltages. Special well contacts around these nodes are needed.
4. To reduce the potential for latch-up, connections between  $n^+$  diffusion and  $V_{CC}$  and between  $p^+$  diffusion and  $V_{SS}$  should be placed wherever possible. These connections reduce well resistance and provide a current sink.
5. Input buffers need special structures, such as double guard rings, to guard the chip from external dangers such as electrostatic discharge.
6. All contact cuts and via cuts should be single size. An array of contact cuts should be used instead of a large single contact cut, because the quality of large contact cuts is difficult to control during processing.

7. To increase yield, in areas that are not determinate of chip density, object spacing should be greater than the minimum design rules.
8. Where metal connects to diffusion, there should be at least one contact placed at every 10  $\mu\text{m}$  of diffusion to minimize the effect of diffusion resistance.
9. Wide gates should be folded to reduce polysilicon resistance. The maximum length of any one polysilicon gate segment should be 50  $\mu\text{m}$ .
10. The first metal layer should avoid dynamic nodes to prevent dynamic coupling.
11. Terminals on the boundaries of cells have special requirements that facilitate routing. For example, polysilicon input wires should be placed far enough apart to allow for contacts to first metal.

Each process will have specific design constraints that add to this list.

***When placing and routing devices inside a leaf cell, a designer must follow the design rules absolutely. Any violation of the rules voids the whole design. . . . A small violation can take a whole day to correct if the correction requires moving many polygons.***

## LAYOUT ALGORITHM

While obeying the design rules and electrical rules, designers should consider some general layout algorithms when creating leaf cells. For example, the following guidelines help designers create dense leaf cells:

- Because the overall layout scheme for a design dictates the design of leaf cells, the designer should have a well-defined strategy for chip layout before beginning leaf cell design. The direction of power and signal lines, the use of particular routing layers, and the terminal locations of design blocks all influence leaf cell layout.
- In very regular designs, leaf cells should share contacts, wells, and power lines to create the most compact circuit blocks.
- For simple regular structures like PLAS, the designer should minimize the area required for product terms through the use of folding (Obreska et al., 1986). Minimization of product terms is another technique. Both techniques apply only to programmable logic array structures.

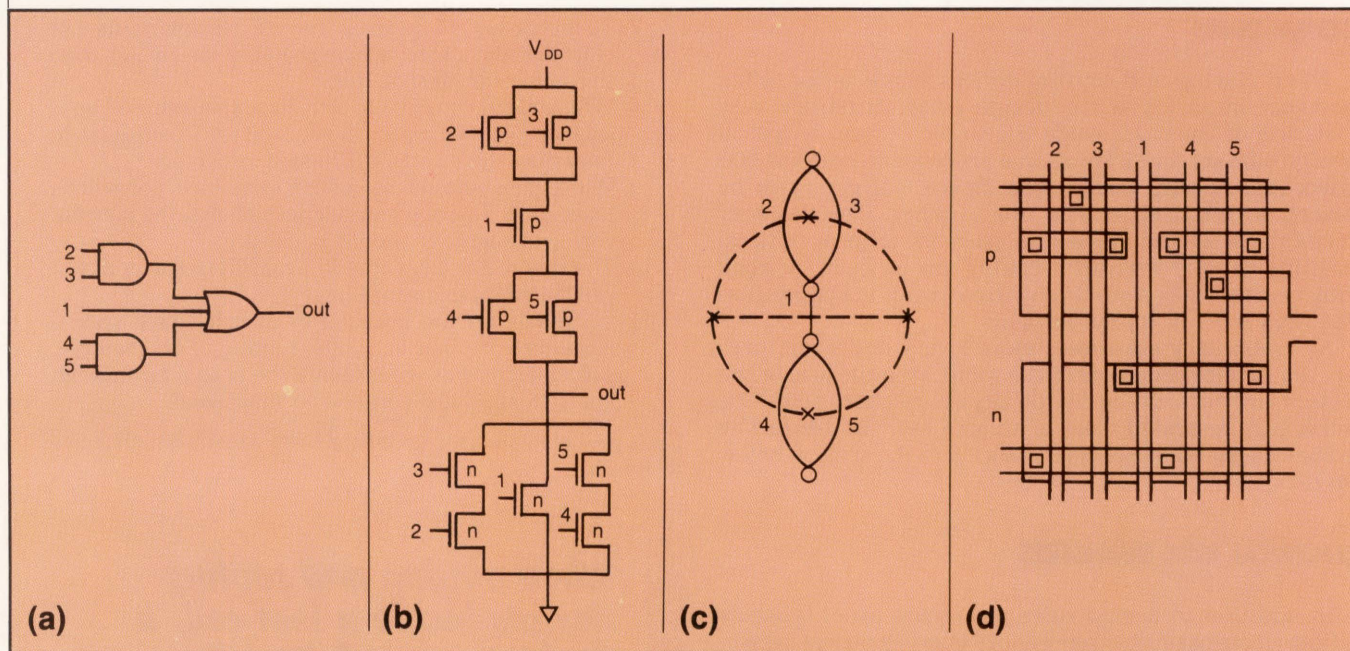


FIGURE 1. An example of an algorithm for the layout of complex CMOS gates: logic diagram (a), circuit (b), graph model (c), layout (d).

## The problems of laying out leaf cells are too complicated for a single algorithm to solve.

- Some layout algorithms exploit layout techniques for transistors that are tied to the voltage rails. Placement for such transistors becomes a linear problem for designs of complex CMOS gates. By converting a layout of a Boolean function into a graph that has diffusion areas (source and drain) as vertices and transistors as edges (see Figure 1c), some algorithms can be applied to minimize area. If two edges are adjacent in the graph model (built of transistors), then it is possible to place the corresponding gates into a physically adjacent location and connect them by diffusion. If there is a Euler path (a sequence of edges that contains all the edges of the graphic model), all the gates can be chained by the diffusion area. This algorithm is useful only for the layout of complex CMOS gates, as shown in the example in Figure 1, drawn from Uehara and Van-Cleemput (1981).

The problems of laying out leaf cells are too complicated for a single algorithm to solve. Layout designers must evaluate individual situations and apply the best algorithms for each case. Design automation systems, therefore, should only provide methods that increase the productivity of layout designers.

Instead of supplying some limited synthesis tools to replace designers, it would be best to develop tools that aid the designer in completing and preserving his de-

signs. Once leaf cells are designed, it would be useful if his original design can be updated to new design rules without painstaking rework. It also is important for design tools to capture not only designs but also the designer's procedures and experiences.

## PRACTICAL LEAF CELL CAD

Leaf cell design has advanced from cutting rubylith sheets to Mylar paper drawings, to manual digitizing CAD systems, and finally to the current polygon editors. Although each advance has increased designers' productivity, each method works on a geometric design that conforms to precise design rules. The designer must take the time to place object just far enough apart to obey the design rules but close enough to create a compact layout.

Designers could be much more productive if they were not bogged down with numerous design rule details. A symbolic layout tool like the SYMBAD Object-Oriented Editor (OED) allows the designer to design topological arrangements of cell devices without dealing with the precise coordinates of each object. By not focusing on the design rules, the designer can create better layouts faster. This advance in leaf cell design has two major enhancements: symbolic objects and design rule independence.

Instead of using polygons as the smallest building unit in a layout, object-oriented layout uses transistors, wires, and contacts as building blocks. These building blocks are composed of polygons, but the designer does not alter the polygons directly. Other symbolic layout techniques, such as sticks (Hseuh, 1979), which forms transistors at intersections of diffusion and polysilicon lines, do not have the following advantages of object-

oriented layout:

- The object can be sized according to user-defined parameters like gate length. Also, the exact physical size of the object is displayed on the screen so that the designer has a better idea of the relative positions of objects in the cell.
- Because the designer works with objects, the layout work resembles schematic capture, making layout more feasible and easy for circuit designers.
- Each object is a distinct entity, so that it is possible to associate attributes with the objects. For example, the user may want to indicate with text the use of each object for his own reference or for input to other design tools. This natural extension of object-oriented design helps designers capture their design intentions on the design itself.

Object-oriented layout increases the efficiency of design entry, but it needs compaction techniques to make the resulting layout compact. Only recently have compaction algorithms satisfied run-time and memory requirements to be practical for layout designers. The compaction program that works with OED can support large designs because, in contrast to earlier algorithms, the run time and memory requirements increase linearly with design size. This compactor, which is based on constraint graph techniques, can compact an average of 10 transistors per CPU-second on a 1-MIPS computer.

The compactor has other new features that increase design density. Its built-in expert rules can merge equipotential elements such as source and drain diffusion areas and contacts. It can minimize the resistance of interconnects and can insert "jogs" (bends) and align the edges of objects automatically. The user can specify constraints in the compaction space to control the outcome of the compaction.

The compactor compacts the design to correct design rules so that the designer need not incorporate the rules in his symbolic design. This approach can help the designer lay out 5 to 10 times as many devices each day compared with manual, polygon-based methods. For example, the clock driver in the upper window of Figure 2 was completed in two days with two levels of hierarchy. The resulting layout, in 1.6- $\mu\text{m}$  technology, is only 9% larger than a manual layout of the same design. In addition, the symbolic design can be easily modified, especially to new design rules, a capability the manual

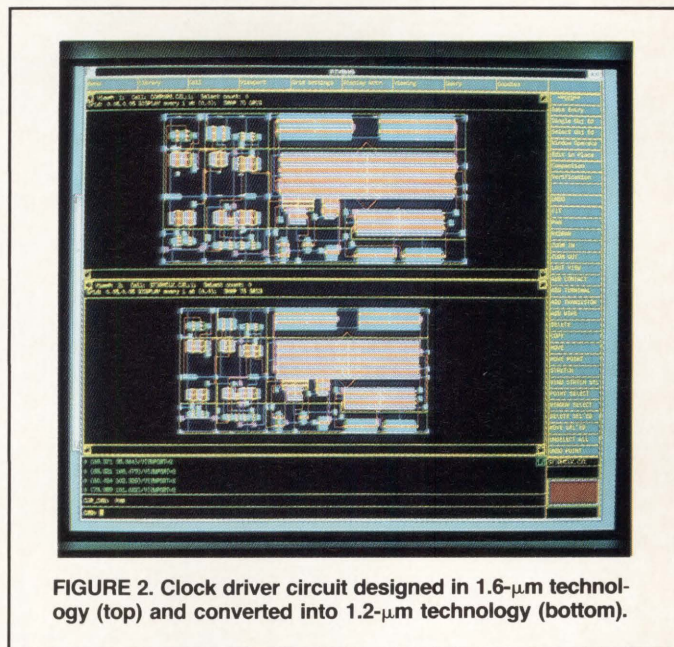


FIGURE 2. Clock driver circuit designed in 1.6- $\mu\text{m}$  technology (top) and converted into 1.2- $\mu\text{m}$  technology (bottom).

polygon-level layout does not have.

Because the design is entered symbolically, the user can map the design into a new set of design rules merely by entering the new rules into the compactor and letting the compactor convert the layout to the new rules. Transistors in the design can be converted to a new desired width or to a fixed ratio of the original width. The lower window in Figure 2 shows the clock driver recomputed to a set of 1.2- $\mu\text{m}$  design rules, a redesign that required no input from the user other than the entry of the new design rules. The faster entry method and easy modification of symbolic design make it a better method for leaf cell design.

## PRESERVING DESIGN PROCEDURES

Symbolic design helps the user preserve his leaf cell designs by allowing the cells to be easily modified. The next step in the automation of leaf cell design is the preservation of the design procedures. Layout programming languages let the designer use programming techniques to instruct design system to lay out the cells. Language constructs like conditional statements, loops, variables, and macros (procedure calls) help designers re-create their design procedures in a layout program.

One of the most important features of the SYMBAD programming language is the ability to handle layout objects easily. The designer does not need to know the details of how the layout objects are stored in the database, drawn on the screen, or handled in a batch environment. If, say, he wants to add an n-type transistor at the origin, he types in *add transistor ntr (0 0)*.

The designer may add more information, such as the width and length of the transistor, to the command string. If any parameters are not specified, the SYMBAD

**Because the design is entered symbolically, the user can map the design into a new set of design rules merely by entering the new rules into the compactor and letting it convert the layout to the new rules.**

# IF YOU'RE DESIGN YOU'RE WAST



Until now, if you wanted to put 50,000 gates on one chip, you usually had to put them in one at a time. You had to put in three months work. And you had to put your launch date into a holding pattern.

Not anymore.

**Announcing the  
only compilers that take you to  
gate arrays and cells. Fast.**

Now with VLSI's new Datapath and State Machine Compilers you can design in high level symbols instead of gates. A design that used to take months, can now

be turned around in days. Even less.

With the help of our new Datapath Compiler you can design a 64-bit RISC datapath on your lunch hour.

But can you use control logic? You bet your sweet datapath you can.

Our State Machine Compiler accepts high level expressions of logic functions and gives you tightly packed state control logic in a fraction of the time it would take to design it by hand.

We did the design entry for an asynchronous receiver in one hour. It would've taken

# ING WITH GATES, ING YOUR TIME.



seven days using traditional schematics.

And not only do we give you high integration design tools, we give you high integration devices.

Our CMOS 1.5 micron VGT100 series of gate arrays puts as many as 50,000 useable gates on a chip. And our 1.5 micron CMOS cell-based technology packs over 100,000.

If you'd like more information about

our new Datapath and State Machine Compilers, and the VGT100 family of gate arrays they work with, write us at 1109 McKay Drive, San Jose, CA 95131. Or give us a call at (800) 872-6753.

We'll show you a good time.

*To find out how much time you can save, call us for a free stopwatch.*



**VLSI TECHNOLOGY, INC.**

CIRCLE NUMBER 15

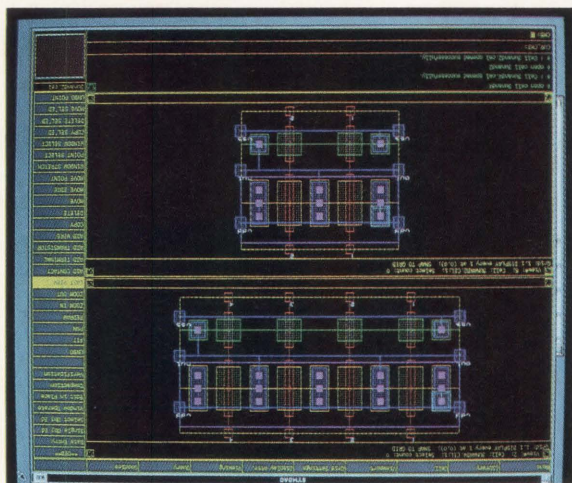


FIGURE 3. Two cells generated by a NAND-gate SPL macro.

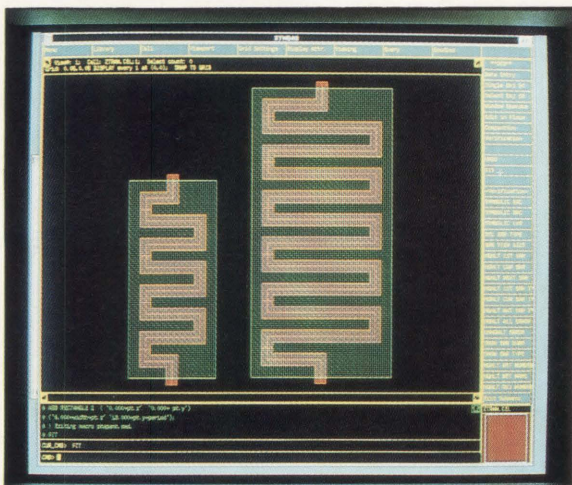


FIGURE 4. Bent transistor generated by SPL macro for driver circuits.

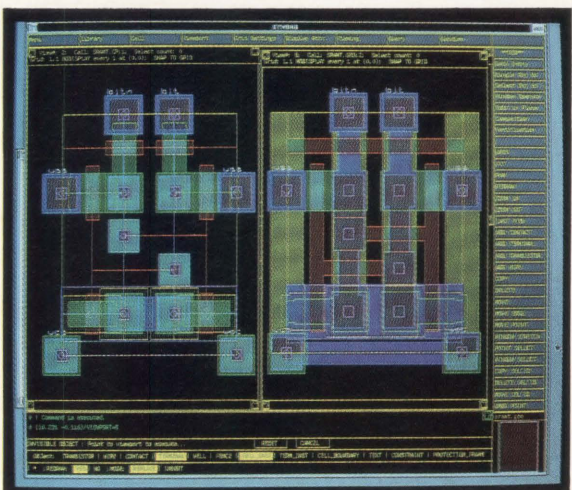


FIGURE 5. Leaf cell for RAM arrays.

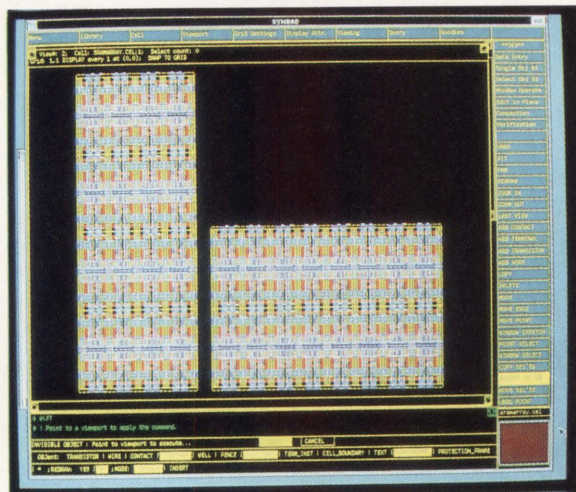


FIGURE 6. RAM arrays built by SPL procedure from RAM leaf cell.

system retrieves the information from a technology file or prompts the user for more input.

The SYMBAD programming language (SPL) can handle not only those objects predefined in the system but also objects designed by layout designers. For example, the designer may design a different type of transistor structure and use that object within his designs, a process that simplifies the development of complex leaf cells. The user-defined objects can be described either by SPL macros or by symbol description files that can be compiled and stored in the symbol libraries. The use of user-defined objects makes SPL more versatile than other layout programming languages. Since the language is designed for use by layout and circuit designers, details of data structures and memory utilization associated with general programming languages remain hidden.

After allowing user-defined objects, the next step in capturing a designer's expertise in the layout system is the development of tools to generate leaf cells to the requirements of a particular design. The programming language lets the designer use parameters and programmed procedures to create a custom generator, thereby enabling him to capture his own design procedure, instead of using predefined cell generators.

The use of parameters is very simple when writing a procedure for a customized cell generator. Instead of assigning a fixed number to each object in the layout, the designer assigns a parameter; when the designer calls up the cell generator, the program receives a value for the parameter. For example, a NAND gate generator may specify the channel width as a parameter to change the cell layout according to performance and area requirements. By keying in a value for the parameter, the designer gets custom leaf cells for different applications.

Often the number of inputs to a logic cell may be defined as a parameter. Some looping and control statements enable the design system, instead of the designer, to perform the repetitive actions needed to build multi-



ple inputs. Figure 3 shows two cells generated by a procedure that accepts the number of inputs as a parameter. Another use of looping and control statements results in a procedure that can build bent transistors for driver circuits. As shown in Figure 4, both the number of bends and the width of the bends can be specified as an input parameter to the procedure.

Such procedures can be implemented most easily on a symbolic design system because the builder of the procedure need not consider design rules. Objects can be placed loosely in the design because the compactor will adjust the locations to create a tighter layout that satisfies design rules. User-designed macros do not need design rules defined in their procedures. Just as the interactive designer places components without worrying about design rules, the macro writer focuses only on design procedures.

**Objects can be placed loosely in the design because the compactor will adjust the locations to create a tighter layout that satisfies design rules.**

The writing of procedures can extend to the creation of generators that build circuit blocks from leaf cells. Consider the RAM cell built in OED that is shown in both symbolic and geometric views in Figure 5. The following SPL macro can build a RAM block from that cell according to a user-specified size:

```
child_cell = s$get_string("Enter child cell name for
quadruple cell")
quad_cell = s$get_string("Enter quadruple cell
name")
! Create quadruple cell
execute quadruple_macro 'child_cell', 'quad_cell'
open cell SRAM_ARRAY
row = s$get_number("How many rows")
column = s$get_number("How many columns")
add array 'quad_cell' /row = 'row' /colu = 'col' &
/xdis = 'dx*2' /ydis = 'dy*2' (00)
close cell
```

"Child\_cell" is the RAM leaf cell, and the quadruple cell is built by the SPL macro "quadruple\_macro" from four leaf cells mirrored about the *x* and *y* axes. The latter is built to allow the leaf cells to share power and signal lines. After prompting the designer for the number of rows and columns, the example builds the RAM block "SRAM\_ARRAYS" starting at the origin (00) and advancing twice the distance of the leaf cell for the placement of each quad cell (see Figure 6). Both the starting point and the name of the block could also be specified as parameters.

All the SYMBAD tools, including the compactor, checker, floorplanner, and placer and router, can be invoked with SPL commands. Also, SPL includes a rich set of SYMBAD system functions that give the user access to the database, control over file management, and even the ability to invoke his own application programs. In effect, the user can set up his own development environment, building his own block compilers and, by nesting macros and procedures, his own design automation tools. □

#### REFERENCES

Cho, Y.E. 1985. "A Subjective Review of Compaction," *22nd Design Automation Conference*, Las Vegas, NV.

Hseuh, M.Y. 1979. "Symbolic Layout and Compaction of Integrated Circuits," UCB/ERL Report M79/80, University of California, Berkeley, CA.

Hu, T.C., and E.N. Kuh. 1985. *VLSI Circuit Layout: Theory and Design*, IEEE Press.

Liao, Y.Z., and C.K. Wong. 1983. "An Algorithm to Compact VLSI Symbolic Layout with Mixed Constraints," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*.

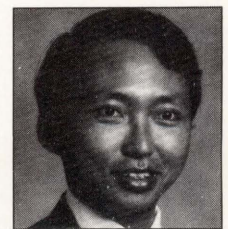
Mead, C., and L. Conway. 1980. "Introduction to VLSI Systems," Addison-Wesley, Reading MA.

Obreska, M., et al. 1986. "PLA and Custom Design," *Design Methodologies*, ed. S. Goto, Elsevier Science Publishers, New York, NY.

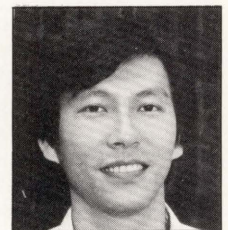
Uehara, T., and W.M. VanCleemput. 1981. "Optimal Layout of CMOS Functional Arrays," *IEEE Transactions on Computers*, C-30.

#### ABOUT THE AUTHORS

**Mon Yen (Mike) Tsai** is director of advanced custom design and synthesis tools at ECAD, where he is working on the SYMBAD suite of custom design tools. Before joining ECAD, he was the manager of CMOS block design at Silicon Compilers Inc. (now Silicon Compiler Systems Corp.). He has also worked at IBM's Thomas J. Watson Research Center, where he did circuit design for the Micro 370 32-bit single-chip processor and other VLSI projects. Dr. Tsai received his MS and PhD in electrical engineering from the University of Illinois at Urbana-Champaign.



**Stephen Tzyh-Lih Wu** is the project leader in charge of the SYMBAD Programming Language (SPL) and Framework at ECAD. Earlier, he developed an interactive software system for optimization-based system design at the Electronics Research Laboratory, University of California at Berkeley, where he received a PhD degree. He was a research consultant for Honeywell in 1983 and a visiting research scholar at Imperial College in London in 1985. His research interests include interactive software system design, system integration, layout synthesis, and optimization theory.



# GRAPHICAL FLOORPLAN DESIGN OF CELL-BASED ICs

Edmond Macaluso, Tektronix CAE Systems Division, Austin, TX

Cell-based IC layout has become more prevalent as IC complexities have increased and design times have shortened. However, current CAD systems for cell-based layout, many of which are based on systems for simpler standard-cell layout, leave the groundwork to the user. What's missing is a floorplan design system that throughout the design process addresses the fundamental problems of working with hierarchical layout and variable cells.

As the first step in IC design, the floorplan is a general guide for the design and layout of an IC. It specifies the size, shape, port locations, and relative locations of devices at each hierarchical level. The initial floorplanning is refined during the design process as more detailed information becomes available. In this way, floorplan specifications control the subsequent detailed layout of a cell-based design. In VLSI design, proper floorplanning ensures that die size is minimized and that constraints between on-chip circuits are met.

Methods for cell-based layout include standard cells, building blocks, or combinations thereof. The building blocks can be custom layouts, groups of standard cells, or compiled layouts. The resulting designs look like custom chips; in fact, most recent large microprocessors are cell-based layouts (DuPont et al., 1986).

In laying out cells of varying size, the cell-based system requires more complex placement and routing algorithms than a standard-cell system. The question of what block shape is best arises. In addition, because cell-based layout is hierarchical, portions of the design will be done independently and combined in a top-level step. This approach facilitates delegation of design tasks and reduces design time.

Of the newer cell-based layout systems (*VLSI Systems Design*, 1987), those derived from standard-cell systems have undergone significant changes. Their routers must be changed to recognize blocks of varying sizes and to complete complex power supply distribution. Similar significant changes are required in adapting placement algorithms. These systems do not always address other aspects of cell-based design, such as defining the shapes

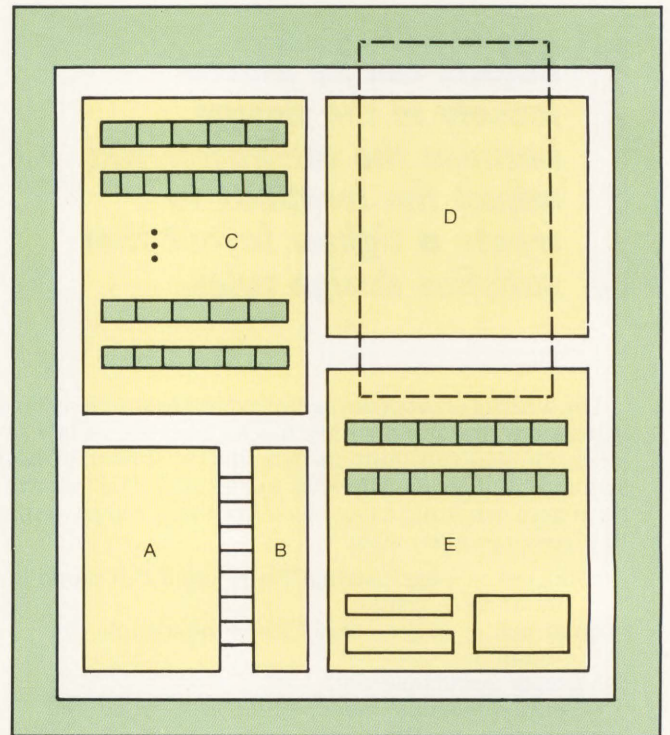


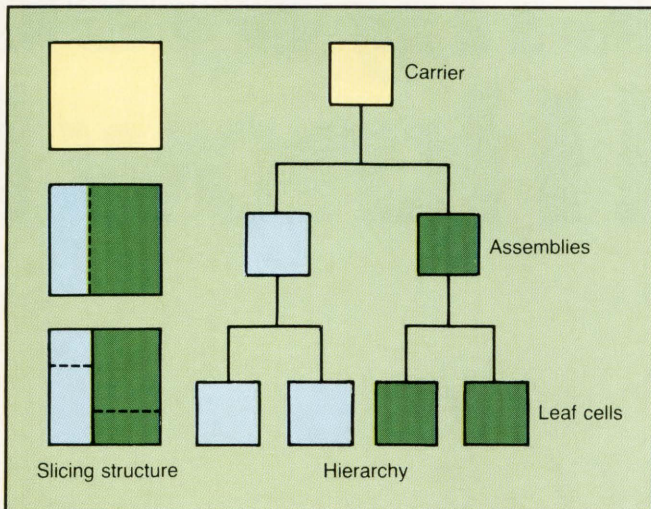
FIGURE 1. The appearance of devices in a cell-based design.

of blocks, defining the layout hierarchy, and controlling the cell-based layout sequence. A floorplan design system performs all these tasks.

## DEFINITION OF FLOORPLAN DESIGN

The objective of cell-based placement is to arrange devices in the smallest possible area without violating constraints on connections between the devices. When devices have a fixed size and shape, this objective becomes a classic placement problem (Lauther, 1979). When devices can assume any size and shape (within specified limits), their arrangement is called floorplan

*This article is based on "Graphical Floorplan Design of Cell-Based VLSI Circuits," which appeared in VLSI Systems Design, April 1987.*



**FIGURE 2.** The slicing structure and hierarchy in a cell-based design.

design (Preas et al., 1979; Heller, 1981; Otten, 1982).

To describe floorplanning concepts, a definition of terms is necessary. A *device* can be a standard cell or a block in a cell-based layout. *Standard cell*, *block*, and *macro* are devices with particular placement and routing characteristics (see Figure 1). A *standard cell* has a variable width and fixed height, so it can abut other standard cells to form rows. A *block*, or *macro*, has an arbitrary size and shape and is not required to abut other devices.

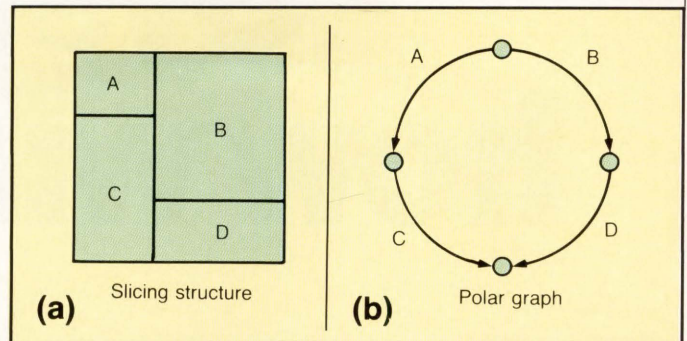
*Leaf cell*, *assembly*, and *carrier* describe how a device fits into a placement hierarchy (see Figure 2). A *leaf cell* is the lowest level device in the hierarchy and cannot be subdivided. An *assembly* (Preas et al., 1978) comprises a group of devices. A *carrier* is the assembly at the hierarchy's top (root).

Finally, the device's physical description is its *package*. A packaged leaf cell or carrier has a defined I/O interface, such as a pin-out for an IC or an edge-conductor specification for a board. The physical description for an assembly is called its *placement area*.

## FLOORPLAN DESIGN SYSTEMS

Proprietary and commercial cell-based-design systems that exist do not address many aspects of floorplan design. For example, most proprietary systems are oriented toward automatic floorplan design using algorithms that optimize the size and shape of devices as they are placed (Woo et al., 1986; Relis et al., 1986; Wilk et al., 1986; McNeary et al., 1986). Although these systems have reported good automatic results, they provide limited interactive control.

Commercial design systems, both full-custom and cell-based, address some aspects of floorplan design but do not provide all the automatic algorithms. Full-custom editors can vary the size and shape of devices, but they don't provide connectivity information. Similarly, commercial cell-based placement tools do not help the designer determine the best device size and shape.



**FIGURE 3.** The slicing structure and polar graph used in partitioning algorithms.

The Tektronix Graphical Floorplanner (GFP) attempts to span the range of tasks necessary for floorplan design. By modifying device specifications and rearranging placement, a GFP user can minimize wasted space in his floorplan. For example, area estimation and device planning allow the GFP to address early stages of a cell-based design to evaluate how layout affects subsequent design steps. Later, by creating and evaluating assemblies, the GFP facilitates partitioning. Finally, placement, channel editing, and layout evaluations help in refining the details.

Automatic methods used in solving floorplanning problems depend on a set of floorplan design concepts, specifically shape models, partitioning, initial floorplanning design, improvement floorplanning design, and estimation. In a floorplan, device sizes and shapes can be either fixed or flexible. Flexible devices can change according to *shape models* assigned to them. Two types of shape models are used: Mathematical models specify device constraints that do not evaluate device contents; procedural models derive size and shape from device contents. The mathematical models are fixed-in-one-dimension, constant-area, split-area, equivalent, and function-bounded.

A device that is fixed in one dimension can be modified in the other within certain constraints. Such a model is useful for stretching a block so that its pins match those of another block. For example, consider that block A in Figure 1 is fixed and block B is fixed only horizontally. Block B has been stretched vertically until its connections line up with those of block A. With their connections matching, the blocks can be placed close together, minimizing their interface area.

Constant-area devices can vary in aspect ratio, within limits. Such devices usually model an assembly that contains many devices. A split-area device is a set of assemblies in which total area is constant. The rows on blocks C and E and those blocks themselves represent split-area assemblies of standard cells.

Equivalent devices assume one of a finite number of discrete shapes. Such devices model custom blocks or cells that have several alternative layouts. For example, an equivalent block can be used to model a PLA that can be folded into several shapes. A function-bounded device is constrained by a general bounding function derived from the shape constraints of the constituent devices (Otten, 1987).

# System, system how will I know if

**S**imulating and testing large, multi-technology systems utilizing ASIC's always involved lots of guess-work. And a little magic.

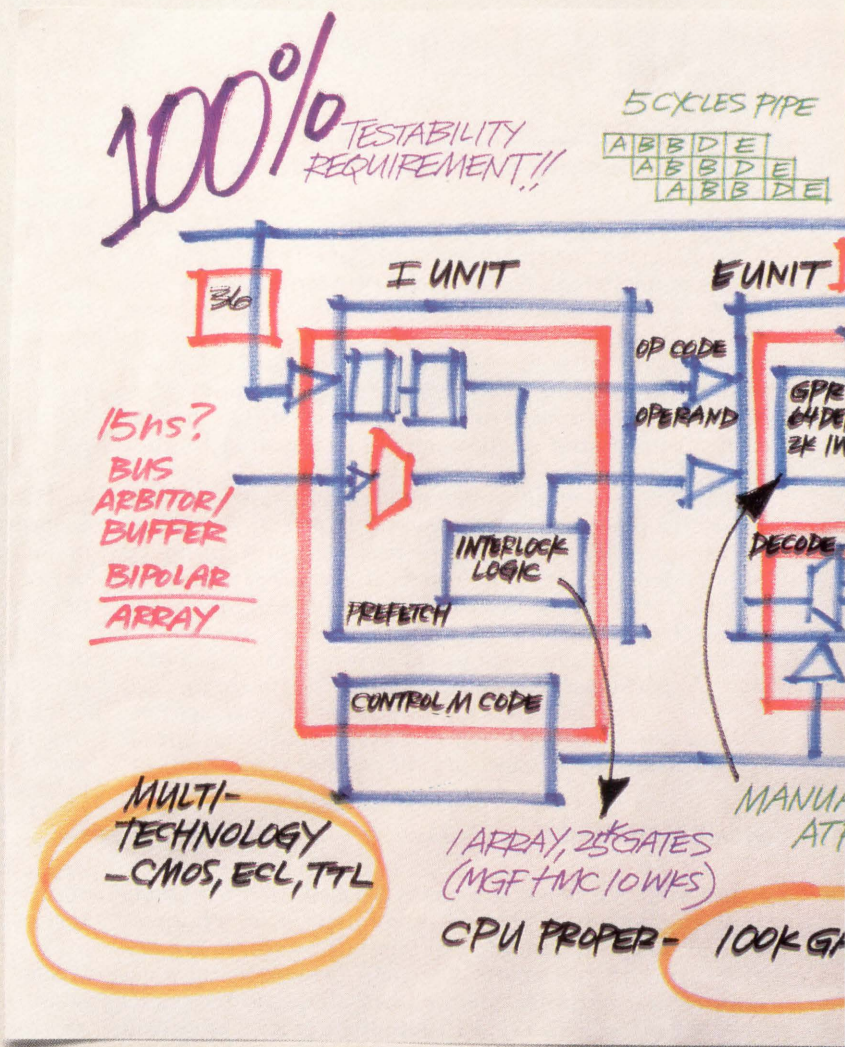
No more.

Now there's the AIDA Design System.

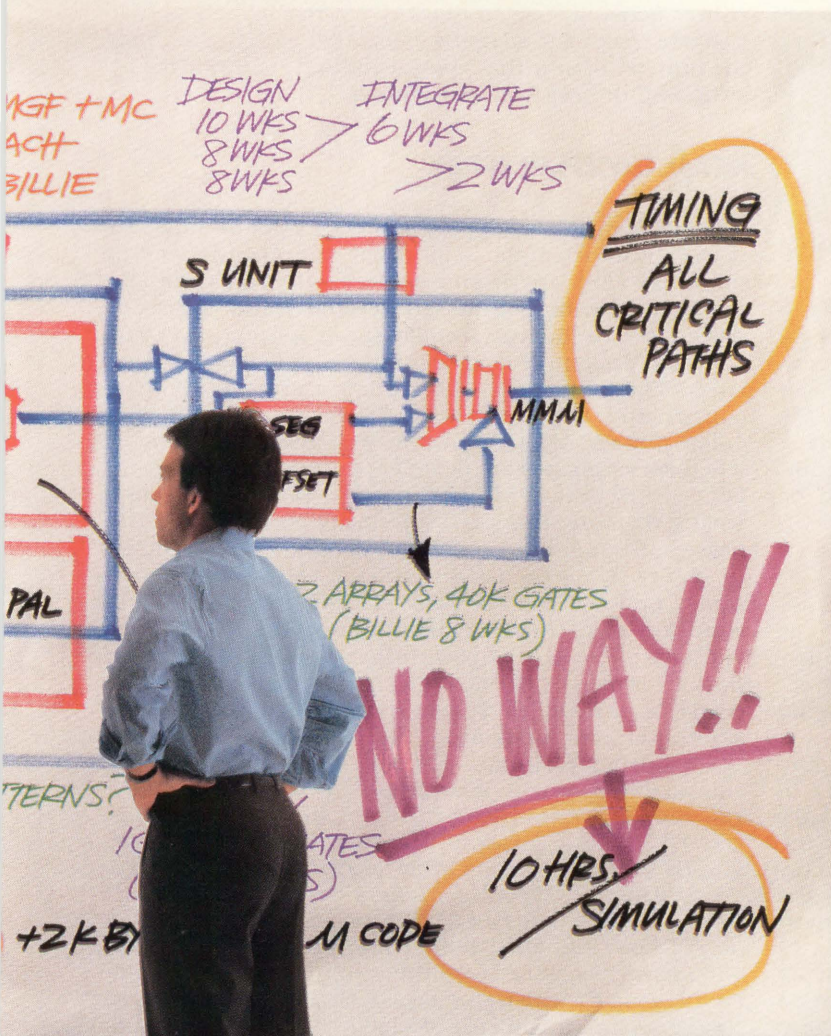
A comprehensive set of software design tools for capture, simulation and automatic generation of test patterns for systems of 5000 to 1 million gates. At all levels—chip, board, system.

Faster and more cost effective than ever before. Giving each designer 20 to 30 simulation turns a day. At one-tenth the cost of hardware accelerators.

The AIDA tool set is so powerful, it lets you do complete and automatic timing analysis of your design to identify all critical paths. And simulate its operation by running actual application programs and diagnostic software.



# m, on the wall, you'll work at all?



Even better, it guarantees 100% test coverage for scanable designs by automatically generating all possible test vectors.

Installation and training are easy, too. Because the AIDA Design System runs on industry-standard workstations. And we have translators that let you use your existing databases with the AIDA design tools.

Skeptical? Call 408-980-5200 now to get a video tape\* of what our customers are saying.

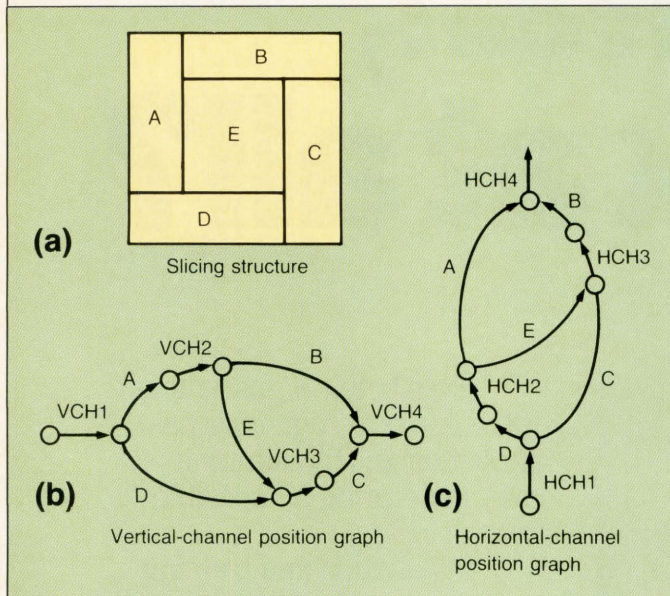
And see how we're helping them get systems off the wall and into production. Fast.

For more information, contact AIDA Corporation, 5155 Old Ironsides Drive, Santa Clara, CA 95054, 408-980-5200.

## AIDA

\*We reserve the right to qualify recipients.

AIDA is the trademark of AIDA Corporation. © 1987.



**FIGURE 4. A nonslicing structure (a) and its resulting vertical (b) and horizontal (c) channel position graphs.**

As opposed to the previous mathematical models, a procedural model specifies a device's size and shape by actually designing the device or by evaluating its contents. For example, a procedural design system can invoke module compilers during layout execution; the compilers design the block and return the shape, size, and pin locations to the design system (Allen et al., 1985). An area estimator for assemblies also can be considered a procedural model.

### **PARTITIONING A CELL-BASED DESIGN**

Partitioning is the process of creating a physical hierarchy of hardware elements in a design (Goto et al., 1986). Two types of partitioning are used: packaging and placement.

Package partitioning separates hardware elements to different boards, to different ICs on a board, and to different blocks within a hierarchical IC layout. Package assignments are usually reflected in the hierarchy of the design's schematic. For each package partition, a schematic system can generate a unique flat netlist for the package's layout. Package partitions have an I/O interface that defines their external connections.

Placement partitioning separates hardware elements to different placement areas within a board, within an IC, or within a packaged block in a hierarchical IC layout. There is typically one netlist for all devices within a package, regardless of placement partitions. The assignment of devices to placement partitions may be specified in the design schematic. An I/O interface for a placement partition is sometimes derived dynamically during automatic placement, but it usually does not appear in the design schematic.

Automatic-placement programs use placement partitioning to break a design into manageable parts. They

tend to create many levels in a design hierarchy, resulting in assemblies with a small number of devices (Dai et al., 1986). Designers can use placement partitioning to control the placement process, either through instructions on schematics or through interactive instructions during floorplan design. Typically, a user-defined placement partition comprises large placement areas with many devices and only a few levels of hierarchy.

Because the complexity of a partitioning problem grows rapidly with the number of devices, there is no optimal algorithm. Two types of heuristic partitioning algorithms are used most often: constructive and iterative improvement. Most constructive algorithms use some type of clustering approach. Improvement algorithms generally use an iterative approach such as that in the net-cut method (Schweikert et al., 1970), in which the number of nets that cross a cut line is minimized by swapping devices between groups.

### **INITIAL FLOORPLAN DESIGN**

Initial floorplan design begins with a set of unplaced devices and attempts to position and size devices to minimize wire length and empty space and eliminate overlaps. The algorithms addressing initial floorplan design are automatic, unlike those for improvement floorplan design, which generally are performed interactively. The four most popular algorithms are min-cut, constructive, Monte Carlo, and deterministic.

The min-cut algorithm works well in floorplan design when device shapes are flexible (Lauther, 1979; LaPotin et al., 1986; Wilk et al., 1986). In the basic algorithm, a slicing structure that partitions available layout space is derived in a top-down approach (Otten, 1982), as shown in Figure 2. As each slice is made, devices are partitioned between the resulting placement areas. Slicing continues until the structure's leaf nodes represent individual devices. At this point, the shape and location of flexible devices is defined. Using this method for fixed devices can result in wasted space or expansion of the placement area, however.

For such algorithms, relationships between blocks can be mapped by a polar graph (see Figure 3), which visually represents the partitioning of placement areas created by the algorithm. Slicing structures and polar graphs are employed by other algorithms as well.

Constructive algorithms take a bottom-up approach in which blocks are clustered into a hierarchy (Preas et al., 1979; Dai et al., 1986). This technique works best with fixed blocks because their shapes can be considered during the clustering decisions. For flexible blocks, the algorithm can process blocks in the hierarchy first to define their shapes, so that wasted area is minimized.

Clustering results in less regular placement areas than slicing structures and can save placement area. However, general clustered structures can create channel constraint cycles, in which no channel can be routed first. To model block relationships, channel position graphs (see Figure 4) replace polar graphs because they may be the only way to represent the relationships.

Monte Carlo techniques attempt to minimize both device overlap and net connectivity (Jepsen et al., 1983; Relis et al., 1986). Starting with a random, overlapping placement, the techniques change device position and shape randomly. Changes are accepted or rejected according to an objective function that evaluates each change in light of the desired result. Simulated annealing, a global optimization technique found in many CAD algorithms, often is applied to Monte Carlo algorithms.

Deterministic algorithms for floorplan design find the near-optimal center locations for all devices based on a connectivity function (Otten, 1982; Blanks, 1984; Woo et al., 1986). The blocks in this algorithm are overlapping and shape is not considered, so a subsequent algorithm must remove overlaps. Therefore, deterministic placement is used to start the floorplan design, and one of the other three algorithms is used for fitting the blocks. Interactive graphical floorplanning can even be used for block fitting.

### IMPROVEMENT FLOORPLAN DESIGN

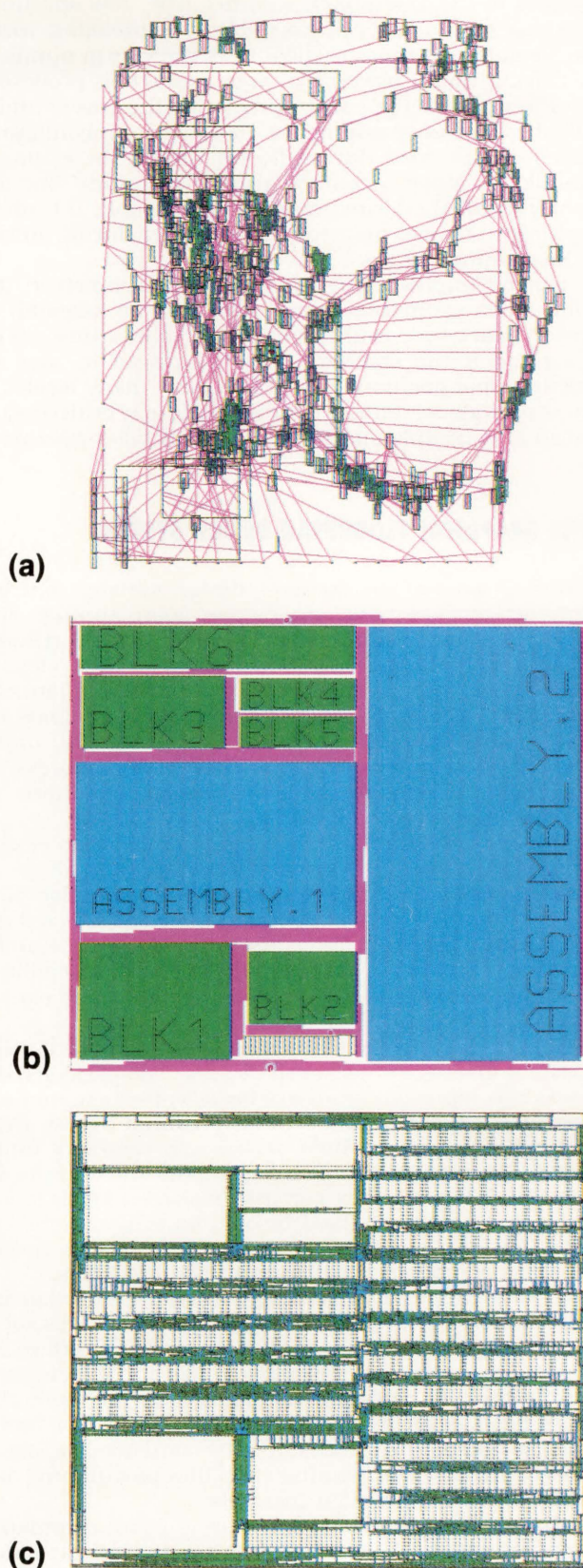
Improvement floorplan design optimizes the placement provided by the initial floorplan design. The same objectives—to minimize wire length and empty space—still apply. Several improvement techniques can be used interactively or automatically, including rotation, reflection, channel editing, and shape optimization. Detailed optimization occurs after an accurate estimation or routing step.

Rotation can optimize a layout without creating significant changes in the physical relationships between devices. The initial floorplan assigns one of four orthogonal orientations to each block, based on initial area estimates. In this case, rotation optimization itself yields little or no improvement (Lauther, 1979). A user can combine rotation with group swapping and block reshaping, however, for optimal results.

Like rotation, reflection about the  $x$  or  $y$  axis is a useful improvement technique because wire length can be reduced with no change in block layout. As a result, it is well applied in automatic algorithms. Reflection states can be optimized at any time during floorplan design.

Channel editing (McNeary et al., 1986) can optimize a layout by swapping a channel intersection from a horizontal feed to a vertical feed, or vice versa. It is particularly effective for eliminating wasted space resulting from routing expansion and for breaking nested channel cycles. For example, the *cut\_swap* command (Lauther, 1979; Woo et al., 1986) swaps the channel intersections between four flexible, constant-area devices, resulting in changed block shapes.

Shape optimization is an improvement technique used for both improvement and detailed floorplanning. One type, repartitioning, removes logic from one assembly and places it in another, thereby changing the



**FIGURE 5. Results of floorplan design: deterministic placement (a), floorplan (b), and final layout (c).**

shapes of the assemblies. Another type, row splitting, optimizes standard-cell assemblies by breaking them into smaller assemblies whose rows can vary in number.

During initial placement, area estimation programs give feedback to the user by predicting the area of undefined carriers and assemblies. Because many configurations can be tried during floorplanning, the estimate should be quick. Accuracy also is important, but because detailed information rarely is available, it is often sufficient to use only rough estimates during initial floorplanning stages.

During improvement floorplan design, however, detailed area estimation is necessary. For an assembly of cells, an accurate estimate of an assembly's area can be obtained if the standard cells are placed in final or almost final position. For layouts with many blocks, a first-pass global route or detailed route may be necessary to identify floorplan improvements that will reduce area.

## THE GRAPHICAL FLOORPLAN DESIGN SYSTEM

GFP, as part of the Merlyn-S design system, controls interactive placement, interactive floorplanning, and the execution of automatic floorplanning algorithms. Display commands allow users to tailor their view of assemblies, rows, connections, pins, barriers, channels, and force vectors. Interactive placement commands include *place/move*, *unplace*, *rotate*, *mirror*, *fix*, *unfix*, *make\_row*, *save*, and *restore*. These commands can be applied to assemblies to achieve such operations as moving a group of devices.

A user works through the interactive floorplan design interface to specify either a device's size and shape or shape models for a device used in automatic floorplan algorithms. The type of floorplan interface selected depends on where the device fits in the design hierarchy shown in Figure 2. Designers edit leaf cells, assemblies, and the carrier using the device, assembly, and carrier floorplan design interfaces.

The device interface can manipulate standard cells, custom macros, or packaged devices. The packaged devices can represent groups of devices; because they are packaged, however, the interface cannot access their constituent devices. Such devices can represent undefined logic circuits in which only inputs and outputs are specified. The interface can assign shape models to leaf cells that are fixed, fixed in one dimension, constant area, or equivalent. In addition, the user can specify device size and change values in shape models.

The assembly floorplan design interface is used in any application that requires a grouping of devices; as such, its function is similar to that of the carrier interface. As the user manipulates assemblies, size and net crossing information is updated. He can flatten the assembly, package it as a leaf cell for hierarchical layout, or save it for use during the placement of its contents. The use of assemblies in floorplanning simplifies partitioning and adherence to the design hierarchy.

The assembly floorplan interface includes commands not found in the device interface; it controls the creation of assemblies and the addition or removal of logic from

an assembly. Users can split assemblies, and the interface supplies both local and total estimates for them. Area estimation tools are applied locally to an assembly that evaluates the current placed or unplaced contents. Netlist information can be entered for batch floorplanning or to initialize the floorplan design system. With forward and backward annotation, the schematic can act as an archive for floorplan information.

## Automatic-Algorithm Interface

GFP's automatic algorithms can be invoked interactively through the GFP interface. Interactive execution gives the user more flexibility and speed in creating and viewing floorplan alternatives. Because many algorithms execute very quickly, an interactive interface allows the user to view results more quickly than if the algorithms execute in a batch mode. Also, a user can interactively define a portion of his design for execution, thereby reducing execution time. He can specify constraints on devices, such as prepositioning. Finally, he can execute algorithms in steps so that he can modify the floorplan between execution steps. In short, executing automatic algorithms interactively results in a more flexible and user-friendly interface than using a complex control scheme for batch execution of an algorithm.

The automatic algorithms found in GFP's device, carrier, and assembly floorplan design interfaces include deterministic, fit, and improvement placement; channel definition and global routing; and cell design algorithms for area estimation and row outline generation.

The automatic placement algorithms operate on the entire carrier or on individual assemblies. The deterministic placement algorithm calculates the equilibrium center location of devices, which can be in predefined positions. At this point, devices are clustered into assemblies or fit into the current carrier. To eliminate overlapping, devices can be fit with interactive placement commands or by an automatic-fit program.

Various automatic algorithms are invoked implicitly during the execution of some interactive floorplan design commands. For example, standard-cell devices can be clustered interactively into assemblies using the assembly-floorplan interface. During clustering, automatic algorithms in the interface continuously provide area estimation and net-crossing information.

The final automatic algorithms manipulate routing channels. Channel definition interactively defines and orders channels, displaying them with the cycles identified. Channel editing consists of *swap\_intersection* and *cut\_swap* commands. The global routing command sizes channels and spreads devices to ensure routability.

## GFP in Placement and Packaging

The following example illustrates how, through GFP, a user controls the layout of a cell-based design. In Figure 5, a cell-based design is shown after deterministic placement (a), after interactive floorplan design (b), and after final layout (c). The design contains 1020 standard cells, 6 macro blocks, and prespecified I/O positions. This lay-



out is simple enough to perform automatically, but some manual control will almost always improve layout.

The user begins layout by generating a deterministic placement of all components. This interactive step takes about 20 CPU-seconds on a Digital Equipment Corp. VAXstation II GPX. This initial placement shows the relative locations of the macros. The user then groups adjacent macros, keeping in mind their geometries. A second deterministic placement, whose output appears in Figure 5a, rearranges the groups to get this near-optimal connectivity of the design. The algorithm has also spread the devices a little so that groups are identified more easily.

The predefined I/O positions form an I/O frame that rings the perimeter in the figure. Because some devices extend beyond the I/O frame, the display is expanded around all devices.

To determine if these macro positions are acceptable, the user must get an accurate estimate of the layout. The floorplan display in Figure 5b shows the six macros (in green) and two standard-cell assemblies (in blue). The assemblies, which resulted from clustering after the deterministic placement, conform to desired aspect ratios and other constraints. The floorplan interface updates assembly estimates continuously during floorplanning, even though its constituent devices are not placed completely. In this example, the standard cells are placed to yield the estimated sizes shown; an estimated routing area completes the floorplan (Figure 5b). The final detailed layout steps complete the layout (Figure 5c).

## CONCLUSION

GFP provides a user interface for interactive and automatic floorplan design of cell-based VLSI chips. GFP can be used in the initial phases of design for partitioning and area estimation of individual assemblies or overall layout. It also supports the optimization of cell-based designs during layout. It assists in package partitioning of VLSI designs to simultaneously optimize pin assignments and device placement, both within the IC and on the circuit board. GFP attempts to provide an interface similar to that of a PCB design system while providing the floorplan design and automatic algorithms necessary for VLSI design complexity. □

## ACKNOWLEDGMENTS

I wish to acknowledge the contributions to the research and development of GFP of the following individuals: Tom Freuler and John Blanks for automatic-algorithm design; Bryan Preas and Steve King for discussions; Ed Downs and Kirti Parmar for display and interactive-placement design; and Nelson Brady, Deborah Cobb, and Pat Karger for technical comments and careful review of this article.

## REFERENCES

Allen, P.E., et al. May 1985. "AIDE2: An Automated Analog IC Design System," *Custom Integrated Circuits Conference*, Portland, OR.

Blanks, J.P. 1984. "Initial Placement of Gate Arrays using Least-Squares Methods," *21st Design Automation Conference*, Albuquerque, NM.

Dai, W.M., et al. 1986. "Hierarchical Floor Planning for Building Block Layout," *International Conference on Computer-Aided Design*, Santa Clara, CA.

DuPont, R.A., et al. 1986. "ROM/MMU Circuit Technology and Chip Design," *IBM PC RT Technology Journal*.

Goto, S., et al. 1986. "Partitioning Assignment and Placement," in *Advances in CAD for VLSI Design*, Elsevier Science Publishers, New York, NY.

Heller, W.R. 1981. "Constraints in Physical Design between LSI and VLSI," *18th Design Automation Conference*, Nashville, TN.

Jepsen, D.W., et al. 1983. "Macro Placement by Monte Carlo Annealing," *IEEE International Conference on Computer Design*, Port Chester, NY.

LaPotin, D., et al. October 1986. "Mason: A Global Floorplanning Approach for VLSI Design," *IEEE Transactions on CAD of Integrated Circuits and Systems*.

Lauther, U. 1979. "A Mincut Placement Algorithm for General Cell Assemblies based on a Graph Representation," *16th Design Automation Conference*, San Diego, CA.

McNeary, S., et al. November 1986. "VITAL: A Cell-Based ASIC Assembler," *VLSI Systems Design*.

Otten, R.H.J.M. 1982. "Automatic Floorplan Design," *19th Design Automation Conference*, Las Vegas, NV.

Otten, R.H.J.M. 1987. "Annealing applied to Floorplan Design," *Advanced Semiconductor Technology and Computer Systems*, Van Nostrand-Reinhold.

Preas, B.T., et al. 1978. "Methods for Hierarchical Layout of Custom LSI Circuit Masks," *15th Design Automation Conference*, Las Vegas, NV.

Preas, B.T., et al. 1979. "Placement Algorithms for Arbitrarily Shaped Blocks," *16th Design Automation Conference*, San Diego, CA.

Relis, Y., et al. 1986. "CRAFT: A Customizable Refinable Automatic Floorplanning Tool," *IEEE International Conference on Computer-Aided Design*, Santa Clara, CA.

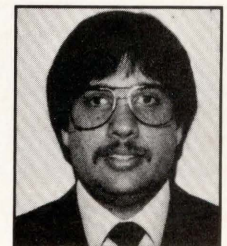
Schweikert, D.G., et al. 1970. "A Proper Model for the Partitioning of Electrical Circuits," *7th Design Automation Workshop*, San Francisco, CA.

Wilk, A., et al. 1986. "VLSI Constraint-Driven Layout System," *IEEE International Conference on Computer-Aided Design*, Santa Clara, CA.

Woo, L.S., et al. August 1986. "PIONEER: A Macro-Based Floor-Planning Design System," *VLSI Systems Design*.

## ABOUT THE AUTHOR

Edmond Macaluso is a senior engineer for the Tektronix CAE Systems Division, where he has worked on automatic routing, CAD integration, and graphical placement. He joined the division in 1982 when it was VR Information Systems. Prior to that, he worked in the field of analog circuit design and silicon compiler tools. Edmond earned his BS and MS degrees in electrical engineering at Texas A&M University.



# A RULES-DRIVEN APPROACH TO CIRCUIT BOARD DESIGN

Joseph Prang and Katherine Gambino, Valid Logic Systems Inc., San Jose, CA

In the traditional systems design cycle, the CAE engineer designs the electrical circuit and the CAD engineer designs the physical layout. As a consequence, layout tools are concerned with the *physical* characteristics of the design and frequently have no means for handling *electrical* considerations. This discrepancy inevitably leads to difficulties. A design engineer could often prevent physical design problems if there were not a sharp break between the electrical and the physical

design of a product (see Figure 1).

The break in the design cycle occurs when schematic entry and design verification are completed and the design is transmitted to a physical layout specialist using a netlist. The netlist defines functional components and their connectivity. The problem is that the netlist defines *nothing else* besides the components and their connectivity. It does not define the functional groupings of components. It does not define which nets make up critical paths and should receive special treatment during placement and routing. Without any other stated direction, PCB designers are driven to achieve tight component placement and 100% routing of their boards. The electrical requirements of the design either are not documented or must take second priority.

The objective of "rules-driven" design is to capture implementation rules with their schematics, along with components and connectivity, and have those rules implemented by downstream layout design tools (as shown in Figure 1).

Although our description of rules-driven design focuses on the electrical engineer's rules and how they are passed on to the layout designer, the methodology can be expanded to include other functional disciplines within the life cycle of a product.

## WHO CONTROLS DESIGN?

How rules-driven design works can best be illustrated with a typical design example. Consider a single-board computer with the power of a small mainframe, which must meet tight specifications within a restricted schedule (see Figure 2). The dashed lines identify related functions that should be grouped together on the board. We will assume that the design uses one of the new 32-bit microprocessors and that each block represents dozens of components. The use of hierarchical design implementation permits several pages of schematics to be captured in one diagram while maintaining design integrity.

For the required performance, a high-speed clock circuit must be implemented in ECL rather than TTL. Choosing ECL may be correct from a functional point of view,

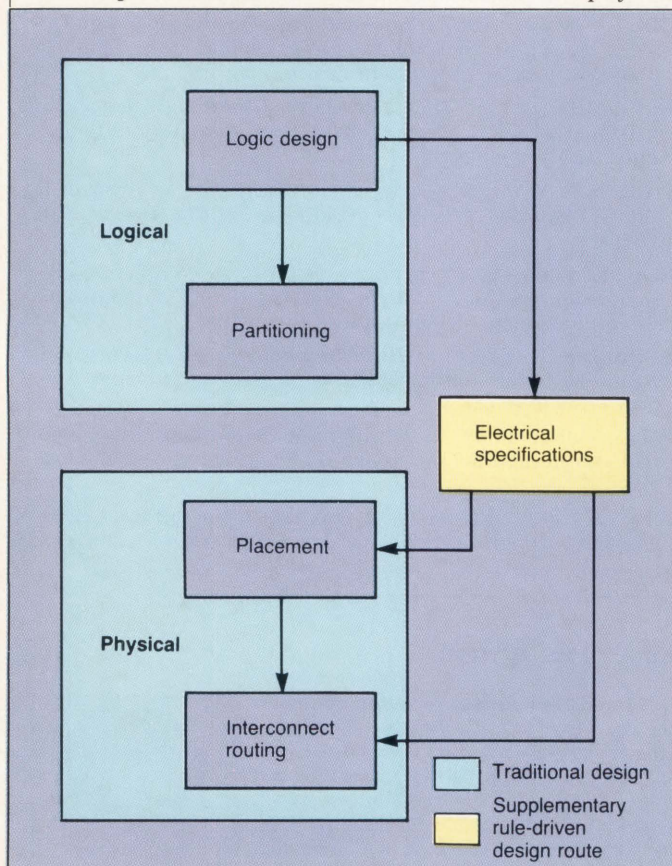


FIGURE 1. In rules-driven design, the traditional design flow is supplemented by electrical parameters.

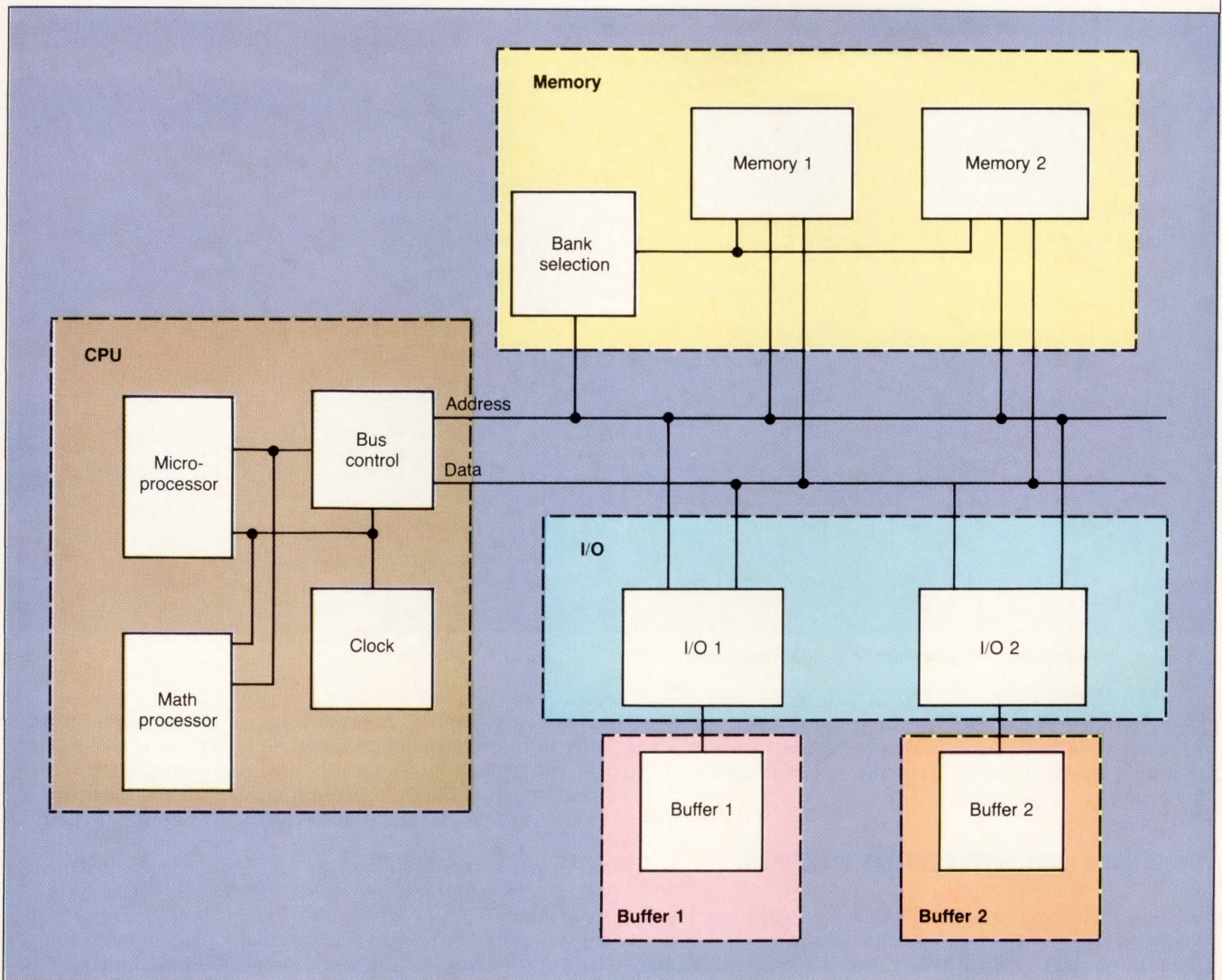


FIGURE 2. A design example showing "rooms."

but with ECL's greater susceptibility to reflections, cross-talk, and heat, what effect does it have on the physical layout, manufacturing, test, and reliability of the design?

In addition, surface-mount technology (SMT) is used for this project, because a 2-foot-square board is out of the question. But SMT has manufacturability and testability implications in PCB design that do not occur with TTL DIPs mounted in through-holes on a board.

Further, the company's test engineers and field service engineers would like related components near to each other on a board. If all the CPU circuitry is in one area, all the memory chips in another, and all the I/O in another, the boards would be much easier to understand and to service.

Test engineers are not impressed when they are told that the PCB CAD system laid the boards out that way. The natural response is, "Who controls the design—the engineers or the CAD systems?"

With a rule-based system, engineers can specify rules to downstream systems by attaching properties (or "directives") to components and nets in their schematics (see Figure 3). Rules-driven design is implemented in two physical design tools called COMPOSE and ALLEGRO. COMPOSE is for IC layout; ALLEGRO is for PCB layout. Both allow the engineer to specify implementation rules and functionality to the IC or PCB designer who will perform the physical layout.

Each directive consists of a prefix, either "PACKAGE" or "NET," and a suffix that identifies the specific rule to be applied. Package directives are attached to components and net directives are attached to nets. A directive may be qualified by a value. Figure 3a illustrates how directives are attached to a schematic. With the graphics editor, the "PROPERTY" command is selected from the menu, and the net selected from the schematic using a mouse.

Since directives can clutter the schematic, the engi-

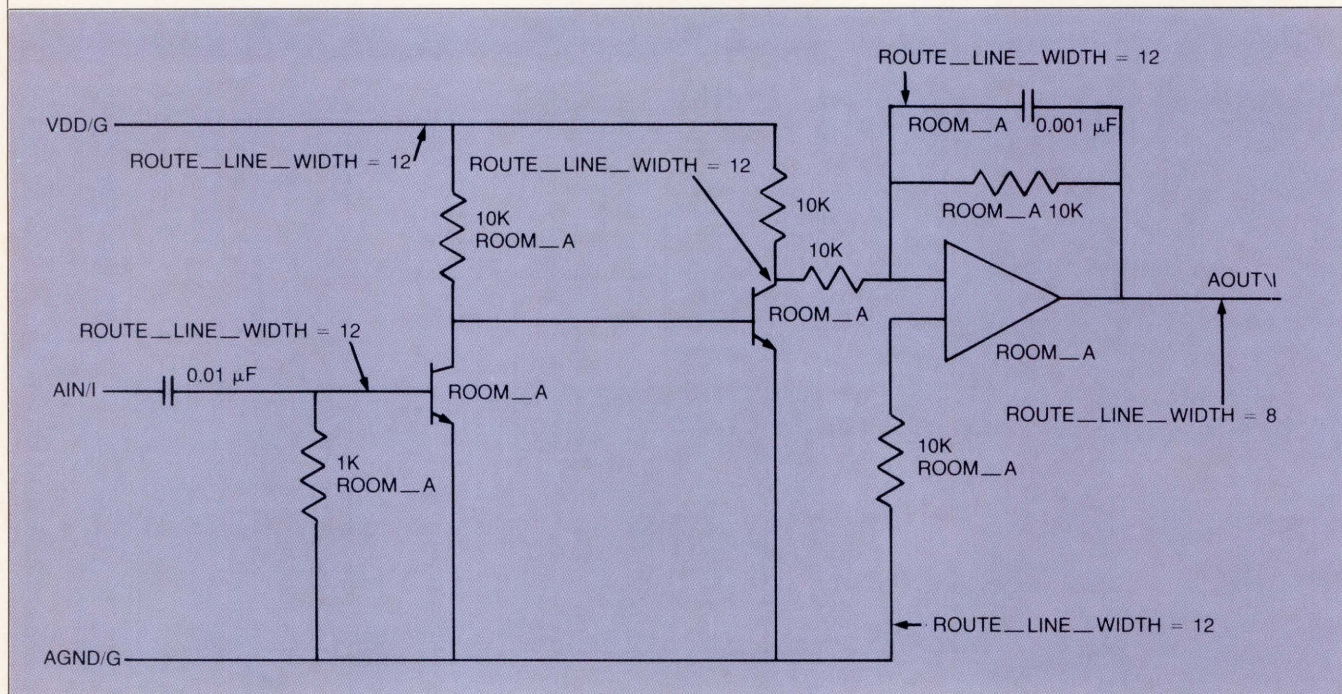


FIGURE 3. Properties are attached to components and nets.

neer can suppress the "PACKAGE" and "NET" prefixes. Alternatively, the property can be suppressed from the display entirely, in order to facilitate examination of the schematic.

### DIRECTIVES THAT AFFECT INITIAL PLACEMENT

There are three essential phases to physical layout: floorplanning, placement, and routing. In ALLEGRO, some directives control the initial placement and handling of components and groups of components in a design. The following are currently in use for this purpose:

- PACKAGE\_\_ROOM (name)
- NET\_\_WEIGHT (value)
- NET\_\_ECL

The "ROOM" directive is used to place all components with the same group name together on a board. The rooms specified by the designer may be segregated by circuit content, package type, thermal considerations, or height restrictions.

Of these considerations, logical grouping is possibly the most important to the CAE designer. It is highly desirable to maintain logical groups in the physical placement patterns for a number of reasons. First, it makes automatic placement extremely fast, because the general grouping decisions have already been made, thus drastically cutting down on the number of possible layout patterns. Second, the board is better electrically, because critical paths (as defined by the engineer) are shorter, resulting in less crosstalk and line reflection.

Without grouping, floorplanning is based on the number of interconnects, not criticality. Third, it makes the board easier to understand, and therefore easier to test and repair. Fourth, CAE engineers are often assigned a subcircuit to design that becomes a room in the board floorplan.

In the example shown in Figure 2, the dashed lines enclosing groups of blocks identify logically related functions that should ideally be grouped together when the board is laid out. Most PCB automatic placement systems use a constructive initial placement algorithm that has no concept of logical groupings. For that reason, many experienced PCB designers refuse to use them, preferring instead to place components interactively. That gives the designers the ability to maintain logical groups and achieve placements that will be highly routable by their autorouter.

Assigning logical functions to specific areas of the board is called floorplanning. With our system, floorplanning is performed before a diagram of how the board should be divided is included. Floorplanning speeds automatic placement of components on a circuit board for one very simple reason: There are fewer components that the autoplacer must consider at any given time. For example, with a 20-component board, there are 20! (or  $2.4 \times 10^{18}$ ) possible component arrangements. If the board is divided into four rooms and five components in each room, the number of combinations then becomes  $5! \times 4$ , or 480.

Floorplanning thus reduces the number of possible combinations by almost 16 orders of magnitude. Of course, the complexity of the problem is reduced, but other factors enter into the actual performance of the



## *In the race to market, GenRad's HILO gives the smooth hand-off you need from your ASIC design to your board design.*

Getting your new product to market faster than your competition means added profits for a longer time. And that's what HILO® is all about.

The HILO Universal Logic Simulator helps you design quality into your ASICs fast. And you'll find many leading ASIC foundries using HILO in their in-house simulation.

This means your ASIC design via HILO is the ideal connection with the foundry's system. The result is faster production of high-quality custom and semi-custom devices.

And since HILO can feed forward to your board

design, your HILO-designed devices move quickly onto your HILO-designed pc boards where they are complemented by the GenRad HICHIP™ physical device modeler.

Think about it. You'll get quality designed devices and boards in less time. And your finished product will get to market faster. Call 1-800-4-GenRad to find out more about HILO.



**GenRad**

placement algorithms. In an actual benchmark, automatic placement of a board containing 150 components took over 12 hours without floorplanning; ALLEGRO's placement time was three minutes for the same board—an improvement factor of 240.

In addition to assigning components to a room, the priority of component placement within each room also can be controlled. To do this, the "WEIGHT" property is employed, which specifies the order in which components will be placed within a room. For example, in the CPU room the bus between the microprocessor and the math coprocessor may be given the highest weight. The software will then position the microprocessor and the math coprocessor to optimize this net. Thus the system can optimize component positions within a room or emphasize connections between rooms. If no weight is assigned to other nets in that room, the software is free to place them as it sees fit.

The PCB designer can work from the rules and the floorplan to optimize layout (see Figure 4). Rooms can be treated almost like small circuit boards in their own right. For example, the designer can set up individual placement grids for each room to optimize them for the types of components that they will contain (axial, DIP, SMD, and so on), define boundaries between rooms as hard or soft to cover any overlap, and define component locations (pin 1 or body center) to accommodate downstream CAM equipment.

Some parts in some rooms will have ECL nets. These nets have special routing rules that actually affect the initial placement in ALLEGRO, which we do not discuss at this point.

## PLACEMENT IMPROVEMENT

Initial placement is governed by overall considerations that result in a first approximation of the ideal placement. Today's layout systems employ swapping techniques to make routing easier. If package A and package B are two identical packages that are located on different parts of the board, it is possible to swap gates within a package, swap gates between packages, and even swap complete packages in order to simplify routing. The following directives determine the extent to which later changes in the original placement can be made without violating the designer's intent:

- PACKAGE\_FIX\_ALL
- PACKAGE\_NO\_SWAP\_COMP
- PACKAGE\_NO\_SWAP\_GATE\_EXTERNAL
- PACKAGE\_NO\_SWAP\_GATE
- PACKAGE\_NO\_SWAP\_PIN

The directives that prohibit movement of elements from their original positions can affect component, gate, and pin positions. "NO\_SWAP\_COMPONENT" prohibits the swapping of that component for any other component. The component swap operation provides the fundamental mechanism of placement improvement; if the components are initially assigned to rooms, it is unlikely that the logical arrangement will be so bad that swaps out of the original rooms will improve routing. Thus logical

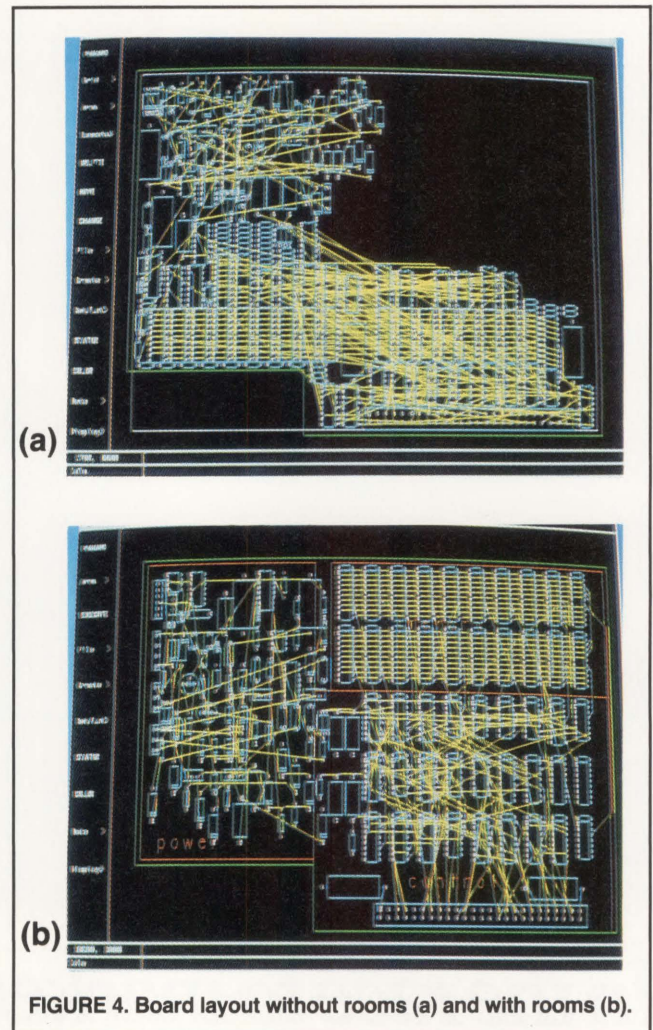


FIGURE 4. Board layout without rooms (a) and with rooms (b).

grouping is usually maintained even when this directive is not invoked. But this directive is useful, for example, when a connector must be in a particular location in the middle of the board.

"NO\_SWAP\_GATE" prohibits any swapping of gates for the gate assigned with that directive. This directive is included for reasons of symmetry and is rarely important to the CAE designer. "NO\_SWAP\_GATE\_EXT," on the other hand, can be used when gates can be moved around inside a component but should not be swapped out of the component. This directive will keep all the circuitry associated with a particular function in the same area of the board and is especially useful when there are a large number of SSI functions.

"NO\_SWAP\_PIN" prohibits swapping of equivalent pins. The "NO\_SWAP\_PIN" directive is useful for preassigned pins on connectors and for prerouted nets of matched lengths.

"FIX\_ALL" fixes all attributes of one part. No component, gate, or pin swapping is allowed during routing for that one component. The part will appear in the exact rooms that the designer specifies, with the original pin



## *“CASE Technology: CAE Solutions Planned Right from the Start”*

CASE Technology's new Vanguard CAE Design System supports a full range of industry standard hardware platforms — DEC VAX (VMS), Sun (UNIX) and PC (DOS) — and a comprehensive set of electronic design applications for PCB and ASIC design. Applications include schematic capture, digital logic simulation, circuit simulation and PCB design capabilities.

Since initial product introductions four years ago, CASE Technology has been setting trends in the computer-aided engineering

industry. CASE was the first to introduce:

- a PC-based CAE solution in June 1983
- an integrated PC to VAX solution in November 1985
- a PC as an intelligent graphics terminal for CAE in February 1986
- a complete Sun Workstation-based CAE solution in October 1986

In 1986, CASE also announced major marketing agreements with Digital Equipment Corporation and Sun Microsystems for the joint promotion of the Vanguard CAE Design System on the VAXstation and Sun-3 series of high performance engineering workstations.

Over the last three years, CASE Technology has experienced explosive growth to the rate of 80 percent per year and has remained profitable every quarter since the first product shipped. Today, CASE has over 3000 installations of the Vanguard CAE Design System worldwide, in companies

such as Hughes Aircraft, Honeywell, and Rockwell International.

Why has CASE Technology been so successful? It's simple. We listen to our customers very carefully. Corporate and engineering managers want *solutions* that work. CASE provides electronic design solutions through a well-conceived, long term plan for product migration, an open database philosophy, and data and operating system independence.

Before you make a decision on CAE, take the time to see what CASE has to offer.

CASE Technology, Inc., 2141 Landings Drive,  
Mountain View, California 94043  
Phone (415) 962-1440; Telex 506513;  
FAX (415) 962-1466.

**CASE  
TECHNOLOGY**

**CIRCLE NUMBER 17**

assignments. This command is useful when design issues are so critical that the initial placement should not be changed at all.

## ROUTING CONTROL

Once packages and gates are established, the next step is to route the actual traces. Designs can contain a mixture of high-frequency nets and less critical nets. Unfortunately, most PCB routers cannot tell the difference. With the rules-driven system, critical nets can be identified at the schematic level, and the router understands exactly how they are to be handled. All directives that control routing are net directives. Currently, ALLEGRO supports the following net directives:

- NO\_\_ROUTE
- ROUTE\_\_PRIORITY (*value*)
- ECL
- STUB\_\_LENGTH (*value*)
- DRIVER\_\_TERM\_\_VAL (*value*)
- LOAD\_\_TERM\_\_VAL (*value*)
- FIXED
- NO\_\_RIPUP
- ROUTE\_\_LINE\_\_WIDTH (*value*)
- NO\_\_PIN\_\_ESCAPE

"NO\_\_ROUTE" excludes a net from the autorouting process. This directive can be used when a net is so critical with respect to its placement or length that the engineer prefers it to be laid out manually.

"ROUTE\_\_PRIORITY" specifies the order in which nets should be routed. More critical nets should be routed earlier, as they will then be shorter and have fewer or no vias.

"ECL" means that a net is a high-frequency net and is to be treated as such. Forty-five-degree routing, daisy-chain connections, loads and terminators, and dynamic ECL rat's-nesting are employed.

Other directives assist in the specification of ECL nets. For example, "DRIVER\_\_TERM\_\_VAL (*value*)" specifies the value of the terminating resistor on the drive end of an "ECL NET"; "LOAD\_\_TERM\_\_VAL (*value*)" specifies the value of the terminating resistor on the load end of an "ECL NET."

"STUB\_\_LENGTH" specifies the maximum allowable stub length for a net. The risk of noise reflection determines the length of the allowable stub: nets with a higher fan-out are more likely to be susceptible to reflections and should be given correspondingly shorter stubs. Higher stub length values allow there to be longer stubs. If the value is zero, no stubs are allowed at all, resulting in a daisy-chain connection.

"FIXED" prohibits any change to a net once routed.

Rip-up-and-reroute is a legitimate routing technique that is often needed to achieve 100% routing. However, when a trace is ripped up, the new trace will usually be longer than the original. If this is unacceptable, the property "NO\_\_RIPUP" can be employed.

"ROUTE\_\_LINE\_\_WIDTH" specifies a trace width other than the standard width to be assigned to a net. Various line widths may be required to match the power draw of

specific circuits. This directive is useful for power lines and some analog lines that may be wider than signal lines.

"NO\_\_PIN\_\_ESCAPE" prohibits the routing of a net to any "pin escapes." In surface-mount designs, some nets must be routed only on the top or bottom layers. This directive prohibits the use of a via as an "escape" mechanism to the inner layers.

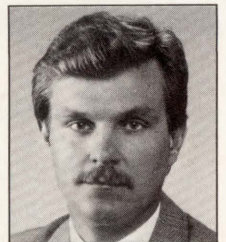
## IMPLEMENTING RULES-DRIVEN DESIGN

Creating a design within a rules-driven approach means creating more than a schematic during design entry. The CAE engineer is creating the implementation rules as well as the functional definition of his design. Rules-driven design can be expanded to include all phases of the design cycle and all design disciplines. In the future, we envisage a completely computerized description of the design and implementation rules. For instance, test engineers can work with design engineers to specify signal accessibility in multilayer boards for ATE. Manufacturability rules to prioritize components for automatic insertion can be included. Additionally, chip design can be expanded to include hybrids.

Rules-driven design is thus a significant tool for the design engineer today and could well be a major stepping stone to the integration of the design cycle with the total product life cycle. Although an integrated database makes engineering changes easier, the real impact is on time to market. Specifying the rules and developing tools that automatically respond to them is the fastest way to reduce the CAD cycle time. □

## ABOUT THE AUTHORS

**Joseph Prang** is vice president of product marketing for Valid Logic Systems. He was product marketing manager at Telesis prior to that company's merger with Valid in early 1987. Before that, Prang spent seven years with GenRad as a marketing engineer, marketing manager, and product line manager. He holds BSEE and MBA degrees from Purdue University and a patent on the "beyond-the-node" diagnostic system.

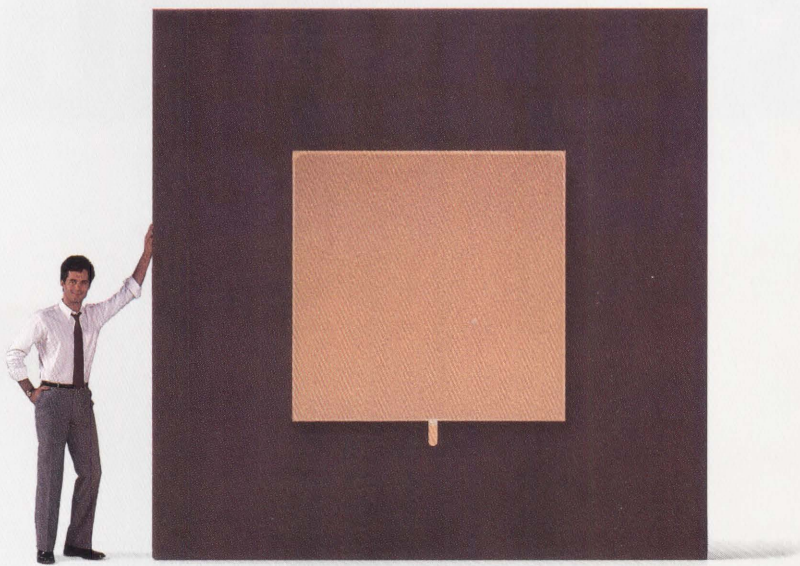


**Katherine Gambino** is product marketing manager for Valid's PCB Division, based in Chelmsford, MA. She has been with the company for three years and previously spent five years in marketing support of CAD products at Applicon. She holds a BA from Barnard College, New York, NY.





**A LOT OF  
PEOPLE  
THINK WE  
ONLY HANDLE  
GIANT PROJECTS.**



*The entire state of New Jersey on a chip.*

**EVEN THOUGH  
WE OFFER  
100,000 GATES,  
YOU DON'T HAVE TO  
USE THEM ALL.**

It's no wonder people expect big things from LSI Logic.

We're the largest HCMOS ASIC company there is.

But that doesn't mean you have to be the largest in your field to come to us.

You just need to be doing ASIC. No matter if it's a few hundred gates or a complete system on a chip. Or even a multi-ASIC system.

It's plain and simple, really: We understand your ASIC might not need 100,000 gates today. But it's good to know it's here when you need it.

To simplify your ASIC design task, we have the largest library available of SSI and MSI building blocks. And more than 400 industry-standard LSI and VLSI building blocks for Channeled and Channel-Free™ Arrays, and Cell-Based designs.

So whether you need an array or cell-based product, you're covered.

If you change methodologies, you don't have to re-design or change vendors. Or worse, change design tools. Simply, our design tools support both.

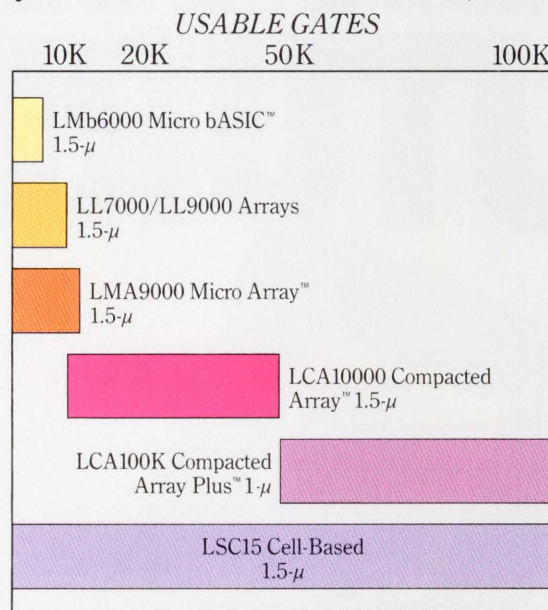
There's something else you won't get anywhere else. Our turn-around for prototypes.

It's only two to three weeks for arrays and five to six weeks for cell-based prototypes. All fully tested and guaranteed to work to your specifications.

If you need your arrays even sooner, we can deliver fully tested prototypes in just seven

days. And when cost is an issue, we have a whole range of cost-effective solutions, too.

So don't worry about how big or small your ASIC need is. We're the right size for you.



LSI Logic's family of products covers the entire ASIC spectrum.

**WE HAVE  
SEEN THE  
FUTURE.**



*Mini  
on a chip.*

**AND IT IS  
VERY TINY.**

Saying our new Modular Design Environment (MDE)<sup>™</sup> is the most advanced ASIC design software anywhere isn't small talk.

Using MDE, we've already accurately designed and simulated systems with more than two million gates of logic. And that's just for starters.

Our software easily migrates into the latest technology. So your design will always have the highest performance and densities.

MDE is actually comprised of three modules.

The *Logic Integrator*<sup>™</sup> is an entry-level module containing the design and simulation tools for building single ASIC chip designs.

The *Silicon Integrator*<sup>™</sup> module handles the design and simulation of complex ASICs ranging from a few hundred to 100,000 usable gates. Its Silicon Compilers allow you to automatically develop logic and memory. Your compiled designs, of course, all have complete simulation and test vectors.



*MDE's interactive graphics deliver fast, flawless design and simulation.*

You can also effortlessly convert PALs to arrays with our Logic Synthesizer.

Our *System Integrator*<sup>™</sup> module has mixed-mode behavioral and gate-level simulation capabilities. Use it to design your entire system, including multiple ASICs and standard components.

All with surprising ease and economy. So you can "electronically breadboard" your complete system before going to prototype.

And you can design your ASICs on more platforms than anywhere else. Like all the popular workstation and mainframe environments. Or commercial CAE systems through our CAD

Connection Program. Or at one of our 24 Design Resource Centers—the world's largest ASIC support network.

MDE is also tightly coupled to our worldwide manufacturing facilities. Which is why LSI Logic delivers working parts 100% of the time. Guaranteed.

And why you'll see your future a lot sooner with us.

With more than 4,000 working designs under our belt, one clear fact emerges.

Our system works.

That's true whether your needs call for an entry-level gate array or cell-based design. Or for the world's most advanced ASIC-based system.

Whatever direction your designs take, you can always use the same proven software: LSI Logic's MDE.

MDE can be tailored to your specific situation. Just select the appropriate design modules.

We have all the performance you'll need. Such as ECL-like speed in 1 or 1.5-micron HCMOS technology with gate delays of 460 picoseconds. And even more coming soon.

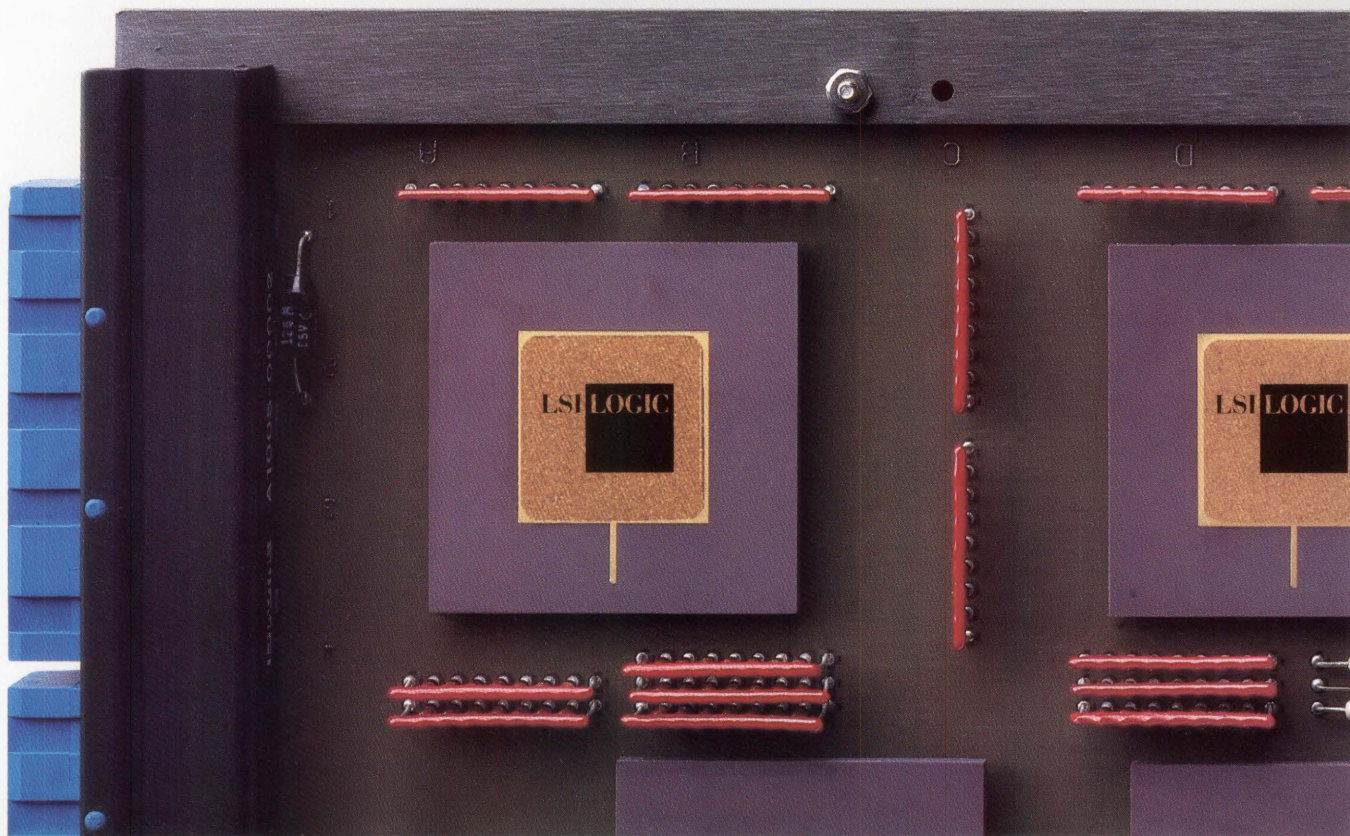
We also have more than 230 package options with up to 299 pins. Including a full range of plastic, ceramic pin grid arrays and chip carriers.

Just as important as our high level of technology is our high level of service. Choose as much as you need. You can design at your own workstation. Or at one of our Design Resource Centers. Or we'll do the entire design for you.

What's most important is that our relationship with you works as well as your device.

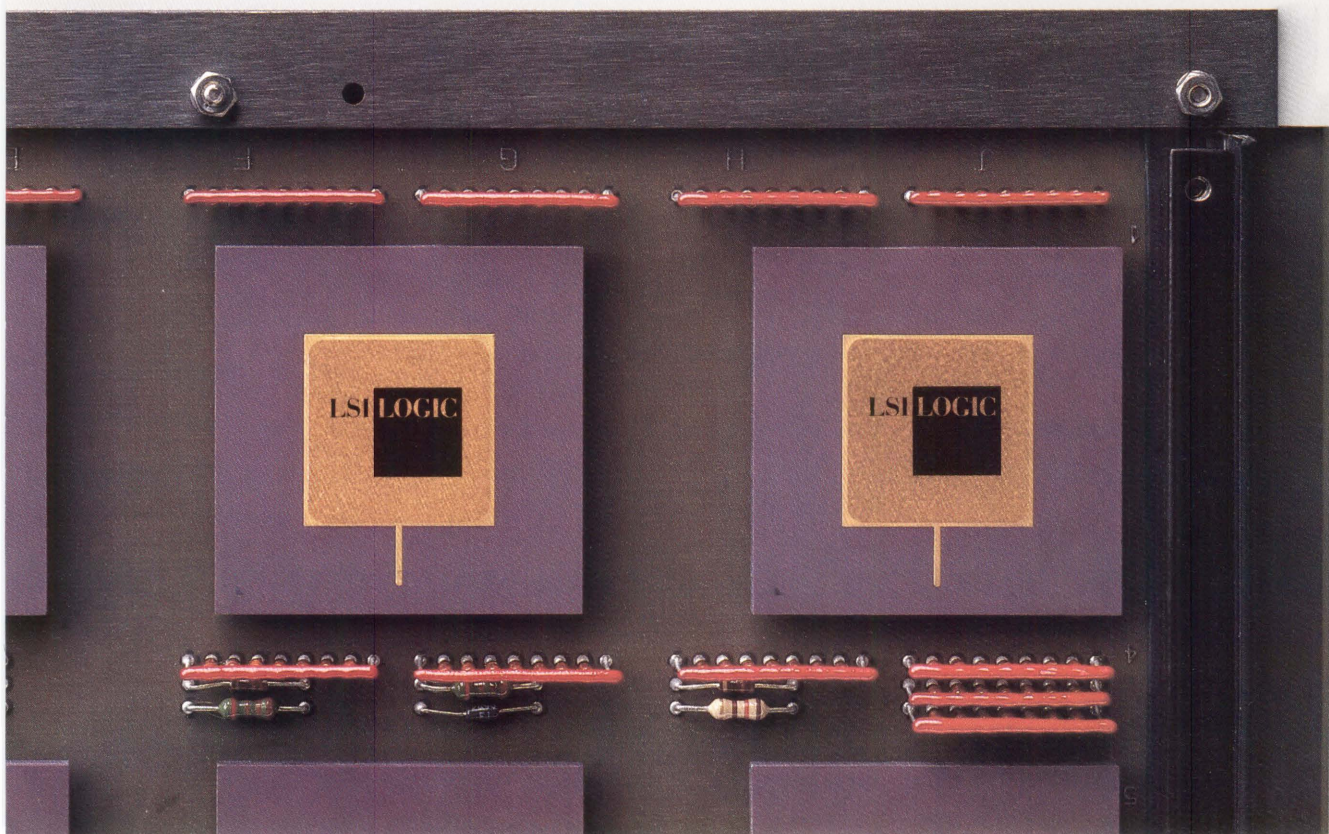
We see to it that it does. With an iron-clad guarantee that states that your device must operate precisely as simulated.

That's our system.  
You can't beat it.



# WE HAVE ASIC DOWN TO A SYSTEM.

*Supercomputer on a board.*



# CALL US WITH EVEN YOUR SMALLEST PROBLEM.

Sure, we handle the biggest ASIC projects. But you don't need a big reason to call LSI Logic.

It can be about designing a low complexity part or buying a small quantity of parts.

The important thing is that you call. We're close by. So nothing ever gets lost in transit or translation.

Our unique integrated approach means we're the only ASIC company you should ever need. And our worldwide manufacturing facilities are dedicated to ASIC. Which means you can quickly turn your designs into working parts.

Contact your nearest LSI Logic Design Resource Center or Sales Office. It's a small step that will lead to much bigger things.

LSI Logic Sales Offices and Design Resource Centers:

Scottsdale, AZ 602-951-4560  
Milpitas, CA 408-433-8000  
San Jose, CA 408-248-5100  
Irvine, CA 714-553-5600  
Sherman Oaks, CA 818-906-0333  
Denver, CO 303-756-8800  
Westport, CT 203-222-9336  
Altamonte Springs, FL 305-339-2242  
Boca Raton, FL 305-395-6200  
Bethesda, MD 301-897-5800  
Chicago, IL 312-773-0111  
Waltham, MA 617-890-0161  
Ann Arbor, MI 313-769-0175  
Minneapolis, MN 612-921-8300  
Bridgewater, NJ 201-722-7522  
Poughkeepsie, NY 914-454-6593

Raleigh, NC 919-783-8833  
Beaverton, OR 503-644-6697  
Trevose, PA 215-638-3010  
Austin, TX 512-343-4513  
Dallas, TX 214-788-2966  
Bellevue, WA 206-822-4384  
Calgary, Alta 403-262-9292  
Edmonton, Alta 403-424-8845  
Burnaby, BC 604-433-5705  
Kanata, Ont 613-592-1263  
Toronto, Ont 416-622-0403  
Pointe Claire, Quebec 514-694-2417  
Paris, France 33-1-46-21-25-25  
Israel 972-3-403741  
Milan, Italy 39-651575  
Ibaragi-ken, Japan 81-298-52-8371  
Tokyo, Japan 81-3-589-2711  
Osaka, Japan 81-6-947-5281  
Seoul, Korea 82-2-785-1693

Bracknell, United Kingdom  
44-344-426544  
Munich, West Germany  
49-89-926903-0  
Dusseldorf, West Germany  
49-211-5961066  
Stuttgart, West Germany  
49-711-2262151

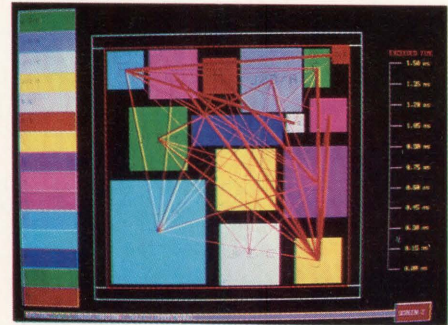
Distributors:  
Hall-Mark  
Hamilton/Avnet  
Wyle

**LSI LOGIC**

**THE ASIC SYSTEMS COMPANY**



## **IV DIRECTORIES**



**96      DIRECTORY OF CAE SYSTEMS**

**110     DIRECTORY OF IC LAYOUT SYSTEMS**

**118     DIRECTORY OF PRINTED CIRCUIT BOARD LAYOUT SYSTEMS**

# Directory of CAE Systems

Vendor	System overview		Design entry		
	Contact	Product name, cost, and host	Applications hardware	Schematics	HDLs
<b>AIDA Corp.</b> 5155 Old Ironsides Dr. Santa Clara, CA 95054  Georgia Marszalek Director of Marketing Communications (408) 980-5200	<b>AIDA Design System</b> \$140k, turnkey Apollo (UNIX, Domain, Domain gateways: SNA, VAX, Ethernet, etc.); Sun 3 (UNIX, NFS)	AIDA simulator accelerators	AIDA Schematic Design Editor	AIDA Design Language	Advanced CMOS gate array libraries: 100+ parts each
<b>Analog Design Tools Inc.</b> 1080 East Arques Ave. Sunnyvale, CA 94086  Michael P. Carroll Vice President, Marketing (408) 737-7300	<b>Analog Workbench</b> \$14.5k, software only; \$24k-\$62k, turnkey Sun 2 and 3 (UNIX, NFS); Apollo (AEGIS, Domain); (through Hewlett-Packard) HP9000/320 and 350 (HP/UX); also proprietary AnalogLink (RS-232) for all systems  <b>PC Workbench</b> \$8k, software only; \$15.8k, turnkey PC AT with Opus coprocessor (UNIX); proprietary AnalogLink (RS-232)	None	Analog Workbench Circuit Editor; PC Workbench Circuit Editor	None	Basic device library: 50 (included with PC Workbench); standard device library: 500; general device library: 1400+ as of fall 1987
<b>Applicon</b> 4251 Plymouth Rd. PO Box 986 Ann Arbor, MI 48106  Brian Barton Director, Electronics (313) 995-6000	<b>BRAVO 3 Electronic Design</b> \$10k-\$50k VAX (VMS), Sun (UNIX)	Interface to hardware modeler	Schematic capture	None	Numerous catalogs, including TI TTL, Motorola STTL, Motorola HCMOS, and Fairchild FAST
<b>Aptos Systems Corp.</b> 10 Victor Square Scotts Valley, CA 95066  James Franklin Product Manager (408) 438-2199	<b>RGRAPH</b> \$11k PC AT with 1024 x 768 display (includes graphics card); schematic and PCB layout  <b>CRITERION I</b> \$495, software only Software for PC AT with 640 x 356 display	None	RGRAPH schematic capture; CRITERION I schematic capture	None	TTL; CMOS; ECL; microprocessors; surface-mount; analog
<b>CADAM Inc.</b> 1935 N. Buena Vista St. Burbank, CA 91504  Alan Cohen Marketing Manager for CADAM Electrical Products (818) 841-9470	<b>Interactive Design System</b> \$65k-\$170k, software only IBM 4331 and up (VM/CMS, MVS, or VS1)  <b>Micro CADAM</b> \$8k, software only <b>Micro CADAM Cornerstone</b> \$2995 software only PC AT (MS-DOS)	None	CADEX	None	2000 SSI/MSI TTL and ECL; 700 memory parts; 8000 schematic and PCB design symbols
<b>CAD Group Inc.</b> 3911 Portola Dr. Santa Cruz, CA 95062  Vinnie Apicella Director of Marketing (408) 475-5800	<b>SALT</b> Software only: \$3.5k, IBM PC; \$15k, workstations; \$40k-\$60k, VAX; \$100k+ supercomputers PC XT, AT (DOS); MicroVAX to VAX 8800 (VMS, UNIX); Cyber and Cray (NOS, COS, UNICOS, CTSS); Sun (UNIX); Apollo (AEGIS, DOMAIN IX); Ridge (ROS); CRDS (UNOS)	SCLIP accelerator kit; logic/fault simulation accelerator (PCs only)	Interfaces to Scientific Calculations; Case Technology; OrCAD; CAECO	SALT Hardware Description Language (SHDL, an enhanced register-transfer language of Hughes Aircraft Corp.)	1500 SSI/MSI TTL and ECL; 72 LSI and VLSI; 100+ generic behavioral models
<b>Cadnetix Corp.</b> 5757 Central Ave. Boulder, CO 80302  Greg Skomp Marketing Communications Manager (303) 444-8075	<b>CDX-3000 PC schematic entry system</b> \$7950 PC ATs and compatibles (standard DOS, Ethernet, PC NFS), optical mouse  <b>CDX-96xx Design Management/Entry Environment</b> \$10.8k-\$15.9k Sun 3/50 and 3/60 (UNIX, Ethernet TCP/IP, NFS)	Configurable Analysis Engine (network processing node/server: accelerated digital simulation, accelerated digital and analog compilation, physical modeling, database management, analog simulation (2Q88))	Hierarchical schematic capture	None	2000 SSI/MSI TTL and ECL; 100 PLDs; 400 miscellaneous (primitives, CMOS); 1000 analog device models; hardware models: ASIC, ECL, TTL, CMOS, advanced functions
<b>CAECO Inc.</b> 2945 Oakmead Village Court Santa Clara, CA 95051  Mark Miller Director of Marketing (408) 988-0128	<b>CAECO schematic</b> \$10k Sun 3 (UNIX 4.2, NFS); Apollo (AEGIS, Domain); MicroVAX (ULTRIX, TCP/IP) (2Q88)	None	CAECO schematic editor/compiler	VERILOG (Gateway Design Automation); HILO-3 (GenRad)	Generic MOS device library

Design analysis		Design transfer		Additional capabilities	
Simulators and capabilities	Timing analysis	Netlist outputs	Test vector outputs	IC/PCB layout	Other tools, comments
<b>AIDA Transient Analysis Program</b> Circuit simulator  <b>AIDA Logic Simulator</b> Logic simulator  <b>AIDA Fault Simulator and AIDA Fault Inferencer</b> Fault simulator	AIDA Timing Verifier	TEGAS Design Language (TDL); foundry-specific (contact AIDA)	Foundry-specific (contact AIDA)	None	AIDA automatic test pattern generation
<b>SimKit</b> Simulator integration kit for access to user's own in-house simulators  <b>SPICE 2G.6</b> <b>SPICE PLUS</b> (enhanced SPICE 3) Circuit simulation, including time- and frequency-domain analysis	None	ASCII format	Not applicable	None	Parameter entry with subcircuits and a symbol editor; function generator and oscilloscope; frequency sweeper and network analyzer; dc meter; spectrum analyzer; statistical analysis; parametric plotting; power design module; stress analysis; IC design, power design, and circuit design tool kits.
<b>Logic Analysis</b> (enhanced CADAT) Functional, logic, behavioral, and fault simulator	Timing simulation in CADAT	CADAT; SPICE; template to reformat for others	Factron; Sentry; GenRad	BRAVO 3 VLSI geometry editor; ECAD layout analysis; BRAVO 3 PCB layout editor; automatic placement and routing (Algorex)	PG and E-beam interfaces; photoplot, drill, insertion, mechanical 3D design and analysis of PCBs
<b>PSPICE</b> (MicroSim) Circuit simulator	None	GDSII: SCICARDS; TEGAS; SILOS; ILOGS; LOGIS; SPICE	Not applicable	ICD-ONE, RGRAPH, and CRITERION II IC and PCB layout tools	None
<b>CADAM CADAT</b> (enhanced CADAT) Switch-level, gate, fault, and behavioral simulator  <b>CATS</b> (HHB Systems) Physical model simulation	None	CADAM; CADAT	None	Interactive Prance Cadam PCB layout	PCB thermal analysis; IPC350B output; 3D and solid model interface
<b>SALT</b> Switch, gate/functional, and behavioral simulator; concurrent timing verification/analysis  <b>SHDL</b> Behavioral model simulator  <b>PFG</b> (Mentor Graphics) Probabilistic fault grading	In SALT	SALT; SCICARDS; standard ASCII format	Sentry	None	Critical path analysis can be performed at the same time as logic simulation; test vectors include all timing of I/O and mask switching in Sentry format; translators from other simulators to SALT simulator
<b>Cadnetix 21-state simulator</b> (enhanced CADAT) Logic, switch-level, worst-case, behavioral simulator  <b>SABER</b> (Analog Inc.) Analog circuit simulation  <b>Cadnetix Fault Simulator</b> (enhanced CADAT)	In Cadnetix 21-state simulator	SPICE; CADAT; TEGAS; SCICARDS; EDIF 2.0	Zehntel; GenRad; Factron; Marconi; HP	CDX-56000 PCB layout stations; CDX-75000XP Route Engine III	User has access to all network resources
<b>HSPICE</b> (Meta-Software) Circuit simulator  <b>VERILOG</b> (Gateway Design Automation) <b>SILOS</b> (SimuCAD) <b>HILO-3</b> (GenRad) Behavioral, register transfer, gate, and switch simulator; symbolic debugging, fault grading	Limited	SPICE; ECAD; HILO-3; VERILOG; HSPICE, SILO; output can be formatted to any ASCII netlist description	None	CAECO custom IC layout; interactive DRC; automatic layout software; layout synthesis, automatic block placement and routing; DRACULA (ECAD)	GDS II stream converter; Versatec plotter interface for file servers

Directory of CAE Systems (continued)

Vendor Contact	System overview		Design entry		
	Product name, cost, and host	Applications hardware	Schematics	HDLs	Standard libraries
<p><b>Calay Systems Inc.</b> 2698 White Rd. Irvine, CA 92714</p> <p>Beverly Lages Marketing Communications Manager (714) 863-1700</p>	<p><b>ZX1000</b> \$8750 PC AT (PC-DOS 3.0+; serial/Kermit, Ethernet TCP/IP networks); Prisma; Sun 3 (UNIX)</p>	Interface to Calay PCB design systems	Schematic capture	None	6000, including SSI/MSI, memory, PLDs, LSI, and VLSI
<p><b>Calma Co.</b> 501 Sycamore Dr. MS C42D Milpitas, CA 95035</p> <p>Phil Arana PCB Product Manager (408) 434-4857</p>	<p><b>Board Scribe</b> \$8k-\$27.4k, software only or turnkey Apollo 3000; Ethernet TCP/IP</p> <p><b>Board Explorer</b> \$19k-\$79k, software only or turnkey Apollo 3000, DN570A</p>	None	Board Scribe	TEGAS Design Language (TDL); TDL/B behavioral language (Board Explorer only)	1500 TTL, ECL, and CMOS parts; Intel and Motorola microprocessors and peripherals
<p><b>Case Technology Inc.</b> 2141 Landings Dr. Mountain View, CA 94043</p> <p>Melanie King Manager, Marketing Communications (415) 962-1440</p>	<p><b>Case Vanguard CAE Design System</b> \$5k-\$25k, software only</p> <p><b>Case Vanguard Stellar CAE Design System</b> \$5k-\$80k, software only</p> <p>PC XT, AT (PC-DOS); VAX (VMS, DECnet); Sun (UNIX, NFS, PC NFS); Ethernet TCP/IP</p>	CATS hardware modeling (HHB Systems); interface to Zycad accelerators	Case Schematic Design System	SCALD design language (Lawrence Livermore National Lab)	5000+ parts: TTL, ECL, CMOS, PLDs, ASICs, and microprocessor families
<p><b>Computervision Corp.</b> 100 Crosby Dr. Bedford, MA 01730</p> <p>Dennis Kelly Product Marketing Manager (617) 275-1800 x2873</p>	<p><b>CADDStation</b> \$55.3k, turnkey 68020-based workstations (UNIX, Ethernet TCP/IP, NFS)</p> <p><b>Personal Engineer</b> \$3.5k, software only IBM PC and compatibles (MS-DOS)</p> <p><b>Personal Engineer/CU386</b> \$11.5k, turnkey 80386-based PCs (MS-DOS 3.2)</p>	CATS hardware modeling system (HHB Systems)	Schematic capture	HDL (GenRad); CADAT, BML (HHB Systems)	3000 TTL/ECL parts; 15 PLDs; 40 memory; 40 LSI/VLSI
<p><b>Control Data Corp.</b> 8100 34 Ave. S. PO Box 0 Minneapolis, MN 55440</p> <p>R.L. Biggs Marketing Manager (612) 853-5255</p>	<p><b>MIDAS</b> \$600k+, turnkey; \$75k+, software only CDC Cyber 180 (NOS, NOS/VE, HASP, X.25, Kermit)</p> <p><b>Electronics Designer</b> \$6k, software only PC XT or AT with Hercules or EGA graphics boards; VAX; IBM; Cyber 800 or Cyber 205; any LAN</p>	Interface to Zycad logic and fault accelerators	Daisy; Mentor	N.2	Gate array families and standard glue logic
<p><b>Daisy Systems Corp.</b> 700 Middlefield Rd. PO Box 7006 Mountain View, CA 94039</p> <p>Rich Dickerson Director of Corporate Communications (415) 960-6674</p>	<p><b>Personal Logician 286/386</b> PC AT or compatible (DNIX)</p> <p><b>Logician/Logician 386</b> Proprietary hardware (DNIX)</p> <p><b>A/D Lab</b> Software for Personal Logician 286 and 386, Logician 386 (DNIX)</p> <p><b>Entry!</b> Software for PC AT (DNIX)</p> <p>Prices not specified All systems: Ethernet/XNS and Daisy Networking Systems, TCP/IP, RJE/bisync</p>	Megalogician, PMX (Physical Modeling Extension); interface to IMS tester	DED II or ACE	Daisy Behavioral Language (DABL)	1100 TTL and ECL parts; 240 PLDs; 475 memory; 300 LSI and VLSI (hardware models); 500 other HCMOS, CMOS; 1200+ analog



Directory of CAE Systems (continued)

Vendor	System overview		Design entry		
	Contact	Product name, cost, and host	Applications hardware	Schematics	HDLs
<p><b>Data General Corp.</b> 6300 South Syracuse Way Englewood, CO 80111</p> <p>Elias Prado Manager, TEO/Electronics Marketing (303) 694-2900</p>	<p><b>TEO/Electronics</b> \$25k-\$28k, turnkey; \$9k, design system with design database; \$7.5k, Interactive Logic System (software only) Data General DS/7500 (DG AOS/VS or UNIX System V and X Windows; IEEE-802.3, X.25, TCP/IP, DG XODIAC, SNA, RJE, 2780, NFS)</p>	Interface to Zycad logic and fault accelerators	TEO/Electronics Design System	MAINSAIL (Xidak Inc.)	3200-parts, including SSI/MSI, LSI, VLSI, memory, PLDs, and analog
<p><b>Electronics Software Products</b> 18013 Sky Park Circle Irvine, CA 92714</p> <p>Behrooz Shariati Western Technical Manager (714) 261-1777</p>	<p><b>USPICE; PCUSPICE</b> <b>CADAT</b> <b>LOGNET</b> Software only; contact company for costs VAX (VMS, UNIX); IBM (MVS); Sun (UNIX); Apollo (AEGIS)</p>	None	LOGNET (VLSI Automation)	None	7400/5400 TTL; MECL; 10K
<p><b>Endot Inc.</b> 11001 Cedar Ave. Cleveland, OH 44106</p> <p>Michael Radovich Public Relations Manager Data I/O Corp. (206) 881-6444</p>	<p><b>N.2 System Design Environment</b> \$30k-\$200k, software only MicroVAX to VAX 8800 (VMS, UNIX); Apollo (AEGIS), Sun (UNIX); CDC (NOS/VE); IBM mainframes (VM/CMS); PC AT (GENIX); Ethernet TCP/IP</p>	None	Interfaces to workstations	ISP'; ISP' to VHDL (7.2 and IEEE) translators	Custom library support for system and subsystem entities through interfaces to workstations
<p><b>EPIC Design Technology Inc.</b> 3080 Olcott St., Ste. 203B Santa Clara, CA 95051</p> <p>Sang S. Wang President (408) 988-2944</p>	<p><b>TIMEMILL</b> \$15k, \$3k (PC version)—software only <b>TIMEMILL-CPA</b> \$7k, \$2k (PC version), \$5k (TIMEMILL add-on)—software only  Sun; Apollo; MicroVAX and VAX 8600 (ULTRIX, VMS); Valid SCALDStar; HP workstations; PC AT; Ethernet TCP/IP</p>	None	GED (Valid Logic Systems); CapFast (Phase Three Logic)	C and TIME-MILL HDL (a superset of C)	LSI Logic 7K; 120; Laserpath LP1000; 140; Raytheon ECL; 100; generic RAM, ROM, PLA, Am2900 family
<p><b>FutureNet</b> 9310 Topanga Canyon Blvd. Chatsworth, CA 91311</p> <p>Michael Radovich Public Relations Manager Data I/O Corp. (206) 881-6444</p>	<p><b>FutureDesigner</b> \$11.5k Mixed-mode design entry and logic synthesis: VAX (VMS); Sun; PC AT, PS/2</p> <p><b>DASH Schematic Designer</b> \$3990 Schematic capture: VAX (VMS); Sun; PC AT, PS/2</p> <p><b>DASH-CADAT Plus and FAULTSIM</b> \$10k Digital simulation system: VAX (VMS); Sun; PC AT, PS/2</p> <p><b>Personal Silicon Foundry</b> \$1495-\$15k Programmable logic development: VAX (VMS); Sun; PC AT, PS/2</p> <p><b>Analog Workbench</b> \$8k-\$45k PC AT</p> <p><b>Benchmark PCB</b> \$25k, PC AT; \$40k, Sun, VAX (VMS)</p>	Interfaces to CATS hardware modeler (HHB Systems), Zycad/SSC accelerators	DASH; schematic translators for Analog Design Tools, Computervision, Mentor	ABEL (FutureNet); ISP' (Endot)	2300+ symbols, including TTL, ECL, CMOS, Intel, Motorola, discrete. Libraries for 40+ ASIC vendors also are available
<p><b>Gateway Design Automation Corp.</b> 6 Lyberty Way PO Box 573 Westford, MA 01886</p> <p>Pete Johnson Marketing Manager (617) 692-9400</p>	<p><b>VERILOG</b> \$25k <b>VERILOG-XL</b> \$35k <b>TESTGRADE</b> \$20k <b>TESTGRADE-A</b> \$35k  Software only VAX and MicroVAX (VMS); IBM (VM/CMS); Sun (UNIX); Apollo (AEGIS); VERILOG, VERILOG-XL: Silicon Graphics; TESTGRADE: Cyber 180</p>	None	Netlist interfaces to CAECO; Daisy; Mentor; TDL; Valid	VERILOG C-based behavioral language	6500 SSI/MSI parts; 7 LSI/VLSI
<p><b>Harris Semiconductor</b> ASIC Operations PO Box 883 Melbourne, FL 32901</p> <p>James P. Spoto Director, Semicustom Services (305) 724-7383</p>	<p><b>Harris/SDA Design System</b> Price not specified; software only or turnkey Sun, Masscomp, MicroVAX, and Harris workstations (UNIX); Ethernet</p>	Interface to CATS accelerator and hardware modeler (HHB Systems)	Schematic capture	CADAT, BDL (HHB Systems); VHDL being developed	Harris standard-cell library contains 200 primitive and TTL functions; 12 LSI parts; and configurable RAM and ROM

Design analysis		Design transfer		Additional capabilities	
Simulators and capabilities	Timing analysis	Netlist outputs	Test vector outputs	IC/PCB layout	Other tools, comments
<b>Interactive Logic Simulator</b> Multimode switch, gate, function, behavioral, and timing simulation	Interactive Logic Simulator	LASAR; SCICARDS; Rascal-Redac; HILO; Caedent; Zycad; DG generic	None	Planned	ILS—simulation without netlist extraction; Sim Analyzer—simulation on schematic; DDL—database queries while designing; DRC—flags errors when made
<b>USPICE, PC-USPICE</b> Circuit Simulator  <b>CADAT</b> (HHB Systems) Logic simulator, fault simulator, behavioral simulator, timing, worst-case	None	None	Sentry; GenRad	Edge IC graphics layout (NCA)	Wirewrap support
<b>N.2</b> Mixed behavior/functional, register-transfer, gate simulator; software system simulator	None	Topology file in-house	User-created; test coverage tool	None	Stochastic performance analyzer; N.2 analysis environment; meta-assembler and retargetable linking loader; C behaviors, programs, and models can be linked; build tool for updating changes to models
<b>TIMEMILL</b> Mixed behavior/functional, register-transfer, gate and switch simulator	Dynamic timing verifier for setup, hold, edge-to-edge, and pulse width verification; static timing verifier for critical path analysis	SPICE; HILO; SILOS; TEGAS; VERILOG; LOGCAP	Sentry	None	None
<b>DASH-CADAT Plus</b> Functional/digital simulator  <b>Personal CADAT</b> Digital simulator  <b>DASH-Analog WorkBench</b> Analog simulator  <b>DASH-SPICE</b> Analog simulator	In DASH-SPICE or DASH-CADAT PLUS	ABEL; Applicon; CADAM; Computer- vision; EDIF; Rascal-Redac; SCICARDS; SPICE; TEGAS; over 50 others	Sentry; GenRad; IMS; Tektronix; over 10 others	Benchmark-PCB	PLDtest (automatic test-vector generation)
<b>VERILOG</b> Behavioral, functional, gate, and switch simulator  <b>VERILOG-XL</b> Accelerated gate and switch simulator (plus all VERILOG capability)  <b>TESTGRADE</b> Concurrent fault simulator  <b>TESTGRADE-A</b> Distributed fault simulator	None	None specified	Through Test Systems Strategies	None	TESTSCAN (ATPG and fault simulation for scan design systems); STATGRADE (statistical fault simulation); BITGRADE (fault simulation for BIST designs)
<b>SLICE</b> Circuit simulator  <b>CADAT</b> (HHB Systems) Switch, gate, behavioral simulator  <b>TA</b> (SDA Systems) Timing analyzer  <b>CADAT</b> (HHB Systems) Fault simulator  <b>CADAT</b> (HHB Systems) Hardware modeler	TA timing analyzer (SDA Systems)	EDIF input, GDS II output	Sentry	Standard-cell place and route (SDA Systems); complete layout analysis (SDA Systems)	Automatic sizing of bipolar transistors; logic optimization for area or speed; synthesis of logic circuits from Boolean expressions; schematic creation from netlist

## Directory of CAE Systems (continued)

Vendor Contact	System overview		Design entry		
	Product name, cost, and host	Applications hardware	Schematics	HDLs	Standard libraries
<b>Hewlett-Packard Co.</b> <b>Logic Systems Division</b> 8245 N. Union Blvd. PO Box 617 Colorado Springs, CO 80901  Art Pettis Marketing Communications (303) 590-5530	<b>Design Capture System</b> \$8k, software only HP Series 300 (68020, HP-UX); HP's Network Services NS/9000; NS-Arpa Services/300 (IEEE 802.3 Ethernet); UNIX BSD 4.2 network services  <b>Design Verification System</b> \$4k, HILO-3 interface; \$9k-\$46k, HILO-3 logic simulator; \$5k-\$30k, HILO-3 fault simulator HP9000, series 300/500/800	Interface to Hi-chip hardware modeling system (GenRad)	Design Capture System (HP's Salt Lake City Operation)	Functional Modeling Language (FML—GenRad)	2600 digital parts (TTL, MOS, ECL, microprocessors, PLDs); 2300 analog symbols; 1200 parts in analog model library
<b>HHB Systems</b> 1000 Wyckoff Ave. Mahwah, NJ 07430  Larry Blessman Marketing Manager (201) 848-8000	<b>CADAT</b> \$3k-\$100k+, software only VAX (UNIX, VMS); Sun (UNIX, NFS); Apollo (DOMAIN IX); IBM (MVS, DOS); Masscomp (UNIX); HP9000/series 300 (UNIX); Ethernet TCP/IP  <b>THESEUS automated test generation</b> Up to \$190k, software only Sun (UNIX, NFS); VAX (VMS, ULTRIX, Ethernet TCP/IP)	CATS logic/concurrent fault acceleration systems; OEM of Mach 1000 accelerator (Zycad); CATS hardware modeler models 6000, 8000, 10,000	Interfaces to Mentor; EDIF; TEGAS; FutureNet; Case Technology; PCAD; Cadnetix; Zuken; Computervision; Racal-Redac	Behavioral Description Language (BDL)	2500 SSI/MSI TTL and ECL; LSI Logic; SMC, VLSI Technology cell libraries; Fairchild, NEC gate arrays; Quadtree VLSI libraries; all FutureNet libraries; Gould
<b>IBM Corp.</b> 2077 Gateway Place San Jose, CA 95110  Stafford Johnson Electrical Design Marketing Program Manager (408) 288-4142	<b>CIEDS</b> Price not specified Software for IBM VM/370; RT PC (AIX 1.1+); PC AT (PC-DOS 3.1+); PS/2 Model 50 or 60 (PC-DOS 3.3); 3278/79 emulation program for communications with host mainframe; token-ring network	None	CIEDS/Design Capture	Hierarchical hardware description language (HHDL); Pascal-based modeling language	3000 TTL parts; 30 generic parts for CIEDS/Analog-Digital Simulator
<b>Integrated Silicon Design Pty. Ltd.</b> 230 North Terrace Adelaide SA 5000 Australia  A.R. Grasso Technical Manager +61-8-223 5802	<b>PHASE ONE</b> <b>PHASE TWO</b> <b>PHASE THREE</b>  Prices not specified VAX (VMS, UNIX), MicroVAX (VMS); Apollo 3000 (DOMAIN IX); Sun 3 (UNIX); PC AT with EGA (MS-DOS, XENIX)	None	None	Integrated system specification language, used in the system simulator	None
<b>Intergraph Corp.</b> 1 Madison Industrial Pk. Huntsville, AL 35807  Beverly Roan Electronics Marketing Engineer (205) 772-2000	<b>Electronics Design System (EDS)</b> \$15k+, turnkey Interpro32 standalone workstation (UNIX System V.3; XNS/Ethernet TCP/IP) tied to a VAX host	Interface to Hi-chip physical modeler (GenRad)	Hierarchical Schematic Design (HSD)	HILO-3 (GenRad)	3000 TTL, ECL, CMOS, memory, microprocessors, peripherals, discrete devices
<b>LSI Logic Corp.</b> 1551 McCarthy Blvd. Milpitas, CA 95035  Van Lewing Director of Software Marketing (408) 433-7204	<b>System Integrator</b> \$75k+, software only Sun 3, Sun 4 (UNIX, NFS); IBM 30xx (VM/CMS-compatible); VAX, MicroVAX (VMS, DECnet); Apollo 3000, DN570, DN580, 4000 (DOMAIN IX)  <b>Silicon Integrator</b> From \$75k, software only Sun 3, Sun 4 (UNIX); IBM 30xx (VM/CMS-compatible); VAX, MicroVAX (VMS); Apollo 3000 DN570, DN580, 4000 (DOMAIN IX); vendor-supplied networks  <b>Logic Integrator</b> \$75k+, software only Sun 3 (UNIX); vendor-supplied networks	Interfaces to LSI Logic accelerators; Zycad LE and FE accelerators  Interfaces to LSI Logic accelerators (ACCELSI) for logic and/or fault simulation, Zycad LE and FE accelerators  None	LSED  LSED	Network description language (NDL); hierarchical network description language (HNDL); Behavioral Specification Language (BSL)	Gate-model libraries; behavioral libraries in development



Design analysis		Design transfer		Additional capabilities	
Simulators and capabilities	Timing analysis	Netlist outputs	outputs	IC/PCB layout	Other tools, comments
<b>Analog Workbench</b> (Analog Design Tools) Circuit simulation and analysis  <b>HILO-3 Logic Simulator</b> (GenRad) Behavioral, functional, gate, and switch simulation; fault simulator	In HILO-3	HP-generic; user-definable; SCICARDS; Racal-Redac RINF; Calay; Computervision	HP 81810S IC verification system; HP3065 board testers; HP16500A logic analysis system	Graphics editors, IC layout analysis, IC symbolic layout and compaction (VTI and SDA Systems); automatic IC layout (VTI); CR 2000 hybrid IC design software (Zuken); Printed Circuit Design System (Hewlett-Packard)	Bundled design database language provides database access
<b>CADAT 6.0</b> Behavioral, functional, gate, and switch simulator; hardware modeling; concurrent fault simulator; simulation acceleration	Worst-case timing simulation through CADAT	Mentor; EDIF; TEGAS; FutureNet; Case; PCAD; Cadnetix; Zuken	Standard interface to CADAT binary databases; Factron; ITG/CADIF; IMS	None	PALGEN PAL model generation; THESEUS testability analysis and test generation for sequential scan or nonscan designs
<b>CIEDS/Logic Simulator</b> Functional and gate simulator  <b>CIEDS/Behavioral Simulator</b> Behavioral simulator  <b>CIEDS/Analog-Digital Simulator</b> Mixed-signal verification using piece-wise linear approximations  <b>CIEDS/Switched-Capacitor Simulator</b> Time and frequency domain analysis of switched-capacitor circuits	Timing checks for pulse width, setup time, hold time, and clock frequency	CBDS; HILO; TDL; SPICE 2G; Silvar-Lisco design verification tools; SDL (Structured Design Language)	None	Circuit Board Design System (CBDS)	Interactive multiwindow display of simulation results; simultaneous display of results from multiple simulations
<b>PROBE</b> Circuit simulator  <b>SYSMOD</b> Functional simulator using specification language	SYSMOD system simulation and timing analysis	SPICE; SIM (Berkeley)	None	PLAN IC geometric editor; SYSGEN symbolic layout; SYMEDIT symbolic layout; SYSPLAN floorplanner; CHECK, NET, ELEC, SYSCHECK layout analysis	None
<b>ACS/CSPICE</b> Analog circuit simulator with virtual test-instrument interface and automatic device characterization  <b>HILO-3</b> (GenRad) Functional and gate simulator; fault simulator; automatic test generation; physical model simulator	None	HILO; Telesis; FairCAD/SDL; user-reformatable ASCII netlist	HIPOST (GenRad 2270 series); Factron 303, 323, 330, 333; HP 3065; ESP Model 7100; Marconi System 80	IEDS PCB layout and routing, DRC, CAM; TANCELL (Tangent Systems) automatic timing-driven cell-based IC layout	MultiWire design; hybrid layout for thick- and thin-film hybrid circuits
Multichip gate-level simulator; multichip mixed behavioral/gate simulator  <b>ACCELSI</b> Hardware-accelerated gate simulator  <b>LDSr</b> Single-chip gate simulator  <b>BSIM</b> Single-chip behavioral/gate simulator  All simulators handle detailed delay prediction; back annotation is supported for delay prediction for distributed delay values	Path timing analyzer for multichip timing analysis	Network description language (NDL)	None	None	Power analysis; automatic schematic generation from NDL netlist; logic synthesis; PAL synthesis; logic compilers; multiplier compilers; memory compilers; TESTLSI automatic test pattern generation
	Path timing analyzer for single-chip timing analysis	Network description language (NDL); hierarchical network description language (HNDL)	Ando; Sentry	LDS layout tools for all LSI Logic technologies; floorplanning tools	
None	None	None	None	None	

## Directory of CAE Systems (continued)

Vendor	System overview		Design entry		
	Contact	Product name, cost, and host	Applications hardware	Schematics	HDLs
<b>Matra Design Semiconductor</b> 2895 Northwestern Pkwy. Santa Clara, CA 95051  Pradip Madan Vice President of Marketing and Sales (408) 986-9000	<b>GATEAID PLUS/PC</b> \$945, software only Compaq 386, PC/XT, AT (MS-DOS); Telenet X.25 connection to MDS host	None	DRAFT (adapted from OrCAD)	Boolean equation language and translator; Structural Description Language (SDL)	Standard cells: TTL, CMOS SSI/MSI; PLDs
	<b>GATEAID II</b> gate array design system \$25k + , software only MicroVAX, VAX (VMS)	None	GED graphics editor	FML (GenRad)	Standard cells: TTL, CMOS SSI/MSI; bit-slice LSI; RAM blocks; multiplier; UART
	<b>LSIntegrator</b> (Silicon Compiler Systems Corp.) Price no specified Sun (UNIX)	None	LED graphics editor	L-Language	Standard cells: ALU; sequencer; RAM; datapath elements
<b>Mentor Graphics Corp.</b> 8500 SW Creekside Place Beaverton, OR 97005  Mohan Nair Marketing Manager (503) 626-7000	<b>Entry Station</b> \$7k, software only PC XT, AT	HML hardware modeling system; Compute Engine accelerator; XSIM interface to Zycad	NETED/SYMED	Behavioral language models (extension of C and Pascal)	1226 TTL; 123 ECL; 416 CMOS, 56 PLDs; 146 memory; 1000 analog; 56 HML
	<b>Capture Station</b> \$20k, turnkey <b>Design Station</b> \$29k, turnkey <b>Idea Station</b> \$40k, turnkey  Apollo (AEGIS, UNIX; Domain, Ethernet TCP/IP, HASP, 3270, X.25)				
<b>MIETEC</b> Raketstraat 62 1100 Brussels, Belgium  Mike Butterworth USIC Business Manager (32) 2-242-5010	<b>MIETEC design system</b> Price not specified DAS 9100	None	SDS (Silvar-Lisco)	DAML	Standard-cell libraries, mixed digital-analog; CMOS; biMOS
<b>Motorola Inc., Semicustom Division</b> 1300 North Alma School Rd. Chandler, AZ 85224  Andy Graham CAD Portfolio Manager (602) 821-4180	<b>Design Verification Module</b> \$7.5k, software only  <b>Design Capture Module</b> \$2.5k, software only  Mentor Idea Station; Daisy Logician, Personal Logician; Valid CAE	6805 Core Development Module	Supports Mentor (NETED); Daisy (DED and ACE); Valid	None	Motorola gate array and standard-cell libraries
<b>Omaton Inc.</b> 1210 East Campbell Rd. Richardson, TX 75081  John Hoskins Vice President of Marketing (214) 231-5167	<b>SCHEMA II</b> \$495, software only <b>SCHEMA-PCB</b> integrated PCB layout \$975, software only <b>SCHEMA-ROUTE</b> autorouter \$850, software only  IBM PC and compatibles (DOS)	None	SCHEMA II	None	Symbols: standard TTL SSI/MSI; memory; PALs; microprocessors; analog; discrete
<b>OrCAD Systems Corp.</b> 1049 SW Baseline St. Suite 500 Hillsboro, OR 97123  Ken Seymour Vice President (503) 640-5007	<b>OrCAD/SDT III</b> \$495, software only <b>OrCAD/VST</b> \$995, software only  PC AT with EGA card (MS-DOS)	None	OrCAD/SDT III Schematic capture	None	3700 parts, including 1700 TTL, 335 CMOS, 184 ECL, 300 microprocessors and peripherals, 660 PALs and memory, 320 discrete, 235 analog
<b>Personal CAD Systems Inc.</b> 1290 Parkmoor Ave. San Jose, CA 95126  Terry Zimmerman Vice President, Marketing (408) 971-1300	<b>CAE-2</b> \$4k, software only PC XT, AT (DOS); HP Vectra; Olivetti; TI Professional; NEC PC-98XA; Ethernet, Novell software, 3Com	None	PCCAPS hierarchical schematic capture	None	3300 devices in 4600 packages, including TTL, CMOS, ECL, microprocessors, memory, analog

Design analysis		Design transfer		Additional capabilities	
Simulators and capabilities	Timing analysis	Netlist outputs	Test vector outputs	IC/PCB layout	Other tools, comments
<b>ARCIS</b> Gate simulator  <b>COFIS</b> Concurrent fault simulator  <b>HILO-3</b> (GenRad) Gate simulator  <b>LSIM</b> (Silicon Compiler Systems) Behavioral, gate, switch, and timing simulator	ARCIS timing analyzer; spike analyzer; WAVE waveform analyzer  ARCIS timing analyzer; in HILO-3  In L-SIM	ARCIS  ARCIS; HILO-3 format  None	Fairchild; ARCOP testability analyzer for 100% testability analysis   None	Proprietary    None	Bulletin board for data transfer and support     None
<b>MSIMON</b> MOS circuit simulator  <b>MSPICE PLUS</b> Circuit simulator with library  <b>QUICKSIM</b> Behavioral, functional, gate and switch simulator  <b>QUICKFAULT</b> (enhanced CADAT) Fault simulator	TVER    None	TDL; EDIF 200; SPICE; Computervision; ILOGS; NCA; SCICARDS; Racal-Redac; MASKAP; Valid; Vectron; LOGCAP    None	HP; GenRad; Advantest; Ando; Marconi; Sentry; Teradyne (through Test System Strategies Inc.)    None	Gate array layout; standard-cell layout; full-custom IC layout; PCB layout    None	None     None
<b>ANASIM</b> Circuit simulator  <b>DIGSIM</b> Functional and gate simulator  <b>DAML</b> Behavioral simulator  <b>FLTSIM</b> Fault simulator  <b>MIXSIM</b> Mixed-mode (digital/analog) simulator	Under development    None	SDL (Silvar-Lisico); Daisy; Valid    None	Sentry VII, 20, 21; Teradyne 300 series; NTDF (ITT-Alcatel)    None	IC layout (Calma and Computervision); DRC, ERC, CPA (MASKAP, ECAD); automatic IC layout (Silvar-Lisico)    None	Silicon compilers; PLA, RAM, ROM module generators; switched-capacitor filter compiler     None
<b>QUICKSIM</b> (Mentor) Behavioral/gate simulator  <b>DLS</b> (Daisy) Gate-level simulator  <b>Valid</b>	MTA timing analysis; VITEST (NCR)    None	TDL; Logcap; EDIF    None	Tester-independent code    None	TANCELL standard-cell layout (Tangent Systems); MERLYN-G gate array layout (Tektronix)    None	None     None
Netlist to PSPICE, SPICE, Susic, P/C SILOS and others	None	Cadnetix; Calay; Computervision; DataCon; FutureNet; Intergraph; PADS PCB; P-CAD; Racal-Redac; SCICARDS; SPICE; Tango-PCB; Telesis; others	None	None	Xilinx XACT and Intel IPLDs II interfaces; backward and forward annotation with full error checking
<b>OrCAD/VST</b> Functional, gate simulator; SPICE shell is built into schematic capture package	In OrCAD/VST	Applicon Bravo and Leap; Algorex; Calay; Cadnetix; Computervision; EDIF; FutureNet; Intergraph; MultiWire; PCAD; Salt; SPICE; SCICARDS; Racal-Redac; Telesis; Vectron; PADS; TANGO	None	Interface to Applicon; Algorex; Calay; Cadnetix; Computervision; FutureNet; Intergraph; PCAD; PADS; SCICARDS; Racal-Redac; Telesis; TANGO; Vectron	Scalable text and objects; hierarchy; object editor; De Morgan conversion; part rotation; on-line part browsing; keyboard macros
<b>PC-LOGS</b> Behavioral, gate and switch simulator	None	EDIF; TEGAS; HILO; SPICE; CADAT; SCICARDS; Computervision; CBDS; Calay; Racal-Redac	None	SMT PCB layout editor; automatic PCB layout; CAM output; Gerber output	PLD Design Tools; hierarchy; on-line design rule checking

Directory of CAE Systems (continued)

Vendor	System overview		Design entry		
	Contact	Product name, cost, and host	Applications hardware	Schematics	HDLs
<p><b>Phase Three Logic Inc.</b> 5510 NE Elam Young Pkwy. PO Box 985 Hillsboro, OR 97123</p> <p>Steve Bryan Technical Marketing Manager (503) 640-2422 x230</p>	<p><b>CFx000</b> \$395-\$4750, software only (except for CF3550) PC AT with EGA/VGA card (MS-DOS, TCP/IP); Sun (NFS, TCP/IP)</p>	<p>Interfaces to HILO-3 (GenRad); Zycad accelerators; Hichip hardware modeling system (GenRad)</p>	<p>SCHEDIT schematic editor</p>	<p>HILO-3 (GenRad)</p>	<p>2000+ parts in symbol library, including TTL, CMOS, ECL, microprocessors, support chips; GenRad simulation model libraries; HILO models for MMI PALs</p>
<p><b>Racal-Redac Ltd.</b> Tewkesbury Gloucestershire G120 8HE U.K.</p> <p>John Martin Marketing Manager (011) 44684294161</p>	<p><b>VISULA CAE</b> \$26k, software only <b>VISULA CAE/CAD</b> \$60k, software only</p> <p>VAXstation II (VMS); Apollo (UNIX)</p> <p><b>REDLOG/REDCAD</b> \$15k, software only PC AT and all compatibles (MS-DOS) with EGA and high-resolution graphics; Ungerman-Bass; Fox Research; 10-Net</p>	<p>CATS hardware modeler and simulation accelerator (HHB Systems)</p> <p>None</p>	<p>VISULA SCM hierarchical schematic capture</p> <p>REDLOG/REDCAD</p>	<p>In CADAT (HHB Systems)</p> <p>None</p>	<p>2000 models (TTL, CMOS) up to LSI complexity; 200+ behavioral and hardware models</p> <p>Variable</p>
<p><b>Royal Digital Systems Inc.</b> 3600 W. Bayshore Rd. Palo Alto, CA 94303</p> <p>Jerry Harvel Executive Vice President (415) 858-0811</p>	<p><b>AutoMate</b> \$25k-\$40k, software only PC AT or compatible (MS-DOS); Prime (PRIMOS, Prime-Link); Sun (UNIX, PC NFS), Data General (AOS/VS), Cyber series (NOS/VS); Ridge (UNIX); Ethernet</p>	<p>None</p>	<p>AutoMate Schematic Designer; enhanced CT-2000 for PC AT and Sun (Case Technologies)</p>	<p>None</p>	<p>2200, including TTL, LSTTL, CMOS, microprocessors and peripherals, ECL, discrete, passive, analog</p>
<p><b>Scientific Calculations Inc.</b> 7796 Victor-Mendon Rd. PO Box H Fishers, NY 14453</p> <p>Douglas Spice Director of Marketing Services (716) 924-9303</p>	<p><b>SCIDESIGN</b> \$6k <b>SCISIM</b> \$3k</p> <p>PC XT, AT (DOS 2.1; Kermit, DECnet-DOS communications)</p>	<p>None</p>	<p>SCIDESIGN</p>	<p>SCISIM behavioral modeling language</p>	<p>500 SSI/MSI TTL and ECL; generic PLA, RAM, and ROM models; 68000 and 2900 families</p>
<p><b>SDA Systems Inc.</b> 555 River Oaks Pkwy. San Jose, CA 95134</p> <p>Lesley Carr Manager, Public Relations and Promotions (408) 943-1234</p>	<p><b>CAE/CAT Tools</b> \$20k, software only Apollo; Sun; DEC; Masscomp; UNIX, Ethernet TCP/IP</p>	<p>None</p>	<p>Schematic capture</p>	<p>None</p>	<p>Standard logic gates for IC design</p>
<p><b>Seattle Silicon Corp.</b> 3075 112th Ave. NE Bellevue, WA 98004</p> <p>David A. Uvelli Director of Product Marketing (206) 828-4422</p>	<p><b>Concorde ASIC compiler</b> \$155k, software only Mentor Idea Series (Apollo 4000, 3000, DN570); Valid Logic (SCALDStar)</p>	<p>Simulation accelerators (Mentor, Zycad); hardware modelling system (Valid Realchip)</p>	<p>Interfaces to Mentor NETED; Valid GED</p>	<p>Joint development project with JRS Research for VHDL interface</p>	<p>150 SSI components; configurable MSI; configurable PLA; configurable RAM, ROM, FIFO memory; datapath and state machine compilers; configurable I/O; analog</p>
<p><b>Silicon Compiler Systems Corp.</b> 2045 Hamilton Ave. San Jose, CA 95125</p> <p>Jeff Elias Marketing Manager (408) 371-2900</p> <p>Jim Griffeth Marketing Manager (201) 580-0102</p>	<p><b>GENESIL silicon compiler</b> \$155k, software only <b>GENEPORT link between GENESIL and GDT</b> \$10k, software only VAX, MicroVAX (ULTRIX); Apollo (DOMAIN IX); Sun (UNIX); Ethernet TCP/IP</p> <p><b>GDT generator development tools</b> \$88k, software only <b>LSIM</b> \$49.5k, software only <b>LTIME</b> \$40k, software only Apollo (DOMAIN IX); DEC (ULTRIX); Sun (UNIX)</p>	<p>Interfaces to Mach 1000 logic/fault accelerator (HHB Systems)</p> <p>None</p>	<p>Interfaces to Mentor; Daisy</p> <p>LED interactive schematic and layout editor</p>	<p>None</p> <p>L language for VLSI; M language for behavioral description</p>	<p>Library of CMOS compilers; CRT controller; core microprocessor; memory; standard cells</p> <p>LCELLS generator libraries include standard cells, memory, CRT controller, core microprocessor</p>
<p><b>Silvar-Lisco</b> 1080 Marsh Rd. Menlo Park, CA 94025</p> <p>Wence Coron Director of EDA Marketing (415) 324-0700</p>	<p><b>Design Entry</b> \$6k + <b>Logic Design System</b> \$25k + <b>System Design</b> \$40k + <b>Mixed Analog/Digital Simulation System</b> \$25k + <b>Switched Capacitor Simulation System</b> \$23k +</p> <p>MicroVAX, VAX (VMS); PCAT, RT, 43xx, 9370, 30xx (MS-DOS, UNIX, VM/CMS); Apollo (AEGIS); Sun (UNIX); DECnet, IBM-supported networks, Ethernet, Domain</p>	<p>Interface to Zycad accelerators</p>	<p>Schematic Design System (SDS)</p>	<p>HELIX HHDL (enhancement of Stanford University's ADLIB)</p>	<p>4000 SSI/MSI TTL; 328 PLDs; 162 memory; 120+ LSI/VLSI (available through Quadtree); 70 miscellaneous CMOS; additional simulation libraries from Quadtree Corp.</p>

Design analysis		Design transfer		Additional capabilities	
Simulators and capabilities	Timing analysis	Netlist outputs	Test vector outputs	IC/PCB layout	Other tools, comments
<b>Berkeley SPICE</b> (for Sun) PSPICE (for IBM PC AT) (MicroSim) Circuit simulator  <b>HILO-3</b> (GenRad) Behavioral, functional, and gate simulator; fault simulator	in HILO-3	Computervision Cadds4X and Cadds3; Racal- Redac; SCICARDS	HIPOST (GenRad)	None	Symbol editor, compiler; programmable netlist library extraction; interactive waveform grapher; plotting program
<b>SPICE</b> (Berkeley Version 2G6) Circuit simulator  <b>CADAT 5.1</b> (HHB Systems) Behavioral, gate and switch simulator; concurrent fault simulator  Interfaces to Personal CADAT, P-SILOS logic simulators  Interfaces to PACSIM, PSPICE circuit simulators	in CADAT  In P-CADAT and P- SILOS (future)	CADAT 5.1; HILO; SPICE  Racal-Redac; HHB Systems	Eaton Model 800; Sentry  None	VISULA PCB automatic layout; REDBOARD PCB automatic layout  Geometry editor; layout analysis; automatic placement and routing; VISULA PCB; Red Draw PCB; Red Therm (future)	VISULA SCM on-line electrical rule checking; REDLOG/REDCAD waveform analysis  REDSOFT customer support/software maintenance product; REDDOC documentation tool (future)
<b>P-SILOS, CADAT</b> Behavioral, functional, gate, and switch simulator; fault simulator; physical modeling  <b>P-SPICE</b> Circuit simulator	for PC AT and Sun: timing verifier (enhanced CASE);	AutoMate; SCICARDS; Calay; Cadnetics; Valid; Daisy; Mentor; FutureNet; Racal- Redac; Case	None	Automatic PCB layout, including SMD and hybrids	None
<b>SCISIM</b> Behavioral, logic, and switch simulator  <b>SPICE</b> Circuit simulator	None	SCIDESIGN ASCII netlist data	Sentry	MEDS IC geometry editor, layout analysis, and automatic placement and routing; SCICARDS PCB geometry editor and automatic placement and routing	None
<b>SILOS</b> (SimuCAD) Switch-level, logic, and fault simulator  <b>HILO</b> (GenRad) Logic and fault simulator  <b>SPICE; HSPICE</b> (Meta-Software) Circuit simulation	TA Timing Analysis (SDA Systems)	EDIF	Sentry; GenRad; Advantest	IC layout design, DRC; automatic IC layout; IC floorplanning	Module compilation; generalized simulation/test language
Interfaces to QUICKSIM (Mentor) and ValidSIM (Valid Logic) logic simulators	Analysis tools	Mentor; SILOS	Sentry	Interface to Mentor Chipgraph and Valid LED outputs GDSII and CIF; automatic IC layout with interactive override; DRACULA DRC (ECAD)	First Silicon Services; services to support ASIC verification, prototyping and production fabrication, packaging and testing
In GENESIL Functional logic simulator with switch-level option; interface to Mentor QUICKSIM  <b>LSIM</b> Behavioral, functional/gate, and switch simulation integrated with ADEPT circuit simulation; fault simulation (both serial and probabilistic); LSPICE analog simulation	GENESIL timing analyzer  LTIME Timing Analyzer (in house); RC tree-based switch- level timing analysis	Mentor  SPICE; EDIF	IMS  IMS; Sentry 7, 8, 20, 21	Interactive and automatic IC layout tools; LogicCompiler automatic layout of logically optimized standard cells  Led IC layout, geometry editing, mask-level symbolic layout, floorplanning; LRC electrical and design rule checking; extract database extraction routines; LPAR automatic placement and routing of standard cells and blocks	Automatic test generation  L Database Interface (LDBI)
<b>HELIX</b> Behavioral simulator  <b>LOGIX</b> Functional/gate simulator  <b>ANDI</b> Switch simulator (mixed analog analysis)  <b>SWAP</b> Switched-capacitor filter analysis	Simulation libraries for HELIX and LOGIX contain "intelligent" behavioral models that also perform timing checks	SPICE; HILO; LOGCAP; TEGAS; SILOS; CBDS; SCICARDS; RACAL- REDAC; SUPER- COMPACT; SECMAI/OPTIMA	Sentry; GenRad; Advantest; Teradyne; IMS (through Test Systems Strategies Inc.)	Automatic gate-array and standard-cell layout; IC layout design; layout analysis; PCB placement and routing	Design databases (both netlist and schematic) are portable between supported brands of hardware

Directory of CAE Systems (continued)

Vendor Contact	System overview		Design entry		
	Product name, cost, and host	Applications hardware	Schematics	HDLs	Standard libraries
<b>Spectrum Software</b> 1021 S. Wolfe Rd. Sunnyvale, CA 94086  Karen Burchfiel Customer Service Representative (408) 738-4387	<b>Micro-Cap I &amp; II schematic capture</b> <b>Micro-Logic I &amp; II logic simulation</b> \$450-\$895, software only PC, XT, AT, or fully compatible (DOS 3.0)	None	Micro-Cap II and Micro-Logic II schematic capture	None	200 elements for Micrologic II
<b>Tektronix Inc.</b> <b>CAE Systems Division</b> PO Box 4600 Beaverton, OR 97076-4600  Ron Workman Division Marketing Manager (503) 629-1036	<b>Designer's WorkSystem</b> From \$18k, software only and turnkey <b>PCB WorkSystem</b> From \$53k, software only and turnkey <b>Gate Array WorkSystem</b> From \$70k, software only and turnkey  Apollo (AEGIS UNIX 4.2 BSD, Domain); VAX, MicroVAX, GPX (VMS, DECnet)  <b>Full Custom WorkSystem</b> From \$50k, software only and turnkey DEC VAX, MicroVAX, GPX (VMS, DECnet)	Interfaces to Zycad; GenRad Hichip; Tektronix DAS	Designer's Database Schematic Capture (DDSC)	HILO-3 (GenRad)	6000+ including TTL74, TTL54, ECL, PLDs, ROM, CMOS, microprocessors, discretes, HPRM (HSPICE)
<b>Teradyne Inc.</b> 321 Harrison Ave. Boston, MA 02118  Daryl Layzer Marketing Services Manager (617) 482-2700 x2808	<b>DATAView design entry system</b> \$5k, software only PC AT and compatibles (DOS); Viewnet (Ethernet XNS, TCP/IP)  <b>LASAR Version 6 simulation software</b> \$25k+, software only VAX (VMS), DATAServer Simulation Engine (UNIX); DECnet; Viewnet (Ethernet XNS, TCP/IP)	DATASource hardware modeling system; DATAServer Simulation Engine (including LASAR)	DATAView (enhancement of Viewdraw, Viewlogic Systems)	TML register-transfer language; LABEL behavioral language	DATAView: 1600 SSI/MSI TTL, ECL; 300 standard symbols  LASAR Version 6: 3800 SSI/MSI TTL, ECL; generators for memory; 200+ LSI/VLSI; gate array macrocells
<b>Valid Logic Systems Inc.</b> 2820 Orchard Pkwy. San Jose, CA 95134  Nancy Madison Director Product Marketing (CAE/IC CAD) (408) 432-9400  Kathy Gambino Product Marketing Manager (PCB CAD) (617) 256-2300	<b>Design Entry System</b> \$5.9k+ <b>Logic Design System</b> \$9.9k+ <b>Design Validation System</b> \$17.5k+ <b>Analog Design System</b> \$10.5k+  VAX, VAXstation (VMS); Sun (UNIX); PC AT (UNIX); SCALSystem (UNIX); Ethernet TCP/IP, DECnet, VAX-cluster (LAVC)	Realchip hardware modeler; Networked Realchip; Realmodel hardware modeler; Realfast simulation accelerator	ValidGED	UCP C-based behavioral modeling language	4000+ TTL, ECL; 30+ PLDs; 60+ memory; 100+ LSI/VLSI; 97+ ASIC design kits; behavioral models from Logic Automation and Quadtree; analog libraries
<b>Vamp Inc.</b> 6753 Selma Ave. PO Box 411 Los Angeles, CA 90028  John Soluk Manager of Marketing (213) 466-5533	<b>McCAD Schematics</b> \$495, software only <b>McCAD DACS</b> \$295, software only  Apple Macintosh 512, Macintosh Plus, Macintosh SE, Macintosh II; Apple Network	None	McCAD Schematics	None	200+ TTL; 100 discrete; 250 CMOS
<b>Viewlogic Systems Inc.</b> 275 Boston Post Rd. West Marlboro, MA 01752  Sri Sriram Vice President of Marketing (617) 480-0881	<b>Workview series software</b> \$5k-\$14k, software only; turnkey systems available PC XT, AT, and compatibles (DOS); Compaq 286, 386 PCs (DOS); Ethernet, RS-232, XN	Zycad simulation accelerator	Viewdraw	VHDL	2000+ TTL; 100+ ECL; 100+ PLDs; 100+ memory; 50+ LSI/VLSI; 600+ analog; semicustom libraries from over a dozen vendors
<b>Visionics Corp.</b> 343 Gibraltar Dr. Sunnyvale, CA 94089  Alex Wellins Public Relations Manager (408) 745-1551	<b>EE DESIGNER, EE DESIGNER II</b> \$995-\$1895, software only <b>AUTOROUTER, AUTOROUTER II</b> \$995-\$1475, software only  PC XT, AT, PS/2, and compatibles (DOS 2.0+)	None	EE DESIGNER schematic capture; also interfaces to SDT III (ORCAD); SCHEMA (Omatation); DASH (FutureNet)	None	850 TTL, CMOS, analog, and SMD parts; library cross-reference files with 200 parts each
<b>VLSI Technology Inc.</b> 1109 McKay Dr. San Jose, CA 95131  Bill Miller Tactical Marketing Manager (408) 434-3000	<b>VLSI Express Systems</b> \$7k-\$210k VAX (VMS); Apollo (AEGIS); HP series 9000, model 300, 350 (UNIX); Sun 3 (UNIX), Elxsi; Ethernet	None	VTIschematic	Hierarchical Net List (HNL)	Portable gate array, standard-cell, and compiler libraries
<b>Xerox EDDS</b> 2441 Mission College Blvd. Santa Clara, CA 95054  Petros Xides CAE Product Division Manager (408) 562-2191	<b>Expert Designer</b> \$7k-\$12k Xerox 6085; Ethernet	None	Expert Schematics	XDDL	1200 TTL, CMOS, and ECL symbols in schematic symbol library; 200 models in simulation model library

Design analysis		Design transfer		Additional capabilities	
Simulators and capabilities	Timing analysis	Netlist outputs	Test vector outputs	IC/PCB layout	Other tools, comments
<b>Micro-Logic</b> Logic simulator  <b>Micro-Cap</b> Circuit simulator	Micrologic, Micro-Logic II	ASCII format	None	None	None
<b>HILO-3</b> (GenRad) Behavioral and gate simulator; fault simulator; physical modeler  <b>SPICE; HSPICE</b> (Meta-Software) Analog circuit simulators	In HILO-3 (GenRad)	Zycad; SPICE; SCICARDS; Calay; CADD5; HSPICE	Tektronix (LT-1000); Advantest; Sentry; Teradyne; HILO-3 automatic test generator (GenRad)	LEIA IC layout editor; MERLYN-G automatic gate-array layout; MERLYN-S automatic standard-cell and block layout; MERLYN-P Automatic PCB layout; DRACULA DRC (ECAD)	TekWriter (Interleaf) publishing software for engineering documentation; MicroLink interface to Tektronix Computer-Aided Software Development Tools; SuperCompact DDSC RF circuit simulator (Compact Software)
<b>LASAR Version 6</b> Behavioral, functional, and gate simulator; hardware modeling; fault simulator; automatic test pattern generation; interface to DATASource hardware modeling system	True worst-case timing analysis	EDIF; LASAR Version 6; SCICARDS	Teradyne L1xx, L2xx, J9xx; Computer Automation 4700, 4900, Marathon; GenRad 197x, 2225, 2235; Hewlett-Packard DTS70; Sentry 7-21	None	Document processor (merges text with engineering graphics), electronic mail
<b>ValidSPICE</b> PRECISE circuit simulator (Electrical Engineering Software)  <b>ValidSIM</b> Behavioral, logic, and switch simulator with timing analysis  <b>TIMEMILL</b> (EPIC Design Technology) Mixed-level timing simulator and critical path analyzer  <b>LASAR 6</b> (Teradyne) Logic and fault simulator	ValidTIME; TIMEMILL (EPIC Design Technology)	HDL; HILO-3; LOGCAP; MCLDL; SPICE; TEGAS5; CADD5; Calay; CBDS; Paragon; Racal-Redac; SCICARDS; Wirewrap; ILOGS; LIL; FAIR-LOGS; CPP; CADAM; SEL; SDL; SDIF; Applicon; LASAR	Teradyne; HP; GenRad; Zehntel	IC layout editor; IC layout analysis; full-custom and standard-cell IC layout (DeNies Resources); Concorde silicon compilers (Seattle Silicon); ALLEGRO interactive and automatic PCB layout	ValidFLAT—automatic schematic generation from netlist; ValidPLD—automatic generation of timing and logic models; ValidEZLIB—creates new library components by modifying existing ones; DIAL—programming language for custom interface creation
<b>McCAD DACS</b> Circuit, logic, and behavioral simulator	In McCAD DACS	Computervision; Calay; Gerber; SCICARDS; McCAD	None	McCAD PCB-1, PCB-ST, and automatic routing tools	McCAD-to-Gerber translator module
<b>VIEWSIM/AD</b> (enhanced PSPICE) Mixed analog-digital simulator  <b>HILO</b> (GenRad); <b>TEGAS</b> (Calma); <b>LASAR</b> (Teradyne); <b>SILOS</b> (SimuCAD); <b>ZILOS</b> (Zycad); <b>PSPICE</b> (MicroSim); <b>SPICE</b> (UC Berkeley); <b>HSPICE</b> (Meta-Software); <b>IG-SPICE</b> (A.B. Associates); <b>PRECISE</b> (EESof); <b>ALLSPICE</b> (Acotech); <b>Touchstone</b> (EESof)	None	PCB: Academi; Applicon; Cadnetix; Calay; CBDS; Computervision; Intergraph; MERLYN; OmniCad; P-CAD; PRANCE; Racal-Redac; SCICARDS; Valid; Datacon. <i>Simulators:</i> CADAT; Computervision; Silvar-Lisco; SDL; BOLT. <i>PLD:</i> Altera, MMI, Xilinx. <i>IC layout:</i> ECAD	None	Configurable design rule checker; VIEWPLACE for critical placement	Document processor (merges text and graphics); electronic mail system; data management; 80386 simulation tools to support ASIC design beyond 30,000 gates; Pre-CAD PCB placement for PCB design
In EE DESIGNER Gate-level logic simulator with timing verification	In EE DESIGNER	ASCII file format	None	PCB design rule checker; netlist extraction; connectivity checks; editors; automatic placement and routing	Interfaces to Gerber, N/C drill tape, remote plotters
<b>VTIsim</b> Behavioral, gate, and transistor simulator  <b>VTISPICE</b> Circuit simulator	TV timing analysis	HNL netlist format	Sentry 10, 20	Sticks symbolic IC editor; automatic IC layout; cell compiler; datapath compiler; state machine compiler	None
<b>Expert Logic Simulator</b> Switch-level and logic simulation	None	LASAR; Computervision; Racal-Redac; SCICARDS; CADAT; HILO; EDIF; SPICE; user-definable data extractor	None	Expert PCB System board layout tools	Interfaces to VIEWPOINT (Xerox office automation system); IGES translator

# Directory of IC Layout Systems

Company contact	System name and configuration	Typical cost	Hardware configuration				
			CPU (operating system)	Main memory secondary storage	Graphics processor and memory	Display resolution size	Local-area network
<b>Andrew Tickle Associates</b> 1222 Richardson Ave. Los Altos, CA 94022  Andrew Tickle President	<b>TURBO-CAD</b> (software only)	Not specified	VAX, Sun, PC AT	640K, PC AT  Hard disk	Standard EGA, PC AT	EGA monitor	None
<b>Applicon</b> 4251 Plymouth Road PO Box 986 Ann Arbor, MI 48106 (313) 995-6000  Brian Barton Director, Electronics	<b>Bravo 3 VLSI</b> (stand-alone workstation or host with attached stations)	\$115k; \$15k (software); \$65K (workstation)	VAX (VMS), Sun (UNIX)	Standard VAX and Sun configurations	Proprietary and standard supported	Up to 1536 × 1157, color  19"	DECnet, Ethernet
<b>Aptos Systems Corp.</b> 10 Victor Square Scotts Valley, CA 95066 (408) 438-2199  James Franklin Product Manager	<b>ICD One</b> (software and graphics card)  <b>DeskTop Cell Pro</b> (software only)	\$11k  \$5k	IBM or Compaq 286 or 386 or compatibles	640 KB  20-MB disk minimum  9-track tape	Artist I series controller card  EGA	1024 × 768 × 4  19"  EGA monitor	3Com; GDS II interface via PC serial interface or Ethernet
<b>CAECO Inc.</b> 2945 Oakmead Village Court Santa Clara, CA 95051 (408) 988-0128  Mark Miller Director of Marketing	<b>CAECO VLSI IC Design System</b> (stand-alone workstation or software only)	\$30k–\$50k (software)	Sun, Apollo, MicroVAX II (Q2/88) (UNIX 4.2 BSD)	4–32 MB  70-, 130-, and 474-MB disks  9-track and 20- or 40-MB 1/4" tape	Standard OEM graphics; no proprietary graphics hardware necessary	Standard OEM graphics: 1024 × 1024 × 8, 1024 × 1024 × 4	Ethernet TCP/IP, Domain
<b>Calma Co.</b> 501 Sycamore Dr. Milpitas, CA 95035 (408) 434-4056  Larry Yamada IC Product Marketing Manager	<b>GDSII/3200</b> (stand-alone workstation)  <b>P4281</b> (host with attached stations)  <b>EDS III</b> (stand-alone workstation or software only)	\$84k  \$332k (4 stations)  \$41k–\$150k bundled; \$20k–\$50k software	Data General MV7800 (AOS/VS)  Data General Eclipse S-280 (CDOS)  Sun, Apollo, MicroVAX	4–14 MB 160–320 MB  1–2 MB 515-MB disk (up to two)  8–16 MB 140–560 MB	Proprietary bit-slice  Lexidata 3400  Standard OEM	1280 × 1024 × 4  19"  640 × 512 × 4  19"  1100 × 900 × 8  19"	Ethernet  Ethernet  Ethernet TCP/IP
<b>Control Data Corp.</b> 8100 34th Ave. South Minneapolis, MN 55440 (612) 853-5255  R.L. Biggs Marketing Manager	<b>MIDAS/LAYOUT</b> (host or cluster controller with attached workstations or graphics terminals)	\$250k (software)	CDC CYBER-180 series (NOS)	8–16 MB  200 MB on line	Apollo processors and/or Orcatech 2000 processor  1 MB	512 × 512, 1024 × 1024  9"–14"	Domain
<b>Daisy Systems Corp.</b> 700 Middlefield Rd. Mountain View, CA 94039 (415) 960-7168  Mark T. Fuccio ChipMaster Product Manager	<b>ChipMaster</b> (stand-alone workstation)  <b>IC Verification Server</b> (attached processor)	\$70k–\$160k  \$90k–\$150k	80386/287 (DNIX: UNIX 4.2 BSD enhanced)  MicroVAX (DNIX, VMS)	4–16 MB; 85-MB disk minimum; 1.2-MB floppy; 9-track tape  3–9 MB; 71-MB disk minimum; 5 1/4" floppy; 60-MB cartridge, 9-track tape	Am29116 bit-slice display controller  None	1024 × 832 (× 4 or × 8)  19"  None	DLAN (Daisy local-area network, Ethernet-based)  DLAN
<b>ECAD Inc.</b> 2455 Augustine Drive, Santa Clara, CA 95054 (408) 727-0264  Shrikat Sathe Product Manager	<b>Dracula</b> (stand-alone workstation or host with attached stations)	\$35k–\$150k	MicroVAX with VMS or ULTRIX; Apollo; Sun; MIPS	8 MB  160-MB disk	Apollo, DEC, Sun	1024 × 1024 × 4, 1024 × 1024 × 8  15"–19"	Domain, DECnet, Ethernet TCP/IP



I/O formats							Artwork database operations							Front-end design tools	Symbolic layout and compaction; module generators
GDS I	GDS II	APL 860	APL 870	CIF	PG	EBES	Sizing	Scaling	Boolean	Design-rule checking	Extraction	ERC	Netlist compare		
○	●	○	○	●	○	○	○	○	●	GDS I and CIF I/O, batch, on-line	Output during layout synthesis as SPICE netlist with parasitics included	○	●	Logic and circuit synthesis; CMOS transistor-level netlist generation; Silvar-Lisco SDS and CAL-MP interface	Automatic self-compacting layout; cell generation; pad layout; RAM generator; netlist comparison
●	●	●	●	○	●	●	●	●	●	Interactive; ECAD batch, incremental, hierarchical	Connectivity; transistor, R, and C parameters	●	●	Design capture, logic analysis (based on CADAT), circuit analysis (based on SPICE)	Cell generator runs off a netlist from a schematic
○	●	○	○	○	○	○	●	●	○	Batch	Connectivity; R and C parameters with circuit elements	●	●	CRITERION I schematic capture; PSPICE circuit simulator	None
○	●	○	○	○	●	●	●	●	●	Batch, incremental, on-line, and hierarchical	Connectivity; transistor, R, and C parameters	●	●	Schematic capture; VERILOG, SILOS logic simulators; various circuit simulators; Gateway test tools	Symbolic layout based on parameterized transistors; automatic layout from circuit description
●	●	○	●	●	●	●	●	●	●	Batch runs on Eclipse and hardware accelerator (Fast Mask Engine)	Connectivity; transistor, R, and C parameters	●	●	Schematic capture; TEGAS 5, TEXSIM/B logic simulators; HSPICE circuit simulator	Stick layout compactor; cell compilers
○	●	○	○	○	●	●	●	●	●	Batch, on-line		●	●	Same	Compactor
○	●	○	○	○	●	●	○	○	○	Batch, incremental, on-line	Capacitance parameters	●	○	Schematic capture; MIDAS and other logic simulators; SPICE, ASPEC circuit simulators; Gateway test tools	None
○	●	●	○	●	●	●	●	●	●	On-line DRC; batch Dracula II (also Dracula I on Verification Server)	Connectivity; transistor, R, and C parameters  Automatic parameter insertion to simulator	●	●	DED II, ACE schematic capture; DLS logic simulator; DSPICE, CHIP-SIM circuit simulators	None
○	●	●	○	●	●	●	●	○	●	Batch, incremental, on-line, hierarchical	MOS, bipolar, CMOS connectivity with R and C parameters	●	●	Schematic capture; SPICE, HSPICE, SIMON, TEGAS, HILO, SILOS and other circuit simulator interfaces	Sticks editor; compactor; procedural language and graphical entry for cell generation

Directory of IC Layout Systems (continued)

Company contact	System name and configuration	Typical cost	Hardware configuration				
			CPU (operating system)	Main memory secondary storage	Graphics processor and memory	Display resolution size	Local-area network
<p><b>EEsof Inc.</b> 31194 LaBaya Dr. Westlake Village, CA 91362 (818) 991-7530</p> <p>Sandra Rochowansky Manager of Marketing Communications</p>	<p><b>MiCAD</b> (stand-alone workstation or software only)</p>	<p>\$8k (hardware); \$8.4k (software)</p>	<p>HP Vectra (MS-DOS 3.1)</p> <p>PC XT, AT, or compatibles (MS-DOS 3.1)</p>	<p>640K</p>	<p>CGA, EGA</p> <p>16 KB for CGA, 128 KB minimum EGA</p>	<p>320 × 200 × 2 (CGA)</p> <p>640 × 350 × 4 (EGA)</p> <p>12"</p>	<p>None</p>
<p><b>Emerald Design Systems Inc.</b> 1043 Stierlin Rd. Mountain View, CA 94043 (415) 965-3300</p> <p>Dave Quinzi Sales Manager</p>	<p><b>GEMstation GEMplot TURBO-CAD</b> (stand-alone workstation, host or cluster controller with attached workstations or graphics terminals, or software only)</p>	<p>\$57k (68020 system with software), \$107k (RISC system with software)</p>	<p>68020, RISC (UNIX V.3)</p>	<p>4-32 MB</p> <p>72-MB-1.1-GB disk</p>	<p>Silicon Graphics' proprietary Geometry Pipeline</p> <p>Hitachi</p>	<p>1280 × 1024 × 32, 1024 × 768 × 32</p> <p>19"</p>	<p>Ethernet TCP/IP, NFS, XNS</p>
<p><b>Hewlett-Packard Co. Logic Systems Division</b> 8245 N. Union Blvd. PO Box 617 Colorado Springs, CO 80901 (303) 590-5530</p> <p>Art Pettis Marketing Communications</p>	<p><b>HP74200 Electric Design System</b> (stand-alone workstation or host with attached workstations or graphics terminals)</p>	<p>\$30k</p>	<p>68010/68020 (HP-UX)</p>	<p>3-8 MB</p>	<p>None</p>	<p>1024 × 768</p>	<p>IEEE-802.3 TCP/IP</p>
<p><b>IC Editors Inc.</b> PO Box 6842 Santa Barbara, CA 93160 (805) 964-1083</p> <p>Mark Stegall</p>	<p><b>ICED</b> (software only)</p>	<p>\$3.3k</p>	<p>PC, XT, AT, or compatible with mouse (MS-DOS 3.X)</p>	<p>640K, uses its own paging</p> <p>ICED pages to disk allowing up to 16M bytes to be used</p>	<p>EGA card with more than 128K bytes</p>	<p>640 × 350 × 8</p> <p>11"-17"</p>	<p>GDS II and CIF interface via PC serial interface</p>
<p><b>Integrated Silicon Design Pty Ltd.</b> 230 North Terrace Adelaide, SA 5000 Australia 61-8-223-5802</p> <p>A.R. Grasso Technical Manager</p>	<p><b>Phase One VLSI Software Suite</b> (software only)</p> <p><b>Phase Two VLSI Software Suite</b> (software only)</p> <p><b>Phase Three VLSI Software Suite</b> (software only)</p>	<p>Not specified</p>	<p>Not specified</p>	<p>Not specified</p>	<p>Not specified</p>	<p>Not specified</p>	<p>Not specified</p>
<p><b>Integrated Silicon Systems Inc. (ISS)</b> Commercial Park West/2327 Englert Dr. PO Box 13665 Research Triangle Park, NC 27709 (919) 361-5814</p> <p>James P. Poitras President</p>	<p><b>LTL 100th</b> (stand-alone workstation)</p>	<p>\$28k- \$32k</p>	<p>386-based PC, PC AT</p>	<p>6 MB</p> <p>40-130 MB</p>	<p>Aries graphics processor</p>	<p>1024 × 1024 × 4</p> <p>19"</p>	<p>3COM</p>

I/O formats							Artwork database operations							Front-end design tools	Symbolic layout and compaction; module generators
GDS I	GDS II	APL 860	APL 870	CIF	PG	EBES	Sizing	Scaling	Boolean	Design-rule checking	Extraction	ERC	Netlist compare		
○	●	○	○	○	○	○	●	●	●	Hierarchical	Pad patterns for chip and package-mounted devices, thin-film resistors	○	●	TOUCHSTONE and MWSPICE circuit simulators	None
●	●	●	○	●	●	●	●	●	●	Batch, incremental, on-line (TURBO-CAD), hierarchical (Q2/88)	Output during layout synthesis as SPICE netlist with parasitics included (TURBO-CAD)	○	●	PC-based schematic capture; logic synthesis; circuit synthesis; CMOS transistor-level netlist generation; timing analysis; support for Silvar-Lisco's SDS and Cal-MP formats	Automatic, self-compacting layout, design-rule-driven; cell generation from Boolean or transistor descriptions; automatic pad layout; RAM generator; symbolic editor; netlist comparison
○	●	○	○	●	○	○	○	○	○	Batch, on-line (not during layout)	Connectivity; transistor, R, and C parameters	●	●	HP design capture system; HILO-3 logic simulator; Analog Design Tools circuit simulation tools	Sticks editor; cell compilers
○	●	○	○	●	○	○	●	●	○	None	None	○	○	None	Sticks editor
○	○	○	○	●	○	○	●	●	○	Interactive; batch (whole design or on a window)	Connectivity, substrate connection, length, width, area of drain and source, perimeter of drain and sources; capacitance: active layers to substrate	●	○	Probe (in-house) full voltage/current circuit simulation; system simulation and specification	Sticks editor; compaction; cell compilation from netlist
○	●	○	○	○	○	○	●	●	●	Available through C interface	Available through C interface	●	●	None	CMOS cell compiler

Directory of IC Layout Systems (continued)

Company contact	System name and configuration	Typical cost	Hardware configuration				
			CPU (operating system)	Main memory secondary storage	Graphics processor and memory	Display resolution size	Local-area network
<b>Intergraph Corp.</b> 1 Madison Industrial Pk. Huntsville, AL 35805 (205) 772-2000  Gary Staley CAE Product Marketing Manager	<b>TANCELL</b> (stand-alone workstation, or host or cluster controller with attached workstations or graphics terminals)	\$55k (work station); \$120k (VAX)	Intergraph 200; MicroVAX II; VAX 11/780, 785, 8600, 8800 (VMS); InterPro 32C workstations	8-32 MB  150 MB	InterPro 32, InterPro 32C  4 MB	1184 × 884  15", 19"	IEEE-802.3 (Ethernet)
<b>Matra Design Semiconductor</b> 2840 San Tomas Expressway Santa Clara, CA 95051 (408) 986-9000  Pradip Madan Vice President, Marketing and Sales	<b>GATEAID PLUS/PC</b> (software only)	\$4.9k	PC XT, AT (DOS 2.0); VAX (VMS)	640 KB  Hard disk recommended	CSI Artist I graphics controller	640 × 480	None
	<b>GATEAID II</b> (software only)	\$95k	VAX 8600, 8650, 11/750, 11/780, or MicroVAX II (VMS)	2 MB (minimum recommended)  200-MB disk	Tektronix 41XX series (or compatible)	640 × 480 × 4	None
<b>Mentor Graphics Corp.</b> 8500 SW Creekside Pl. Beaverton, OR 97005 (503) 626-7000  Brain Kiernan Director of Corporate Communications	<b>Chip Station</b> (stand-alone workstation)	\$50k (DN-3000)	68020/68881 (12-25 MHz) (AEGIS or UNIX BSD 4.2/ System V)	4-32 MB  170-380 MB, 60-MB cartridge tape	Apollo  1 MB	1024 × 800 × 4 (DN3000); 1024 × 800 × 8 (DN4000)	Domain or Ethernet TCP/IP
	<b>Cell Station</b> (stand-alone workstation)	\$90k (DN-4000)				15", 19"	
	<b>Gate Station</b> (stand-alone workstation)						
<b>MIETEC</b> Raketstraat 62 1130 Brussels, Belgium (32) 2-242-5010  Mike Butterworth USIC Business Manager	<b>Made</b> (host or cluster controller with attached workstations or graphics terminals)	\$20k +	VAX 8530, VAX 750, MicroVAX II, VAXStation 2000 (VMS)	5 + MB  140 + MB	GPX, VAXstation 2000, Tektronix	480 × 640, 1000 × 1200  19"	LAVC, Ethernet
<b>Racal-Redac Ltd.</b> Tewkesbury Gloucestershire GL20 8HE, UK (011) 44 684 294161  John Martin Marketing Manager	<b>Isis</b> (stand-alone workstation)	\$95k	MicroVAX II, GPX (MicroVMS)	3 MB  50 MB	Dedicated GPX	1024 × 864  19"	Ethernet
<b>Scientific Calculations Inc.</b> 7796 Victor-Mendon Rd. PO Box H Fishers, NY 14453 (716) 924-9303  David Marini Vice President of Sales	<b>MEDS</b> (host with attached stations or software only)	\$50k (software)	VAX 11/780, 11/785, 8600, (VMS)  MicroVAX II (VMS)	6 MB  500-MB disk  9 MB  213-MB disk	Megatek 7250 and VT100  64 KB	512 × 512 × 4  19"	Ethernet
	<b>Place and Route</b> (software only)	\$25k-\$150k	DEC (ULTRIX)	4 + MB	DEC GPX	864 × 1024	Ethernet, DECnet
				Apollo (Domain)	130 MB plus disk	8-plane graphics	1024 × 1024
			Sun (UNIX)		Standard color graphics	910 × 1152	Ethernet
			Masscomp		IGP or Aurora	600 × 832, 910 × 1152	Ethernet

I/O formats							Artwork database operations							Front-end design tools	Symbolic layout and compaction; module generators
GDS I	GDS II	APL 860	APL 870	CIF	PG	EBES	Sizing	Scaling	Boolean	Design-rule checking	Extraction	ERC	Netlist compare		
○	●	○	○	○	○	○	○	○	○	Batch, incremental, on-line	R/C tree interconnect delay analysis, back annotation	●	●	Schematic capture: Intergraph EDS, EDIF; logic simulation: HILO-3, TEGAS (TDL), ILOGS/SILOS	Compactor type: batch and interactive
○	●	○	○	○	●	○	○	○	●	Batch, on-line (limited)	R and C parameters	●	●	ORCAD schematic capture; logic simulation; graphical analyzer; layout extractor; testability program; fault simulator	Sticks editor; user-defined SRAM compilers
○	●	○	○	●	●	●	●	●	●	Batch via Dracula II and III, on-line, and hierarchical via Dracula III	Transistor type, size, and connectivity: Dracula, R, and C parameters via Dracula	●	●	Schematic capture is NETED; QUICKSIM logic simulator; MSPICE, MSIMON circuit simulators	None
●	●	○	○	○	○	○	●	●	●	MASKAP; ECAD, batch, on-line	Transistor type, size, and connectivity: MASKAP; ECAD; capacitance	●	●	SDS, ACE, GED schematic capture; logic simulation; circuit simulation; mixed-mode multilevel simulator	PLA, ROM, RAM, switched-capacitor filter compilers
●	●	○	●	●	●	●	●	●	●	Interactive, batch, incremental, on-line, hierarchical	Connectivity; transistor, R, and C parameters; track length	●	●	Schematic capture; behavioral modeling; mixed-mode simulation	Sticks symbolic layout
●	●	○	○	○	●	●	○	●	○	(Correct-by-construction approach)	Connectivity	○	○	(User interface)	Interactive symbol editor
○	●	○	○	○	○	○	●	●	●	Batch, incremental, on-line, hierarchical pattern recognition	Connectivity; transistor, R, and C parameters	○	●	Schematic capture; SILOS; SPICE; timing analysis; STL; SCOAP	RAM, ROM, PLA, and ALU module generators

Directory of IC Layout Systems (continued)

Company contact	System name and configuration	Typical cost	Hardware configuration				
			CPU (operating system)	Main memory secondary storage	Graphics processor and memory	Display resolution size	Local-area network
<b>Silicon Compiler Systems Corp.</b> 2045 Hamilton Ave. San Jose, CA 95125 (408) 371-2900  Jim Griffeth Marketing Manager	<b>GDT</b> (software only)	Not specified	Sun (UNIX); Apollo (Domain IX); DEC VAX- station II, GPX (VMS)	4 MB minimum  80 MB for 10,000-transistor design	Not specified  4 and 8 planes	1152 × 900, 1280 × 1024, or 1024 × 864; 8 bit planes  Varies; typically 13", 15", or 19"	Ethernet, Domain
<b>Silvar-Lisco</b> 1080 Marsh Rd. Menlo Park, CA 94025 (415) 324-0700  Dirk Wauters Director, ICD Product Marketing	<b>PRINCESS</b>  <b>DVP</b>  (Both: stand-alone workstation, host or cluster controller with attached workstations or graphics terminals, or software only)	\$25k (software only)  \$50k (software only)	VAXstation (VMS), IBM (VM/CMS), Apollo (AEGIS), Sun (UNIX)	4–16 MB  70–300-MB disk	Digital GPX and 4125, IBM 5080, Apollo, Sun	1024 × 1024, 910 × 1152 (Sun)  19"	DECnet, Ethernet, Domain
<b>Tangent Systems Corp.</b> 2840 San Tomas Expway. Suite 200 Santa Clara, CA 95051 (408) 980-0600  John Seaton Manager of Product Marketing	<b>TANCELL</b> (stand-alone workstation, host or cluster controller with attached workstations or graphics terminals, or software only)	\$55k (work station), \$120k (main frame)	Apollo (Domain), Intergraph InterPro 32C (UNIX); VAX, GPX (VMS); Sun (UNIX)	4 MB  150 MB	Intergraph InterPro 32	15", 19"	Ethernet, Domain
<b>Tektronix CAE Systems</b> PO Box 4600 Beaverton, OR 97076 (503) 629-1036  Ron Workman Division Marketing Manager	<b>Full Custom WorkSystem</b> (stand-alone workstation, host with attached stations, or software only)	\$40k– \$150k	VAX 8000 series, MicroVAX (VMS)	5+ MB  71+ MB	Tektronix 4125	1280 × 1024 × 8  19"	DECnet
<b>Valid Logic Systems Inc.</b> 2820 Orchard Pkwy. San Jose, CA 95134 (408) 432-9400  Donna Rigali IC Product Marketing Manager	<b>Mask Design System,</b> <b>IC Design System,</b> <b>Silicon Design System</b> (stand-alone workstation or software only)	\$20k +	VAXstation II (VMS)  Sun (UNIX)  SCALDstar (68020; UNIX BSD 4.2)	5 MB  71 MB  5–8 MB  280 MB  2–12 MB  70 MB–20 GB	GPX  Sun graphics processor  68020-based parallel processor	1025 × 864 color 19"  1152 × 900 color 19"  1024 × 800 dual color/mono 19"	DECnet, LAVC, Ethernet TCP/IP  NFS, Ethernet TCP/IP  Ethernet TCP/IP
<b>VLSI Technology Inc.</b> 1109 McKay Dr. San Jose, CA 95131 (408) 434-3000  Bill Murray Tactical Marketing Manager	<b>VTtools</b> (software only; also runs on Ridge, Elxsi, and HP 9000 Model 320)	\$20k– \$140k	VAX 760-8800, MicroVAX (VMS)  Apollo (AEGIS)  Sun 3 (UNIX)	4–8 MB 100 MB	AED or Tektronix 4115, 4125  Standard configurations  Standard configurations	1280 × 767, 1280 × 1024  1024 × 800, 1280 × 1024  1152 × 910,	DECnet, Ethernet TCP/IP, Domain

I/O formats							Artwork database operations							Front-end design tools	Symbolic layout and compaction; module generators
GDS I	GDS II	APL 860	APL 870	CIF	PG	EBES	Sizing	Scaling	Boolean	Design-rule checking	Extraction	ERC	Netlist compare		
○	●	○	○	○	○	○	●	●	○	Batch, on-line, hierarchical	Transistor type, size, connectivity, capacitance, resistance stored in the L database—no extraction necessary	●	○	Interactive schematic and layout editor; mixed-mode analog and digital logic simulator; behavioral simulation; fault simulation; timing analysis; test vector translation for Sentry and IMS	Mask-level symbolic layout; virtual grid compaction; RAM, ROM, PLA, random logic compilers
●	●	●	●	●	●	●	●	●	●	Batch, incremental, on-line, hierarchical	Connectivity; transistor, R, and C parameters	●	●	SDS schematic capture; LOGIX-SL (logic), ANDI/SWAP (mixed-mode), and HELIX (behavioral) simulators; interface to SPICE, Zycad, HILO, and TSSI	Procedural language for module generation (e.g., ROM, RAM, PLA)
○	●	○	○	○	○	○	○	○	○	Batch, incremental, on-line	Full parameter extraction of final routing	●	●	Netlist interface to HILO-3, TEGAS (TDL), SILOS logic simulators, plus EDIF	None
○	●	○	○	○	●	●	●	●	●	Dracula II batch	Connectivity; transistor, R, and C parameters	●	●	Schematic capture, logic simulation, circuit simulation	None
○	●	○	○	●	●	●	●	●	●	Batch, incremental, on-line, hierarchical	Connectivity; transistor, R, and C parameters	●	●	ValidGED schematic capture; ValidSIM logic simulator; PRECISE circuit simulator; TIMEMILL switch-level simulator and critical path analyzer	Symbolic interconnectivity editor, compactor, and floor-planner; placement and routing; MSI/SSI, FIFO, PLA, RAM, ROM, datapath, and other compilers
○	●	○	○	●	○	○	○	○	○	Batch, on-line	Connectivity; transistor and C parameters	●	●	VTIschematic, Daisy schematic; VTIsim mixed-mode simulator	Sticks, composition, autorouting, compaction; RAM, ROM, PLA, multiplier, datapath, and other compilers

# Directory of Printed Circuit Board Layout Systems

Vendor Contact	System name, typical cost	Mainframe	Workstation	Software only	Hardware environment • System support • Memory support • Graphics support	• Design entry • Simulation	• ERC • DRC • Extraction	Output formats • Printer/plotter • Assembler • Tester; other
<b>Academi Systems Inc.</b> 2418 Armstrong St. Livermore, CA 94539 (415) 449-3294 Cynthia Peddie Vice President	<b>Academi 56000</b> \$62.5k <b>Academi 54000</b> \$39.8k	—	•	•	• DEC with attached processors • 1-MB main memory • Tablet input; 19" display with 640 × 512 resolution	• Case, FutureNet interfaces • —	• — • Layout rule checker (batch) • CAE interface for circuit extractor	• Calcomp, DEC, HI, HP • Generic NC interface • —
<b>Accel Technologies Inc.</b> 7358 Trade St. San Diego, CA 92121 (619) 695-2000 Ray Schnorr Vice President of Marketing	<b>Tango-PCB</b> <b>Tango-Route</b> \$0.5k each	—	—	•	• PC XT, AT, PS/2 (DOS 2.X); local-area network • 256-KB main memory and 2 disk drives • Mouse input; 13" CGA or EGA monitor	• Tango-Schematic; • Omation, OrCAD, interfaces • —	• Electrical rule checker with Tango-Schematic • Design rule checker by Tango-Route • —	• Epson, DM-PL, Gerber, HP-PL • Excellon • —
<b>Applicon</b> 4251 Plymouth Rd. PO Box 986 Ann Arbor, MI 48106 (313) 995-6000 Brian F. Barton Director of Electronics	<b>Bravo 3</b> \$50k	•	•	—	• VAX (VMS); Sun (UNIX); Applicon Graphicstations; Ethernet, DECNet • 5-MB main memory and 350-MB hard disk • Mouse, tablet input; 19" Applicon monitor with up to 1536 × 1197 × 8 resolution	• Applicon design capture • CADAT; SPICE	• High-speed electrical rule checker (batch) • Design rule checker • Circuit extractor	• Generic plotter interface • Generic assembly equipment interface • Generic tester interface
<b>Aptos Systems Corp.</b> 10 Victor Square Scotts Valley, CA 95066 (408) 438-2199 John Roth President	<b>CRITERION</b> \$1.5k <b>RGRAPH</b> (with graphics coprocessor) \$10k	—	—	•	• PC AT, 80386-based PC; local area-network • 640-KB main memory and hard disk • Mouse or digitizer input; 13" monitor with EGA (CRITERION); 19" monitor with 1024 × 768 resolution (RGRAPH)	• Aptos schematic capture; • FutureNet, Mentor, OrCAD, PCAD interfaces • TEGAS and PSPICE interfaces	• Common node checker • Layout rule checker • —	• Calcomp, Epson FX series, Gerber, HI, HP, Zeta • Excellon • —
<b>Automated Systems Inc.</b> 1505 Commerce Ave. Brookfield, WI 53005 (414) 784-6400 M. Wilson Marketing Manager	<b>Prance</b> \$150k	—	—	•	• IBM 43xx (VM/SP) • 8-MB main memory and 635-MB hard disk • Keyboard, tablet input; IBM 5080 19" monitor with 1000 × 1000 × 4 resolution	• Case, Daisy, FutureNet, Mentor, PCAD, Valid, and Viewlogic interfaces • —	• On-line electrical rule checker • Layout rule checker • —	• Benson, Calcomp, Versatec • Excellon, Trudrill; generic output for assembly • —
<b>Bishop Graphics CAD Systems Corp.</b> 5388 Sterling Center Dr. Westlake Village, CA 91359 (818) 991-2600 Robert Reiss Marketing Manager	<b>PATHFINDER</b> \$5k <b>QuikCircuit</b> \$0.5k (see Douglas Electronics)	—	—	•	• PC XT (MS-DOS); 68000 coprocessor card • 640-KB main memory, 1.2-MB floppy, and 20-MB hard disk • Mouse, tablet, and tracker ball inputs; Hercules, CGA, EGA, VGA, and PGA monitors	• PATHFINDER schematic capture • —	• — • — • —	• Generic plotter interface • — • —
<b>Cadam Inc.</b> 1935 N. Buena Vista St. Burbank, CA 91504 (818) 841-9470 Alan Cohen Electrical Products Marketing	<b>IPC</b> (Interactive Prance CADAM) \$116k per CPU	•	—	—	• Any IBM mainframe or compatibles (VM or MVS) • Standard memory • Mouse or tablet input; IBM 5080 scopes or compatibles	• Interface through CADEX • CADAT; two-way netlist translation	• Tolerance checker (batch and on-line) • Layout rule checker • CADEX circuit extractor; thermal analysis	• Benson, Calcomp, Gerber, HP, Versatec • Generic NC interface; APT Data; generic assembly interface • Generic ATE interface
<b>Cadnetix Corp.</b> 5757 Central Ave. Boulder, CO 80301 (303) 444-8075 Greg Skomp Manager of Marketing and Communications	<b>CDX-50,000S</b> \$69.9k <b>CDX-5000S</b> \$54k <b>CDX-56,000SP</b> \$89.9k	—	•	—	• Sun (UNIX); CDX-50,000S, CDX-5000A (UNIX BSD 4.2); Ethernet • 3.5–4-MB main memory and 3.5–280-MB hard disk • Mouse, keyboard input; bit-slice graphics processor; 19" color monitor with 1024 × 800 resolution	• CDX PC schematic entry • CDX digital design environment; enhanced SABER circuit simulator; simulation accelerator; physical modeler	• Electrical rule checker (on-line and batch) • Design rule checker (on-line with ECL applications)	• Benson, Calcomp, Gerber, HI, HP-GL • Excellon; Amistar, Dynapert, Universal; EDIF, IGES, IPC-350, DQL • Ditmco, Factron, GenRad, Integr-Test, Marconi, ATG interfaces



<ul style="list-style-type: none"> <li>• Initial placement tools</li> <li>• Placement improvement tools</li> </ul>	Routing tools	<ul style="list-style-type: none"> <li>• Max board size</li> <li>• Layer count</li> <li>• Grid sizes</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic libraries</li> <li>• Library support</li> <li>• Package libraries</li> </ul>	Surface-mount design support
<ul style="list-style-type: none"> <li>• Manual and automatic initial preplacement; rat's-nest display</li> <li>• Automatic component, pin, and gate swapping</li> </ul>	Signal prerouting; priority, channel routing; compaction; reentrant; 45°; keep-out zones	<ul style="list-style-type: none"> <li>• 32" × 32"</li> <li>• 64</li> <li>• Any in 1-mil increments</li> </ul>	<ul style="list-style-type: none"> <li>• CAE interface</li> <li>• New parts can be defined in metric units and mils; new library entries are formed for each package of each device</li> <li>• All typical package types needed for basic design are included.</li> </ul>	Flip-flip screen; flip components between sides instantaneously; blind and buried vias
<ul style="list-style-type: none"> <li>• Manual preplacement; rat's-nest display</li> <li>• Manual reconnection</li> </ul>	Signal prerouting; proprietary routing algorithm; reentrant; 45° routing; keep-out zones	<ul style="list-style-type: none"> <li>• 39" × 19"</li> <li>• 2 signal, 2 power, 2 ground, silkscreen overlay</li> <li>• 1, 5, 10, 25, 50, 100, 125, 156, 200 mils</li> </ul>	<ul style="list-style-type: none"> <li>• Analog, 1500 schematic symbols; digital, 1675 schematic symbols (in 15 libraries)</li> <li>• New physical parts can be defined graphically; parts can be defined in mils only; new library entries are formed for each package of each device</li> <li>• Package options: DIPs, 11; edge connectors, 2; connectors, 10; axial, 8; diodes, 2; other, 30</li> </ul>	Surface-mount packages: 60 pin options
<ul style="list-style-type: none"> <li>• Constructive and force-directed initial placement; rat's-nest display</li> <li>• Component, pin, and gate swapping</li> </ul>	Signal prerouting; channel, maze, priority multilayer router on 5-, 10-, 20-, 25-, 50-, or 100-mil grid; compaction; rip-up; 45° routing; keep-out zones	<ul style="list-style-type: none"> <li>• Any board size</li> <li>• 32 layers</li> <li>• Manual routing grid on any grid size</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic library: TI TTL, 1882; Motorola STTL, 280; Motorola HCMOS, 99; Intel microprocessors, 92</li> <li>• New physical parts can be defined in metric units and mils; new library entries are formed for each package of each device</li> <li>• Package options: DIPs, 15; also flat packs, transistors, can-type ICs</li> </ul>	User-defined surface-mount land patterns; board can be viewed from any angle; double-sided boards; blind and buried vias
<ul style="list-style-type: none"> <li>• Manual placement; rat's-nest display</li> <li>• —</li> </ul>	Signal prerouting; priority, channel, maze, expert system routing; keep-out zones for wires	<ul style="list-style-type: none"> <li>• 64" × 64"</li> <li>• 50</li> <li>• 25, 50, 100 mils; fineline; microline; staggered</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic symbols: TTL, 107; ECL, 77; CMOS, 130; analog, 71; discretes, 84; microprocessors, 84</li> <li>• New physical parts can be defined in metric units and mils; new library entries are formed for each package of each device</li> <li>• Package options: passives, 2; DIPs, 256; SIPs, 256</li> </ul>	129 surface-mount schematic symbols and packages (J-lead and gull); blind and buried vias
<ul style="list-style-type: none"> <li>• Force-directed and min-cut initial placement</li> <li>• Component, pin, and gate swapping; Steinberg placement improvement</li> </ul>	Maze 8-layer autorouting; rip-up; "squeeze-through" routing	<ul style="list-style-type: none"> <li>• Board size function of grid (25" × 25" board with 25-mil grid)</li> <li>• 8 layers</li> <li>• Any in 1-mil increments</li> </ul>	<ul style="list-style-type: none"> <li>• Standard packaging</li> <li>• New physical parts defined in mils only; new library entry for each package of each device.</li> <li>• Standard package options</li> </ul>	Flip-flip screen; double-sided boards; blind and buried vias
<ul style="list-style-type: none"> <li>• Manual preplacement; collapse and drag; matrix placement; WYSIWYG rat's-nest display</li> <li>• —</li> </ul>	Signal prerouting; priority, channel, maze layer-pair reentrant router; rip-up; real-time 45° routing; keep-out zones for wires, vias, and parts	<ul style="list-style-type: none"> <li>• Unlimited</li> <li>• Unlimited with layer pair routing</li> <li>• 12.5, 25 mils</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic symbols: TTL, 74; ALS, 146; AS, 129; F, 150; LS, 256; S, 74; electromechanical (board outlines); Intel microprocessors, 132; Motorola microprocessors, 137; Fairchild 100K ECL, 45; SMDs; PALs (MMI)</li> <li>• Physical parts can be defined; parts can be defined in metric units and mils; new library entries are formed for each package of each device</li> <li>• Package options not given</li> </ul>	—
<ul style="list-style-type: none"> <li>• Min-cut preplacement with user-specified restrictions; rat's-nest display and histograms</li> <li>• Interactive and automatic component, pin, and gate swapping</li> </ul>	Signal prerouting; priority, maze, channel reentrant router on any grid and 1–20 layers; compaction; rip-up; ECL design rules supported; 45° routing; keep-out zones	<ul style="list-style-type: none"> <li>• Board size function of grid (50" × 50" board with 50-mil grid)</li> <li>• 20+ up to 99 signal planes</li> <li>• Any in 1-mil increments</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic symbols: discrete, 6500; digital, 2800; ANSI, 1200</li> <li>• New physical parts can be defined in metric units and mils; schematic and physical libraries are separate</li> <li>• Package options: DIPs, 30; SMDs, 30; analog, 60</li> </ul>	Flip-flip screen and double-sided boards; blind and buried vias
<ul style="list-style-type: none"> <li>• Constructive and min-cut preplacement, netlist-driven; rat's-nest display</li> <li>• Simulated annealing; component, pin, and gate swapping; automatic decoupling capacitor assignment</li> </ul>	Signal prerouting; priority, maze, heuristic (for memory arrays), reentrant router with up to 8 layers on any grid; rip-up; ECL layout rule adherence; 45° routing; keep-out zones	<ul style="list-style-type: none"> <li>• 34" × 22"</li> <li>• 24 trace, 24 draft</li> <li>• Any combination of grids on "plastic grid"</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic library: basic, 83 pads, 248 components, 298 shapes; CAE, 699; CAD, 2255 components; 74 TTL, 789; 54 TTL, 791 components; commercial CMOS, 235; military CMOS, 224 components; ECL, 75; advanced functions, 89; physical modeling, 55</li> <li>• New physical parts can be defined in metric units and mils; separate library entry not required for each package option of each function.</li> <li>• Package options: capacitors, 35; connectors, 23; discretes, 45; DIPs, 21; SIPs, 7; PGAs, 6</li> </ul>	Blind and buried vias

Directory of Printed Circuit Board Layout Systems (Continued)

Vendor Contact	System name, typical cost	Mainframe	Workstation	Software only	Hardware environment • System support • Memory support • Graphics support	Design entry • Simulation	ERC • DRC • Extraction	Output formats • Printer/plotter • Assembler • Tester; other
<b>CAD Software Inc.</b> Box 1142 Littleton, MA 01460 (617) 486-9521 Joe Lapoint	<b>PADS-PCB</b> \$1k-\$2.9k	—	—	•	• PC XT, AT, 386, PS/2 • 512-640-KB main memory and 10-20-MB hard disk • Mouse input; 12" or 19" display with EGA or GEM	• PADS-CAE • —	• Schematic vs. layout, AirCap • Batch design rule checker • —	• Gerber, HI, HP, and generic matrix plotter interface • Excellon • —
<b>Calay Systems Inc.</b> 2698 White Rd. Irvine, CA 92714 (714) 863-1700 Beverley J. Lages Marketing Communications Manager	<b>Design Automation Series</b> \$65k-\$120k	—	•	—	• DEC with Q-Bus; routing accelerator; TCP/IP • 5-MB main memory and 42-MB hard disk • Digitizer input; graphics coprocessor with 4-MB RAM; 19" display with 640 x 480 x 4 resolution	• Case, Daisy, FutureNet, Mentor, PCAD, Viewlogic interfaces • CADAT interface; SPICE interface	• On-line electrical rule checker • Batch design rule checker; ECL checker • —	• Calcomp CC907, Gerber 4000, HP-GL • Excellon, Trudrill, generic drill interface; Fuji, Panasonic pick-and-place; Dynapert, Universal interfaces • GenRad, Zehntel interfaces
<b>Calma Co.</b> 501 Sycamore Dr. Milpitas, CA 95035 (408) 434-4056 Phil Arana PCB Product Manager	<b>Board Series</b> \$21k-\$60k	—	•	•	• Apollo (AEGIS), DSP4000 server, Domain and Ethernet • 4-32-MB main memory and 155-380-MB hard disk • Mouse input; 15" or 19" display with 1024 x 800 x 8 resolution	• Explorer (Calma); netlist interface • TEXSIM; HSPICE	• On-line electrical rule checker • On-line design rule checker • 3D wireframe and solid-modeling interfaces	• Aristo, Calcomp, Gerber, HP, and generic interface • Excellon, Digital, Trudrill, generic interfaces; Dynapert, Panasonic, Universal interfaces • Generic tester interface; Everett Charles, Test Systems, Trace Instruments bare-board interfaces
<b>Case Technology Inc.</b> 2141 Landings Dr. Mountain View, CA 94043 (415) 962-1440 Melanie King Manager, Marketing Communications (415) 962-1440	<b>Vanguard PC Design System</b> \$4.25k-\$12.5k	—	•	•	• Compaq 386; Sun 3/260; MicroVAX 2000/GPX; Ethernet • 4-16-MB main memory and 30-140-MB hard disk • Mouse, keyboard input; EGA; Microfield Graphics monitor with 1280 x 1024 x 8 resolution	• STELLAR (Case) • CADAT; SILOS; SALT; HILO; HSPICE; IG-SPICE; Touchstone; PSPICE; LDS; timing verifier	• Electrical rule checker from menu • Consistency, keep-out checker, parameterizable design rule checker (Case) • —	• Gerber, HP • Excellon drill; Universal pick-and-place; Techno insertion • Zehntel interface
<b>Computervision Corp.</b> 100 Crosby Dr. Bedford, MA 01734 Al Lipinski Director, Electronic Marketing (617) 275-1800	<b>Autoboard SMT CADDstation System</b> \$80k	—	•	—	• 68020 (UNIX); Ethernet TCP/IP, NFS • 8-32 MB main memory, 170-MB hard disks, and 280-MB 8" SMD drive • Mouse, keyboard; 19" display size with 1152 x 900 x 8 resolution	• Schematic Design (Computervision) • CADAT logic simulation (HHB Systems); SABER circuit simulation (Analogy Inc.); thermal analysis (Pacific Numerix); HILO-3 (GenRad); SPICE 2G.6	• On-line • — • —	• Gerber, Quest, Benson, Calcomp, Versatec, Hewlett-Packard • Excellon, Posalaz, Siemens • Marconi, Panasonic, Universal
<b>Control Data Corp.</b> 1450 Energy Park Dr. M/S ETC235 St. Paul, MN 55108 (612) 642-3845 John T. Willey Manager, Electronics	<b>ICEM Electronics</b> \$14k-\$75k	•	•	•	• PC-XT, (MS-DOS); Cyber 910, (UNIX); TCP/IP • 640-KB main memory and 4-MB hard disk (PC XT); 30-MB main memory and 70-MB hard disk (Cyber) • Mouse input; 12"/17" monitor with 1024 x 700 x 8 resolution	• ED-Schematics, ICEM DDN (CDC) • SALT (The CAD Group); ASPEC (CDC)	• Batch electrical rule checker (CDC) • On-line and batch layout rule check (CDC); keep-out zones supported • —	• Calcomp, Gerber, HP, Versatec • Excellon drill; generic NC format • Fairchild, GenRad
<b>Daisy Systems Corp.</b> 700 Middlefield Rd. Mountain View, CA 94039 (415) 960-0123 Pat Meyer Product Marketing Manager	<b>Boardmaster STAR Router</b> \$22.5k-\$28.5k	—	•	•	• 80286, 80386/DNIX, Ethernet • 4-16-MB main memory and 85-475-MB hard disks • Mouse, tablet input; Daisy hardware, bit-map graphics; 15"/19" monitor with 1024 x 832 x 8 resolution	• DED II, ACE (Daisy) • DLS, MDLS, DSPICE, ADLAB, VLAB (Daisy)	• BoardMaster on-line and batch; layout schematic vs. consistency checker • BoardMaster on-line and batch; keep-out areas • —	• Gerber, HP, Printronix, Versatec • Excellon, Trudrill drill • MultiWire interface

<ul style="list-style-type: none"> <li>• Initial placement</li> <li>• Placement improvement</li> </ul>	Routing tools	<ul style="list-style-type: none"> <li>• Max board size</li> <li>• Layer count</li> <li>• Grid sizes</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic libraries</li> <li>• Library support</li> <li>• Package libraries</li> </ul>	Surface-mount design support
<ul style="list-style-type: none"> <li>• Matrix; force-directed; rat's-nest display</li> <li>• Component, pin, and gate swapping</li> </ul>	Signal prerouting; priority, maze reentrant multilayer router with 1, 5, 10, 20, 25, 50 grids; 45° routing; keep-out zones	<ul style="list-style-type: none"> <li>• 32" × 32"</li> <li>• Unlimited layers</li> <li>• 1–1000 mils in 1-mil increments</li> </ul>	<ul style="list-style-type: none"> <li>• —</li> <li>• Physical parts can be defined</li> <li>• Approximately 1000 parts in 8-128 pins, DIPs, SMDs, and pin-grid arrays</li> </ul>	Blind and buried vias; standard and micro-size vias
<ul style="list-style-type: none"> <li>• Force-directed; min-cut; constructive; proportional-space gridless algorithm; rat's-nest display</li> <li>• —</li> </ul>	Signal prerouting; priority, maze, reentrant multilayer router with user-definable grid; rip-up; ECL layout rule adherence; 45° routing (postprocessed or digital); keep-out zones	<ul style="list-style-type: none"> <li>• 32" × 32"</li> <li>• 16 signal, unlimited power layers</li> <li>• Variable in 1-mil increments</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic library: TTL, 1000; memory, 500; device, 250; analog, 350; ECL, 500; Intel, 200; Motorola, 200</li> <li>• New physical parts can be defined in metric units and mils; new library entries are formed for each package of each device</li> <li>• Package options: rectangular, 28 or 32 pins; small-outline, 8, 14, 16, 20, 24; DIPs, 8, 14, 16, 20, 24, 28, 40; PLCCs, 44 or 68; zigzag, 16 or 20, various discretes, through-holes and connectors</li> </ul>	Flip-flip screen; double-sided boards; blind and buried vias, although not automatically
<ul style="list-style-type: none"> <li>• Constructive; rat's-nest display</li> <li>• Component, pin, and gate swapping</li> </ul>	Signal prerouting; priority, maze, reentrant router with variable grid; rip-up; 45° routing; all keep-out zones	<ul style="list-style-type: none"> <li>• 32" × 32"</li> <li>• 32 layers</li> <li>• Variable in 1-mil increments</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic library: standard, 4500</li> <li>• New physical parts can be defined in metric units and mils; new library entries are formed for each package of each device</li> <li>• Package options: DIPs, 8, 14, 16, 18, 20, 24, 28 etc.; SIPs; flat packs; LCCCs; PLCCs; SOICs</li> </ul>	Flip-flip screen with color, bit offset; double-sided boards; blind and buried vias; TAB; SMT CAM file interfaces; via restriction under devices; thermal pads
<ul style="list-style-type: none"> <li>• Ordered by connectivity and user filter; rat's-nest display</li> <li>• —</li> </ul>	Signal prerouting; priority, channel, maze multilayer reentrant router; compaction; rip-up; ECL layout rule adherence; 45° routing; keep-out zones	<ul style="list-style-type: none"> <li>• 32" × 32"</li> <li>• Layers limited by available memory</li> <li>• Variable from 1 to 500 mils</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic symbols: digital, 4000; analog, 500; other, 500</li> <li>• New physical parts can be defined in metric units and mils; new library entries are formed for each package of each device (normally parts are attributes)</li> <li>• Package options: DIPs; SIPs</li> </ul>	Blind and buried vias
<ul style="list-style-type: none"> <li>• Constructive</li> <li>• Pairwise swapping; automatic pin and gate swapping</li> </ul>	Single-layer, layer-pair, and simultaneous multilayer grids are supported; prerouting of signals; reentrant maze router with extensions to reposition traces and vias; 45° routing; keep-out zones at board and component level	<ul style="list-style-type: none"> <li>• 39" × 39" (999 mm × 999 mm)</li> <li>• 20 signal; 10 power and ground</li> <li>• 1 mil</li> </ul>	<ul style="list-style-type: none"> <li>• —</li> <li>• New physical parts can be defined graphically or in ASCII; parts can be defined in metric units and mils; physical parts are attributes of the schematic or library entries are formed for each package of each device—hierarchical within library (many components can use same package information)</li> <li>• —</li> </ul>	Blind and buried vias
<ul style="list-style-type: none"> <li>• Force-directed, constructive; rat's-nest display</li> <li>• Simulated annealing; automatic component, pin, and gate swapping</li> </ul>	Signal prerouting; priority, channel, maze reentrant layer-pair router; compaction; keep-out zones	<ul style="list-style-type: none"> <li>• 36" × 36"</li> <li>• 20 signal, 40 power ground</li> <li>• 0.1 mil and up</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic libraries: TTL, 1200; CMOS, 500; ECL, 200; Intel/AMD, 200</li> <li>• New physical parts can be defined in metric units and mils; new library entries are formed for each package of each type</li> <li>• Package options: DIPs, 50; discretes, 100; others, 100</li> </ul>	Blind and buried vias
<ul style="list-style-type: none"> <li>• Force-directed; density and/or Manhattan distance is user-definable; fixed-pre-placement; rat's-nest display</li> <li>• Pairwise swapping and multiple swaps with same shape (package)</li> </ul>	Signal prerouting; priority, costed-maze reentrant router on up to 16 layers; ECL layout rule adherence; on-line 45° routing; keep-out zones for wires and vias	<ul style="list-style-type: none"> <li>• 32" × 32"</li> <li>• 255 (BoardMaster); 16 signal, 16 power (Star)</li> <li>• Gridless (BoardMaster); 10, 20, 25, 50, 42-16-42, 40-20-40, 40-10-10-40, 42-8-8-42, 38-12-12-38 (Star)</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic symbols: 74TTL, 398; 54TTL, 398; 4000C, 194; 74HC, 203; 54HC, 175; ECL 10K/10KH/100K, 279/215/118; memory, 360; PLAs/PALs, 64; microprocessors, 232; semicustom devices, 54; basic library, 93</li> <li>• New physical parts can be defined in metric units and mils; new library entries are formed for each package using information extracted from the schematics.</li> <li>• Package options: DIPs, 6–64 pins; SIPs, 8–12; pin-grid arrays, 68–132; discretes, 2–8; all through-hole</li> </ul>	—

## Directory of Printed Circuit Board Layout Systems (Continued)

Vendor Contact	System name, typical cost	Mainframe	Workstation	Software only	Hardware environment • System support • Memory support • Graphics support	• Design entry • Simulation	• ERC • DRC • Extraction	Output formats • Printer/plotter • Assembler • Tester; other
<b>Douglas Electronics Inc.</b> 718 Marina San Leandro, CA 94577 (415) 483-8770 Dana Sattler Marketing Manager	<b>Douglas CAD/CAM</b> \$0.5k (Also available as "QuikCircuit" from Bishop Graphics)	—	—	•	• Apple Macintosh 512K, 512E/Plus/SE, Mac II; Apple Talk local-area network • 512-KB main memory; 400-KB disk drive; hard disk optional • Keyboard, tablet, mouse input; Apple monitor	• — • —	• — • — • —	• Gerber; HI; HP-GL; ImageWriter, Laser-Writer • Douglas drill • —
<b>FutureNet</b> 9310 Topanga Canyon Blvd. Chatsworth, CA 91311 (206) 881-6444 Michael Radovich Public Relations Manager Data I/O Corp.	<b>DASH-PCB</b> \$14k	—	•	•	• PC (PC-DOS 3.X); Opus coprocessor (NS32032); DASH-NET (3Com 3+) • 640-KB main memory plus 2 MB on coprocessor and 80-MB hard disk • Mouse input; EGA; Microfield T4A-768 with 1024 × 768 resolution	• DASH • DC-Plus (CADAT); PSPICE	• Batch electrical rule checker (schematic only) • Design rule checker • —	• Calcomp, Gerber, HP, HI • Excellon • —
<b>Hewlett-Packard Co.</b> 3000 Hanover St. Palo Alto, CA 94304 (800) 367-4772 Regional Inquiries Manager	<b>HP Printed Circuit Design System (HP PCDS)</b> \$59k–\$95.5k	—	•	—	• HP300 + HP-UX; HP800 server; IEEE-802.3 LAN • 4–8-MB main memory and 131–571-MB hard disk • Mouse and tablet input; bit-mapped video bit-slice accelerator; 16", 19" monitor with 1024 × 768 × 8 resolution	• Design Capture System (HP) • Design Verification System, fault simulator, HICHIP hardware modeling system (HP)	• On-line and batch electrical rule checker (HP) • Design rule checker (HP) • —	• HP-GL • Excellon, Trudrill, generic drill interfaces; generic pick and place interfaces • HP3065 board test family; EDIF, IGES interfaces
<b>IBM Corp.</b> 2077 Gateway Place (2nd floor) San Jose, CA 95110 (408) 288-4100 Stafford Johnson Electronic Design Marketing Programs Manager	<b>Circuit Board Design System (CBDS)</b> \$49k	—	—	•	• System 370 architecture (9370, 43xx,30xx) • 8-MB main memory and 50-MB/user hard disk • Mouse, keyboard input; 5080 monitor	• LOKI, CIEDS schematic capture; netlist inputs • CIEDS logic, behavioral, mixed-mode simulator; Logan, PPR-6 simulation analysis	• On-line electrical rule checker • On-line design rule checker • —	• Gerber G7000E • Excellon, Neutral, Trudrill, standard insertion • Fairchild, GenRad interfaces
<b>Interactive CAD Systems</b> 2352 Rambo Court Santa Clara, CA 95054 (408) 970-0852 Eddy Ozomaro President	<b>PROCAD Xtra</b> \$0.6k	—	—	•	• PC XT, AT • 640-KB main memory • Mouse, digitizer, joystick input; 19" monitor with 2048 × 2048 resolution	• Schematic capture included • SILOS, MDL; SPICE, HSPICE, LVS	• ECAD, NCA software • Layout rule checker	• Epson, Gerber, HI, HP Laser-Jet, HP-PL, Toshiba • — • —
<b>Intergraph Corp.</b> 1 Madison Industrial Park Huntsville, AL 35807 (205) 772-2000 Shiv Tasker Product Marketing Manager	<b>Integrated Electronics Design System (IEDS)</b> \$40k–\$70k	•	—	—	• VAX, MicroVAX, Clipper-based InterPro coprocessor; XNS/Ethernet • 9–80-MB main memory and 156-MB–4-GB hard disk • Mouse, digitizer input; 15"/19" monitor with 1184 × 884 × 6 resolution	• HSD (Intergraph) • HILO-3; CSPIICE, ACS (Intergraph)	• HSD on-line and batch electrical rule checker • IEDS, HSD (Intergraph) • —	• Gerber, HP, Optronics, Versatec • Dynapert, Excellon, Panasonic, Oki, Trudrill, Universal • Factron, GenRad, HP, Marconi, Zehntel interfaces; MultiWire
<b>Mentor Graphics Corp.</b> 1940 Zanker Rd. San Jose, CA 95014 (408) 295-1000 Lee Smith Product Marketing Manager	<b>BoardStation</b> \$73.9k	—	•	•	• Apollo 3000, 4000, and DN570 Turbo (AEGIS); Compute Engine global accelerator; Domain/Idea series DBMS • 4–32-MB main memory and 155–348-MB hard disk • Keyboard, mouse input; 19" monitor with 1024 × 800 × 4 resolution	• NETED (Mentor) • QUICKSIM, MSPICE, MSIMON, mixed-mode simulator (Mentor)	• On-line electrical rule checker • On-line design rule checker • Circuit extractor with hierarchical expander	• Calcomp, Gerber, HP, Versatec • Excellon and Trudrill interfaces • GenRad 227X, Zehntel 850 interfaces

<ul style="list-style-type: none"> <li>• Initial placement</li> <li>• Placement improvement</li> </ul>	Routing tools	<ul style="list-style-type: none"> <li>• Max board size</li> <li>• Layer count</li> <li>• Grid sizes</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic libraries</li> <li>• Library support</li> <li>• Package libraries</li> </ul>	Surface-mount design support
<ul style="list-style-type: none"> <li>• Manual only</li> <li>• —</li> </ul>	—	<ul style="list-style-type: none"> <li>• 32" × 32"</li> <li>• 2 signal, 1 power, 1 ground, 1 silkscreen, 1 soldermask</li> <li>• 1–1000 mils in 1-mil increments</li> </ul>	<ul style="list-style-type: none"> <li>• —</li> <li>• Physical parts defined in mils only</li> <li>• User-created</li> </ul>	—
<ul style="list-style-type: none"> <li>• Rat's-nest display</li> <li>• Automatic and manual pin and gate swapping</li> </ul>	Signal prerouting; priority, channel, maze routing in multiple layer pairs; compaction; 45° routing; keep-out zones	<ul style="list-style-type: none"> <li>• 32" × 32"</li> <li>• 8 signal and 2 power or 10 signal layers</li> <li>• Greater than or equal to 5 mils</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic symbols: CMOS, TTL, ECL, memory, Intel, Motorola, discrete, IEEE</li> <li>• New physical parts can be defined; new library entries are formed for each package of each device</li> <li>• Package options: standard</li> </ul>	—
<ul style="list-style-type: none"> <li>• Constructive; force-directed placement by device classification; rat's-nest display</li> <li>• Force-directed pairwise relaxation; automatic pin and gate swapping</li> </ul>	Signal prerouting; maze reentrant router on 4 layers; 45° routing; keep-out zones for wires and vias	<ul style="list-style-type: none"> <li>• 36" × 36"</li> <li>• 99 layers</li> <li>• User-defined grids</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic libraries: TTL, 2698; MOS, 576; ECL, 131; microprocessors and PLDs, 163</li> <li>• New physical parts can be defined; new library entries are formed for each package of each device</li> <li>• Package options: DIPs, 11; SIPs, 7; SOICs, 6; PLCCs, 5; PGAs, 6; discretcs</li> </ul>	Blind and buried vias; SMD discrete device library
<ul style="list-style-type: none"> <li>• Constructive; force-directed placement with user-specified weights; rat's-nest display</li> <li>• Automatic and interactive component, gate, and pin swapping across windows</li> </ul>	Signal prerouting; priority, channel reentrant router on 4 layers; 45° routing; keep-out zones for wires and vias	<ul style="list-style-type: none"> <li>• 64" × 64" board</li> <li>• 256 layers</li> <li>• 1 mil up</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic symbols: 7500 elements</li> <li>• New physical parts can be defined in mils only; each symbol requires only one symbol in the database and multiple package versions are defined for it</li> <li>• Package options: DIPs, SIPs, axial, radial, mechanical, pin-grid arrays, SMDs, TAB</li> </ul>	Flip-flip screen; double-sided boards; hidden and buried via support; SMD, TAB package libraries; SMT CAM file interfaces and hybrid support
<ul style="list-style-type: none"> <li>• Manual placement; step and repeat (fixed placement); rat's-nest display</li> <li>• Pin swapping</li> </ul>	Signal prerouting; priority, channel, look-out reentrant router on up to 25 layer pairs; compaction; some ECL compatibility; 45° routing; keep-out zones	<ul style="list-style-type: none"> <li>• 64" × 64"</li> <li>• Unlimited</li> <li>• 1-mil resolution</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic libraries: TTL, 200; CMOS, 150; linear, 150; other, 50; microprocessors, 100</li> <li>• New physical parts can be defined in metric units and mils; new library entries are formed for each package of each device</li> <li>• Package options: DIPs, 20; SIPs, 10; through-hole, 10</li> </ul>	Blind and buried vias
<ul style="list-style-type: none"> <li>• Constructive (includes "what if" algorithms); force-directed; min-cut with autoplacement on both board sides; rat's-nest display generated for minimum length or for ECL</li> <li>• Component, pin, and gate swapping</li> </ul>	Signal prerouting; priority, channel, maze multilayer router; compaction; rip-up; ECL layout rule adherence; 45° routing (out of pin and bends); keep-out zones for wires and vias	<ul style="list-style-type: none"> <li>• 65" × 65"</li> <li>• 16 routing layers, 2016 drawing layers</li> <li>• Grid-free</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic symbols: TTL, 1000; military, 1000; CMOS, analog, other, 500</li> <li>• New physical parts can be defined in metric units and mils; each device may refer to a single physical description (no data redundancy)</li> <li>• Package options: DIPs, SIPs, SMDs, and ZIPs, 3000</li> </ul>	Blind and buried vias; automatic placement on both sides of the board; hybrid support
<ul style="list-style-type: none"> <li>• Random; constructive; force-directed; two-sided; rat's-nest display</li> <li>• Automatic and interactive component, gate and pin swapping</li> </ul>	Signal prerouting; priority, channel, maze, memory reentrant layer pair router; compaction simulated through cost controls; ECL layout rule adherence; 45° routing; keep-outs for components, wires, and vias	<ul style="list-style-type: none"> <li>• 100" × 100"</li> <li>• 255 layers</li> <li>• Automatic generation of grid from design rules, with 0.1 mil minimum in variable grid range; no limit to pin count or pins per net</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic symbols: 54TTL, 74TTL, 74HC, 54HC, HCT, ECL 10K and 100K, memory, PLDs, PALs, PLAs—3000</li> <li>• New physical parts can be defined in metric units and mils; symbols are automatically packaged during layout</li> <li>• Package options: 90</li> </ul>	Blind and buried vias; user-definable surface-mount package and land-pattern libraries; placement on both sides of double-sided boards

## Directory of Printed Circuit Board Layout Systems (Continued)

Vendor Contact	System name, typical cost	Mainframe	Workstation	Software only	Hardware environment • System support • Memory support • Graphics support	• Design entry • Simulation	• ERC • DRC • Extraction	Output formats • Printer/plotter • Assembler • Tester; other
<b>Modula Corp.</b> 1673 W. 820 N. Provo, UT 84601 (801) 375-7400  Wally Marsden Chief Engineer	<b>Schematic and Circuit Board Design System</b> \$8.5k	•	—	—	• PC XT, AT; coprocessor with 2901 bit-slice engine • 4-MB main memory (in coprocessor) • Mouse, keyboard input; 14"/19" monitor with 768 x 592 x 8 resolution	• Modula schematic capture included • —	• On-line electrical rule checker • Design rule checker • —	• Canon CX, Gerber, HI, HP-LJ, HP-PL, VersaCAD • Excellon • —
<b>Optima Technology Inc.</b> 900 Middlesex Tpk. Bldg. 5 Billerica, MA 01821 (617) 667-7877  Robert Cowna Vice President, Sales	<b>OPTIMATE</b> \$35k	—	•	•	• All Apollo and DEC platforms; local-area networks • 4/16-MB main memory and 72-MB hard disk • Mouse or digitizer input; 19" monitor with 1000 x 1280 x 8 planes	• OPTIMATE schematic capture • Logic simulation and circuit simulation interfaces	• On-line electrical rule checker • Layout correct by construction; transmission-line analysis; circuit extractors • —	• Calcomp, HP, Versatec • Universal, EIA assembly interfaces • —
<b>Personal CAD Systems Inc.</b> 981 University Ave. Los Gatos, CA 95030 (408) 354-7193  Kirk G. Shorte PCB Product Line Manager	<b>PCB-3</b> \$13k	—	—	•	• PC AT (MS-DOS 2.0+); 3Com Etherlink • 640-KB main memory and 20-MB secondary storage • Mouse, digitizer, keyboard inputs; 13"/19" monitor with CGA, EGA	• PC-CAPS (P-CAD) • LOGS II (P-CAD)	• PC-ERC (batch) • PC-NLC	• Bruning, Calcomp, C.Itoh, Epson, HI, HP, IBM-GTCO, Interleaf, Muto, Okidata, Seiko, TI • Remex (paper tape punch) • —
<b>Pro.Lib, Inc.</b> 624 E. Evelyn Ave. Sunnyvale, CA 94086 (408) 732-1832  Sergio Szeinberg Vice Presidet of Sales and Marketing	<b>AutoPCB</b> \$2.5k-\$7.5k	—	—	•	• PC AT; PC-based network systems; Sun (UNIX) • 640-KB main memory and 10-MB hard disk • Mouse or digitizer input; 15"/19" monitor with EGA, VGA	• AutoSCEMA (Pro.Lib); generic interface • CADAT; SPICE; thermal analysis	• Electrical rule checker • Design rule checker • Circuit extractor	• AutoCAD-supported printers and plotters • Excellon • —
<b>Racal-Redac Inc.</b> 4 Lyberty Way Westford, MA 01886 (617) 692-4900  Susan Cook Marketing Communications Specialist	<b>VISULA</b> \$120k	•	•	•	• Apollo; Sun; MIPS coprocessor; Domain, Ether Net, DECnet, NFS • 8-MB main memory • Mouse, keyboard, IGES interface; 15"/19" monitor with resolution according to platform	• VISULA schematics • CADAT; SPICE • Pacific Numerix finite-element analysis	• On-line and batch electrical rule checker • — • —	• Calcomp, Ferranti, Gerber, HI, HP, Versatec • Sieb Mier controller; Fuji; others • GenRad, Marconi, Zehntel
<b>Royal Digital Systems Inc.</b> 3600 W. Bayshore Rd. Palo Alto, CA 94303 (415) 858-0811  Jerry Harvel Executive Vice President	<b>AutoMate</b> \$25k-\$40k	—	•	•	• PC AT, 80386-based PC; Prime; Sun; Data General minicomputers; MicroVAX II; Cyber supercomputers • Depends on platform • Depends on platform	• AutoMate schematic capture • P-SILOS; CADAT; P-SPICE; timing verifier	• On-line and batch electrical rule checker • Design rule checker • Circuit extractor	• Calcomp; Gerber; ASCII • Excellon; Universal; ASCII • Fairchild; Zehntel; Everett Charles; ASCII interface
<b>Scientific Calculations Inc.</b> 7796 Victor-Mendon Rd. Fishers, NY 14453 (716) 924-9303  Douglas W. Spice Manager, Marketing Services	<b>SCICARDS</b> \$25k	•	•	•	• VAX (VMS); SC Design Station (UNIX); ARX-20 coprocessor; Ethernet, DECnet • 6-MB main memory • Mouse, keyboard input; 19" monitor with up to 1024 x 1024 resolution	• SCIDesign schematic capture • SCISIM simulation	• On-line electrical rule checker • On-line design rule checker • —	• Benson, Calcomp, HP, Versatec • Excellon; Universal • GenRad, HP, Tektronix interfaces
<b>Shared Resources Inc.</b> 3047 Orchard Park San Jose, CA 95134 (408) 434-0444  Robert Wong Executive Vice President	<b>Koloa PCB Design System</b> \$50k-\$100k	—	—	•	• Elxsi; Multiflow; Prime; IBM 43xx; Apollo; Sun; VAX • 2-4-MB main memory and 70-MB hard disk • Mouse, keyboard input; 15"/19" monitor with 1280 x 1024 x 4 resolution	• Generic CAE interface • Direct link to AIDA simulator	• On-line and batch electrical rule checker • Correct-by-construction layout	• Calcomp; Gerber; Versatec • Excellon, Trudrill; Universal • Faultfinder; GenRad; HP; Teradyne; Zehntel; Trace, most bare-board testers; Wirewrap

<ul style="list-style-type: none"> <li>• Initial placement</li> <li>• Placement improvement</li> </ul>	Routing tools	<ul style="list-style-type: none"> <li>• Max board size</li> <li>• Layer count</li> <li>• Grid sizes</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic libraries</li> <li>• Library support</li> <li>• Package libraries</li> </ul>	Surface-mount design support
<ul style="list-style-type: none"> <li>• Force-directed; rat's-nest display</li> <li>• Placement cut and paste</li> </ul>	Calay autorouting supported	<ul style="list-style-type: none"> <li>• 65" × 65"</li> <li>• 30 layers</li> <li>• 1 mil to 65000 mils in 1-mil increments</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic libraries: customer-defined</li> <li>• New parts can be defined in metric units and mils; new library entry is formed for each package of each device</li> <li>• Package options: customer-defined</li> </ul>	Blind and buried vias
<ul style="list-style-type: none"> <li>• Optimized constructive; force-directed; matrix; rat's-nest display</li> <li>• Simulated annealing; automatic component, gate and pin swapping in any order</li> </ul>	Signal prerouting; priority, channel, maze reentrant multilayer router; pad hugging; ECL layout adherence; 45° routing on-line or postprocess; keep-out zones for wires and vias	<ul style="list-style-type: none"> <li>• 32" × 32"</li> <li>• 16 layers and 1000 components; 32 power layers; 40 other</li> <li>• 100, 50, 33.33, 25, 20, 16.67, 10, and 5 mils</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic symbols: 100</li> <li>• New physical parts can be defined in metric units and mils</li> <li>• Package options: 400</li> </ul>	User-defined land patterns; blind and buried vias; mirrored components for both board sides
<ul style="list-style-type: none"> <li>• Constructive (user-defined lattices); Kernighan-Mathaea min-cut; rat's-nest display</li> <li>• Simulated annealing; component and gate swapping</li> </ul>	Signal prerouting; priority, maze reentrant layer-pair router; 45° routing; keep-out zones	<ul style="list-style-type: none"> <li>• 64" × 64"</li> <li>• 100 layers (including graphics and signal layers)</li> <li>• User-definable grid</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic symbols: TTL, 905; CMOS, 427; linear, 400; discrete, 65; electromechanical, 89; Intel microprocessors, 160; Motorola microprocessors, 140</li> <li>• New physical parts can be defined in metric units and mils; physical parts are separate entries</li> <li>• Package options: DIPs, 8, 14, 16, 18, 20, 24, 28, 40, 48 pins; SOICs, 14, 16, 20, 24, 28; discrete SMDs, 19 packages; SIPs, 6, 8, 10</li> </ul>	Predefined footprints; blind and buried vias
<ul style="list-style-type: none"> <li>• Interactive; rat's-nest display; prompting under three strategies</li> <li>• —</li> </ul>	Manual and automatic signal prerouting; memory, but router; hybrid channel/maze router; daisy-chain routing for ECL; compaction; reentrant; 45° routing; keep-out zones for wires and vias	<ul style="list-style-type: none"> <li>• No specified limit</li> <li>• No specified limit</li> <li>• Gridless system</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic symbols: TTL, 3700; CMOS, 300; ECL, 350; analog, 400; PALs, 25; microprocessors, 35; connectors, 130</li> <li>• New physical parts can be defined in metric units and mils; some physical parts may be defined in the library</li> <li>• Package options: DIPs, 13; TOs, 10, DOs, 10; SMDs, 10; connectors, 130; cases, 5</li> </ul>	SMD footprints; blind and buried vias
<ul style="list-style-type: none"> <li>• Constructive; force-directed; min-cut; rat's-nest display; placement aids for both sides of board</li> <li>• Simulated annealing; automatic gate and pin swapping</li> </ul>	Signal prerouting; priority reentrant multilayer router; rip up; 45° routing; keep-out zones	<ul style="list-style-type: none"> <li>• 5M × 5M grid points up to 32767 pins</li> <li>• 50 layers, 200 documentation</li> <li>• 0.01 mil</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic symbols: over 3000</li> <li>• New physical parts can be defined and old parts edited in metric units and mils; library database requires only one physical model for each part</li> <li>• Package options: not specified</li> </ul>	Blind and buried vias; flip-flip screen; double-sided boards; SMT CAM file interfaces
<ul style="list-style-type: none"> <li>• Constructive; user-directed; rat's-nest display; expert system derives placement from knowledge base of 100 designs; automatic placement on both sides of board</li> <li>• Component, gate, and pin swapping</li> </ul>	Signal prerouting; channel router; finishing router with user-defined window; strategic compaction; ECL layout by net; reentrant; 45° routing; keep-out zones	<ul style="list-style-type: none"> <li>• 32" × 32"</li> <li>• 20 signal layers; 256 documentation layers</li> <li>• 1 mil and up in 1-mil increments</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic symbols: TTL, 600; LSTTL, 300; CMOS, 700; ECL, 100; analog, 250; LSI, 600; nonpackageable symbol type</li> <li>• New physical parts can be defined in metric units and mils; new library entries are formed for each package of each device</li> <li>• Package options: DIPs; SIPs; SMDs; pin-grid arrays; axial; chip-on-board; hybrids</li> </ul>	Surface-mount packages and land-pattern libraries; blind and buried vias; open file format for SMT CAM file interfaces; automatic via depth control; hybrid design support
<ul style="list-style-type: none"> <li>• Force-directed; rat's-nest display</li> <li>• Simulated annealing; component, gate, and pin swapping</li> </ul>	Signal prerouting (all sizes); priority, channel, maze reentrant router; ECL layout rule adherence; 45° routing; keep-out zones	<ul style="list-style-type: none"> <li>• 60" × 60"</li> <li>• Not specified</li> <li>• Any from 1 mil up</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic library: user-defined</li> <li>• New physical parts can be defined in metric units and mils</li> <li>• Package options: user-defined</li> </ul>	User-defined library for packaging and surface-mount land patterns; blind and buried vias; double-sided boards; SMT CAM file interfaces
<ul style="list-style-type: none"> <li>• Manual preplacement; rat's-nest display</li> <li>• Placement aids</li> </ul>	Priority routing with tuning; channel-like router; automatic trace centering; complete transmission line management; reentrant; 45° routing; keep-out zones for wires and vias	<ul style="list-style-type: none"> <li>• 100" × 100"</li> <li>• Essentially unlimited</li> <li>• Any grid from 1 μm to 1"; up to 10 wires between pins</li> </ul>	<ul style="list-style-type: none"> <li>• —</li> <li>• New physical parts can be defined in metric units and mils; all attributes come from package library</li> <li>• Package options: all TTL, ECL, and standard VLSI standard packaging; surface-mount packaging and surface-mount land patterns for most standard components</li> </ul>	Flip-flip screen; double-sided boards; hidden and buried via support

Directory of Printed Circuit Board Layout Systems (Continued)

Vendor Contact	System name, typical cost	Mainframe	Workstation	Software only	Hardware environment • System support • Memory support • Graphics support	Design entry • Simulation	ERC • DRC • Extraction	Output formats • Printer/plotter • Assembler • Tester; other
<b>Tektronix CAE Systems</b> PO Box 4600 Beaverton, OR 97076 (503) 629-3056  Jerry Tallinger Product Marketing Manager PCB Physical Layout Systems	<b>PCB WorkSystem</b> \$40k	•	•	•	• DEC (VMS), DECnet; Apollo (UNIX 4.2 BSD), Domain  • 9-MB main memory and 159-MB hard disk (DEC); 4-MB main memory and 155-MB hard disk (Apollo)  • Mouse input; 15" or 19" monitor with 1024 × 864 × 8 resolution	• Designer's Database Schematic Capture (Tektronix)  • HILO 3; HSPICE	• On-line MERLYN-P electrical rule checker; layout vs. schematic consistency checker  • MERLYN-P automated physical layout system; keep-out checker  • —	• HP interface  • Excellon; Gerber film tool  • MultiWire, generic insertion, and database extraction file
<b>Valid Logic Systems Inc.</b> 2 Omni Way Chelmsford, MA 01824 (617) 256-2300  Katherine Gambino Product Marketing Manager	<b>Board Design System (ALLEGRO)</b> From \$20k	—	•	•	• Sun 3/60, 3/110, 3/260, 4/200; Ethernet TCP/IP, NFS  • 8-MB main memory and 141-MB–1-GB hard disk  • Mouse, keyboard input; 19" monitor with 1152 × 900 × 10 resolution	• ValidGED graphics editor; ValidFLAT automatic schematic generation from netlist input  • ValidSIM logic simulator; Analog Design System circuit simulator; Realchip hardware modeler; Realfast simulation accelerator; Realmodel hardware modeler and accelerator	• On-line electrical rule checker  • On-line design rule checker  • Valid PACKAGER circuit extractor	• Calcomp; HP; Versatec; Gerber Photoplot  • Excellon; universal pick-and-place, generic interfaces  • GenRad, Facteron interfaces
<b>Vamp Inc.,</b> 6753 Selma Ave. Los Angeles, CA 90028 (213) 466-5533  J. Soluk Marketing Director	<b>McCAD EDS-1</b> \$1.5k	—	—	•	• Macintosh, Appletalk  • 1–2-MB main memory and 20-MB hard disk  • Mouse, digitizer input; 19" monitor with 1024 × 780 × 8 resolution	• McCAD schematics  • McCAD SIM; circuit simulation under development	• —  • —  • —	• Apple; HI; HP  • Excellon; Allied Linotronic interface  • —
<b>Vectron Graphic Systems Inc.</b> PO Box 271566 Concord, CA 95054 (408) 253-9555  Lee Woods Marketing Manager	<b>MAX PC Designer</b> \$1.5k	—	—	•	• IBM PC AT  • 640-KB main memory and 30-MB hard disk  • Mouse input; EGA monitor	• Vectron schematic capture  • —	• Batch electrical rule checker  • Design rule checker  • Circuit extractor	• Gerber; HP  • Excellon  • —
<b>Visionics Corp.</b> 343 Gibraltar Dr. Sunnyvale, CA 94089 (408) 745-1551  Alex Wellins Public Relations Manager	<b>EE Designer II</b> \$3k	—	—	•	• PC XT, AT, PS/2  • 640-KB main memory  • Mouse or digitizer input; CGA or EGA monitor	• Visionics schematic capture; interfaces to OrCAD and Omation  • Visionics logic and circuit simulation; interfaces to OrCAD and Omation	• Electrical rule checker  • Design rule checker  • Circuit extractor	• DM-PL; Epson; Gerber; HP-GL  • Excellon  • —
<b>Wintek Corp.</b> 1801 South St. Lafayette, IN 47904 (317) 742-8428  Marcia Borton CAD Sales	<b>smARTWORK</b> \$2k	—	—	•	• PC XT, AT (DOS 2.0+)  • 512-KB main memory  • Mouse input; CGA, EGA, VGA monitor	• Wintek schematic capture  • —	• Layout vs. schematic consistency checker  • On-line design rule checker  • Circuit extractors	• Epson; Gerber; IBM; HI; HP  • Excellon; DAC; IPC-NC/349  • —
<b>Xerox Corp.</b> 2441 Mission College Blvd. Santa Clara, CA 95054 (408) 562-2181  Scott Greene PCB Product Manager	<b>Expert</b> \$30k	—	•	—	• 2901-based workstation; Ethernet  • 4–12-MB main memory and 40–80-MB hard disk  • Mouse input; 19" monitor	• Expert (Xerox); FutureNet, SCI interfaces  • Expert logic simulator; CADAT, HILO, LASAR, SPICE interfaces	• Batch and interactive electrical rule checker  • Correct-by-construction layout; keep-out area checker  • Back annotation to FutureNet/SCI	• Calcomp, HI, HP, Versatec, Xerox  • Excellon; tape punch; Gardner Denver Wirewrap interface  • —



<ul style="list-style-type: none"> <li>• Initial placement</li> <li>• Placement improvement</li> </ul>	<b>Routing tools</b>	<ul style="list-style-type: none"> <li>• Max board size</li> <li>• Layer count</li> <li>• Grid sizes</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic libraries</li> <li>• Library support</li> <li>• Package libraries</li> </ul>	<b>Surface-mount design support</b>
<ul style="list-style-type: none"> <li>• Constructive initial placement; rat's-nest display</li> <li>• Automatic and interactive component, gate, and pin swapping</li> </ul>	Signal prerouting; priority routing by net name; reentrant maze router multilayer; 45° post-processing; keep-out zones for placement, routing, and vias separately or all-inclusive	<ul style="list-style-type: none"> <li>• 32" × 32"</li> <li>• 16 signal layers and 16 power/ground layers</li> <li>• 1-mil incremental grid</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic symbols: 74TTL, 2658; CMOS, 253; discrete, 1535; ROM, 96; PLDs, 86; microprocessors, 95</li> <li>• New physical parts can be defined in Mils only; schematic attributes identify package</li> <li>• Package types: DIPs, SIPs, SOICs, PLCCs</li> </ul>	Multiple colors to show layers through boards; double-sided boards in process; blind and buried vias; optional via fanouts
<ul style="list-style-type: none"> <li>• Board floorplanning feature allows use of different spacing and special placement algorithms in different board areas; exceptionally fast; 150 ICs placed in three minutes; constructive dynamic on-line rat's-nest display</li> <li>• Pin and gate swapping</li> </ul>	Signal prerouting; priority routing driven by schematic; expert router with combined costed-maze/rip-up router; push and shove of lines to make room for vias; full ECL design support; reentrant; on-line 45° routing with user-defined 45° length; keep-out zones for vias	<ul style="list-style-type: none"> <li>• 6.8 square miles at 1-mil resolution</li> <li>• 56 routing layers and unlimited data layers</li> <li>• Database resolution down to 0.00001 mil</li> </ul>	<ul style="list-style-type: none"> <li>• 4000+ TTL, ECL; 30+ PLDs; 60+ memory; 100+ LSI/VLSI logic parts; 97+ ASIC design kits available from ASIC vendors; behavioral models from Logic Automation and Quadtree; analog libraries also available</li> <li>• New physical parts can be defined in metric units and mils; a single device definition can be used for multiple package descriptions</li> <li>• Symbol library contains over 100 package options: DIP, 8-64 pins; SIPs, 6-12; SOICs, 8-24; PLCCs, 20-68; pin-grid arrays, 68-172, 1/4-1-W resistors; capacitors; many connectors</li> </ul>	Double-sided boards; automatic blind and buried via selection by router; automatic pin escape generation
<ul style="list-style-type: none"> <li>• Manual only</li> <li>• —</li> </ul>	Signal prerouting; priority routing; compaction; rip-up under development; reentrant; 45° routing; keep-out zones	<ul style="list-style-type: none"> <li>• 30" × 30" (manual); 16" × 16" (automatic)</li> <li>• 9 layers</li> <li>• 10, 20, 25, 50, 100, 125, 150, 156, and 200 mils</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic symbols: basic set, optional additional libraries</li> <li>• New physical parts can be defined in metric units and mils; physical parts are attributes of the schematic</li> <li>• Package options: 3000</li> </ul>	—
<ul style="list-style-type: none"> <li>• Manual only; rubberbanding; rat's-nest display</li> <li>• —</li> </ul>	Signal prerouting; priority, channel, maze reentrant layer-pair router; hugging; 45° routing; keep-out zones for wires and vias	<ul style="list-style-type: none"> <li>• 32" × 32", 1000 components</li> <li>• 256 layers</li> <li>• User-defined in 1-mil increments</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic symbols: 700</li> <li>• New physical parts can be defined in mils only; library entry references symbols may be preexisting</li> <li>• Package options: standard footprints</li> </ul>	Surface-mount packaging; surface-mount land pattern component library; double-sided boards; blind and buried via support
<ul style="list-style-type: none"> <li>• Autoplacement based on board size, density, and routing requirements; rat's-nest display (partial or complete)</li> <li>• Manual component, gate, and pin swapping only</li> </ul>	Signal prerouting; priority, channel, maze multilayer router; keep-out zones	<ul style="list-style-type: none"> <li>• 24" × 24"</li> <li>• 36 layers</li> <li>• 5, 12.5 mils</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic symbols: TTL, CMOS, ROM, RAM, EPROM, PROM, analog, 200; additional 200-part library available</li> <li>• New physical parts can be defined in metric units and mils</li> <li>• Package types: DIPs, 2; SIPs, 1; discrete, 3; SMD, 1</li> </ul>	Surface-mount package options
<ul style="list-style-type: none"> <li>• Manual placement only</li> <li>• —</li> </ul>	Signal prerouting; maze layer-pair router; reentrant; 45° routing; keep-out zones	<ul style="list-style-type: none"> <li>• 10" × 16"</li> <li>• 2 signal layers, 2 soldermask layers, 1 silkscreen layer</li> <li>• 50-mil grid</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic symbols: TTL, 294; CMOS, 148; ECL, 55; microprocessors, 144; miscellaneous, 74; ladders, 86; borders, 19</li> <li>• New physical parts can be defined in mils only</li> <li>• Package options: footprints created individually</li> </ul>	—
<ul style="list-style-type: none"> <li>• Constructive (minimum wire length); interactive rat's-nest display</li> <li>• Interactive component and pin swapping</li> </ul>	Signal prerouting; channel reentrant layer-pair router; compaction with wires moved on the fly; 45° routing; keep-out zones for vias, etch, and components independently for sides and internal board layers	<ul style="list-style-type: none"> <li>• 32" × 32"</li> <li>• 16 layers</li> <li>• User-defined grids</li> </ul>	<ul style="list-style-type: none"> <li>• Schematic symbols: TTL, 442; Intel, 71; Zilog, 42; Motorola, 107; CMOS, 102; LSI, 330</li> <li>• New physical parts can be defined in metric units and mils</li> <li>• Package options: DIP, 1; SIP, 1; passive, 1; through-hole, 1</li> </ul>	Blind vias; double-sided placement

# 3

## The Full Custom WorkSystem™ Lets You Handcraft High Performance.

IN A SERIES

**Tektronix  
Aided  
Engineering**

With Tektronix, you've got custom design power well in hand, thanks to the Full Custom WorkSystem.

Developed by Tektronix as part of Tektronix Aided Engineering, the Full Custom WorkSystem combines custom design capture, simulation and layout tools into one powerful solution.

You get the Designer's Database Schematic Capture (DDSC™), logic and circuit simulation, LEIA™ custom IC and hybrid layout editor and DRACULA™ IC mask verification and fracturing tool. All in the same performance-driven design environment.

DDSC provides you with a fast, user-customizable, menu-driven system for design capture of custom IC schematics.

When it's time for layout, you can do so with more design freedom using LEIA, our powerful interactive graphical layout editor. LEIA is especially suited for custom analog bipolar as well as gallium arsenide microwave applications.

Its flexible user interface provides you with multiple windows, macros, pop-up menus and the ability to customize real-time, user-configurable menus. LEIA also provides unparalleled support for hierarchical design and all-angle geometries.

What's more, LEIA's direct read and write of Calma GDSII™ structures speeds the integration of your design into verification and production software, including existing CAD systems.

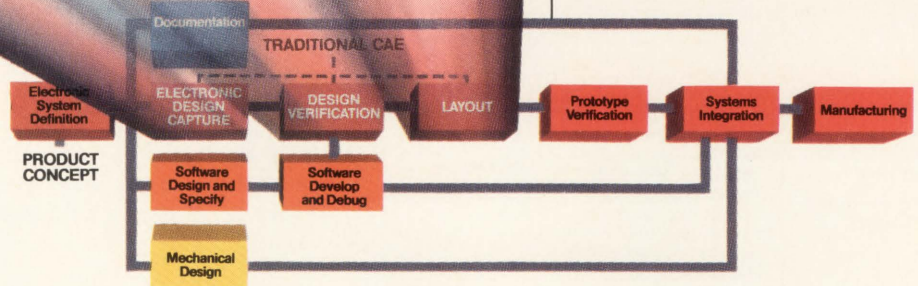
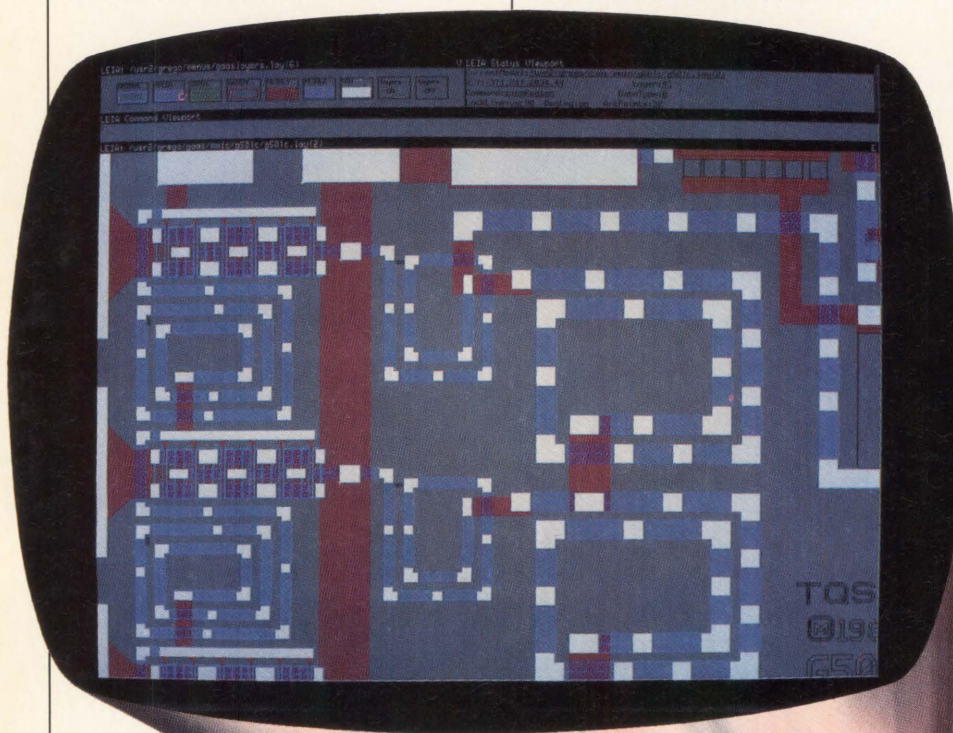
Before you fabricate your custom circuit, you can use DRACULA integrated software to verify your IC mask layout.

DRACULA let you measure and display design rule checker errors, extract parasitic electrical parameters and compare your layout to your schematic. So you can ensure total design integrity before you commit to fabrication. DRACULA's advanced fracture algorithms lower mask-making costs by greatly reducing flash count.

The Full Custom WorkSystem is part of Tektronix Aided Engineering. A family of integrated WorkSystems that takes you beyond traditional CAE solutions. And into prototype verification, software development and testing, systems integration, mechanical design and manufacturing. All running on industry-standard hardware platforms.

Best of all, it's from Tektronix. The name you've always trusted to get the engineering job done. So you're assured of worldwide service, support and training.

If you'd like to try your hand at high performance with the Full Custom WorkSystem, contact your local Tektronix, CAE Systems Division, sales office. Or call 800/547-1512. Tektronix, CAE Systems Division, P.O. Box 4600, Beaverton, OR 97076.



WorkSystem, DDSC, and LEIA are trademarks of Tektronix, Inc.  
DRACULA is a trademark of ECAD, Inc.  
Calma GDSII is a trademark of Calma, Inc.

CIRCLE NUMBER 19

**Tektronix**  
COMMITTED TO EXCELLENCE



# GUIDE TO THE LITERATURE

## VLSI DESIGN

- Glasser, L.A., and D.W. Dobberpuhl. 1985. *The Design and Analysis of VLSI Circuits*, Addison-Wesley, Reading, MA.
- Hodges, D.A., and H.G. Jackson. 1983. *Analysis and Design of Digital Integrated Circuits*, McGraw-Hill, New York, NY.
- Mead, C., and L. Conway. 1980. *Introduction to VLSI Systems*, Addison Wesley, Reading, MA.
- Trimberger, S. June 1982. "Automating Chip Layout," *IEEE Spectrum*.
- Weste, N.H.E., and K. Eshraghian. 1985. *Principles of CMOS VLSI Design*, Addison-Wesley, Reading, MA.

## Design Automation

- Blank, T. August 1984. "A Survey of Hardware Accelerators Used in Computer-Aided Design," *IEEE Design and Test*.
- Breuer, M., ed. 1972. *Design Automation of Digital Systems: Theory and Techniques (Vol. 1)*, Prentice-Hall, Englewood Cliffs, NJ.
- Breuer, M., and A. Friedman. 1976. *Diagnosis and Reliable Design of Digital Systems*, Computer Science Press, Rockville, MD.
- Elmasry, M.I., ed. 1985. *Digital VLSI Systems*, IEEE Press, New York, NY.
- Fujiwara, H. 1985. *Logic Testing and Design for Testability*, The MIT Press, Cambridge, MA.
- Hachtel, G.D., and A.L. Sangionvanni-Vincentelli. October 1981. "A Survey of Third-Generation Simulation Techniques," *Proceedings of the IEEE*, Vol. 69, No. 10.
- Hong, S.J., and R. Nair. January 1983. "Wire-Routing Machines—New Tools for VLSI Physical Design," *Proceedings of the IEEE*, Vol. 71.
- Ruehli, A.E., and G.S. Ditlow. January 1983. "Circuit Analysis, Logic Simulation, and Design Verification for VLSI," *Proceedings of the IEEE*.
- Shiva, S.G. December 1979. "Computer Hardware Description Languages—A Tutorial," *Proceedings of the IEEE*, Vol. 67.
- Williams, T., and K. Parker. January 1982. "Design for Testability—A Survey," *IEEE Transactions on Computers*, Vol. C31.

## STANDARD CELLS / CELL LIBRARIES

- Feller, A., A.M. Smith, R. Noto, P.W. Ramondetta, R.L. Pryor, and J.N. Greenhouse. October/November 1971. "Computer-Generated Low-Cost CMOS Custom Arrays," *RCA Engineer*.
- Kozawa, T., H. Horino, K. Watanabe, M. Nagata, and H. Hukuda. 1972. "Block and Track Method for Automated Layout Generation of MOS LSI Arrays," *International Solid State Circuits Conference*.
- Lopez, A.D., and H.-F. Law. August 1980. "A Dense Gate Matrix Layout Method for MOS LSI," *IEEE Journal of Solid State Circuits*, Vol. ED-27.

- Stebnisky, M.W., A. Feller, A.M. Smith, F. Borgini, S.S. Sharma, and M.J. McGinnis. 1983. "Approaches and Tradeoffs in Optimal Standard Cell Design," *Custom Integrated Circuits Conference*, Rochester, NY.
- Tobias, J. August 1981. "LSI/VLSI Building Blocks," *IEEE Computer*.
- Tosuntikool, N., and C.L. Saxe. 1983. "Automated Design of Standard Cells," *Custom Integrated Circuits Conference*.
- Weinberger, A. 1967. "Large-Scale Integration of MOS Complex Logic: A Layout Method," *IEEE Journal of Solid State Circuits*, Vol. SC-2.

## Gate and Logic Arrays

- Blumberg, R.J., and S. Brenner. October 1979. "A 1500 Gate, Random Logic, Large-Scale Integrated (LSI) Masterslice," *IEEE Journal of Solid State Circuits*, Vol. SC-14.
- Brackelmann, W., W. Wilhelm, J. Gaul, and H. Kaiser. 1979. "A Masterslice LSI for Subnanosecond Random Logic," *IEEE Journal of Solid State Circuits*, Vol. SC-14.
- D'Agostino, M.V., and A. Feller. 1968. "A Flexible Approach to Emitter-Coupled Logic Arrays," *International Solid State Circuits Conference*.
- Gray, J.P., I. Buchanan, and P. Robertson. 1982. "Designing Gate Arrays Using a Silicon Compiler," *19th Design Automation Conference*, Las Vegas, NV.
- Lipp, R. May 1983. "Advanced Architecture (Channel-less) Dual-Layer Metal CMOS Gate Arrays," *Custom Integrated Circuits Conference*.
- Ohkura, I., T. Noguchi, K. Sakashita, H. Ishida, T. Ichiyama, and T. Enomoto. 1982. "Gate Isolation—A Novel Basic Cell Configuration for CMOS Gate Arrays," *Custom Integrated Circuits Conference*, Rochester, NY.
- Patil, S.S., and T.A. Welch. September 1979. "A Programmable Logic Approach for VLSI," *IEEE Transactions on Computers*.

## Partitioning and Interconnect

- Carter, D.L., and D.F. Guise. November 1983. "Analysis of Signal Propagation Delays and Chip-Level Performance Due to Chip Interconnections," *International Conference on Computer Design*, Port Chester, NY.
- Donath, W.E. April 1979. "Placement and Average Interconnection Lengths of Computer Logic," *IEEE Transactions on Circuits and Systems*, Vol. CAS-26.
- Ferry, D.K. July 1985. "Interconnection Lengths and VLSI," *IEEE Circuits and Devices*.
- Landman, B.S., and R.L. Russo. 1971. "On a Pin Versus Block Relationship for Partitions of Logic Graphs," *IEEE Transactions on Computers*.

## Silicon Compilers

- Bergmann, N. 1983. "A Case Study of the FIRST Silicon Compiler," *Third Caltech Conference on VLSI*, Pasadena, CA.
- Buric, M.R., and T.G. Matheson. 1985. "Silicon Compilation Environments," *Custom Integrated Circuits Conference*.

- Edgington, D., B. Walker, S. Nance, C. Starr, S. Dholakia, and M. Kliment. 1984. "CMOS Cell-Layout Compilers for Custom IC Design," *Custom Integrated Circuits Conference*.
- Johannsen, D.L. 1979. "Bristle Blocks: A Silicon Compiler," *Caltech Conference on VLSI*, Pasadena, CA.
- Siskind, J.M., J.R. Southard, and K.W. Crouch. 1982. "Generating Custom High Performance VLSI Designs from Succinct Algorithmic Descriptions," *MIT Conference on Advanced Research in VLSI*, Cambridge, MA.

### Design Synthesis

- Brayton, R.K., G.D. Hachtel, R.T. McMullen, and A.L. Sangiovanni-Vincentelli. 1984. *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publishers, Norwell, MA.
- Darringer, J., W. Joyner, L. Berman, and L. Trevillyan. July 1981. "Logic Synthesis through Local Transformations," *IBM Journal of Research and Development*, Vol. 25, No. 4.
- McCluskey, E.J. November 1956. "Minimization of Boolean Functions," *Bell System Technical Journal*, Vol. 35.
- Thomas, D.E., C.Y. Hitchcock, T.J. Kowalski, J.V. Rajan, and R. Walker. December 1983. "Automatic Data Path Synthesis," *IEEE Computer*, Vol. 16.

### Integrated Circuit Technology

- Hoefflinger, B. 1982. "CMOS Technologies and Circuits," *Custom Integrated Circuits Conference*, Rochester, NY.
- Lohstroh, J. July 1981. "Devices and Circuits for Bipolar VLSI," *Proceedings of the IEEE*, Vol. 69, No. 7.
- Murphy, B.T., H.A. Waggener, and J.E. Iwersen. 1965. "Non-Saturating Monolithic Logic Circuits with Improved Stability," *International Solid State Circuits Conference*.
- Solomon, P.M. May 1982. "A Comparison of Semiconductor Devices for High-Speed Logic," *Proceedings of the IEEE*, Vol. 70, No. 5.
- Wanlass, F.M., and C.T. Sah. 1963. "Nanowatt Logic Using Field-Effect Metal-Oxide Semiconductor Triodes," *International Solid State Circuits Conference*.

## SYSTEM SIMULATION

- Barbacci, M.R. and A.W. Nagle. March 1978. *An ISPS Simulator*, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA.
- Hill, D., and W. van Cleemput. 1979. "SABLE: A Tool for Generating Structured, Multi-Level Simulations," *16th Design Automation Conference*, San Diego, CA.
- Knuth, D.E., and J.K. McNeley. August 1964. "SOL—A Symbolic Language for General-Purpose Systems Simulation," *IEEE Transactions on Electronic Computers*.
- Rose, C.W., L.A. Rogers, and R.V. Straubs. 1979. "The N.mPc System Description Facility," *16th Design Automation Conference*, San Diego, CA.
- Straubs, R.V. August 1978. "A Compiler for a Register Transfer Based Simulation Language," *Report CES-79-8 (Master's thesis)*, Department of Computer Engineering and Science, Case Western Reserve University, Cleveland, OH.

### Logic Simulation — Basic Techniques

- Bening, L.C. 1969. "Simulation of High Speed Computer Logic," *Design Automation Workshop*, Miami Beach, FL.
- Case, P.W., H.H. Graff, L.E. Griffith, A.R. Leclercq, W.B. Murley, and T.M. Spence. April 1964. "Solid Logic Design Automation," *IBM Journal of Research and Development*.
- Eichelberger, E.B. March 1965. "Hazard Detection in Combinational and Sequential Digital Circuits," *IBM Journal of Research and Development*.

- Jephson, J.S., R.P. McQuarrie, and R.E. Vogelsberg. 1969. "A Three Value Design Verification System," *IBM Systems Journal*, Vol. 8, No. 3.
- Ulrich, E. 1965. "Time-Sequenced Logical Simulation Based on Circuit Delay and Selective Tracing of Active Network Paths," *ACM National Conference*.

### Logic Simulation Acceleration

- Ausdale, A.W. 1971. "Use of the Boeing Computer Simulator for Logic Design Confirmation and Failure Diagnostic Programs," *International Aerospace Conference*.
- Howard, J., R. Malm, and L. Warren. 1983. "Introduction to the IBM Los Gatos Simulation Machine," *International Conference on Computer Design*, Port Chester, NY.
- Pfister, G.F. 1982. "The Yorktown Simulation Engine: Introduction," *19th Design Automation Conference*, Las Vegas, NV.
- Sasaki, T., N. Koike, K. Ohmore, and K. Tomita. 1983. "HAL: A Block-Level Hardware Logic Simulator," *20th Design Automation Conference*, Miami Beach, FL.

### Fault Simulation

- Armstrong, D.B. 1972. "A Deductive Method for Simulating Faults in Logic Circuits," *IEEE Transactions on Computers*, C-21(5).
- Seshu, S. 1965. "On An Improved Diagnosis Program," *IEEE Transactions on Electronic Computers*, EC-12(2).
- Ulrich, E., T. Baker, and L. Williams. 1972. "Fault-Test Analysis Techniques Based on Logic Simulation," *9th Design Automation Workshop*, Dallas, TX.
- Ulrich, E., and T. Baker. 1973. "The Concurrent Simulation of Nearly Identical Digital Networks," *10th Design Automation Workshop*, Portland, OR.

### Timing Simulation

- Arnout, G., and H. De Man. June 1978. "The Use of Threshold Functions and Boolean-Controlled Network Elements for Macro-modeling of LSI Circuits," *IEEE Journal of Solid State Circuits*, Vol. SC-13.
- Bryant, R.E. Fourth Quarter 1980. "An Algorithm for MOS Logic Simulation," *Lambda*.
- Chawla, B.R., H.K. Gummel, and P. Kozak. December 1975. "MOTIS—An MOS Timing Simulator," *IEEE Transactions on Circuits and Systems*.
- Newton, A.R. September 1979. "The Simulation of Large Scale Integrated Circuits," *IEEE Transactions on Circuits and Systems*, Vol. CAS-26.
- Terman, C.J. 1983. "RSIM—A Logic-Level Timing Simulator," *International Conference on Computer Design*.

### Timing Analysis

- Jouppi, N. 1983. "TV: An nMOS Timing Analyzer," *3rd Caltech Conference on VLSI*, Pasadena, CA.
- McWilliams, T.M. 1980. "Verification of Timing Constraints on Large Digital Systems," *17th Design Automation Conference*, Minneapolis, MN.
- Ousterhout, J. 1983. "Crystal: A Timing Analyzer for nMOS VLSI Circuits," *3rd Caltech Conference on VLSI*, Pasadena, CA.
- Penfield, P., Jr., and J. Rubinstein. 1981. "Signal Delay in RC Tree Networks," *18th Design Automation Conference*, Nashville, TN.
- Pilling, D.J., and H.B. Sun. 1973. "Computer Aided Prediction of Delays in LSI Logic Systems," *10th Design Automation Workshop*, Portland, OR.

### Circuit Simulation — Direct Methods

- Nagel, L.W., and D.O. Pederson. 1973. "Simulation Program with Integrated Circuit Emphasis," *16th Midwest Symposium on Circuit Theory*, Waterloo, Ontario.

- Nagel, L. May 1975. "SPICE2: A Computer Program to Simulate Semiconductor Circuits," *ERL Memo ERL-M520*, University of California at Berkeley.
- Weeks, W.T., A.J. Jimenez, G.W. Mahoney, D. Mehta, J. Qasemzadeh, and T.R. Scott. November 1973. "Algorithms for ASTAP—A Network Analysis Program," *IEEE Transactions on Circuit Theory*.

### Circuit Simulation — Relaxation Techniques

- Lelarsmee, E., A.E. Ruchli, and A.L. Sangiovanni-Vincentelli. July 1982. "The Waveform Relaxation Method for Time Domain Analysis of Large Scale Integrated Circuits," *IEEE Transactions on Computer-Aided Design*, Vol. CAD-1, No. 3.
- Rabbat, N.B.G., A.L. Sangiovanni-Vincentelli, and H.Y. Hsieh. September 1979. "A Multilevel Newton Algorithm with Macro-Modeling and Latency for the Analysis of Large-Scale Nonlinear Circuits in the Time Domain," *IEEE Transactions on Circuits and Systems*, Vol. CAD-26.
- Saleh, R.A., J.E. Kleckner, J.E., and A.R. Newton. 1983. "Iterated Timing Analysis in SPLICE1," *International Conference on Computer-Aided Design*, Santa Clara, CA.

### Modeling for Circuit Simulation

- Gummel, H.K., and H.C. Poon. 1970. "A Compact Bipolar Transistor Model," *International Solid State Circuits Conference*.
- Shichman, H., and D.A. Hodges. 1968. "Modeling and Simulation of Insulated-Gate Field-Effect Transistor Switching Circuits," *IEEE Journal of Solid State Circuits*, Vol. SC-3.

## LAYOUT SYSTEMS

- Case, P.W., M. Correia, W. Gianopoulos, W.R. Heller, H. Ofek, P.C. Raymond, R.L. Simek, C.B. Stieglitz. 1981. "Design Automation in IBM," *IBM Journal of Research and Development*, Vol. 25, No. 5.
- Chen, K.A., M. Feuer, K.H. Khokhani, N. Nan, and S. Schmidt. 1977. "The Chip Layout Problem: An Automatic Wiring Procedure," *14th Design Automation Conference*, New Orleans, LA.
- Persky, G., D.N. Deutsch, and D.G. Schweikert. May 1977. "LTX—A Minicomputer-Based System for Automated LSI Layout," *Design Automation and Fault-Tolerant Computing*.
- Sechen, C., and A.L. Sangiovanni-Vincentelli. April 1985. "The TimberWolf Placement and Routing Package," *IEEE Journal of Solid State Circuits*, Vol. SC-20.

### Force-Directed Placement

- Charney, H.R., and D. Plato. 1968. "Efficient Partitioning of Components," *5th Design Automation Workshop*, Washington, DC.
- Lu, S., and R. Dutton. 1985. "An Analytical Algorithm for Placement of Arbitrarily Sized Blocks," *22nd Design Automation Conference*, Las Vegas, NV.
- Quinn, C., and M. Breuer. July 1969. "A Force Directed Component Placement Procedure for Printed Circuit Boards," *IEEE Transactions on Circuits and Systems*.

### Clustering

- Kurtzberg, J. 1965. "Algorithms for Backplane Formation," *Microelectronics in Large Systems*, Spartan Books.
- Schuler, D., and E. Ulrich. 1972. "Clustering and Linear Placement," *9th Design Automation Workshop*, Dallas, TX.

### Simulated Annealing

- Jepsen, D.W., and C.D. Gelatt, Jr. 1983. "Macro Placement by Monte Carlo Annealing," *International Conference on Com-*

*puter Design*, Port Chester, NY.

- Kirkpatrick, S., C.D. Gelatt, and M.P. Vecchi. March 1983. "Optimization by Simulated Annealing," *Science*, 220.
- White, S.R. 1984. "Concepts of Scale in Simulated Annealing," *International Conference on Computer Design: VLSI in Computers*, Port Chester, NY.

### Partitioning

- Breuer, M. 1977. "Min-Cut Placement," *Journal of Design Automation and Fault-Tolerant Computing*.
- Dunlop, A., and B. Kernighan. January 1985. "A Procedure for Layout of Standard Cell VLSI Circuits," *IEEE Transactions on Computer-Aided Design*.
- Kernighan, B., and S. Lin. February 1970. "An Efficient Heuristic Procedure for Partitioning Graphs," *Bell System Technical Journal*.
- Lauther, U. 1979. "A Min-Cut Placement Algorithm for General Cell Assemblies Based on a Graph Representation," *16th Design Automation Conference*, San Diego, CA.

### Placement Improvement/Evaluation

- Burstein, M., and M.N. Youssef. 1985. "Timing Influenced Layout Design," *22nd Design Automation Conference*, Las Vegas, NV.
- Dunlop, A.E., et al. 1984. "Chip Layout Optimization Using Critical Path Weighting," *21st Design Automation Conference*, Albuquerque, NM.
- Hanan, M., P. Wolff, and B. Agule. 1976. "Some Experimental Results on Placement Techniques," *13th Design Automation Conference*, San Francisco, CA.
- Hartoog, M.R. 1986. "Analysis of Placement Procedures for VLSI Standard Cell Layout," *23rd Design Automation Conference*, Las Vegas, NV.

### Global Routing

- Burstein, M., and R. Pelavin. 1983. "Hierarchical Wire Routing," *IEEE Transactions on Computer-Aided Design*, Vol. 2, No. 4.
- Hwang, F. 1978. "The Rectilinear Steiner Tree Problem," *Journal of Design Automation and Fault-Tolerant Computing*, Vol. 2.
- Lee, C.Y. September 1961. "An Algorithm for Path Connection and Its Applications," *IRE Transactions on Electronic Computers*.
- Soukup, J. 1978. "Fast Maze Router," *15th Design Automation Conference*, Las Vegas, NV.

### Channel Routing

- Deutsch, D. 1976. "A 'Dogleg' Channel Router," *13th Design Automation Conference*, San Francisco, CA.
- Hashimoto, A., and J. Stevens. 1971. "Wire Routing by Optimizing Channel Assignment within Large Apertures," *8th Design Automation Workshop*, Atlantic City, NJ.
- Rivest, R., and C. Fiduccia. 1982. "A 'Greedy' Channel Router," *19th Design Automation Conference*, Las Vegas, NV.
- Sangiovanni-Vincentelli, A., and M. Santomauro. 1983. "YACR—Yet Another Channel Router," *Custom Integrated Circuits Conference*, Rochester, NY.
- Yoshimura, T., and E.S. Kuh. 1982. "Efficient Algorithms for Channel Routing," *IEEE Transactions on Computer-Aided Design*, Vol. 1, No. 1.

### Data Structures

- Ousterhout, J. January 1984. "Corner Stitching: A Data Structuring Technique for VLSI Layouts," *IEEE Transactions on Computer-Aided Design*.
- Rosenberg, J. January 1985. "Geographical Data Structures Compared: A Study of Data Structures Supporting Region Queries," *IEEE Transactions on Computer-Aided Design*.

# VLSI SYSTEMS DESIGN

## SUBJECT INDEX, 1986-1987

### A

#### Analog modeling

"GaAs MESFET Model for Precision Analog IC Design," *Ravender Goyal and Norman Scheinberg*, May 4, 1987, p. 52.

#### Analog MOS design

"Analog Macrocell Assembler," *Greg Winner, Tuan A. Nguyen, and Carroll Slemaker*, May 4, 1987, p. 68.

"Analog Module Generators for Silicon Compilation," *Jay Kuhn*, May 4, 1987, p. 74.

"Device Design for Analog ICs," *Mohammad A. Rehman*, March 1986, p. 100.

"MOS Device Design for Analog ICs," *Mohammad A. Rehman*, June 1986, p. 133.

"Survey of Analog Semicustom ICs," *VLSI Systems Design Staff*, May 4, 1987, p. 89.

"A Switched-Capacitor Filter Compiler," *Therasse L. Reynders, R. Lannoo, and B. Dupont*, September 1987, p. 85.

See also Automatic IC layout, Semicustom ICs

#### Automatic IC layout

"An Automatic Layout System for Linear Arrays," *M.A.K. Moussa*, April 1987, p. 60.

"PIONEER: A Macro-Based Floor-Planning Design System," *Lin S. Woo, C.K. Wong, and D.T. Tang*, August 1986, p. 32.

"Place and Route Strategies," *Jiri Soukup*, November 1986, p. 34.

"Survey of Automatic Layout Software," April 1987, p. 78.

"Timing-Driven Layout of Cell-Based ICs," *Steven Teig, Randall L. Smith, and John Seaton*, May 1986, p. 63.

See also Floorplanning, IC layout systems

**Automatic test equipment.** See Testing

**ADAS.** See Behavioral simulation

### B

#### Behavioral simulation

"Behavioral Modeling in Logic Simulation," *David J. Wharton*, August 1986, p. 46.

"A Behavioral Simulation Shell," *Gary Beihl and Shekhar Borkar*, November 1986, p. 96.

"Board-Level Simulation using Models with X-State Handling," *William D. Billowitch*, February 1987, p. 56.

#### Bipolar technology

"A Bipolar Process for Radiation Hardness," *Jim Wells and Ed Jeffrey*, September 1986, p. 102.

"Creating Low-Power Bipolar ECL at VLSI Densities," *George Wilson*,

May 1986, p. 84.

"Handling the Power Dissipation of ECL Gate Arrays," *Biswa Banerjee*, January 1987, p. 40.

"High-Performance CML Systems," *Richard Carmichael, Kevin Schutz, David Still, and David G. Wick*, September 1986, p. 40.

"Semicustom ECL Array Becomes Custom Analog Pin Driver," *Jim Graydon and Nghiem Phan*, May 20, 1987, p. 80.

See also Automatic IC layout, High-speed logic, Programmable logic

#### Built-in self-test

"A CMOS Cell Library Design for Testability," *Dick L. Liu and Edward J. McCluskey*, May 4, 1987, p. 58.

"Feedback Shift Registers for Self-Testing Circuits," *Laung-Terng Wang and Edward J. McCluskey*, December 1986, p. 50.

"Signature Analysis Testing in Large Standard-Cell ASICs," *Eric L. Smitt*, May 20, 1987, p. 46.

See also Computer and system design

### C

#### CAD systems

"CAD for Surface Mount: Still a No-Via Zone," *Ernest L. Meyer*, February 1986, p. 74.

"CAD Software Users Address Database Problems," *John Andrews*, September 1986, p. 58.

"VITAL: A Cell-Based ASIC Assembler," *Stephen McNeary, Rathin Putatunda, Howard Rifin, Robert Robillard, and David Smith*, November 1986, p. 22.

See also Automatic IC layout, IC layout systems, Printed circuit board design and manufacture

#### CAE systems

"Automatically Flatten Hierarchical Schematics," *Stuart Swan*, September 1986, p. 82.

"The Desktop Workstations: Life, Liberty and the Pursuit of Personal Computers," *Stephen Evanczuk*, July 1986, p. 56.

"Survey of CAE Systems," *VLSI Systems Design Staff*, June 1986, p. 85.

"Survey of CAE Systems," *VLSI Systems Design Staff*, June 1987, p. 51.

See also CAD systems, Design system integration, IC layout systems

**Capacitor layout.** See Analog MOS design

#### Cell compilers

"On Module Generation," *Ernest L. Meyer*, March 1987, p. 48.

"Structured Design with Module Generators," *Dar-Sun Tsien*, March 1987, p. 40.

See also Analog MOS design

#### Cell-based design

"Guide to Core Processors," *VLSI Systems Design Staff*, December 1986, p. 32.

"PIONEER: A Macro-Based Floor-Planning Design System," *Lin S. Woo, C.K. Wong, and D.T. Tang*, August 1986, p. 32.

- "Semicustom Design with a Microcontroller Core," *Ralph Haines and Chris Phillips*, December 1986, p. 26.
- "Super Integration: Using Standard Products as Megacells," *Jerry G. Goetsch*, June 1987, p. 106.
- "Survey of Gate Arrays and Cell Libraries," *VLSI Systems Design Staff*, November 1986, p. 54.
- "Survey of Gate Arrays and Cell Libraries," *VLSI Systems Design Staff*, November 1987, p. 76.
- "Using Redefinable IC Technology to Develop a VGA Chip Set," *Dean Hays and Bill Knapp*, November 1987, p. 68.
- "VITAL: A Cell-Based ASIC Assembler," *Stephen McNeary, Rathin Putatunda, Howard Rifkin, Robert Robillard, and David Smith*, November 1986, p. 22.
- See also Built-in self-test, Cell compilers, Computer and system design, Core processors, High-speed logic

#### Circuit design

- "A Fast Systolic FIFO Queue," *Asim J. Al-Khalili and Zikad Ali*, May 1986, p. 76.
- "Layout Design of a Systolic Multiplier Unit for 2's Complement Numbers," *A.R. Hurson, B. Shirazi, and S. H. Pakzad*, February 1986, p. 78.
- "Space-Time Logic," *Charles R. Moeller*, August 1986, p. 92.
- "Wait-State Remover Improves System Performance," *Kapil Shankar*, November 1986, p. 102.

#### Circuit simulation

- "The ADEPT Timing Simulation Algorithm," *Peter Odryna and Sani Nassif*, March 1986, p. 24.
- "Choosing a Circuit Simulator," *Joe Domitrowich*, September 1986, p. 90.
- "Circuit Simulators at a Glance," *Roderic Beresford and Joe Domitrowich*, August 1987, p. 76.
- "Converting SPICE to Vector Code," *Bruce Greet*, January 1986, p. 30.
- "Interactive Control of Analog System Simulation," *David W. Smith, Scott A. Majdecki and Doug Johnson*, July 1987, p. 46.
- "Mixed-Domain Analysis for Circuit Simulation," *Kevin Walsh and Bruce Wolfe*, August 1987, p. 44.
- "On Parallel Circuit Simulation," *Pamela J. Waterman*, July 1987, p. 56.
- "Parallel Computing for VLSI Circuit Simulation," *Jeffrey T. Deutsch, Thomas D. Lovett, and Michael Lee Squires*, July 1986, p. 46.
- "Selecting a Personal Computer for Circuit Simulation," *J. Richard Hines*, March 1987, p. 66.
- "A Survey of Circuit Simulation Programs," *Greg Barros*, July 1986, p. 32.
- "Survey of Circuit Simulators," *Roderic Beresford and Joe Domitrowich*, July 1987, p. 70.

See also Analog modeling

CML. See Bipolar technology

#### CMOS technology

- "Application of Silicides to Standard Cells," *Werner Metz*, February 1986, p. 42.
- "CMOS in Radiation Environments," *Gordon P. Ansell and Joe S. Tirado*, September 1986, p. 28.

See also Silicon on insulator, Supercomputers

#### Computer and system design

- "Design of the Edgel Superminicomputer," *Joseph C. Circello*, May 1986, p. 22.
- "Inserting VLSI Technology: A Case Study," *Juris Ozols and Robert Humphrey*, September 1986, p. 52.
- "A Multiprocessor Design in Custom VLSI," *David Jurasek, William Richardson, and Doran Wilde*, June 1986, p. 26.
- "A Pin-Logic Gate Array for a Programmable Logic Programmer," *Kellie Crisafulli*, October 1986, p. 30.

See also Multiprocessing

#### Conferences

- "Conference Preview: ICCAD-87," *VLSI Systems Design Staff*, October 1987, p. 81.
- "Conference Preview: The 1986 Custom Integrated Circuits Conference," *Ernest L. Meyer*, April 1986, p. 20.
- "Conference Preview: The 1986 Design Automation Conference," *Roderic Beresford*, May 1986, p. 32.
- "The Design Automation Conference," May 4, 1987, p. 26.
- "ICCAD-86 Conference Preview," *VLSI Systems Design Staff*, October 1986, p. 25.
- "The International Conference on Computer Design," *VLSI Systems*

*Design Staff*, September 1987, p. 20.

- "International Conference on Computer Design: VLSI in Computers and Processors," *VLSI Systems Design Staff*, September 1986, p. 22.
- "The 1987 Custom Integrated Circuits Conference," April 1987, p. 36.

#### Core microprocessors

- "FORTH Processor Core for Integrated 16-Bit Systems," *Peter S. Danile and Chris W. Malinowski*, June 1987, p. 98.
- "Getting to the Core," *David Smith*, June 1987, p. 90.

See also Cell-based design

Current-mode logic. See Cell-based design

Custom ICs. See Gallium arsenide, Market forecasts

## D

Database management. See Design system integration

#### Design centers

- "Directory of ASIC Design Centers" *VLSI Systems Design Staff*, January 1987, p. 60.

See also Semicustom ICs

Design for testability. See Built-in self-test, Testability

#### Design system integration

- "An Engineering Design Management System," *Mark Mayotte*, January 1987, p. 30.
- "Integrating the Electronic Design Process," *Jean Brouwers and Moshe Gray*, June 1987, p. 38.

Distributors. See Semicustom ICs

## E

Emitter-coupled logic. See Bipolar technology, High-speed logic

Emulation. See Prototyping

#### Encryption

- "Using Programmable Logic Cell Arrays In a Satellite Earthstation," April 1987, p. 26.

Engineering workstations. See CAE systems

ECL. See Bipolar technology, High-speed logic

## F

#### Fault simulation

- "Eliminating Failures in the Field," *Geoffrey Mott and John Newkirk*, October 1986, p. 88.
- "Finding Fault: An Update on Fault Simulation," *David Smith*, October 1987, p. 28.
- "The Need for Fault Simulation," *Fred Buelow and Ed Porter*, October 1986, p. 84.
- "Performance of Probabilistic Fault Grading," *Robert D. Hess and William C. Berg, Jr.*, January 1986, p. 40.
- "Techniques for Logic and Fault Simulation," *Ernst Ulrich and Itsuo Suetsugu*, October 1986, p. 68.

See also Test generation

Fault tolerance. See Memory design

#### Floorplanning

- "An Automatic Floorplanner for up to 100,000 Gates," *H. Modarres and A. Kelapure*, December 1987, p. 38.
- "Graphical Floorplan Design of Cell-Based VLSI Circuits," *Edmond Macaluso*, April 1987, p. 50.

#### Foundries

- "Economics of Fast-Turn Wafer Production," *James A. Schoeffel and Michael L. Rieger*, February 1987, p. 22.



- "Survey of Semiconductor Foundries," *Judy Elster*, August 1986, p. 62.  
 "Survey of Semiconductor Foundries," *VLSI Systems Design Staff*, August 1987, p. 60.

## G

### Gallium arsenide

- "A Custom 300-Gate GaAs Circuit for Frequency Encoding, Designed on a Personal Computer," *Charles G. Ekroot*, November 1987, p. 19.  
 "Gallium Arsenide on Silicon," *John C.C. Fan*, December 1987, p. 80.  
 "A High-Performance Low-Power GaAs Gate Array Family," *Gary M. Lee, Charles M. Lee, Rick A. Easley, Hector F. Lai, and Al G. Schmidt*, July 1987, p. 24.  
 "A Knowledge-Based GaAs Design System," *George Janac, Carlos Garcia, and Richard Davis*, April 1987, p. 68.  
 "Survey of Gate Arrays and Cell Libraries," *VLSI Systems Design Staff*, November 1987, p. 76.

See also Analog modeling, Circuit simulation

### Gate arrays

- "Embedded RAM in Gate Arrays: Configurability and Testability," *P. Simon Bennett, Robert P. Dixon, and Kent C. Oertle*, November 1987, p. 60.  
 "A Fast 20K Gate Array with On-Chip Test System," *Ron Lake*, June 1986, p. 46.  
 "Gate Array Testability: A Customer Perspective," *Ernest L. Meyer*, June 1986, p. 34.  
 "An IBM-Compatible Mainframe in 20,000-Gate CMOS Arrays," *Mark Golkar and Jacques Losq*, May 20, 1987, p. 64.  
 "On-Chip Testability Circuit for CMOS Gate Arrays," *Kunnau Chen*, January 1986, p. 48.  
 "A PC/XT Integration Chip on a Single 10K Gate Array," *Kevin Lee, Charles Chi, and Raymond Chan*, June 1987, p. 28.  
 "The Role of CMOS Gate Arrays in High-Speed System Design," *Harold Dozier*, January 1986, p. 20.  
 "Survey of Gate Arrays and Cell Libraries," *VLSI Systems Design Staff*, November 1986, p. 54.  
 "Survey of Gate Arrays and Cell Libraries," *VLSI Systems Design Staff*, November 1987, p. 76.  
 "The Tortuous Route to High-Volume Production of a Hard-Disk Controller," *David Baughman*, January 1987, p. 22.

See also Automatic IC layout, Computer design, Gallium arsenide, High-speed logic, Market forecasts, Supercomputers

### Graphics processing

- "A Semicustom Video Controller for Personal Computer Graphics," *Nandu Marketkar*, August 1986, p. 24.

See also Cell-based design, Signal processing

### Graphics terminals. See IC layout systems

## H

### Hardware accelerators

- "Automation and Simulation in Large System Design," *Bruce Erickson*, December 1986, p. 42.  
 "Hardware Simulator Models for Logic Devices," *William Fazakerly*, November 1987, p. 30.  
 "Mixed-Level Simulation Acceleration," *Stephen Evanczuk*, February 1987, p. 62.  
 "A Second-Generation Routing Accelerator for PCB Designs," *Frank Weisenberger and David Rager*, December 1986, p. 20.

### High-speed logic

- "Champion ASIC Technologies," *Ernest L. Meyer*, July 1987, p. 18.  
 "A High-Performance Bipolar Cell Library," *John S. Shier and Jeffrey M. Wisted, and Zdenek E. Skokan*, July 1987, p. 32.  
 "Solving Problems in High-Speed ASIC Design," *Donald C. Kirkpatrick*, March 1987, p. 26.

See also Bipolar technology, Gallium arsenide

### Hybrid circuits

- "An Overview of Hybrid Circuit Vendors," *Greg Barros*, October 1986, p. 92.

## I

### Interconnection effects. See Parasitics

### IC layout systems

- "Architecture of a Computer-Integrated Engineering System," *Evan Auran and Tani Shavit*, June 1986, p. 58.  
 "Beyond GDS II: Combining Flexibility and Automation in Full-Custom IC Design," *Dave Hightower, Deanna McCusker, Kazuya Shinozuka, and Helge Szweringi*, October 1987, p. 38.  
 "Sticks Layout Notation for MESFETs and Refractory-Metal FETs," *Fayez El Guibaly and M.I. Elmasry*, February 1986, p. 88.  
 "Survey of IC Layout CAD Systems," *VLSI Systems Design Staff*, September 1986, p. 67.  
 "Survey of IC Layout Systems," *VLSI Systems Design Staff*, December 1987, p. 63.  
 "A Workstation for Cell-Based IC Layout," *Harold M. Rabbie, Robert W. Dahlberg, and Herbert L. Hinstorff, Jr.*, June 1986, p. 70.

See also Automatic IC layout, Floorplanning, Gallium arsenide, Symbolic layout

### ISDN

- "Integrating the ISDN Basic-Rate Functions," *Dale Gulick*, September 1987, p. 24.

## L

**Layout.** See Automatic IC layout, IC layout systems, Printed circuit board design and manufacture

**Layout analysis.** See Parasitics

**Libraries.** See Cell-based design, Logic simulation, Workstation libraries

### Logic simulation

- "Automatic Translation of Logic Models for Workstation Libraries," *Mehdi Amani, Gary Griffin, and Scott Hamm*, December 1987, p. 56.  
 "Behavioral Modeling in Logic Simulation," *David J. Wharton*, August 1986, p. 46.  
 "Logic Simulation on PCs," *Shahriar Emami and Steve Brunner*, January 1986, p. 36.  
 "1987 Survey of Logic Simulators," *VLSI Systems Design Staff*, February 1987, p. 70.  
 "1986 Survey of Logic Simulators," *VLSI Systems Design Staff*, February 1986, p. 32.  
 "The Quick Simulator Benchmark," *David L. Greer*, November 1987, p. 40.  
 "Techniques for Logic and Fault Simulation," *Ernst Ulrich and Itsuo Suetsugu*, October 1986, p. 68.  
 "Transferring Simulation Models in EDIF," *Wayne Angevine*, May 4, 1987, p. 32.  
 "Your Logic Simulation Is Only as Good as Your Board Layout," *Robert Cutler*, July 1987, p. 40.

See also Hardware accelerators

## M

**Macrocells.** See Cell-based design, Gate arrays

### Market forecasts

- "Employing Semicustom: A Study of Users and Potential Users," *Victoria L. Hinder and Andrew S. Rappaport*, May 20, 1987, p. 6.

"Vendors' Views: Semicustom Today and Tomorrow," May 20, 1987, p. 28.

#### **Memory design**

"A Memory Interface Chip Designed for Fault Tolerance," *Karl E. Grosspietsch, Lothar Muhlack, Karl L. Paap, and Guenther Wagner*, June 1987, p. 112.

See also Gate arrays, Microprocessors, Supercomputers

#### **Microprocessors**

"Memory Management in the 68030 Microprocessor," *Kirk Holden, David Mothersole, and Raju Vegesna*, February 1987, p. 88.

"Microprocessor Cache Coherency," *David Schanin*, August 1987, p. 40.

"A Perspective on Standard VLSI Circuits," *Roderic Beresford*, March 1986, p. 90.

See also Parallel processing, Reduced instruction set computers, RISC technology

#### **Military VLSI technology**

"Qualifying the Military ASIC Vendor," *Shelly Mattson, Don Howland, and Walt Buchanan*, December 1987, p. 32.

See also Gallium arsenide, Silicon on insulator

**Module generation.** See Cell compilers

**Multipliers.** See Circuit design

#### **Multiprocessing**

"Algorithmically Accelerated CAD," *Richard D. Nielson*, February 1986, p. 65.

"The Butterfly Parallel Processor," *VLSI Systems Design Staff*, March 1986, p. 20.

"A Multiprocessor Design in Custom VLSI," *David Jurasek, William Richardson and Doran Wilde*, June 1986, p. 26.

"Parallel Computing for VLSI Circuit Simulation," *Jeffrey T. Deutsch, Thomas D. Lovett, and Michael Lee Squires*, July 1986, p. 46.

"A Parallel Processing Computer with Hardware-Based Concurrency Control," *Stan Lackey and Dave Peck*, February 1986, p. 26.

See also Parallel processing

## **P**

#### **Packaging**

"Board and Microwave Chip Carriers for GaAs Systems," *Richard Davis, Jerry Schappacher, and Scott Gilstad*, June 1986, p. 125.

"IC Packaging: An Introduction for the VLSI Designer," *Dean P. Johnson and Jim Lipman*, June 1986, p. 108.

"Microcoaxial Board Interconnect," *Leonard Schieber*, March 1986, p. 57.

"Multi-Chip Packaging," *Sanford Lebow*, November 1986, p. 92.

"The Wafer Transmission Module," *Capt. B.J. Donlan, J.F. McDonald, R.H. Steinvorth, M.K. Dhodhi, G.F. Taylor, and A.S. Bergendahl*, January 1986, p. 54.

See also Printed circuit board design and manufacture, Surface-mount technology, Wafer-scale integration

#### **Parallel processing**

"A Balanced Scalable Parallel Processor," *Shaiy Pilpel*, March 1987, p. 80.

"Design Methodology for a VLSI Multiprocessor Workstation," *Shing Kong, David Wood, Garth Gibson, Randy Katz, and David Patterson*, February 1987, p. 46.

"Floating-Point Acceleration in Programmable Logic," *Don Faria and Bob Duncan*, April 1987, p. 102.

See also Multiprocessing

#### **Parameter extraction**

"Choosing Parameter Extraction Software," *Joe Domitrowich*, July 1987, p. 64.

#### **Parasitics**

"Ground Pin Optimization in High-Speed Digital Systems," *Robert D. Cutler*, March 1986, p. 46.

"Transmission-Line Analysis of PC Boards," *Juliusz Poltz and Al Wexler*, March 1986, p. 38.

**Peripheral hardware.** See Cell-based design, Computer and system design, Gate arrays

**Placement.** See Automatic IC layout, Floorplanning, Printed circuit board design and manufacture

#### **Programmable logic**

"Automatic Generation of Functional Tests for PLDs," *Karen Blyda and Peter de Bruyn Kops*, April 1986, p. 53.

"Automatic Partitioning of Programmable Logic Devices," *Ronald R. Munoz and Charles E. Stroud*, October 1987, p. 74.

"In-Circuit Emulation for ASIC-Based Designs," *Pardner Wynn*, October 1986, p. 38.

"A Pilgrim's Progress through Programmable Logic Land," *Rob W. Hartwig and Chuck Hastings*, October 1987, p. 46.

"A Pin-Logic Gate Array for a Programmable Logic Programmer," *Kelle Crisafulli*, October 1986, p. 30.

"PLD Enhancements for Implementing Subsystems," *Keith H. Gudger*, October 1986, p. 48.

"Programmable Logic Overview," *Ernest Meyer*, October 1987, p. 62.

"Survey of Programmable Logic Devices," *VLSI Systems Design Staff*, October 1986, p. 55.

"Wait-State Remover Improves System Performance," *Kapil Shankar*, November 1986, p. 102.

See also Computer and system design, Encryption

#### **Printed circuit board design and manufacture**

"The Impact of VLSI on PCB Design," *Lee W. Ritchey*, August 1986, p. 86.

"Routing Algorithms and Grids for Advanced Board Design," *R. Anastasi*, September 1987, p. 54.

"A Second-Generation Routing Accelerator for PCB Designs," *Frank Weisenberger and David Rager*, December 1986, p. 20.

"Strategies for Complete PCB Autorouting," *Pat Meyer and Robert Lewis*, September 1987, p. 60.

"Survey of Circuit Board CAD Systems," *VLSI Systems Design Staff*, March 1986, p. 64.

"Survey of PCB Layout CAD Systems," *VLSI Systems Design Staff*, September 1987, p. 64.

"Survey of PCB Service Bureaus," *VLSI Systems Design Staff*, May 1986, p. 88.

"Transmission-Line Analysis of PC Boards," *Juliusz Poltz and Al Wexler*, March 1986, p. 38.

"The VLSI-to-Substrate Connection," *Kwaku Mensah*, December 1986, p. 68.

See also Logic simulation

#### **Prototyping**

"Emulation of VLSI Devices using LCAs," *Nick Schmitz*, May 20, 1987, p. 54.

"PLD Breadboarding of Gate Array Designs," *Michael D. McClure*, February 1987, p. 36.

See also Foundries

## **R**

**Radiation-hardened ICs.** See Bipolar technology, CMOS technology, Gallium arsenide

#### **Reduced instruction set computers**

"RISC VLSI Design for System-Level Performance," *Chris Rowen, Les Crudele, Dan Freitas, Craig Hansen, Ed Hudson, John Kinsel, John Moussouris, Steven Przybylski, and Tom Riordan*, March 1986, p. 81.

#### **RISC technology**

"Putting RISC Efficiency to Work in CISC Architectures," *Ronald D. Bernal and Joseph C. Circello*, September 1987, p. 46.

**Routing.** See Automatic IC layout, Printed circuit board design and manufacture

**RTL (register-transfer-level) simulation.** See Behavioral simulation, Logic simulation

## **S**

**Schematic capture.** See CAE systems

#### **Semicustom ICs**

"An Introduction to Linear Semicustom Design," *Pierre Irissou and*

Eugene Lee, July 1986, p. 24.

"Survey of ASIC Design Centers," *VLSI Systems Design Staff*, January 1986, p. 60.

"Survey of Semicustom IC Distributors," *Judy Elster*, April 1986, p. 30. See also Cell-based design, Gate arrays, Market forecasts, Silicon compilers

#### Signal processing

"High-Performance Graphics via Silicon Compilation," *Michael Jones and Michael Bailey*, March 1987, p. 30.

"The Use of Silicon Compilation in the Design of a Gaussian Filter and a Template Matching Processor," *Mira Majewski and Srinivasan Pichumani*, October 1987, p. 20.

#### Silicon compilers

"Inside a 2901 Datapath Compiler," *Jim Rowson and Bill Walker*, May 1986, p. 40.

"Intelligent Compilation," *David L. Johannsen, Ken McElvain, and Steve K. Tsubota*, April 1987, p. 40.

"A Microprocessor Datapath Synthesized with a Translator from Schematics to Silicon," *S. Trimmerger and Flo Paroli*, April 1986, p. 26.

"Selecting a Silicon Compiler," *Al Baker*, May 1986, p. 52. See also Analog MOS design, Cell compilers, Logic synthesis, Signal processing

#### Silicon on insulator

"Silicon on Insulator: Substrates for Tomorrow's VLSI?" *Wade Krull and Ken Ports*, December 1987, p. 22.

**Simulation.** See Behavioral simulation, Circuit simulation, Fault simulation, Hardware accelerators, Logic simulation, Switch-level simulation

**Standard cells.** See Cell-based design

**Sticks.** See Symbolic layout

#### Supercomputers

"A Cryogenically Cooled CMOS VLSI Supercomputer," *Tony Vacca, David Resnick, David Frankel, Randall Bach, James Kreilich, and Douglas Carlson*, June 1987, p. 80.

"A Multiport Register-File Chip for the CHoPP Supercomputer," *Gary R. Burke*, August 1987, p. 19.

#### Superconductivity

"Properties of Superconducting Electronics," *Sadeg M. Faris and James M. Barry*, April 1986, p. 68.

#### Surface-mount technology

"CAD for Surface Mount: Still a No-Via Zone," *Ernest L. Meyer*, February 1986, p. 74.

#### Symbolic layout

"Layout-to-Layout Compaction for Technology Conversion," *Jonathan W. Greene*, November 1986, p. 46.

"Sticks Layout Notation for MESFETs and Refractory-Metal FETs," *Fayez El Guibaly and M.I. Elmasry*, February 1986, p. 88.

#### Systolic architecture

"A Fast Systolic FIFO Queue," *Asim J. Al-Khalili and Ziad Ali*, May 1986, p. 76.

## T

#### Testability

See Built-in self-test, Gate arrays

#### Testability analysis

"A Fast 20K Gate Array with On-Chip Test System," *Ron Lake*, June 1986, p. 46.

"Gate Array Testability: A Customer Perspective," *Ernest L. Meyer*, June 1986, p. 46.

"On-Chip Testability Circuit for CMOS Gate Arrays," *Kunnau Chen*, January 1986, p. 48.

#### Test generation

"The ASIC Designer's Test Engineering Responsibilities," *James H. Walker*, February 1986, p. 56.

"Automated Test Generation for Integrated Circuits," *Ralph Marlett*, July 1986, p. 68.

"Automatic Generation of Functional Tests for PLDs," *Karen Blyda and Peter de Bruyn Kops*, April 1986, p. 53.

"A Method for the Extensive Verification of Programmable VLSI Devices," *Renato Gadenz and W. Patrick Hays*, July 1986, p. 84.

"Stimulus Data Interchange Format, Part 1: Test Issues," *Chris Pieper*, July 1986, p. 76.

"Stimulus Data Interchange Format, Part 2: Test Specifications," *Chris Pieper*, August 1986, p. 56.

"Tools for Test Development," *William E. Den Beste*, July 1986, p. 92.

#### Testing

"A Designer's View of Automatic Test Equipment," *VLSI Systems Design Staff*, September 1986, p. 96.

"E-Beam Probing for VLSI Circuit Debug," *Neil Richardson*, August 1987, p. 24.

"IC Prototype Verification: Test and Tribulation," *Stephen Evanczuk*, April 1986, p. 44.

"A Laboratory System for Statistical IC Characterization," *Jeff Anderson and George Ansel*, April 1986, p. 79.

"A Mixed-Signal Test Program Development Environment," *Mogens Ravn*, August 1987, p. 32.

"Systematic and Structured Methods For Digital Board Testing," *F.P.M. Beenker*, January 1987, p. 50.

"Why a Test Chip?," *Susana Stoica*, May 20, 1987, p. 36.

See also Test generation

#### Timing analysis

"The ADEPT Timing Simulation Algorithm," *Peter Odryna and Sani Nassif*, March 1986, p. 24.

"A Mixed-Level Timing Verifier for Digital Systems," *Zhong Mo, Yen-Jen Oyang, and Sang S. Wang*, March 1987, p. 74.

"Timing Analysis of MOS VLSI Circuits," *You-Pang Wei and Cliff Lyons and Shawn Hailey*, August 1987, p. 52.

"Timing-Driven Layout of Cell-Based ICs," *Steven Teig, Randall L. Smith, and John Seaton*, May 1986, p. 63.

"Timing Verification for System-Level Designs," *Michael Chiang and Michael Bloom*, December 1987, p. 46.

See also Tracing

#### Tracing

"A Real-Time Performance Analyzer," *Max Baron and Sorin Iacobovici*, May 4, 1987, p. 44.

## V

#### Vector processing

"Converting SPICE to Vector Code," *Bruce Greet*, January 1986, p. 30.

**Verification.** See Behavioral simulation, Circuit simulation, IC layout systems, Logic simulation, Switch-level simulation, Timing analysis

**VHSIC.** See Military VLSI technology

**VLSI design systems.** See CAD systems

## W

#### Wafer-scale integration

"The Wafer Transmission Module," *Capt. B.J. Donlan, J.F. McDonald, R.H. Steinworth, M.K. Dhodhi, G.F. Taylor, and A.S. Bergendahl*, January 1986, p. 54.

"Yield of Wafer-Scale Interconnections," *J.F. McDonald, Capt. B.J. Donlan, R.H. Steinworth, H. Greub, M. Dhodhi, J.S. Kim, and A.S. Bergendahl*, December 1986, p. 62.

#### Workstation libraries

"Automatic Translation of Logic Models for Workstation Libraries," *Mehdi Amani, Gary Griffin, and Scott Hamm*, December 1987, p. 56.

**Workstations.** See Automatic IC layout, CAE systems, IC layout systems, Printed circuit board design and manufacture

# ADVERTISERS' INDEX

Reader Service#		Page Number
16	AIDA Corporation.....	75
21	Applied Microcircuits Inc.....	17
1	ASIX Systems Corporation.....	1
17	Case Technology Inc.....	85
	Daisy Systems Corp.....	7
12	ECAD.....	45
13	Ferranti Interdesign Inc.....	49
3	FutureNet/A Data I/O Company.....	CV4
18	GenRad.....	83
5,6	HiLevel Technology.....	32
10	IKOS Systems, Inc.....	55
11	Intergraph Corp.....	35
120	LSI Logic.....	87
	Mentor Graphics.....	39
20	NEC Electronics Inc.....	25
8	OKI Semiconductor.....	21
9	Personal CAD Systems.....	28
7	SMOS Systems Inc.....	15
4,19,2	Tektronix/CAE Systems Division.....	4,128, CV3
14	Tektronix Inc.....	53
	Valid Logic Systems.....	57
15	VLSI Technology, Inc.....	69

This index is provided as an additional service. The publisher does not assume any liability for errors or omissions.

# 4

# The Gate Array WorkSystem™ Makes Layout As Easy As Pushing A Button.

IN A SERIES

**Tektronix  
Aided  
Engineering**



Now logic designers can control the entire physical layout of gate array designs. From a single schematic entry environment. Just by pushing a button.

Developed by Tektronix as part of Tektronix Aided Engineering, the Gate Array WorkSystem eliminates the need for IC layout expertise. Because it gives you everything you need to quickly develop ASIC vendor-certified layouts.

And since you're controlling the layout from the schematic, you can tune your design using iterations of simulation and automatic layout to achieve your performance requirements.

The Gate Array WorkSystem creates a unique, performance-driven design environment integrating Designer's Database Schematic Capture (DDSC™) and industry-standard HILO®-3 logic simulation with MERLYN-G™ automatic physical layout.

The system also introduces vendor-certified TurnChip® ASIC Layout Modules for knowledge-based, automatic control of MERLYN-G layout of specific array families.

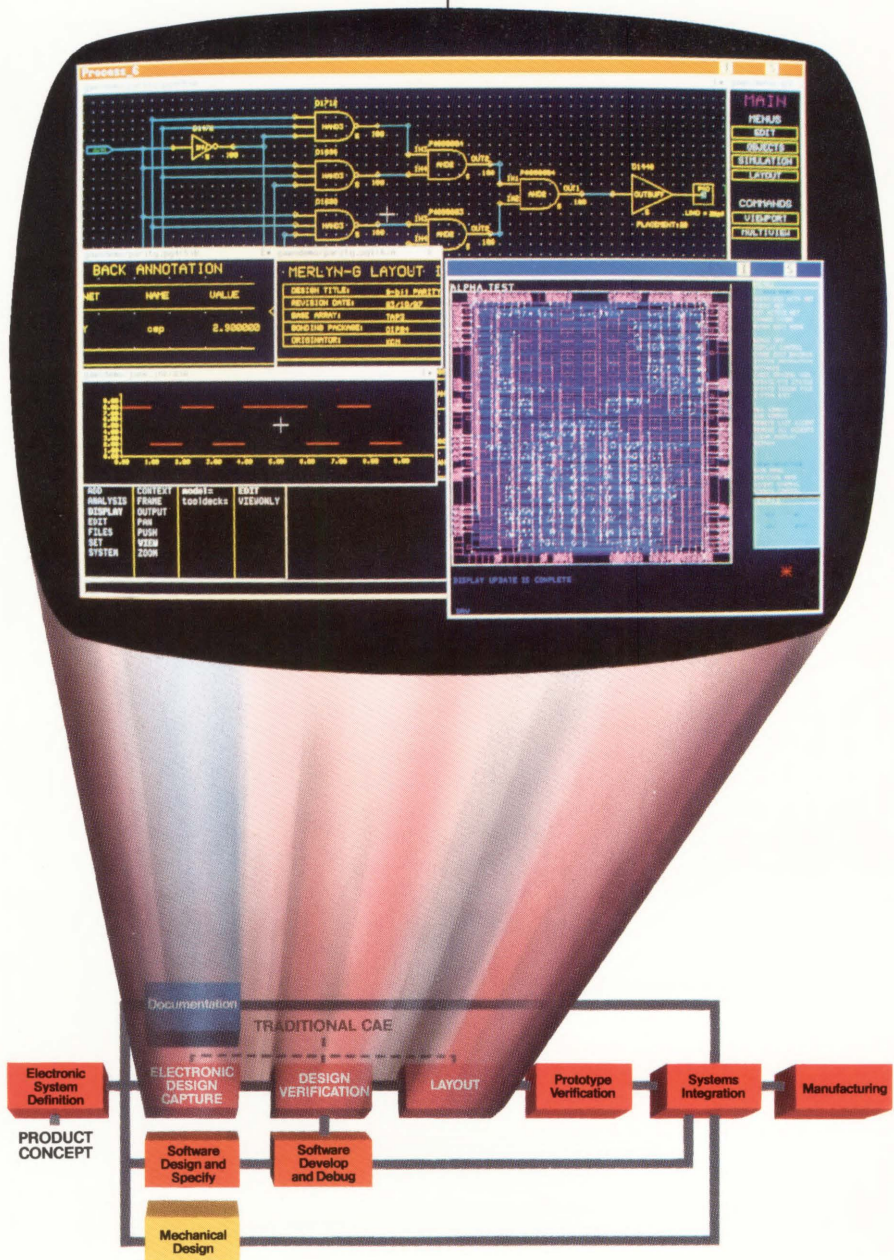
Layout so automated you can place and route a 5000-gate array 100% automatically. Just by pushing a button. With results so accurate that your layout is ASIC vendor-endorsed.

Using TurnChip modules, you can generate ASIC vendor-certified layout designs, then send them directly to the ASIC vendor. Which cuts your design time, lowers your costs and delivers complete control of your sensitive design data.

It's all part of Tektronix Aided Engineering. Integrated WorkSystems that take you beyond traditional CAE solutions. And into prototype verification, software development and testing, systems integration, mechanical design and manufacturing. All running on industry-standard platforms from Apollo® and DEC™. Best of all, it's from Tektronix. The name

you've always trusted to get the engineering job done. So you're assured of worldwide service, support and training.

If you'd like to take control of physical layout, contact your local Tektronix, CAE Systems Division, sales office. Or call 800/547-1512. Tektronix, CAE Systems Division, P.O. Box 4600, Beaverton, OR 97076-4600.



WorkSystem, DDSC, and MERLYN-G are trademarks of Tektronix, Inc. TurnChip is a registered trademark of Tektronix, Inc. HILO is a registered trademark of GenRad, Inc. Apollo is a registered trademark of Apollo Computer, Inc. DEC is a trademark of Digital Equipment Corp.

