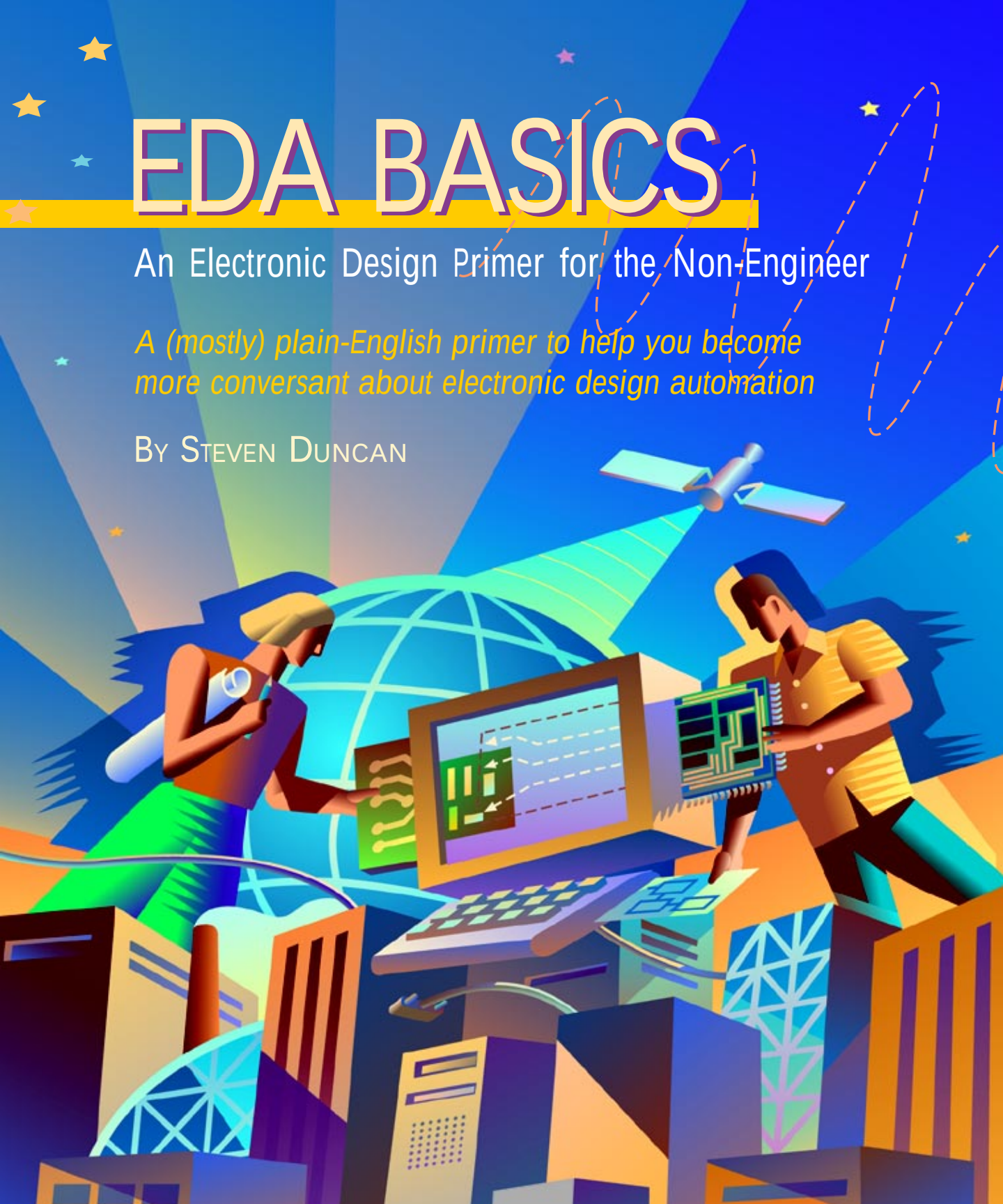
EDA BASICS

An Electronic Design Primer for the Non-Engineer

A (mostly) plain-English primer to help you become more conversant about electronic design automation

By Steven Duncan



EDA BASICS

An Electronic Design Primer for the Non-Engineer

A (mostly) plain-English primer to help you become more conversant about electronic design automation

By Steven Duncan
Technical Consultant
Hewlett-Packard Company

© Copyright Hewlett-Packard Company 1998.

The information contained in this document is subject to change without notice. Hewlett-Packard Company makes no warranty of any kind with regard to the information in this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard Company shall not be liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance or use of this material.

This documentation contains information which is protected by copyright. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, transmitted, adapted, or translated, in any form or by any means, electronic, mechanical, photocopying, or otherwise, unless the document is copied verbatim, in its entirely, with all Hewlett-Packard copyrights clearly displayed exactly as in the original, and providing that the resulting derived work is distributed under the terms identical to this one.

About the applications mentioned in this document...

The applications mentioned in this document are trademarks of their respective owners. Although mergers and acquisitions occur frequently (such as Synopsys and Viewlogic), I have chosen to continue to separate some product lines at this time. Tool names mentioned were chosen because they are offered by major ISVs or because they are the leading suppliers in their respective fields.

Illustrations courtesy of Mentor Graphics Corporation. All rights reserved.

Table of Contents

Part 1: The Beginnings of EDA

CHAPTER 1. INTRODUCTION
A Note to the Computer Sales Professional. 1

CHAPTER 2. BACKGROUND
Origins of EDA. 2
Benefits of EDA 2
EDA Processes or “Flows” 3
Integrated Applications versus “Point Tools” 4

Part II: Flows & Tools

CHAPTER 3. DESIGN ENTRY OR CAPTURE TOOLS
Schematic Entry 5
System Design Capture 6
HDL (VHDL, Verilog or C Language) Entry 7

CHAPTER 4. SYNTHESIS TOOLS
ASIC and FPGA Logic Synthesis 8

CHAPTER 5. SIMULATION TOOLS
Logic Simulators 10
Timing Simulators 11
Fault Simulators 12
Circuit and Analog Simulators 13
Mixed-Level, Analog-Digital Simulators 14
Hardware/Software Co-Verification 14
Logic Emulation and Hardware Modeling Systems 16

CHAPTER 6. DESIGN FOR TESTABILITY TOOLS
DFT Tools—Partial Scan, Full Scan, Boundary Scan and BIST 17
ATPG—Automatic Test Pattern Generation 19

CHAPTER 7. DESIGN FOR MANUFACTURABILITY (DFM) & THERMAL ANALYSIS TOOLS
DFM—Design for Manufacturability Tools 20
Thermal Analysis Tools 21

CHAPTER 8. IC PHYSICAL DESIGN AND LAYOUT TOOLS
Layout or Polygon Editors 22
Layout Generator and Optimization Tools 23
Floorplanning and Routing Tools 23
Intellectual Property—Reusable Design Cores 24

CHAPTER 9. IC VERIFICATION TOOLS
Design Rule Checking (DRC) 25
Layout vs. Schematic (LVS) 25
Layout Parameter Extraction 26
Device and Process Modeling 26

CHAPTER 10. PCB DESIGN AND ANALYSIS TOOLS
Library and Geometry Creation and Management 27
Automatic and Manual Parts Placement 27
Automatic and Manual Routing Tools 28
High-Speed Analysis Tools 29
Interconnect Synthesis 30

Part III: EDA Hardware Recommendations

Low-End UNIX or NT Workstations for EDA 31
Mid-Range UNIX Workstations for EDA 32
High-End UNIX Workstations and Servers for EDA 32
High-End UNIX Parallel Servers for EDA 33

PART IV: EDA Terminology Glossary

Glossary 35

Preface & Author's Note

Although you can use this document as a reference, it is logically ordered so if read from start to finish you will build on the terms and concepts introduced in the previous sections. Even though it was authored for the computer sales professional, it could also be of interest to others in related supporting fields.

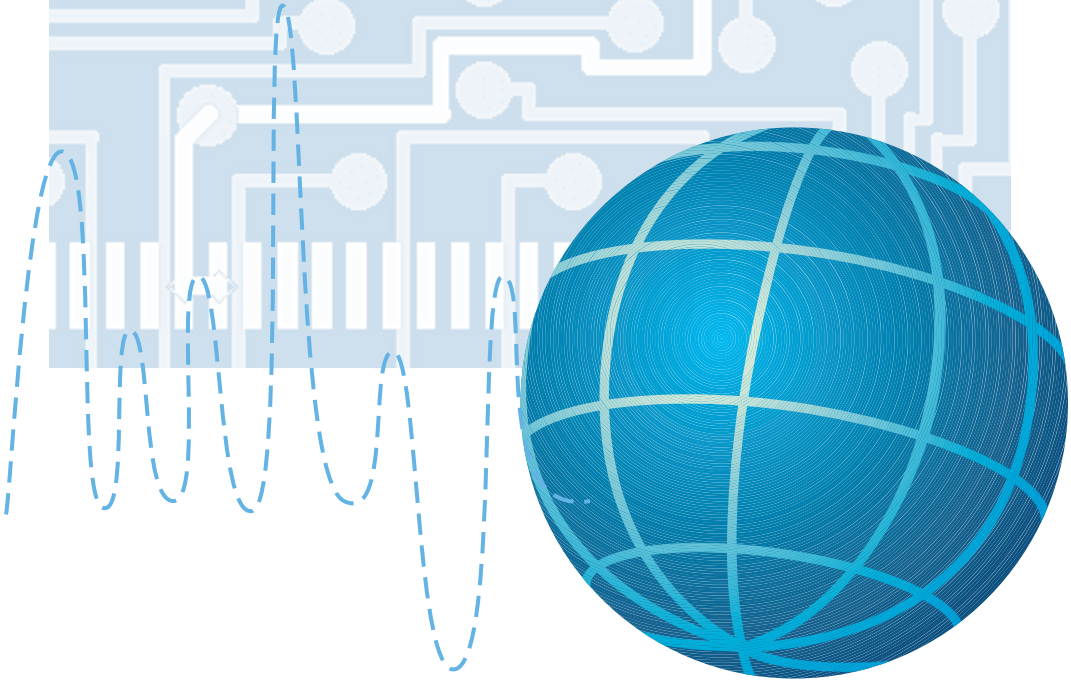
In my more than 20 years in the electronics industry (which has included a great deal of communications and training), I have come to believe that any concept—no matter how detailed—can be explained in terms that will not under or oversimplify the subject matter. This is the principle I have tried to apply here.

So take a few minutes, and become an EE without a degree! No pocket protector requited.

—Steve Duncan

Part I

The Beginnings of EDA



CHAPTER 1 INTRODUCTION

A Note to the Computer Sales Professional

With the staggering number of EDA software applications, it would be impossible for the average sales person to completely understand each one's function and hardware requirements. With a little background, however, he or she can have an intelligent dialog to help move the sales process along. An understanding of the customer's processes can also increase the salesperson's confidence level and competitiveness. After all, HP is traditionally the engineer's choice; so who is better equipped to satisfy the need in this market?

This document will attempt to explain the processes and tools in plain English, but will introduce terms you may hear your customer use. Each new term will be defined when it is first used, and more comprehensive definitions may be found in the glossary.

CHAPTER 2 BACKGROUND

Origins of EDA

Although the 1970s gave birth to early simulation and CAD tools, modern, graphical EDA *(Electronic Design Automation)* applications were first marketed in the early 1980s. Slow and almost unusable, these tools still sold for tens of thousands of dollars. Huge companies flocked to the small ISVs (*Independent Software Vendors*) and frequently bought tools without any evaluation, all in a rush to improve time to market.

Initially branded CAE *(Computer-Aided Engineering)* by the industry, the term eventually migrated to EDA to provide an umbrella for all the disciplines that make up the entire design process—not just the electronic engineering aspects.

Benefits of EDA

What does industry get for its investment in these incredibly complex tools and processes? It's important to understand the customer's needs since HP hardware is an important part of the equation.

EDA applications give your customer the ability to:

- Manage designs of arbitrary size and complexity
- Verify functionality without the creation of a prototype
- Conceive and verify the design's functionality before the underlying circuitry is completed
- Enter electronic designs as textual descriptions and let the computer generate the circuitry

- Try different densities and package options to observe impact on performance, cost and reliability
- Build in manufacturability and testability
- Ensure speed and timing goals are met
- Design independently of the fabrication process
- Automate the actual physical layout processes

If there is an ultimate goal in EDA, it is perhaps the ability to design, test and revise the design as quickly as possible. When performed rapidly, these iterations (called *design turns*), can dramatically shorten time to market. This is where faster hardware becomes a tremendous differentiator.

EDA Processes or Flows

Perhaps more important than learning about individual EDA tools is to understand the design processes or *flows*. These flows take a design from an engineer's concept, to a logical representation, to *physicalization*—the data needed to fabricate a chip, a circuit board, or even a complete system.

While the flows may differ, most contain the same basic steps:

- Design Entry or “Capture”
- Logic Synthesis
- Logic, Fault and Circuit Simulation
- Designing in Testability and Manufacturability
- Physical Design Verification
- Layout of the Silicon or Printed Circuit Board (PCB)

Integrated Applications Versus Point Tools

Initially, EDA vendors tried to provide tool sets that took the design all the way to completion in a highly integrated fashion. Customers soon convinced them that they wanted to choose applications “a la carte”, and integration soon gave way to standard data interfaces. Customers could then assemble their own flows from the vendor’s *point tools*—single applications that fit into a heterogeneous tool environment.

Part II

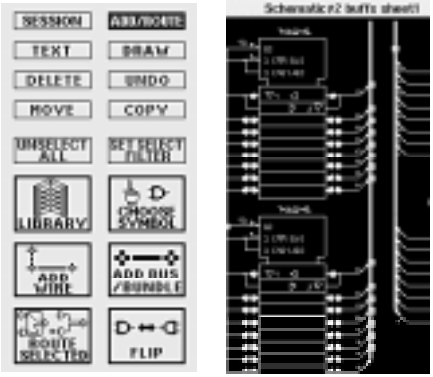
Flows & Tools

CHAPTER 3 DESIGN ENTRY OR "CAPTURE" TOOLS

The first step in creating any electronic design used to be drawing a diagram on paper. Now, designs are entered directly into the computer to create a *virtual design* that will go all the way from concept to fabrication.

Schematic Entry

Category: **Interactive** Compute requirements: **Low**



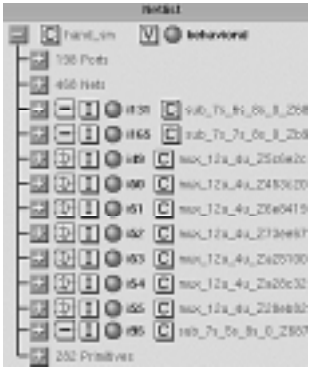
Left. A static menu, also called a palette
Right. A schematic sheet in the process of being edited

Long the mainstay of electronic designers, schematics still play an important role to document the design and drive downstream applications. Parts are picked from a menu, placed on the schematic sheet and wired together by the designer or technician. In addition to the series of visual pages (or *sheets*) that are created, the output is an electrical representation of the design, often called the *netlist*—basically a parts and connection list. Netlists come in many flavors, but the most common is EDIF, the *Electronic Design Interchange Format*. Schematics can also be automatically generated after the synthesis process, which is described in Chapter 4.

Schematic entry tools: Cadence Composer; Mentor Graphics Design Architect; Viewlogic ViewDraw.

System Design Capture

Category: **Interactive** Compute requirements: **Low**



The hierarchy of an ASIC design displayed in a browser window

Most designs these days end up as part of a system, and each module within a system has a clearly defined function. These modules have inputs and outputs that provide for the flow and control of data. By starting at a high level of abstraction, architectural "what-if" experiments can take place—determining the optimal design of each block or subsystem independently of the full system.

System design capture tools allow the designer to enter complete systems using a combination of graphical and textual methods:

- Block diagrams to represent module functions and signal paths
- Flowcharts that define logical operations
- State Machine diagrams to describe the transformation of data within a design
- Truth tables that generate predefined outputs based on input signals
- Textual HDL (*hardware description language*) to describe module behavior

Once a system is entered, some initial verification can occur. Unlike a detailed logic simulation, system-level verification makes sure that data types match up, blocks communicate with the correct protocol and the original functional requirements are still met. Many of these tools can generate synthesizable HDL from the completed descriptions.

System design capture tools: Summit Design Visual HDL; Mentor Graphics Renoir; Cadence Concept-HDL and BONeS; Viewlogic State CAD.

HDL Entry

Category: **Interactive** Compute requirements: **Low**

```
Source

55   z <= not a after tpd;
56 end behave;
57
58 library ieee;
59 use ieee.std_logic_1164.all;
60 entity nand2_gate is
61   generic(tpd : time := 1 ns);
62   port(a,b : in std_ulogic;
63        z: out std_ulogic);
64 end nand2_gate;
65
66 architecture behave of nand2_gate is
67 begin
```

A sample of VHDL source code

Designing with schematics and library parts becomes hard to manage with large designs, and it becomes almost impossible to make sure the design's functionality continues to match the original specification. To break these complexity barriers, high-level design languages were invented. The most common of these languages are Verilog and VHDL. The C language is also used as a modeling method in some proprietary simulation tools.

HDLs allow the designer to write a textual description of the circuit's behavior that becomes the specification for a given design or module. HDL can be written in a free-form fashion called *behavioral*, or in a more structured way known as RTL—for *register transfer level*.

RTL code is usually the objective, because traditional logic synthesis tools cannot synthesize behavioral HDL. HDL entry is largely text manipulation and compilation, accounting for the relatively low CPU demand.

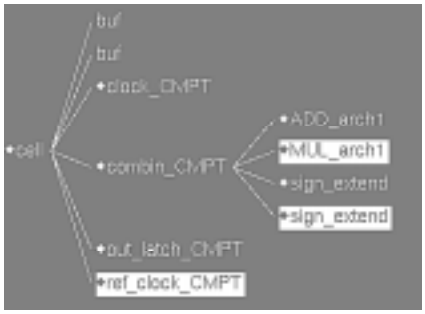
HDL entry tools: See "System design capture tools," page 6. Standard text editors are also used for HDL entry.

CHAPTER 4 SYNTHESIS TOOLS

Modern ICs are too complex for even a team of engineers to design by hand. Synthesis tools let the computer produce designs that faithfully adhere to the engineer's specifications—in far less time.

ASIC and FPGA Logic Synthesis

Category: **Batch** Compute requirements: **High**



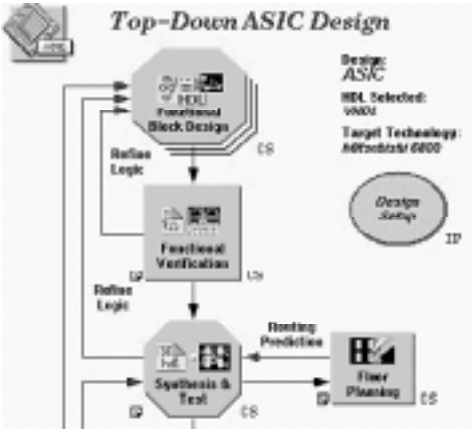
A "tree" representation of a cell within an ASIC

In 1985, a 3-5,000 gate ASIC (for *Application Specific Integrated Circuit*) was a common size. Now, gate counts can easily exceed 100,000 for a typical ASIC and continue to grow every year. Virtually all new large ASICs are designed using logic synthesis. As mentioned above, designers write an HDL file that is compiled and simulated. Once the correct operation is determined, synthesis can begin.

Selecting the target technology and HDL source for ASIC synthesis

In this manner, a team of engineers can agree on the specifications, divide the circuit into functional blocks and work independently. Later, they combine the HDL into a larger system design and put the synthesis tool to work. Synthesis runs can take hours or even days—so fast, large memory computers are required.

One of the key values of logic synthesis is the ability to target a design to a given manufacturer's ASIC or IC design process. If the device is too large or too slow, the designer can try another vendor's process. In EDA, *process* refers to the vendor's fabrication technology.



Documentation of the design and synthesis of an ASIC

FPGA (*Field Programmable Gate Array*) synthesis tools are similar in nature to their ASIC equivalents but their smaller gate counts often permit them to run on PC-class workstations.

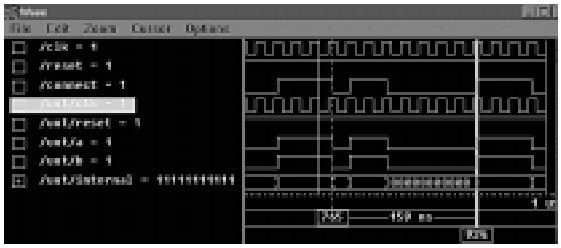
Logic synthesis tools: Synopsys Design Compiler; Mentor Graphics Leonardo; Synplicity Synplify; Viewlogic Aurora

CHAPTER 5 SIMULATION TOOLS

Now that the design is entered into the computer, the object is to verify its operation—before a prototype is built. Simulators reduce development time and costs by catching mistakes early in the design cycle.

Logic Simulators

Category: **Interactive and Batch** Compute requirements: **High**



A digital simulation waveform display

Logic simulators were some of the earliest CAE tools. They offered the promise of a design that worked the first time it was built and are now a routine step in the design process. Using a schematic, a netlist or an HDL source as input, the designer applies a set of input signals (known as *vectors*) to her design as it exists in the computer. As time is advanced, each connection point, or *node*, is evaluated for its logic level and drive strength. This is done for every signal in the design to determine the value pending for the next timestep. The results can be a text file of ones and zeros, but are more commonly represented as a graphical waveform display. In a good interactive simulator, virtual *probes* can be placed, signals forced to certain levels, and time can be advanced and rewound over and over to debug the circuit's operation.

When a signal reaches a part in the design, a software *model* is run to determine how the inputs will affect its outputs. Models for simple gates are built into most simulators, but the ASIC vendor or a third party usually provides more complex models. Models are platform-specific. This is why a new or dark horse platform is usually very slow to get support from fabs and simulator vendors.

Logic simulators can be HDL only, gate-level (including various other model types), or mixed HDL and gate. Traditional simulators are *event-driven*. Another variant called *cycle-based* simulation runs much faster at the expense of some accuracy and is defined further in the glossary.

Event-driven logic simulators: Mentor Graphics QuickSim II; Viewlogic ViewSim

HDL simulators: Cadence Verilog XL, NC Verilog, and Leapfrog; Mentor Graphics QuickHDL, Model Technology ModelSim; Viewlogic VCS and SpeedWave VHDL; Synopsys VSS

Cycle-based simulators: Synopsys Cyclone; Mentor Graphics QuickHDL-XLC; Avant! Polaris; Quickturn Design SpeedSim

Timing Simulators

Category: **Batch and Interactive** Compute requirements: **Medium to High**

A static timing tool palette

Logic simulation is only half the equation. Delays through components or wires can cause a signal to arrive too late causing circuit malfunctions. Although logic simulators do account for delays through components, they are not intended to verify proper circuit timing in all cases.

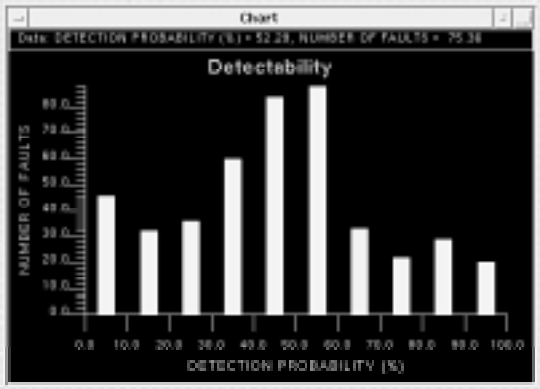
Dedicated timing analysis tools usually use one of two methods. *Dynamic timing analyzers* require a set of vectors to stimulate the circuit. They vary or *skew* the timing values during simulation to see if the circuit will operate properly under all possible delay combinations. *Static timing analyzers* do not use vectors and do not run a simulation. They simply add up all the delays along a circuit path and give the user a report of which paths have a possibility of violating timing requirements.

Static timing analyzers: Viewlogic/Quad Motive, Synopsys PathMill and PrimeTime; Mentor Graphics QuickPath and SST Velocity; Cadence Pearl

Dynamic timing analyzers: Synopsys TimeMill; VEDA HILO/HITIME

Fault Simulators

Category: **Interactive and Batch** Compute requirements: **High**



Percentage of faults likely to be detected by test cycle

Another flavor of simulator is the *fault* simulator. Instead of verifying the logical operation of the design, its ability to be production tested is evaluated. One at a time, every node in the circuit is logically *stuck*, or forced to a level of one and zero. The complete set of test vectors is then applied to determine if the "stuck at" fault will be observed on the design's outputs. At the end of the run, a report tells the effectiveness of the vectors—which will usually have to be re-engineered several times.

Because an entire simulation is run twice for each node in the design, fault simulations can take days! Some fault simulators permit parallel operation, but another technique known as *fault grading* is often applied to get the test vectors very close so only a few of the lengthier *deterministic* runs are required. Deterministic is a term applied to the scenario above, where each stuck-at fault is actually simulated. Fault graders use a mathematical technique to predict the probability that a fault will be detected, and although not 100 percent accurate, get the vectors close while saving much time.

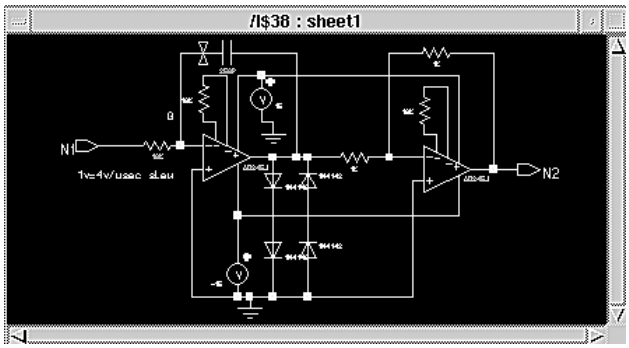
Fault simulators and graders: Viewlogic/Sunrise FaultSim; Mentor Graphics QuickFault II and QuickGrade; IKOS Voyager FS

Circuit and Analog Simulators

Category: **Batch and Interactive**

Compute requirements: **High (floating-point intensive)**

Circuit simulators differ from logic simulators in the respect that the analysis is being performed at the transistor, or *device* level—not the gate. Analog devices operate over a wide range of values—not just ones and zeros, and as such rely on floating point rather than integer calculations. The most famous circuit simulator is *SPICE*, and was conceived at UC Berkeley. Many early vendors simply wrapped graphical front-ends around the public-domain SPICE program and even today many circuit simulators are still SPICE derivatives.



A small, discreet analog design

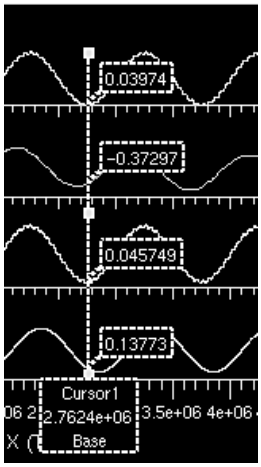
Circuit simulators are usually specialized for the IC level since the problems to be solved and the device models are so different from PCBs. *Analog simulator* usually refers to a tool used to simulate *discrete* (individual components) or PCB designs. Analog simulations are also notoriously slow. Some analog simulators can use a hardware description language specifically designed for analog called *HDL-A*.

Analog simulators: Cadence Analog Workbench; Analogy Sabre; Mentor Graphics AccuSim II and Mentor Graphics Anacad ELDO

Circuit simulators: Avant! Star-HSPICE and Star-ADM, Mentor Graphics Lsim, Cadence Spectre and Cadence SPICE; Silvaco SmartSpice

Mixed-level, Analog-Digital Simulators

Category: **Batch and Interactive** Compute requirements: **Extremely High**



Cursor indicates voltage levels in waveforms

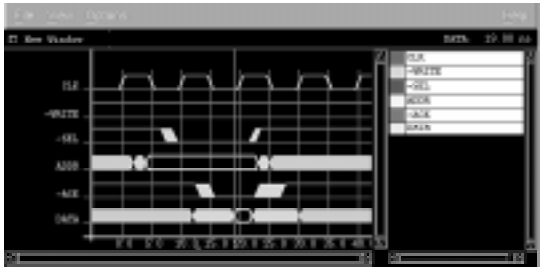
Mixed-level simulators attempt to combine analog and digital simulation technologies in one product that will perform a simulation on designs containing both flavors of circuitry. A dream of electronic designers and a nightmare for workstations, mixed-level simulators are some of the most demanding EDA applications. The simulator must understand digital-analog circuit boundaries and apply level threshold tests to determine when or if a valid one-to-zero or zero-to-one logic transition occurs.

The analog portion usually gates the performance of these tools since it is the most time-consuming. Some mixed-level simulators can even run HDL code and other modeling methods that further slow the process.

Analog/digital simulators: Mentor Graphics Continuum; Viewlogic ViewAnalog; Analogy Mixed-Signal (various); Synopsys ACE

Hardware/Software Co-verification

Category: **Interactive** Compute requirements: **High**



Bus transactions in a waveform display

This is a relatively new EDA application area. When designs contain internal or "embedded" processors, a new problem occurs for engineers—making the hardware and the instructions for the embedded processor function together correctly. The previous paradigm was for the software and hardware teams to

design independently, and combine their efforts at the prototype stage. This approach rarely worked the first time and inevitably led to compromises in the software, which was much easier to change. With co-verification, hardware and software are combined much earlier in the design phase.

Here's an example: simulating the instructions needed to get a cellular phone to ring could take days or weeks. Hardware/software co-verification attacks this problem by combining traditional simulation tools with what is called an *instruction set simulator*; software routines that model the exchanges with the processor, but at a higher-level of abstraction. When increased accuracy is required, an actual processor model can be substituted. In this manner, the simulation can be run very quickly using the instruction set simulator to the point where the phone is about to ring, then "switched" to accuracy mode for a detailed analysis of critical circuit operation.

Hardware/Software Co-verification tools: Mentor Graphics Seamless CVE; Quickturn DesignSystems Quest II; Viewlogic Eagle Tools

Logic Emulation and Hardware Modeling Systems

Category: **Interactive** Compute requirements: **High**

Software models of large chips can cause simulators to run very slow. Furthermore, a software model cannot accurately capture all the intricacies of real ICs, especially complex ones. Two related technologies address these issues.

In HARDWARE MODELING, the user installs the actual chip into a device that plugs into the workstation running the logic simulation. When the simulation needs to involve the chip, signals (*vectors*) are sent through an interface to the modeler. It stimulates the IC and reads the response on it's outputs, which is then sent back to the simulator.

LOGIC EMULATION devices are similar in the respect that the end result is a hardware model. What is different is that the chip does not have to exist yet. The designer can download a design to the emulator in the form of a netlist or other circuit description. Now, before the IC is even fabricated, it can be tested both stand-alone and as part of the system being developed.

Hardware Modelers: Synopsys Logic Modeling LM-family

Logic Emulators: Quickturn Design Systems System Realizer, CoBALT; Mentor Graphics SimExpress *(not sold in the USA)*

CHAPTER 6 DESIGN FOR TESTABILITY TOOLS

Now that the design is entered and verified, can it be tested? Customers of ICs want to know that the vendor's tests will catch the vast majority of defects that can occur in the manufacturing process.

DFT Tools - Partial Scan, Full Scan, Boundary Scan and BIST

Category: **Batch and Interactive** Compute requirements: **Medium to High**

The acronym DFT *(Design for Test)* refers more to a goal of electronic designers rather than any specific tool or process. If a design is not testable, the vendor will not be able to provide customers with a level of confidence that the part or system will be fully functional. Fault simulators provide a means to ensure vectors will detect manufacturing defects. However, in complex circuitry, internal nodes are often difficult or impossible to reach unless special DFT techniques are implemented during the design phase.

SCAN is one of these techniques. In a scan design, special circuit elements that can grab, or *latch* data are substituted into the design and wired together in a large string (called a *chain*)—much like Christmas lights. Then, in a special type of fault simulation, a pattern of data is loaded, or *shifted* into these elements. Test vectors are then applied to the circuit's primary inputs and simulated, and the data in the chain is shifted back out. The input data string is compared with the output string, and it is now possible to observe faults at points within the circuit that were previously buried inside.

In FULL SCAN, every possible circuit element that can be substituted is included in the chain.

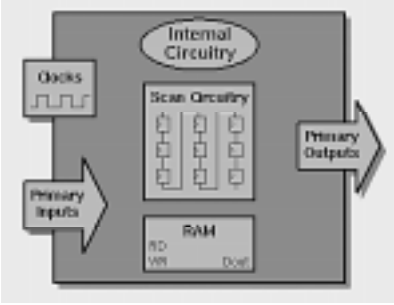
In PARTIAL SCAN, the designer or the tool makes decisions about which elements will provide the greatest benefit. Full and partial scan are both techniques for designing ICs.

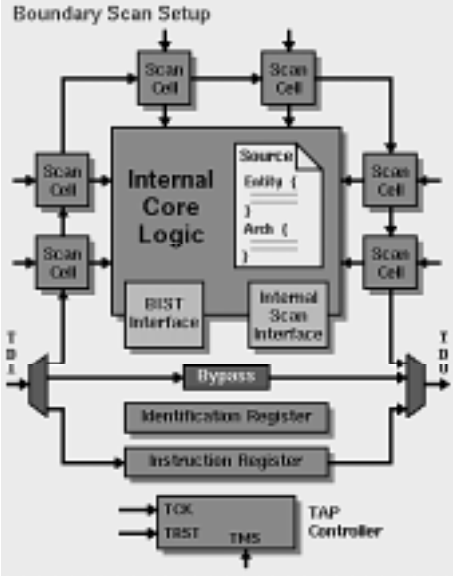
BOUNDARY SCAN addresses a separate problem; how to find faults on the pins of chips soldered onto boards. In boundary scan, additional circuit elements are placed on every pin that goes outside the chip. This method allows board testers to ensure parts are correctly inserted and soldered.

BIST, also known as embedded test, stands for *Built-In Self Test*. In BIST, some of the complexity of the chip tester is moved inside the silicon. A standard interface called the *TAP* controller communicates with internal circuitry that generates patterns and compares *signatures* returned by the logic.

LBIST, for *Logic BIST,* is an extension of scan principles.

MBIST, for *Memory BIST,* is specialized for embedded RAM arrays.

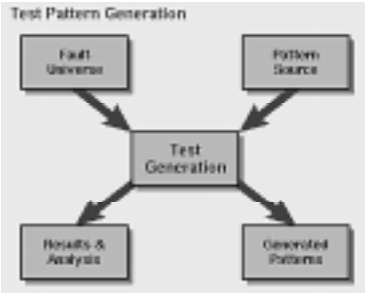




Top. A set-up menu for a popular DFT tool
Right. Block-level diagram of Boundary Scan and BIST test structures

ATPG – Automatic Test Pattern Generation

Category: **Batch** Compute requirements: **High**



High-level view of the ATPG process

Creating test vectors by hand is difficult and time consuming. ATPG tools are special-purpose fault simulators that can either create test patterns, or refine existing pattern sets to improve their effectiveness. ATPG tools are almost always used in conjunction with DFT tools and therefore create vectors for manufacturing tests rather than logic simulation. Creating patterns for non-scan or partial scan designs is a much more computationally intensive task than it is for full-scan designs.

DFT tools and ATPG tools: Mentor Graphics DFTAdvisor, FastScan, FlexTest, MBISTArchitect, LBISTArchitect; Viewlogic Sunrise TestGen and PathTest; Synopsys Test Compiler

CHAPTER 7 DESIGN FOR MANUFACTURABILITY & THERMAL ANALYSIS TOOLS

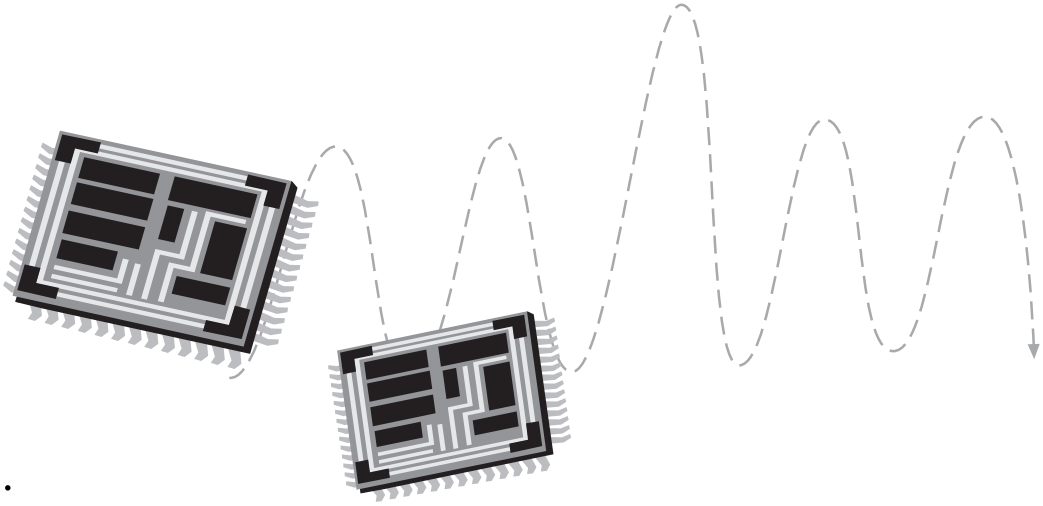
Will it run hot? Are costs too high? Will components on a board be too tall and touch the next board? These are concerns addressed by DFM and thermal analysis tools.

DFM—Design for Manufacturability Tools

Category: **Interactive** Compute requirements: **Medium**

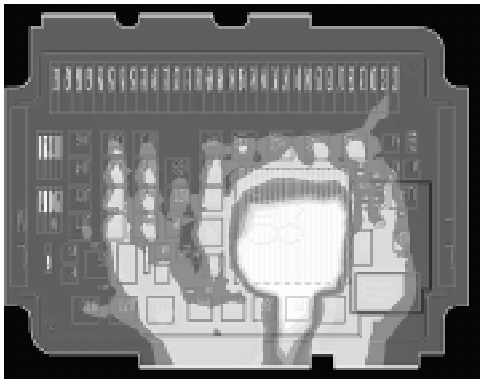
While DFT tools help ensure a design is testable, DFM tools help ensure a design is manufacturable and reliable. Manufacturability tools analyze factors such as component costs, power requirements and physical constraints. Design for Reliability tools make a prediction based on the design components' MTBF *(mean time between failure)* and thermal characteristics.

DFM Tools: Mentor Graphics Manufacturing Advisor, MDV/Scepter, Reliablity Manager

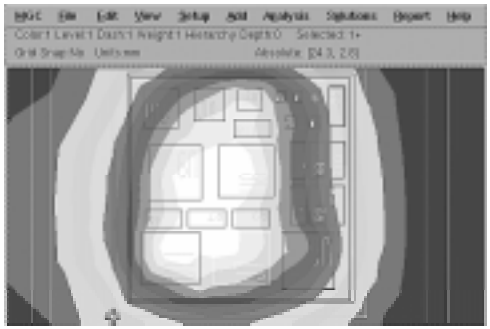


Thermal Analysis Tools

Category: **Interactive** Compute requirements: **Medium-High**



Thermal Analysis tools for PCB design show where “hot spots” occur on boards or in systems. Using the results of a thermal analysis, a PCB or systems designer can re-arrange component and fan locations to improve air flow and cooling. Data about component power, heat dissipation and PCB conductive properties must be available to perform a thermal analysis.



Thermal maps depicting heat radiating from components on circuit boards

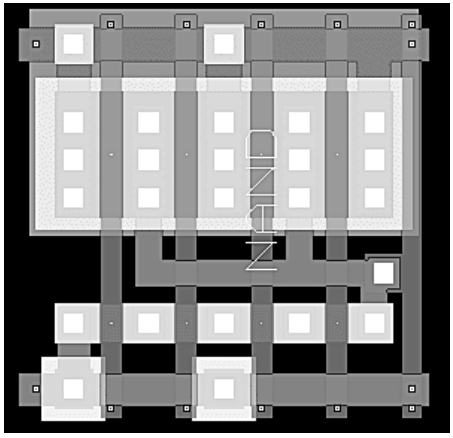
Thermal analysis tools: Mentor Graphics AutoTherm, FlowTherm; Veribest Beta-soft; Cadence DF/Thermax

CHAPTER 8 IC PHYSICAL DESIGN & LAYOUT TOOLS

IC designs were once completely hand-crafted. Now, a combination of computer generation, design reuse and traditional "polygon pushing" are used to design today's increasingly complex chips.

Layout or Polygon Editors

Category: **Interactive** Compute requirements: **Medium**



An IC layout of a simple logic function

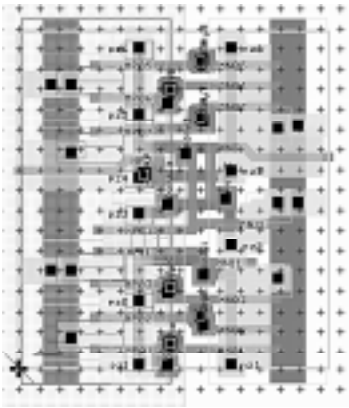
The most fundamental method of IC design involves directly creating and editing the geometries that will be placed on the material layers of the chip to become the transistors and gates. These tools are specialized drawing editors and can work in a *flat*, all at the same level, or *hierarchical* fashion, where boxes drawn at one level can represent circuitry below.

Inside the polygon editor, interactive DRC *(design rule checking)* and *extraction* (see below) can take place. In interactive DRC, process and designer-imposed rules can be verified. Violations are reported if the geometries are incorrectly manipulated by the operator. Some editors can also view schematics and correlate locations in the silicon to the circuitry in the design—an interactive form of LVS *(layout vs. schematic)*.

Polygon editing tools: Cadence Virtuoso Layout Editor; Avant! Aquarius BV; Mentor Graphics ICGraph

Layout Generator and Optimization Tools

Category: **Batch** Compute requirements: **High**



An automatically generated "macro" cell

Generator tools automate the silicon design process by taking input from a netlist or other design database description and generating physical representations. They are typically specialized for cell or gate array based designs, and further specialized for *datapath* (like computers) or memory-based designs. The designer can also supply timing constraints and critical path placement information to help the tool determine the layout of the chip.

Optimization tools take existing IC layouts and rearrange the blocks and interconnections for improved density or speed.

Generator and optimization tools: Cadence Virtuoso Compactor, SmartPath Generator; Avant! Aquarius GA and Solar; Mentor Graphics GDT Designer

Floorplanning and Routing Tools

Category: **Batch and Interactive** Compute requirements: **High**

The floorplan of a chip is a map where the functional areas, or blocks of circuitry, are placed. Their interconnections and other factors such as high-speed effects determine the location of these blocks. The end goal in most cases is to create designs that are faster and more compact—thus fitting into a smaller chip or *die* size.

When an IC has to be fabricated using the next larger die size, each one becomes more expensive to produce, raising the cost of the finished product.

After a preliminary floorplan has been determined, predicted circuit delays can be *back-annotated*, or fed back to the timing analysis tool to see if the design still meets its goals for speed.

Floorplanning and routing tools: Cadence Gate Ensemble, LogicDesign and Physical Design Planners, Preview floorplanner; Avant! Aquarius XO place and route, Planet floorplanner; Mentor Graphics AutoCells, DataPath and MicroRoute; Synopsys Floorplan Manager

Intellectual Property—Reusable Design Cores

Category and Compute Requirements: **Varies according to application**

Although the definition of *IP* can vary widely, it is generally used to refer to a design description that can resold for reuse. Even proprietary ICs contain common functions, and reinventing these functions for every design slows time to market. To speed up the process, companies are willing to pay for reusable design *cores*.

Design cores are typically either *soft* or *hard*. Soft cores are represented as an HDL or netlist description that can be targeted to any number of different IC fabricators. Hard cores are usually developed in partnership with a given IC fabricator and are already developed for use with a particular process.

IP Vendors: Mentor Graphics Inventra; Cadence Design Systems; Innovative Semiconductors; Tundra Semiconductors

CHAPTER 9 IC VERIFICATION TOOLS

These tools get a lot of attention from workstation vendors because their demands on the hardware make them some of the most compute-intensive applications. They tend to run for hours and sometimes days, which can make turning the design a very lengthy process.

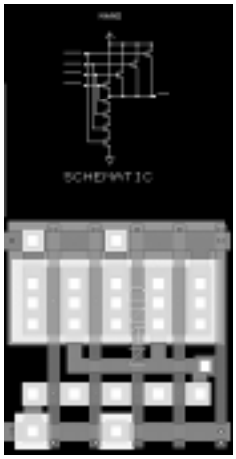
Design Rule Checking (DRC)

Category: **Batch** Compute requirements: **High**

This is the process of verifying that the layout adheres to all the constraints that make it manufacturable and ensure that it meets specifications. Limits on lengths, distances and other physical attributes are checked based on rules set by the designer or fab house. DRC tools work on layers of the chip, which allows some DRC tools to divide the job among multiple CPUs. This capability makes compute servers with lots of memory the preferred platform for these applications.

Layout vs. Schematic (LVS)

Category: **Batch** Compute requirements: **High**



LVS is a connectivity comparison performed to ensure that the physical layout matches the logical design—typically a netlist. The physical database (usually stored in a format called GDS2) is compared to the netlist—often a SPICE circuit design. Because both databases need to be at least partially resident in memory at the same time, LVS is not only time-consuming but also very memory intensive.

DRC and LVS tools: Cadence Dracula and Vampire; Mentor Graphics Calibre; Avant! Hercules

Left. Layout and corresponding schematic for a NAND gate

Layout Parameter Extraction

Category: **Batch** Compute requirements: **Medium–High**

Menu to control extraction process

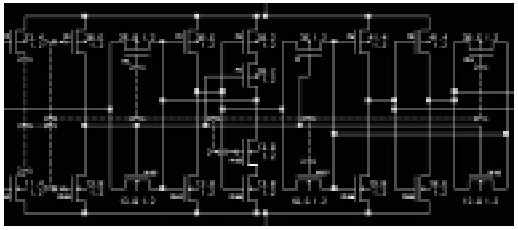
These tools extract the resistances and capacitances at nodes within circuitry that are inherent in an IC design. The values are then back-annotated to a circuit simulation, and the design can be re-simulated to ensure the physical aspects of the design will not cause it to fail to meet its functional or performance specifications.

Extraction tools: Mentor Graphics xCalibre; Avant! Star-RC; Cadence Fasnet; Synopsys Arcadia RC Extract; Sivaco Tempest

Device and Process Modeling

Category: **Interactive** Compute requirements: **Low–Medium**

Every IC design process is different, and so are its electrical characteristics. In order to simulate the IC before it is built, accurate models of the low-level "primitives" inside the chip must be created. Process modeling tools are used to resolve the electrical characteristics of the physical elements inside an integrated circuit.



Transistors that make up the underlying structure of a logic function

As design sizes shrink, the importance of accurate models is even more important. Where an estimation of the circuits operation used to be adequate, modern deep submicron designs will not function correctly unless subtleties are taken into account in the simulation models. Fabrication houses build device libraries which they provide to their customers, the circuit designers.

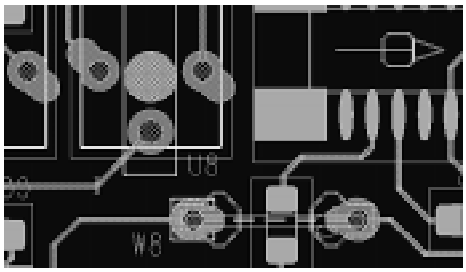
Device modeling tools: HP IC-CAP; Silvaco UTMOST

CHAPTER 10 PCB DESIGN & ANALYSIS TOOLS

While larger companies often design and lay out their own circuit boards, many others take their designs to *service bureaus*. By allowing a service bureau to perform the PCB design, a company does not have to hire the people or purchase the expensive tools required to do PCB layout in-house.

Library and Geometry Creation and Management

Category: **Interactive** Compute requirements: **Low–Medium**

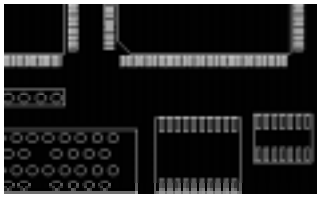


Pads, traces, and vias on a small section of a PCB

Just as a schematic is drawn using symbols that represent circuit elements, PCB design begins with the creation of symbols that represent physical components on the board. Most companies have their own custom libraries of these symbols, often called *geometries*. These symbols also carry information used to create a PCB database such as a part number, reference number (like U55 or R21) and can even carry cost, power consumption, component height and heat dissipation data.

Automatic and Manual Parts Placement

Category: **Batch and Interactive** Compute requirements: **Medium**

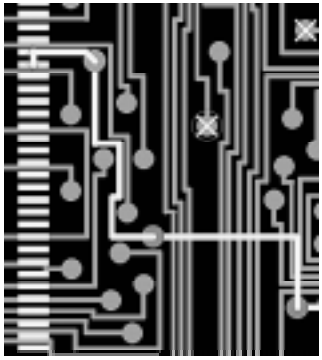


Component and connector geometries, or "footprints"

Once geometries are available for all the parts on a board, placement can begin. Placement can be manual, automatic, or a combination of both methods. Usually, the designer starts by specifying board outlines, "keep out" areas and manual placement of critical components. Then, an automated tool can make placement decisions, primarily based on the connections between parts.

Automatic and Manual Routing Tools

Category: **Batch and Interactive** Compute requirements: **High**



An automatically routed PCB design

Routing is the process of making the connections between the pins of the components with *traces*. Since there can be thousands of traces, the process is often automated.

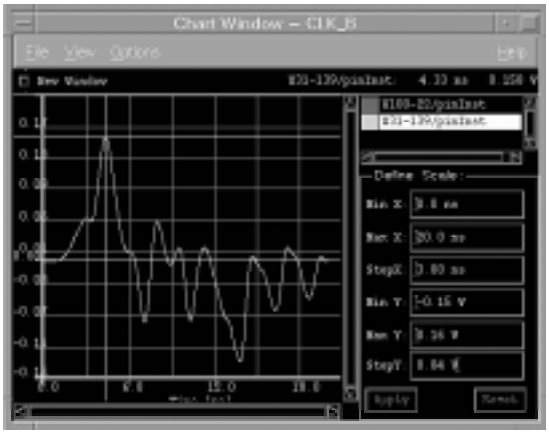
Routing applications get the majority of the wires connected, but usually leave a significant number of *unroutes*. This is where the manual *stitching* capability of the routing tool becomes critical, as do the skills of the designer who must find paths for the traces that the computer failed to route.

Complete PCB design packages: Mentor Graphics Board Station; Cadence Allegro; Zuken-Redac CADSTAR and Visula; Veribest Veribest-PCB

PCB Autorouters: Cooper and Chyan Technologies CCT Router; Cadence Specctra; Mentor ArtRouter and SmartRouter

High-Speed Analysis Tools

Category: **Batch** Compute requirements: **High**



Analysis of the "ringing" effect on a clock signal

With increasing clocks speeds, the integrity of digital logic signals can degrade or suffer from any one of a category of high-speed effects. Ringing, overshoot, undershoot, cross talk and transmission line effects are all examples of these. High-speed analysis tools depend on layout information as well as physical characteristics of the PCB to operate, such as the thickness of the insulating layers of the board. High-speed tools are a specialty niche that are often plug-ins to mainstream PCB layout tool vendors' applications.

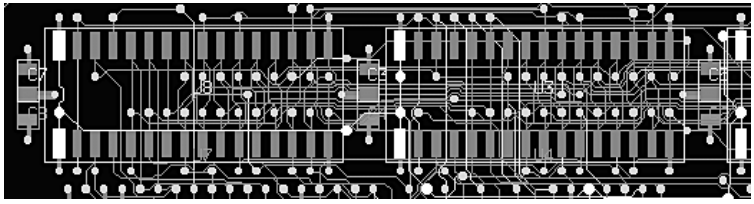
High-speed tools: Viewlogic/Quad Design XTK; Veribest Signal Vision; Cadence SigNoise; Mentor Graphics Board Station 500

Interconnect Synthesis

Category: **Batch** Compute requirements: **High**

This is the latest incarnation of high-speed design methodologies where the physical placement of components and wires on a circuit board or boards within a system are derived from an electrical analysis of the interconnections. By making the design correct by construction, lengthy re-working can be avoided with this form of early analysis by preventing high-speed effects before they materialize.

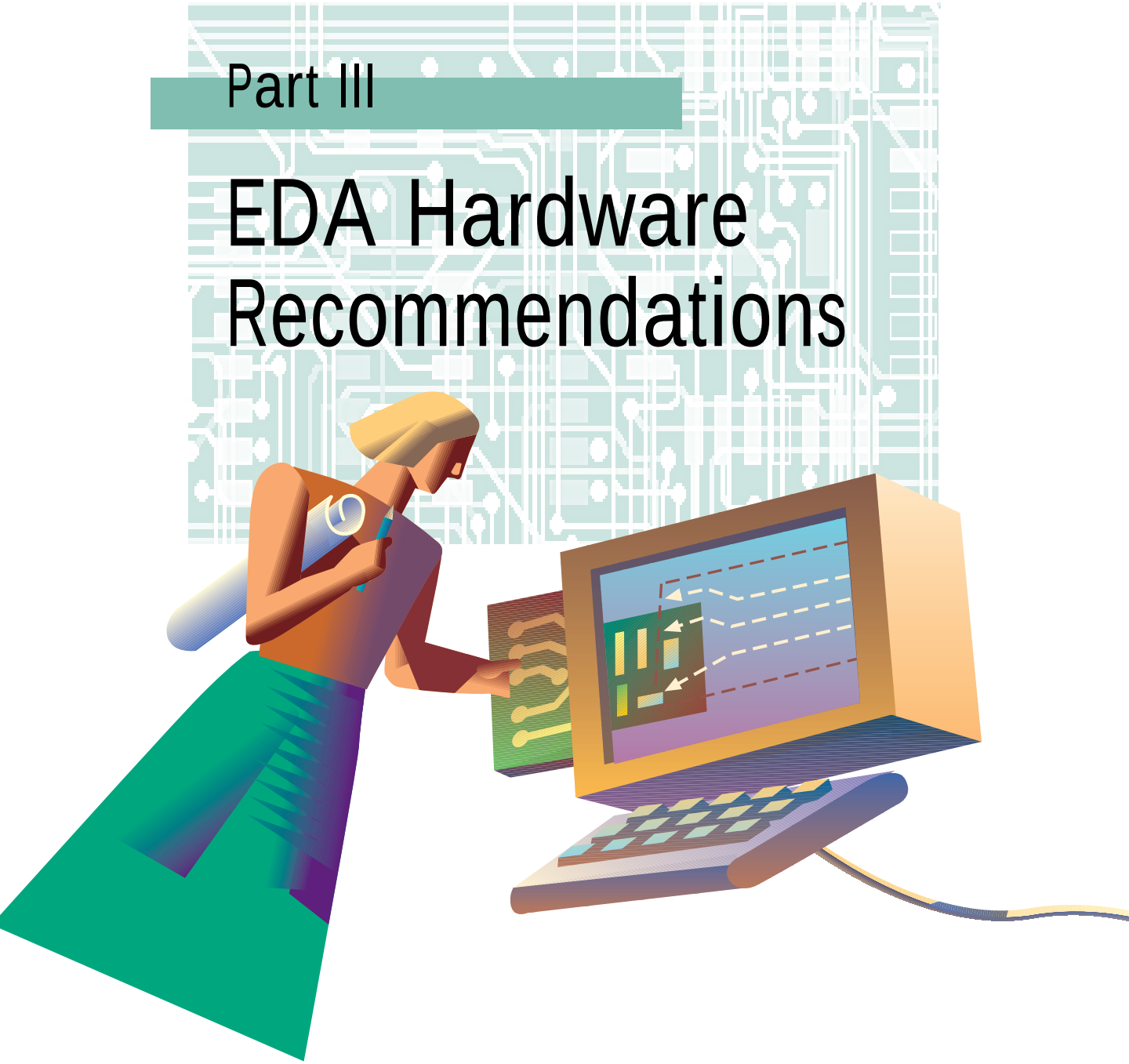
Interconnect synthesis tools: Mentor Graphics Interconnectix; Cadence Specctra Quest; Viewlogic ISIS PreVUE



The placement and routing of two DIP *(dual-inline packaged) ICs on a PCB*

Part III

EDA Hardware Recommendations



CHAPTER 11 EDA HARDWARE RECOMMENDATIONS

Note: Because hardware advances rapidly, new models will supercede those listed here. Disk and memory recommendations are only general guidelines.

Low-End UNIX or NT Workstations for EDA

These three low-end systems are suitable for the listed EDA tasks and applications. For UNIX, the B-class and it's PA-7300 CPU provides high performance relative to the price. Logical upgrades are more memory and more disk capacity.

Application areas:

- Basic IC design–polygon editing
- Basic schematic and HDL entry
- Simulation of individual circuit or HDL modules
- PCB library creation, component placement

Kayak XU NT (300/333MHz): 64–256MB RAM, 4–9GB Disk, Millenium II AGP Graphics

B132L+ (132MHz): 128–256MB RAM, 4–9GB Disk, EG Graphics

B180L (180MHz): 128–256MB RAM, 4–9GB Disk, EG Graphics

C200 (200MHz): 128–256MB RAM, 4–9GB Disk, EG Graphics

Mid-Range UNIX Workstation for EDA

A powerful desktop workstation for system and IC designers. Suitable for all of the above tasks plus the following:

- IC DRC and LPE of moderate-size designs
- IC place and route (with higher RAM sizes)
- Full circuit DRC (with higher RAM sizes)
- System-level analog and digital simulation

C200 (200MHz): 256MB–1.5GB RAM, 4–12GB Disk, EG Graphics

High-End UNIX Workstations and Servers for EDA

These high-end systems make very powerful IC design systems or compute servers. They are suitable for the following EDA tasks:

- Large IC DRC and LVS runs
- Large IC place and route
- System-level simulations involving hardware/software co-verification
- Mixed-mode and mixed-level analog/digital/HDL simulations
- Deep submicron circuit extraction and simulation
- PCB autorouting of large multi-layer circuit boards

C240 (240MHz): 512MB–1.5GB RAM, 4–12GB Disk, EG Graphics

J2240 (240MHz): 1–2 CPU, 1–4GB RAM, 9–18GB Disk

High-End UNIX Parallel Servers for EDA

Four to 16 processor systems for parallelized code or very large, memory-intensive applications. These systems make very powerful IC design compute servers. They are suitable for the following EDA tasks:

- Very large design extraction (LPE) for simulation
- Very large place and route jobs
- Very large full-circuit DRC
- Very large full-circuit LVS
- Very large full-circuit analog or digital simulations
- Deep submicron circuit process modeling, extraction and simulation

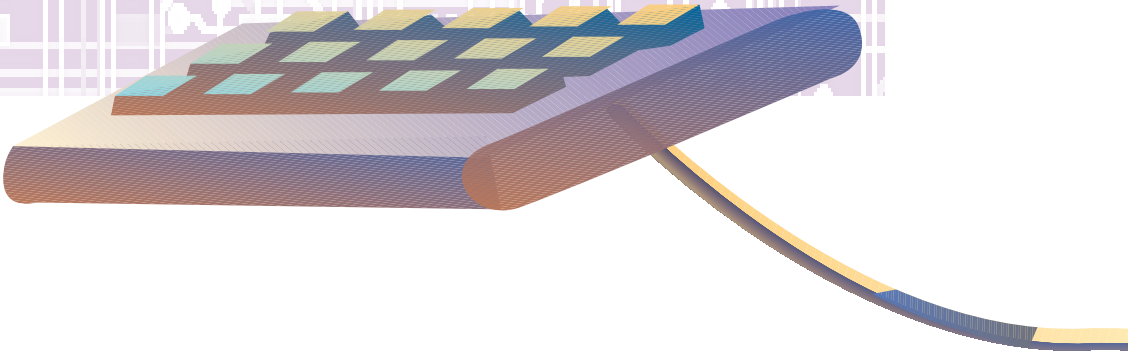
K370/K380 (200/240MHz): 4–6 CPU, 2–4GB RAM, 12–36GB Disk

K570/K580 (200/240MHz): 4–6 CPU, 1.5–8GB RAM, 12–36GB Disk

V Class Exemplar Server: 4–16 CPU, 2–16GB RAM, 18–56GB Disk

Part IV

EDA Terminology Glossary



CHAPTER 12 EDA TERMINOLOGY GLOSSARY

OK so you want to know what these terms really mean. It's not really complicated when you hear it in (kind of) plain language...

Acceleration: A technique to speed simulations by modeling logic functions in special-purpose, high-speed hardware to accelerate the verification process.

Analog simulation: The simulation of analog designs, but more specifically discrete (non-integrated) circuitry such as circuit boards.

ASIC: *Application Specific Integrated Circuit.* An IC created for a dedicated purpose, often fabricated using gate array or standard cell technologies.

ASIC libraries: A vendor-supplied library of circuit elements for use in one or more of the following processes: schematic drafting, logic simulation, timing verification and logic synthesis. Once designed, a *golden* (sometimes called the *sign-off*) simulator is usually used to validate an ASIC before fabrication.

ATPG: *Automatic Test Pattern Generation.* ATPG tools are special-purpose fault simulators that can either create test patterns, or refine existing pattern sets to improve their effectiveness. ATPG is used for manufacturing tests, not logic simulation.

Back-annotation: The process of adding a value or property to a component or net in a design that will be used for documentation or to direct a simulation or layout tool.

Behavioral HDL description: A free-form technique for writing HDL that describes the behavior of the circuit, but does not necessarily lend itself to synthesis. *See also "RTL."*

BIST: *Built-in Self-Test.* A DFT methodology where certain functions of the chip tester are moved onto the chip itself. Those functions typically include a serial interface (the TAP controller) and pattern generators or scan circuitry. The two main variations of BIST are LBIST (*logic BIST*) and MBIST (*memory BIST*).

Circuit simulation: Circuit simulation in general refers to analog design, but is more often used to designate a transistor-level simulation. Transistors are at the lowest level of basic logic gates, so IC designers perform transistor-level simulations even for digital designs.

Crosstalk analysis: The simulation of the possible interference that can occur between two adjacent signals on a PCB.

Cycle-based simulation: These simulators achieve higher performance than event-driven simulators by limiting the analysis to basic logic levels 1 and 0, not accounting for timing calculations, and only evaluating results at the end of the timing cycle. They are useful when many different scenarios must be simulated.

Deep submicron: *DSM.* Submicron (less than 1 micron) and deep sub-micron (less than .25 micron) are references to the minimum geometry size in an IC. DSM designs put unique, more stringent requirements on design tools.

Design reuse: The ability to reuse portions of existing circuitry in new designs. When design cores are sold to a customer for reuse, it is sometime referred to as intellectual property.

Design turns: Every time a portion of a design is reengineered, it must be re-simulated or reverified. The process of making a change and reverifying is called a design turn. When a design turn takes hours or days, there is a sales opportunity for high-performance technical computing solutions.

Deterministic fault simulation: The traditional, extremely time-consuming method of fault simulation where the design is simulated over and over, with each connection or *node* in the design "stuck at" a logical 1 and 0 to emulate manufacturing faults.

DFM: *Design for Manufacturability.* These tools serve the purpose of evaluating a design against the customer's own criteria for manufacturability. These criteria can include cost, heat, clearances, and mean time between failure. They can also be used to predict the product's ultimate reliability.

DFT: *Design-for-Test.* A methodology of designing ICs and PCBs where the ability to test the circuit is considered as part of the engineering process. Tools in this process use techniques such as scan, BIST and ATPG.

DRC: *Design Rule Checking.* A form of IC verification where the physical layout is compared against a set of rules to make sure that it does not violate any manufacturing constraints such as wire lengths and proximities.

EDIF: The *Electronic Design Interchange Format*, perhaps the most common netlist format for the transfer of electronic circuit data between design applications.

Embedded systems: Designs which contain cores of processors or other general-purpose components that pose unique design and verification issues.

Emulators: Special-purpose hardware that can be programmed with the functionality of a component within the design for simulation purposes, often an ASIC, FPGA or custom IC.

Event-driven simulation: A traditional form of digital logic simulation where circuit evaluation takes place when an external or internal change is pending on a pin or wire.

Fault simulation: The process of verifying that test patterns written or generated for an IC or PCB will detect most common manufacturing defects.

Formal verification: An emerging technology that allows a designer to postulate some property about his design and prove mathematically that it is true under all circumstances. Formal verification is desirable because it does not require test vectors.

FPGA: *Field Programmable Gate Array.* An IC that is made up of a structured array of logic elements configured by the user to provide a specific functional application. These ICs are programmed on special-purpose hardware, usually from a netlist generated on an EDA workstation. FPGAs are used for lower volume, lower density applications than ASICs made using gate array or standard cell methodologies.

Full-custom IC: Unlike gate arrays or standard cells where the design is fit into a matrix of existing logic elements, full-custom ICs are designed by laying down the polygons that define the junctions of transistors to create basic gates. Custom ICs are more costly to produce, but generally create faster, higher density designs.

Gate array: A type of ASIC where a customer's design is fit into an existing matrix of circuit macro functions. Gate arrays are fabricated at the ASIC vendor's foundry from the customer's specification, usually submitted as a netlist or other design database.

Golden simulator: Before an ASIC vendor will accept a design for fabrication, it usually must be verified on a logic simulator that is considered accurate and reliable by the vendor. A simulator that meets their requirements is called *golden* or a *sign-off* simulator.

Hardware modeling: Some designs contains parts that are too complex to be represented by software models in logic simulation. In these cases, the actual part is placed in a special-purpose device (the hardware modeler) that is connected to the workstation running the simulator. When the circuit being simulated wants to involve the hardware model, the input signals are sent to the part, and its outputs are read and fed back to the software simulator.

Hardware/software co-verification: Tools that facilitate the concurrent verification of the hardware and software in a system, usually containing an embedded controller.

HDL: *Hardware Design Language*. A method of writing a textual description of a circuit's behavior that becomes the specification for that circuit or module. HDL can be simulated and subsequently synthesized into gates. The most common HDLs are Verilog and VHDL.

HDL-A: A type of hardware description language that contains extensions allowing it to describe analog circuitry. This is a relatively new language, and the analog simulator must have the capability to simulate HDL-A to take advantage of this methodology.

Hierarchical design: The creation of a higher-level of abstraction, often represented by a *black box* that can be defined later (*top down design*) or first (*bottom up design*).

IC verification: An umbrella term for tools involved in the process of verifying that the layout adheres to all the constraints that make it manufacturable, meet specifications and reflect the logical design description.

IGES: *International Graphics Exchange Specification*. A standard format for storing and transmitting physical and mechanical design information, often used to exchange circuit board outlines and other mechanical features.

IP: *Intellectual Property*. A reference to the marketing of predesigned electronic cores for use in a customer's own designs. Cores can either be *soft*, such as HDL descriptions not yet synthesized, or *hard*—already deployed in silicon or as gate-level descriptions. Soft cores have the advantage that they are tool and foundry independent.

Interconnect synthesis: A methodology where the physical placement of components and wires on a circuit board or boards within a system are derived from an electrical analysis of the interconnections.

Logic simulation: A software application that simulates the operation of digital circuitry in order to perfect its operation before it is actually produced or prototyped.

LVS: *Layout vs. Schematic*. A type of IC verification tool that compares the layout of the chip to the schematic database to insure an accurate representation in silicon before it is produced.

MBIST: *Memory BIST*. A technique to put test circuitry inside the chip that is specialized for embedded memory arrays. *See "BIST."*

Microwave simulation: A high-speed version of analog simulation that verifies and analyzes operation of circuits taking into account effects that occur at microwave frequencies.

Mixed-mode simulation: A form of digital simulation that permits simultaneous co-simulation of varying modeling methods usually including but not limited to gates and HDLs. Sometimes confused with mixed-level (analog-digital) simulation.

MTBF: *Mean Time Between Failure*. A prediction (usually in hours) of the average time that a random sample of a component or system will run without failure.

Net: A wire or connection in an electronic design, either physical or logical.

Netlist: A method to represent electronic designs, usually an ASCII file that contains part lists, interconnections and other annotations used to “feed” downstream applications.

Netlister: A software application that extracts or prepares netlists for use by targeted downstream tools.

Node: A connection point in an electronic design.

Parametric extraction: The extraction of resistances and capacitances at nodes within circuitry that are inherent in an IC design for purposes of annotating circuit simulations with the physical effects of IC layout.

PCB: *Printed Circuit Board.* Also sometimes called a PWB *(Printed Wire Board).*

PLD: *Programmable Logic Device.* A method of IC design where a customer's design is fit into an off-the-shelf part. Programming the PLD usually occurs on special-purpose hardware from a design description created on an EDA workstation. PLDs differ from FPGAs in their smaller size and lower cost.

Probabilistic fault simulation: Also known as fault grading, a mathematical technique to verify production test vectors, usually used as a precursor to normal deterministic fault simulation.

RTL: *Register Transfer Level.* A style of HDL that describes circuit behavior in a structured fashion that more closely resembles logic elements. Most synthesis tools require HDL to be in RTL form before they can process it.

Scan: The process of substituting functionally equivalent circuit elements into a design that permit observation of internal test points for detection of manufacturing defects.

Schematic entry/drafting: Interactive schematic drafting to capture a diagram for both documentation and the creation of a database for downstream layout and analysis tools.

SPICE: The original circuit (analog) simulator developed at UC Berkeley that still exists in both proprietary and public domain versions.

Standard cell: A method of IC design similar to gate array, except the customer's design is applied to an array of basic gates instead of an array of more complex macro functions.

Structure: A design description composed of gates or other library elements that usually corresponds to the schematic representation of the circuit

Synthesis: The process of converting a textual design description (such as an HDL) into the gates of a targeted ASIC vendor's technology.

System design: In general, system design refers to designing and analyzing circuits containing multiple ICs or PCBs. System design can also refer to the process of creating a high-level design description that becomes the specification for implementation.

TAP controller: *Test Access Port*. The standard serial interface for BIST defined by JTAG, the *Joint Test Action Group*.

Technology: How ASIC vendors and customers refer to the fabrication process of an IC, often in terms of design size: (such as 1 micron or .25 micron and technology like CMOS, GAS or ECL).

Timing analysis: A complementary application to logic simulation where the design's ability to run at the desired speed and continue to function is verified.

Transmission line analysis: A subcategory of high-speed design tools intended to analyze the effects of electrical interference between adjacent paths.

Workflow management: Applications intended to monitor the design process and enforce certain disciplines and procedures to enhance efficiency and quality.

Vectors: Also referred to as stimulus, vectors are signals applied to a design to stimulate it for logic, fault or other simulation and test tools.

Verilog: A C-like proprietary hardware description language in wide use for describing, simulating and synthesizing digital logic.

VHDL: *VHSIC (Very High Speed Integrated Circuit—that's the "V") Hardware Description Language*. Only the Department of Defense would imbed an acronym within an acronym. A hardware description language that in its syntax can accurately describe logic, either in a structural, RTL or behavioral fashion. Used for creation, simulation and synthesis of digital designs.

About the Author

Steve Duncan is a Northwest native whose interest in electronics, and later computers, began at the age of five. Shortly after studying electronic engineering at the Oregon Institute of Technology, Steve went to work for Tektronix' Graphic Computing Systems Division in various roles for the next seven years. In 1983, Steve became employee number 45 at Mentor Graphics, hired to help start up their customer service department. He went on to become a sales application engineer in 1988, serving local Portland area customers such as Intel and Sequent Computer Systems.

Moving to Hewlett-Packard in 1996, Steve's responsibility was to help support the Mentor Graphics account team with both technical and marketing duties—a natural for his background in EDA. Today he continues on the Mentor team as a workstation specialist and member of the Independent Software Vendors (ISV) Global Consultant Team.

Steve is president of his homeowner's association in the neighborhood of Aspen Park in Southwest Beaverton. He enjoys digital photography, home theatre and is an avid electronic hobbyist. With his wife, Helene, and two children, he enjoys camping and playing in the shadow of Mount St. Helens at a private campground situated near Lake Merwin, Washington.

HEWLETT® PACKARD