Copyright © 1981 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 60, No. 6, July–August 1981
Printed in U.S.A.

MULGA—An Interactive Symbolic Layout System for the Design of Integrated Circuits

By N. H. E. WESTE

(Manuscript received November 13, 1980)

To aid the design of MOS *circuits, a suite of programs residing on the UNIX* operating system have been designed and written. These programs allow the interactive editing, layout compaction, circuit connectivity extraction, parasitic audit, and timing simulation of* MOS ICs *within the symbolic domain. The programs make use of an intermediate circuit description language (*ICDL*), which captures both geometric placement and circuit connectivity. A convenient interface is provided to enable the procedural definition of symbolic layouts in the C programming language. All design may be carried out at a single low-cost work station which incorporates a high-performance color display. In this paper we summarize the operation and use of these programs. In particular, we describe a new compaction algorithm.*

I. INTRODUCTION

Symbolic layout methodologies are a means of abstracting the detailed and often laborious task of mask design of integrated circuits. They offer the advantages of hand-packed mask design with regard to density of layout, while also having advantages over manual layout with respect to time to design a circuit and reduction in the number of manual errors introduced into a design. In essence, the use of symbology reduces the complexity of the IC design process, which in addition to the advantages mentioned above, allows experienced designers to undertake more complicated circuits than would otherwise be possible, and, more importantly, allows novice designers to complete designs with a high degree of confidence. This last point is regarded as

* *UNIX* is a trademark of Bell Laboratories.

especially important as system designers move to use silicon as an implementation medium, rather than more conventional techniques.

A particular IC design may be described in three domains, namely, the structural, physical and behavioral domains. Structurally, a design may be thought of as a graph where components are represented by nodes in the graph and their interconnection modeled by branches. The graph nodes may in turn be represented by subgraphs. The physical domain in an integrated circuit is typified by a collection of geometric areas defined on masks used in the various process steps used to fabricate the chip. Finally, the behavioral domain indicates how the design functions from the level of electrical circuit performance to possibly higher levels of architectural simulation. Manual layout, for instance, captures only the physical attributes of a design. Standard cell layout may achieve consistent descriptions in all three domains at the expense of fixing the physical layout to a set of predefined cells.

This paper describes a symbolic layout system (given the name MULGA) which attempts to achieve consistent descriptions in all three domains in addition to providing a highly interactive environment. The cells designed with this system may be used with standard cell layout systems such as LTX,[1] or combined using structured design techniques as popularized by Mead and Conway[2] to form super-cells with considerable functionality. In the limit, complete chip descriptions may be completed using the tools that will be described in the body of this paper. All programs are written in the C programming language and run under the *UNIX*™ operating system. The system has been designed to enhance man–machine interaction, providing a friendly interface to designers with wide ranges of experience in the areas of electrical engineering and computer science.

The paper is divided into four main sections. The first section relates this work to previously published work by examining symbolic design methodologies. We then describe the circuit description language which forms a basis for this system. The design methodology used in this system, the design aids and the hardware that support this methodology are then described. The final section summarizes the work and derives conclusions about the design system and methodology.

II. SYMBOLIC LAYOUT METHODOLOGIES

Symbolic layout methods attempt to abstract the detailed task of designing IC masks to clarify this operation. Normally this is achieved by eliminating or reducing the complexity of the design rules for a given process. These design rules include the minimum spacings and widths of the mask layers used in the technology. They also include electrical rules for interconnecting layers and the formation of active devices. These simplified rules ideally result in a quicker turn-around of designs and a reduction in errors compared to manual layout. This section illustrates various symbolic design methodologies and their contributions to the IC design process.

2.1 Coarse-grid layout

Coarse-grid layout systems divide the chip surface into a uniformly spaced grid in both the x and y directions. The grid size represents the minimum feature or placement tolerance that is desired in a given process. For each combination of mask layers that exist at a grid location, a symbol is defined. Given a particular design system, these symbols are then placed on the grid to construct the desired circuit much in the same way as one would tile a floor. Symbol sets may be defined as characters or perhaps graphical symbols if a graphics display is used for design.

American Microsystems International (AMI) and Rockwell International have made use of character-based symbolic layout for some time.[3–5] The symbolic interactive design system (SIDS)[5] uses a color character terminal as a design station which provides a high degree of user feedback. In addition to these character-based systems, Hewlett-Packard has developed an interactive graphics system (IGS),[6] which is capable of accepting symbolic input on a fixed grid. IGS also uses symbolic representations to reduce the time to display hand-designed layouts.

The design process in these systems consists of laying symbols down on the coarse grid. The use of fixed-size symbols simplifies geometric design rules but does not totally alleviate them. SIDS therefore provides on-line design rule checking for geometric design rule violations, and a "trace" facility to trace circuit nets to visually check electrical connectivity. Similarly, IGS provides "bumpers" which surround symbols to aid designers in the placement of them.

2.2 Gate-matrix layout

Recently, a character-based symbolic layout system has been in use at Bell Laboratories for the design of large CMOS circuits.[7] This system departs from fixed-grid systems principally because it adopts a design style which involves structuring the way the active and interconnection layers of the process are used. This style has been given the name "gate matrix."

In this layout system, polysilicon is allowed to run in vertical columns of fixed width and pitch. Diffusion is allowed to run horizontally or vertically in "nonpoly" columns. The intersection of a poly column and a diffusion row forms a transistor. The metal layer is allowed to run in both vertical and horizontal directions to complete connections. The design process involves identifying gate signals and assigning a poly column per signal. Transistors are then placed and interconnected using diffusion columns or Manhattan metal connections. If necessary, extra poly columns may be added to complete a circuit.

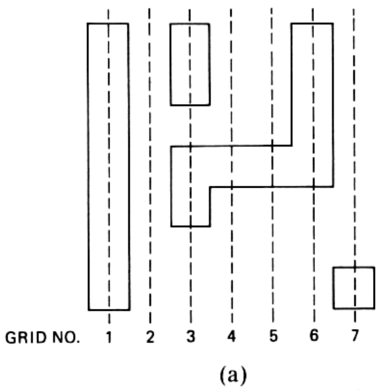
This system achieves better density than a comparable fixed-grid layout and moreover has demonstrated "hand-packed" densities on large circuits. This seems to be due to the fact that the designer is provided with a rigid but structured design technique which allows the corresponding "grids" to be placed closer together than in coarse-grid systems.

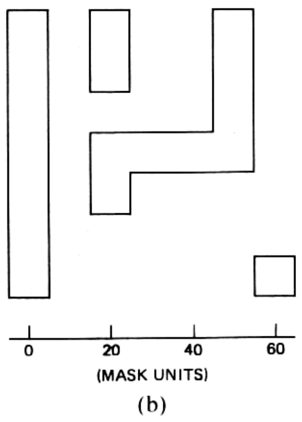
2.3 Sticks layout

The term "sticks" is a generic term given to symbolic design systems that do not necessarily constrain the designer to a grid and generally require the designer to enter a free-form topological description of a layout via an interactive graphics system. Graphical symbols are placed relative to each other rather than in an absolute manner. Systems representing this form of layout have been reported by Williams,[8] Dunlop,[9,10] and Hseuh et al.[11] Following the definition phase, the symbolic descriptions are converted to valid mask descriptions using a variety of compaction strategies designed to space symbols in accordance with the process design rules. In this manner, a rough topology is entered prior to an exact geometric description being generated. This is as a result of the designer being totally freed from geometric design rules. However, examples of ICs designed using these systems have not been reported to date.

2.4 Virtual-grid layout

The MULGA system uses a grid-based placement scheme as in coarse-grid layout methods but allows the final geometric spacing between grid lines to be determined by the density and interference of elements on neighboring grid locations. This leads to the notion of a "virtual grid."[12] This concept is best illustrated by a simple example as shown in Fig. 1a. Three vertical wires are shown centered on a grid. The result of using a fixed grid of 10 units per grid unit and a wire width and separation of 10 units leads to the mask description shown in Fig. 1b. By using a grid in which the spacing varies according to topology, the mask description in Fig. 1c is constructed. Grid-based placement allows rapid entry of geometric topology by "snapping" elements to the grid. The use of a grid also aids the capture of the circuit details and the subsequent processing needed prior to preparing a valid set of masks.





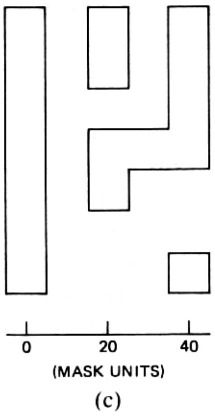


Fig. 1—Comparison of fixed-grid and virtual-grid layouts. (a) Circuit specification based on a grid. (b) Fixed-grid expansion. (c) Virtual-grid expansion.

The notion of a "coordinode" as introduced by Buchanan[13] is used as an aid to capturing circuit connectivity. A coordinode is defined as, "a named object which has structural, physical, and behavioral significance in the IC design process."

As its name suggests, a coordinode has the properties of a coordinate, namely some *xy* position that will eventually map onto the silicon surface. In addition, it may possess the properties of a node in a circuit, perhaps a voltage or a simulation state. Thus both physical and behavioral models may be assigned to the coordinode. The structural significance of a coordinode is that it defines nodes in the graph description of the network that is used to extract the behavioral aspects of the design.

In the MULGA system the nature of a coordinode is altered slightly to suit the interactive symbolic environment in which a designer works. Coordinodes may only exist at coordinates on the conceptual grid, a result of dealing at the symbolic level rather than the geometric level. In addition, coordinodes need not be specifically named, a result of the desire to have a free-flowing interactive dialog between designer and computer.

2.5 Benefits of symbolic layout

Symbolic layout may aid the design process in two ways. First, the simplification of geometric design rules relieves the designer of detail that can cloud more global and important issues, such as achieving the correct circuit or communication requirements. Transparent design rules also make designs relatively process-independent. If a process design rule changes, the mask descriptions for a circuit may be regenerated with a minimum effort. The second way that symbolic layout may aid the designer is by capturing designer intent. This means that apart from capturing the geometrical details of a design, the design methodology supports capturing the circuit embodied by a design. In other words, the physical and structural aspects of a design may be caught early in the design cycle. An example of this is the case when a designer places a metal path on top of a diffusion region and places a contact at the center of the common area. Physically, a set of rectangular areas on differing mask layers have been specified in size and placed somewhere in the xy plane representing the complete design. Structurally, the net represented by the metal wire on some circuit diagram has now been extended to include the diffusion region, which could, for instance, be the source or drain of a transistor.

The first computer-aided symbolic-layout systems, as represented by coarse-grid layout systems, made only partial use of the full advantages of symbolic layout. In particular, very few if any structural details were captured as the designer was forced to think in terms of tiles of various process layers rather than circuit elements. In addition, circuit verification had to take place on the basis of a program recognizing these tiles in particular configurations that constitute circuits. As noted previously, geometric design rules were only simplified, not totally alleviated.

The more recent sticks systems have, in addition to the perceived benefits of a design rule free environment, the basis for capturing circuit connectivity, although none have treated this benefit in detail. This is due to the fact that specific problems have been addressed, in particular compaction, rather than the complete design cycle and the relevant tools required.

The system described in this paper takes full advantage of the above-cited benefits of symbolic layout in a consistent and logical fashion. Section III describes a convenient language interface.

III. INTERMEDIATE CIRCUIT DESCRIPTION LANGUAGE (ICDL)

As previously discussed, one wishes to capture physical, structural, and behavioral attributes of a design during the specification phase. In addition, in this system a prime requirement was for a highly interactive environment to support "on-line" design. In contrast to the integrated circuit system (ICSYS)[13] which is implemented in terms of an object-based language, the MULGA system has been implemented in terms of two languages. The first is a primitive but effective intermediate circuit description language (ICDL) that contains physical, structural, and derivable behavioral characteristics. The second level of design supports procedural design in the *C* language, which enables a designer to write a program that can generate ICDL. All of the software described in this paper uses the ICDL text descriptions of cells as a central data base.

The basic element of the language is a cell, which is composed of elements that may be devices, wires, contacts, pins, or cell instances. This is similar to the set of elements used in Ref. 11. However, rather than being specified in mask coordinates, these elements exist on a virtual grid. As will be seen, this grid serves only as a relative placement framework during the design process and has attributable mask-geometric properties only after further processing.

3.1 Devices

Devices may be a variety of types and are specified by the keyword **device** for a driver type transistor, or **load** for a resistive transistor followed by the device type which may be:

p	*p*-type device,
n	*n*-type device.

Devices have attributes of position on the coarse grid, and optional orientation, width, and length parameters. A device is specified as follows:

device *type xpos ypos* <**w** = *width*> <**1** = *length*>
< **or** = *orientation* >

The *type* parameter may define different subclasses of devices, e.g., different implant transistors. Dependent on the position, orientation, width, and length of the device, the position of the source, drain, and gate connections are defined. In this implementation, the position also depends on whether a transistor is a load device or a driver device. The position may be symbolic grid point, line or area, dependent on the connection restrictions that are desired on a device.

For instance, in this implementation a standard driver type device has terminals as shown in Fig. 2. The connection points, lines, or areas, and the basic definitions of the devices are somewhat process depend-

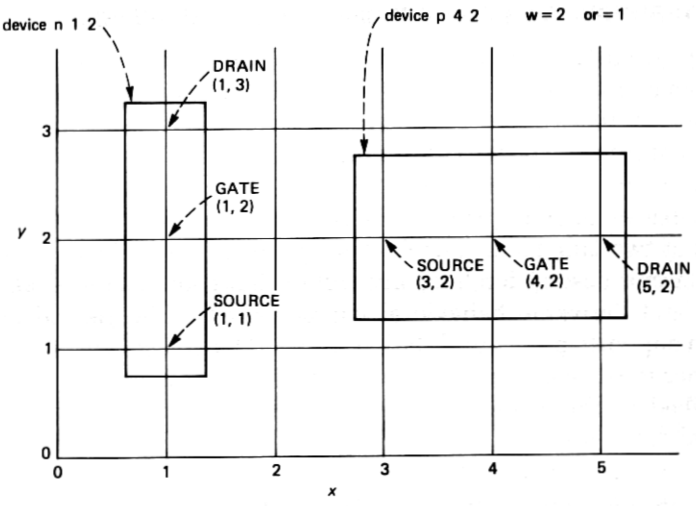


Fig. 2—Driver transistor connections with varying orientations.

ent and must be selected in such a way as to yield a meaningful symbolic representation of the actual layout—both physically and structurally.

The width and length parameters in this implementation are referenced to default minimum width and length devices. Usually the final size of the device gate region is directly scaled using the width and length values. This significance may be changed to suit the process or flexibility required. For instance, increasing the width by a factor of one may increase the actual device size by, say, 2 μm.

It may be seen that transistors dependent on type and parameters have a certain physical realization. The definition of predefined contact points ensures that the structural requirements of a circuit may be met in a logical fashion at the time of capture of the circuit. The behavioral characteristics of a transistor are subsequently found by simulation, based on the physical and structural attributes.

3.2 Wires

Wires can exist on any of the valid interconnection layers of a given process. They are specified by the following statement:

wire *layer* <**w** = *width*> *x1 y1*, . . . , *xn yn*,

where *x1 y1*, . . . , *xn yn* are a list of Manhattan vertices. Structurally, wires serve to connect devices and cells together. Crossing two wires on the same interconnection layer provides electrical connection while certain crossings such as poly over diffusion are illegal and result in an error message being generated in a subsequent process. This is as a result of allowing explicit devices, as defined by the **device** construct. Wires on different layers that intersect at a grid point are only connected if a valid contact is also found at that grid point. Thus wires serve to interconnect coordinodes. Physically, a wire is defined by its width **w**. If no width is given, the default minimum for the interconnection layer is used. Symbolic wires have no absolute length. Their final physical length depends on the compaction process (see Section VI).

3.3 Contacts

A contact joins two or more layers at a grid coordinate. For example, a device may be connected to a wire via the appropriate contact. Specification is as follows:

contact *type xpos ypos*

The range of contacts available depends on the particular process. In CMOS, contacts are of four types: **md**, **mp**, **vdd**, or **vss** which are respectively metal-diffusion, metal-polysilicon, and two substrate contacts. In NMOS depletion load the contacts may be **md**, **mp**, **pd**, or **pmd** for metal-diffusion, metal-polysilicon, poly-diffusion, or poly-metal-diffusion.

3.4 Pins

A pin may be on any interconnection layer and serves to name interconnection points and name specific nets within a circuit. A pin is specified by

pin *type xpos ypos name,*

where *type* may be,

a	aluminum,
p	polysilicon,
N	N-diffusion,
P	P-diffusion.

xpos ypos are the symbolic grid coordinates and *name* is the pin name. Although pins have no final geometric significance when it comes to generating mask data, they are of central importance in specifying the circuit corresponding to the physical description. They in essence merge the physical, structural, and behavioral characteristics of a circuit.

3.5 Cell instances

With the above-mentioned four types of element, an MOS circuit may be specified. Hierarchy is added by the cell instance element.

Other cells may be included in a cell by using the following statement:

instance *cellname* *xpos* *ypos* <**n** = *rep*> <**dx** = *xx*> <**dy** = *yy*> <"*instname*">

where,

cellname is the name of the cell instanced,
xpos *ypos* is the symbolic grid location of the origin of the instanced cell,
rep is an optional number of repetitions (default = 1),
xx is the optional *x* grid displacement/repetition,
yy is the optional *y* grid displacement/repetition,
instname is the particular name denoting a particular instance of a cell.

3.6 Graphical representations

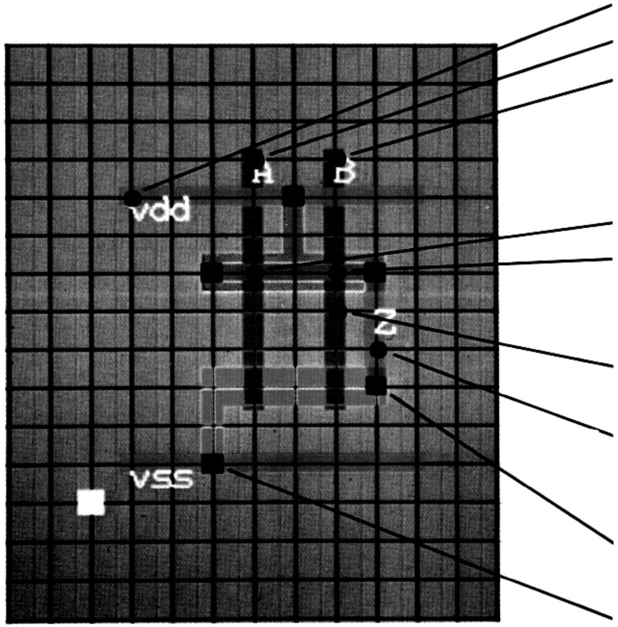
As well as the textual descriptions of ICDL, graphical descriptions exist for interactive editing and documentation. Figure 3 gives an example of the symbols used for interactive editing and the associated ICDL text. Note that these symbols mirror the width of wires and the size of different devices, although not necessarily to scale. The object of the representation is to give the designer some idea of *relative* thickness or size to aid in placement of circuit elements. This "thick sticks" or "logs" notation has been found to be more suitable for the symbolic design of ICs than some of the more conventional sticks notations. If one designs a topology using only stick figures, then when the geometrical aspects are taken into account, these neat topologies are often blown apart. The aim in this system is to provide as much support as possible for the designer to structure his design from the start, by taking into account in a symbolic manner the relative geometries of elements.

Symbols used for hardcopy documentation are illustrated in Fig. 4. In addition, a set of characters have been designated so that character terminals may plot layouts in the same manner as coarse-grid symbolic layout systems.

IV. DESIGN METHODOLOGY

4.1 Chip design

The chip design methodology that has been used with this system to date may be described as a top-down plan followed by a bottom-up implementation. The designer starts with the function or functions necessary and completes a chip "floor plan" as illustrated in Fig. 5. This combines a partitioning of the functionality along with some global wiring strategy. In this example of a data path, data runs horizontally in aluminum and control runs vertically in polysilicon.



```
begin   nand 2
      pin  al      1   1   vss
      pin  al      1   8   vdd
      pin  poly        4   9   A
      pin  poly        6   9   B
      pin  al      7   5   Z
      dev  n  or  =  1   4   3
      dev  n  or  =  1   6   3
      dev  p  or  =  1   4   6
      dev  p  or  =  1   6   6
      wire  al     1    1      8   1
      wire  al     1    8      8   8
      wire  poly        4   3      4   9
      wire  poly        6   3      6   9
      wire  al     3    6      7   6      7   3
      wire  Ndif        3   1      3   3
      wire  Pdif        5   6      5   8
      con  cut   7    3
      con  cut   7    6
      con  cut   3    1
      con  cut   3    6
      con  cut   5    8
end
```

Fig. 3—An ICDL circuit—graphical and textual descriptions.

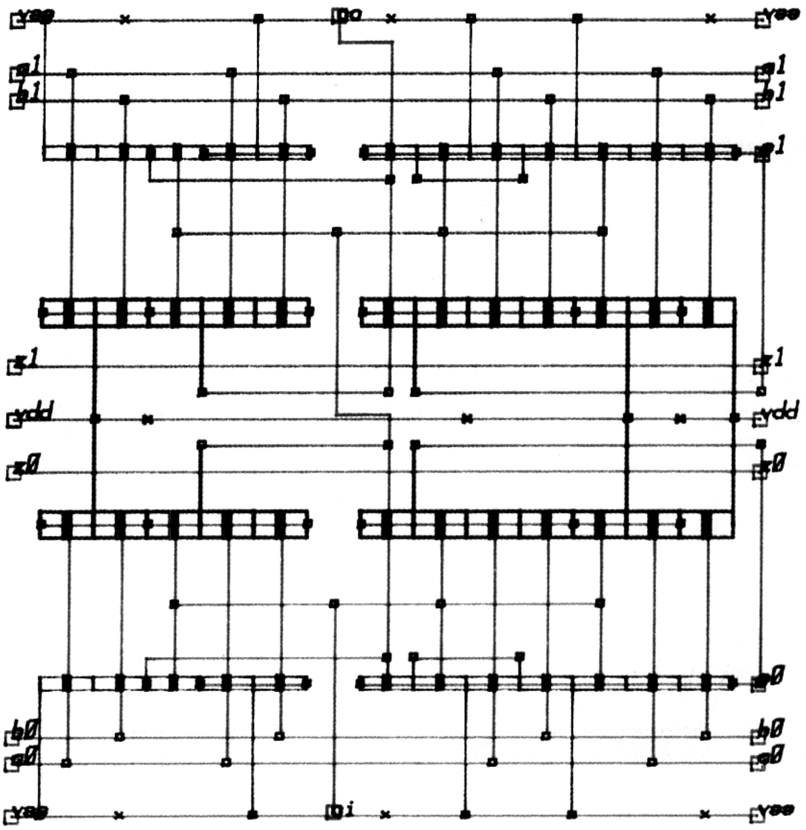


Fig. 4—ICDL output produced on plotter.

Once the overall wiring strategy is defined, the circuit blocks of the "floor plan" are decomposed until manageable cells have been defined. For instance, the Register ALU (RALU) block may be divided into a REGISTER section, CONTROL blocks, and an ALU section. The ALU may be further divided into an XOR, ZERO, ADDER, and SHIFTER section. At this point the ADDER is defined as a 2-bit section taking into account the original data and control directions and interconnection layers. This functional reduction is outlined in Fig. 5 and the resulting ADDER is shown in Fig. 6.

Cells are designed in this manner until the complete circuit block has been defined (RALU). The cell design is then iterated as explained below.

4.2 Cell design

The method of cell design used in this system is as follows:

(*i*) Cells are first entered into the system. This may be achieved by drawing a rough draft of a cell on squared paper and manually entering an ICDL description via a standard text editor. Alternatively, an interactive graphics editor may be used to edit ICDL files in an on-line

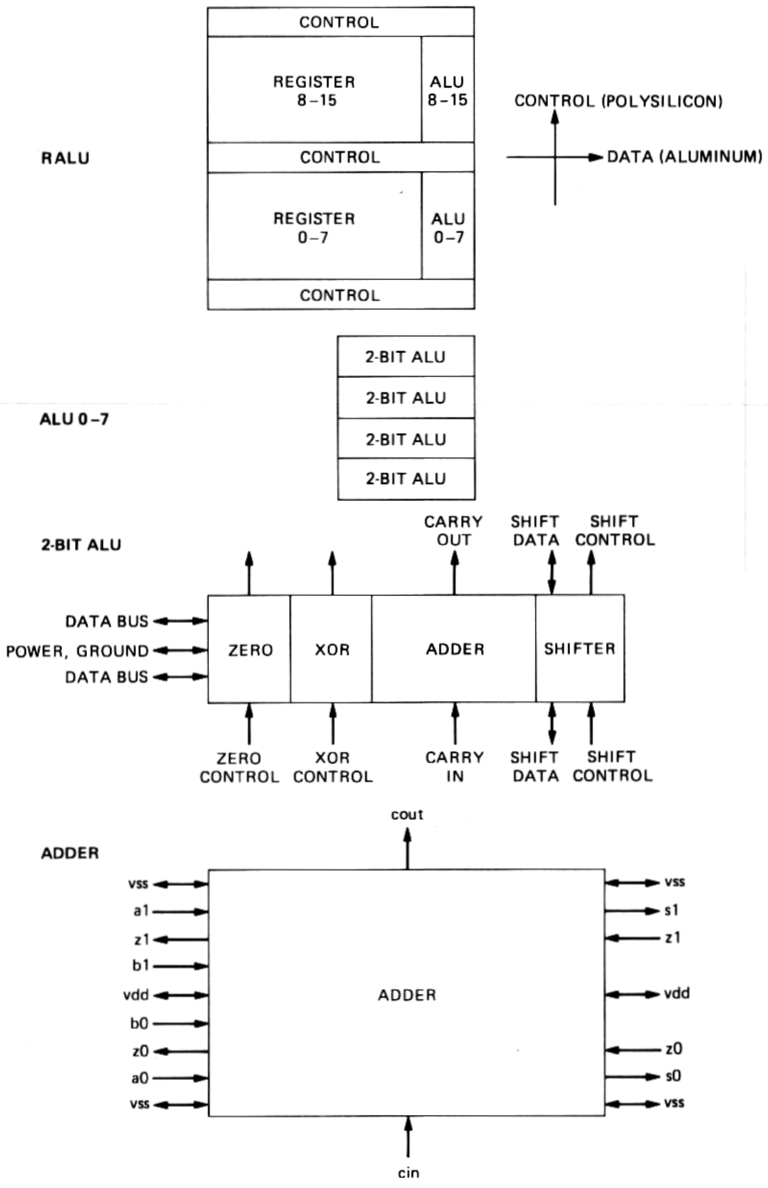


Fig. 5—Chip floor plan showing sample reduction technique to obtain a simple gate.

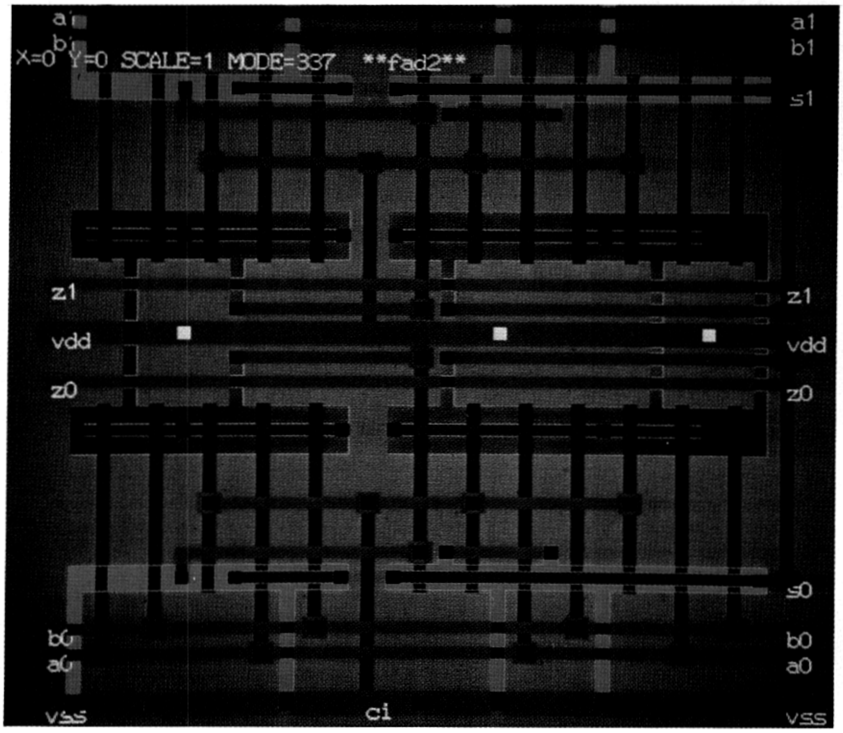


Fig. 6—A 2-bit CMOS adder shown on the color-graphics display.

fashion. Another method of input involves writing programs in the C language to procedurally define ICDL files.

(*ii*) The structural details of a cell may then be verified via a connectivity extraction program. Cells are re-edited at the symbolic level until the circuit is correct.

(*iii*) Cells so designed may then be compacted. This relates the grid positions used in the ICDL language to the actual coordinates allowed in the final mask set. Feedback is provided to the designer to enable him to optimize his design at the symbolic level with regard to packing density.

(*iv*) Behavioral aspects of a cell or group of cells may then be found by using this structural description with an in-built timing simulator. The connectivity extractor now uses the actual physical mask values to calculate parasitic capacitances and transistor sizes which are used by the simulator.

(*v*) Finally, various methods of expansion to the mask level are used to produce a final physical description of the circuit. If necessary, the cell may be re-edited at the symbolic level to improve packing or to interface with neighboring cells. Cells are incorporated into larger cells by a combination of repetition, butting, and simple routing.

The following sections describe the design aids needed to support this design style.

V. INTERACTIVE TEXT AND GRAPHICS EDITOR

Because the basis for the system is the ICDL language, we would like to be able to edit this language and see the effects of the edit operation on the layout. This requires a simultaneous text and graphics editor, which is relatively easy to implement with ICDL by a simple interpreter that works on a common data base.[14]

A design session begins by reading a cell into the editor. If a cell calls other cells, then they are read in order into the data base. If the design is such that all the cells cannot be read into memory, then cells are thrown out according to a least-recently used algorithm. As cells are read in, their bounding box dimensions are calculated and this information along with the cell name is kept in a global data structure which is maintained for the duration of the edit session.

The cell may then be viewed on a high-performance color graphic display.[15] All details may be plotted or just the bounding boxes of cells shown. The textual output corresponding to the layout displayed on the color display is presented on a standard text terminal. The designer may manipulate the cell description much in the same way that he would use a text editor, except that the actions are mirrored on the graphic display. For instance, if a particular line of text is displayed, then the element it represents is highlighted on the color screen. If the "delete element" command is given, then the element disappears from the screen and is deleted from the ICDL data structure. In addition to being able to perform operations via the terminal keyboard, a designer may use a data tablet for input and receive graphical feedback via the color display. For instance, an element may be identified by placing a tracking cross over the object and invoking the appropriate command. The editor then prints the line of text associated with the object. In the case of multiple objects occurring at the same grid point, each individual element is returned in sequence.

Representative commands are as follows:

view (**v**)
  plots the cell at current scale and *x,y* origin in current mode;
scale (**s**)
  changes the plotting scale;
append (**a**)
  places the editor into the append mode in which ICDL may be

entered at the keyboard or via a data tablet. Input via the data tablet is directed by a set of menus on the graphics display;

print (**p**)
prints the current line and highlights the element on the screen;

delete (**d**)
deletes the current line (which is highlighted on the screen). The element is removed from the screen;

deletearea (**D**)
deletes all elements in an area defined by keyboard or tablet;

.=
prints the value of the current line;

?
prints the position of the cursor;

move (**m**)
moves a rectangular section of ICDL by an arbitrary x and y displacement as specified by the tablet or keyboard;

copy (**c**)
makes a copy of a cell with specified translation, reflection and rotation;

box (**b**)
flips the editor between the "draw bounding box" and "draw detail" mode;

text (**t**)
toggles the text flag which enables pin names to be displayed;

identify (**i**)
identifies a particular element indicated by coordinates typed at the keyboard or by the tracking cursor and data tablet. If so desired, an identified element may be deleted;

Pan (**P**)
allows the designer to enter a mode which enables him to pan across a layout in real time in any direction. This is done with minimum change to the screen so that the designer does not lose visual perspective of the layout;

grid (**g**)
turns the grid flag on (or off). This causes a grid to be drawn when *view* is invoked with a grid spacing equal to current *scale* value;

write (**w**)
writes the cell to disk;

quit (**q**)
quits the editor.

Figures 7a and 7b show the RALU mentioned in Section 4.1 in symbolic format. Figure 7c shows the ADDER in XYMASK format. Figure 8 shows a section of the RALU at a reduced scale factor.

VI. COMPACTION

6.1 Introduction

Once a symbolic layout has been entered and checked as structurally correct, it is desirable to convert this symbolic description into a set of mask data. This is achieved by what is termed a compaction strategy. In general the symbolic description is examined in, say, the x direction and the minimum geometric spacing between elements or symbols is found. This is then repeated in the y direction. Akers developed a compaction algorithm based on a fixed-grid system.[16] In this algorithm blank grid sites are sought, say, in the y direction. Figure 9 illustrates a simple example. The path of blank cells may be disjoint along so-called shear lines as shown in Fig. 10. When a path is found across the layout, that space is taken out. Dunlop reported a system in which the symbolic input did not have to be placed on a coarse grid but instead used a *node-and-line* form of relative placement.[10] A similar approach is treated in Ref. 11. The reader is referred to these papers for an in-depth discussion of these techniques.

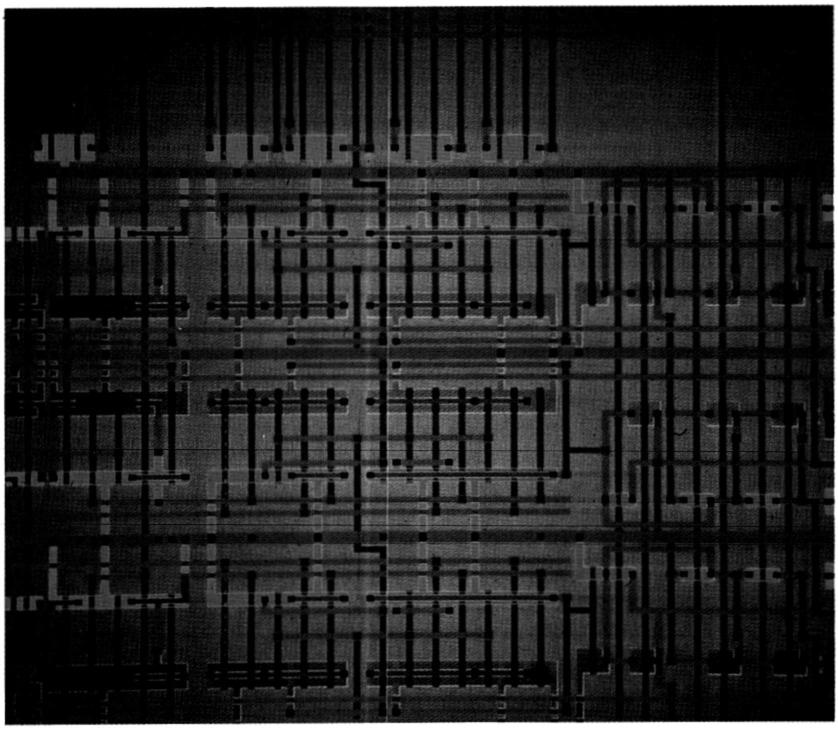


Fig. 7a—Part of the RALU in symbolic format scale = 2.

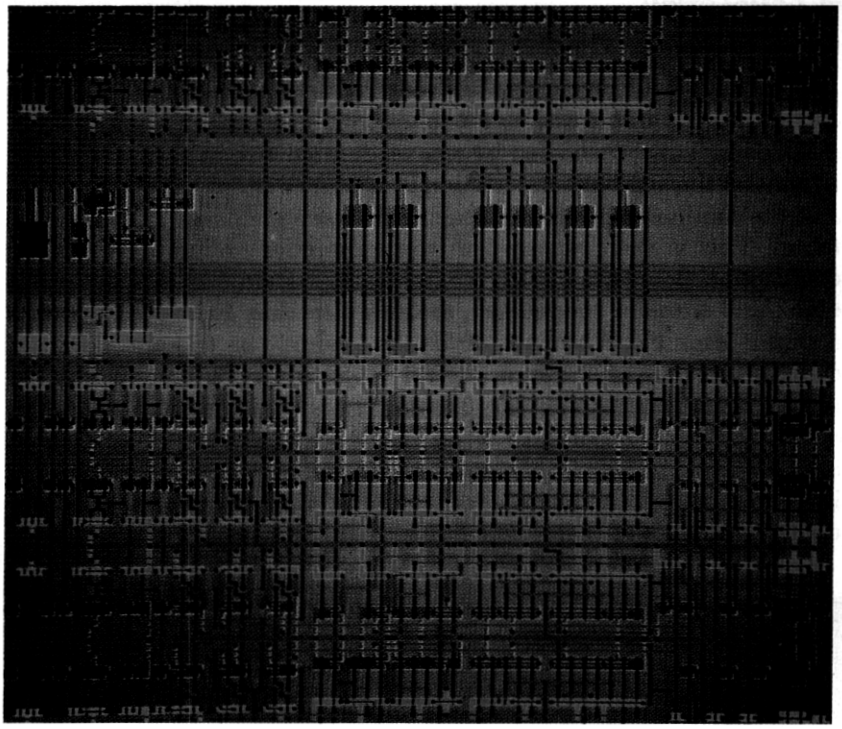


Fig. 7b—RALU in symbolic format scale = 4.

The compaction algorithm used in this system is conceptually very simple and is similar to all those mentioned. Primarily, the object of the algorithm was to be fast while producing adequate compaction. Very little optimization such as jog generation or automatic routing has been included. Rather, copious feedback is provided to the designer to enable optimization of a design interactively at the symbolic level. It is also reasoned that by providing the designer with informative and lucid descriptions (the color display), and a knowledge of compacter performance, a better symbolic design will be entered from the start. Practice has shown this to be a valid design method over the many cells that have been designed on the system.

6.2 Virtual-grid compaction

Recall that the data base consists of elements or objects such as transistors and wires that exist on a virtual grid. These two facts are used to form the basis of the compacter. Virtual-grid compaction takes adjacent grid lines and assigns mask coordinates to these grid lines

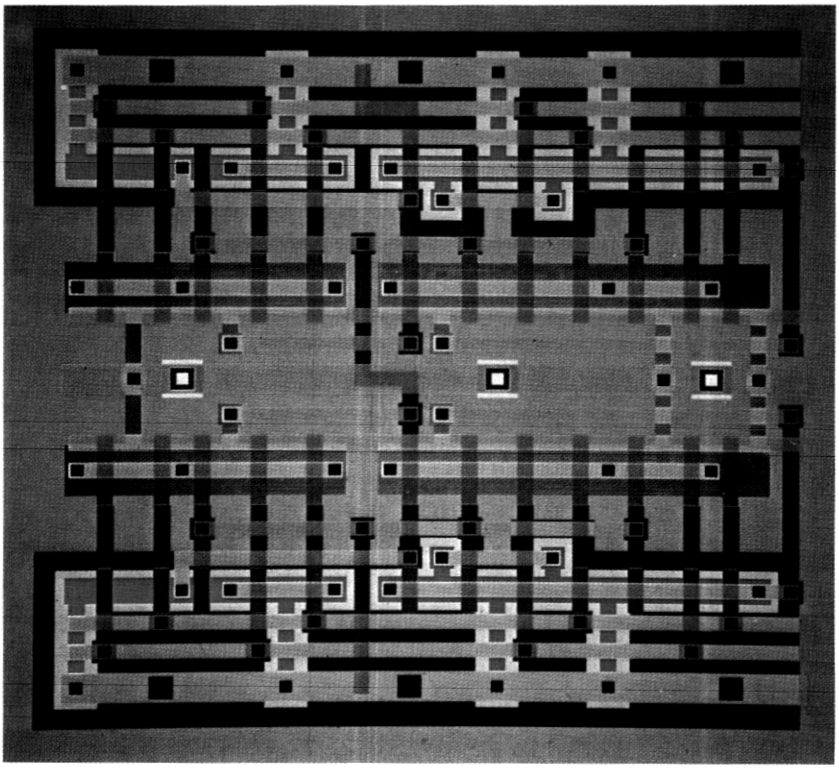


Fig. 7c—2-bit full ADDER in XYMASK format.

according to the relative placement of elements on the grid lines. Thus although elements exist on a grid from the input phase, the spacing of the grid is a function of the particular topology of the circuit being designed, and in fact its environment in the form of neighboring cells.

Elements are initially plotted into a matrix representing the area to be compacted. For instance, wires are plotted as a single line for wire segments parallel to the direction of compaction, and as endpoints otherwise. Transistors are normally plotted as three elements, one each for drain, source, and gate. Pins are not plotted and contacts are plotted as one element. The data structure at each element in the matrix is shown in Fig. 11. Thus in this implementation, at most one device, two wires, and one contact may be recorded at one element. In the case of a conflict the largest of any conflicting element is recorded. From the pointers to the ICDL data structure available at each element, it is relatively easy to calculate not only physical dimensions but also to deduce connectivity. Note that while the data exists on a grid, the

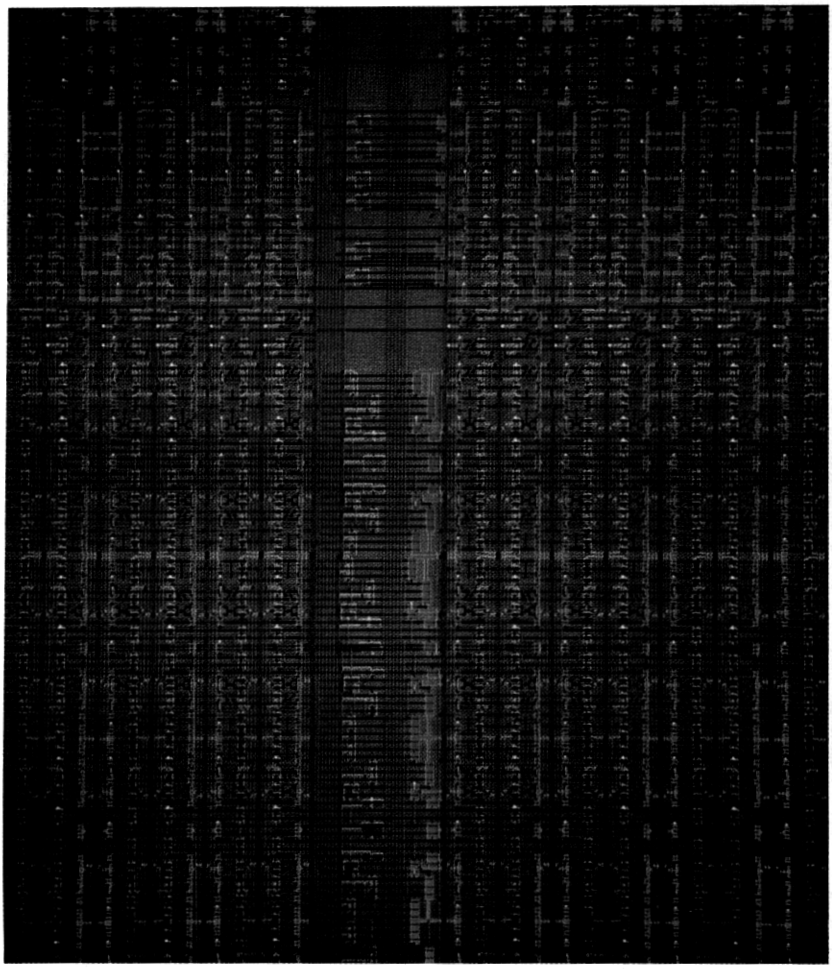


Fig. 8—RALU in symbolic format scale = 8.

grid has no specific geometric relationship to other grid locations until the compaction is concluded.

Having plotted the matrix, it is scanned columnwise, comparing adjacent elements. The worst-case element spacing for a given column is recorded. The mask coordinate corresponding to the current symbolic grid location is this value plus the previous mask grid coordinate. In addition to comparing adjacent columns, the x compaction backtracks to previous columns until the distance between the current column and prior column exceeds some worst-case value. In this way spacings that exert their influence over a number of symbolic grid locations are discovered. As the array is scanned, the position of the dominating interference element is monitored and stored for future reference.

When the x compaction is completed, the process is repeated for the y direction. However, in addition to backtracking, an arc is swept out from the leading element edge to account for oblique design rule violations. This effect is shown in Fig. 12.

On completion of x and y compaction, the correspondence between the symbolic grid location and the minimum mask coordinate allowed are stored in a "design grid" file. As will be demonstrated this is subsequently used with the original ICDL description to create a valid mask description.

6.3 Performance

The performance of the algorithm is plotted in Fig. 13. Note that the algorithm has essentially a linear execution time with respect to

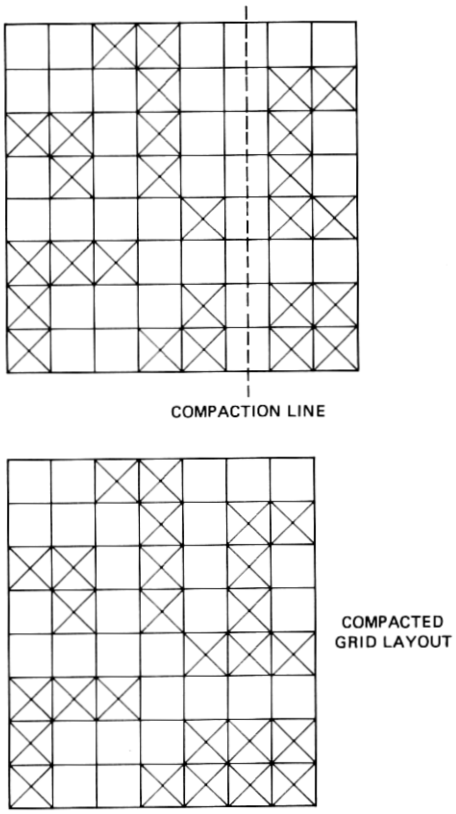


Fig. 9—Simple example of compaction on a coarse grid.

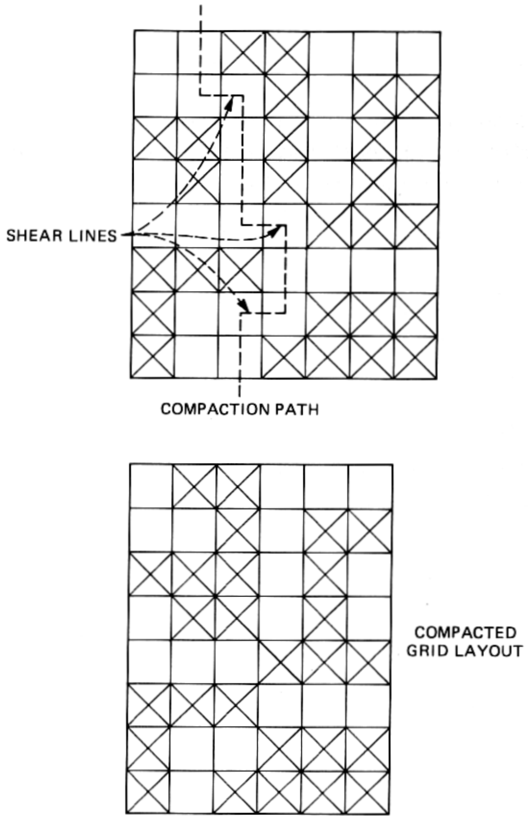


Fig. 10—Shear lines on compacted coarse grid.

the area (in grid units) of a cell. Providing that the density of cells is comparable this is equivalent to a linear dependence with respect to the number of elements. In contrast, both approaches in Refs. 10 and 11 indicate complexity of $N^{1.5}$. Hence, this approach remains tractable for large cells.

VII. MASK CONVERSION

As mentioned, the result of the compaction process is a design grid file, which indicates the relationship between symbolic x or y coordinates and final mask coordinates. Note that the design grid file designates the minimum spacing of mask grid points.

By segmenting the tasks as described above however, more useful methods of cell expansion than direct expansion may be employed. For example, Fig. 14 shows an example of a cell composed of subcells A, B, C, and D. "Pseudocells" AB, CD, AC, and BD are constructed. The compaction process is then carried out on each of the pseudocells. Cell

A is then expanded using the y design grid file of pseudocell AB and the x design grid file of cell AC. Cell A has now been pitch matched by virtue of the compaction process to cells B and C which it abuts. Any interconnection point matching in the symbolic description automatically matches in the geometric domain. The other cells are expanded in a similar fashion. This symbolic pitch matching may be attained in this system either procedurally or by visual inspection. The speed of the compacter is important here, as quite often large bit slices have to be compacted.

The second technique was evolved to deal with NMOS circuits, which tend to be less regular than CMOS circuits. Each cell is first compacted individually. Following this, the connection points and direction of connection of adjacent cells are specified in a file. A program adjusts the individual design grid files of common cells to match at the connection points. In the NMOS circuit examples, this saved from 10 percent to 20 percent in area, and will form the basis for future work. Another refinement may include allowing connection points to wander along a connection boundary.

Cells that are only defined at the mask level may also be merged using this technique. Such cells have fixed design grid files and have

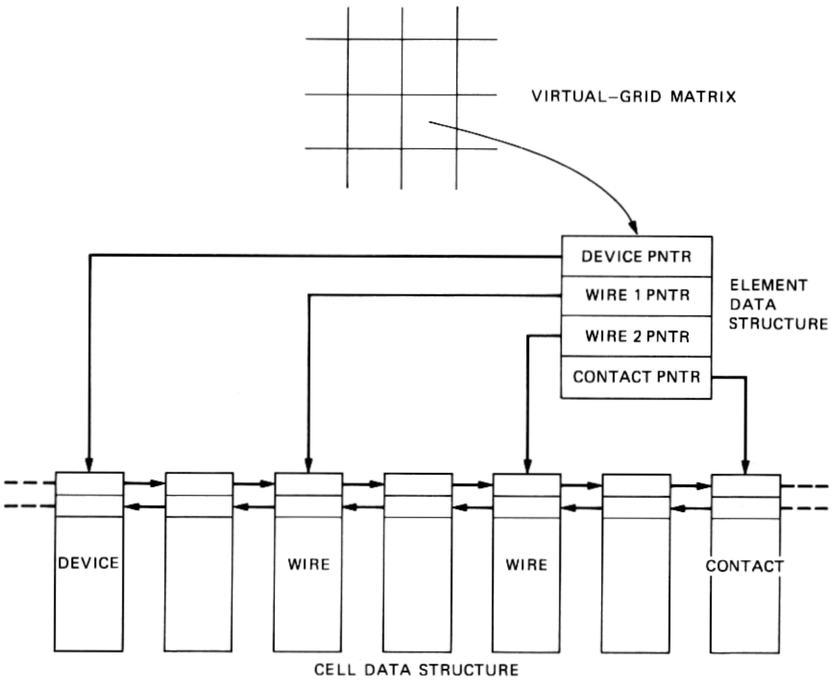


Fig. 11—Virtual-grid data structure.

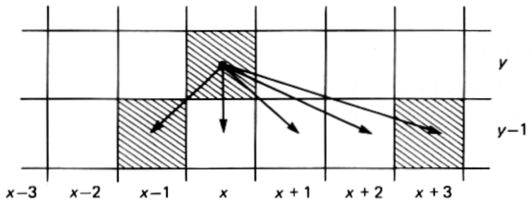


Fig. 12—Oblique violation checking. Arrows show grid points that may be checked during *y* compaction.

corresponding ICDL files which contain only pin information for interconnection purposes. ICDL cells may be pitch matched to these fixed cells or if necessary a routing channel may be inserted to interconnect them. An example of such a cell is an I/O pad from a standard cell library.

On completion of mask conversion, the output may be viewed in a mask level version of the interactive graphics editor. The same panning abilities are provided. In addition, feedback is provided at the symbolic level showing the critical compaction points in the design. The designer may use this graphical feedback to guide a redesign of the cell at the symbolic level. One advantage of the compacter is its predictable actions which may be used to advantage by the designer to obtain more compact mask layouts.

VIII. PROCEDURAL DESIGN

Situations arise during the course of a design in which it is desirable to procedurally define a design. This may include the parameterization

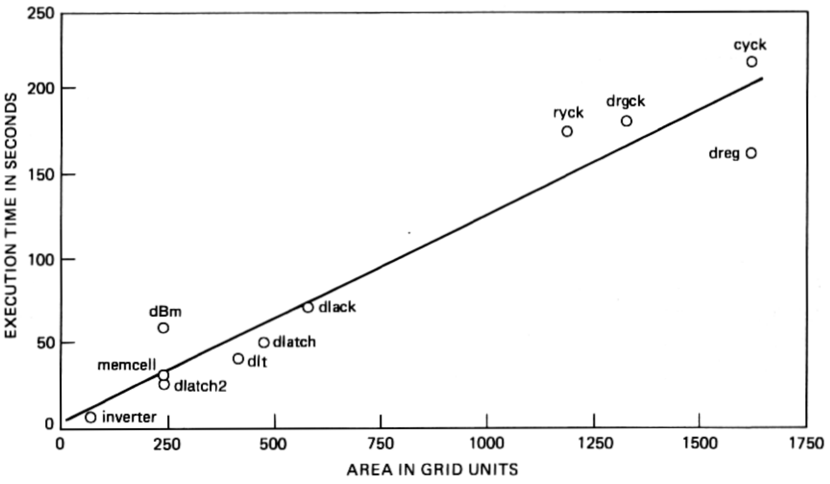


Fig. 13—Compaction algorithm performance.

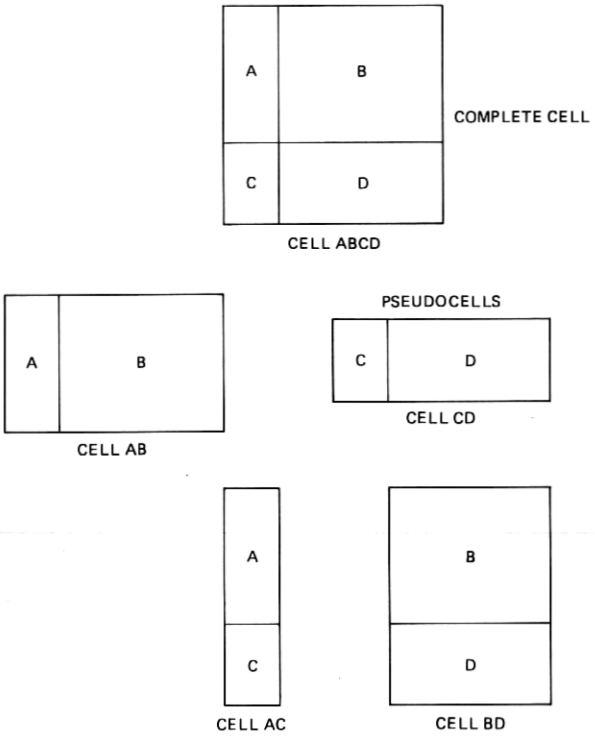


Fig. 14—Mask conversion for butting cells. Subcell A can be expanded using x design grid file of cell AC and y design grid file of cell AB.

of a cell, or the algorithmic specification of cells using looping and conditionals. A higher level language is also useful when it becomes necessary to put large pieces of circuitry together. To obtain this power, a set of C programs were written which generate an ICDL data structure, and may be used to procedurally define ICs.

In brief, subroutines are available to initialize the data structure, to insert devices, wires, contacts, and pins, and to interconnect cells. The subroutines allow the naming of transistor terminals and the subsequent interconnection to these named terminals. As an alternative to this method of procedural design, a user may just use parameterized print statements to generate ICDL files. Of course this method may be programmed in any higher-level language.

IX. CIRCUIT EXTRACTION

One of the benefits of using ICDL is that it carries an implicit circuit-connectivity description of a layout that has been entered either interactively or procedurally. As explained in Section III, devices have designated connection points which are related by simple geometric rules to the center of a device. Wires serve to connect devices and external connections via interlayer contacts. The pin construct aids the designer in naming specific nodes and connection points. In contrast to circuit extraction programs which work at the mask level, the inherent connectivity in the ICDL description is used to arrive at a transistor node table which does not have to infer the existence of transistors. In addition, values for parasitic capacitance are computed.

The algorithm used is quite simple and is illustrated to indicate the ease with which this may be achieved at the symbolic level by using a representation such as ICDL.

Following input of the ICDL, each pin is given a different node number. Each contact is also given a node number and then each transistor connection not covered by a pin or contact grid position is given a node number. In effect then, each virtual grid position that is occupied by a contact, pin, or transistor has a different node number. These form the coordinodes of the circuit.

Each wire is then taken in sequence and tested for intersection with wires of similar type. At the same time wires on different layers are tested for intersection via interlayer contacts. As each wire is tested a list of nodes that it crosses is maintained. At the end of this step each wire has a record of all wires that connect to it either directly or via valid contacts and a list of coodinodes that it effectively intersects.

A recursive technique is then used to reduce the node numbers common to a particular wire, to one value. All pin, contact, wire, and device node values are adjusted accordingly. For each device the node table is then printed out, with the appropriate pin names used where possible. Unnamed nodes are given an internally generated name.

Wire capacitance values for each named node are then summarized. Mapping from symbolic to mask coordinates may take place by either previously compacting the ICDL description and using values from the design grid file or using a statistically averaged grid spacing value.

The output of the circuit extractor may be piped through various filters to suit a range of circuit analysis and simulation programs. An example of a form suitable for input into a circuit analysis program is shown in Fig. 15.

9.1 Simulation

Once the physical and structural attributes of a cell have been defined, it is necessary to obtain some idea of the behavioral characteristics. To achieve this, a transistor and gate level simulator was written and installed on the MULGA system. It is modeled on a MOS timing simulator (MOTIS) and enables quick turnaround simulation of cells via the circuit extractor.[17] Simulation may take place at the transistor level or transistors may be merged into gate descriptions in

```
•SUBCKT   fad  (  vss   vdd   a   b   _sum   ci   _co  )
MN1   I0   a   vss   vss   N7A
MN2   _co   b   I0   vss   N7A
MN3   I1   ci   _co   vss   N7A
MN4   I1   a   vss   vss   N7A
MN5   _co   b   vss   vss   N7A
MN6   I2   _co   _sum   vss   N7A
MN7   I2   ci   vss   vss   N7A
MN8   I2   a   vss   vss   N7A
MN9   I2   b   vss   vss   N7A
MN10   I3   ci   vss   vss   N7A
MN11   I4   a   I3   vss   N7A
MP1   _co   a   vdd   vdd   P14A
MP2   _co   b   vdd   vdd   P14A
MP3   _co   ci   vdd   vdd   P14A
MP4   I5   a   vdd   vdd   P14A
MP5   _co   b   I5   vdd   P14A
MP6   _sum   _co   I6   vdd   P14A
MP7   I7   ci   _sum   vdd   P14A
MP8   I8   a   I7   vdd   P14A
MP9   I6   b   I8   vdd   P14A
MP10   I6   ci   vdd   vdd   P14A
MP11   I6   a   vdd   vdd   P14A
MN12   _sum   b   I4   vss   N7A
MP12   I6   b   vdd   vdd   P14A
CMvss  vss  0  CMTOSH  324
CNTvss  vss  0  CN + H  48
CMvdd  vdd  0  CMTOSH  324
CPTvdd  vdd  0  CP + H  48
CPa  a  0  CPTOSH  168
CMa  a  0  CMTOSH  162
CPb  b  0  CPTOSH  168
CMb  b  0  CMTOSH  162
CP_sum  _sum  0  CPTOSH  30
CM_sum  _sum  0  CMTOSH  168
CPT_sum  _sum  0  CP + H  12
CPci  ci  0  CPTOSH  150
CMci  ci  0  CMTOSH  90
CP_co  _co  0  CPTOSH  54
CM_co  _co  0  CMTOSH  138
CMI2  I2  0  CMTOSH  24
CNTI2  I2  0  CN + H  12
CMI6  I6  0  CMTOSH  72
•FINIS
```

Fig. 15—Circuit extractor output for circuit-analysis program.

the manner that MOTIS operates. A bonafide transmission gate may be modeled and buses are handled inherently by the data structures used. This simulator is called EMU for emulation of MOS circuits on the *UNIX* system.[18]

A filter on the circuit extractor output allows the cell's logic description to be specified in terms of a C subroutine. These subroutines may be then combined hierarchically to suit the high-level functions that the cells constitute. This description may then be compiled to provide an intermediate simulator code which is input to the simulator. A full discussion of the simulator may be found in Ref. 18.

Output from the simulator may be viewed on the color CRT or plotted on a four-color plotter for hard-copy purposes. Typical output is shown in Fig. 16.

The integration of the simulator into the design station is regarded as very important as it allows a designer the same fast feedback

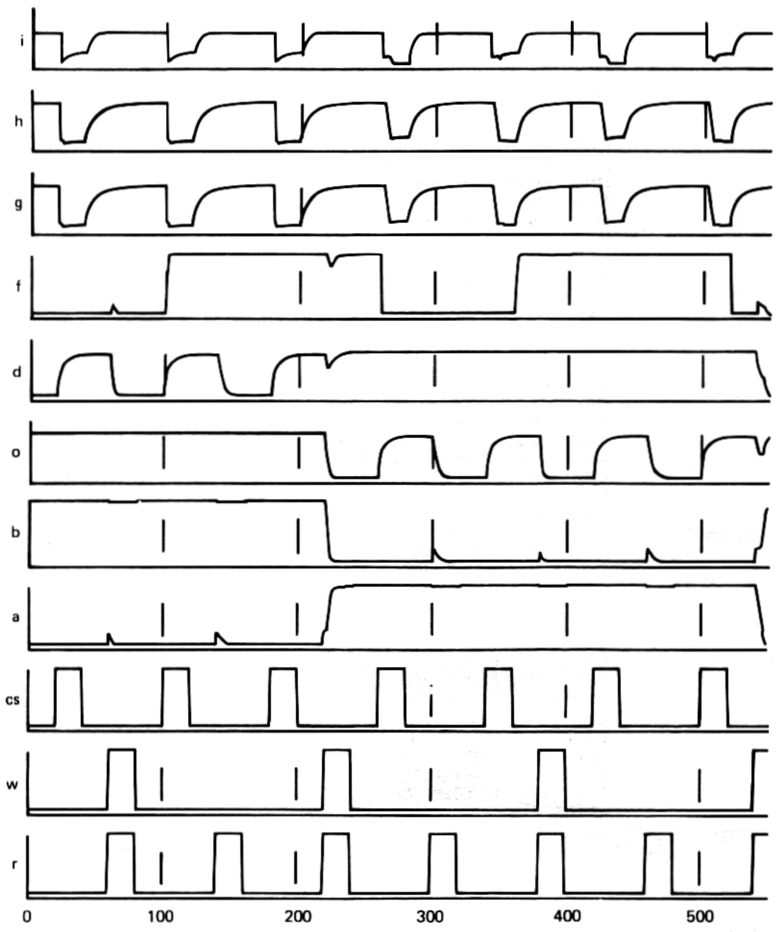


Fig. 16—Typical simulator output.

regarding behavioral aspects of a design, as are available physically and structurally.

X. HARDWARE

10.1 General

Aspects of the hardware have been alluded to during the course of this paper. Figure 17 is a sketch of the hardware used in the design station.

The general purpose host is a DEC LSI 11/23,* with 256 K bytes of

* DEC is a trademark of Digital Equipment Corporation.

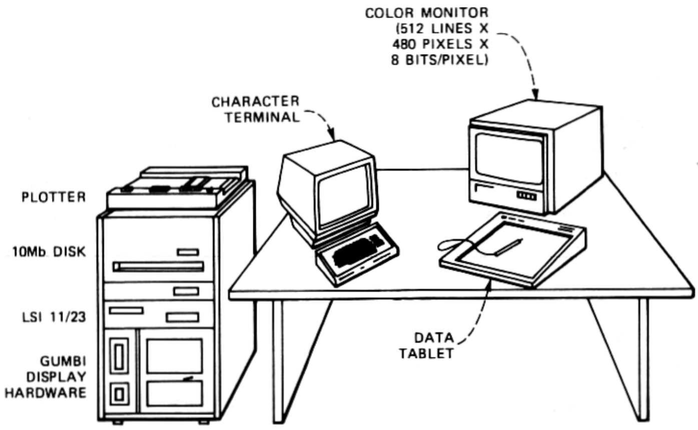


Fig. 17—Hardware used in the design station.

semiconductor RAM. A CDC "Hawk" disc provides 10 M bytes of secondary storage.* Serial ports support a printer, glass TTY, dial-up lines and a four-color plotter. A serial link is also provided to a DEC VAX 11/780 for backup storage and large numeric processing tasks. The *UNIX* multiuser operating system controls all devices.

On the interactive side, an 11-inch data tablet is used in conjunction with a high performance color display. The color display contains a 512 pixel/line by 512 line (480 viewable) frame buffer with 8 bits per pixel. A color map provides 256 colors from a palette of 2^{24}. A 16-bit microprogrammed graphics processor (GUMBI) is responsible for raster conversion into the frame memory.[15] The total cost of the hardware is in the vicinity of 50 thousand dollars, but near term reductions in cost would lead to a total cost in the range 20 to 30 thousand dollars.

10.2 Display attributes

Many of the human-engineered features of the editor rely heavily on the display terminal used, and it is worthwhile to review some of the requirements that have been necessary to obtain rapid response times.

10.2.1 Speed of display

This attribute is important in an interactive situation to reduce operator fatigue and boredom and provide a fluid man-machine dialog. Speed is achieved in two ways:

(*i*) A high-speed interpretive graphics processor.

* CDC is a trademark of Control Data Corporation.

(*ii*) Information bandwidth matching between host and graphics processor.

The first mentioned item is satisfied by a high-speed microprogrammed graphics processor called GUMBI.[15] GUMBI takes commands such as draw box from a host processor and is responsible for plotting the required bit patterns into a frame-store memory. This offloading of menial processing means that the host processor may concentrate on data-manipulation operations rather than time consuming bit-manipulation tasks.

Information bandwidth matching is achieved by:

(*i*) Efficient prefiltering of data to be plotted.

(*ii*) Direct interpretation of the ICDL data structure.

(*iii*) Optimization of the *UNIX* operating system DMA driver.

It is aided by resolving all plotting operations into two steps. First, a symbolic coordinate window of interest is maintained. This is the coordinate representation of the data base and only simple translations need be made to determine actual positions of elements in symbolic coordinates. Second, a screen coordinate window is kept to determine clipped shapes that must be passed to the display processor. As the hierarchical data base is transversed, the bounding box of a cell is first tested against the symbolic window. If it is outside the window, the next cell is considered. Alternatively, the elements within a successful cell are tested against the symbolic window. The successful candidates of this check are transformed into screen coordinates and then clipped against the screen window. Rectangles are then passed to a buffer which is transferred via a DMA interface to the graphics processor. Note that the "display list" for this process is in fact the ICDL data structure, alleviating the need for maintaining intermediate graphic data structures. The normal DMA driver for the *UNIX* operating system was modified to allow direct transfer of data from "user" space to the DMA port. In addition, once a DMA transfer has been initiated the "kernel" process returns rather than waiting for the termination of the DMA transfer as is normal. This change alone improved the speed of data transfer by 10:1.

10.2.2 Selective update

Because a refresh raster scan display is being used as the display medium, alterations to the screen do not necessitate the screen being erased and replotted as is the case for a storage display. Thus in a "delete" operation only the element deleted is removed without disturbing the rest of the screen. In actual fact, the operation first deletes the element in question and then replots the cell for the symbolic window defined by the deleted element. This alleviates any problem of partial erasure of remaining elements which can occur if the removed element shares screen area with another element of the same color.

The "pan" operation uses this attribute and the ability to scroll the complete screen to give the designer the ability to move across an entire layout as if flying over the final chip in an airplane at low altitude. This operation is summarized in Fig. 18 for moving the display to the left, although any direction movement is possible. First, the display frame memories are scrolled to the left by 16 pixels and a 16-pixel strip is erased at the right vertical border. The symbolic window is adjusted to include elements that will fall within this right border and the data base is searched. Valid elements are then passed to the screen transformation, clipper, and then to the graphics processor. Results have shown that even complete chip layouts may be scrolled in periods commensurate with human response times (100 mS to 1 S).

10.2.3 Color display

The dimension of color is invaluable to the designer. First, all the interconnection layers are shown in different colors and the appropriate cross-overs are colored to indicate transparency or any other subjective effect which enhances designer feedback. Different contacts

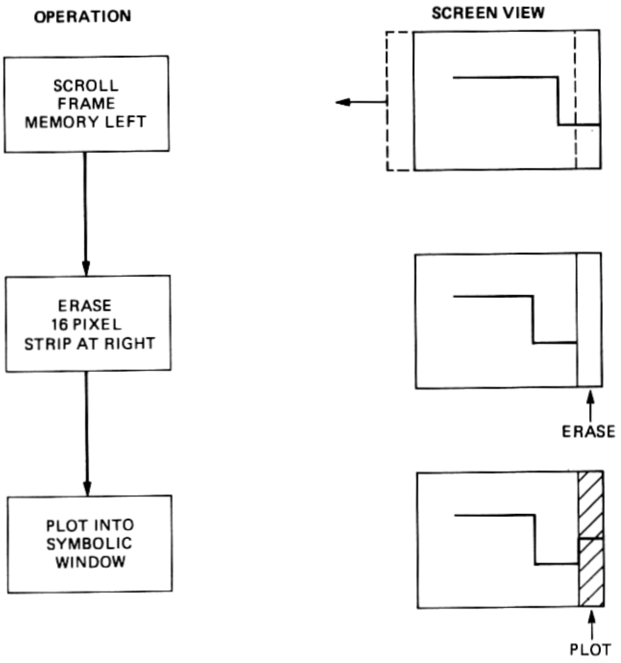


Fig. 18—Flow diagram of the panning operation.

are also coded by their color. Illegal overlaps are designated by eye-catching colors not used in other parts of the layout. In general, a designer may choose the color that he associates with a given layer. This tends to reinforce any recognition tasks that occur. The use of color is also an immense learning aid for novice designers. By giving designers a few rules such as "don't cross green over blue," experience has shown that it takes only a brief familiarization time for candidates to produce useful circuits. Of course, highlighting particular areas of the circuit is trivially accomplished. Even at large-scale factors, when plotting a complete chip the color display yields more information regarding relative wiring densities and placement than storage displays with nominally higher-spatial resolutions. The main reason for this is, despite a lower-spatial resolution (512*480 vs 4096*3000) the higher-color resolution (256 colors) more than compensates. A simple test to show this consists of plotting a monochrome and color layout side by side and reducing the scale until the circuit in each respective case is an unintelligible "blob." Results indicate a factor of four improvement for the color layout.

The use of solid boxes to represent elements alleviates any confusion as to the interior of a box in congested areas. Thus, in conjunction with the "pan" command, a wire may be followed across an entire layout without the designer losing visual perspective, as is the case in more conventional displays.

XI. RESULTS

After using the system, a number of trends in design may be identified. Prior to having the data tablet working, input was primarily via keyboard. At this stage C procedures were used frequently to define circuits. Subsequently, when the data tablet was operational, all cell design was completed at the color terminal with higher-level chip building functions handled by C procedures and *UNIX* operating system shell scripts. In fact, many cells were entered from scratch at the screen, rather than planning a version on paper first. The author has tried interactive input when designing at the mask level and found it to be unsatisfactory without prior planning. However, it would seem that the reduction in complexity afforded by the symbolic notation, and the smooth man-machine interface provided by the editor and color graphics display, yield a truly interactive environment. This leads one to try a number of variations of cells for a particular situation exploring performance and global interface requirements.

The data path of a special purpose 16-bit CMOS processor is shown in Fig. 19. This chip was designed on the MULGA system in 8 man-weeks (without tablet). It contains 5000 transistors in a highly stylized layout modeled on the gate-matrix style.

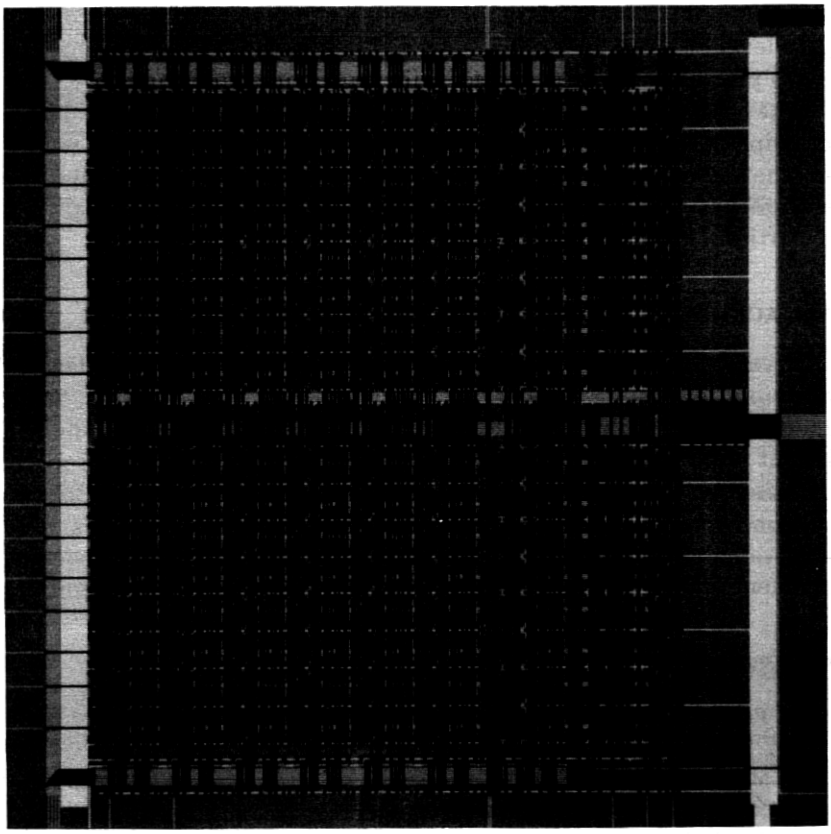


Fig. 19—Five-thousand-transistor CMOS circuit designed with the MULGA system.

In addition to the CMOS chip, two smaller NMOS chips have been designed, each comprising approximately 1000 transistors. These chips were designed in 3 man-weeks.

XII. CONCLUSIONS

We have presented a practical self-contained system to design integrated circuits in the symbolic domain. It is based on a hierarchical circuit description language (ICDL), which describes IC subcircuits in both geometric (physical) and circuit (structural) domains.

The following software supports design at the symbolic level:

(*i*) an interactive text and graphics editor based on a high-performance color display,

(*ii*) a fast compaction procedure based on the virtual-grid technique,

(*iii*) a circuit extraction and parasitic audit program, (*iv*) an inbuilt timing stimulator which works at both transistor and gate level,

(*v*) procedural design in the C programming language.

The efficacy of this system has been demonstrated during the design of chips in both CMOS and NMOS technologies. A predominant reason for the success of the system is the friendly interactive nature that the system provides to designers with a wide range of experience and expertise.

XIII. ACKNOWLEDGMENTS

Bryan Ackland provided many fruitful discussions and ideas in addition to valuable feedback as the first "user" of this system. Steve Law and Alex Lopez provided an introduction to the gate-matrix layout style and an insight into CMOS design techniques. The many discussions with Irene Buchanan and participants of the Caltech Silicon Structures Project about structured design techniques were also valuable to this work. W. H. Ninke provided necessary encouragement and critical input during the development of the system.

REFERENCES

1. G. Persky, D. N. Deutsch, and D. G. Schweikert, "LTX—A Minicomputer-Based System for Automated LSI Layout," J. Design Automat. Fault-Tolerant Comput. *1,* No. 3 (May 1977), pp. 217–55.
2. C. Mead and L. Conway, *Introduction to VLSI Systems*, Reading, Massachusetts: Addison-Wesley, 1980.
3. D. Gibson and S. Nance, "SLIC—Symbolic Layout of Integrated Circuits," Proc. 13th Design Automat. Conf., June 1976, pp. 434–40.
4. R. P. Larson, "Versatile Mask Generation Techniques For Customer Microelectronic Devices," Proc. 15th. Design Automat. Conf., June 1978, pp. 193–8.
5. D. Clary, R. Kirk, and S. Sapiro, "SIDS—A Symbolic Interactive Design System," Proc. 17th Design Automat. Conf., June 1980, pp. 292–5.
6. B. Infante et al., "An Interactive Graphics System for the Design of Integrated Circuits," Proc. 15th Design Automat. Conf., June 1978, pp. 182–7.
7. A. D. Lopez and H. F. Law, "A Dense Gate Matrix Layout Style for MOS LSI," Proc. ISSSC, February 1980.
8. J. Williams, "STICKS—A Graphical Compiler for High Level LSI Design," Proc. 1978 NCC, May 1978, pp. 289–95.
9. A. Dunlop, "SLIP: Symbolic Layout of Integrated Circuits with Compaction," Comput. Aided Design, *10,* No. 6 (November 1978), pp. 387–91.
10. A. Dunlop, "SLIM—The Translation of Symbolic Layouts into Mask Data," Proc. 17th Design Automat. Conf., June 1980, pp. 595–602.
11. M. Y. Hsueh and D. O. Pederson, "Computer-Aided Layout of LSI Circuit Building Blocks," Proc. 1979 Int. Symp. Circuits and Systems, July 1979, pp. 474–77.
12. N. Weste, "Virtual Grid Symbolic Layout," Proc. 18th Design Automat. Conf., June 1981.
13. I. Buchanan, "Modelling and Verification in Structured Integrated Circuit Design," Ph.D. thesis, University of Edinburgh, Scotland, 1980.
14. S. Trimberger, "Combining Graphics and Layout in a Single Interactive System," oral presentation, 17th Design Automat. Conf., July 1980.
15. N. Weste and B. Ackland, "GUMBI—A Graphic User Microprogrammable Bit-Slice Interpreter," Proc. Compcon Fall, September 1979, pp. 232–7.
16. S. B. Akers, J. M. Geyer, and D. L. Roberts, "IC Mask Layout with a Single

Conductor Layer," Proc. 7th Design Automat. Workshop, San Francisco, 1970, pp. 7–16.
17. B. R. Chawla, H. K. Gummel, and P. Kozak, "MOTIS—An MOS Timing Simulator," IEEE Trans. Circuits Systems, *22,* No. 12 (December 1975), pp. 901–10.
18. B. Ackland and N. Weste, "Functional Verification in an Interactive Symbolic Design Environment," Proc. 1981 Caltech VLSI Conf., January 1981.