An Interactive Learning Environment for VLSI Design

JONATHAN ALLEN, FELLOW, IEEE, AND CHRISTOPHER J. TERMAN

Invited Paper

The interactive learning environment (ILE) is designed to combine the traditional resources of a textbook with the "hands-on" design experiences that are vital to a real understanding of basic engineering principles. The ILE builds on the technology developed for the World Wide Web to provide a learning environment that can be easily accessed from any browser. In addition to browseable text, JAVA-based computer-aided design (CAD) tools can be accessed through interactive figures embedded in the text, where students can investigate circuit behavior under the guidance of focused tutorials.

***Keywords*—Computer-aided instruction, design automation, educational technology, electrical engineering education, integrated circuit design, interactive computing, unsupervised learning, very large scale integration (VLSI).**

I. Today's Learning Environment

Very large scale integration (VLSI) design classes have been very popular at universities for at least two decades and continue to be in high demand as students seek to exploit the rapidly expanding technological base. Today, these design techniques are taught by a combination of lectures, recitations, teaching assistant tutorial sessions, textbook reading assignments, problem sets, and laboratory exercises, culminating in a term project that is often submitted for fabrication, the resulting chip being returned later for testing. Students appreciate the increasingly central role of VLSI in our technical culture and seek to understand and appreciate the MOS VLSI design process, even if they do not intend to become professional VLSI designers. They sense the excitement of creating a design from scratch (i.e., synthesis) and of actually building something useful. Synthesis implies choice among design alternatives and, hence, exploration of the consequences of design decisions. However, today's learning environment does not tightly coordinate the available resources to easily and effectively explore a design. Textbooks are usually not designed with a particular computer-aided design (CAD) tool environment in mind, and CAD tools are usually not designed with learning in mind, but instead, the tools are configured for intensive production use by industrial designers. In an educational environment, these tools are often hard to load, configure, train for, and use, although they are undoubtedly powerful in the hands of a skilled designer with strong system maintenance support.

The context of today's learning environment, however, is rapidly changing. Today's personal computers are powerful and capable of performing complex algorithms with rapid response, even in laptop configurations. Learning environments must also accommodate an expanding audience. In addition to the traditional institutional setting that caters to undergraduates and graduate students at the beginning of their careers, there is also increased interest in distance learning and life-long learning. So there is a diverse set of learners with varying backgrounds and educational resources at their disposal, thus generating an increasing demand for learning environments that are suitable for self-study. A broad palette of resources needs to be provided for the individual student, so that each can adapt these resources to his or her personal learning style. That is, there is increasing need for a coordinated set of learning resources from which a solid understanding can be built. It is probably not possible to devise a single "best" learning environment for all students, but it is possible to create a tightly coupled set of interactive learning components that can serve today's diverse student requirements in an effective way. With this perspective in mind, we turn first to an elaboration of our goals, then a characterization of an "ideal" learning environment, and finally a description of our functional implementation of an interactive learning environment (ILE) that seeks to approximate these ideals.

Our goal is to help students learn goal-oriented design of MOS digital circuits using realistic technologies.

Manuscript received February 2, 1999; revised July 13, 1999.

J. Allen is with the MIT Research Laboratory of Electronics, Cambridge, MA 02139 USA (e-mail: jallen@mit.edu).

C. J. Terman is with the MIT Laboratory for Computer Science, Cambridge, MA 02139 USA (e-mail: cjt@mit.edu).

Publisher Item Identifier S 0018-9219(00)00204-8.

- Our emphasis is on learning rather than teaching. We believe in learning by doing. The student can best learn to create these circuits by actually designing them personally, as well as exploring all aspects of existing designs.
- Our emphasis is on design, or synthesis, although we realize that, to create the best designs, it is essential to analyze new and existing designs comprehensively. There is an interactive tension between analysis and synthesis, which is required, in part, due to the lack of perfect modularity; e.g., the interaction between connected modules that may require adjustment for both correctness and performance.
- We emphasize goal-oriented design, where tradeoffs are examined over a set of designs that range over varying speed, size, and power dissipation.
- We emphasize the centrality of the circuit representation as key to design. The circuit is the highest level of IC design abstraction that still directly represents those physical parameters responsible for performance. (The next level higher is the logic level.) Although we also focus on layout as the necessary specification to interface with fabrication processes, we always extract the circuit representation from layout in order to assess both logical correctness and circuit performance.
- We are concerned with a design environment for digital MOS circuits. These are large-swing nonlinear circuits, where the focus is on the digital logic abstraction relation to the circuit, including voltage levels and transitions, and where the possibility of noise perturbations (through both resistive and capacitive coupling) must be considered. As such, although circuit equations can be written for these circuits, they cannot be solved in closed form. So, detailed circuit understanding requires simulation to accurately derive the relevant voltages and currents.
- Our design environment utilizes realistic device models, including parasitics, so that meaningful quantitative simulation results can be computed. Accurate models for interconnect are also derived from the technology.

II. The Ideal Learning Environment

In order to realize our goals, we first conceptualize the nature of a design environment that would be ideally suited for providing the sorts of learning experiences we have in mind. There are several components of this ideal environment that we think should be tightly integrated in a single environment (see Fig. 1).

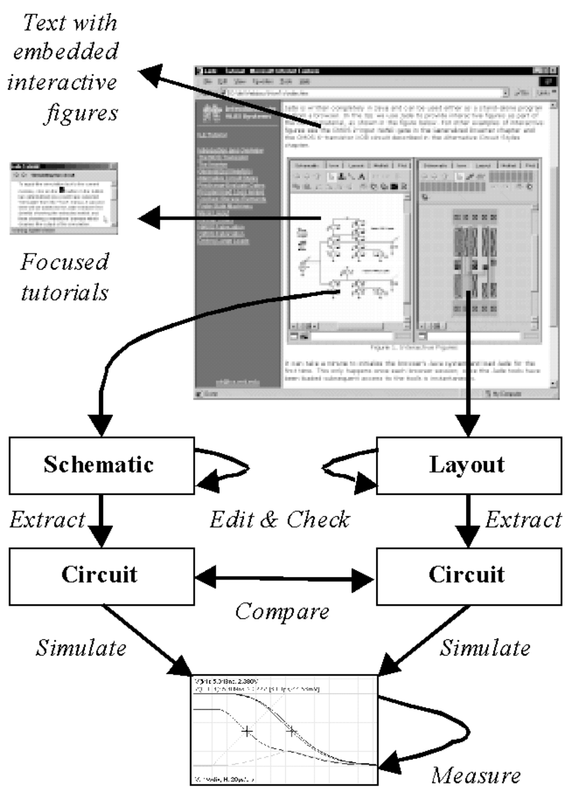


Fig. 1. ILE block diagram.

A. Texts

Traditionally, many students have learned IC design from a text. Good design texts provide background, motivation, organization of a substantial volume of material, and exploration of many facets of IC design at varying levels of detail. They provide the theory to support the intuition and experience that comes from exploring many designs. Texts are inherently linear, hence they are not well suited to showing multiple simultaneous relationships, although they are well suited to providing a taxonomy of material. Texts have their own hierarchy and modularity, and they are usually organized by the type of circuit or style (e.g., generalized inverters driving a capacitive load), so that a comprehensive circuit repertoire is exhibited. Good texts highlight general principles, provide insightful explanations of circuit action and features, and reveal intercircuit relationships. They are less well suited for the exploration of quantitative circuit detail and how circuit parameters and topology interact to provide the resultant performance. Thus, while texts have both advantages and disadvantages, they are a useful component in an overall ideal learning environment.

B. Toolkit

In order to explore each circuit or layout (the two representations that are central to IC design), an ideal learning environment must include an interactive toolkit of programs. Our vision for the appropriate tools has been inspired by [1]–[4]. The toolkit includes the capability to capture and edit schematic and layout representations, to extract circuits from schematics and layouts, and to simulate circuits. The toolkit also includes electrical and layout well-formedness checking (e.g., design rule checking), and it can perform equivalence checking between schematic and layout representations. Hierarchy, as used in both schematics and layout, is manipulated consistently by all programs of the toolkit. By using the tools, existing schematics and layouts can be modified (in any respect) and new designs can be created. Thus circuit variations can be studied and comparisons made. All designs, including those provided by the environment and those created by the user, are savable in a group of libraries (see below). We think it is important to make a direct, seamless transition from the circuit schematics and layouts in the text to what the student experiences in the toolkit environment. This can be easily accomplished if one uses the toolkit to present the figures that appear in the text; this has the added benefit that the figures can be interactive. Usually, the toolkit focuses on intracircuit properties and relationships, rather than the greater emphasis on intercircuit relationships revealed in the text component.

Once a desired circuit has been created (either from an initial circuit schematic or extracted from a layout) then its temporal performance needs to be simulated and displayed in a way that makes it easy to browse and measure. Ideally, it should be possible to display any current or voltage of the circuit, as well as to display any subset of them simultaneously, by inserting appropriate voltage probes and current meters in the schematic editor. From the display, it must be possible to check on circuit constraints, such as logic levels and pullup/pulldown ratios. It is also highly useful to derive synthetic measurements such as device drive $V_{GS} - V_{TH}$, which help to give an idea of the device's contribution to circuit performance. In this way, the learning environment toolkit provides a more extensive understanding of the circuit than that provided by the simulator alone. The probes provide a selective level of circuit performance detail and allow the learner to examine the circuit behavior in any desired way. The probe waveforms are also essential in order to train an intuition for circuit dynamics, e.g., for charging and discharging of capacitive nodes.

The ideal design environment must provide for the creation of circuit hierarchy, so that modules can be used repetitively in an overall design, and so that complex circuits can be composed for richer functionality from smaller modules that can be characterized on their own. Of course, the price of modularity and hierarchy is often performance, due to the nature of module interfaces, and the ideal environment will facilitate study of this effect.

In order to compose circuits hierarchically and to promote reuse, modules are organized into libraries. Students can share libraries that contain primitive devices, models, sources, probes, example circuits, and useful building blocks, e.g., standard cells and memories. Each module has one or more aspects drawn from an extensible set (e.g., icon, schematic, layout, netlist, and waveform). Per-module property/value pairs permit the construction of parameterized or generated modules.

C. *Tutorials*

We have seen above that every circuit and layout of the ideal text can be instantly edited and simulated, and its waveforms displayed. This provides an extraordinarily cohesive environment for learning, but there is so much that can be studied in even the simplest circuit that guidance in the form of focused tutorials is necessary to steer the student, step by step, to those aspects of the circuit that are especially important for interactive exploration. For this reason, the ideal learning environment should provide minitutorials that point out aspects of circuit behavior and suggest ways to modify or explore it further. In this way, the student is always directed to the study of important effects but still has the capability to study any aspect of the circuit.

D. *Ease of Access*

The ideal learning environment should be available anywhere and anytime. It must also allow for easy sharing of designs. It must not present any installation and configuration hurdles to the user, so that his or her focus and energy is centered on the design techniques to be learned, rather than complex system overhead.

III. The ILE

Given our goals and our sense of an ideal learning environment for VLSI design, we now turn to a characterization of our implementation of the ILE for VLSI design.

The ILE is an intimate fusion, or tight coupling, of several components:

- text, illustrations, toolkit, libraries, focused tutorials, video segments showing talking heads, animation, and other material (and possibly others, as the technology allows);
- the toolkit (which we call JADE: JAVA Design Environment) provides for the design of schematic and layout representations, their extraction to circuits, simulation (including comparison), and measurement.

Usually these components are loosely juxtaposed in diverse environments and media, e.g., paper print, CAD systems, and image media including stills and video. Traditional ways of packaging the components (books, computer software, and video tapes) can be cumbersome and not very portable. In the ILE, all components are represented digitally in a unified environment, and any component is easily accessed at any time. We first discuss the individual components and then how they may be used in the ILE.

The text component of the ILE implementation is based on our approach to teaching a semester-length course in VLSI design at the Massachusetts Institute of Technology (MIT). The course presumes some background in basic logic for digital design, as well as the device physics of PN junctions and the MOS capacitor, but these topics are covered at an introductory level in the text. Students who want more background in these areas can consult the references and other links provided in the text.

The text is meant to provide motivation for schematic designs, rigorous explanations of their general modes of behavior (without numeric values for node voltages and loop currents), and a development of general principles for broad classes of MOS digital circuits and systems. It provides the overall structure of the course and is the backbone component of the ILE since it offers a rich multimedia structured exposition of the entire subject.

There are three major sections of the textual ILE component. After a brief description of the MOSFET, its principles of operation, and its current–voltage relationships, the text builds a complete schematic repertoire for combinational and sequential schematics. Once this array of schematic styles (i.e., families) is provided, the text presents technology mapping of circuits, in order to mask specifications via process fabrication families, and the corresponding layout. Finally, the text considers performance issues (e.g., high-speed drivers) and special circuits, such as I/O pads.

As we have noted, much of the text develops a comprehensive repertoire of both combinational and sequential schematics. After motivating each of these schematics, the text argues for the topological structure of the schematic, in terms of its ability to deliver appropriate digital outputs for selected inputs. The reader should think of these schematics, which are presented in JADE windows through much of the text, as presentations of abstract circuit families, since they must be complemented by parameters for all devices, input source, and output load circuit fragments, a specification of the technology to be used, and the desired JADE analysis, before they can be submitted to JADE for simulation. That is, the text-based schematics contain the kernel idea of a computational module, and the accompanying text generalizes over all circuits with the same schematic kernel. The text seeks to provide a basic understanding of this schematic as representing a family, or class, of circuits, all of which share the same basic functional properties. The text is, thus, a good place to study a family's basic properties, usually without regard to detailed numerical performance issues.

The idea of family is further utilized in the text component of the ILE. Thus the text introduces general discussions of NMOS and CMOS technologies as families of technologies, whose members share many common features. However, one member from a technology family must be specified to JADE before simulation is possible. The modularity that is provided by recognizing these families with common properties is of great value in understanding the basic ideas in MOS circuits and in readily switching between members of the family. For example, it is simple to switch process specifications within a technology family in order to investigate changes in performance due to modifications in technology.

Abstract families for source input circuits and output load circuits are also provided. Thus, these can readily provide libraries with useful circuit fragments that can be easily used, along with an element of the schematic family of the kernel, to compose a complete circuit with appropriate parameterization suitable for JADE analysis.

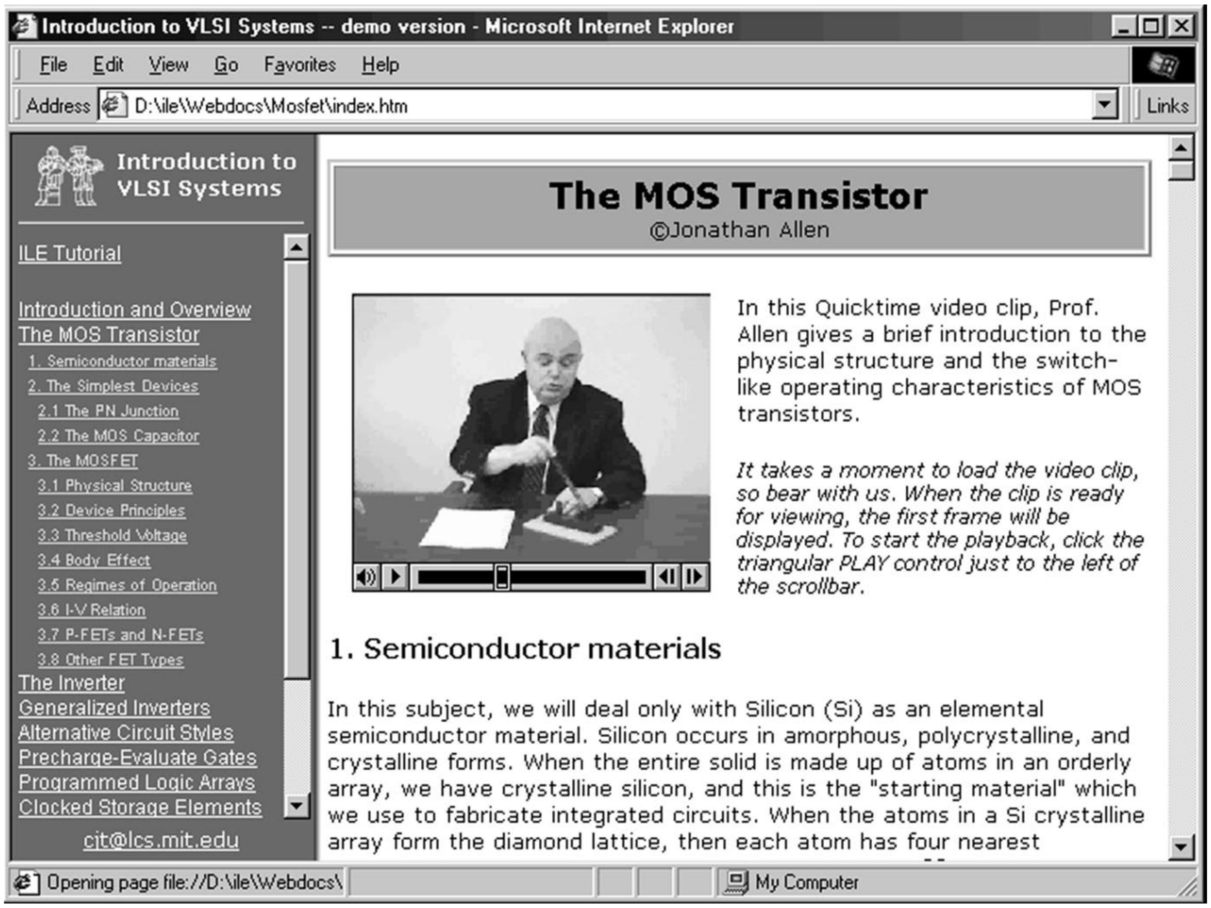
Families, or abstract classes, thus provide the text with the means to discuss general behavior in an insightful but nonnumeric way. This is a great help when appreciating the schematic function before one is ready to explore detailed circuit behavior using JADE. The reader can focus on the important ideas presented in the kernel schematic before composing a complete circuit for JADE analysis. Once the key ideas of the schematic family are assimilated, in order to enable further numerical analysis: the input and output circuits must be added to yield a topologically complete schematic; all devices must be parameterized; a technology must be picked from the technology families; and a desired simulation task must be specified (e.g., transient analysis). All of these are composed in a single JADE window, which is ready for the initiation of analysis. Note how the family abstract concept preserves the modular aspects of circuit design, and it provides the proper role for the text component to deal with the generalities common to all members of a family. This view defines the role of text within the ILE and leaves the detailed numeric study of circuit performance to the JADE environment, which is equipped to accept the elements of families as inputs needed to study specific circuits. Many of these complete circuits are provided in sample libraries, along with minitutorials, in order to guide the reader in discovering interesting circuit performance.

The text has been implemented in hypertext form in order to provide easy integration with other components of the ILE. Fig. 2 shows how the text is displayed with a hierarchical table of contents shown on the left side. This arrangement facilitates jumping to another section of the text without the need to go to a special section that shows the table of contents or index.

Another component of the ILE provides video sequences. These are used to show "talking head" overviews (Fig. 2) of each chapter of the text, which are brief introductions to the concepts introduced in the chapter. Video sequences are also used, sometimes within the "talking head" presentations, to animate device and circuit function. Thus the creation of an inverted channel between source and drain, and how it tapers with increasing drain-to-source voltage, is presented with a voice monologue explaining the process. The use of talking heads personalizes the presentation of material in the ILE and gives the student some sense of the instructor's style. The text also includes pictures of early devices and circuits in order to provide some sense of historical context. In addition, the text component provides the derivation of circuit equations and an explanation of the regions of operation for MOSFET's.

It is important to note that the text is organized in a bottom-up way. We start with the individual MOS device, then describe inverters, and continue to build the schematic repertoire until we are ready to describe how they are mapped to layout masks, which in turn are interpreted by fabrication processes to produce the final IC chip. In this way, the student has a firm understanding of physical circuit action before higher level abstractions are introduced. The text repeatedly emphasizes the need to quickly translate between different levels of circuit representation.

We emphasize that schematics and layout, together with their mappings to circuits, are special in our learning environment. For this reason, a key feature of the ILE is how these two representations are introduced in the text. In a conventional text, we would expect to see a drawing of a circuit schematic accompanying the discussion of that circuit. However, in the ILE, the drawing is replaced by an embedded window of the JADE toolkit that shows the schematic of in-


Introduction to VLSI Systems -- demo version - Microsoft Internet Explorer
File
Edit
View
Go
Favorites
Help
Address
D:\ile\Webdocs\Mosfet\index.htm
Links
Opening page file://D:\ile\Webdocs\
My Computer

Introduction to VLSI Systems

ILE Tutorial

Introduction and Overview
The MOS Transistor
1. Semiconductor materials
2. The Simplest Devices
2.1 The PN Junction
2.2 The MOS Capacitor
3. The MOSFET
3.1 Physical Structure
3.2 Device Principles
3.3 Threshold Voltage
3.4 Body Effect
3.5 Regimes of Operation
3.6 I-V Relation
3.7 P-FETs and N-FETs
3.8 Other FET Types
The Inverter
Generalized Inverters
Alternative Circuit Styles
Precharge-Evaluate Gates
Programmed Logic Arrays
Clocked Storage Elements

cjt@lcs.mit.edu

The MOS Transistor
©Jonathan Allen

In this Quicktime video clip, Prof. Allen gives a brief introduction to the physical structure and the switch-like operating characteristics of MOS transistors.

It takes a moment to load the video clip, so bear with us. When the clip is ready for viewing, the first frame will be displayed. To start the playback, click the triangular PLAY control just to the left of the scrollbar.

1. Semiconductor materials

In this subject, we will deal only with Silicon (Si) as an elemental semiconductor material. Silicon occurs in amorphous, polycrystalline, and crystalline forms. When the entire solid is made up of atoms in an orderly array, we have crystalline silicon, and this is the "starting material" which we use to fabricate integrated circuits. When the atoms in a Si crystalline array form the diamond lattice, then each atom has four nearest

Fig. 2. Browsing an ILE document with contents sidebar, video clip, and text.

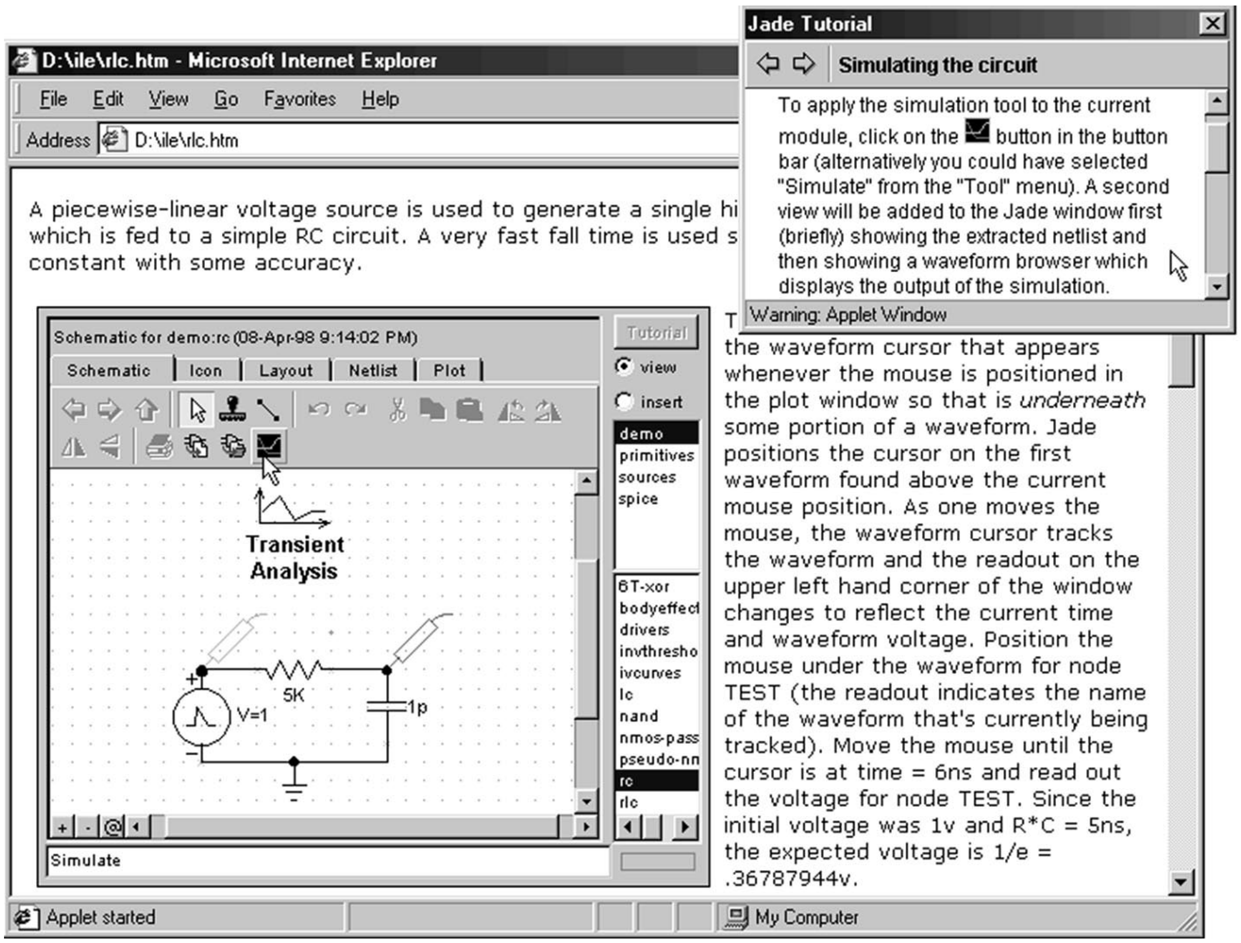
terest (Fig. 3). This means the student can instantly edit the augmented schematic, extract its SPICE netlist, and simulate it without leaving the text component. Thus the toolkit and text are very tightly coupled components of the ILE, and the schematic is already drawn within the text in editable form for further exploration. In this way, the ability of the student to discover many aspects of the circuit behavior is strongly enhanced. In order to guide this exploration, focused tutorials associated with each schematic are presented to guide the student to interesting aspects of the circuit behavior.

Layout is treated in a similar way. When a layout view is presented in the text, it is not just a static drawing, but an embedded window into the JADE toolkit environment where the layout can be edited, extracted to circuit netlist form, and simulated. For either schematics or layout, the associated simulated circuit can be augmented with voltage probes and current meters at any node or branch. Precise measurements can be made, thus facilitating not only basic understanding of the circuit action, but also important aspects of its performance. The user of the ILE can sense the schematics and layouts of the text "coming to life" by means of the toolkit component resources.

The text component presents a series of topics, for each of which the other components can add a complementary dimension. The user of the ILE can select a tightly coupled subset of component resources to associate with any given text topic. Thus the JADE embedded windows used as text figures and the "talking heads" used as text chapter overviews are examples of how the ILE provides closely bound component resources associated with the textual topic themes. It is important to note that the ILE components have an extent or depth that allows the student to pursue aspects of topics comprehensively. Thus, for example, the student can explore a circuit's behavior by: using the schematic editing, extraction, simulation, and measurement capabilities of the JADE toolkit; following suggestions in the text component; using a focused tutorial; or following one's own inclination.

Because of the tight coupling, it is possible to selectively bind in adavnce certain aspects of some components to the topic of interest. Thus the capture of a specific schematic, the associated textual discussion, and the set of minitutorials relevant to the investigation of a circuit that can be bound early, just as a figure of a circuit schematic is bound to the part of the text that discusses that circuit in a conventional text.

Hierarchy is manipulated consistently by all programs of the toolkit, and hence there are correspondence zones between corresponding parts of different representations of the same fundamental entity (circuit). Note that a fully specified circuit includes its inputs too, and hence (in a dependence sense) all voltage and current waveforms for that particular circuit configuration.


D:\ile\rlc.htm - Microsoft Internet Explorer

Address D:\ile\rlc.htm

A piecewise-linear voltage source is used to generate a single hi
which is fed to a simple RC circuit. A very fast fall time is used s
constant with some accuracy.

Jade Tutorial

Simulating the circuit

To apply the simulation tool to the current module, click on the button in the button bar (alternatively you could have selected "Simulate" from the "Tool" menu). A second view will be added to the Jade window first (briefly) showing the extracted netlist and then showing a waveform browser which displays the output of the simulation.

Warning: Applet Window

Schematic for demo:rc (08-Apr-98 9:14:02 PM)

T
the waveform cursor that appears whenever the mouse is positioned in the plot window so that is *underneath* some portion of a waveform. Jade positions the cursor on the first waveform found above the current mouse position. As one moves the mouse, the waveform cursor tracks the waveform and the readout on the upper left hand corner of the window changes to reflect the current time and waveform voltage. Position the mouse under the waveform for node TEST (the readout indicates the name of the waveform that's currently being tracked). Move the mouse until the cursor is at time = 6ns and read out the voltage for node TEST. Since the initial voltage was 1v and R*C = 5ns, the expected voltage is 1/e = .36787944v.

Simulate

Applet started

My Computer

Fig. 3. ILE text with embedded interactive figure; active tutorial window is open at upper right.

IV. Examples

We have found that the ability to study circuit behavior easily, in detail, and with realistic accuracy permits a vastly deeper understanding of circuits, both for instructors and students, than can be provided by a conventional text with static figures. Even the simplest circuit provides endless opportunity to gain new insight into circuit function, thus raising questions and furnishing the means to acquire a new perspective on basic circuit function. The toolkit provides a virtual laboratory that can be used anywhere and anytime.

In this section, we provide a sampling of how we used the ILE to study circuits. For each circuit, we raise important issues (which would be suggested by the focused tutorial components) and then explore the circuit to gain understanding and insight. We summarize what has been learned through such exploration and comment on how the exploration process itself was used. The toolkit provides professional-level resources for studying circuits, so the student can be confident that the results obtained are accurate, realistic (in terms of available processes), and extendable to large project-sized systems.

A. Two-Input NAND *Gate*

Students are often exposed to simple explanations of circuit action, which mask issues of performance, so there is often a tendency to move on to "more interesting" circuits. Here, we see that "slowing down" to appreciate circuit behavior fully can bring rich rewards that will provide a more mature context for the investigation of more complicated circuits.

Our first example is a simple four-transistor circuit where only one of the transistors can exhibit body effect. Its behavior is often described in VLSI texts in terms of the AND effect of series FET's, and the OR effect of parallel FET's. The reader is readily convinced that the NAND gate produces the correct static output levels and that it can be easily extended to more inputs with the same principles of operation. Beginning students who are used to turning devices on and off by controlling the gate voltage are often surprised by the upper pulldown NFET, since, when its input is high, but the lower input is low, it will be turned off by the rising level of its source terminal, which decreases V_{GS} until it equals the (body-effected) device threshold. The NAND gate also has interesting effects that significantly impact delay in ways that are contrary to intuition and conventional wisdom. The student may not be able to argue simple cause and effect due to the interplay of many factors and the transient nature of effects that manifest themselves only within certain regions of operation.

Here, we study the effect of input arrival time on circuit delay, as shown in Figs. 4–6, where we simulate two identical NAND circuits with their inputs reversed. Hence, for the top gate, the top input transitions first; whereas for the bottom

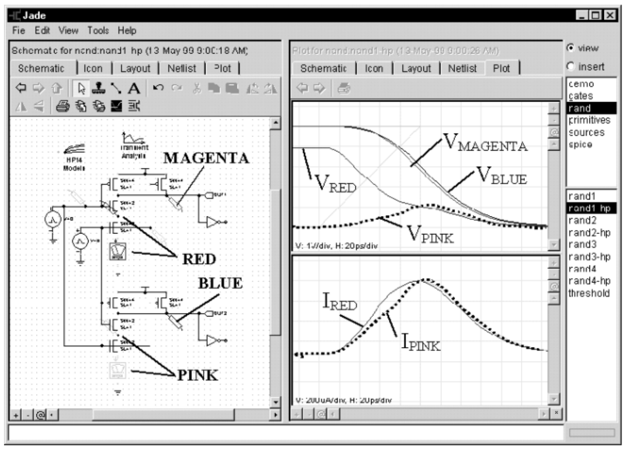


Fig. 4. NAND circuit for comparing output delays using different inputs.

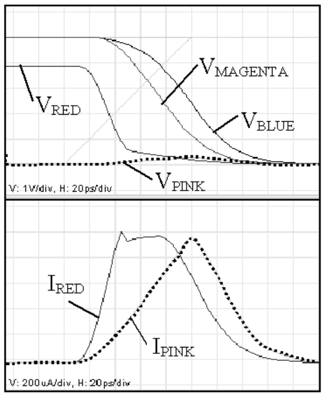


Fig. 5. NAND with wide bottom pulldown.

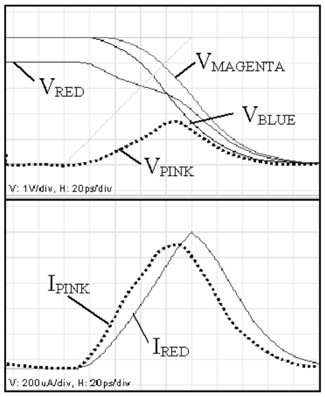


Fig. 6. NAND with wide top pulldown.

gate, the bottom input transitions first. We focus on the falling output transition.

If all NFET pulldown devices are of the same (minimal) size (Fig. 4), delays for the two circuits are approximately equal. In the top gate, although the intermediate pulldown node has been charged up, it is quickly pulled down at the onset of the late bottom input transition. The output for both circuits is dominated by discharge of the larger load capacitance. The saturation current of the devices limits the speed of discharge of the load.

The widening of pulldowns close to the supply "rails" is often advocated in order to reduce the body effect on those devices further from the rail. However, in Fig. 5, with the selected fabrication process, widening the bottom pulldown does not help the delay, even when its input transition occurs first. The waveforms show the rapid discharge of the intermediate pulldown node, thus leading to strong drive (little body effect) on the upper pulldown device, which in turn makes the upper circuit delay quite small, contrary to conventional understanding.

Finally, widening the upper pulldown (Fig. 6) couples the output node and the intermediate pulldown node strongly, thus retaining considerable body effect in the upper pulldown of the upper circuit ($V_{\rm RED} > V_{\rm PINK}$). In this case, the bottom circuit is clearly quicker, where the bottom input transition arrives first.

From these three experiments, one can conclude that "whichever circuit has its wide pulldown switch last is fastest." This is an interesting result, not usually encountered in texts. Is this finding due to the relatively high mobility and low body effect (γ) of the process used in the simulation? (Processes can be easily changed in the toolkit environment.) Would the result change if the inputs were driven from inverters rather than piecewise-linear segments specified in the SPICE input?

From this example, we see that even very simple circuits have surprising behavior due to the interaction of many factors. It is difficult for even experienced designers to predict waveforms that may result from the relative timing of a number of these different effects.

B. Domino Logic

The introduction of timing disciplines to "combinational" circuits brings in many new factors, including isolated sets of nodes that will try to redistribute their collective charge. In this example, we use the toolkit to examine this "charge sharing" and to explore the means to combat its unwanted effects.

In order to reduce the delay through PFET pullup chains, domino circuits have been introduced that use a precharge/evaluate timing discipline. These are coupled with a single PFET precharge pullup, a single NFET evaluate pulldown, and an output inverter to avoid race conditions. Here, we investigate the charge sharing problem during the evaluate phase for two versions of a domino circuit. Both versions introduce an extra load capacitance C_X on the intermediate node below the precharge node in order to study the effects of charge sharing.

In the first domino circuit (Fig. 7), charge may move from the precharge node down the pulldown chain during the evaluate phase. Since the precharge node is otherwise isolated during the evaluate phase, enough charge may flow down the pulldown chain to inadvertently lower the voltage of the precharge node enough so that the inverter output switches when, in fact, we expected it to remain low. One way to deal with this problem is to widen the inverter devices in order to increase the capacitance of the precharge node relative to the capacitance of the nodes in the pulldown chain, at some possible cost in delay. In Fig. 7 we illustrate the same effect

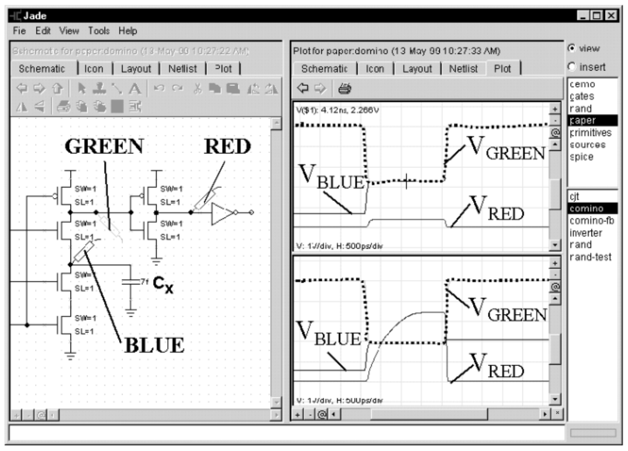


Fig. 7. Charge sharing in a domino NAND gate.

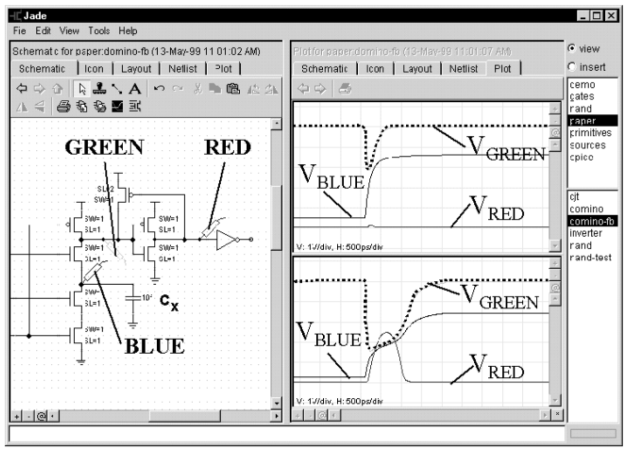


Fig. 8. Charge sharing with feedback keeper.

by manipulating the extra load C_X. Note that when C_X is 7 fF (top plot), the inverter output voltage V_{RED} rises a little (less than V_{TH}), while the precharge node voltage V_{GREEN} drops to meet the intermediate node voltage V_{BLUE} below it in the pulldown chain. However, when C_X is 10 fF or larger (bottom plot), the inverter output switches during the evaluate phase, giving rise to a "glitch" that will propagate through the domino chain.

In the second domino circuit, we explore another way to combat the inadvertent switching of the output inverter due to charge sharing from the precharge node. In this circuit, a feedback PFET "keeper" device is introduced around the output inverter, as shown in Fig. 8. When the precharge node voltage is high, the inverter output voltage is low, thus holding the keeper device on, which bleeds charge from the V_{DD} rail into the precharge node, and replenishing charge that has been shared down the NFET pulldown chain. (Note that the keeper also helps to avoid capacitive noise coupling to the precharge node during the evaluate phase by providing a resistive path to the V_{DD} rail.) Now, as with the previous domino version, we can increase C_X in order to draw charge off the precharge node during the evaluate phase. Even when C_X is increased to 10 fF (top plot), we see in Fig. 8 that the precharge node drops in voltage, but there is only a small glitch (less than V_{TH}) on the output. C_X has to rise to over 27 fF (bottom plot) before the inverter output voltage switches to a high value for the duration of the evaluate phase—which will propagate an error through the domino chain—illustrating the value of the keeper. By experimenting with the size of the output inverter devices, the size of the keeper (which must be weak enough to cut off the inverter pulldown when the inverter should correctly switch to a high voltage), the size of the precharge device, and possible opportunities for minimizing the capacitance of nodes in the NFET pulldown chain, a high-performance domino circuit can be obtained that will not misbehave due to charge sharing.

C. Sense Amplifier

Most CMOS circuits are designed around the switch-like behavior of the MOSFET and, after a few weeks of analyzing such circuits, students have a good feel for how they work. When confronted with a circuit that exploits a different property of MOSFET's (in this case, the relationship between currents and voltages in a saturated device), many students are at a loss to explain how the circuit works, since more is involved than the devices simply being "on" or "off." The purpose of this example is not only to teach how sense amps work, but also to show how to ferret out a reasonable explanation in a methodical fashion.

An interesting design problem is to construct a sense amplifier, which is a circuit used to rapidly detect very small changes in the voltage of memory bit lines. While CMOS gates exhibit very high voltage gain at their switching threshold, changes in the switching threshold due to variations of temperature, power supply voltages, manufacturing parameters, etc., make it exceedingly difficult to reliably bias the bit line voltage into the high-gain region of the gate input. So, many memories use a variation of the double-ended sense amplifier shown in Fig. 9. As shown in the accompanying waveform plot, a small dip in the voltage of the BIT input quickly results in a large drop in the voltage of the DATA output. Two quick experiments with the circuit show how it works.

Inserting ammeters into each leg of the pullup and another ammeter in the common pulldown path (Fig. 10) lets us make some interesting observations about current flows. Notice that the current I_{CYAN} flowing into the bottom NFET is relatively constant: that NFET is configured as a current source. Since the total current through the current source is fixed, the current in one leg of the pullup must "mirror" the current in the other leg, i.e., when the current in one leg drops, the current in the other leg must increase by the same amount (and vice versa). A small drop in the BIT voltage results in a small drop in the current through the right leg of the pullup, which is then mirrored by a small increase in the current of the other leg. But why does this lead to such a dramatic change in the voltage of the DATA output? The key lies in the behavior of the PFET in the left pullup chain, through which the increased current must flow. Let us characterize the behavior of this PFET using another experiment shown in Fig. 11.

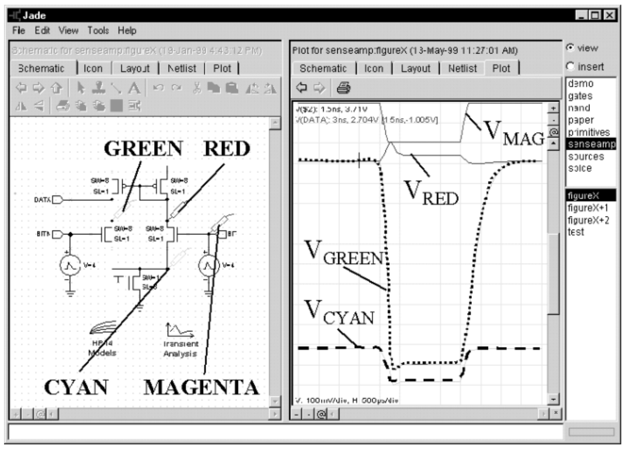


Fig. 9. Double-ended sense amp (voltages).

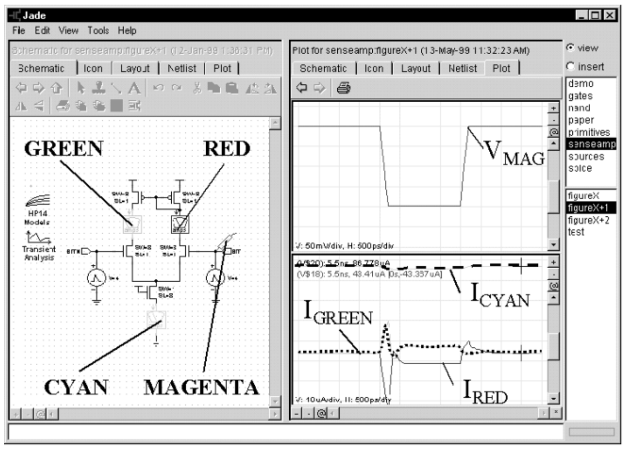


Fig. 10. Double-ended sense amp (currents).

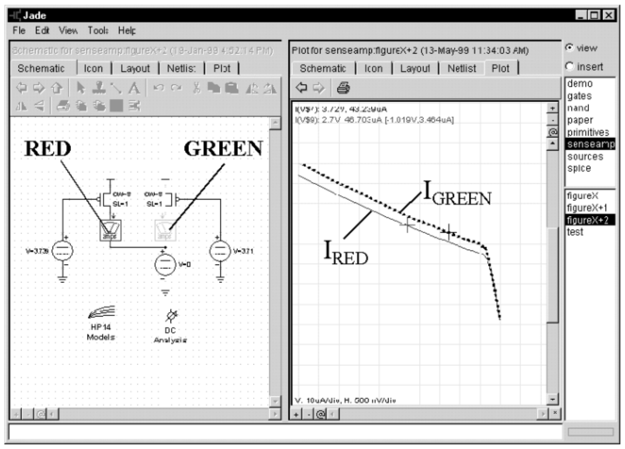


Fig. 11. I_{DS} curves for PFET pullup in saturation.

Noting that the source and gate voltages of the PFET are relatively constant (in fact, the right PFET is configured as a voltage source), we can perform a dc sweep to produce the I_{DS} versus V_{DS} diagram shown in the accompanying waveform plot. We can see that the PFET is barely on and, that for most values of V_{DS}, the device is in saturation. This means to get a small increase in I_{DS}, we must have a dramatic decrease in V_{DS}.

We can now put these two observations together to explain how the sense amplifier works: a small change in the bit line voltage produces a small drop in the current through one leg of the current mirror which must be offset by a small increase in the current through the other leg. Since the left PFET is in saturation, even a small increase in current requires a large decrease in V_{DS} across the PFET and, hence, a large drop in the output voltage.

This example illustrates the value of conducting circuit experiments quickly using the JADE tools in the interactive learning environment. Students can be encouraged to examine the circuit *in situ* (e.g., by adding voltage probes and ammeters) to determine how each piece is working. They can also build separate "throw-away" test benches to focus on a particular behavior or region of operation. Easy experimentation is a key to gaining a thorough understanding of what is really going on.

D. Mask Layout

Mask layout may seem to be on its way out as a legitimate topic in a design course. Many designers do not directly manipulate masks (or for that matter transistors)—they use automated synthesis targeting cells from vendor-supplied libraries, which are then automatically placed and routed. Can layout issues be made to completely disappear behind some suitable layer of abstraction? Arguing by analogy, most computer software courses no longer discuss assembly language or bit-level data representations since those details can be successfully hidden from most users. Why not take the same approach to the low-level physical details of MOS circuit design? In fact, many designers try to do this later only to discover to their chagrin that layout "details" such as interconnect have a dramatic performance or cost impact on their design. We include layout and extraction (with parasitics) as part of the ILE to provide some exposure to these issues.

Constructing a mask layout for a circuit is often tedious and fussy, especially when checking the geometry information for compliance with design rules is performed separately from the actual editing process. Drawing on experience with the Magic layout tools [3], the JADE layout tools (see Fig. 12) have several important features that make the process more palatable (even fun).

Design-rule checking is performed incrementally in the background in "real-time" [4], thus providing quick feedback about the correctness of most editing operations. The immediate visual feedback makes editing the layout a very forgiving process, since the offending geometry is flagged and can be fixed before the error, is built into the design in ways that would be hard to fix. It is usually a simple matter to nudge the misplaced mask layer in an appropriate direction.

Pseudolayers automate the tedium of laying out multilayer constructs such as contacts or diffusions. Since the detailed composition of mask layers is performed automatically by the system, they never get out of registration.

Mask information is viewed as "paint" rather than a collection of polygons. Most editing operations involve adding or removing "paint" on a particular mask layer—the system automatically maintains the underlying geometry database. So,

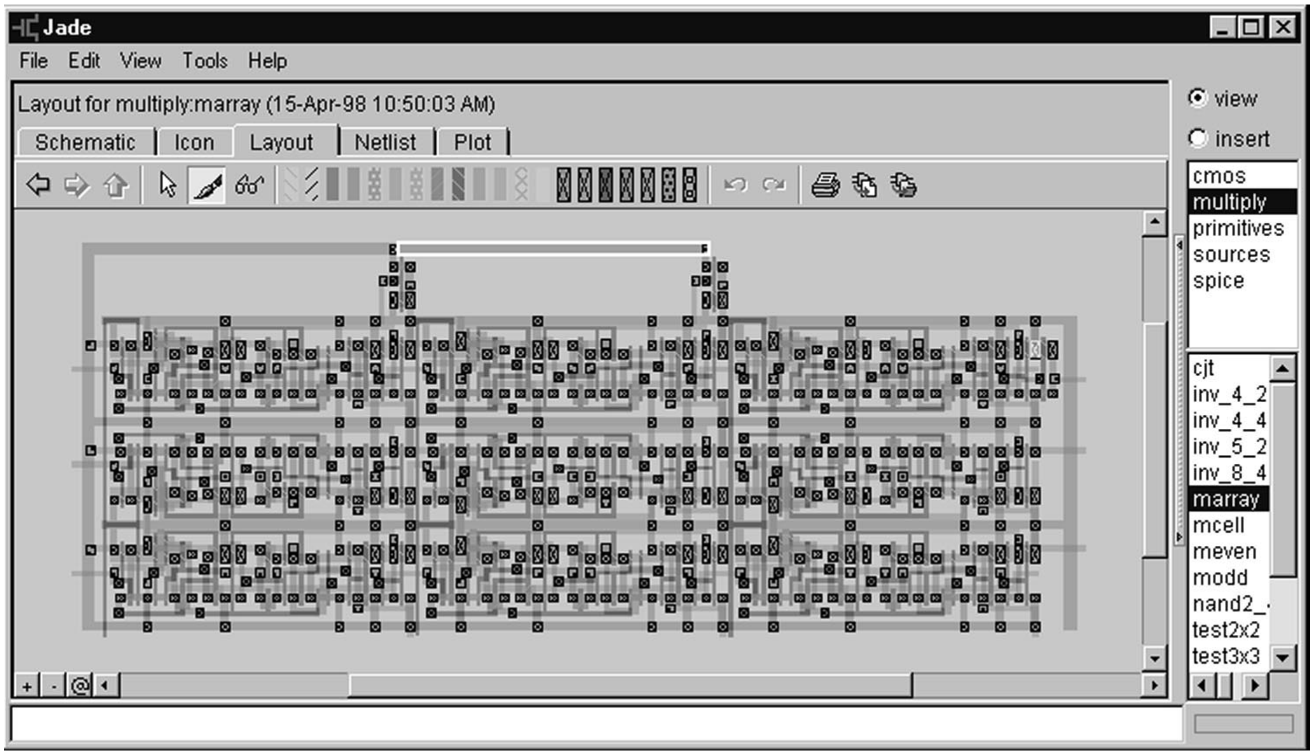


Fig. 12. Top-level cell showing mask layout of 3 × 3 CMOS multiplier using two levels of metal.

for example, creating a donut-shaped piece of mask requires only two steps—painting the background and then erasing the hole—rather than having to assemble separate pieces of mask around the perimeter. It also eliminates much of the fussiness of mask layout, since it is easy to extend or trim mask elements once their correct dimensions have been determined.

V. Implementation

In order to achieve a high degree of integration between the components of the ILE, we were obliged to construct the CAD environment from scratch. For several reasons, we decided to use a browser platform as the basis for the implementation. Browsers potentially offered a standardized cross-platform environment that comes pre-installed on most new machines. This finesses a lot of the traditional problems of deploying the ILE environment to multiple operating systems and hardware platforms. In the end, the browsers were not as successful as we had hoped (or they claimed) at shielding us from platform issues, but they still were a significant improvement over the alternative. Browsers are the focus of intense development efforts by all the major software vendors, so we can expect these platforms to continue to improve in terms of performance, stability, and sophistication.

Accessing the environment over the Web makes it easy for users to always run the most up-to-date versions without the traditional headaches of keeping many copies synchronized with a central repository. Even when it is necessary to make local copies of the environment for portability reasons (e.g., taking a laptop home), there are good Web-based mechanisms for automatically updating the local caches.

Browsers can display material in many different formats (text, video, images, animations, etc.) either directly or via plug-ins. Happily, there are many tools available for creating and serving these materials, so almost no tool work was required to create a rich multimedia presentation. The only real disappointment was the poor facilities for displaying equations in a straightforward manner. We did some spit-and-bailing wire improvisation that got the job done, and XML-based browsers will soon provide a more acceptable solution.

There were several alternatives for integrating the CAD tools into the browser environment. Early on, we decided to implement all the functionality on the client, including computationally intensive tasks such as simulation. A possible alternative was to allocate some tasks to a server, but that scales poorly as the number of users increases in terms of both performance and the storage required to maintain the state of each session. A server-based approach also makes it harder to distribute the system to other institutions or individual users who may not have the necessary server resources. Finally, the computational performance of most client machines does not lag far behind that of most servers, particularly if a significant number of users is sharing the server.

We also had to choose between the two available mechanisms for extending the browser: plug-ins or JAVA applets. Plug-ins are subsystems that interact with the browser code using a well-defined interface; the plug-in code itself is compiled explicitly for whatever platform the browser is running on. This can provide all the performance benefits of a native application: good compilers producing efficient code with the added advantage of potentially reusing existing tool kernels. Unfortunately, plug-ins saddle the developer with all the worries of developing and installing on multiple platforms. We chose to develop the tools in JAVA, which we felt had several advantages. "Write once, run anywhere" is an attractive siren call if one is targeting a diverse community of users. Sadly, this was more illusion than substance—we quickly discovered that there were dismaying differences in functionality and performance among the different JAVA implementations. As a stop-gap measure, we focused on a particular JAVA implementation as the initial target, and it appears this situation will be remedied in the foreseeable future.

Contrary to expectations, the performance of the tools written in JAVA was actually quite good, and it continues to improve with each new release of the JAVA virtual machine. Coupled with the dizzying improvements in the performance of the underlying processors, the tools are more than capable of accommodating reasonable size projects while maintaining an acceptable degree of interactivity.

The multithreading support in JAVA made it easy to use background multitasking for computationally intensive tasks like simulation and design rule checking. This greatly improves the usability of the tools by avoiding the natural tension between accuracy that often requires time-consuming computations and responsiveness, which relies on low-latency execution of commands.

On-the-fly loading of classes should make it possible to add new capabilities to the tools (e.g., support for a new module aspect such as an HDL representation) without modifications to the existing code.

In order to make the tools as approachable as possible, considerable effort was made to make the user interface consistent with other graphical user interfaces—selection, cut/copy/paste, drag and drop, navigation, choice of buttons, and organization of menus are all as one would expect. Many users can get useful work done after only a few minutes with the "getting started" tutorial. It is actually fun to use the tools, not a feeling one would expect to have after experiencing many of the commercially available CAD tools.

VI. Contributions

The ILE seamlessly integrates its components. Upon encountering a circuit schematic or layout in the text, it is easy to browse and modify the figure using the integrated toolkit. It is easy to pose questions and get answers at any level of detail. Thus, students can quickly learn where to look in order to make low-cost experiments. These experiments build intuition about digital MOS circuit behavior and, together with the appropriate theory supplied in the text component and the experience gained by repeated experiments, they form a solid background for creating new, innovative MOS circuits. The editor and simulator environment brings "life" to textual descriptions of circuits, and it facilitates a student's personalized understanding of circuits. Because the environment uses well-known user interface gestures, there is a shallow learning curve, thus providing the student user with valuable insight at minimum cost. Being Web-based, the interactive learning environment poses zero administrative cost to the user, thus making it well suited for distance learning and life-long learning away from an institutional teaching environment.

VII. For More Information

For an online discussion of this special issue, please visit the discussion website at http://ieee.research.umich.edu.

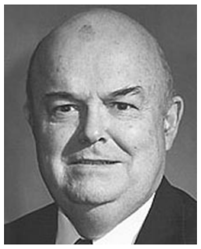
References

[1] S. Rubin, *Computer Aids for VLSI Design*. Reading, MA: Addison-Wesley, 1987.

[2] J. Cherry, "CAD programming in an object-oriented programming environment," in *VLSI CAD Tools and Applications*, W. Fichtner and M. Morf, Eds. Norwell, MA: Kluwer Academic Publishers, 1987, ch. 9.

[3] J. Ousterhout, G. Hamachi, R. Mayo, W. Scott, and G. Taylor, "Magic: A VLSI layout system," in *Proc. 21st Design Automation Conf.*, 1984, pp. 152–159.

[4] G. Taylor and J. Ousterhout, "Magic's incremental design-rule checker," in *Proc. 21st Design Automation Conf.*, 1984, pp. 160–165.



Jonathan Allen (Fellow IEEE) received the A.B. and M.S. degrees from Dartmouth College, Hanover, NH, and the Ph.D. degree from Massachusetts Institute of Technology (MIT), Cambridge.

From 1962 to 1968, he worked at Bell Telephone Laboratories, where he became the Supervisor of Human Factors Engineering in 1966. In 1968, he joined the faculty of the Electrical Engineering and Computer Science Department at MIT, where he is currently Professor. In 1981, he was appointed Director of MIT's Research Laboratory of Electronics. His research activities at Bell Laboratories involved the design of semi-automatic telephone information bureaus and vocoder systems. At MIT, his areas of interest include linguistic techniques for converting unrestricted text to speech, system design for continuous speech recognition, computer architecture, and custom integrated circuit design.

Dr. Allen is a member of Phi Beta Kappa, Tau Beta Pi, and Sigma Xi. He is Past President of the Association for Computational Linguistics.

Christopher J. Terman received the B.A. degree in physics from Wesleyan University, Middletown, CT, and the M.S., E.E., and Ph.D. degrees in computer science and engineering from Massachusetts Institute of Technology (MIT), Cambridge.

He is a Senior Lecturer in the MIT Department of Electrical Engineering and Computer Science and a member of the Laboratory for Computer Science. As a researcher and faculty member at MIT, he implemented several early prototypes for microprocessor-based workstations that led to the development of the IEEE–1996 Bus Architecture (the NuBus). In his work on computer-aided design (CAD) tools for very large scale integration (VLSI) circuits, he developed algorithms for full-chip transistor-level simulation. During his ten years in industry, he cofounded several firms, including Symbolics, Inc. (manufacturer of Lisp Machines), TLW, Inc. (VLSI designs for communications and multimedia), and Curl Corp. (a software technology for the Web). After returning to MIT, he has worked on developing educational technology for use in teaching design-oriented courses.