# MCNC's Vertically Integrated Symbolic Design System

*C. Durward Rogers*
*J. B. Rosenberg*
*S. W. Daniel*

Microelectronics Center of North Carolina
Research Triangle Park, North Carolina 27709

## ABSTRACT

*This paper describes the* VIVID *(Vertically Integrated VLSI Design) System developed at the Microelectronics Center of North Carolina. The system is based on a symbolic, virtual-grid design methodology that greatly reduces the design time for custom VLSI circuits. This methodology has made it possible to provide, in a single integrated system, several unique features: technology independent tools for a wide range of MOS processes (CMOS, nMOS, SOI); scale independent circuit designs; open architecture that simplifies both integration with existing tools and creation of new tools; fast layout debugging using symbolic level circuit simulation; and fully automated mask generation and automated chip assembly.*

## INTRODUCTION

The current VIVID System provides a set of tools for symbolic, virtual-grid layout [WEST81a] of custom digital VLSI circuits. These tools facilitate work at a high level, free from concern with design rules. The designer places circuit primitives, such as transistors and wires, on the virtual grid much as one would sketch the layout on a piece of graph paper. The system uses a compactor program for translating the primitives into their appropriate mask representation and for spacing them (according to the design rules) to generate the mask layout.

This approach offers the designer several advantages over the traditional mask editing approach to custom layout:

- Technology independence

This is perhaps the most important advantage. The same tools work over a broad range of MOS technologies (nMOS, CMOS, and SOI) and, within a specific MOS technology, a given design tracks changes in the design rules without redesign.

- Correctness-by-construction

The symbolic, virtual-grid approach promotes correctness-by-construction by eliminating the physical layout step. The compactor generates the physical layout; therefore, the designer does not have to be aware of the design rules in order to create an error-free layout.

- Natural interface

The symbolic, virtual-grid environment provides a clear and natural interface not only for the designer but also for the tool developer. The VIVID system can be used to hide the details of the design rules from other tools (such as structure generators or special purpose routers) and greatly simplify their development.

For the designer, the main result of these advantages is a significant reduction in the time needed to generate a working design. In addition, the ability to quickly and easily change the layout encourages designers to take a more global view of their design. These changes often improve circuit performance and layout density over conventional detailed mask editing.

## SYMBOLIC, VIRTUAL-GRID LAYOUT

Symbolic, virtual-grid layout can be viewed as an evolutionary refinement of physical mask layout. In physical mask layout, the designer specifies the circuit by drawing a set of polygons that indicate how to create a mask for each layer in the fabrication process. At the physical mask level, the basic elements of circuit design (such as transistors or contact cuts) are composite structures. Each transistor or contact cut is composed of polygons on several layers that are sized and positioned according to the design rules of the target fabrication process. In physical mask layout, each time one of these composite structures is needed, it is re-created from the component



0738-100X/85/0062$01.00 © 1985 IEEE

polygons. Symbolic layout provides a solution that eliminates this tedious and error-prone task.

With symbolic layout, symbols are provided to represent the most common structures. The designer organizes the symbols into a layout and the computer translates them into the proper mask representation. In its simplest form, the translation is done by replacing the symbol with a fixed collection of polygons that implement the desired structure. (Many physical layout systems provide translation with a "macro" feature.) A more flexible approach to symbol translation is to associate parameters with the symbols and to have a program use the parameters for generating a broad range of structures. For example, the symbol for a transistor might be accompanied by two parameters that specify the width and length of the gate region. The transistor generation program would then use the parameters to size the transistor when constructing the mask layout.

Like symbolic design, virtual-grid layout is an extension of physical mask design. In physical mask design, the layout is usually created on a grid. The spacing of the grid represents some "real" spacing (for example, $3\mu$) and the designer uses the grid as an aid to establish correct spacing between objects. The function of the virtual grid is the same as for a "real" grid except that the spacing between grid lines does not represent a fixed physical spacing. A symbol's placement captures only the relative geometry of the circuit. (For example, transistor A is above and to the right of transistor B.) The actual spacing between two adjacent grid lines is determined by the compactor program. The compactor examines the objects on adjacent grid lines and, based on the design rules, determines the correct spacing between the grid lines. This approach results in an appropriate division of labor — the designer makes the global decisions about the circuit topology and the computer performs the detailed geometric construction.

## SYSTEM IMPLEMENTATION

Figure 1 shows a conceptual view of the VIVID System. It is divided into two main sections: symbolic layout and physical layout.

The symbolic layout section contains the tools used to generate and manipulate symbolic, virtual-grid layouts, specified in the ABCD (A Better Circuit Description) language. The ABCD language provides a direct and well-defined interface to the symbolic, virtual-grid portion of the VIVID System. In the physical layout section, the mask descriptions generated by the compactor are specified in the LLAMA (Layout Language for Mask Artwork) language. Tools are provided to translate between LLAMA and other mask description languages [NEWE83,HON80] .

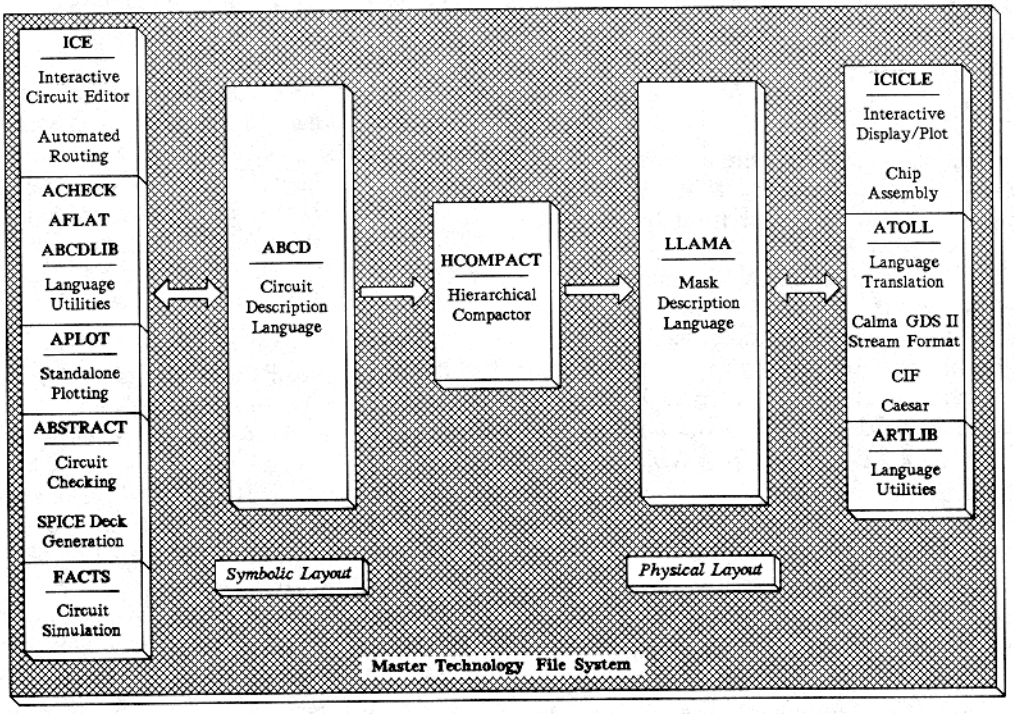



**Figure 1. Components of the VIVID System**

## 1. SYMBOLIC VIRTUAL-GRID LAYOUT

The key element of the symbolic layout section of the system is the ABCD language [ROSE83b] . Figure 2 shows the ABCD description of a complementary pair of transistors and a graphical representation of their symbolic, virtual-grid layout.

```
#
# Cmos cell with pair of transistors.
#
begin pair bbox=(1,2,3,4)
        pullup:     device p_type      (2,4) w=2 l=1
        pulldown:   device n_type      (2,2)
```

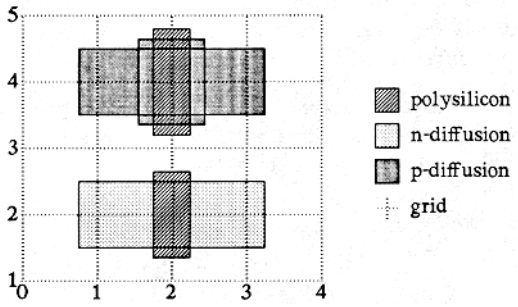



**Figure 2. Symbolic, Virtual-grid Layout of Two Transistors**

Note that the example is drawn using a symbolic, virtual-grid representation and not a mask representation. Throughout the system, however, the high-level symbolic representations are chosen to be similar to the final mask layout.

## 2. VIRTUAL GRID COORDINATES

In the figure, the cell "pair" occupies a rectanglar section of the virtual grid, bounded at the lower left by the point (1,2) and at the upper right by the point (3,4). The virtual grid extends in all directions from (0,0) and is numbered with integer values. The two devices in the illustration are positioned at points (2,4) and (2,2), respectively. There is nothing special about this or any other section of the grid; the devices could have been placed just as easily at (-200,-20) and (-200, -22). The important information captured by this grid is that one device is above the other.

## 3. TYPE PARAMETER

The "device" symbols are qualified by several parameters. The "type" parameter indicates the general classification of the device; in this CMOS example the two types of devices are "p_type" and "n_type". In an nMOS design the types might be enhancement, depletion, and load. The Master Technology File (MTF) System [SMIT85] actually provides the designer with a large amount of flexibility to define new types of devices.

## 4. SIZE PARAMETERS

The p-type device has two parameters, "w" and "l", that specify the width and length of the device's gate region. As with the virtual grid locations, these parameters do not necessarily represent a fixed physical value. In fact, much of the system's ability to express designs that will correctly scale with a changing technology is dependent on the interpretation of these size parameters. Like device types, this interpretation is controlled by the MTF System, giving the designer a broad range of choices for the interpretation. In the system at MCNC, the size of a device is expressed relative to a "unit" size device. Therefore, the p-type transistor in figure 2 has a gate that is the same length as the unit device but that is twice its width. (Note that the graphical representation indicates the extra gate width by a wider diffusion box around the gate region.)

How "unit" size is defined is very important to the interpretation of the symbolic layout. It is possible to define the unit size as a fixed physical size such as $2\mu$ and then to scale the other devices proportionally. The unit size also could be expressed in some basic process scaling unit such as $1\lambda$. There is, however, a simple and natural definition for the unit size that captures the designer's intent and allows the system to intelligently scale the design; the unit size for any feature is its minimum size in the fabrication technology being used. This definition provides the designer with a convenient simplification for requesting the smallest (and fastest) device offered by the process. This "smallest-available-size" rule also applies to wire widths so that a width 1 wire represents the smallest wire available to the designer. The value of this definition of unit size is that it ties the sizing specifications to the electrical performance of the technology and not to fixed physical dimensions.

When larger sized devices (or wires) are specified, the designer can configure the MTF System so the size specification expresses relative electrical performance. For example, the designer can define the width-2 device to be a size that results in twice the drive capability of a unit device. In a relatively large scale technology, this will simply be a device twice the width of the unit device. However, in smaller technologies where higher order effects may dominate device performance, this simple linear relationship may no longer hold. To obtain twice the drive capability the designer may have to define the width-2 device to be something other than twice the unit size width. Once this is done, however, the electrical intent of the original specification is

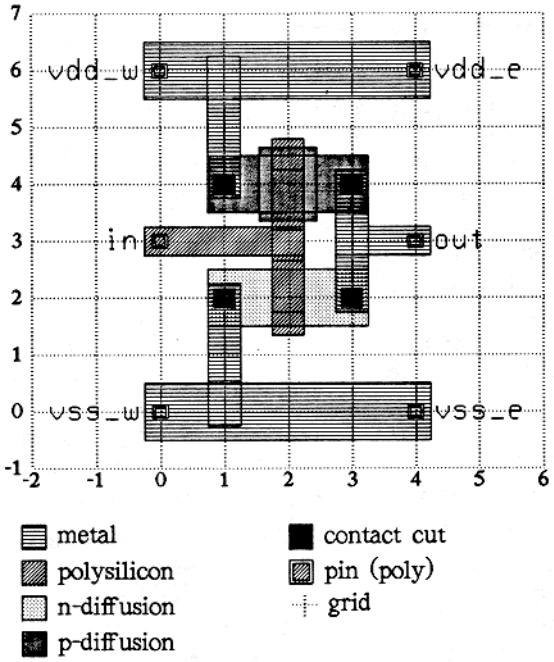
preserved. Existing designs can then be re-compacted to produce circuits that function correctly and take full advantage of the smaller technology.

## 5. NAME PARAMETER

Figure 3 represents a complete implementation of a CMOS inverter created from the cell "pair".

**Figure 3. CMOS Inverter Cell**

```
#
# Cmos inverter
#
begin inverter bbox=(0,0,4,6)
# Establish I/O locations
    vdd_w:    pin    alum    (0,6)
    vdd_e:    pin    alum    (4,6)
    vss_w:    pin    alum    (0,0)
    vss_e:           pin     alum    (4,0)
    in:       pin    poly    (0,3)
    out:      pin    alum    (4,3)
# Width 2 power and ground rails
    wire      alum   w=2     (vdd_w) (vdd_e)
    wire      alum   w=2     (vss_w) (vss_e)
# Transistors
    pullup:   device p_type  (2,4)  w=2
    pulldown: device n_type  (2,2)
# Gate connection
    wire      poly   (pullup.g) (pulldown.g)
    wire      poly   (pullup.g,in) (in)
# Connect sources to power and ground
    wire      alum   (pullup.s) (pullup.s, vdd_w)
    contact   md     (pullup.s)
    wire      alum   (pulldown.s) (pulldown.s, vss_w)
    contact   md     (pulldown.s)
# Connect drains
    wire      alum   (pullup.d) (pulldown.d)
    contact   md     (pullup.d)
    contact   md     (pulldown.d)
    wire      alum   (pullup.d, out) (out)
end inverter
```




This example illustrates how the name parameter can be used to specify the locations of circuit primitives. Most of the primitives in the cell are placed by simply referencing a named location. This reference-by-name feature is useful in several important ways.

- Text-based design is much easier.

In the VIVID System most layouts are created graphically using ICE (Interactive Circuit Editor); however, it is often convenient to create simple cells through standard keyboard entry. (Textual input may be particularly useful in instructional settings in which a sufficient number of graphics devices are not available.) Furthermore, if the specification of cell placement does not involve numerical calculations, assembly and manipulation of the hierarchy are easily performed with a text editor.

- Reference-by-name is self documenting.

A wire connection referenced to *pullup.g* (the gate of the device called "pullup") is more meaningful to both the designer and the system than a reference to the point (2,4). At present, the system preserves and uses references so that circuit elements can track changes in the positions of the elements they reference. Future versions of the system will make more extensive use of this feature to capture and preserve the electrical connectivity of circuits.

- Tool development is easier.

Software using reference by name can generate and manipulate designs that are not sensitive to the absolute locations of circuit elements. A few key positions can be located numerically and the rest of the cell constructed relative to those positions. The software can then adjust the size of the cell without making extensive changes to its specification.

## 6. CREATING ABCD LAYOUTS

The examples of ABCD designs given above perhaps over emphasize the textual nature of the ABCD language. Although it is human readable, it was not designed as a text-entry layout language. ABCD is intended to be a target language for high-level design tools such as graphical editors [EADE76,OUST81,OUST83,KELL81] , structure generators, and silicon compilers [JOHA79,SISK82] . The advantage it offers these tools are similar to the advantage offered by a computer's assembly language — hiding of the implementation details. In the case of the ABCD language, these details are the design rules for the fabrication technology.

The capabilities of the ABCD language are used by ICE, the VIVID System's graphical editor. Rather than

editing geometrical objects, ICE edits statements in the ABCD language, which it represents graphically. Using ABCD relieves the editor of the burden of managing numerous geometric details and simplifies many of the editing functions. This simplification reduces layout time and allows the editor to provide higher-level functions such as area operations, floorplanning [HELL82, MUDG80], and automated routing [Pinter1981, LEE61].

## LAYOUT VERIFICATION

The VIVID System provides two tools for verifying symbolic virtual-grid layouts: a symbolic level circuit extractor and an interactive circuit simulator. Design verification directly from the symbolic, virtual-grid layout rather than from the mask layout offers the advantage of fast response. This quick response allows the designer to perform extensive circuit debugging early in the layout process.

### 1. CIRCUIT EXTRACTION

The symbolic level circuit extraction is performed by the ABSTRACT (ABCD Circuit Extractor) program. ABSTRACT references the MTF System to calculate the electrical parameters associated with each circuit element. The calculated values are, by necessity, estimates since the mask generation has not been performed. However, these estimates are relatively accurate for all of the primitives except wires, which are directly dependent on the final size of the layout. Reasonable estimates of wire length can be obtained by assuming that the spacing between the virtual grid lines will average out over the design. This average grid spacing parameter is coded in the MTF System and can be tuned by the designer according to the technology being used and the performance of the compactor.

In conjunction with the circuit extraction, ABSTRACT performs error checking and provides the designer either textual or, via ICE, graphical feedback. The types of errors it can detect are overlapping or improper abutment of cells, unconnected or short-circuited components, and improperly named signals. Both the extraction process and the error checking are relatively fast since the extractor does not have to go through the costly process of inferring the circuit structure from the mask geometry. For example, the extraction of a 1200 device serial multiplier took about 1 minute on a Digital Equipment Corporation (DEC) VAX 11/780.

### 2. SIMULATION

Circuit simulation is performed by the FACTS (Fast Circuit Simulator) program. The simulator has been designed for MOS simulations and can be used with circuits as large as several thousand devices. The speed of FACTS results from its selection of models and internal structure. Only MOSFET models are used and FACTS precalculates tables of simulation values before beginning a simulation. During the simulation, the designer can choose between two types of current modeling: a simple transistor current model or a more accurate second order model with saturation, linear, and cutoff regions; channel length modulation; drain and source threshold dependence; and capacitance modeling. FACTS also monitors all node voltages and, when the changes are small, increases the time step to avoid redundant or insignificant calculations.

Because of its simpler modeling and use of symbolic, virtual-grid extraction, FACTS does not provide the accuracy of a full network analysis program. However, FACTS fills a gap between such programs and logic level simulators. It is faster than a detailed circuit simulator but still accurate enough to provide the waveform information necessary for debugging the analog behavior of a circuit. FACTS offers interactive features (such as a probe capability to interrupt a simulation and observe any set of circuit nodes) to support its function as a debugging tool.

## MASK GENERATION

The creation of a mask description from a symbolic, virtual-grid layout is accomplished by HCOMPACT (Hierarchical Compactor) [ENTE85]. The compactor reads ABCD designs and generates a LLAMA mask description of the circuit. The compactor operates in two distinct steps: leaf cell compaction [HSUE79, WATA84] and hierarchical compaction.

### 1. LEAF CELL COMPACTION

Leaf cells contain only circuit primitives: wires, devices, contacts and pins arrayed on the virtual grid. The compactor translates the symbols into their mask representation and then spaces them according to the design rules. Much of the difficulty of compaction arises because these two steps are not independent. In particular, the placement of wires and contacts on rigid structures (such as transistors) is dependent on the location of adjacent circuit elements. The compactor solves this problem by augmenting rigid structures with flexible wires that the compactor may extend when making connections. These wires "decouple" the rigid structures from the rest of the layout and allow the compactor to treat the mask spacing problem in a uniform manner.

Mask spacing is determined during two passes (one vertical and one horizontal) across the cell. For each grid line it determines the minimum spacing possible between it and its adjacent grid line. Empty grid lines do not effect the layout because they are

assigned zero spacing. Once the spacing has been established for a grid line all of the objects on the line are positioned together. This type of compaction inevitably sacrifices some area but makes it possible for the hierarchical portion of the compactor to automatically pitch match cells. This approach also has the advantage of simplicity. The designer can quickly develop an understanding of the compactor and, if needed, adjust the symbolic design to improve the layout.

### 2. HIERARCHICAL COMPACTION

The hierarchical portion of the compactor compresses the hierarchy into a set of cells that completely cover the circuit layout. For each distinct leaf cell, the compactor analyzes all of the environments in which the cell occurs and generates a worst case version of the cell. The compactor then performs a leaf cell compaction and obtains a mask cell that can be placed in any of the original environments without causing design rule errors. Once all of the leaf cells have been compacted, the mask cells are assembled according to the original layout.

The main problem in the assembly phase is pitch matching adjacent cells. Since each of the leaf cells is compacted separately, there may be wires on abutting cell boundaries that matched on the virtual grid but are offset now that the physical spacing has been established. The compactor, however, retains information on the original virtual grid layout so it can stretch the cells by the appropriate amount to insure a match. The cells cannot be compressed to achieve a match because the leaf cell compactor has already produced the smallest layout it can for each cell. The final output of the compactor is a mask description of the circuit in the LLAMA language.

## PHYSICAL LAYOUT

The generation of a mask description by the compactor is the last step in the symbolic portion of the system and the first step in the physical portion. The current beta release version of the system does not provide tools for manipulating mask layouts. Rather, it serves as a front-end processor for existing mask layout systems. To aid in this role, ATOLL (A Translator of Layout Languages) is provided to translate between LLAMA and the other mask description languages supported by the system (Calma GDS II, CIF, and Caesar). ATOLL actually functions as a crossbar switch so it can be used to convert between CIF and Calma if necessary. The architecture of ATOLL has been structured to facilitate the addition of other mask description languages.

The VIVID System's dependence on other mask layout tools will soon be eliminated by two tools that are currently under development: ICICLE (Interactive Circuit Layout Editor) and the chip assembler. An output only version of ICICLE is included currently. It can be used to view LLAMA designs and generate hardcopy. ICICLE makes use of a fast geometric data structure (KD-trees) [ROSE85] .

The chip assembler will be available in the first release or as an update shortly thereafter. To use the chip assembler the designer specifies a frame format, pad types, and a net list. The chip assembler builds the pad frame, inserts the circuit, and performs the routing. The system include pads and pad frame formats for the MOSIS nMOS and bulk CMOS processes. Additional pads and the pad frame formats can be created by the designer.

## MTF SYSTEM

We have discussed how the VIVID system makes circuit design easier and more efficient by relieving the chip designer of burdensome detail work. This does not mean that these details are not important or that they are not under designer control. The VIVID system configures itself to a particular process technology and environment by interrogating the Master Technology File System (MTF).

The MTF system contains all of the VIVID system's knowledge of circuit primitives. It controls their representation in ABCD by defining the names of device types and process layers. It controls circuit display by defining the symbol shapes, colors, and stipple patterns. It controls the rendering of circuit primitives in mask layout form by providing symbol to mask translation rules. Finally, it contains information about the primitives electrical properties, capacitances, sizing, and "best," "worst," and "average" case transistor models

Although it is difficult to create a new MTF, it is not difficult to modify an existing file to reflect changes in the process parameters. The current system is supplied with MTF's for the MOSIS nMOS and bulk CMOS processes. These have a great deal in common with other processes and can be used as models for creating a new MTF.

## REFERENCES


[EADE76] EADES, J. D., "The Design of an Interactive Computer System for Microelectronics Mask Making," *PhD Thesis*, University of Edinburgh, (1976).

[ENTE85] ENTENMAN, GEORGE AND STEPHEN W. DANIEL, "A Fully Automatic Hierarchical Compactor," *Proc. 22nd Design Automation Conf. [forthcoming]*, (June 1985).

[HELL82] HELLER, WILLIAM R., G. SORKIN, AND KLIM MALING, "The Plannar Package Planner for Systems Designers," *19th Design Automation Conference*, pp. 253-260 (1982).

[HON80] HON, ROBERT W. AND CARLO H SEQUIN, *A Guide to LSI Implementation*, Xerox PARC, Palo Alto, CA (1980).

[HSUE79] HSUEH, D. O. AND D. O. PEDERSON, "Computer-aided Layout Of LSI Circuit Building-blocks," *Proceedings 1979 Iscas*, pp. 474-477 (1979).

[JOHA79] JOHANNSEN, D., "Bristle Blocks: A Silicon Compiler," in *Proceedings of the Cal Tech Conference on VLSI*, (January 1979).

[KELL81] KELLER, K. H., "KIC: A Graphics Editor for Integrated Circuits," *Masters Thesis*, EECS Dept, U.C. Berkeley, (1981).

[LEE61] LEE, C. Y., "An Algorithm for Path Connections and Its Applications," *IRE Transactions on Electronic Computers* EC-10(3) pp. 346-365 (September 1961).

[MUDG80] MUDGE, J. CRAIG, C. PETERS, AND G. M. TAROLLI, "A VLSI Chip Assembler," pp. 329-355 in *Design Methodologies for VLSI*, NATO Advanced Summer Institute, Belgium (1980).

[NEWE83] NEWELL, SYDNEY B., AART J. DE GEUS, AND RONALD A. RHORER, "Design Automation for Integrated Circuits," *Science* 220(4596) pp. 465-471 (April 29, 1983).

[OUST81] OUSTERHOUT, JOHN K., "Caesar: An Interactive Editor for VLSI Layouts," *VLSI Design*, pp. 34-38 (Fourth Quarter, 1981).

[OUST83] OUSTERHOUT, JOHN K., GORDON T. HAMACHI, ROBER N. MAYO, WALTER S. SCOTT, AND GEORGE S. TAYLOR, "Magic: A VLSI Layout System," A Collection of Papers on Magic, EECS UC Berkeley, Berkeley, CA (December 1983).

[Pinter1981] PINTER, RON Y., "Optimal Routing in Rectilinear Channels (for VLSI)," in *Unpublished Research Report*, MIT Laboratory for Computer Science, Cambridge, MA (April 15, 1981).

[ROSE83b] ROSENBERG, JONATHAN B. AND NEIL H. E. WESTE, "ABCD - A Better Circuit Description," MCNC Tech. Rep. No. 4983-01, Microelectronics Center of NC, RTP, NC 27709 (Feb. 1983).

[ROSE85] ROSENBERG, JONATHAN B., "Geographical Data Structures Compared: A Study of Data Structures Supporting Region Queries," *IEEE Trans. on Computer-Aided Design* CAD-4(1)(Jan. 1985).

[SISK82] SISKIND, JEFFREY M., JAY R. SOUTHARD, AND KENNETH W. CROUCH, "Generating Custom High Performance VLSI Designs From Succinct Algorithmic Descriptions," *MIT Conference on Advanced Research in VLSI*, pp. 28-40 (1982).

[SMIT85] SMITH, PHILIP G. AND STEPHEN W. DANIEL, "The Master Technology Files System for Technology Independence," *Proc. 22nd Design Automation Conf. [forthcoming]*, (June 1985).

[WATA84] WATANABE, HIROYUKI, *IC Layout Generation and Compaction Using Mathematical Optimization*, Semiconductor Research Corporation, Research Triangle Park NC (February 1984).

[WEST81a] WESTE, NEIL H. E., "Virtual Grid Symbolic Layout," *Proceedings 18th Design Automat. Conf.*, pp. 225-233 (June 1981).