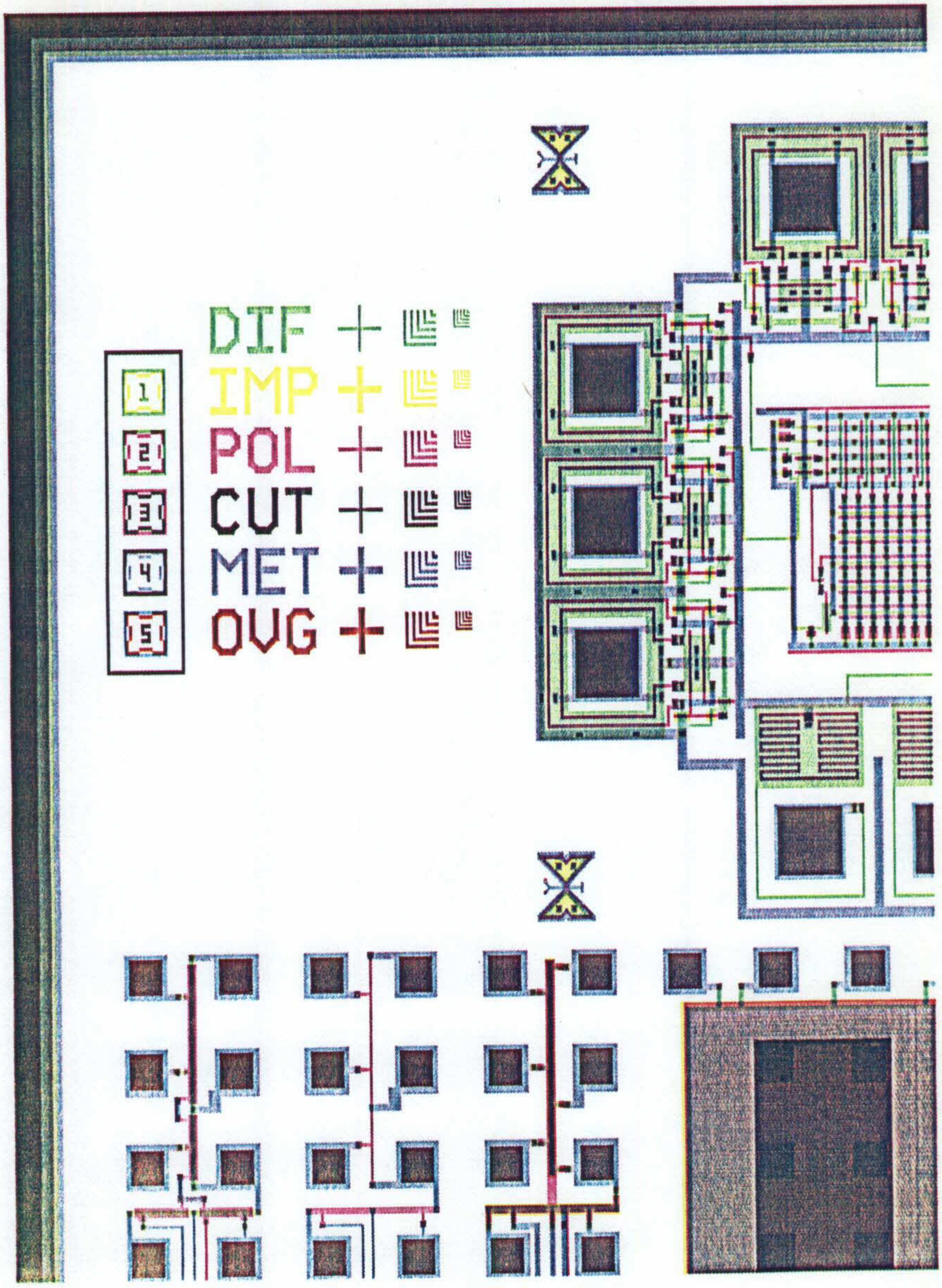
# A Guide to LSI Implementation

Robert Hon and Carlo Sequin

SEPTEMBER 27, 1978

XEROX
PALO ALTO RESEARCH CENTER
3333 Coyote Hill Road
Palo Alto, California 94304


1
DIF
2
IMP
3
POL
4
CUT
5
MET
OVG

## Table of Contents

## Foreword

This book was produced as a result of the 1978 Summer Program of the LSI Systems Area of Xerox Palo Alto Research Center (PARC). This program involved the participation of summer employees Bob Baldwin (undergraduate at MIT), Peter Dobrowolski (master's student at UC Berkeley), Bob Hon (Ph.D. student at Carnegie-Mellon University), and Steve Trimberger (Ph.D. student at UC Irvine). These students and our consultants Carlo Sequin (professor at UC Berkeley), Carver Mead (professor at Caltech), and Jim Rowson and Dave Johannsen (Ph.D. students at Caltech) have spent the summer working with us on various aspects of integrated systems design and implementation, including a multi-project chip and this document. Xerox employees who participated in the program include Rick Davies (General Sciences Lab), Maureen Stone (Advanced Systems Department), and our own group members Lynn Conway, Doug Fairbairn, Dick Lyon, and Wayne Wilner.

In spite of the difficulty of working on a short schedule with people of widely differing backgrounds, co-editors Bob Hon and Carlo Sequin have managed to produce this timely document, to help bring integrated system capability to a wide range of users, including universities. Bob Hon, the principal author, has collected considerable relevant information about current patterning and fabrication technologies, and prospects for future changes that readers should be aware of; he also personally carried the PARC summer-1978 multi-project chip from individual project designs through maskmaking, and arranged for subsequent processing.

The other three students coöperated in the writing of this book by contributing sections, and became involved in the multi-project chip by producing project designs and by helping with design tools. Bob Baldwin assisted in establishing a cell library, designing two projects, writing design conversion software, and documenting his work in sections of this book. Steve Trimberger is building our new design system, and has built some of our existing tools; he has provided the writeup on automated design systems, and designed a project with the aid of a hybrid of our graphical layout system and a general purpose programming language. Peter Dobrowolski designed and built some IC test hardware for us this summer, and wrote the book section which summarizes testing strategies; in addition, his chip project illustrates the use of PLA's for a novel bit-staggered timing scheme for fast arithmetic functions.

Rick Davies provided the section on process test patterns, documenting his project on the chip. Maureen Stone wrote a section on symbolic layout languages and the appendix on ICLIC. Within our own group, Wayne Wilner has described the problem of design rule checking, and Dick Lyon

has provided the section on circuit simulation.

Carver Mead provided important technical advice and encouragement. Lynn Conway (manager of our LSI Systems Area) provided the drive that it took to make this book happen. She and Carver have recently written the book *Introduction to VLSI Systems*, which is assumed to be available as a reference with this document.

Richard F. Lyon
Douglas G. Fairbairn

The authors would like to acknowledge the help of Dick Lyon, who spent considerable time working on this document as the final deadline approached.

Bob Hon
Carlo Sequin

## 1. The New IC Designer

Traditionally the design and development of integrated circuits (IC's) has been the domain of specialists with substantial training in this "art". With the emergence of more powerful computer aids and of reasonably standardized IC processing techniques, IC design can be simplified to the point where it becomes a routine engineering step in the development of a special purpose system. MOS devices are particularly simple and straightforward as long as one stays away from the smallest geometries feasible. With reasonable, relaxed design rules the performance of standard MOS circuit blocks such as inverters, pass gates, buffers, NOR gates and composites of these blocks become as predictable as TTL circuits. Using a structured design approach, such as the one promoted by Mead and Conway in *Introduction to VLSI Systems* [Mead 1978], it is possible for people with only a minimal understanding of the device physics of a MOS transistor to produce operational integrated circuits of substantial size. Thus, a systems designer can now sit in front of an interactive graphics terminal and produce the layouts of a set of masks for a special purpose integrated circuit. Such personalized IC's can greatly enhance the functionality of the system to be built or alternatively may dramatically reduce the total chip count for a system of given specifications.

In such a venture it is often not important to produce an integrated circuit of the highest layout density or of the highest performance, which could only be obtained by pushing the limits of present-day technology. Normally the main concern is to get a properly working chip with the shortest possible turnaround time. It is here that effective design tools and, even more importantly, the proper design methodology, are crucial. These issues are discussed in *Introduction to VLSI Systems*. The second, equally important part is to get the IC designs implemented. In an environment that is not already set up to produce custom-designed IC's as a routine step, the designer himself often has to be the driving force behind the implementation of the first few IC's. In this situation many months are often wasted because of unsuitable preparation or unavailability of the necessary information, leading to frustrating delays in the project schedules and to abandoning the custom-made IC approach altogether. These are problems that we hope this document will prevent.

Converting an integrated circuit design into a finished, packaged, and tested chip is more a time consuming task than a difficult one. Dozens of important details have to be observed to prevent disasters or costly delays. Up to this point a concise description of the specific details and necessary steps has not been available. Specific information had to be gathered from scattered sources including personal interviews with "old hands in the trade".

This document is intended to be a guide along the entire path from the design of the layout of the integrated circuit, through mask generation, IC wafer fabrication, and chip packaging to the testing of the finished circuits. Important decision points and potential pitfalls along the way are clearly spelled out.

Often several IC designs will be combined into a single multi-project chip. In this manner the cost of mask generation and wafer fabrication, as well as the organizational overhead involved in pushing the future IC through all critical stages can be shared among a larger group of people. In an academic or research environment this coordination of several experiments into one IC project is particularly important, so that not every student has to worry about all of the details of mask and wafer processing. Certain features such as test patterns to measure device performance, alignment marks, and chip separation lines (scribe lines) can be standardized and re-used in subsequent multi-project chips. Sticking to the same features, similar basic chip formats and established procedures to generate the multi-project chips will help to streamline this process and enhance the chance for satisfactory results. Someone, therefore, will have to act as a coordinator. His or her first task will be to merge the different files describing the various IC designs with the starting frame containing the mentioned standard features. He will then interact with the mask house and the fabrication line, making sure that both places have all the information that they need and that there is no misunderstanding in what they are expected to do. In order to avoid unnecessary delays he should constantly keep track of the state of the project and try to effect smooth interactions between the various parties involved. This includes hand-carrying the magnetic tape with the designs from the research site to the mask house, the set of working plates from the mask house to the fabrication line, finished wafers from the fab line to the dicing and bonding station, and finally a number of packaged chips to the testing area. This document describes the real-life problems and details which the coordinator must be aware of to effectively carry out these tasks. But even the occasional designer of an individual IC should be aware of the overall process, so that he or she may better understand certain implications on their own activity. As with any other system implementation technology, the types of design aids and the methods of fabrication impact the type and quality of design which is done. The most effective systems designers will therefore understand at least the basic aspects of design aids and checking tools as well as mask and wafer fabrication.

Chapters 2 through 6 outline the basic path from IC design to the finished product. Chapters 2, 3 and 6 should be studied even by people who never dream of becoming coordinators of multi-project chips, since they set the stage for proper IC design. For the coordinator chapters 4 and 5 are absolutely vital. Chapter 7 gives an example of a multi-project chip produced at Xerox PARC during the summer of 1978. It carried 10 experiments including a wide range of logic, arithmetic and memory circuits and a test pattern. Appendix A contains a listing of the instructions sent to the mask house concerning this particular chip.

Throughout the text *italics* are used to introduce vocabulary which the designer should know. No attempt is made to tabulate precise definitions of terms, but enough information can be inferred from the context that the reader can search for more details if necessary. Appendix F provides pointers to in-depth articles and texts covering particular aspects of IC implementation. Integrated circuit manufacturers can often provide valuable information and insight into most phases of IC implementation; Appendix B is a listing of some that we have dealt with. It is by no means exhaustive, and the reader should not hesitate to make his own contacts where possible.

# 2. IC Design Tools

The key to fast-turnaround integrated circuit design is the emergence of powerful tools which make it possible for the designer to be assisted by computers in an effective manner. To enter his ideas into the design system, suitable interactive graphics terminals have become available. As another option, layout languages are being developed which one day will provide a means to enter designs in a symbolic manner at a rather high level of abstraction. Displays and plots of the various mask levels of an IC are important to close the interactive loop between man and machine, and are indispensable in the final debugging phase. There are additional ways in which the computer can assist the designer. A "mechanical" check for design rule violations helps eliminate potential problem spots in the fabrication of IC's. Circuit and logic simulation can be used to predict the performance of critical parts of the circuit and to test the correctness and the timing on a larger scale, respectively. Further, the computer can be a tremendous help in the management of the large amount of information associated with an IC design.

This chapter provides an overview of the types of tools available and under development. It also tries to bring out the point that, with only little expense and effort, a minimal set of tools can be acquired that makes IC design possible.

## 2.1 Automated Design Aids

*[section contributed by Stephen Trimberger, UC Irvine]*

The role of a design aid is to reduce errors and design time. Therefore, a design aid should first attack tedious and error-prone aspects of design. In addition, the design aid should present the design to the designer so that he can catch errors. Rather than have a computer take over the entire design task, the design effort should be a cooperative one between the designer and the design aid, in which the design aid relieves the tedious and exacting chores, giving the designer more time to do what he does best: design.

### *2.1.1 Plotting*

A necessary design aid in any design environment is hardcopy output. *Checkplots* are absolutely essential to reduce the number of errors in IC layouts.. They enable the designer to graphically check alignment and positioning to catch design rule errors as well as typographical and logical errors in the design. Such checks cannot be made from examination of CIF code (Caltech Intermediate Form -- see chapter 4 of [Mead 1978]). See section 2.3 for more on checking.

Good checkplots must distinguish between mask layers and show their overlaps. Checkplots with filled-in rectangles clearly show the overlap of layers in the circuit. Filled-in color checkplots are ideal because of their high information density, but plotters of this type are new and fairly expensive. Color line-drawing plotters are nearly as good, especially if the layers are "hatched" in the appropriate color to show the interior of boxes. Icarus's *stipples* [Fairbairn 1978], gray-pattern shading, or differently hatched rectangles can be used in a black-and-white environment to distinguish between layers.

Usable checkplots of low resolution can be obtained from an ordinary lineprinter, using different characters to represent different layers and separate characters or overstruck characters to show overlapping layers [Gibson 1976, Larsen 1978]. Storage-tube displays (sometimes with hardcopy units) are often used to display IC designs; but in a complicated design it is difficult to visualize overlapped areas from outlines. Their advantages are fast response, relatively small expense and short development time.

### *2.1.2 Implementation of a Basic Design Aid System*

A minimum IC design system might consist of a text editor to enter a design in CIF code, and checkplot-generation routines to view the design on the lineprinter. Although this may sound tedious to those who have used high-powered CAD systems, this method of IC layout has been used with acceptable results. However, for just a small investment of time, a much nicer layout system can be had.

### *2.1.3 Implementation of a Better Design Aid System*

CIF code was not intended to be used as an IC design language, and so it is not oriented to a human operator. A more efficient, human-oriented method is needed to enter designs into the CAD system. There are essentially two ways to build a more efficient system, each of which addresses a different class of problems in IC design. A more powerful *design language* reduces problems of relative positioning of objects and an *interactive graphic system* reduces problems of coordinate entry and editing. The implementation of a design language may initially appear to be a monumental task. However, if approached in the correct manner, for example by embedding it in an existing language, it can be developed in a few months. This effort is well spent, since such a language can serve as an extremely powerful and satisfying design tool. An interactive graphic system requires more hardware and more complex software, but such a system can reduce the circuit design time greatly, since even the initial sketches are made directly within the CAD system.

### *2.1.4 Layout Language*

Many errors in IC design stem from mis-positioning of objects due to the movement of adjacent objects. The mis-positioning problem could be avoided if there were facilities in the language for *parameterization* of objects, for example, basing the position or size of one object on the position or size of another. This is similar to passing parameters to a procedure in a programming language. In addition, it would be nice to specify a shift register by the number of bits, or a PLA by its program. This requires loops and conditional statements in the layout language; the addition of such features makes the layout language look a lot like a general programming language.

An easy way to get a powerful layout language is to implement the CIF commands of drawing a box, wire, symbol and so forth as procedure calls in your favorite programming language. Then, you can use the entire power of FORTRAN, PASCAL, SIMULA or WHATEVER to do the conditionals, the loops, the parameterization of the geometry and any other special constructs that might be necessary or convenient. Several systems of this type have been built, and their advantages include short lead time to a very powerful system as well as infinite expandability and an incredible richness of language [Locanthi 1978]. This type of language is recommended as an initial design system. More specifics on layout languages are found in section 2.2.

### *2.1.5 Graphic Input*

There are two ways to get graphic designs like IC layouts into a computer without typing numbers, by *digitizing* hand drawings and by drawing the design directly into the computer with an interactive graphic system. Digitizing starts with a clean scale drawing of the layout, which is entered by an operator using a digitizing table or camera; it is less prone to errors than typing numbers, but leaves the tedious task of editing the design to the layout person with pencil and eraser. These tasks are performed more easily with an interactive graphic system.

A special purpose interactive graphic editor, engineered for the special needs of IC layout, is a most effective design tool. Ideally it should be as easy to use as paper and pencil, yet directly produce precise layouts on a specified grid. An interactive graphic editor enables the designer to experiment with a number of possible designs quickly, shortening the design time immensely [Fairbairn 1978].

Unfortunately, the ideal system is not yet commercially available, and to develop such a system on your own is a substantial task. The necessary hardware includes a *pointing device* with which to draw the layout, and a *graphic display* to show the design as it is being entered, allowing the designer to instantly correct any errors that occur. A discussion of pointing devices and graphic displays can be found in [Newman 1973]. Even with a good display and pointing device, it is difficult, for example, to route long wires around a complicated design.

### *2.1.6 Considerations for an Advanced IC Design System*

An unaided graphic system cannot easily handle parameterization and conditional placements. On the other hand, layout languages have the problem of tedious cell layout. Clearly, the ultimate IC design system should allow both interactive graphics and programmatical positioning of objects. An ideal system would have a near instantaneous response from a change in the layout program to a change on the layout graphics and vice versa, and have both representations interactively displayed on a high-resolution video display.

An important idea in design systems is the concept of *hierarchical* or *structured design* -- dividing the given problem into more easily handled subproblems. This cuts down the amount of information the human designer must handle at one time in order to solve the design problem. In IC design, the elements of the hierarchy are called *cells* or *symbols*. An *instance*, or "use" of a cell can be embedded within another cell, meaning that the contents of the cell are supposed to be inserted into the design at that point (see section 2.2). Instances of cells are embedded in other cells much the same way procedure calls are embedded within other procedures in a programming language. This hierarchical design guarantees that all instances of a cell are correct, provided the original cell is correct.

Multiple *representations* allow the designer to include in the description of a cell all the information relevant to that cell, enabling him to describe cells graphically or programmatically or any one of a number of different ways. Thus an integrated circuit can be viewed as a mask layout, a stick diagram, a functional description, an electronic circuit, text documentation, and so forth. It is important to keep this information together because no one piece adequately describes the cell.

Representations can be generated from one another, thereby ensuring that the circuit that was sent to the circuit simulation program, for example, was indeed the same circuit that appeared on the mask. It would aid the designer greatly if he could visualize the cell with all its representations as a unit, but it adds tremendously to the expandability of the system if each representation data piece is implemented independently of the other representations because we cannot now foresee what design concepts will be available in the future. For this reason, the design database should be flexible enough to accommodate new, as yet unspecified, representations.

### *2.1.7 Conclusions*

With the advent of VLSI, our circuits will have the capability of being orders of magnitude more complex than they are now. The use of cells and instances will help ease the complexity with which the individual must cope. Representations will enable us to keep all the relevant information together. The computer will ease the task of dealing with this complexity, handling the enormous housekeeping chores and organizing the overall design effort.

## 2.2 IC Layout Languages

*[section contributed by Maureen Stone, Xerox ASD]*

The most desirable format of a language that describes IC mask layouts depends strongly on the point in the IC design process where a description of the IC is needed. At the end, in the mask generation process, the format of the description is entirely determined by the needs of the pattern generator employed. At an intermediate level, it is most desirable to have a description in a form that is most suitable for easy conversion into the many different formats needed by different output devices. An example of such a language is Caltech Intermediate Form (CIF), described in [Mead 1978]. For the original creation of an IC design, the human engineering aspect is most important. That is, the language description should reflect the design process. Repetitive and redundant information should be compressed, and the syntax should seem straightforward to the designer.

This section will describe how languages can be used in IC design, what constitutes a basic set of functions for a simple description language and how such systems should be organized. The last section will discuss more advanced layout languages, especially in reference to using the power of programming languages as a design tool. Appendix D contains a description of ICLIC, a layout language that was originally developed at Caltech this spring (1978). This language and the design examples presented in the Appendix will be referenced throughout this section.

### *2.2.1 Designing with a Layout Language*

The issue of describing a mask can be examined in from two different points of view. At the lowest level is a geometric description of each layer. The shape and position of each element is described with respect to some coordinate system. The design process is then just a matter of digitizing the layout or typing in the coordinates for each shape. However, a layout language should not be just a digitizer in text form. It should, ideally, approximate the way the designer thinks about the layout, in terms of its function and its contraints. Even very simple languages can be organized in a manner that is more descriptive of a design than just its geometry.

The process of using a language in a CAD system involves the following steps:

> First, the design is partitioned into cells and subcells. The more the layout is partitioned into repetitive modules, the less language it will take to describe it.

Second, each cell is sketched, or drawn to scale as much as is necessary. At this point the coordinate system for the layout will have to be defined. Critical points and distances are defined and labeled. In general, the more primitive the language, the more will have to be drawn to scale.

Third, for each section the description is entered into the system. Some sort of graphical output must be produced to verify the correctness of the description. Corrections are made to the source file until the cell is completed. Lower level cells are then combined to describe upper level cells until the whole chip is defined.

The response or turnaround of the CAD system will determine how the designer is going to use it. For example, if the facilities are batch mode processing with a four hour turnaround, the designer will be likely to draw most of the circuit to scale and painstakingly check the source code. The other extreme could be a color interactive graphics terminal with very fast translation from input to display.

### *2.2.2 Basic Features of a Layout Language*

This section describes the features of a simple layout language and how they might be organized. An example of such a language is in the first 3 chapters of the ICLIC description given in Appendix D.

Most simple layouts use only a few basic shapes, predominantly rectangles and wires. These can be described on a unit square coordinate system or grid and constrained to right and 45 degree angles. Therefore, a layout language can begin with a description of these shapes, and a way to designate their layers.

A simple rectangle or *box* is a rectangular area with the sides aligned parallel to the axes. It can be specified by two opposing corner points, or by one point (e.g. center or corner) with a width and a height.

A *wire* is a track of uniform width defined by its center line and width. The path of the wire is defined as a list of coordinates. In general, a shorthand notation is used for paths that only make right angle turns. That is, only one coordinate changes at once so only that change is specified.

For example: X=x1, Y=y1 $ Y=y2 $ X=x2 $ X=x3,Y=y3 $

The "$" means make new wire section. A more powerful syntax uses an infix notation for points and has a symbol for the current X and Y.

For example: x1#y1 ; .#y2 ; x2#. ; x3#y3

The "." means current coordinate, and ";" is a delimiter. Having a notation for the current coordinate means that it is possible to compute with it, thus leading to relative coordinates.

For example: x1#y1 ; .#.+d1 ; .-d2#. ; x3#y3.

The notation ".+d1" means use the current coordinate plus the distance d1 for the new coordinate.

The mask layer to which a particular shape belongs is specified by some mnemonic. ICLIC uses color codes, such as red for polysilicon, green for diffusion, as suggested in [Mead 1978]. Some languages associate a layer specification with each item. Others make it a global switch that affects all subsequent objects, as does CIF.

Attempting to define a layout with just these simple primitives will result in an explosion of data. Therefore, it is essential to have some mechanism for grouping and reusing sets of shapes. Such a collection of shapes is often called a *symbol* or *cell.* A layout language needs some way of defining and using symbols. It should be possible to nest symbol *calls,* and to use symbol calls in definitions of other symbols. However, it is not necessary to be able to nest symbol *definitions.*

Using symbols implies some way to position each instance of the symbol. In general, the call will map the *symbol origin* to the current X,Y. A rectangular array of symbols is such a basic layout feature that some method for easily generating it should be a part of any layout language. The syntax of such a construct needs only the starting position, the number of symbols in the X and Y directions, and the spacing in the X and Y directions. Such a construct is described in section 3.5 of the ICLIC manual in Appendix D.

Besides being positioned, symbols may also be *transformed.* Standard transformations are: *scaling, mirroring, translation,* and *rotation* (see [Newman 1973] for a full discussion of transformations). For a simple layout language, translation, rotations by increments of 90 degrees, and mirroring about the axes are sufficient. Scaling, which is changing the size of a symbol, and rotation by arbitrary angles are rarely used.

The use of transformations brings up the issue of nesting transformations. The order in which transformations are performed is important, and even the simple case of combining a rotation with a translation can produce very different results depending on the order in which the operations are executed. Therefore, the syntax for calling transformation routines should be very clear about the order in which the functions are performed.

Besides the issue of the user interface, there are many difficulties involved with the definition of general transformations for integrated circuits. One example is the problem of how to adjust the results of a transformation back to a grid (rounding of coordinates) without causing unintendend breaks or overlaps. See the description of CIF in [Mead 1978] for a more complete discussion of the problems of transformations in integrated circuit design.

### *2.2.3 Variables, Parameters, and Relative Positioning*

A *variable* is simply a name which can be assigned a value. The benefits resulting from the use of variables within an IC design language are equivalent to those obtained in a general purpose language. That is, once a name is used for a value, all instances of that value can be changed simply by changing the assignment to that variable. Furthermore, a name can be descriptive, such as "InverterCenterX". It is much easier to make changes and find errors in a list of descriptive names than among a mass of anonymous numbers.

A symbol whose definition is controlled by a set of named values is said to be *parameterized*, and the names are said to be its *parameters*. A simple example of this is a symbol which uses variables for size and positioning information. Such a symbol can be changed by assigning different values to the variables used in its description. In a language which supports more advanced programming constructs, such as subroutines, loops and conditionals, parameters can be used more extensively to control the properties of the symbol. For example, the parameters of a PLA subroutine could describe its size and function.

The use of *relative coordinates* involves defining symbol parameters with respect to other elements in the cell, instead of with respect to the absolute coordinate system for the layout. Changes made in a symbol which is defined in this manner can ripple through the layout, keeping design rules intact.

For example: `RW({OUT1 ; .+RWFRMG#. ; .#IN2.Y ; IN2})`

This example draws a red (poly) wire from the point OUT1 to the point IN2. The path followed is: horizontal from OUT1 to the minimum distance a red wire can be from a green area; vertical to the Y coordinate of IN2; horizontal to IN2. This wire will remain connected and running at the minimum design rule distance from the first device whenever OUT1 and IN2 are changed.

### *2.2.4 Program Organization*

Methods of organizing a language description for a design are similar to techniques used in general purpose programming languages. They also relate strongly to methods discussed in reference to the design process as a whole. The following points will be summarized here: documentation, modular design, parameterization, defaults and conventions.

Documentation includes not only comments on the source code, but the use of descriptive names for variables. All programs should be thoroughly documented, of course.

Modular design is the process of coding and debugging each cell separately, then combining the working cells to make larger ones, as has been emphasized throughout this document. By making large programs out of a collection of small ones, not only is the design cleaner, but the process of coding, executing and debugging the programs is simplified. Uniformity reduces the complexity, and hence the errors, in a layout. Therefore, using standard symbols for circuit elements is highly recommended. Some symbols, such as contacts, are common enough to be defined in the language. Other symbols could be kept in library files, see Appendix E.

The advantages of parameterization can be broken into two main areas. First, changes to a cell can be more easily made; for example, if the parameters are defined as relative coordinates, the cell can be somewhat self-adjusting with respect to small changes. Second, the same code can be used to produce similar, as opposed to identical, cells. Therefore, there is less code to debug, reducing errors.

One of the initial steps in writing any program is defining *defaults* and conventions for variable names and parameter values. An example of this is the convention in ICLIC of specifying all design rule distances as cFRMc, where the c is replaced by a layer mnemonic. For example, RFRMG or GFRMR is the minimum design rule distance between red and green areas. The default value is 1 lambda (3 microns in 1978). Defaults and conventions should be documented at the start of the program, and adhered to throughout. It is useful for the language to contain a set of default parameters and naming conventions for wire widths, wire spacings, minimum design rule distances, etc.

### *2.2.5 Advanced Layout Languages*

There are two ways to make extensions to the basic language concepts outlined in the previous sections. One is to continue to treat the design process as a set of calls to pre-defined functions, and to extend the language by increasing the complexity of the functions. For example, ICLIC has a function for defining wires which run on more than one layer which automatically inserts the contacts between layers. The second, more powerful method involves treating IC design as a programming process. Therefore, the design language needs to be extended to include subroutines, loops, and conditionals. The issues then become those of conventional language design.

A good way to implement a layout language is to define a set of procedure calls within an existing language. The elegance of the finished set will depend somewhat on the base language used, but it seems safe to say that something akin to CIF with variables could be implemented in any language that allows dynamic storage allocation. A language like ICLIC depends heavily on the concepts of concatenating symbols together, and the ability to compute relative coordinates symbolically from the current coordinate. ICLIC was implemented as a set of procedures in ICL [Ayres 1978] which already supported these concepts. A similar system has been implemented at Xerox PARC in SMALLTALK as a programming example for a forthcoming book. A different system, which contained a basic layout language plus some automatic wire routing routines, has been implemented in SIMULA at Caltech [Locanthi 1978].

There are a number of advantages to basing a layout language on an existing language. Development time, compared to writing a compiler from scratch, is much reduced, and the programming environment is one that is already familiar to the user community. In addition, the resulting layout language has the full power of the underlying language.

The concept of a language description provides a means for a functional description of the circuit being designed. At the layout level, one could imagine the process of developing cells that are passed a set of input specifications and return the layout plus a set of output specifications. Such modules could then be connected by running wires from an output definition to an input definition, either by hand or by an automatic wire routing system. The point being made here is that a program which describes a design element can be extended to provide a much richer description than just the data for the mask. Therefore this description can be used for a variety of purposes, such as wire routing, design rule checkers, and simulation.

### *2.2.6 Conclusions*

While the use of a set of basic functions to describe a layout is straightforward, there are important issues with regard to the design and use of such languages. These issues have been discussed in sections 2.2.1-2.2.4. Even such simple systems can be very useful for design. However, the real power of a language description comes not from the expansion to fancier functions, but from invoking the power of a general purpose programming language as a design tool.

## 2.3 Checking Your Design

*[section contributed by Wayne Wilner, Xerox PARC]*

Integrated circuit design requires near-perfection. Certain flaws, such as a short between two clock lines, can render the whole chip useless. Checking your design adds a week of tedium to your project, but without that week, four months can go down the drain (to say nothing of cost).

Checking can be done by eye and by computer. Since sight-checking is available to all, we'll discuss it first. Actually, sight-checking is superior to automatic checking in several ways.

### *2.3.1 Checking by the designer*

Many fatal errors in a design do not exhibit themselves as violations of design rules. Consider logic errors. Consider state machines which are initialized to terminal states. Consider transistors which are wired incorrectly, but within the design rules. Consider an array of cells which are supposed to abut and do not; if the space between them is larger than the minimum spacing for all layers, design rules may be observed while the array is grossly in error. Consider the placement and continuity of busses. These errors are representative of flaws for which the designer is singularly responsible.

Many of these can be spotted on checkplots of relevant subsets of layers. A plot of metal and contacts can reveal errors in continuity which would otherwise be lost in the details of a full plot. A plot of poly, diffusion, contacts and implants enables one to check their all-important overlap. The effectiveness of this technique is very high when the plots are large, clean, and of high contrast.

A plot of each individual layer should be sight-checked by someone besides the designer. It's amazing how many design errors look odd on such plots, even to those who don't know the circuit.

### *2.3.2 Checking by computer*

Some flaws are violations of design rules. Two basic tactics can be employed to avoid them: (1) the use of programs which check the final mask descriptions for certain violations, and (2) the use of programs which generate layouts in a manner that guarantees that there are no violations. At present, there are no programs which generate layouts completely. An intermediate step in that direction is the "Sticks" approach [Williams 1977] which starts with a grossly-spaced circuit which is then trimmed mechanically to minimum spacing by the CAD system.

### *2.3.3 Error-checking programs*

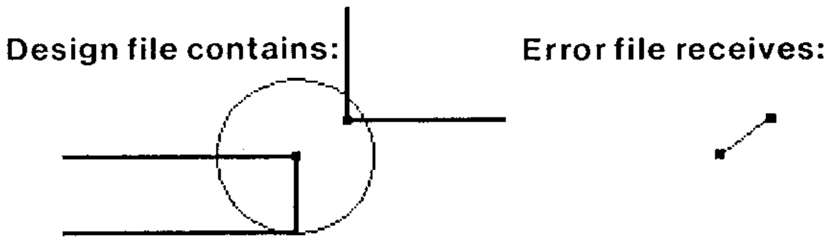
Error-checking programs embody the rules for a particular process and examine pattern generation tapes for violations, reporting each instance in terms of coordinates or patterns, along with the nature of the violation. Design rules typically assign minimum distances to:

dimensions of features, such as breadth of runs or size of contact cuts;

spacing between features in the same layer, such as distance between runs;

spacing between features in different layers, such as overlap of metal and contact windows.

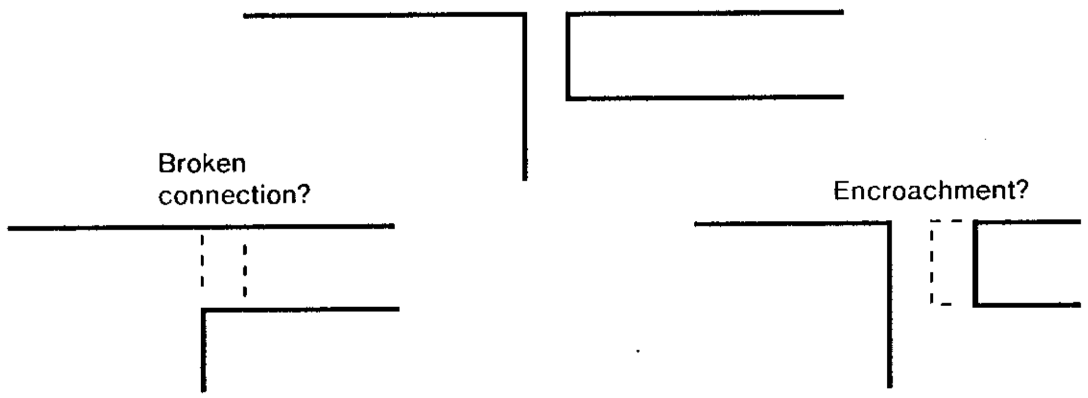
For example, suppose unconnected areas of polysilicon must be 2λ apart. In the diagram below, a circle of radius 2λ centered at the upper right corner of the left-hand area reveals that the right-hand area is too close.



This design rule violation may be reported in terms of a line segment, that is, two points, one at the periphery of each area, and their (insufficient) separation. It is a non-trivial problem to present violations to the designer in the most convenient way.

### 2.3.4 *Inherent limitations*

An inherent limitation of design rules comes from their pertaining solely to the lowest level of detail. Consider the following diagram. Two areas are separated by less than their minimum spacing.



It is clearly a design rule violation, but is it a broken connection or is it an encroachment? The designer will have to decide and fix it appropriately.

Checking for design-rule violations appears to be a problem of geometry, of a well-defined, computationally-tractable nature. This is an illusion. Checking design rules requires so many computations that the problem is really one of managing main and secondary storage, in other words, an operating system problem, not well-defined, and different for each individual computer.

### 2.3.5 *Design rule checking in the context of structured design*

Another point of view on design rule checking is that of structured design. Using structured design, most projects are constructed from a few basic cells which are very simple. Their simplicity makes sight-checking design rules adequate.

Automatic checking is quite useful, still, for several situations. Assembling systems from cells introduces errors in cell placement. Cells which interconnect must abut, not overlap or be separated. Cells which overlap may introduce design rule violations, even though the cells themselves are correct. Cells which provide alternate interconnections for a given signal must dispose of all unused connections. Wide-ranging interconnections between cells are often hard to scan thoroughly and show up small and indistinct on checkplots which span their extent.

### *2.3.6 Parameterizable design rule checking*

In laboratories which experiment with different processes, the critical distances may vary from month to month. In experiments with custom circuits, the objective may be to find how exceptions to the design rules can be exploited. Therefore, while the types of rules may be rigidly bound into a checking program, specific distances should not be.

Is it feasible to make a design rule checker in which the design rules are parameters? A fundamental problem is that the nature of design rules varies. Some involve unconditional distances: "metal runs must be at least seven microns wide." (Of course, verifying this can be extremely difficult.) Others involve conditions: "polysilicon must extend at least four microns beyond diffusion, unless the gate dimensions are small, where six microns are required". It is an unsolved problem to mechanically create a program which can efficiently verify geometrical constraints of varying nature. It would seem harder to mechanically generate checkers than to mechanically generate layouts. The latter is clearly preferable because of the greater range of errors which are detected or eliminated.

### *2.3.7 Summary*

Unless circuits are generated mechanically, they must be checked thoroughly. Sight-checking of critical distances is astonishingly effective when done on a large, clean, high-contrast plot, due to the pattern-recognition power of the brain. Mechanical checking can serve as a further check; it is limited by machine resources and imprecise or insufficient descriptions of the rules.

## 2.4 Simulation as an IC Design Tool

*[section contributed by Richard Lyon, Xerox PARC]*

Simulation is a design technique widely used in a variety of engineering disciplines. When it is too difficult to verify the correctness of a design by inspection, by proof, or by test, simulation may help. Simulation allows the designer to test a design before building it, by modelling in detail the components from which the design is built, and by computing their interactions under various conditions. Simulation is useful at many levels in integrated circuit and system design; system-level, register-transfer-level, logic-level, and circuit-level simulators are useful at various stages of the IC design process. A related activity is the design of IC fabrication processes, which can benefit from process simulation; the simulation of process variations may become more important as VLSI approaches the physical limits of device sizes, where the set of devices used by the system designer must be carefully matched to the technology.

Unfortunately, not many generally useful simulators are readily available. Even when such a program is available to run on your computer, the problem of preparing data in a form suitable to the simulator can be formidable. It is easy to write a register-transfer-level simulator, for example, but the hard part that makes it useful is to provide an automatic link from the design language to the simulator input language. There is not yet enough commonality of design methods in the digital system design field to result in wide availability of such a program. In the circuit design field, on the other hand, the method of design has traditionally been standardized to drawing by hand on paper the interconnection of standard types of lumped circuit elements. From here it is logical to assume hand translation to the language of a circuit simulator. For this reason, circuit simulators have been developed in standard languages (Fortran IV) and are widely available. Two such simulators, somewhat tailored for IC simulation, are SPICE from U. C. Berkeley, and MSINC from Stanford; their input languages are similar, and one example should serve to illustrate both.

Circuit simulation can be very useful to the integrated circuit/system designer if it is applied to those problems that require it, but should not be relied on to verify the correctness of a complicated system design. In digital system design with a consistent design philosophy, it is usually possible to identify the critical parts of the design (for example the longest chain of pass transistors, the new RAM cell, or the node with the highest fanout); in this way, critical parts can be identified for simulation (see [Mead 1978] chapters 1 and 7 for information on critical timing; see chapter 4 for more on simulation and testing). Of course, even simulation will not verify that the design will run fast enough if the simulation parameters and models do not realistically reflect the process used to make the circuit.

As an example, we have simulated the output pad driver called *PadOut*, which was designed in *Icarus* (Integrated Circuit ARtwork Utility System, an interactive layout design system) according to the Mead and Conway design rules, with lambda equal to 3 microns. This is a driver intended to interface NMOS chips to other popular logic families, at speeds and voltages comparable to TTL. It uses push-pull enhancement-mode output drivers, driven in turn by *super-buffers* (see [Mead 1978] chapter 1). The fanouts are generally somewhat higher than the theoretical optimum of *e*, to reduce space and power at the expense of speed. Figure 2.4.1 is the Icarus layout picture of PadOut; notice that the output transistors are both wrapped around the pad. The schematic diagram is shown in Figure 2.4.2; it includes node numbers and element names which are needed for translation to the simulator input language.

The simulator SPICE was used at Xerox PARC, on the MAXC2 computer, which has no floating-point hardware; therefore, the execution of the Fortran program was blindingly slow. Figure 2.4.3 shows the *input deck*, an ASCII text file. The SPICE program, like most widely available programs, was written for the card-reader/line-printer/batch-computing environment which is found at the typical university computing center. Therefore, be careful of input formats; only 72 columns of 80-column cards are used -- long lines use continuation marks in column 1, as in Fortran. The documentation is sparse, but keep in mind that you should not do anything you could not do on a keypunch, such as lower case letters. See *User's Guide to SPICE* by E. Cohen and D. O. Pederson, from U. C. Berkeley Dept. of Electrical Engineering and Computer Science.

In the input listing, each line is called a *card*. The first line is the *title card*, and lines starting with * are *comment cards*. Each *element card* names a component (the first letter of the name determines the element type, such as M for MOSFET), tells what nodes it is connected to (in order, such as drain, gate, source, substrate), and gives a few parameters (such as width and length in centimeters). There are also *model cards* and *control cards*, which will not be described here, but can be seen in the listing.

We have described in the element cards the circuit of Figure 2.4.2 (some of the parameters are estimates, such as AS and AD, areas of source and drain). The first inverter is not part of PadOut, but represents a typical signal source, which is in turn driven by a 3.5 volt, 20 Mhz square wave generator with 2 nsec rise and fall times.

The output file produced by SPICE from the input shown was too long to include here. The most interesting part of it is shown in Figure 2.4.4, the graph of the time response of the various nodes, which is plotted line-printer style by typing the node numbers in appropriate columns. To make it

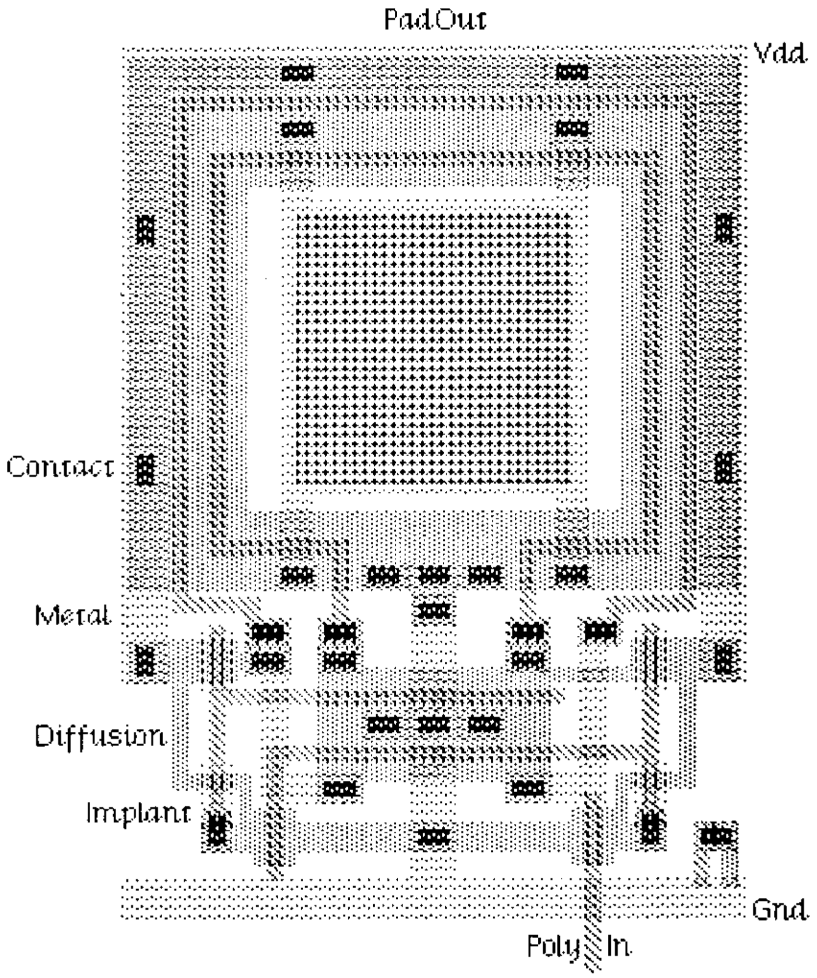


Figure 2.4.1. Icarus Layout of "PadOut"

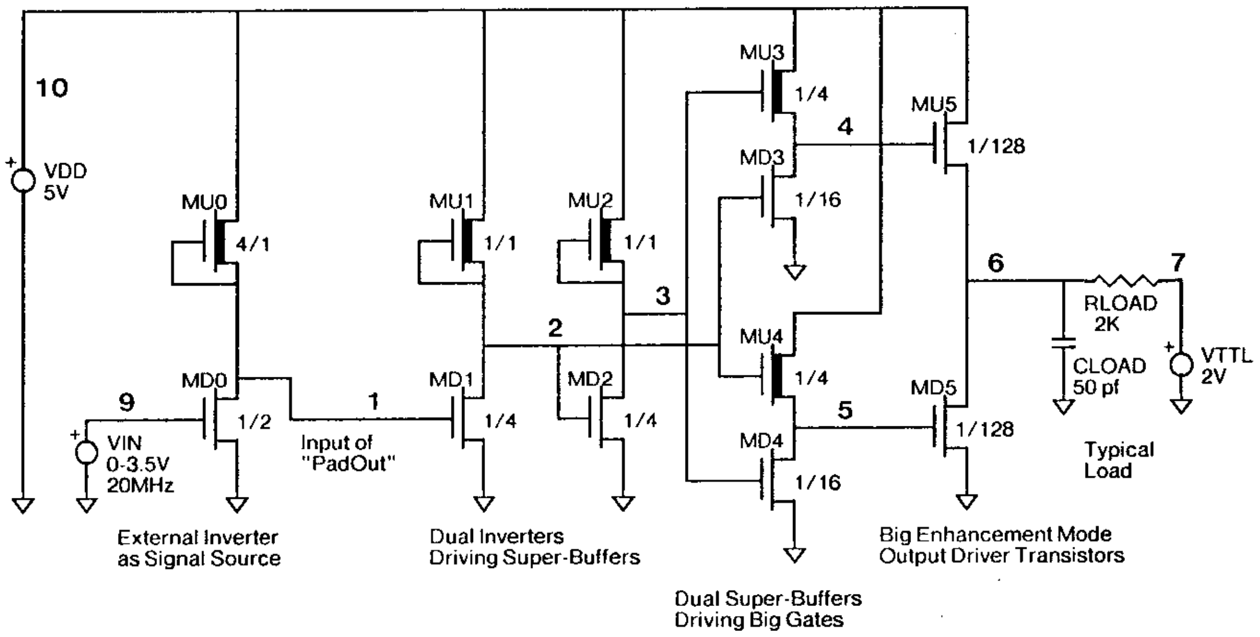


**Figure 2.4.2.**

**Diagram of "PadOut" Output Pad Driver with element names and node numbers for simulation with SPICE.**

```
PAD DRIVER SIMULATION
* RF LYON -- JULY 13, 1978
*
VDD 10 0 DC 5VOLTS
VTTL 7 0 DC 2VOLTS
VIN 9 0 PULSE 3.5VOLTS 0VOLTS 2NS 2NS 2NS 23NS 50NS
*
MD0  1 9 0 0 ENH W=12E-4 L=06E-4 AS=144E-8 AD=144E-8
MU0 10 1 1 0 DEP W=06E-4 L=24E-4 AS=144E-8 AD=144E-8
MD1  2 1 0 0 ENH W=24E-4 L=06E-4 AS=144E-8 AD=144E-8
MU1 10 2 2 0 DEP W=06E-4 L=06E-4 AS=144E-8 AD=144E-8
MD2  3 2 0 0 ENH W=24E-4 L=06E-4 AS=144E-8 AD=144E-8
MU2 10 3 3 0 DEP W=06E-4 L=06E-4 AS=144E-8 AD=144E-8
MD3  4 2 0 0 ENH W=96E-4 L=06E-4 AS=600E-8 AD=600E-8
MU3 10 3 4 0 DEP W=24E-4 L=06E-4 AS=144E-8 AD=144E-8
MD4  5 3 0 0 ENH W=96E-4 L=06E-4 AS=600E-8 AD=600E-8
MU4 10 2 5 0 DEP W=24E-4 L=06E-4 AS=144E-8 AD=144E-8
MD5  6 5 0 0 ENH W=768E-4 L=6E-4 AS=4000E-8 AD=4000E-8
MU5 10 4 6 0 ENH W=768E-4 L=6E-4 AS=4000E-8 AD=4000E-8
CLOAD  6 0   50P
RLOAD  6 7   2K
*
.MODEL ENH NMOS (NGATE=1E20 TPS=1 XJ=1E-4
+ CGD=4E-12 CGS=4E-12 CGB=2E-12 TOX=95E-7
+ NSS=-22E10 NSUB=8E14 )
.MODEL DEP NMOS (NGATE=1E20 TPS=1 XJ=1E-4
+ CGD=4E-12 CGS=4E-12 CGB=2E-12 TOX=95E-7
+ NSS=80E10 NSUB=8E14 )
*
.TRAN 1.0NS 80NS
.PLOT TRAN V(1) V(2) V(3) V(4) V(5) V(6) (0,8)
.WIDTH OUT=72
.END
```

Figure 2.4.3. SPICE Input Deck for *PadOut*

```
           0.000E-01                2.000E+00                4.000E+00                6.000E+00
                 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
* 0.000E-01   .61  X                   .                        .           X            .
* 1.000E-09   .61  X                   .                        .           X            .
* 2.000E-09   .61  X                   .                        .           X            .
* 3.000E-09   .X   X                   .                        .            X           .
* 4.000E-09   .6 1 X                   .                        .           52           .
* 5.000E-09   .6   X    1              .                        .            X           .
* 6.000E-09   .6   X          1        .                        .           X            .
* 7.000E-09   .6   X                  1.                        .          25            .
* 8.000E-09   .6    X                  .   1                    .       2   5            .
* 9.000E-09   .6    X                  .        1               . 2         5            .
* 1.000E-08   .6     X                 .            1      2    .           5            .
* 1.100E-08   .6       X               .          2     1       .           5            .
* 1.200E-08   .6           4  3        . 2                1     .           5            .
* 1.300E-08   .6                 24   3.                   1    .         5              .
* 1.400E-08   . 6         2            .    4    3          1   .   5                    .
* 1.500E-08   .  6     2               .             4   53   1 .                        .
* 1.600E-08   .     62                 .   5                  431                        .
* 1.700E-08   .      2  6     5        .                        .31  4                   .
* 1.800E-08   .      2 5     6         .                        .  31     4              .
* 1.900E-08   .      X           6     .                        .     X      4           .
* 2.000E-08   .     25               6 .                        .      13       4        .
* 2.100E-08   .     X                  . 6                      .       1 3      4       .
* 2.200E-08   .     X                  .    6                   .        1 3      4      .
* 2.300E-08   .     X                  .      6                 .        1 3      4      .
* 2.400E-08   .     X                  .       6                .         1 3    4       .
* 2.500E-08   .     X                  .         6              .         1 3    4       .
* 2.600E-08   .     X                  .          6             .          13   4        .
* 2.700E-08   .     X                  .           6            .          13   4        .
* 2.800E-08   .    25                  .            6           .           X  4         .
* 2.900E-08   .    X                   .            6     1     .           3  4         .
* 3.000E-08   .     5   1 2            .             6          .            3 4         .
* 3.100E-08   .  1    5              2 .             6          .           3 4          .
* 3.200E-08   .  1       5             .       2     6          .      34                .
* 3.300E-08   . 1           5          .             62      3 4.                        .
* 3.400E-08   . 1                  5   .      3    4 6      2   .                        .
* 3.500E-08   . 1            3        4.      5    6           2.                        .
* 3.600E-08   . 1     3     4          .        6      5        .2                       .
* 3.700E-08   . 1    3 4               .    6                5  .  2                     .
* 3.800E-08   . 1   43                 6                        5     2                  .
* 3.900E-08   . 1   X             6    .                        .5      2                .
* 4.000E-08   . 1   X        6         .                        .  5     2               .
* 4.100E-08   . 1  43    6             .                        .   5     2              .
* 4.200E-08   . 1  43 6                .                        .    5     2             .
* 4.300E-08   . 1  4X                  .                        .     5     2            .
* 4.400E-08   . 1 643                  .                        .      5    2            .
* 4.500E-08   . 16 X                   .                        .       5   2            .
* 4.600E-08   . X  X                   .                        .        5  2            .
* 4.700E-08   . X  X                   .                        .         5 2            .
* 4.800E-08   .61  X                   .                        .         5  2           .
* 4.900E-08   .61  X                   .                        .          5 2           .
* 5.000E-08   .61  X                   .                        .          5 2           .
* 5.100E-08   .61  X                   .                        .          5 2           .
* 5.200E-08   .61  X                   .                        .           52           .
* 5.300E-08   .61  X                   .                        .           52           .
* 5.400E-08   .6  1X                   .                        .           52           .
* 5.500E-08   .6   X     1             .                        .           52           .
* 5.600E-08   .6   X             1     .                        .           X            .
* 5.700E-08   .6   X                   1                        .         2 5            .
* 5.800E-08   .6    X                  .     1                  .     2     5            .
* 5.900E-08   .6    X                  .          1             2           5            .
* 6.000E-08   .6     43                .             1  2       .           5            .
* 6.100E-08   .6        43             .        2       1       .           5            .
* 6.200E-08   .6             4  3      2                  1     .          5             .
* 6.300E-08   .6               2     4 . 3                  1   .       5                .
* 6.400E-08   . 6        2             .       4    3        1  . 5                      .
* 6.500E-08   .   6   2                .              5 4   3  1.                        .
* 6.600E-08   .      X                 5                        X1                       .
* 6.700E-08   .      2   65            .                        . 31  4                  .
* 6.800E-08   .      25       6        .                        .   31     4             .
* 6.900E-08   .      X             6   .                        .     13      4          .
* 7.000E-08   .     X                 6.                        .      1 3      4        .
* 7.100E-08   .     X                  .  6                     .       1 3      4       .
* 7.200E-08   .     X                  .    6                   .        1 3      4      .
* 7.300E-08   .     X                  .      6                 .        1  3     4      .
```

Figure 2.4.4.
SPICE Output

readable, take a bunch of colored markers and draw in the curves for the nodes of interest. You will see that the response from node 1 to node 6 is noninverting, with $t_{PLH}$ of 13 nsec and $t_{PHL}$ of 9 nsec, measured at a 2 volt threshold (or more nearly symmetrical at 11 nsec if measured somewhere below 1 volt).

Is PadOut really this fast? Probably not on most processes; the model cards used here have estimates of the Spice model parameters which were felt to be realistic, but which gave results that are probably too optimistic for most typical 1978 processes. The inverter-pair delay from node 1 to node 3 is seen to be 6 nsec, where the inverter ratios are $k=4$ and the fanouts are $f=5$ (actually 6 for the first inverter). The delay estimate according to [Mead 1978] is then $(k+1)f\tau=25\tau=6$ nsec, so we may conclude that we have simulated a process with $\tau=0.24$ nsec (transit time), which certainly is optimistic. The actual performance of PadOut will have to be determined by test, and will depend on where it is fabricated; some lines would be three times slower than this simulation.

IC designers have relied on simulation as a design tool for years. When the performance of a part being designed is critical (as is typical in manufacturing for sale), and the production/test turnaround is slow (also typical in the IC manufacturing business), circuit simulation is a necessity. However, in the preliminary topological design phase, circuit simulation is not needed; and if turnaround is fast, measurement may be a better way to determine performance than simulation is. Thus, we are now at the point of being able to design complicated *digital* systems without the aid of circuit simulation, by following strict design conventions; however, circuit simulation is still useful for the analog and interface aspects of system design, and is a valuable aid in understanding circuit behavior. We encourage the use of circuit simulation where it is appropriate. We also encourage the development and use of higher-level simulation tools to aid the design of digital systems.

# 3. An Overview of IC Implementation

## 3.1 Mask Generation

MOS circuits are constructed as a sequence of patterned layers on the surface of a silicon wafer. Each layer requires a different *working plate*, or *mask*, to provide the pattern; for Si gate NMOS there are typically six working plates in a set. A working plate is a sheet of glass about 100mm square covered with patterned opaque material on one side. This material is usually one of three materials -- photographic emulsion, iron oxide, or chromium. Emulsion is the least expensive but relatively easily damaged in the contact photolithography process (see section 3.2 The Basic Fabrication Process). Chromium and iron oxide give better line resolution and are very hard, but they are more expensive.

The first step in creating the working plates is plotting the files provided by the designer on a photosensitized glass plate. This first plate, called a *reticle*, differs from a working plate in that it contains only one copy of the relevant chip layer and is plotted at 10x the size of the chip on the wafer. The plotting process takes place in a *pattern generator*; the Mann 3000 is typical of such machines. The 3000 projects (*flashes*) the image of a variable size rectangle on the reticle. The size of the rectangle, or *aperture*, the x and y coordinates of the center and the angle with respect to the x axis are specified by the designer. The nature of the reticle making process has a number of important implications for the designer. All shapes on the masks must be decomposed into simple rectangles. The pattern generation process involves complex mechanical motion; proper sorting of the individual rectangles of which the chip is composed can speed up the pattern generation process considerably -- and thus lower the price. For example a Mann 3000 PG machine is fastest at moving in the x direction, followed by aperture change, y direction, and finally, angle change.

Special features have to be included on the reticle which are used in mask making and in IC processing. *Critical dimensions* (CD's) are simple lines or crosses of a fixed size appearing on each layer; they are used by the mask house to adjust exposure and developing time to insure that these marks and hence other features on the mask are the correct size. It should be noted that exposure time has a definite effect on the feature sizes on the reticle, in particular overexposed areas tend to "grow" slightly; for this reason the designer should avoid substantial overlap between flashes. A *parity mark*, consisting of an arrow or triangle, is sometimes included on each mask layer to help the operator orient the mask. The mark is placed outside of the boundary of the chip pattern. *Fiducials* are small crosses which also appear on each layer outside of the boundaries of the chip. These are used in the *step and repeat* process (see below). Often the parity marks and fiducials are provided by the mask house thus making it unnecessary and undesirable for the designer to supply them. Parity marks and fiducials appear only on the reticles and not on the finished working plates.

Because the interactions between layers are all important in MOS integrated circuits each layer (with the exception of the first) must be critically aligned with a previous layer so that features overlap in the proper way. The fabrication line operators use a set of special patterns on the working plates called *alignment marks* (further described in chapter 4 and section 7.1) to accomplish this.

The reticles are used to make a set of *master plates* in a step and repeat machine which projects an image of the reticle (reduced 10x) onto a photosensitized plate. By precisely stepping the image across the master a matrix of images of the reticle is created. The fiducials are used to control the distance between exposures and to align the reticle images relative to each other.

The working plates are made directly from the masters by contact printing. In cases where a large number of working plates are needed the mask house may make several sets of *submasters* and print the working plates from them.

To facilitate checking of the various layers for mistakes the designer may request color enlargements of the reticles for checking the various layers for mistakes; these *blowbacks* are typically about 100x-150x actual (chip) size. *Black and clear* transparencies (usually 8½" x 11") may also be made at the same time. They are sometimes used in the interaction between the operators on the fab line and the designer to indicate features on the mask such as alignment marks.

## 3.2 The Basic Fabrication Process

The process discussed here is the standard Si gate n-channel MOS process. The reader need not be concerned with learning all of the details of the process, indeed most of the decisions concerning processing are made by the fabrication line. The designer may not know which particular techniques the fab line uses, and may not care as long as his standard circuits exhibit normal performance. This section is presented to provide background for those interested in what really happens behind the clean room doors.

Integrated circuits are built in layers, some of which are patterned by a *photolithographic* process. Such layers are created in a sequence of steps beginning with the deposition (or growth) of some material on the surface of the wafer. It is then coated with a thin layer of photosensitive chemicals, called *photoresist*, and exposed to ultraviolet light through the proper working plate. The exposure can take place with the working plate pressed against the wafer (*contact photolithography*) or by projecting an image of the working plate onto the wafer (*projection photolithography*). (Projection techniques are becoming more widely used in spite of the extra equipment and maintenance needed since the masks are subject to less wear and contamination than contact masks. Consequently masks last longer and it is easier to control certain kinds of defects incurred in the photolithography steps.) If *negative resist* was used those areas of resist which were exposed to light will be hardened while *positive resist* is softened in the exposed areas. The resist is *developed* by immersing it in a solvent which dissolves the unexposed (for negative resist) or exposed (for positive resist) portions, leaving the desired pattern.

After the developed resist is hardened by baking at a low temperature the material in the uncovered areas is removed by *etching*. Two techniques are widely used today. In the older *wet etching* process the wafer is immersed in a bath of chemical etchant under controlled temperature conditions for a specific amount of time. Wet etching depends on the availability of an etchant which will dissolve the layer beneath the photoresist, yet not significantly attack the resist. The wet etchants for some materials, for example silicon nitride, dissolve photoresist as well as the desired material. Such materials require an intermediate pattern to be formed in another layer which serves as the actual etching mask. *Dry etching* techniques such as *plasma* etching use a stream of ions and electrons to blast away material. The plasma etching technique gives better results for fine geometries and also permits the direct use of resist as an etching mask. Following the etching step the remaining resist is removed, leaving a pattern in the underlying material.

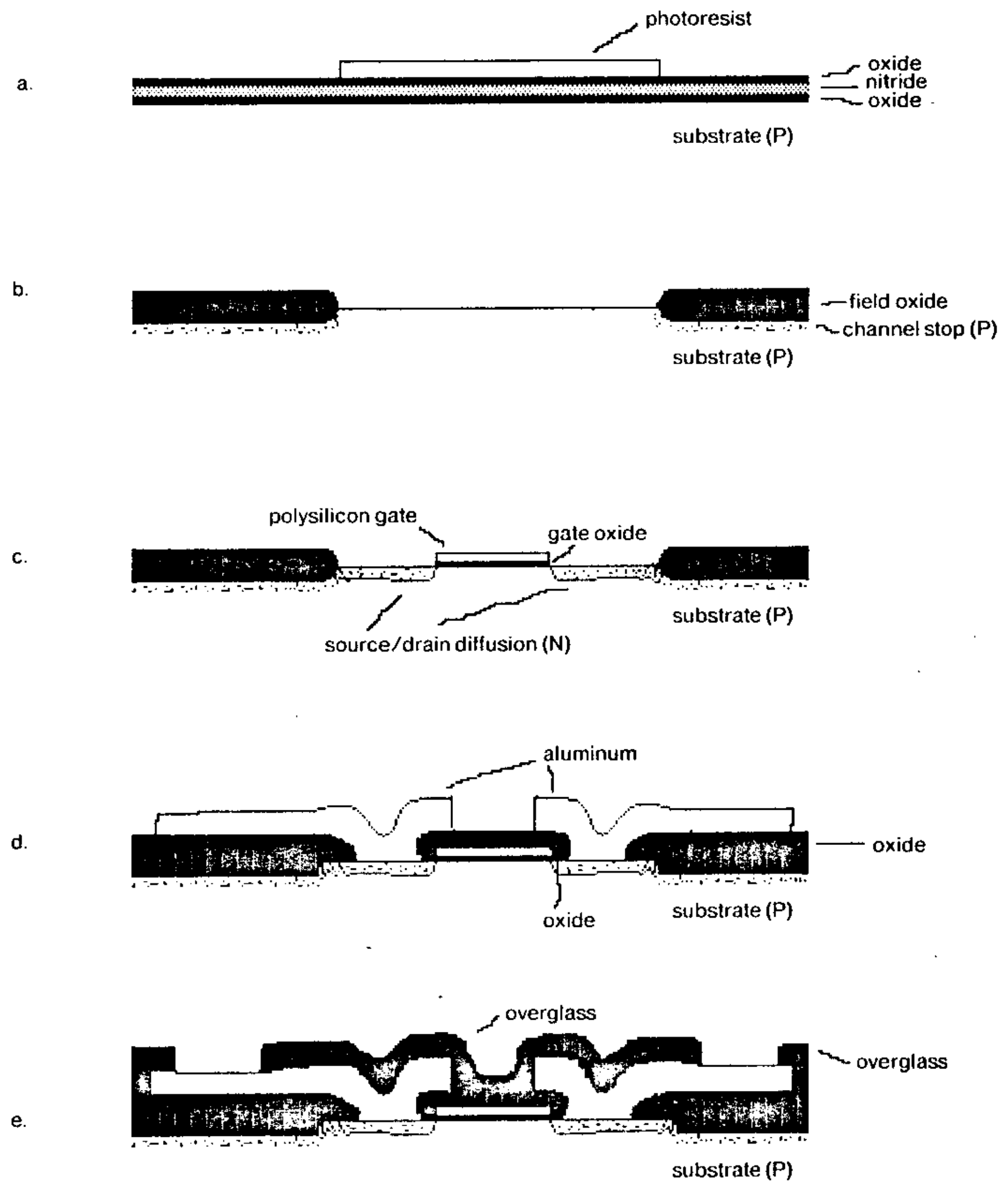
This sequence is repeated for the various layers of the circuit. About six photolithography/etching cycles are required to build up a typical Si gate NMOS circuit. The entire process entails over forty individual steps, outlined here.

The wafer of *p type* (*100 crystal orientation* for lowest interface state density) silicon is scrubbed and a thin layer of silicon dioxide (hereafter called "oxide") is thermally grown on the surface. This layer serves as a mechanical buffer zone for the silicon nitride ($Si_3N_4$) that follows. The buffer zone is needed to relieve stress caused by differences in the coefficients of thermal expansion of silicon and silicon nitride. A layer of $Si_3N_4$ is deposited by *chemical vapor deposition*, then another layer of oxide is grown. Photoresist is applied over the entire surface and the wafer is exposed to ultraviolet light through the *diffusion layer* mask. The resist is developed, leaving open areas over the *field* region (figure 3.2.1a). The top layer of oxide is etched away wherever there is no photoresist using a hydrofluoric acid solution. After the resist is removed this top layer of oxide is used as a mask for patterning the nitride since the photoresist alone will not stand up to the chemicals used in the wet etching of silicon nitride. A third etching step is used to remove the bottom layer of oxide. *Ion implantation* is used to place the *channel stop* region and a thick *field oxide* is grown over those areas. The field oxide and the channel stop are *self-aligned* with respect to the source/drain diffused areas (the nitride covers the source/drain areas during channel stop implant and prevents oxidation of the underlying silicon during field oxide growth). The remaining nitride and the thin oxide under it are removed resulting in the profile shown in figure 3.2.1b.

Next a layer of photoresist is applied and the wafer is exposed through the *depletion mode implant* mask. The resist is developed, leaving open spaces in the gate regions of the depletion load transistors. Another ion implantation step occurs here (using the resist as a mask) to alter the threshold voltages of the depletion load transistors. The resist is removed and a thin layer of *gate oxide* is grown. If there are *buried contacts* used in the IC design more photoresist is applied, the wafer is exposed through the buried contact mask, the resist is developed, the gate oxide is etched away in the contact areas, and the resist is removed. This allows the *polysilicon gate* material to contact the substrate in selected areas.

A layer of polysilicon is deposited from a chemical vapor and a thin layer of oxide is grown on top of that to provide a surface that photoresist will adhere to. Resist is applied and the wafer is exposed through the polysilicon layer mask. The development of the resist leaves the gates of the transistors covered; the uncovered areas of oxide and polysilicon are etched away (a little field oxide is also removed). After the resist is removed the *source* and *drain* regions are doped (figure 3.2.1c) in a phosphine gas atmosphere. Since the edges of the polysilicon gates define where the source/drain regions begin these features are also self-aligned. Here self-alignment results in a

Figure 3.2.1 Si Gate NMOS Processing Steps



(not to scale)

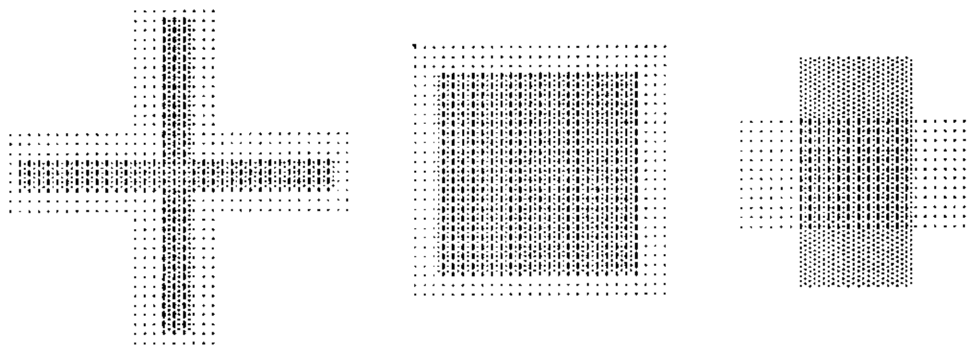
significant reduction in parasitic capacitance due to the near zero gate to source/drain overlap. A thick layer of oxide containing $P_2O_5$ is deposited over the surface of the wafer. This layer is reflowed for better coverage of the steps in the surface and a layer of photoresist is applied. *Contact hole* areas are defined using the contact cut mask and the oxide is etched away where the metal layer will contact the underlying features. After the resist is removed the source and drain are doped again (this is to prevent a phenomenon called *spike-through* -- essentially shorting of the aluminum contacts and the substrate through the shallow source and drain regions). A layer of aluminum is evaporated onto the surface of the wafer, followed by the application of more photoresist. Exposure (and subsequent development) through the metal layer mask leaves resist protecting the metal runs and contacts. The uncovered aluminum is etched away and the resist is removed (figure 3.2.1d). The wafer is then *annealed* (heated at a low temperature) to remove radiation damage resulting from the electron beam which is used to heat the aluminum during the evaporation process.

A thick layer of oxide is deposited on the entire surface of the wafer to provide physical protection. Windows to the bonding pads are etched through this layer in another photolithography step using the *overglass layer* mask. At this point (figure 3.2.1e) the wafer is finished, ready to be broken apart, bonded and tested.

## 4. Nasty Details and IC Pattern Preparation

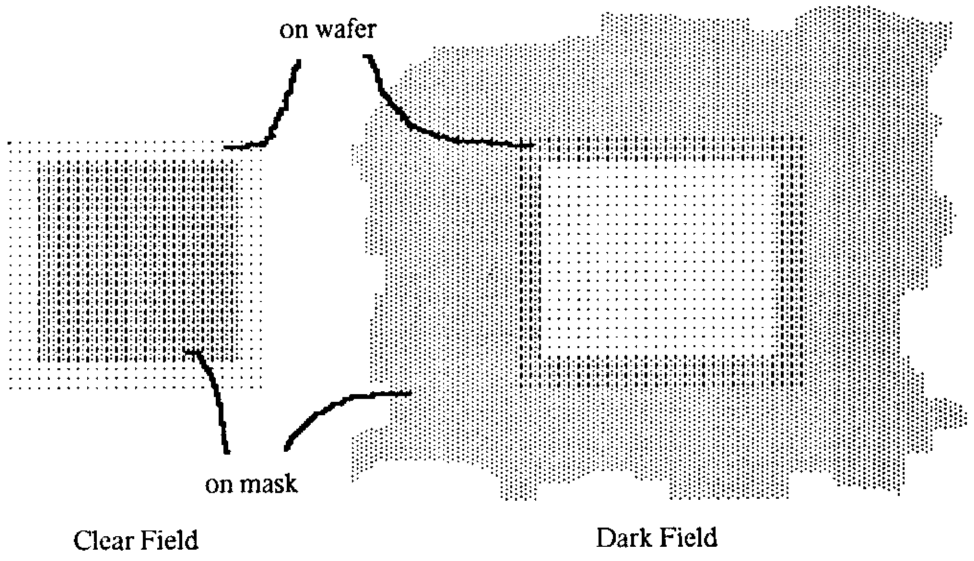
At some point the designer has several complete IC designs which are ready to be turned into chips. Before his design can be realized, several important details have to be taken care of which are not part of the actual circuit design process. Among these are the physical placement of several projects and test patterns on the multi-project chip plus the the addition of some extra features required by the fabrication line. Test patterns are useful for an evaluation of the quality of the wafer processing, for checking standard circuit parameters, as well as post mortem debugging should a chip fail to perform correctly. Most of these relatively fixed, universally required features (e.g. CD's, fiducial and parity marks, alignment marks and scribe lines) can be collected at each research site and grouped together in a *starting frame*. This starting frame provides a set of "symbols" (or whatever construct is appropriate in the local design system) which can be combined with the individual design projects to provide the masks for a complete multi-project chip.

The most important features that must be added to the net circuit is the set of alignment marks, which are needed to register subsequent layers on the IC with one another. Alignment marks take many forms :



-- but their purpose is the same. There is little magic in designing alignment marks; in fact, almost any reasonable features will do. However, a carefully designed set can mean the difference between good devices and those which are only marginal.

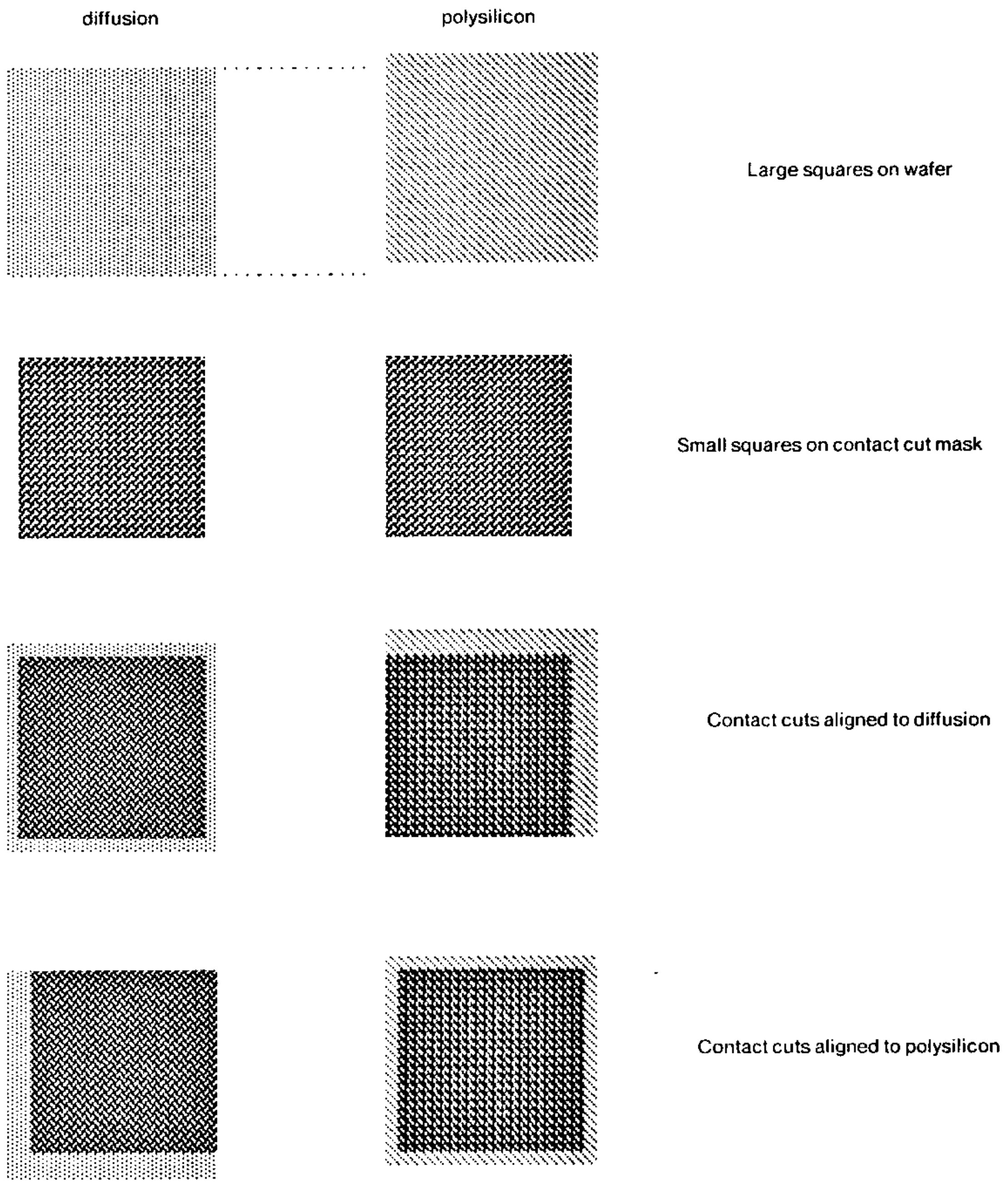
When the designer is deciding on which alignment marks to use it may help to consider the following scenario. The fabrication line operator puts a partially processed wafer onto the movable (x, y, rotation) stage of the alignment machine. The next mask is held over the wafer and the operator looks through a microscope from above. First of all, is it possible to locate the alignment marks? The designer can help by providing "black and clears" on which he has indicated the location of the alignment marks. A line or box enclosing the marks may also draw the operator's attention amidst the confusion of the other features. After the marks are located, is it possible for the operator to successfully align with them? Consider the case where the alignment marks consist of a large square on the wafer and a small square on the mask.



The operator is supposed to center the small square over the large one. This system is fine if the small box is opaque on a clear field (see the explanation of working plate polarity in chapter 5), but not when the small box is clear on an otherwise opaque mask. In the latter case the designer should have an alternate version of the alignment marks for opaque field masks, then one or the other set will work for the mask polarity used in the particular fabrication step. Another alternative is to design a set of marks which can be used regardless of mask polarity.

As the operator tries to line up the alignment marks, is it obvious which small square goes over which large one? A one square shift is certain to be disastrous. Some type of *gross* alignment mark should be provided to prevent shifting; again there are many alternatives -- an enclosing box or a simple square which is superimposed over one already on the wafer, or even numbering the small/large square combinations. Furthermore, it is important to eliminate ambiguity regarding the layer to which the current mask is aligned. For example, in figure 4.1 there are two large squares in place on the wafer, one in diffusion, the other in polysilicon. The operator has two small squares on the contact cut mask to line up over the two large squares. Unfortunately the large

Figure 4.1 Alignment of Contact Cut Mask


diffusion
polysilicon
Large squares on wafer
Small squares on contact cut mask
Contact cuts aligned to diffusion
Contact cuts aligned to polysilicon

ones are not exactly in line because of a small misregistration introduced in a previous step. The operator must decide whether to align to the diffusion or the poly feature, or perhaps to split the difference between the two. Whichever course the operator chooses may affect device operation. The designer should make this decision by providing only one pair of marks. (The actual set of alignment marks chosen for our starting frame are discussed in section 7.1.)

Ultimately the various individual designs and the starting frame have to be combined into a single IC description. This involves merging of a number of files, usually in a geometric design language, into a single file containing all of the integrated circuit designs. Fiducials and parity marks may need to be added outside of the area occupied by the designs and the starting frame. The chip pattern is repeated on the surface of the silicon wafer many times; 2", 3", and 4" wafers are commonly available -- a 3" wafer holds about 45 10mm by 10mm chips. *Exterior scribe lines* (see figure 4.2) are placed around the periphery of the area occupied by the project set. The purpose of the scribe lines is to provide a "lane" down to the silicon substrate in which the diamond-tipped scribe tool (see section 6.1) will ride. The wafer will be broken into chips (also called *dies*), as defined by the scribe lines, following fabrication.

The designs are arranged to minimize the area of the chip bearing in mind a number of important factors. Optical equipment limitations at the mask house make it difficult to generate masks for chips larger than 10mm by 10mm. Defect-free reticles become harder to generate as the chip size increases, thus it is disproportionately expensive to make masks as the chip gets large. The 10mm x 10mm size limit is somewhat misleading because of an additional restriction imposed by current packaging technology. The *cavity size* of a standard 40 pin dual inline package (DIP) is about 7.5mm x 7.5mm, thus projects should be limited to this size unless there is access to special packages. The 10mm x 10mm chip must be subdivided to meet this constraint by placing *interior scribe lines* between projects; these scribe lines must extend all the way across the chip (and thus across the wafer), that is, interior "tees" are not allowed.

*Yield*, the fraction of the IC's which function correctly, is greatly affected by chip size. As the *active area* (the area containing active devices but excluding empty space, bonding pads, etc.) grows the yield decreases geometrically. Typical yields for a 6mm by 6mm circuit, assuming standard defect densities, are about 20-40%; in industry, yields much below this figure are not acceptable profit-wise. Designers in a research environment may well be able to tolerate low yields since even a yield of a few percent gives the designer enough chips to verify his design, measure the performance and demonstrate feasibility.

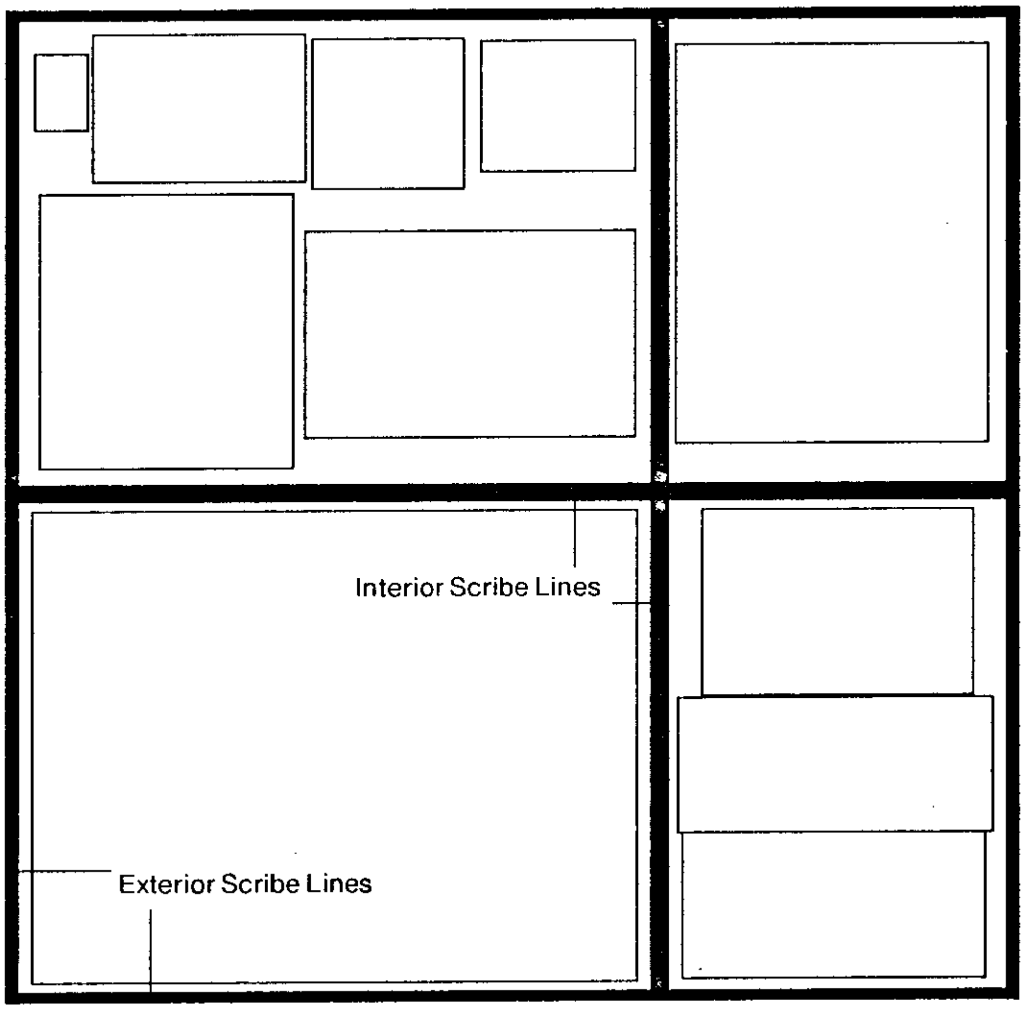


Figure 4.2. Overall View of a Multi-Project Chip

After the integrated circuits are fabricated one must determine whether or not they are functioning correctly. While the designer has the option of simply powering up his circuit and seeing if its input/ouput behavior is correct, a more satisfactory test method might use several *test structures* included on the chip. Simple structures like inverters can answer yes/no questions (Were the wafers processed to a minimum level of competence? Do individual transistors work?) and thus indicate whether more complete testing is warranted. More importantly, test patterns can provide information which is useful in determining why a batch of chips does not work, or performs poorly. Properly designed test patterns can show wafer processing problems, or eliminate this cause, narrowing the search to the area of design errors.

## 5. Mask Specification

When the chip has been laid out it must be converted into a format suitable for the pattern generator used by the mask house. This involves converting all of the shapes in the design file into rectangles and sorting them. The sorting order depends on the pattern generator; a penalty is paid in terms of the amount of time it takes to flash the reticle if the rectangles are out of order. Unfortunately the optimum order is based on a complex function which depends on mechanical considerations as well as the pattern being flashed; in general this function is not known to the designer. The moral is that unless the designer has detailed knowledge about the PG machine being used he is probably better off using a simple sorting algorithm (for instance lexicographic ordering based on what the particular PG machine is fastest at) than trying to second guess the pattern generator.

Before the PG tape can be sent off to the mask house some information must be obtained from the fabrication line regarding their process. The designer must tell the mask house the *polarity* of the working plates -- whether the plates for each layer should be *opaque field* (clear features) or *clear field* (opaque features). Typically a fab line will require a mixture of opaque and clear field plates, depending on the process step and the type of photoresist. The linewidths required by the process influence the choice of resist. More important, however, is the field area involved with the particular working plate. A speck of dust on an otherwise clear area of the working plate will cause a pattern to be made in the photoresist. If negative resist is being used the speck will make a hole in the resist which will enlarge somewhat due to undercutting in the subsequent etching process. Positive resist will leave a small dot where the speck was; this dot will probably be etched into oblivion. The fabrication line decides which of these factors to trade off in choosing the polarity of the working plates.

Varying etch conditions may cause the fab line to request that features on the masks for certain layers be altered by a constant amount (e.g. 0.5 micron around any border) in order to produce the desired dimensions on the silicon. The dimensional adjustments can be made in one of three ways:

> The circuit designer can be required to change his design to take into account the over- or under-etching at the fabrication line. This probably entails considerable work on the part of the designer each time the circuit is implemented on a different fab line, but has the advantage that the designer retains complete control of the layout geometry.
>
> Software could be provided to input the original design file and produce a new design file

which had the borders of features expanded or contracted in the appropriate way. This approach may require the use of complex algorithms in order to correctly modify the original file since minimum spacing design rules may be violated by enlarging adjacent features while gaps and discontinuities may be introduced by shrinking features which abut in the original design.

The mask house may be able to effect the changes by adjusting exposure time and other parameters in the mask generation process. This produces satisfactory results if the expansion or contraction is within the range attainable by the mask house.

In general it's a good practice to identify which layer each mask belongs to; the fabrication line may have specific codes that they wish placed on the masks for identification purposes.

Once all of this information has been collected and reduced to a PG tape and some written instructions the mask house takes over. When the working plates are returned they are passed in turn to the fabrication line along with more instructions. The total elapsed time for mask making and wafer fabrication can be 8-12 weeks. During this time the designer should be preparing for the day when the wafers are finished.

## 6. When The Wafers are Delivered...

The return of the finished wafers is an exciting moment for the IC designer. Proper preparation for this day can help to prevent frustrating delays in wafer separation and packaging, thus bringing the wafers to the testing stage more quickly. This, in turn, promotes rapid feedback concerning the success of the designs. The following discussion of wafer separation and chip bonding techniques is primarily aimed at those in a research, rather than a production environment.

### 6.1 Wafer Separation

The wafers, as returned from the fabrication line, are 3" disks of silicon which must be broken into chips the size of a DIP cavity. Wafer separation is accomplished in one of two ways: *scribing* or *sawing.*

Scribing is a simple operation similar to glass cutting. The wafer is held on a vacuum table (which is an integral part of the scribing machine, or *scriber*) and a diamond-tipped scribing tool is dragged across the surface within the confines of the scribe lines. The tool, where it slides over bare silicon, induces stress cracks in the vertical direction under the diamond tip. The pressure that the scribing tool exerts on the silicon is critical, too little results in random breakage in the fracturing operation, while too much produces stress cracks in the horizontal direction. Such cracks cause splintering of the wafer radially from the scribe lines, probably into active circuit elements. After each scribe line in the grid has been "scratched" in this fashion, the wafer (at this point it is still in one piece) is removed from the scriber and broken into chips. This may be achieved in a number of ways, for example by sandwiching the wafer in some soft material (e.g. rubber sheeting, filter paper), supporting it on a foam rubber block, and rolling a cylindrical bar over it. If the wafer was properly scribed the flexing force is concentrated at the scribe lines and the wafer fractures cleanly along the scribe lines.

Sawing is an alternative to scribing. In this technique a thin saw blade with an edge containing diamond-dust is used to cut approximately half-way through the silicon wafer. Again, the wafer is removed from the saw in one piece and fractured by techniques similar to the one outlined above.

Sawing offers several advantages over scribing. The saw can slice anywhere on the wafer, thus no scribe lines are needed. This allows dense packing of projects on a multi-project chip; when the wafers are returned from fabrication each designer can have a wafer sliced up without regard to the location of other projects on the wafer (i.e. by sacrificing neighboring projects to the saw blade). Sawing also leaves square edges after fracturing which makes manipulating the chips with tweezers an easy task.

Disadvantages include the necessity of removing the silicon dust (*slurry*) generated in the sawing process -- this means an extra cleaning step following wafer separation. More importantly, sawing equipment is complex and expensive ($12,000-$20,000) compared to scribers ($2,000-$5,000). There is also more maintainance required and more setup overhead involved. At this time it seems that scribing is a more economical, less finicky approach to wafer separation, especially in an environment where only a few wafers are handled each month.

## 6.2 Chip Bonding

Once the wafers are fractured into chips only *bonding* remains before they are ready to be tested. Bonding encompasses two different operations, *chip attachment* and *wire bonding*. In the first operation the chip is permanently affixed to the IC package; the second involves connecting the aluminum pads on the chip to posts surrounding the package cavity. These posts are connected through the package to the external pins.

Chip attachment is a straightforward process, especially in a low volume research environment. The chip must be solidly attached to the mounting pad (*header*) in the package cavity. The bond should exhibit low thermal resistance and make good electrical contact with the silicon substrate. Common means of attachment include *solder* bonding (both header and chip must be heated to the melting point of the solder used) and *eutectic* bonding (usually utilizing a gold-silicon alloy, see [Glaser 1977]). By far the most convenient for the researcher is *epoxy* bonding: the backside of the chip is dabbed with a commercially available gold/epoxy mixture and then pressed onto the header. Tweezers suffice for handling the chips. The header and chip are baked at a low temperature for a few hours to cure the epoxy and the assembly is ready for wire bonding.

Both of the manual wire bonding techniques in widespread use require considerable skill on the machine operator's part. *Thermocompression* bonding relies on pressure and heat to produce a strong bond. Typically a gold ball (on the end of a fine gold wire) is squashed against the aluminum bonding pad on the chip. The header and the capillary holding the wire are maintained at about 300 degrees centigrade and the bond is formed in a fraction of a second. As the capillary is withdrawn from the bonding pad, wire is automatically payed out; the operator maneuvers the capillary over the desired post and the wire is mashed against it, forming the second bond. As the capillary is backed away, the wire is cut by a gas flame, which simultaneously forms a gold ball for the next bond.

*Ultrasonic* bonding utilizes aluminum wire and ultrasonic energy to make bonds. The aluminum wire is pressed against the bonding pad and a short burst of ultrasonic energy locally heats the wire/pad interface so that a bond is formed. Similarly, a second bond is made on a post, and the wire is cut, usually by mechanical means. The header may be heated to assist the bonding process.

Ultrasonic bonding offers low materials cost but is less flexible than the thermocompression technique, which allows "daisy-chaining" of connection points. Thermocompression also gives more freedom to choose the angles at which wires leave the bonding pads, enabling some further flexibility which may be needed in a research environment.

In general, research chips need not be hermetically sealed in their packages, often a piece of tape over the cavity (or no cover at all) will prove adequate. Users should be aware, however, that MOS circuits exhibit very different device characteristics when operated in light.

## 6.3 IC Testing

*[section contributed by Peter Dobrowolski, UC Berkeley]*

Although IC's are quickly becoming cheaper and easier to use, their complexity, and the difficulty of testing them, is increasing quickly, too. It is important for both IC designers and users to consider testing in some detail.

The process of IC design may soon resemble the programming process. Programs are written, tested, rewritten and retested. This iteration continues until correct and perhaps optimized software is produced. Since it isn't absolutely necessary that a program work the first time, it can be written somewhat less conservatively, with an emphasis on creativity rather than cautious restraint. Once the procedure for producing IC's from symbolic logic diagrams is streamlined, IC designers will be able to apply the same principles that programmers use now. Quick and effective testing and debugging of prototype chips then becomes a crucial issue for this experimental approach, which is much different from today's industrial environments which must, of necessity, be more conservative, than research groups and individuals who quickly want to produce a few IC's to try out some novel ideas.

Testing involves such concepts as:

| | |
|---|---|
| functionality | Does the IC work as it should? |

*if yes:*

| | |
|---|---|
| quality | How well does the IC perform? |
| reliability | Will the IC always work as it should? |
| testability | How easy is it to determine functionality, quality, and reliability? |

*if no:*

| | |
|---|---|
| reason for failure | Bad processing run<br>Bad mask set<br>Misplaced or misshaped features in the layout<br>Logic errors at the circuit level |

These questions should be kept in mind during all phases of the the design process, and during evaluation of the finished chips.

Assuming that the reader will participate in all the stages of IC design, this section provides some guidance for the testing process by dividing it into two parts: *defensive design* and *systematic testing.*

### *6.3.1 Before ICs are made - defensive design*

In order to facilitate IC design, a number of simple rules should be followed:

*Observe the design rules.* The chip may work even if a design rule is broken, but the odds are against it.

*Keep your chip size within reasonable bounds.* The smaller the better. A typical maximum size (for 1978) should be about 6 millimeters on a side. Yield decreases sharply with increasing circuit size and with design rule violations.

*Include test patterns on your wafer.* Test patterns consist of simple structures such as single transistors or *ring oscillators*; examples are described in detail in section 7.2. Test patterns can be used to determine the process and circuit (device) parameters. For example, we can determine the basic inverter stage delay by taking measurements on a ring oscillator test structure.

*Think about how you will test every module you design.* When you design an LSI cell or subsystem, think through how you will test it, just as in writing a subroutine you should plan some tests to check its correctness before using it in a larger program.

*Consider input/output.* Use a *lightning arrestor* circuit to protect your input pads against damaging overvoltages from static discharges. Outputs should be buffered to enable them to drive capacitive loads of up to 50pf. Standard input and output structures are described in Appendix E.

*Provide access to internal paths of the circuit.* In the event of a malfunction it may be desirable to access internal nodes for debugging. Typically on experimental IC chips there is room for additional bonding pads. Not all of them have to be connected when the chip is mounted in a package. Several chips of the same IC can be mounted in different ways in separate dual in-line packages. Alternatively the test bonding pads can be accessed on the *wafer prober.*

*Consider the testability of the overall system.* It is very likely that the LSI module that you are designing is going to exist as a small part of a large system. Then it may not be sufficient that every module be testable by itself, but necessary that every module be *individually testable within a large system.* This is because modules, unlike subroutines, cannot be reproduced exactly. Certain flaws may depend critically on the working environment of the chip (loading factor, noise pickup) and can not be seen in a separate test setup. Providing the capability to detect and isolate, or even correct, defects in large systems is perhaps one of the most challenging tasks facing LSI system designers today.

*Provide self-testability on very complex structures.* Microprocessors and other related devices should contain testing algorithms in their *microcode store.*

### *6.3.2 After ICs are made - systematic testing*

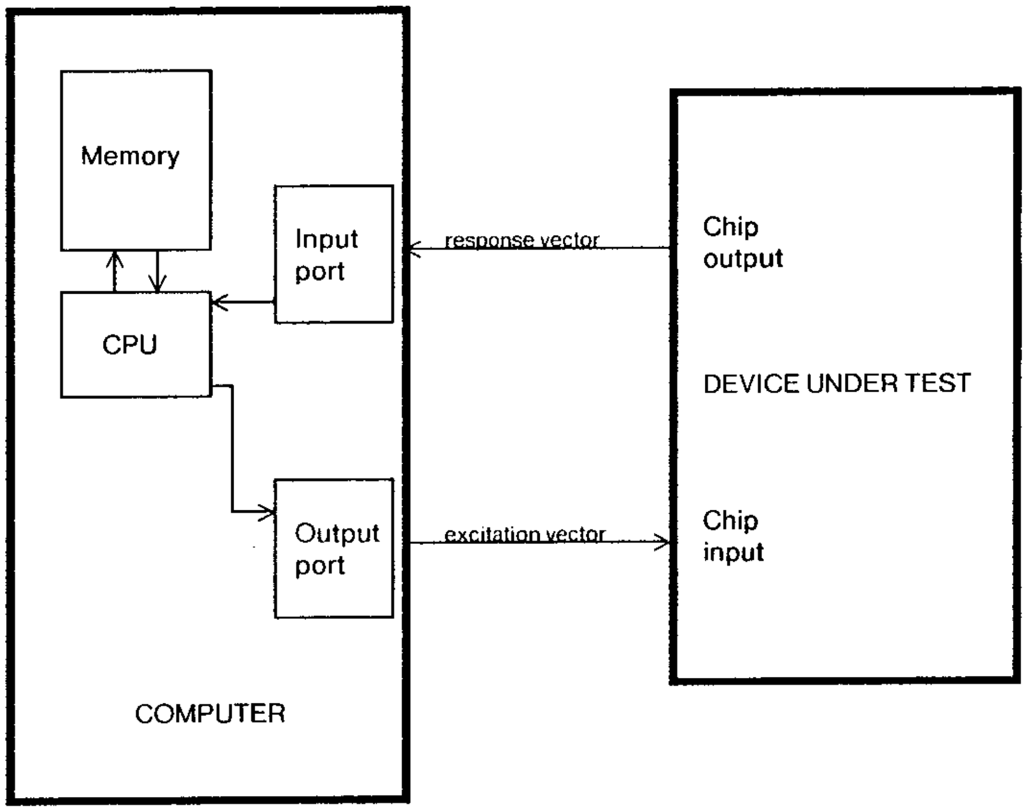
The first and most burning question is: *does the chip work?* When it doesn't, it is often possible to find the problem by looking at the chip under a microscope or by studying photomicrographs of it. If the chip is functional, one should test its performance and reliability (speed and power dissipation, for example).

Today's IC's are rarely simple enough to allow manual testing. Circuit complexity, the large number of inputs and outputs, and the multitude of possible states make manual testing prohibitively time-consuming. A better solution is to have a computer perform the tests (see figure 6.3.1). The IC is exercised by applying a properly defined *excitation vector,* provided by the computer's *output port,* to the inputs of the circuit under test. The *response vector* of the device under test can then be read by the computer's *input port* and compared to a stored correct response. The computer could be programmed to respond with an error message upon detecting deviation from the expected pattern. The test patterns must be carefully selected if they are to supply any information about the nature of the error. A more intelligent program may even do suitable branching dependent on the outcome of a few preliminary tests.

The excitation vector may need to be wider than the typical 8 or 16 bits in a word of the host computer; if so, use *multiplexing* to assemble successive words end-to-end in a vector of latches. Those latches should be contained on an interface board together with drivers and connectors.

The method just described requires a minimum of hardware in addition to the computer. Such a software-based testing system, however, might not be fast enough to capture some quickly

Figure 6.3.1
Software approach to IC testing. A computer is dedicated to exciting the device under test (DUT) and collecting the response. The disadvantage of this method is that it is slow and wasteful of computer time. Note that the computer is directly connected to the device under test.



changing response vectors, or it may have trouble exercising a complex device such as a microprocessor which may require some minimum data/clock rates for proper operation. Testing the speed performance of an IC also requires vector speeds which often can not be provided by a simple software system.

Higher I/O vector speeds can generally be obtained by moving more functions from software into hardware. Such a tester can use a semiconductor memory to store all excitation vectors. A counter is used to sequence the memory through the required words. The response vectors are simultaneously captured by another memory, and can later be analyzed at a slower rate. The tester is now a *peripheral* device to the host computer (see figure 6.3.2). The advantage of this method is two-fold: it allows higher testing speeds and frees the computer to perform other tasks while testing is in progress.

One would like a tester to be flexible enough to test any conceivable digital IC. If we imagine the excitation vector for the DUT to be a set of *control words* emanating from a computer's *control unit*, and the result vector out of the DUT to act as a *condition vector* to this control unit, we can construct a tester based on the principle of a *microprogrammed* controller. This kind of a tester could be easily adapted to any task by simply changing the *microcode* (see figure 6.3.3). A tester based on this principle can exercise very complex devices due to its inherent ability to make logical decisions based on some of the results. When the test is concluded, relevant results are as before stored in a result RAM, and the host computer is signaled to fetch them.

Almost any computer or microcomputer can be used as the host. The only requirement is that it have an accessible I/O port. The control unit for the tester could be built using one of the fast bipolar *bit-slice* microprocessors.

It should be emphasized that preparing for the day when the wafers come back from the fab line may be as large and complicated a task as the original design. The proper custom made interface board between the DUT and the test system has to be built and the test routines have to be written. In preparing these routines the designer should keep in mind the possibility that the chip does not work at all and plan a strategy to deal with this case.

In summary, testing is the responsibility of the designer and should be kept in mind from the early stages of the design process to the day of the delivery of a finished product to a customer. The availability of quick turnaround IC implementation permits large integrated systems to be designed modularly and hierarchically, somewhat like programmers now design large software systems. It is

Figure 6.3.2
Hardware approach to IC testing. The computer initializes an IC Tester which is connected as a peripheral device. The sequencer (counter) steps both RAMs, sending an axcitation vector and collecting the result vector. When the test is over, the sequencer interrupts the computer. This method is fast and requires little CPU time. Note that the computer is no longer directly connected to the DUT.

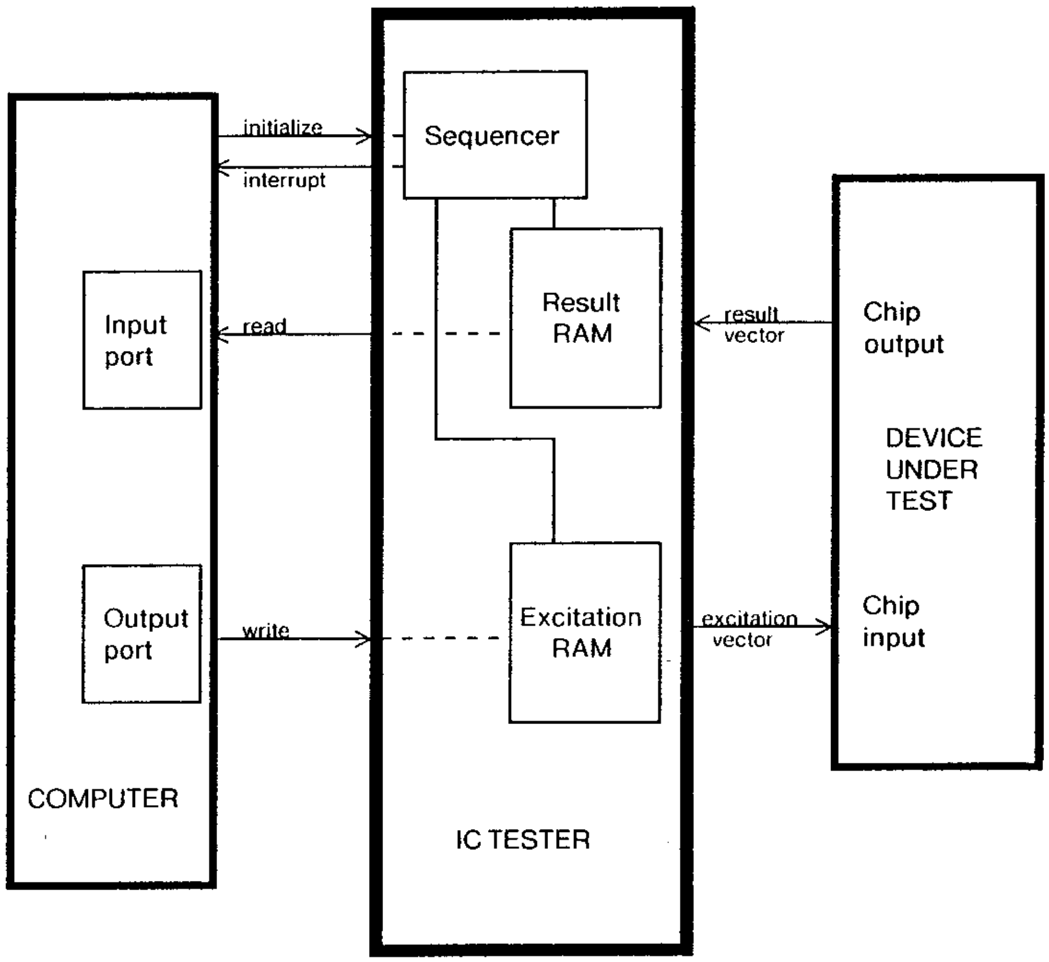
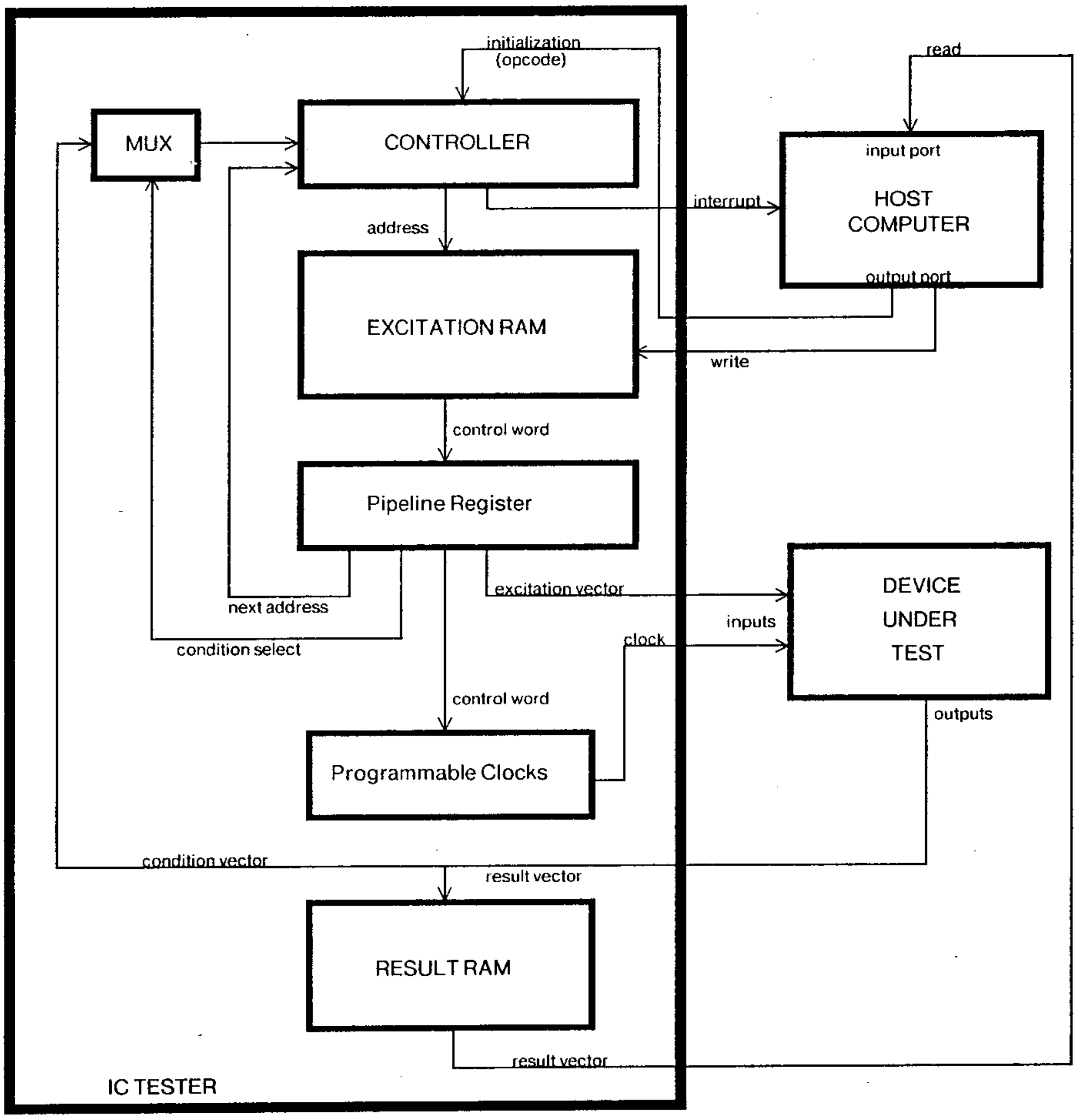


Figure 6.3.3
Block diagram of a Microprogrammed IC Tester. This approach allows maximum flexibility by being programmable. The computer initializes the controller which outputs the address of the first microinstruction to the microprogram RAM. The microprogram RAM supplies the next address and enables the controller to sequence through various testing microsubroutines.



clear that this iterative design loop will be closed when IC designers possess the necessary tools and practical knowledge for effectively testing the LSI modules they design.

# 7. An Example Project Chip and Starting Frame

The steps outlined in this guide have been applied to build an actual IC at Xerox PARC during the summer of 1978. Four summer students and four employees of Xerox PARC contributed designs, ranging from wafer processing evaluation testers to novel arithmetic and memory circuits. Ten such projects have been combined into a Multi-Project-Chip by Bob Hon. Figure 7.1 shows the overall layout and one mask layer of the chip. The chip includes a set of alignment marks and line-width testers laid out by Bob Hon and Dick Lyon and a general processing test chip laid out by Rick Davies. In this chapter the features of the starting frame are first presented, followed by a discussion of the general test chip. Finally in section 7.3, Robert Baldwin, the youngest student on the team, describes his experience as a novice who had to design his first IC without prior knowledge of the subject.

## 7.1 The Starting Frame

The multi-project chip is divided into two parts separated by one internal scribe line so that both parts are small enough to fit inside the cavity of a 40 pin DIP. Further, the layout is enclosed by exterior scribe lines placed around the periphery. The exterior lines differ from the interior lines in that the former are missing the outside "shoulder" (figure 7.1.1), the exterior lines of one pattern are completed by the overlap with the exterior line of the next chip. The scribe lines were designed simply to provide access to the Si substrate for the scribe tool to contact during wafer separation (see section 6.1).

The alignment marks designed were intended to unambiguously indicate which layers are to be aligned relative to each other. The marks consist of a number of "squares" and "fortresses".

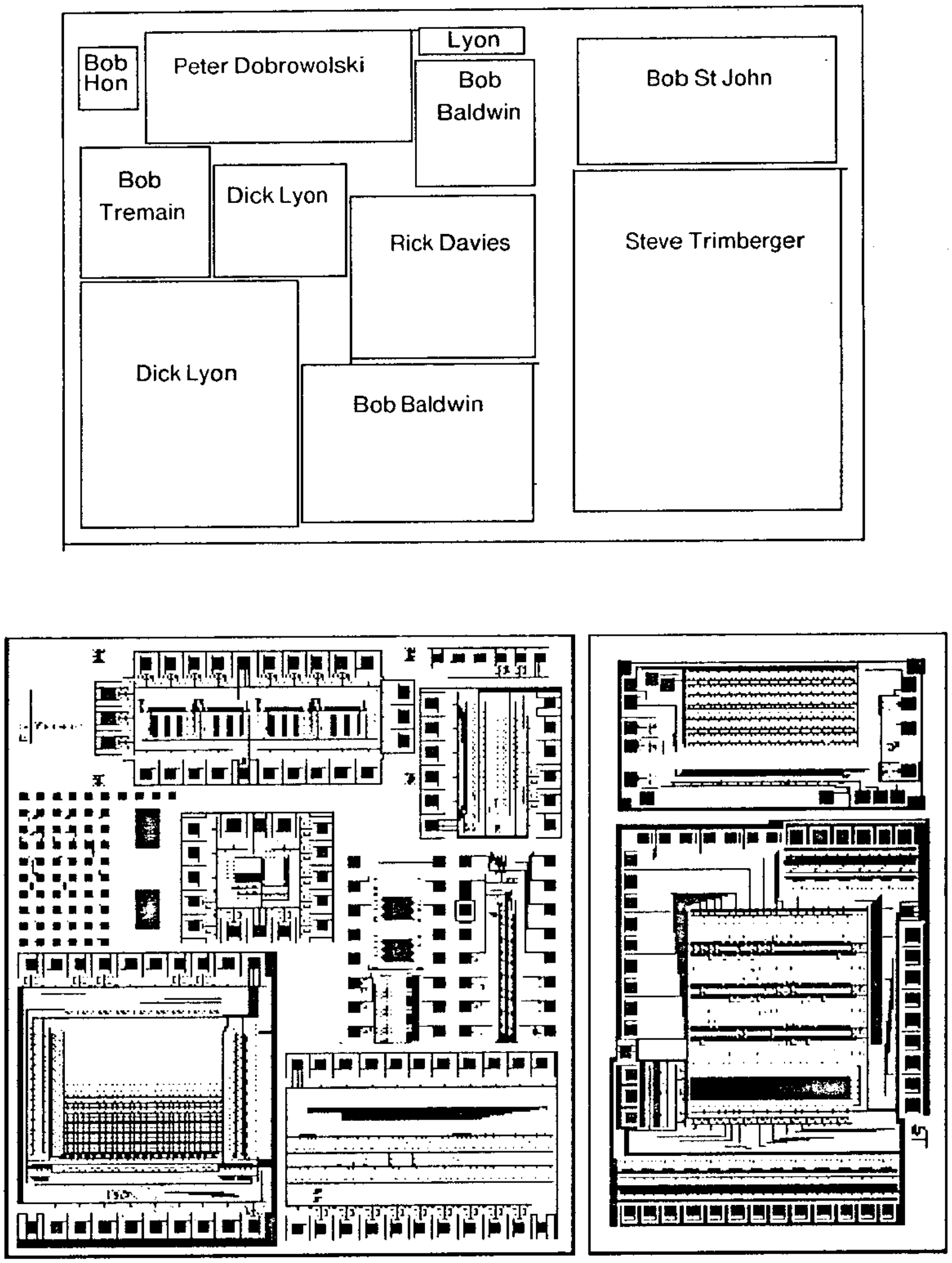
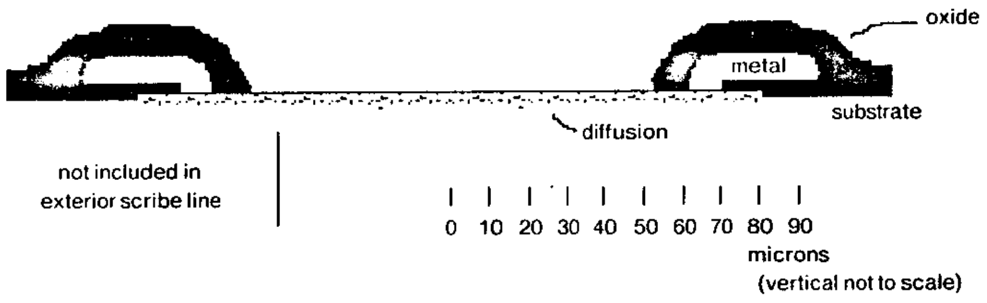
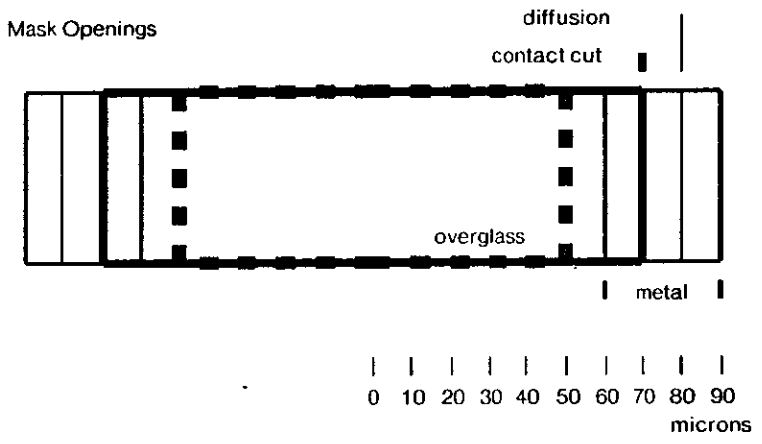


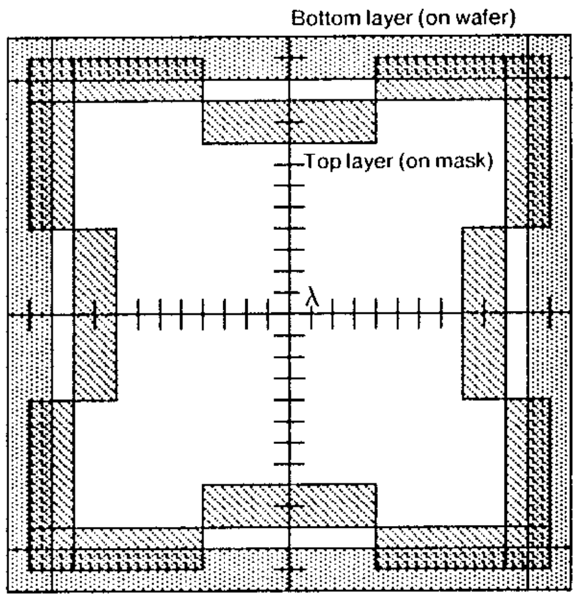
Figure 7.1. The Summer 1978 PARC Multi-Project Chip

Figure 7.1.1 Scribe Line Profile





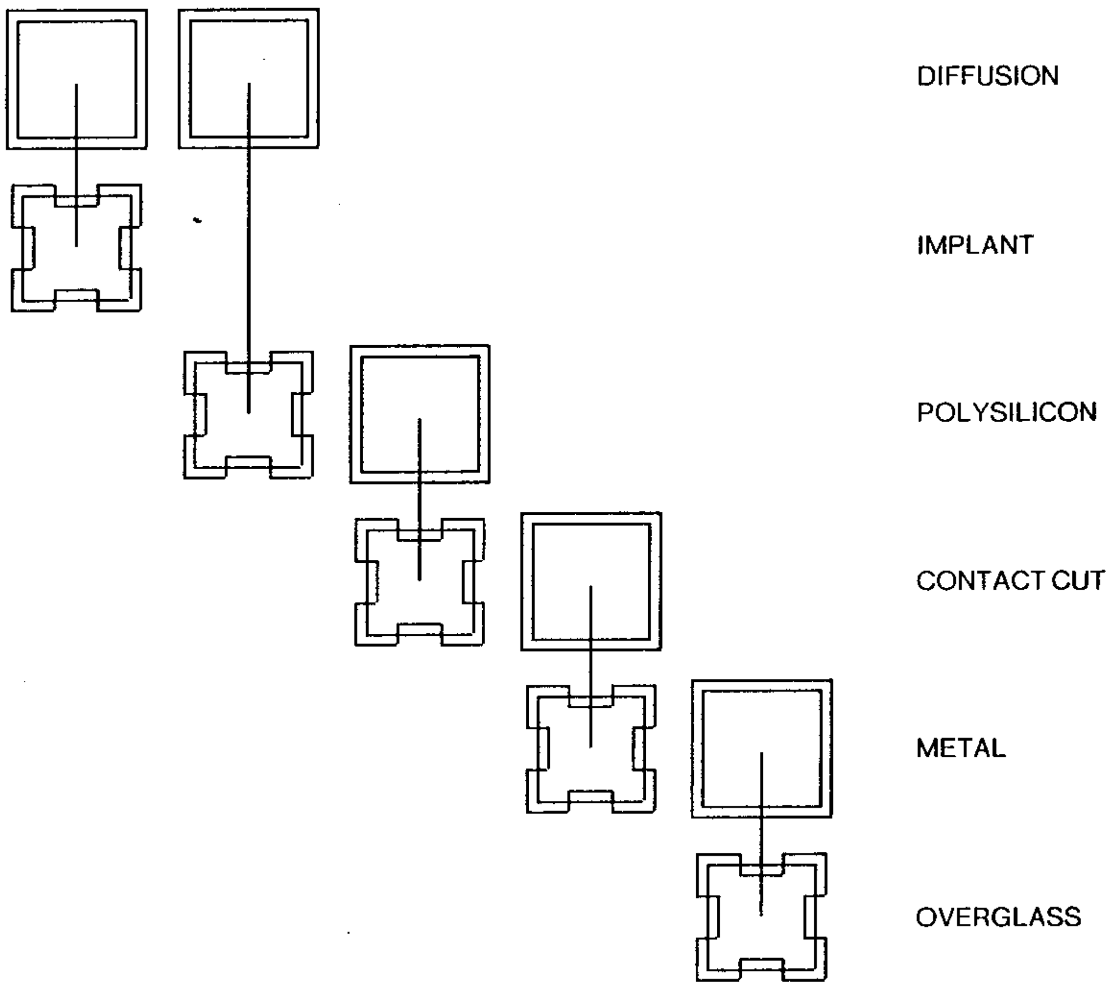
Alignment Mark

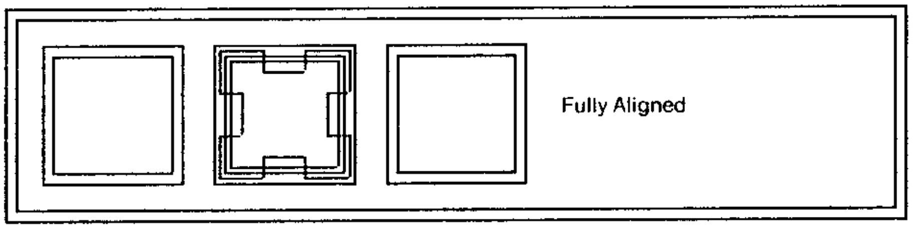


A square mark is placed on those layers which will serve as reference layers for masks in following fab steps. Each square has a corresponding fortress, located on a different mask, which will be aligned over it during the appropriate step (see figure 7.1.2). Each layer includes a large rectangle around the alignment marks to help the operator to locate them and to insure that the sequence is not shifted. The features are lines rather than areas, permitting the operator to align edges relative to one another. This makes the marks usable for clear as well as opaque working plate fields.

The fortress/square pairs are used in a left to right progression; a digit (omitted in the figures below for clarity) is placed in each fortress to indicate when it is to be used. A fortress is always aligned over a square and there is never more than one fortress per mask. The alignment sequence has the depletion mode implant, buried contacts (when used), and the polysilicon layer both aligned relative to the diffusion layer. The contact cuts are aligned relative to the polysilicon since there appeared to be more tolerance to misalignment between contact cuts and diffusion. The metal aligns to the contact cuts and the overglass to the metal layer. The following illustration represents the way an operator might align the mask for the polysilicon layer to the wafer. The pattern on the partially processed wafer is:

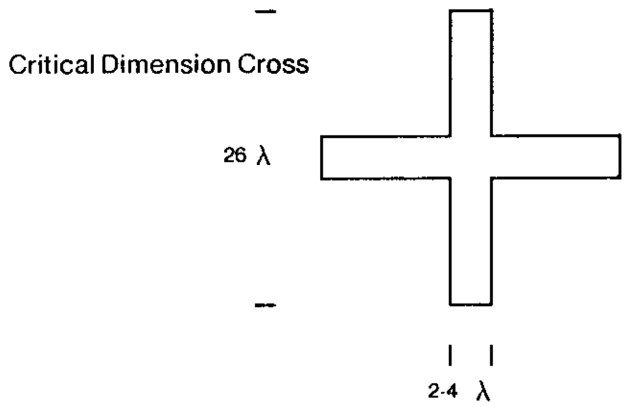
Figure 7.1.2 Alignment Marks for Mask Layers



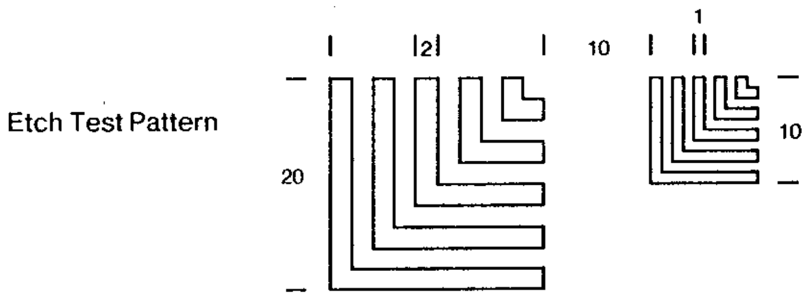


The third square will be used to align the contact cut layer; a fourth square is placed on the wafer during the contact cut step and will be used to align the metal layer.

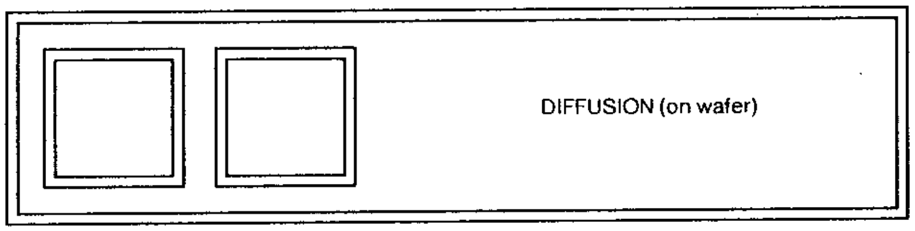
The critical dimension marks are simple crosses made of lines. The line widths vary from layer to layer, and are typical of the feature dimensions found on the particular layer.



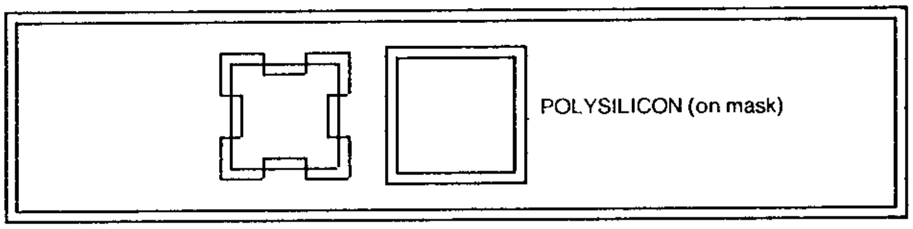
A set of features used to monitor the quality of the working plates and fabrication process was also included. This *etch test pattern* consists of a set of nested "L"'s with the same spacing between L's as the width of the feature; two different sizes were placed to check the quality of the mask and the quality of the photolithographic process.



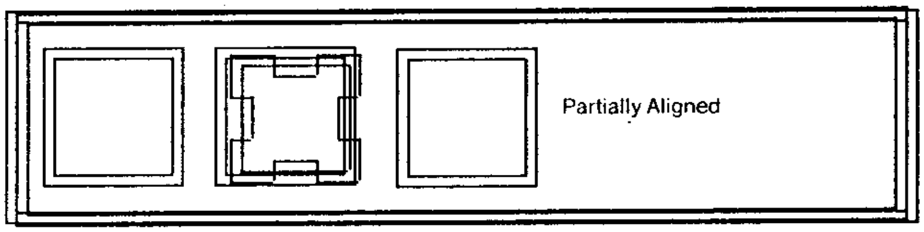
all dimensions in λ ( = 3 microns in 1978)



The operator must now align the polysilicon mask which contains one fortress and one square.



The diffusion and depletion implant steps are already complete, thus the first fortress/square pair has been used. The operator lines up the second pair using the gross alignment mark for guidance. The following diagram shows the mask in partial alignment with the features in place on the wafer.



Final corrections are made using the fortress/square pair.

Measurements on each mask layer provide a check on the dimensional correctness of the working plates while measurements on the wafer are used to verify that the fab line performed as anticipated (e.g. that lines over- or under-etched as expected).

Appendix A contains a copy of the information sent to the mask house.

## 7.2 Test Patterns

*[section contributed by Rick Davies, Xerox PARC]*

The starting frame contains a number of simple test structures to answer the following two questions.

> Was the wafer properly processed? Specifically are all the layers properly patterned, are gate oxide and deposited oxide films of acceptable dielectric integrity, are contact holes properly opened, etc.?
>
> What are the first-order device and circuit performance characteristics such as transistor threshold voltages, extent of short- or narrow-channel effects [Dennard 1974, Wang 1978], polysilicon sheet resistivity, and inverter propagation delay obtainable with the process?

The test structures described here are general enough so that it is assumed that they will prove useful to most participants in a multi-project chip. This should not deter any designer from adding his own special test structures. It may be desirable in the future to add additional patterns to the common test structure, to test such parameters as the quality of the buried contacts connection of diffusion to polysilicon or the limits of wafer processing (e.g. At what spacing do metal lines begin to show bridging?).

The test pattern consists of several separate regions which are described below. It occupies 2 mm x 2 mm and has this general layout:

```
*   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
*                                   *                       *
*          CV Capacitor             *      Test             *
*                                   *      Devices          *
*   *   *   *   *   *   *   *   *   *                       *
*                                   *                       *
*              Layer                *   *   *   *   *   *   *
*              Tester               *                       *
*                                   *                       *
*                                   *                       *
*   *   *   *   *   *   *   *   *   *      Shift            *
*                   *               *      Register         *
*   Ring            *   Contact     *                       *
*   Oscillator      *   Hole        *                       *
*   of Inverters    *   Tester      *                       *
*                   *               *                       *
*   *   *   *   *   *   *   *   *   *   *   *   *   *   *   *
```

*7.2.1 Layer Tester (Fig. 7.2.1)*

This is a long serpentine metallization path that runs between two interdigitated metal combs and lies over a serpentine of polysilicon and active transistor area. It tests the following features.

a. *Metal bridging.* There is a 25,000 $\mu$m periphery of minimum-spaced metal lines (9 $\mu$m, 3$\lambda$ spacing) between the serpentine and combs. Conductance between the serpentine and either comb indicates bridging caused by failure to properly image the pattern in photoresist and then etch it in the aluminum.

b. *Metal step coverage.* The 12 $\mu$m-wide serpentine passes 266 times over a 6 $\mu$m-wide region of polysilicon, over gate oxide, between 6 $\mu$m-wide source-drain diffusions. This should provide a good indication of step-coverage quality (assuming that the above bridging test passed), measured as a low end-to-end impedance ($< 100\Omega$). Metal running over diffusion and polysilicon is the worst-case condition for metal step coverage; the limited solid angle provided by the evaporation source can make it difficult to transport aluminum to the vertical features in small-geometry device structures.

c. *Gate-oxide dielectric integrity.* The presence of approximately 100,000 $\mu m^2$ of polysilicon over gate oxide provides a test for pinholes and shorts between gate and transistor-channel area. This is equivalent to the gate area of about 1000 transistors of typical transistor geometry. To pass this test there must not be measurable conductance between the polysilicon and the diffusion.

d. *Deposited-Oxide dielectric integrity.* The presence of approximately 100,000 $\mu m^2$ of metal over active area (polysilicon gate or source-drain diffusion) provides a test for

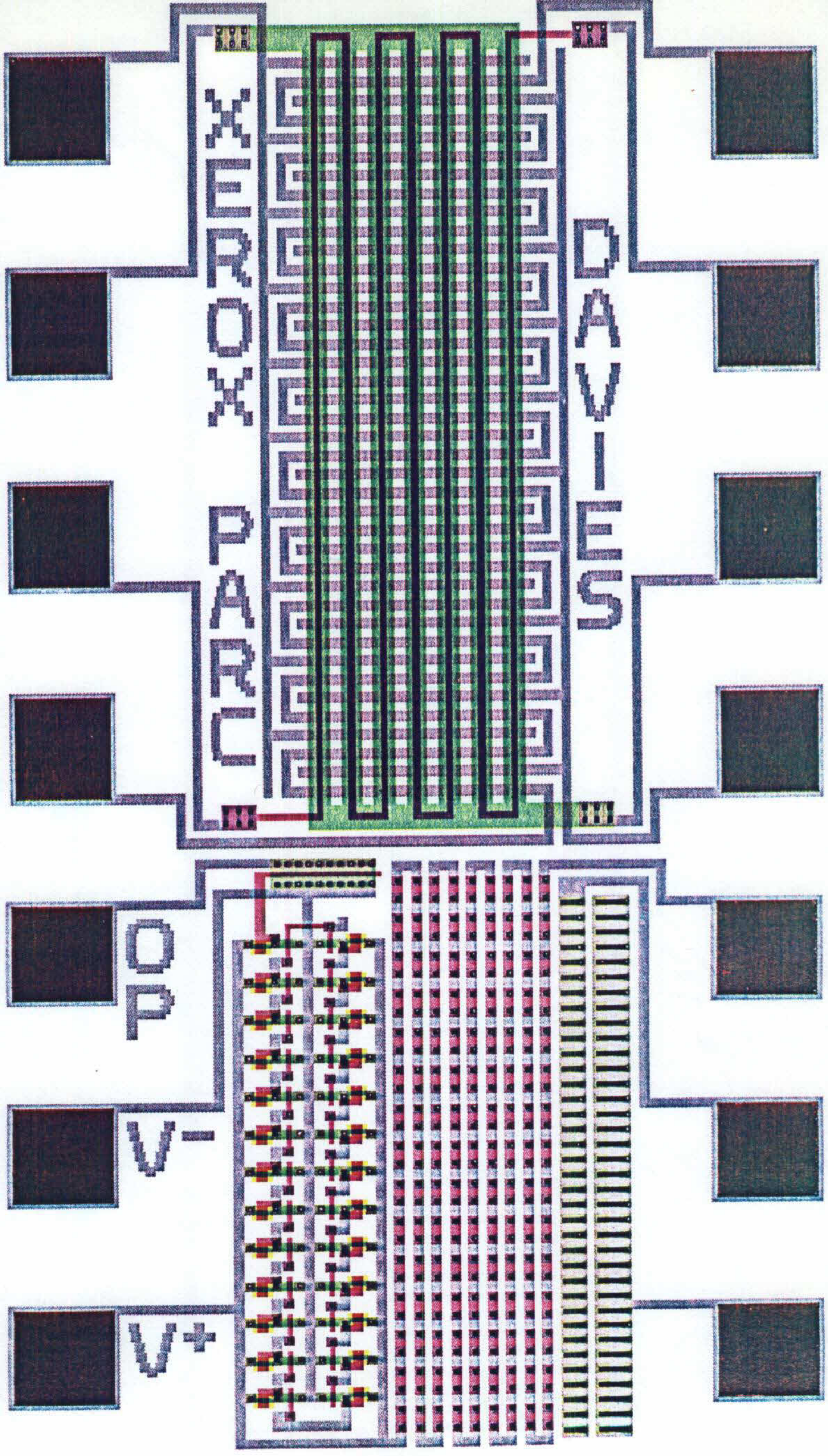


Figure 7.2.1 Test Structures

pinholes and shorts to the metallization; this could result from improper annealing of the metal causing spiking through the deposited oxide layer. No conductance should be observed between the metal and either diffusion or polysilicon.

e. *Polysilicon and aluminum sheet resistivity.* Although the most accurate resistivity measurement uses a Van der Pauw structure (separate forced-current and sensed-voltage terminal pairs), the present structure provides a quick estimate by inspection of the end-to-end resistance. About 700 squares are present in either level.

### *7.2.2 CV Capacitor.*

A 200$\mu$m x 900$\mu$m MOS capacitor with a diffusion guard-ring is provided for analysis of the process parameters $Q_{SS}$ (density of fixed charges at the oxide-silicon interface), $N_{SS}$ (density of trapping states at the interface) and the gate oxide thickness [Grove 1967]. A minor amount of final wafer preparation may be required to form a suitable ohmic backside substrate contact for reliable measurements.

Measurements on this structure would allow separate determination of the implant dose and interface characteristics components of the threshold voltages of the enhancement and depletion NMOS transistors. This structure may also be used to test gate oxide dielectric integrity; 180,000 $\mu m^2$ are present.

### *7.2.3 Contact Hole Tester*

The following two contact-hole tests are incorporated on this test pattern (Fig. 7.2.2).

a. A series connection of 270 metal-polysilicon contacts (6$\mu$m x 6$\mu$m) tests for failure to make electrical contact between metallization and the underlying layer. A measured resistance significantly above the expected impedance corresponding to the parasitic 135 squares of connecting polysilicon indicates poor quality ohmic contacts. This could be caused by improper imaging of the pattern in the photoresist (in particular a scum residue might have been left in the bottom of a hole), improper etching of the oxide, or by metal breakage around the rim of the etched contact hole.

b. A special tester consisting of 2 columns of 36 metal-diffusion contacts compares contact holes which are properly centered over a diffused area and others which overlap the field oxide region. The latter is used to test whether one could overlap contact holes onto the field oxide, in order to save the area otherwise consumed by alignment tolerance. Because the phosphorus-doped $SiO_2$ (sometimes called *P-glass*, see section 3.2 for a description of the fabrication process) deposited before contact hole definition etches much faster than does the thermally grown field oxide, the contact holes should open before the field region is etched through.

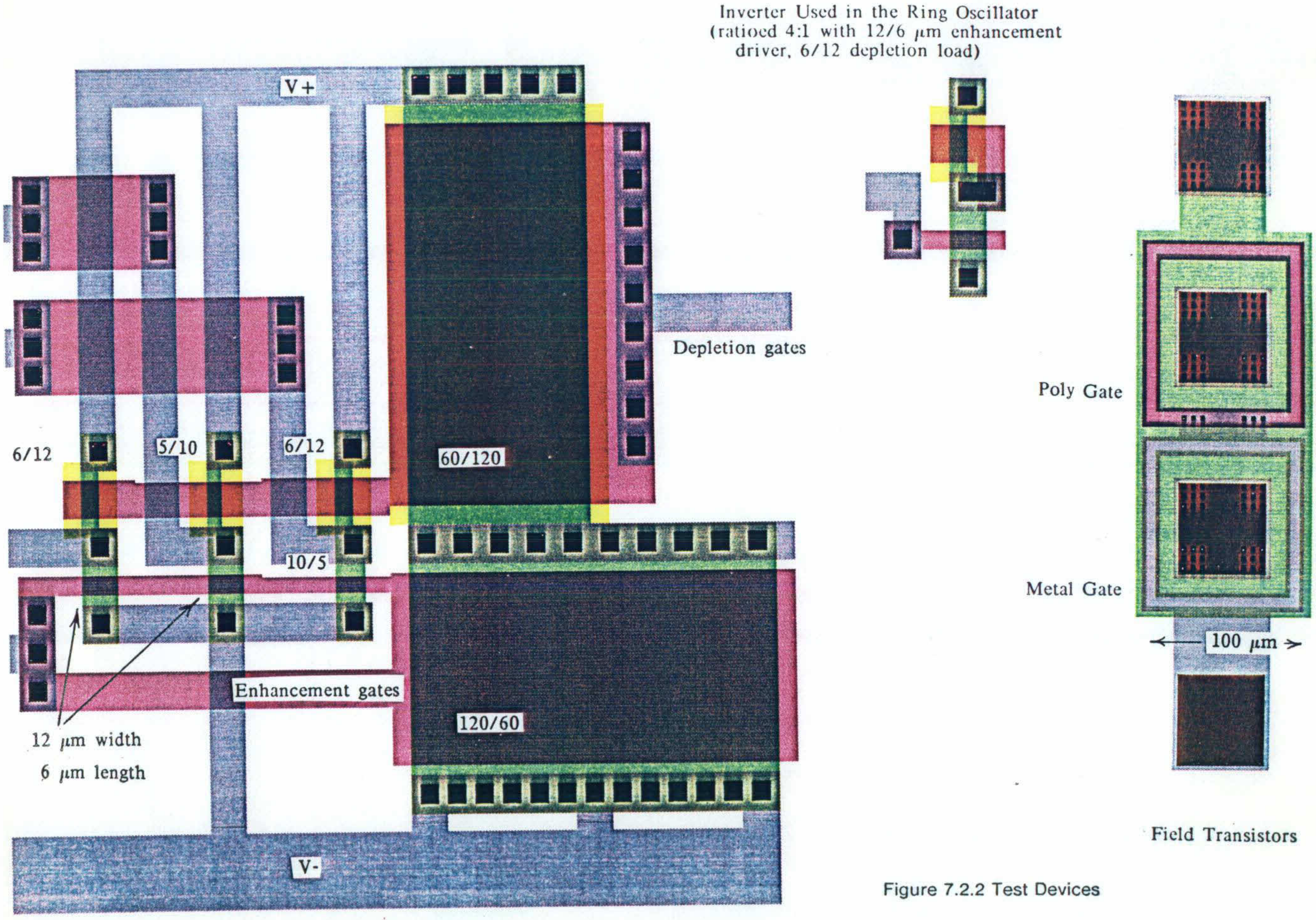


Figure 7.2.2 Test Devices

The two columns have identical bottom-wall diffusion area, diffusion periphery, and contact hole area over diffusion; the right one differs from the left only by the incorporation of a strip of field oxide in the middle of the contact hole. If the two columns reveal the same leakage characteristics to the substrate, then this overlap technique is probably acceptable.

### *7.2.4 Discrete devices.*

Four enhancement and four depletion mode transistors are provided for dc testing (Figure 7.2.2). They are organized as four inverters for convenient transfer curve analysis, with uncommitted gates for the depletion-mode transistors to allow full testing of those devices. To minimize the number of bonding pads, the enhancement-gates are shared, as are the depletion-gates; all enhancement-sources and all depletion-drains are also shared. The four inverters are:

a. 12$\mu$m-width/6$\mu$m-length enhancement NMOS with 6/12 depletion NMOS, forming a standard 4:1 inverter with 2$\lambda$ layout rules.

b. 12/6 enhancement and slightly narrowed 5/10 depletion load device. One expects a higher threshold in the narrowed channel because the channel potential is raised by the increased influence of edge effects. The use of scaling [Wang 1978] (which involves altering the fabrication process) would permit this smaller layout without disturbing the dc characteristics.

c. 10/5 enhancement and 6/12 depletion load devices. Short-channel effects should cause a lowering of the threshold voltage and produce increased output conduction in the enhancement device [Dennard 1974].

d. 120/60 enhancement and 60/120 depletion load devices. These devices should permit one to check device characteristics with little interference from peripheral effects.

Two transistors with closed layouts -- one with a metal and the other with a polysilicon gate -- on thick field oxide, are provided to test for isolation-region channeling or other parasitic leakage. A threshold voltage above about 25v should exist on each device. In both cases the transistor gate electrode overlaps the source and drain regions. The poly-gate structure makes gate-oxide devices at source and drain that are in series with the field-region under test.

### *7.2.5 Ring Oscillator of Inverters.*

This structure (Fig. 7.2.1) tests ac device performance in a probe environment without hindrance from the inherent parasitic capacitances. Because it is an actual circuit, it should provide more reliable and directly usable information than making separate detailed ac device measurements and then using circuit equations. The ring oscillator is 25 stages long to provide a low frequency output signal. The average propagation delay is one half of one twenty-fifth of the inverse of the loop natural oscillation frequency. Inverters with 12/6$\mu$m enhancement drivers and 6/12$\mu$m depletion loads are used; a buffer/inverter taps the loop, giving one of the 25 loop inverters a fanout of two. The buffer in turn feeds an output transistor with W/L of 20/1; this may be used as a common-source output driver or could be used as a source-follower if desired.

### *7.2.6 Shift Register.*

This 33-stage circuit is similar to the above ring oscillator, with the addition of a passgate in front of each inverter (they are 24/6$\mu$m enhancement drivers and 6/12$\mu$m depletion loads; the passgates are 6/6$\mu$m devices). The passgates are bussed in two phases that alternate between inverters (16 passgates per phase). The 33rd passgate is brought out to a separate bonding pad so that one can open the shift register loop. The complement of this signal is applied to another passgate which connects the shift register to a separate, inverting input buffer.

This configuration provides the following operations:

a. With both passgate phases and the control passgate line high, the circuit implements a 33-stage ring oscillator using passgate signal transmission. The measured average propagation delay may be compared to that of the simple ring oscillator (described in section 7.2.5).

b. With the control passgate line low and alternate clocking of the passgate phases, the circuit acts as a shift register which may be loaded with an arbitrary bit pattern. Raising the control passgate line while continuing two-phase clocking (at a rate below that of the loop self-oscillation frequency) forms a recirculating shift register.

## 7.3 An Example Project

*[section contributed by Robert Baldwin, MIT]*

The project illustrated in this section, named *AccumulatorTest*, was my first LSI design. It may be representative of what a beginner, who knows nothing about device physics or LSI, can expect to accomplish in three or four weeks. The organization of this section follows the design process chronologically. The four parts are: Conception, Refinement, Decomposition, and Implementation.

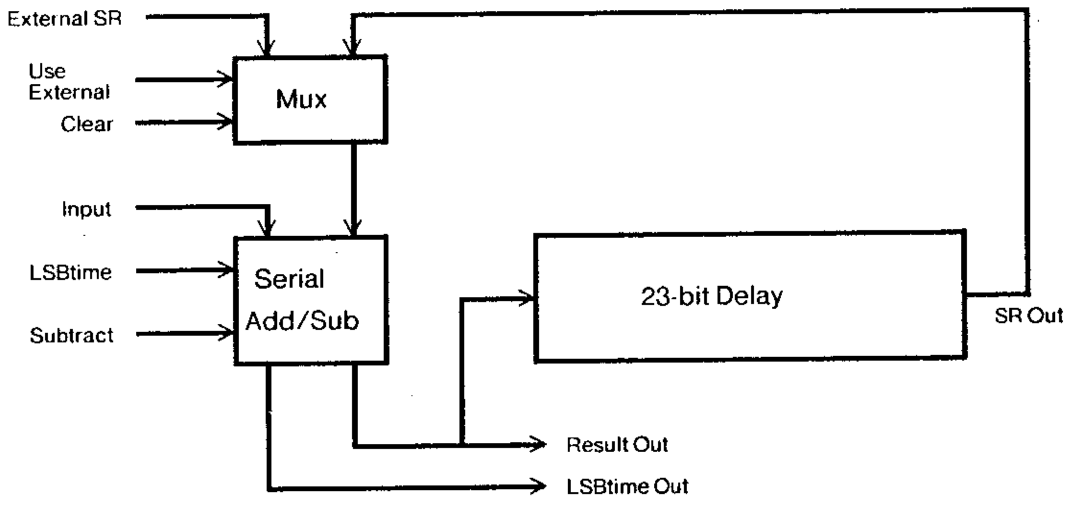
### *7.3.1 Conception and Refinement*

This project started out with the desire to test a serial Adder/Subtractor cell that Dick Lyon had previously designed, but not implemented. The A/S cell takes two numbers in serial form, LSB first, and produces a serial output, which is either their sum or their difference, depending on the mode control input. Internally, it saves the carry (which represents a borrow if subtracting), and adds or subtracts it from the next pair of input bits. The A/S needs to know when to reset the carry, so it has an *LSBtime* signal input that marks the time slot of the LSB of each input. In order to be able to cascade the A/S's it also outputs an *LSBtime* signal, which marks the LSB of the result.

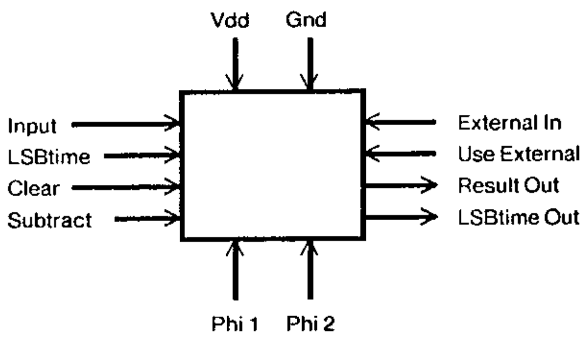
To enhance the usefulness of the test chip and reduce the external testing hardware required, I made one of the inputs the result of the last operation. All of the numbers are supposed to be 24 bits long (to conform to the format of another project), so a 23-bit shift register is needed between the A/S's output and one of its inputs (the A/S contributes one bit of delay itself). This leaves the chip with a single serial input, which can be added to or subtracted from the accumulator. In addition, there should be some way to initialize the accumulator to zero, thus a *clear* signal is provided.

Two more refinements were made to make testing easier. First, a multiplexer was added so that the A/S could be tested even if the shift register failed. Second, a circuit was added to synchronize the control signals (*mode* and *clear*); see section 7.3.3 (Implementation).

The whole chip has the following block diagram:



At the next higher level of abstraction, the chip has the following connections to the outside world:



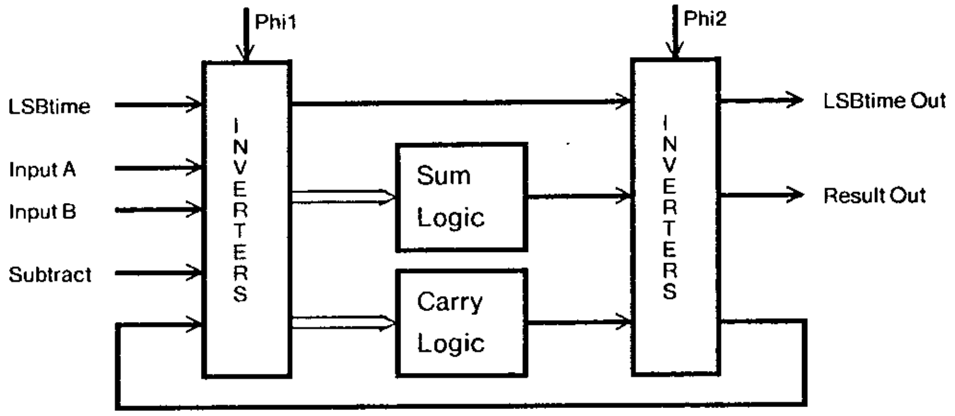
### *7.3.2 Decomposition*

The I/O was done through two columns of pads placed on either side of the project. Since the Cell Library (see Appendix E) contained pads and drivers, I didn't have to design them myself.

Each block in the above functional diagram needs to be decomposed until it is clear how to implement it in silicon. First the decomposition of the A/S will be given, then the SR, and finally the Mux.

The first thing that was decided about the adder was the timing. A non-overlapping two-phase clock is employed. All the computation is done during phase Phi1, and latched during Phi2.

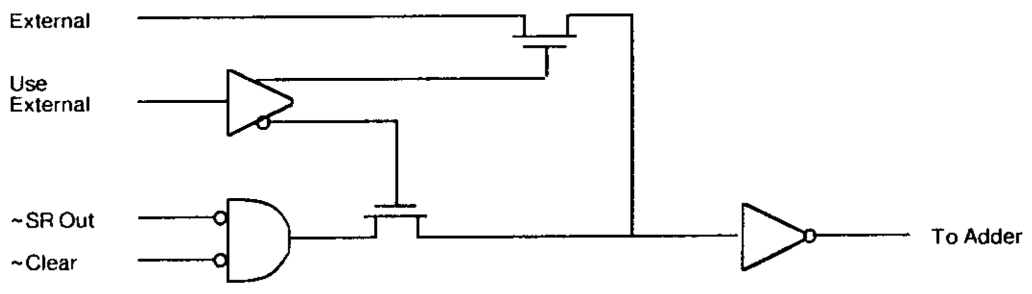
During Phi1 the adder computes the sum/difference bit which is used as the result, and the carry/borrow bit which is used during the next bit time. This suggested that the adder should be implemented with two logic blocks, and a set of double inverters to produce the true and complement of the inputs (a function block performs a three-input XOR function, and a 2-of-3 majority gate generates the carry). The outputs of these blocks are then latched by a second set of double inverters. Double inverters are used wherever possible in order to take advantage of existing library cells, and to reduce the number of cells used in the design. The A/S also computes *LSBtime Out* by delaying *LSBtime In* by one clock cycle (i.e., doubly inverted during phase1, and latched during phase2). The following diagram shows the general layout of the adder.



Thus the entire Adder/Subtractor can be layed out in a simple and regular way. For this project, layout optimization would have been pointless and wasteful of time. Often in such experiments it is appropriate to trade density and speed for shorter design time, increased reliability, and greater ease of modification. As in software optimization, only a few frequently used critical cells (subroutines) in the system need be optimized in order to realize most of the potential benefit.

To give the project an overall squarish shape, the shift register was folded so that the data weaves back and forth between the clock lines. This layout is composed of eleven and one-half double cells stacked on top of each other; each contains one complete weave (i.e. two bits of delay). I would like to acknowledge the help of Dick Lyon in laying out the final version of DoubleDelayCell, a descendant of the library's InverterPair and BackwardInverterPair.

The multiplexer selects either *SR Out* or *External In* to be used as one of the inputs to the adder. Its decomposition (which includes the logic for clearing *SR Out*) is best described by a circuit diagram:



The inverters are implemented using the library's InverterPair cell, and the NOR gate is implemented as an InverterPair with an alternate path to ground for the input node.

*7.3.3 Implementation*

Because library cells were used wherever possible, very little needs to be added about the implementation. The detailed layout of each cell can be seen in figure 7.3.1. The basic blocks were stacked on top of each other in the order: Mux, SR, and A/S.

All the control signals are synchronized with *LSBtime* by passing them through pass gates controlled by *LSBtime.*

The layout of power and clock lines was not random. I chose to run both power and clocks in vertical metal, because I knew there would be horizontal feedback signals in the A/S and the SR, which were most conveniently run in poly. Vertical power lines made it easy to design horizontal pullups, which in turn led to low wide inverters, suitable for vertical stacking. The spacing between the metal lines allowed inverters after each clock line, and a function block between clock phases.

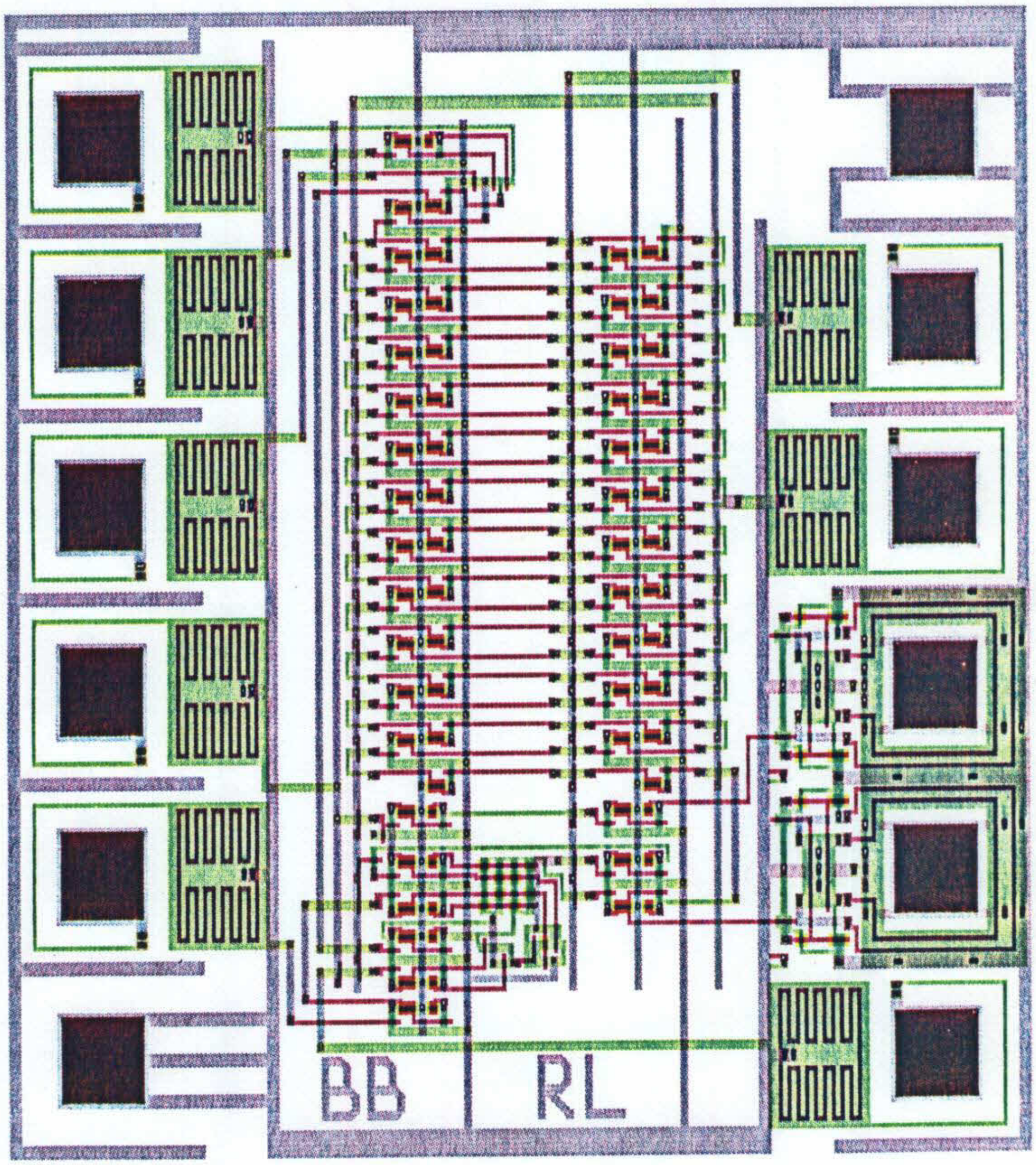
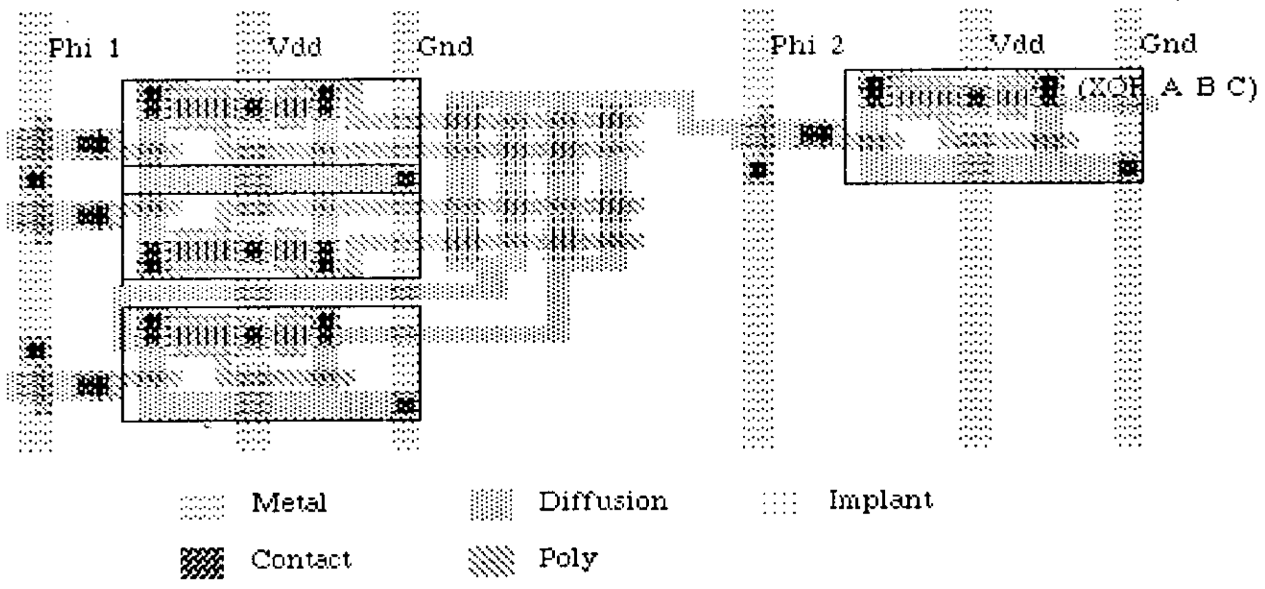


Figure 7.3.1 An Example Project: AccumulatorTest

The following example checkplot of the three-input XOR gate illustrates these ideas:



When checking the completed design, one major source of problems was found to be the mis-use of library cells. One such error was a misunderstanding of the interface to the library cell PadIn, which I had connected with poly instead of diffusion. Other bugs were due to treating library cells as black boxes, and wiring them up without checking for resulting design rule violations. Several times I forgot that butting contacts have metal over them, and ran other metal lines too close; for example, there is a butting contact in the InverterPair, which is only 2 lambda away from the connection point. I located these errors by plotting the project without the metal layer to check for poly-diffusion-implant violations, and without the poly and diffusion layers to check for metal-metal and metal-contact violations.

## Appendix A. Specifications Sent to the Mask House

**Date:** August 23, 1978
**Requestor:** Bob Hon 494-4324
**Project Name:** PARC-MPC

**Reticle Specifications**

NOTE: Please return all pattern generator output to the customer.

**Number of Reticle Sets:** 1
**Number of Reticles per Set:** 5
**Maximum Pattern Dimensions (outside scribes)** X: 9348$\mu$ Y: 6324$\mu$
**Step and Repeat distance** X: 9288$\mu$ Y: 6264$\mu$
**Reticle Magnification:** 10x
**Parity Marks on PG Tape?** No, please add as needed.
**Fiducials on PG Tape?** No, please add as needed.
**Blow Backs?** Yes (color) **Magnification:** 150x **Number of Sets:** 2
Please use the following colors:

| | | |
|---|---|---|
| DIF | - | GREEN |
| IMP | - | YELLOW |
| POL | - | RED |
| CUT | - | BLACK |
| MET | - | BLUE |

**Black and Clears:** Yes (8½" x 11") **Number of Sets:** 2

**Mask Specifications**

**Working Plate Material:** AR Chrome
**Working Plate Size:** 4"
**Pattern Size:** 3"
**Number of Working Plates:** 2 per level, except 3 of CUT level
**Master Plate Defect Density:** Standard
**Working Plate Specs:** (each reticle has the process step name in upper left corner)

| Process Step | Number of Flashes | WP Field | CD digitized width | Tolerance |
|---|---|---|---|---|
| DIF | 41783 | OPAQUE | 6.0$\mu$ | 0.5$\mu$ |
| IMP | 3741 | CLEAR | 12.0$\mu$ | 0.5$\mu$ |
| POL | 54184 | OPAQUE | 6.0$\mu$ | 0.5$\mu$ |
| CUT | 18331 | CLEAR | 6.0$\mu$ | 0.5$\mu$ |
| MET | 29764 | OPAQUE | 12.0$\mu$ | 0.5$\mu$ |

**Working Plate Labels:**

| Process Step | Working Plate Label (19 characters max) |
|---|---|
| DIF | PARCMPC 878 DIF |
| IMP | PARCMPC 878 IMP |
| POL | PARCMPC 878 POL |
| CUT | PARCMPC 878 CUT |
| MET | PARCMPC 878 MET |

**PG Tape:**

The files are on the PG tape in Mann 3000 format. The order is: IMP, DIF, POL, CUT, MET. A copy of the directory follows.

Directory (file 0):

```
WHOLECHIPZ0C00010081WHOLECHIPZ1C00020865WHOLECHIPZ2C00031095WHOLECHIP
Z3C00040342WHOLECHIPZ4C00050636$
```

Overall View of the MET layer

Critical Dimensions are in this area (see closeup on next page)

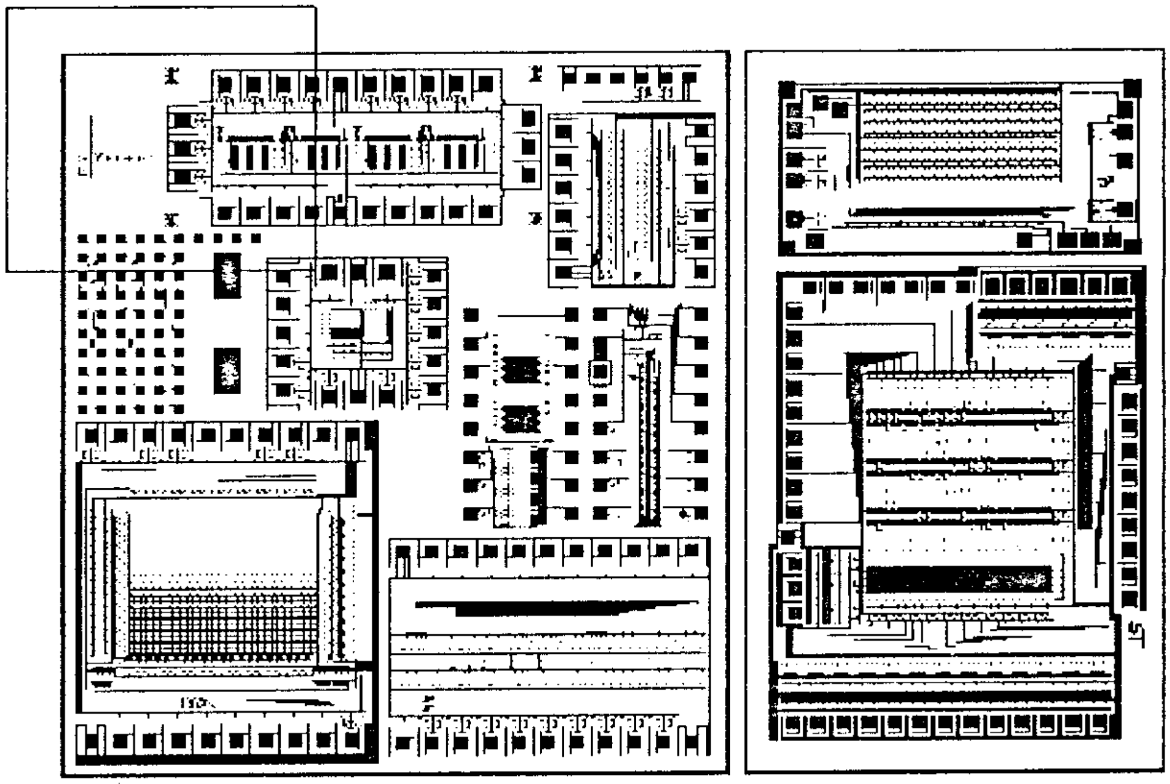


Figure A.1 Diagram of the MET Layer Showing CD Location

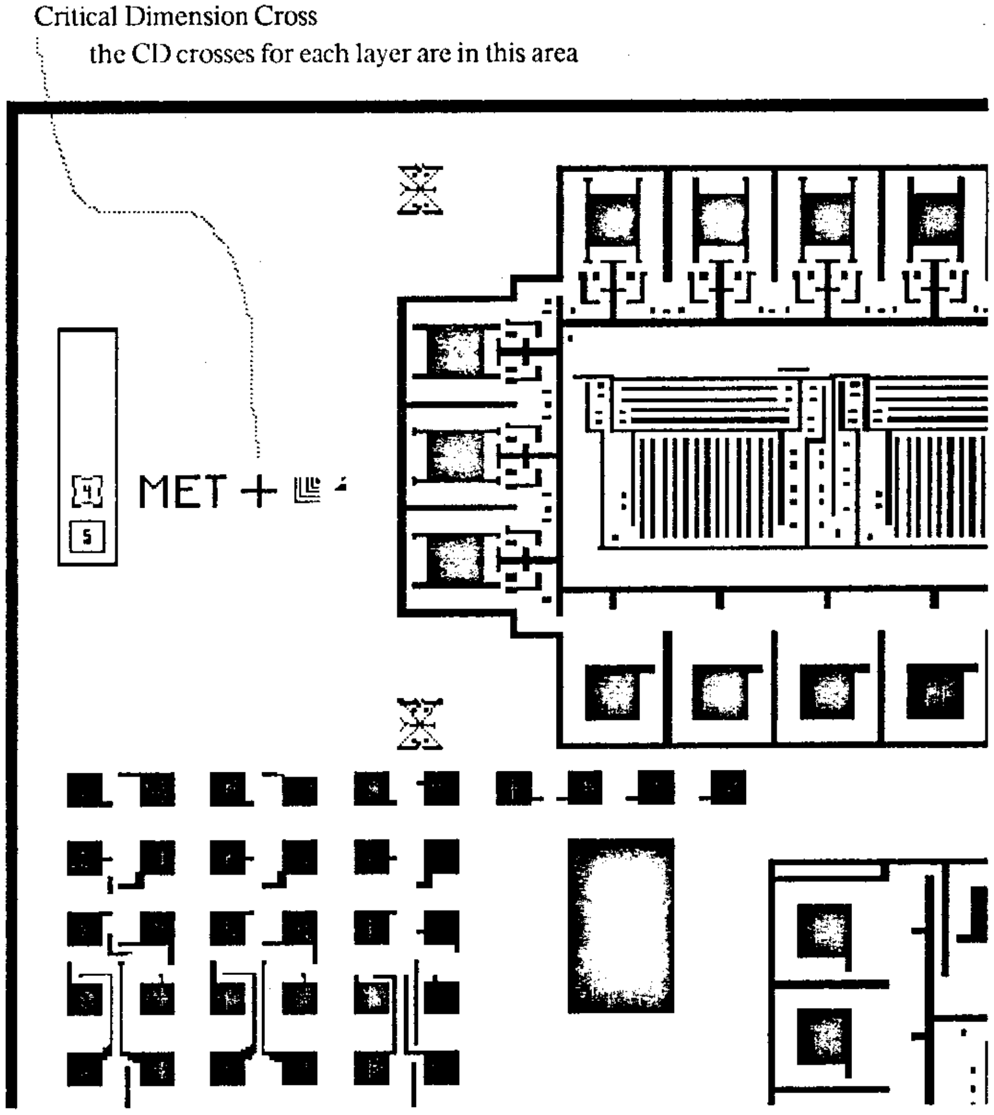


Figure A.2 Close-up of the CD on the MET Layer

## Appendix B. Index of Manufacturers

### MASKS

**Micro Mask Inc.**
Contact: Joel Sorem (408) 245-7342
695 Vaqueros Ave.
Sunnyvale, CA 94086

**Microfab Systems Corp.**
Contact: John Traub (415) 965-1750
3960 Fabian Way
Palo Alto, CA 94303

**NBK Corp.**
Contact: Sharad Patel (408) 988-2600
3010 Olcott
Santa Clara, CA 95051

**Transmask Corp.**
Contact: Dave Dove (408) 247-4577
or (714) 540-6080
3952 Campus Drive
Newport Beach, CA 92660

**Ultratech Corp.**
Contact: Dave Lee (408) 247-0451
Photomask Division
1950 Coronado Drive
Santa Clara, CA 95051

## WAFER FAB

**Maruman Integrated Circuits Inc.**
Contact: Bill Robson, Wes Miles (408) 739-0560
1220 Midas Way
Sunnyvale, CA 94086

**Polycore Electronics Inc.**
Contact: S. K. Leong (213) 991-1061
1107 Tourmaline Drive
Newbury Park, CA 91320

**Synertek Inc.**
Contact: Dick Stangl (408) 988-5600
3001 Stender Way
Santa Clara, CA 95051

## EQUIPMENT

**GCA Corp.**
Contact: Al Ferry (programming manager) (617) 272-5600
174 Middlesex Turnpike
Burlington, MA 01803
Contact: Robert Darrow (sales engineer) (408) 732-5330
3570 Ryder Street
Santa Clara, CA 95050

## Appendix C. Mann 3000 Pattern Generator Format

The data format of the Mann 3000 Pattern Generator is a useful output format for maskmaking, since many mask makers have GCA Mann equipment, which can either work directly from this format or convert it appropriately. Electromask machines, which are also commonly used, have a similar but not identical format.

The following description is excerpted from "Type 3000 Pattern Generator Data Formats for Metric Units" from GCA/Burlington Division (there is also a format for English units, which we suggest you avoid). We assume use of 9-track tape, though the Mann 3000 can also accept paper tape or 7-track tape. The encoding is 800 bpi EBCDIC (not ASCII) with CRC and LRC characters, odd lateral parity and even longitudinal parity, and 512-character records (all this is standard data format for IBM-compatible magnetic tape).

### *Syntax*

The true format is whatever the program accepts, which does not conform to a simple syntax, since it is not based on one; however, we have tried to formalize the format description, making safe assumptions where necessary. In the following syntax description, vertical bar | means *or*, curly brackets {} mean repetition zero or more times, nonterminals are written in lower case letters only, and everything else is a terminal.

```
tape        = <BOT> directory <EOF> {file <EOF>}
directory   = {entry} $
entry       = name filenumber numrecords
name        = namepart namepart
namepart    = cccccc | sccccc | sscccc | sssccc | sssscc | ssssc
filenumber  = digit digit digit digit
numrecords  = digit digit digit digit
file        = patternfile | otherfile
otherfile   = any legal EBCDIC characters
patternfile = { {newline} item {newline} } $
item        = message | exposure
message     = " {char} <CR>
exposure    = {parameter} ;
parameter   = X number | Y number | H number | W number | A number
newline     = <CR> | <LF>
number      = digit {digit}
char        = c | . | , | - | s
c           = letter | digit
s           = <SPACE>
letter      = A through Z
digit       = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

*Semantics*

The *directory* identifies each *patternfile* with a unique *name* (such as "_CHIP2_MASK3") made of two 6-character *nameparts* (right justified and padded with spaces if shorter). The patternfiles are numbered in order of occurrence on the tape from 0001 (up to 2047), and the *filenumbers* are put in the directory with the names. The number of 512-character records in each patternfile is also in the directory; according to Mann documentation, *numrecords* is not used and need not be correct, except that it must be 4 decimal digits to fit the format. Each directory *entry* is exactly 20 characters, so if you stick to one design per *tape*, the directory will fit in one record; if you need 25 or more entries, fill the last 12 characters of the record with spaces and use another record, even if only for the terminal $ (yes, that really is a dollar sign to terminate the file; an EOF won't work as it did on older Mann machines).

A word of caution is in order; the GCA document does not say that directory entries should be in the same order as the files, and implies that *otherfiles* may be included on the tape but must not be mentioned in the directory. But the GCA document on the 3600 format, which is in many ways similar, says that otherfiles must have a corresponding 20-character directory entry (though the characters have no meaning), and implies that their entries are in order. To be safe, do not put files other than patternfiles on the tape, and put directory entries in order.

Each patternfile is a description of a pattern to be generated (a mask layer, or reticle, for IC applications). After the operator tells the machine which pattern to run, each *item* in the file causes the machine to take an action; *messages* cause typeout on the terminal, and *exposures* cause boxes to be flashed on the plate (glass photographic plate). Messages are not of much use, except to type limericks to the operator (notice that spaces are allowed only in messages, not in exposures or between items). Exposures are the meat of the pattern generator format, and their semantics depend on the coordinate system.

The coordinate system is left-handed, with X increasing to the left and Y increasing upward as you look at the plate. All dimensions are positive, so the origin is the lower right corner. The center coordinate, for any plate size, is half the maximum coordinate. Thus the origin is not generally anywhere within a pattern, though it will be exactly at the corner of a maximum size pattern (10 cm square). See figure C.1 for the relation of the axes, the plate, and a typical exposure.

The units of measure for X and Y, which represent the center of a box, are 1 micron at the reticle (0.1 micron at the chip for 10X reticles); the range is 0 to 100000, which allows features to actually

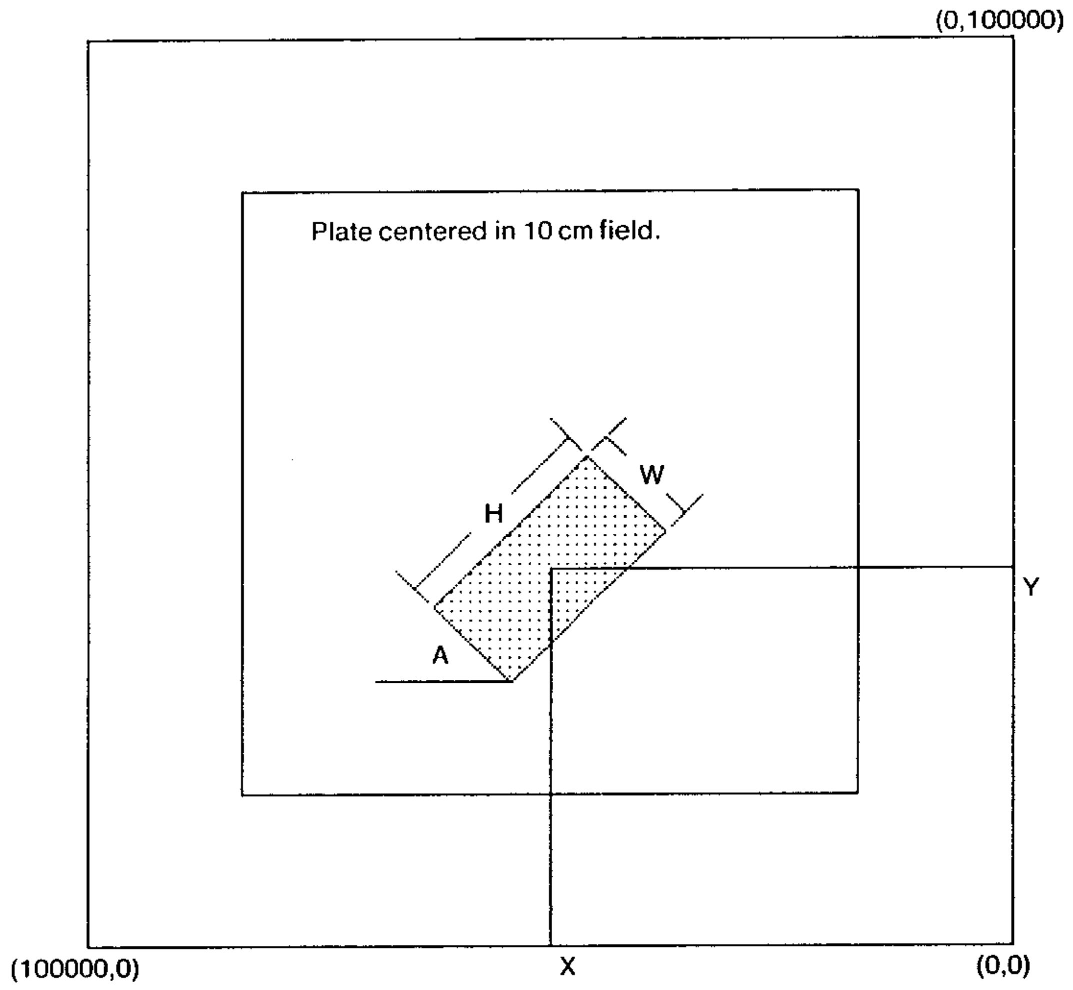


Figure C.1
Mann 3000 Coordinate System.

extend outside the 10 cm field, as long as their centers are within the field. All *numbers* are decimal integers, most significant *digit* first. Leading zeros should be suppressed, since the format actually restricts the number of digits that may be used to six after X or Y, four after H or W, and three after A (the digit-count limit was not indicated in the above syntax, is not a sufficient restriction, and may not even be necessary).

The height H and width W are measured in the same units as X and Y, but the range is 4 microns to 3000 microns. W is the box size along the X axis before rotation, and H is the perpendicular dimension.

The angle of rotation A is in units of 0.1 degree clockwise, in the range of 0 to 899 (89.9 degrees). If CIF is converted using dimensions directly and angles ArcTan(dir.X,dir.Y), which are counterclockwise from the +X axis, the result is the proper pattern, but mirrored. If X coordinates are then negated to fix mirroring, angles should also be negated (and everything converted to proper ranges).

As the syntax shows, each exposure consists of any number of *parameters*, terminated by a semicolon. The machine keeps track of the latest value of each of the five parameters X, Y, H, W, and A, and generates the aperature accordingly (the first exposure must specify all five). Thus, if no parameters are listed between semicolons, the same aperture should be flashed over again; it is not specified that this actually works, and should probably not be tried. If some parameter is listed more than once between semicolons, only the latest value is used (this was useful for correcting mistakes when punching paper tapes).

### *CIF to Mann conversion*

The conversion of CIF files to Mann format should soon be a widely available utility. Hopefully, this format description will help make it happen sooner. A major stumbling block in this effort will be *sorting*, since the pattern generator works very slowly if the exposures are not well sorted relative to the way the machine likes to work. Unfortunately, GCA has not been willing to provide an algorithm for doing a good sort. Due to the way the machine works, it is easy to sort wrong, i.e. much worse than no sort at all.

We do know some basic strategies for sorting, which we summarize here. The X dimension is called the *scan direction*, and the Y dimension is called the *stepover direction*, implying that the machine likes to scan continuously in X, while flashing everything within a narrow band in Y. But changing aperture size or angle is slow, and if a change is not finished by the time the X coordinate is reached an *overrun* occurs. Since the machine is moving in the X direction, overruns cannot be flashed immediately; rather than stop and come back, the machine saves the exposure in a buffer. When the overrun buffer is full, those exposures must then be done (which can be very slow, since they may be anywhere). To accommodate this feature, sort primarily on angle, then on aperture size, so changes occur only rarely (this also reduces the number of parameters in the patternfile). Then within each aperture sort spatially by whatever algorithm you think might work; don't try to outguess the machine.

# Appendix D. ICLIC Manual

*[section contributed by Maureen Stone, Xerox ASD]*

**Acknowledgements**

Special credit goes to Ron Ayres for his help with the design and implementation of the original ICLIC. I would also like to thank the following people for their contributions: Jim Blinn, Carver Mead, Dave Johannsen, and John Gray.

## 1. Introduction

ICLIC was developed at Caltech in the spring of 1978. The initial goal was to provide a simple language that could be used by engineers to do IC design on the DECSystem-20. Previously, all design had been done in PAL [Auto 1966]. The original description [Stone 1978] was directed toward the computer novice. Later in the year, ICLIC was expanded and more emphasis placed on user defined functions and programming.

The system was implemented by Ron Ayres as an extension of the language ICL [Ayres 1978]. Therefore, such features as looping and conditionals have always been available, though not described in the original manual. The syntax and much of the underlying structure, such as the strong and varied data types, very much reflect the structure of ICL.

This section is a rewrite and expansion of the ICLIC users manual done specifically for this document. That is, it is no longer intended to be a description of the Caltech system. Many of the details specific to the Caltech implementation have been omitted. The language of this document has been changed slightly to assume an audience that is already somewhat familiar with programming.

The purpose of this writeup is to provide an example of an IC layout language. Most of the language principles described in section 2.2 of the main document are reflected here.

## 2. Designing Circuits in ICLIC

The design of circuits in ICLIC involves creating symbols (i.e. cells, groups, subpictures) and concatenating them together into larger symbols, up to the chip level. A symbol is described by either a variable or a function call. A number of primitive symbols have been globally defined, which means they can be used by any program running ICLIC (see section 3).

To form a program in ICLIC one defines a number of symbol variables and assigns them values which are strings of function calls and variables. The string, or concatenation operator, is the {} and is described for symbols in section 2.3. Symbols can be concatenated with themselves as well as other variables and function calls to build up more complex symbols. The end result is a symbol which describes the chip.

Variables may be thought of as places to hold values. In ICLIC, each variable name must be given a TYPE that describes what sort of value will be assigned to that name. Declaring and assigning values to variables will be described in sections 2.1-2.3.

A function is a process that is called using the function name and possibly some additional information, called parameters. Section 2.4 describes how to specify functions and their parameters.

### 2.1 Type Declarations

In order to use a variable to hold a value, the place must be reserved with a declaration of the following form:

VAR <varname> = <type>;

    example: VAR I,J=INT; R=REAL; MAINCELL=SYMBOL;

All variables other than the system defined global variables (section 3.4) must be declared in the user's source file before any value is assigned to them. The following TYPEs are defined:

| | |
|---|---|
| INT | integer number. i.e. one with no decimal point. |
| REAL | real (floating point) number. |
| POINT | describes a point on the layout as coordinate#coordinate. Each coordinate is either an absolute value, expressed as a real number, or a relative value, expressed as ". + real" or ". - |

real". The "." signifies the current X or Y coordinate.

example: { 1.5#2.3; 1.5#1.8; 2.0#1.8 }

is equivalent to: { 1.5#2.3; .#.-5; .+.5#. }

The X and Y coordinates of a POINT can be treated separately as REAL numbers by using the notation: POINT.X and POINT.Y.

PATH A list of points such as used to describe the location of the center line of a wire. The following notation is used:

example: { POINT; POINT; ... ; POINT }

SYMBOL The name given to a collection of variables and function calls that describe IC mask elements. SYMBOLs can be concatenated to create larger SYMBOLs by surrounding them with brackets as shown below:

example: {SYMBOL ; SYMBOL ; ... ; SYMBOL}

SYMBOLS are discussed further in section 2.3.

## 2.2 Assignment Statements

A variable can have a value assigned to it using the following form:

<varname> := <value>;

Notice the ":=" for assignment, and the terminating ";". The assignment is not made until the ";" is reached, so nested assignments can be used to build up a the complexity of a symbol. All variable names must be declared by using VAR before the first assignment. Examples of valid assignments to the different type variables follow.

```
I := 1;                "I is an INTeger"
R := 3.146;            "R is a REAL"
ENDPOINT := STARTX+100#55.6;     "ENDPOINT and STARTX are POINTs"
LEFTX := MINPOINT.X;             "LEFTX is REAL, MINPOINT is a POINT"
BUSPATH := { 0#0 ; .+67.3#. ; .#CELLHEIGHT };  "BUSPATH is a PATH"
CELL := { CELL ; ENDCELL ; OUTPAD };          "All variables are SYMBOLs"
```

## 2.3 Symbols

The basic component of an integrated circuit mask is the SYMBOL. Once a variable has been declared to be type SYMBOL, it may be assigned a value one of three ways:

1. Calling a function that returns a SYMBOL as a value. All the system IC design functions described in section 3 fall into this category.

2. Using a variable that already has a symbol value. This includes variables displaced by the system positioning functions.

3. Concatenating together SYMBOLs using the { } notation. This is the way large cells can be built up out of smaller components. The concatenation syntax is:

```
{SYMBOL; SYMBOL; ... ;   SYMBOL}

      example:  CELL:={BUS; HALFCELL; HALFCELL\MIRX};
```

Any number of symbols, separated by semi-colons, and surrounded by "{...}" can be concatenated together into a list of symbols and assigned to a variable of TYPE SYMBOL.

## 2.4 Function Calls

To use a function, you need to know its name and what parameters, if any, are needed. The TYPE of the parameters is important, as well as the TYPE of the value returned by the function. If there is more than one parameter, the order in which they are listed is also important. There are two notations in ICLIC for calling functions which have exactly one or two parameters.

| function name | parameters | notation 1 | notation 2 |
|---|---|---|---|
| F | X | F(X) | X\F |
| F | X,Y | F(X,Y) | X\F Y |

If the function has more than two parameters, notation 1 must be used.

ICLIC system functions all have 2 or fewer parameters, so they can be called using either notation. One convention that is suggested for readability is to call the functions which describe SYMBOLs using notation 1, and the functions which do positioning using notation 2. This convention is followed in the examples in this document.

A function call returns a value. Therefore, function calls can be nested as long as each function returns the correct TYPE. Note that in notation 2 everything before the "\" is the first parameter, so function calls are executed from left to right.

```
example: RB(-1#-1;1#1)\SIZE 5 \ROT 45
```

The example makes a red box which is 10 lambda on a side, rotated 45 degrees counterclockwise around its center point.

## 3. Predefined Functions

This section lists the ICLIC system defined functions useful for the description of IC layouts. They include boxes, contacts, and wires, some predefined variables for design rule distances, and the transformation functions. All these functions return a variable of type SYMBOL. Many names have colors abbreviated in them, using the following conventions:

| | |
|---|---|
| R = red | (for poly) |
| G = green | (for diffusion) |
| B = blue | (for metal) |
| K = black | (for contact cut) |
| Y = yellow | (for implant) |
| W = brown | (for glass) |

The definitions given below will use a "c" to indicate that any of these colors can be used.

### 3.1 Boxes

cB( < low > , < high > )

parameters: low,high = POINT

```
example: TRANS := {RB(-2#-1,2#1); GB(-1#-2,1#2)};
```

The low and high POINTs define the lower left and upper right corners of the rectangle. The box is defined oriented parallel to the axes. That is, X is horizontal, and Y is vertical.

## 3.2 Contacts

Standard contact definitions are built in as symbols. Each symbol includes a box on each layer specified, plus a contact cut (see [Mead 1978] for a complete description of contact cuts).

| | |
|---|---|
| RTOB | red to blue square contact, 4 lambda/side |
| GTOB | green to blue square contact, 4 lambda/side |
| RTOG | red to green butting contact, 6 by 4 lambda oriented horizontally, with red on the left. The 6 by 4 metal layer is included. |

## 3.3 Wires

A wire is described by a PATH, a WIDTH and a LAYER.

| | |
|---|---|
| LAYER | The layer is a description of the layer the wire runs in. |
| WIDTH | The width is width of the wired measured in lambdas. Default wires widths have been defined for each layer. |
| PATH | The PATH describes the location of the center line of the wire. |

### *3.3.1 Single-Layer Wires*

To express a wire on a single layer, the following set of functions have been defined in ICLIC.

cW( < width > , < path > )

parameters: path=PATH, width=REAL (optional)

```
        example:  RW({0#0;  .+1.3#2.7;  .+.7#.});

                  BUS:=BW(6,{-1#-1;  10#10});
```

The path defines the center line of the wire. The actual wire will be expanded a half-width all around the path, even at the endpoints. The global variable for wire width on that layer will be used if the width parameter is omitted. If present, the width parameter applies only for the given wire.

### *3.3.2 Multi-Layer Wires*

In many cases, a wire as an electrical path is not defined on a single layer. Therefore, the following syntax has been defined to describe such wires.

{ LAYER; POINT; POINT; ... ; LAYER; POINT; ... ; POINT }

example: { RED; 0#0; 10#0; BLUE; 12#0; RED; 14#0 }

POINT is either an absolute or relative point, as described in section 2.1.

LAYER is the word RED or GREEN or BLUE, subject to the restrictions presented below.

The overall path in the multi-layer wire is the sequence of points with the layer definitions omitted. The width of each section of wire is the standard width for that layer. Each new layer definition inserts a contact at the last point specified before the layer mnemonic. This point is called the feed-through point. For various reasons, only transitions involving the metal layer have been implemented. An illegal transition generates an error message at execution time.

At each feed through point, one of the system defined square contacts, RTOB or GTOB is generated. The center of the contact is aligned with the feedthrough point.

### *3.3.3 Cables*

To facilitate design using multiple wires following roughly parallel paths, the CABLE function has been defined. The syntax is:

CABLE( < layer > , < #wires > , < next start > , < next end > , < sample path > )

parameters:

layer = LAYER specification, RED or BLUE or GREEN.

#wires = The integer number of wires in the cable

sample path = { STARTP; POINT; ... ; ENDP}. This path describes the center line of one of the end wires. STARTP and ENDP are the starting and ending POINTs for the path.

next start = The starting point for the wire next to the sample path is given as either an absolute or relative POINT.

end start = The ending point for the wire next to the sample path is given as either an absolute or relative POINT.

example: CABLE(RED, 3, .#.-RWSPACE, .#.-10, {0#10; .+5#.; .#0; 10#0})

All wires will be parallel to the sample path at the globally defined minimum spacing except where the start and end points are connected. Default values for the minimum spacing on each layer are given in section 2.4.

No test is made for crossed wires. Whatever wiring pattern is generated by the above algorithm is drawn without comment.

Figure D.1 shows an example of a CABLE.

## 3.4 Global Variables

Global variables are a means to ensure that commonly used distances remain uniform across the layout. Furthermore, global changes can be made by simply reassigning a variable. Therefore, it is strongly recommended that the user use variables for distances wherever possible.

The following system defined global variables have been built in to reflect the design rules described in [Mead 1978].

| | |
|---|---|
| LAMBDA | Definition of the basic measurent unit, in microns. The default value is 3 microns. |
| cWIDTH | width of colored wire |
| cFRMc | design rule distance between colors |
| cWSPACE | minimum spacing between wires (R, G, B only) |
| cWFRMc | minimum spacing between a wire (R, G) and an area that is, the design rule distance plus half the wire width |

These variables have the following defaults:

| | |
|---|---|
| BWIDTH | 3 lambda |
| cWIDTH | 2 lambda (all other colors) |
| RFRMR | 2 lambda |
| RFRMG | 1 lambda |

```
C1:=CABLE(RED,4,.#.-RWSPACE,.#.-RWSPACE,
{0#70;.+18#.;.#.-20;.+11#.;.+10#.-10;.+20#.});
```

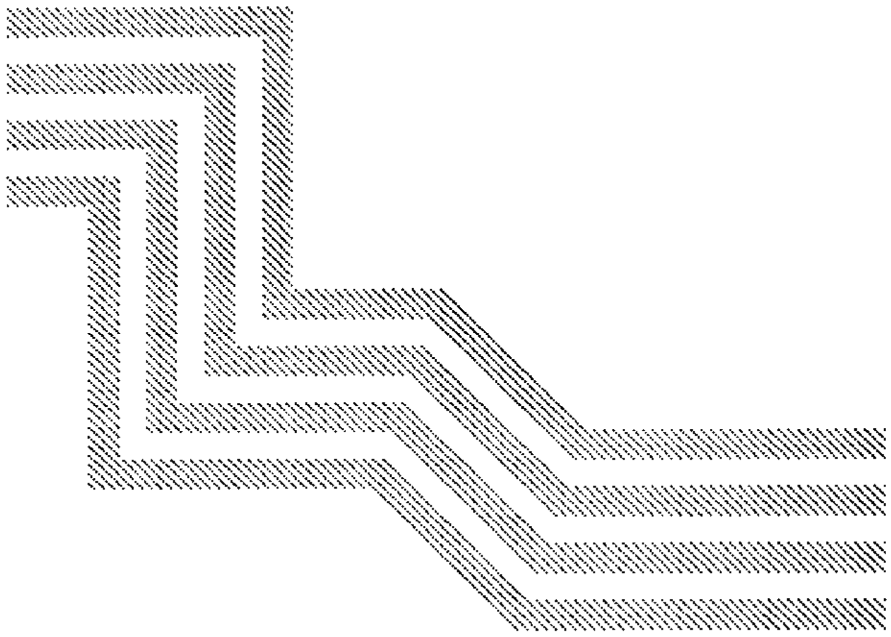


Figure D.1. An Example of a CABLE

| | |
|---|---|
| GFRMG | 3 lambda |
| RWSPACE | 4 lambda |
| GWSPACE | 5 lambda |
| BWSPACE | 6 lambda |
| RWFRMR | 3 lambda |
| RWFRMG | 2 lambda |
| GWFRMR | 2 lambda |
| GWFRMG | 4 lambda |

These variables can be assigned new values by using them on the left hand side of an assignment statement, just like any variable in ICLIC.

## 3.5 Positioning Functions

These functions are used to position symbols, and can be used on any variable of type SYMBOL.

*Displacement*

```
<symbol>\AT  <point>

<symbol>\AT  {<string of points>}

<symbol>\AT  [IX:<real>  IY:<real>  NX:<int>  NY:<int>]

        example:  INVERT\AT 4#4

                  INVERT\AT {0#0 ; .+5#3.7 ; 0#.}

                  CELL\AT [IX: 20.7 IY: 44.7 NX: 5 NY: 3]
```

AT translates the symbol by the amount point.x, point.y, i.e. it adds these values to all X,Y values in the symbol. If a string of points is given, the symbol is duplicated at each of the locations specified. Note that the relative point notation can be used, just as in wire PATHs.

The third form of the AT command specifies a regularly repeated pattern. The square bracket notation defines an array of positions, with increment (IX, IY) and count (NX, NY) parameters in X and Y. The origin of the lower left cell is positioned over the current X,Y. The example describes a 5 by 3 array of cells, spaced 20.7 lambda apart in X and 44.7 lambda apart in Y.

If the parameter is not specified, IX,IY default to 0,0 and NX,NY default to 1. This is useful for making single rows or columns of symbols.

```
        example:  CELL\AT [IX: 20.7  NX: 5]
```

This makes a single row of the previous example.

Note that to position an array at an arbitrary point instead of using the current X,Y it is possible to use nested AT commands.

```
example:  CELL\AT 55.5#25  \AT [IX: 20.7  NX: 5]
```

This example makes the previous example starting at X=55.5 and Y=25 lambda.

*Rotation*

```
<symbol>\ROT  <angle>

        example:  INVERT\ROT  90
```

Rotate the symbol by the value of the angle. The angle is a real number, given in degrees. Positive angles are counterclockwise.

*Scaling*

```
<symbol>\SIZE  <real>

<symbol>\SIZE  <point>

        example:  GTOB\SIZE  2

                  TRANS\SIZE  1#2
```

Scale (multiply all X,Y values) by the parameter value. If the SIZE parameter is a real number, it is used to scale both the X and Y coordinates. If the parameter is a point, the SYMBOL is scaled by the value of the X coordinate in X, and by the value of the Y coordinate in Y.

*Mirroring*

```
<symbol>\MIRX
```

< symbol > \MIRY

< symbol > \MIRXY

These functions mirror (reflect) the symbol around the X axis, Y axis or both axes respectively.

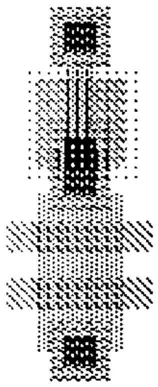
example: LATCH := { HLATCH ; HLATCH\MIRX };

## 3.6 Simple IC Example

The following is a description of a 2 input NAND gate. The pullup length has been made a variable so the pullup ratio can be easily modified.

```
VAR    LEN=REAL;
       LEN:=5.5;                            "LENGTH OF ACTIVE AREA OF PULLUP"
       PULLUP=SYMBOL;                       "VARIABLE LENGTH PULLUP"
PULLUP:=       {
               RTOG \ROT -90;                        "CONTACT TO PULLDOWN"
               GW( { 0#0 ; .#LEN+4 } );              "ACTIVE AREA"
               YB( -4#-1.5, 4#LEN+2.5 );             "DEPL MODE IMPLANT"
               RB( -3#0, 3#LEN+1 );                  "GATE"
               GTOB \AT 0#LEN+4                      "VDD CONTACT"
               };

VAR    NAND2=SYMBOL;                                 "TWO INPUT NAND GATE"
NAND2:=        {
               GTOB;                                 "CONNECT TO GND AT 0#0"
               RW( { 4#4; -4#.} );                   "INPUT 1"
               RW( {-4#8; 4#.} );                    "INPUT 2"
               GB( -3#0, 3#11 );
               PULLUP \AT 0#13
               };
```

The checkplot of this example follows.



## 4. Building Your Own Tools

The functions described in section 3 provide a way of digitizing a layout. A more powerful approach to description is to actually compute the layout. This section describes the tools in ICLIC that are directed toward that goal.

### 4.1 Loops and Conditionals

The form of the conditional statement in ICLIC is:

```
IF <BOOL> THEN <EXPR> ELSE <EXPR> FI
```

where <BOOL> is some boolean expression, and <EXPR> is one or more assignment statements.

```
example: IF N=3 THEN ARRAY:= { ARRAY; INVERTER; CELL};
         ELSE ARRAY:={ ARRAY ; CELL}; FI
```

The example inserts an extra inverter when N equals 3.

The basic form of the loop control in ICLIC is:

```
DO
      <computations>
FOR <loop counter> FROM <start> TO <end> BY <increment>;
```

The loop counter must be a variable of type INTEGER or REAL, and start, end, and increment must reduce to numerical values of the same type. The BY value is optional. The loop counter is tested at the end of the block, so all loops are executed at least once.

```
example: DO
             NEWROW:=ROW(NBITS,I);          "MAKE A NEW ROW, I IS A FLAG"
             Y:=Y+INCY;
             SRA:={SRA: NEWROW \AT X#Y}; "ADD THE NEW ROW TO SRA"
         FOR I FROM 2 TO NBITS;
```

This example is taken from the barrel shifter example in section 4.4 and builds an array of similar rows as dictated by the flag I.

## 4.2 Minimum Bounding Box

A function is provided that returns the minimum bounding box of any SYMBOL. The return in in the form of the minimum and maximum X,Y for the symbol. The easiest way to use the function is to assign the return to two separate POINT variables as shown in the example.

```
MBB( <symbol> )

parameter: SYMBOL
        example: LOWPOINT:=MBB(CELL).LOW;
                 HIPOINT:=MBB(CELL).HIGH;
                 LEFTX:=LOWPOINT.X;        "LOWPOINT is POINT, LEFTX is REAL"
                 TOPY:=HIPOINT.Y;          "HIPOINT is POINT, TOPY is REAL"
                 SIZE:=HIPOINT-LOWPOINT;     "SIZE is a POINT"
                 WIDTH:=SIZE.X+1;            "WIDTH is REAL"
```

## 4.3 User Defined Functions

The most basic type of function takes a parameter list and returns a value. While this value could be of any defined TYPE, to produce SYMBOLs it must be of TYPE SYMBOL. Parameters to functions are passed by value.

```
DEFINE <name>(<parm list>)=<return type>:

<body>

ENDDEFN
```

Note the colon after the return type, and the lack of semi-colon after the ENDDEFN. The <name > is the name used in the function call (section 2.4). The <parm list> is the parameter list definition, and must contain the dummy parameters and their TYPEs in the order they will be called. The <return type> is the TYPE the function returns, usually SYMBOL. Some examples of the first line, or header, of a function definition follow.

```
DEFINE RB(LOW,HIGH:POINT)=SYMBOL:

DEFINE ROT(SYMB:SYMBOL R:REAL)=SYMBOL:
```

It is possible to define more than one function with the same name as long as the parameter lists are different. That is, the parameters must be of different TYPEs, or in a different order. The value of this is that it is then possible to define several functions of the same name that do different things, depending on what TYPE of parameters are passed to them. For example:

```
DEFINE SIZE(SYMB:SYMBOL  R:REAL)=SYMBOL:

DEFINE SIZE(SYMB:SYMBOL  P:POINT)=SYMBOL:
```

The <body> of the function contains ICL statements just as in the main program. In the simplest case, this will be just a symbol definition. Notice that there is no semi-colon after the symbol description.

```
"VARIABLE LENGTH PULLUP"
"LENGTH IS IN LAMBDAS"
DEFINE VPULLUP(LEN:REAL)=SYMBOL:
      {
      RTOG \ROT -90;                    "CONTACT TO PULLDOWN"
      GW({ 0#0 ; .#LEN +4 });           "ACTIVE AREA"
      YB( -4#-1.5, 4#LEN+2.5 );         "DEPL MODE IMPLANT"
      RB( -3# 0  , 3#LEN+1  );          "GATE"
      GTOB \AT  0#LEN+4                 "VDD CONTACT"
      }
ENDDEFN
```

If local variables are required, the following form is used.

```
DEFINE <name>(<parm list>) = <return type>:
  BEGIN
  VAR   <variable declarations>;
     DO
           <computations>
     GIVE  <return value>
  END
ENDDEFN
```

Notice the lack of semi-colons after the DO, BEGIN, GIVE, and END statements. Following is an example of this form of function definition.

```
"GIVEN THE WIDTH OF THE PULLDOWN, MAKE A 2 INPUT NAND"
"USE MINIMUM WIDTH RED WIRES FOR INPUT GATES"
"CALLS VPULLUP, WHICH IS DEFINED ABOVE"
"USES SYSTEM DEFINED GLOBAL VARIABLES RWIDTH AND GWIDTH"
```

```
DEFINE  NAND2(  PDWIDTH:REAL  )=SYMBOL:
   BEGIN
   VAR NAND=SYMBOL:     LEN,W1=REAL;
        DO
        LEN:=(8*RWIDTH*GWIDTH)/PDWIDTH:    "COMPUTE 8:1 RATIO"
        W1:=1+PDWIDTH/2:                   "HALF WIDTH OF INPUT GATE"
        NAND:=    {
                  GTOB:                    "CONNECT TO GND AT 0#0"
                  RW({  W1#4:  -W1#. }):   "INPUT 1"
                  RW({ -W1#8:   W1#. }):   "INPUT 2"
                  GB( -PDWIDTH/2#0, PDWIDTH/2#11  ):
                  VPULLUP(LEN) \AT  0#13
                  }:
       GIVE  NAND
   END
ENDDEFN
```

(This NAND gate is the same as the one in section 3.6, for PDWIDTH=6.)

## 4.4 More Circuit Examples

The code for two larger circuits is shown here. The first circuit is the 4 bit inverter shift register shown in figure D.2. The inverter shift register is a single function, SR(NBITS), that takes as its parameters the number of bits desired. The array form of the AT function (see section 3.4) is used to duplicate the elements. Placement of wires is achieved by using pre-defined size constants such as cell width.

```
"INVERTER SHIFT REGISTER. PARAMETER IS NUMBER OF BITS"
"NUMBER OF CELLS = 2*NUMBER OF BITS"

DEFINE SR(NBITS:INT)=SYMBOL:
BEGIN
 VAR  PULLUP2, CELL, ENDCELL, CELL1=SYMBOL:
      INX, CINY, COUTX, COUTY =REAL:
      CELL, SRA=SYMBOL:
      1, D2, D3, SRWIDE, CWIDE, CHIGH, DFRMBT, DFRMPAD=REAL:

 DO
      PULLUP2:=       {
                      RTOG\AT 0#0\ROT -90:
                      RB(-10#0, 4#7):
                      GTOB\AT -7#10:
                      YW(5, {1#0: .#3: -7#.: .#8}):
                      GW({1#0: .#3: -7#.: .#8})
                      }:
      CINX:=-10:   CINY:=9:
      COUTX:=-CINX:   COUTY:=CINY:
      CELL1:=         {
                      PULLUP2\AT 3#11:
                      GB(-1#0, 5#10):
                      GW({5#9: COUTX#COUTY}):
                      RW({-2#6: 6#6})
                      }:
      CELL:=          {
                      RTOG\ROT 180\AT -5#7:
```

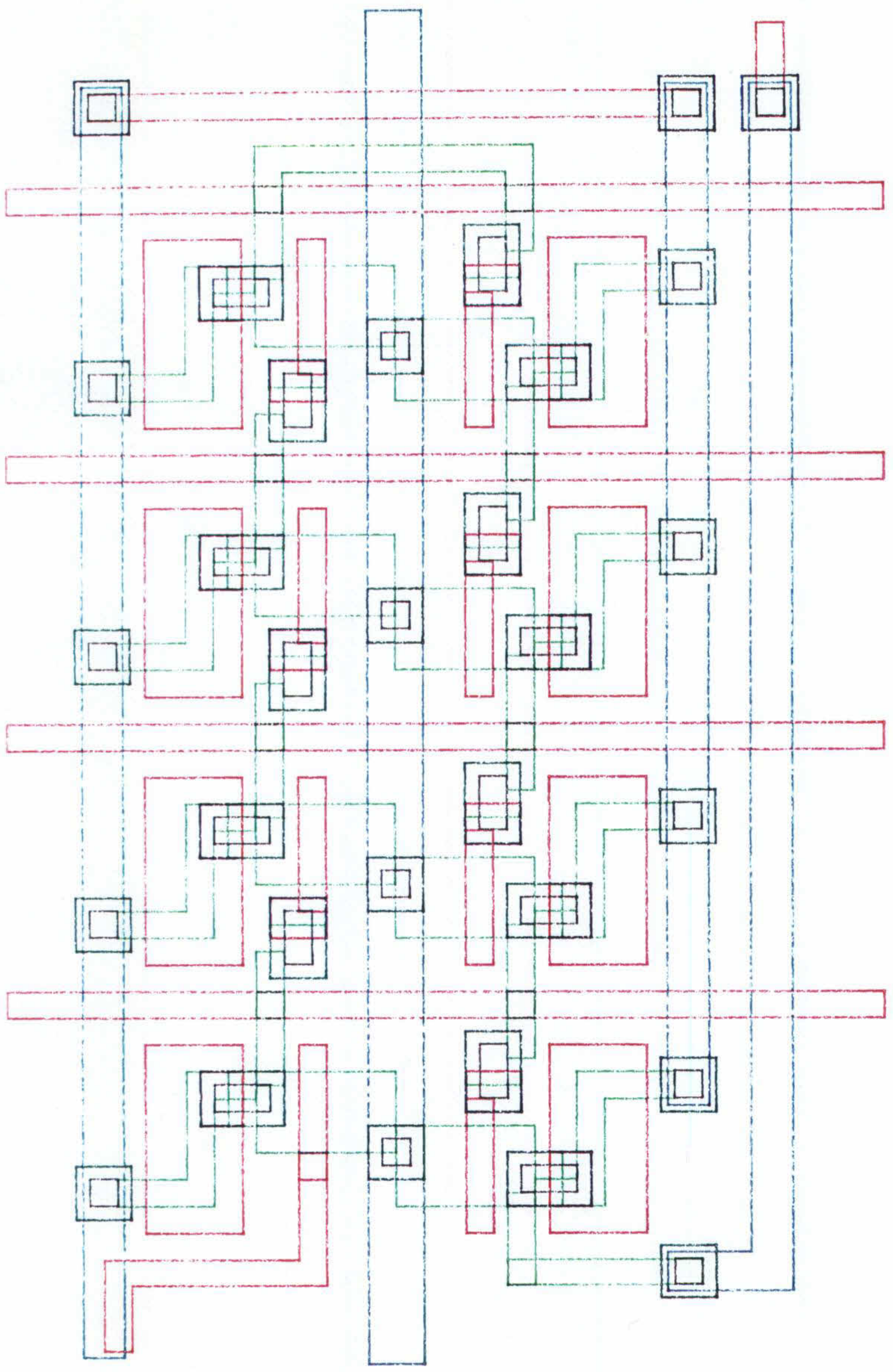


Figure D.2 Four Bit Inverter Shift Register

```
                      GW({CINX#CINY: .+3#.});
                      CELL1
                      };
      DCELL:=         {
                      CELL\ROT 180;
                      CELL;
                      GTOB\AT -1#0
                      };

      ENDCELL:=       {
                      CELL\ROT 180;
                      CELL1;
                      GTOB\AT -1#0
                      };
      CWIDE:=20;   CHIGH:=21;

      "MAKE THE ARRAY PART"
      SRA:=           {
                      ENDCELL\AT CWIDE/2#0;
                      DCELL\AT [NX:NBITS-1 IX:CWIDE NY:1 IY:0]
                      \AT 1.5*CWIDE#0
                      };

      "PUT IN THE WIRES"
      DFRMBT:=5;   "DISTANCE FROM BUS TERMINATOR"
      DFRMPAD:=12;      "DISTANCE FROM PAD"
      D1:=7;  "FROMLAST CELL TO VDD CONN"
      D2:=3;  "FRM LAST CELL TO FEED AROUND WIRE"
      SRWIDE:=CWIDE*NBITS;  "DISTANCE TO END OF SR ARRAY"

      SRA:=           {
                      SRA;
                      BW(4, {-DFRMBT#0;  (SRWIDE+DFRMPAD)#0});   "GND"

      " VDD...2 BLUE WIRES CONNECTED BY A RED ONE"
                      {BLUE;  14#-CHIGH; SRWIDE+D1#.; RED: .#CHIGH; BLUE:
                      - DFRMBT#.};

       "CONNECT UPPER TO LOWER AND SHORT ONE OUT"
                      GW({(SRWIDE)#COUTY; .+D2#.; .#-COUTY; .-D2#.});
                      YB(SRWIDE+D2-5#COUTY-1.5, SRWIDE+D2-.5#COUTY+1.5);

      "OUTPUT WIRE"
                      {GREEN:  0#-COUTY;  .#-CHIGH;  BLUE;  .#.-BWSPACE;
                      SRWIDE+D1#.;  RED;  .+(DFRMPAD-D1)#.};

      "INPUT WIRE"
                      RW({-DFRMBT#CHIGH-1; 0#.; .#6; 8#.});

      "NOW PUT IN CLOCK LINES.  NEED NBITS OF THEM"
                      RW({0#-CHIGH-13; 0#CHIGH+6})\AT CWIDE#0
                      \AT [IX:CWIDE IY:0 NX:BITS NY:1]
                      };
      GIVE SRA
 END
ENDDEFN
```

The second circuit is a barrel shifter. The layout is shown in figure D.3. The barrel shifter is structured in similar, but not identical rows. The rows differ in the placement of one special cell. The barrel shifter function, SHIFTR(NBITS), takes the number of bits as its parameter, and calls a subfunction once for each row. The subfunction, ROW(NBITS,NPOS), takes the number of bits and an integer indicating which position the special cell sits in the row and generates one row of

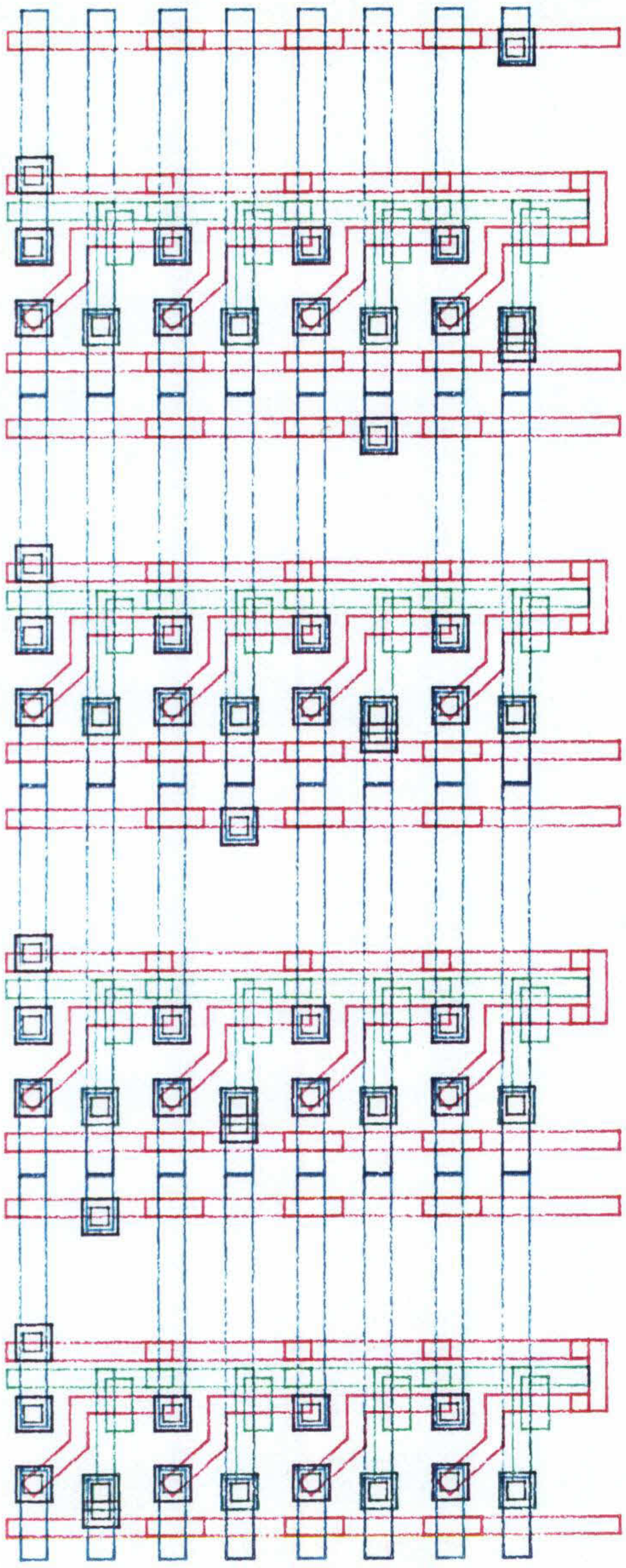


Figure D.3 Four Bit Barrel Shifter

the layout. The outer function computes the minimum bounding box for the row to determine the displacement in y.

```
"THIS IS THE CODE FOR THE BARREL SHIFTER"
"UPPER LEVEL FUNCTION IS SHIFTR(NBITS)"
DEFINE SHIFTR(NBITS:INT) = SYMBOL:
BEGIN
 VAR   NEWROW, SRA=SYMBOL;
       X, Y, INCY=REAL;
       P1, P2=POINT;
       I=INT;

 DO
       X:=0; Y:=0;
       SRA:=ROW(NBITS, 1);
 "COMPUTE THE Y INCREMENT FROM THE MINIMUM"
 "BOUNDING BOX OF ROW"
       P1:=MBB(SRA).LOW;    P2:=MBB(SRA).HIGH;
       INCY:=(P2-P1).Y-1;
       DO
         NEWROW:=ROW(NBITS, I);
         Y:=Y+INCY;
         SRA:=        {
                      SRA;
                      NEWROW\AT X#Y
                      };
         FOR I FROM 2 TO NBITS;
         GIVE SRA
 END
ENDDEFN

"DEFINES ONE ROW OF THE SHIFT REGISTER"
DEFINE ROW(NBITS:INT POS:INT)=SYMBOL:
BEGIN
 VAR   SUBCELL, CELL, CELL1, CELLN, ONEROW=SYMBOL;
       BUSA, BUSB, OUT, SHIFT, RCON1, RCON2, GATES,
       GCON, X1, X2, X3, CWIDTH, CTOP=REAL;
       IX=REAL;

 DO
       "SET UP THE DEFINED COORDINATES
        IF WE DO THIS RIGHT, THE REST IS EASY"
       BUSB:=2;    SHIFT:=18;
       OUT:=SHIFT+RFRMG+RWIDTH/2+GWIDTH/2;
       GATES:=OUT+RFRMG+RWIDTH/2+GWIDTH/2;
       RCON1:=GATES+1; RCON2:=RCON1+8;
       GCON:=RCON2+1;
       BUSA:=GCON+4;
       CTOP:=BUSA+2;
       X1:=3.5;     X2:=X1+7;     X3:=X2+7.5;
       CWIDTH:=X3+2;

       "BASIC  CELL"
       SUBCELL:=     {
                     RW({0#BUSB; CWIDTH#BUSB});
                     RW({X1#SHIFT; CWIDTH#SHIFT});
                     GW({X1#OUT; CWIDTH#OUT});
                     RW({0#BUSA; CWIDTH#BUSA});
                     {RED;  X1#GATES;  X2+2.5#.;  .#.+4.5;
                       X3#RCON2; BLUE; .#CTOP}
                     {GREEN; X2-.5#OUT; .#GCON; .+.5#.; BLUE; .#CTOP};
                     {BLUE; X3#0; .#RCON1; RED};
                     GB(X1+3.5#OUT, X2-.5#RCON1+2)
                     };
        "SPECIAL CELL"
        CELL1:=        {
                       SUBCELL;
                       RTOG\ROT-90\AT X2#GCON+1;
                       {BLUE;  X2#0;  .#BUSB+1;  RED}
                       };

        CELL:=         {
                       SUBCELL;
                       BW({X2#0;  X2#CTOP})
                       };

        "COMPUTE THE SPACING IN X"
        IX:=X3-X1+1;  "X3 ALIGNS WITH X1"

        "POS IS THE POSITION OF CELL1"
        ONEROW:=       {
                       CELL\AT [IX:IX IY:0 NX:POS-1 NY:1];
                       RW({X1#GATES; .-2#.; .#SHIFT; .+2#.});
                       CELL1\AT (POS-1)*IX#0;
                       CELL\AT  POS*IX#0
                         \AT [IX:IX IY:0 NX:NBITS-POS NY:1];
                       RTOB\AT NBITS*IX+2.5#SHIFT-1
                       };

        GIVE ONEROW
  END
ENDDEFN
```

## 5. Operating Procedures

This section should contain information specific to the system and facilities available. Important features are listed under each topic.

### 5.1 Plotting

The following functions are needed:

Mapping from the layer specification to the plotter (colors, stipples, whatever).

Specifying what layers and what circuit elements will be included in the plot.

Specifying circuit size and location relative to the plotting device.

If facilities dictate, how to pre-process plot files off-line from the plotter to optimize plotter usage.

How to abort plots.

### 5.2 Getting In and Getting Out

The areas that need to be discussed are:

File creation and storage.

Operating the compiler.

Errors and their explanation.

Conversion of finished circuit to CIF or other intermediate description form.

# Appendix E. Basic Library of Cells

*[section contributed by Robert Baldwin, MIT, and Richard Lyon, Xerox PARC]*

The purpose of a cell library is to make the design process easier, by saving redundant design and debugging time. For example, it would be a waste of time if each person had to redesign his own pad drivers. Some cells like pullups and butting contacts are easy to design quickly, but they're also easy to enter incorrectly. Forgetting the implant layer or the cover metal is a common mistake. By having a library of cells these problems can be avoided.

A library also encourages people to document and share their designs. A designer should tend to build his project out of existing symbols, and in turn should be encouraged to make his own cells easier to interface to, so other people could use them. A collection of designs is particularly useful to beginners, not only for their direct usage, but also as a set of examples which they could modify for their own purposes. See section 7.3 for an example of how the library can be used.

## *Managing a Library*

In a library of reasonable size, problems of *naming* and *multiple versions* arise. For example, more than one person might contribute a cell called "adder", or a designer might modify one of his cells in such a way that projects which used the old version won't work with the new version. One way to handle these problems is to have each project make a copy of all the library cells that it uses. This raises the problem of *recalls* of library cells that are found to have problems. Whatever method of library access and distribution is used, careful personal attention by a library manager will be necessary to minimize such problems and to make the library a truly useful tool.

The cell library needs good documentation and an effective catalog. The PARC *Icarus* cell library has been created within our shared file system, using one file for the catalog or table of contents, and separate documentation and design files for each cell or family of cells. Contributors to the library are required to follow the standard of completing an on-line documentation form before their cells are added to the catalog by the library manager.

*Specific Documented Cells*

Due to the nature of our design system, none of the documentation specifies where a cell's origin is. In Icarus this is no problem, since cells are easily positioned after being drawn; in a different design system it may be necessary to know where the origin is. For the following examples the origin is usually in the upper left hand corner, but the butting contacts have the origin in the middle of the poly edge.

Most of the cells in the table of contents are documented in the pages that follow. Some PARC-specific information has been deleted for this version. For each file, CIF (version 2.0) code and Icarus checkplots are attached. The checkplots are similar to the ones shown in chapter 4 of [Mead 1978], but the stipple patterns which are used to represent the layers are different. The patterns used in the documentation are as shown here:

| | | | | | |
|---|---|---|---|---|---|
| ░ | Metal | ▒ | Diffusion | ⁘ | Implant |
| ▓ | Contact Cut | ▧ | Polysilicon | ⁙ | Overglass Cut |

**Cell Library**
**Table of Contents**

This file (CellLib.TOC) is the table of contents for the cells in the library, with entries organized by function.

The template form.LibDoc is the standard for new symbol documentation, and must be completed before entries are added to this table of contents.

| SymbolName | Description | Document File |
|---|---|---|
| **Pads** | | |
| PadIn | Input pad with Lightning arrestor. | Pads.LibDoc |
| PadOut | Output pad with driver. | Pads.LibDoc |
| PadVdd | Input pad for Vdd. | Pads.LibDoc |
| PadGround | Input pad for Ground. | Pads.LibDoc |
| PadBlank | Blank pad used for spacing. | Pads.LibDoc |
| PadSample | Shows how to use these pads. | Pads.LibDoc |
| **Inverters** | | |
| InverterPair | Two cascaded inverters | Inverters.LibDoc |
| InveterQuad | Two of the above back to back. This was designed for use with a two input general function block. | Inverters.LibDoc |
| BackwardInverterPair | Like InverterPair only data flows from right to left. | Inverters.LibDoc |
| BackwardInverterQuad | Like InverterQuad only data flows from right to left. | Inverters.LibDoc |
| **Primitive Cells** | | |
| RtoG | Horizontal contact with red on left. | ButtingContacts.LibDoc |
| RdownG | Vertical contact with red on top. | ButtingContacts.LibDoc |
| GtoR | Same as RtoG, just rotated 180. | ButtingContacts.LibDoc |
| GdownR | Same as RdownG, just rotated 180. | ButtingContacts.LibDoc |
| VPullup4:1 | Vertical pullup with L/W = 4. | Pullups.LibDoc |
| HPullup4:1 | Horizontal pullup with L/W = 4. This is a mirrored and rotated version of VPullup4:1. | Pullups.LibDoc |
| VPullup2:1 | Vertical pullup with L/W = 2. | Pullups.LibDoc |
| HPullup2:1 | Horizontal pullup with L/W = 2. This is a mirrored and rotated version of VPullup2:1. | Pullups.LibDoc |

## PLA's

| | | |
|---|---|---|
| PlaCellPair | Two poly and two metal lines, with four diffusion contacts and a ground line. | PLA.LibDoc |
| PlaGround | For around the edges of PLA's, and in the middle of big ones. | PLA.LibDoc |
| PlaConnect | Connects AND-plane to OR-plane. | PLA.LibDoc |
| PlaIn | Clocked inverting and noninverting drivers. | PLA.LibDoc |
| PlaInPair | Two PlaIn's sharing a ground line. | PLA.LibDoc |
| PlaOut | Pair of clocked inverters for PLA output. | PLA.LibDoc |
| PullupPair | Pair of pullups for PLA diffusion lines. | PLA.LibDoc |
| PlaProgFlash | Connection for programming the PLA. | PLA.LibDoc |
| PLA-2-4-4 | Shows how to put together a PLA. | PLA.LibDoc |

## Logic and Arithmetic

| | | |
|---|---|---|
| FuncBlock | This is the general Function Block which is described in Mead&Conway. It can be used to generate any function of two inputs. | FuncBlock.LibDoc |
| SerialAdder | Cell that adds or subtracts two serial numbers given LSB first. | SerialAdder.LibDoc |
| ExtAdder | Like SerialAdder except it can output a sign extension, and it does not produce a new LSBtime signal. | ExtAdder.LibDoc |

## Miscellaneous

| | | |
|---|---|---|
| SRCELL | The shift register cell from chapter 4 of Mead&Conway. | SRCELL.LibDoc |

## Starting Frame

| | | |
|---|---|---|
| Align | Six alignment marks, for seven total layers (includes BUR and OVG). | AlignMarks6.LibDoc |
| LayerNames | Layer names and critical dimension crosses. | AlignMarks6.LibDoc |
| EtchTest | Etch test patterns. | AlignMarks6.LibDoc |
| LayerAlign | Align, LayerNames, and EtchTest. | AlignMarks6.LibDoc |
| vertscribe, etc. | Various pieces of interior and exterior scribe lines. | ScribeLines.LibDoc |

## Documentation for Pads: PadIn -- PadOut -- PadBlank PadGround -- PadVdd -- PadSample

| | | | |
|---|---|---|---|
| Date | August 7, 1978 | Status | used in summer 1978 MPC |
| Designer | Dick Lyon | Address/Phone | PARC SSL x4325 |
| Design Rules | Mead/Conway | Scale | λ = 3 μm |
| Info File | Pads.LibDoc | CIF File | Pads.cif |
| Dimensions | X: 86λ Y: 124λ | Replication | DX: 80λ DY: No |

Further Info

See attached figures.

Function/Use

This is a family of general-purpose input and output pads.
PadIn: Input pad with lightning arrestor.
PadOut: Output pad with three-stage fast driver.
PadBlank: Plain metal pad for whatever.
PadGround: Pad connected to ground.
PadVdd: Pad connected to Vdd.
PadSample: Shows how they fit together.

Connections

Vdd runs along the top and sides. Ground runs along the bottom. Both are in 6λ metal. Pads should be overlapped by 6λ, so they share the Vdd strip along the sides of the pads.
PadIn: Connect to the diffusion at the bottom of the cell.
PadOut: The input is near the bottom right hand corner in poly, and a butting contact is provided so that a diffusion line can also be connected. The cell can be mirrored about Y to get inputs on the bottom left.
PadBlank: Connect directly to the pad in metal.
PadGround: The pad is connected to the ground stripe.
PadVdd: The pad is connected to the Vdd stripe.

Included Cells

None

Performance

PadOut: See section 2.4 of *A Guide to LSI Implementation.*

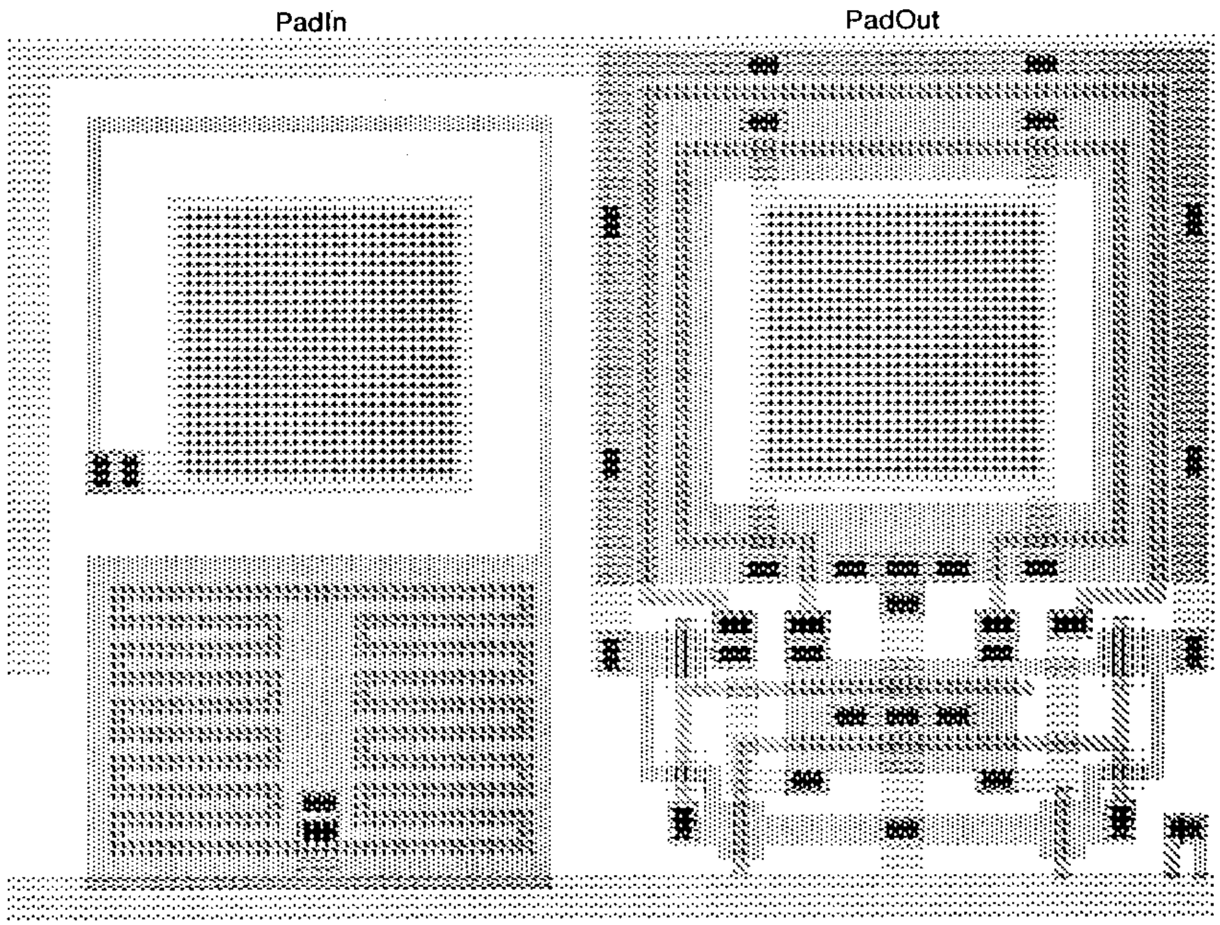
How it works

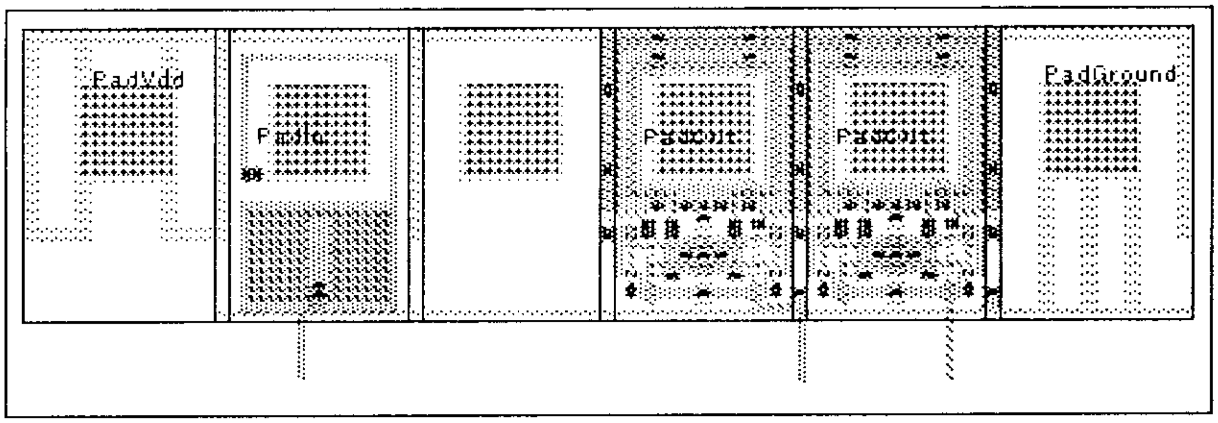
PadIn: The circuit consists of three parts: a metal bonding pad, which is connected to a diffusion resistor, which is connected to a transistor, which is connected to ground. The output of the circuit is between the resistor and the transistor. Normally the transistor is off, since its gate is connected to ground, so the circuit behaves like a resistor connected to a bonding pad. But, if the voltage across the transistor becomes large (e.g., the input pad is struck by lightning) current will flow due to punch-through, dropping the overvoltage across the resistor. Punch-through occurs at a lower voltage than gate oxide breakdown, so gates are protected.

PadOut: The general strategy is to have a chain of inverters, each with a wider gate than the preceding one. Chapter 1 of [Mead 1978] describes this in more detail. This pad driver has three stages. The first stage contains two inverters in series which are used to drive the next stage, a pair of superbuffers. These, in turn, drive the last stage, a pair of wide enhancement mode transistors. The input gate is four times larger than the minimum gate area, so the pad driver looks like four minimum loads. The gates in the second stage are four times larger than in the first stage, so the inverters in the first stage have a fanout of five (remember that each inverter drives both superbuffers). The last stage has gates eight time larger than the previous gates, but since superbuffers are used, this is like a regular fanout of four. Thus by the third stage L/W = 128, which is large enough to drive a several TTL loads at 10 MHz or faster.

## Design Decisions

The first set of decisions has to do with size. The pads have to be big enough and far enough apart that they are easy to bond to, but not so big as to take up all the space. As they are now, the minimum size of a project with sixteen pads is 1500 microns on a side (approximately 0.6 nanoacres). Another consideration is the spacing between the pad and the rest of the circuit. It has to be far enough away that the bonding wire does not short out or destroy any part of the circuit. Of course, if overglassing is used the pad can be placed closer, but overglassing is often omitted from prototype chips. It was decided that a pad size of 126 microns (114 microns with overglassing), and a pad spacing of 240 microns would be a reasonable compromise.


PadIn
PadOut



Gross View of PadSample,

Showing the general form of PadVdd, PadGround, and PadBlank file: Pads.cif

```
( Created by Sif from Pads.ic );

DS 1;    ( Name: PadGround );
( 10 Items. );
Layer NMet:  Box  Len 25800  Wid 1800  Center 12900,-36300 ;
Layer NMet:  Box  Len 1800  Wid 25500  Center 900,-14250 ;
Layer NMet:  Box  Len 25800  Wid 1800  Center 12900,-900 ;
Layer NMet:  Box  Len 12600  Wid 12600  Center 12900,-12900 ;
Layer NMet:  Box  Len 1800  Wid 17700  Center 7500,-27450 ;
Layer NGls:  Box  Len 11400  Wid 11400  Center 12900,-12900 ;
( PadGround );
Layer NMet:  Box  Len 1800  Wid 17700  Center 12900,-27150 ;
Layer NMet:  Box  Len 1800  Wid 17700  Center 18300,-27450 ;
Layer NMet:  Box  Len 1800  Wid 25500  Center 24900,-14250 ;
DF;


DS 2;    ( Name: PadBlank );
( 6 Items. );
Layer NMet:  Box  Len 25800  Wid 1800  Center 12900,-36300 ;
Layer NMet:  Box  Len 1800  Wid 25500  Center 900,-14250 ;
Layer NMet:  Box  Len 25800  Wid 1800  Center 12900,-900 ;
Layer NMet:  Box  Len 12600  Wid 12600  Center 12900,-12900 ;
Layer NGls:  Box  Len 11400  Wid 11400  Center 12900,-12900 ;
Layer NMet:  Box  Len 1800  Wid 25500  Center 24900,-14250 ;
DF;


DS 3;    ( Name: PadOut );
( 121 Items. );
Layer NMet:  Box  Len 25800  Wid 1800  Center 12900,-36300 ;
Layer NMet:  Box  Len 1800  Wid 25500  Center 900,-14250 ;
Layer NMet:  Box  Len 25800  Wid 1800  Center 12900,-900 ;
Layer NDif:  Box  Len 4800  Wid 22500  Center 2700,-11850 ;
Layer NDif:  Box  Len 4500  Wid 1800  Center 2550,-26100 ;
Layer NCut:  Box  Len 600  Wid 1200  Center 900,-26100 ;
Layer NCut:  Box  Len 600  Wid 1200  Center 900,-18000 ;
Layer NCut:  Box  Len 600  Wid 1200  Center 900,-7800 ;
Layer NDif:  Box  Len 3300  Wid 600  Center 3750,-31200 ;
Layer NDif:  Box  Len 600  Wid 4500  Center 2400,-28950 ;
Layer NPol:  Box  Len 600  Wid 21600  Center 2400,-13200 ;
Layer NPol:  Box  Len 21600  Wid 600  Center 12900,-2400 ;
Layer NPol:  Box  Len 3000  Wid 600  Center 3900,-23700 ;
Layer NImp:  Box  Len 1800  Wid 3000  Center 3900,-26100 ;
Layer NImp:  Box  Len 1800  Wid 1800  Center 3900,-31200 ;
Layer NDif:  Box  Len 2100  Wid 1200  Center 4350,-33600 ;
Layer NMet:  Box  Len 1200  Wid 1800  Center 3900,-33300 ;
Layer NPol:  Box  Len 1200  Wid 900  Center 3900,-32850 ;
Layer NCut:  Box  Len 600  Wid 1200  Center 3900,-33300 ;
Layer NPol:  Box  Len 600  Wid 8100  Center 3900,-28650 ;
Layer NPol:  Box  Len 600  Wid 17100  Center 3900,-13050 ;
Layer NPol:  Box  Len 18600  Wid 600  Center 12900,-4800 ;
Layer NPol:  Box  Len 14400  Wid 600  Center 11100,-27600 ;
Layer NPol:  Box  Len 5400  Wid 600  Center 6600,-21300 ;
Layer NDif:  Box  Len 900  Wid 600  Center 4950,-26100 ;
Layer NDif:  Box  Len 600  Wid 2100  Center 5100,-32250 ;
Layer NDif:  Box  Len 16200  Wid 3300  Center 12900,-21450 ;
Layer NDif:  Box  Len 16200  Wid 5400  Center 12900,-3300 ;
Layer NMet:  Box  Len 1800  Wid 2400  Center 6000,-25500 ;
Layer NPol:  Box  Len 1800  Wid 1200  Center 6000,-24900 ;
Layer NPol:  Box  Len 600  Wid 1200  Center 5400,-24000 ;
Layer NDif:  Box  Len 1800  Wid 1200  Center 6000,-26100 ;
Layer NDif:  Box  Len 1800  Wid 2400  Center 6300,-33600 ;
( PadOut );
Layer NCut:  Box  Len 1200  Wid 600  Center 6000,-24900 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 6000,-26100 ;
Layer NMet:  Box  Len 1200  Wid 7200  Center 6300,-28500 ;
Layer NPol:  Box  Len 600  Wid 5700  Center 6300,-32550 ;
Layer NPol:  Box  Len 15600  Wid 600  Center 14100,-30000 ;
Layer NMet:  Box  Len 1800  Wid 1200  Center 7200,-22500 ;
Layer NMet:  Box  Len 1800  Wid 1200  Center 7200,-3600 ;
Layer NMet:  Box  Len 3300  Wid 1200  Center 8250,-31500 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 7200,-22500 ;
Layer NMet:  Box  Len 1200  Wid 3600  Center 7200,-20700 ;
Layer NMet:  Box  Len 12600  Wid 12600  Center 12900,-12900 ;
Layer NMet:  Box  Len 1200  Wid 3300  Center 7200,-5250 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 7200,-3600 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 7200,-1200 ;
Layer NDif:  Box  Len 12000  Wid 1200  Center 12900,-33600 ;
Layer NGls:  Box  Len 11400  Wid 11400  Center 12900,-12900 ;
Layer NDif:  Box  Len 1800  Wid 1200  Center 9000,-31500 ;
Layer NDif:  Box  Len 9600  Wid 4800  Center 12900,-28800 ;
Layer NMet:  Box  Len 1800  Wid 2400  Center 9000,-25500 ;
Layer NPol:  Box  Len 1800  Wid 1200  Center 9000,-24900 ;
Layer NDif:  Box  Len 1800  Wid 1200  Center 9000,-26100 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 9000,-31500 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 9000,-26100 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 9000,-24900 ;
Layer NPol:  Box  Len 600  Wid 4200  Center 9000,-23100 ;
Layer NMet:  Box  Len 6000  Wid 1200  Center 12900,-28800 ;
Layer NMet:  Box  Len 6000  Wid 1200  Center 12900,-22500 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 10800,-28800 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 10800,-22500 ;
Layer NMet:  Box  Len 1800  Wid 12900  Center 12900,-29250 ;
Layer NDif:  Box  Len 1800  Wid 1500  Center 12900,-23850 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 12900,-33600 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 12900,-28800 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 12900,-22500 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 12900,-24000 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 15000,-28800 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 15000,-22500 ;
Layer NDif:  Box  Len 1800  Wid 1200  Center 16800,-31500 ;
Layer NMet:  Box  Len 3300  Wid 1200  Center 17550,-31500 ;
Layer NMet:  Box  Len 1800  Wid 2400  Center 16800,-25500 ;
Layer NPol:  Box  Len 1800  Wid 1200  Center 16800,-24900 ;
Layer NDif:  Box  Len 1800  Wid 1200  Center 16800,-26100 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 16800,-31500 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 16800,-26100 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 16800,-24900 ;
Layer NPol:  Box  Len 600  Wid 4200  Center 16800,-23100 ;
Layer NPol:  Box  Len 5400  Wid 600  Center 19200,-21300 ;
Layer NDif:  Box  Len 3900  Wid 600  Center 19350,-26700 ;
Layer NMet:  Box  Len 1800  Wid 1200  Center 18600,-22500 ;
Layer NMet:  Box  Len 1800  Wid 1200  Center 18600,-3600 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 18600,-22500 ;
Layer NMet:  Box  Len 1200  Wid 3600  Center 18600,-20700 ;
Layer NMet:  Box  Len 1200  Wid 3300  Center 18600,-5250 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 18600,-3600 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 18600,-1200 ;
Layer NDif:  Box  Len 1800  Wid 2400  Center 19500,-33600 ;
Layer NMet:  Box  Len 1200  Wid 7800  Center 19500,-28200 ;
Layer NPol:  Box  Len 1800  Wid 1200  Center 19800,-24900 ;
Layer NMet:  Box  Len 1800  Wid 1200  Center 19800,-24900 ;
Layer NPol:  Box  Len 600  Wid 3900  Center 19500,-33750 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 19800,-24900 ;
Layer NDif:  Box  Len 2400  Wid 1200  Center 21300,-33600 ;
Layer NPol:  Box  Len 600  Wid 1200  Center 20400,-24000 ;
Layer NDif:  Box  Len 600  Wid 2100  Center 20700,-32250 ;
Layer NDif:  Box  Len 3300  Wid 600  Center 22050,-31200 ;
Layer NPol:  Box  Len 3000  Wid 600  Center 21900,-23700 ;
Layer NDif:  Box  Len 4800  Wid 22500  Center 23100,-11850 ;
Layer NDif:  Box  Len 4500  Wid 1800  Center 23250,-26100 ;
Layer NImp:  Box  Len 1800  Wid 3000  Center 21900,-26100 ;
Layer NImp:  Box  Len 1800  Wid 1800  Center 21900,-31200 ;
Layer NMet:  Box  Len 1200  Wid 1800  Center 21900,-33300 ;
Layer NPol:  Box  Len 1200  Wid 900  Center 21900,-32850 ;
Layer NCut:  Box  Len 600  Wid 1200  Center 21900,-33300 ;
Layer NPol:  Box  Len 600  Wid 8100  Center 21900,-28650 ;
Layer NPol:  Box  Len 600  Wid 17100  Center 21900,-13050 ;
Layer NDif:  Box  Len 600  Wid 4500  Center 23400,-28950 ;
Layer NPol:  Box  Len 600  Wid 21600  Center 23400,-13200 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 24000,-34800 ;
Layer NMet:  Box  Len 1800  Wid 1200  Center 24600,-33600 ;
Layer NPol:  Box  Len 900  Wid 1200  Center 24150,-33600 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 24600,-33600 ;
Layer NMet:  Box  Len 1800  Wid 25500  Center 24900,-14250 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 24900,-33600 ;
Layer NCut:  Box  Len 600  Wid 1200  Center 24900,-26100 ;
Layer NCut:  Box  Len 600  Wid 1200  Center 24900,-18000 ;
Layer NCut:  Box  Len 600  Wid 1200  Center 24900,-7800 ;
Layer NDif:  Box  Len 600  Wid 1800  Center 25200,-34800 ;
DF;
```

```
DS 4:    ( Name: PadIn );
( 59 Items. );
Layer NMet:  Box  Len 25800  Wid 1800  Center 12900,-36300 ;
Layer NMet:  Box  Len 1800  Wid 25500  Center 900,-14250 ;
Layer NMet:  Box  Len 25800  Wid 1800  Center 12900,-900 ;
Layer NDif:  Box  Len 19200  Wid 14100  Center 12900,-28950 ;
Layer NDif:  Box  Len 600  Wid 14100  Center 3600,-10650 ;
Layer NDif:  Box  Len 19200  Wid 600  Center 12900,-3600 ;
Layer NMet:  Box  Len 3600  Wid 1800  Center 5100,-18300 ;
Layer NDif:  Box  Len 2400  Wid 1800  Center 4500,-18300 ;
Layer NCut:  Box  Len 600  Wid 600  Center 3900,-18000 ;
Layer NCut:  Box  Len 600  Wid 600  Center 3900,-18600 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 4500,-33600 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 4500,-31200 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 4500,-28800 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 4500,-26400 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 4500,-24000 ;
Layer NPol:  Box  Len 16800  Wid 600  Center 12900,-34200 ;
Layer NPol:  Box  Len 6600  Wid 600  Center 7800,-33000 ;
Layer NPol:  Box  Len 6600  Wid 600  Center 7800,-31800 ;
Layer NPol:  Box  Len 6600  Wid 600  Center 7800,-30600 ;
Layer NPol:  Box  Len 6600  Wid 600  Center 7800,-29400 ;
Layer NPol:  Box  Len 6600  Wid 600  Center 7800,-28200 ;
Layer NPol:  Box  Len 6600  Wid 600  Center 7800,-27000 ;
Layer NPol:  Box  Len 6600  Wid 600  Center 7800,-25800 ;
Layer NPol:  Box  Len 6600  Wid 600  Center 7800,-24600 ;
Layer NPol:  Box  Len 6600  Wid 600  Center 7800,-23400 ;
Layer NCut:  Box  Len 600  Wid 600  Center 5100,-18000 ;
Layer NCut:  Box  Len 600  Wid 600  Center 5100,-18600 ;
( PadIn );
Layer NMet:  Box  Len 12600  Wid 12600  Center 12900,-12900 ;
Layer NGls:  Box  Len 11400  Wid 11400  Center 12900,-12900 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 11100,-32400 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 11100,-30000 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 11100,-27600 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 11100,-25200 ;
Layer NPol:  Box  Len 4200  Wid 600  Center 12900,-23400 ;
Layer NMet:  Box  Len 1800  Wid 4200  Center 12900,-33900 ;
Layer NPol:  Box  Len 1800  Wid 1200  Center 12900,-33600 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 12900,-33600 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 12900,-32400 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 14700,-32400 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 14700,-30000 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 14700,-27600 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 14700,-25200 ;
Layer NPol:  Box  Len 6600  Wid 600  Center 18000,-33000 ;
Layer NPol:  Box  Len 6600  Wid 600  Center 18000,-31800 ;
Layer NPol:  Box  Len 6600  Wid 600  Center 18000,-30600 ;
Layer NPol:  Box  Len 6600  Wid 600  Center 18000,-29400 ;
Layer NPol:  Box  Len 6600  Wid 600  Center 18000,-28200 ;
Layer NPol:  Box  Len 6600  Wid 600  Center 18000,-27000 ;
Layer NPol:  Box  Len 6600  Wid 600  Center 18000,-25800 ;
Layer NPol:  Box  Len 6600  Wid 600  Center 18000,-24600 ;
Layer NPol:  Box  Len 6600  Wid 600  Center 18000,-23400 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 21300,-33600 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 21300,-31200 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 21300,-28800 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 21300,-26400 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 21300,-24000 ;
Layer NDif:  Box  Len 600  Wid 18900  Center 22200,-13050 ;
Layer NMet:  Box  Len 1800  Wid 25500  Center 24900,-14250 ;
DF;

DS 5:    ( Name: PadVdd );
( 12 Items. );
Layer NMet:  Box  Len 1800  Wid 25500  Center 900,-14250 ;
Layer NMet:  Box  Len 25800  Wid 1800  Center 12900,-900 ;
Layer NMet:  Box  Len 6000  Wid 1800  Center 3900,-26100 ;
Layer NMet:  Box  Len 12600  Wid 12600  Center 12900,-12900 ;
Layer NMet:  Box  Len 1800  Wid 5700  Center 7500,-4050 ;
Layer NMet:  Box  Len 1800  Wid 8400  Center 7500,-22800 ;
Layer NGls:  Box  Len 11400  Wid 11400  Center 12900,-12900 ;
( PadVdd );
Layer NMet:  Box  Len 1800  Wid 5700  Center 18300,-4050 ;
Layer NMet:  Box  Len 1800  Wid 8400  Center 18300,-22800 ;
Layer NMet:  Box  Len 6000  Wid 1800  Center 21600,-26100 ;
Layer NMet:  Box  Len 1800  Wid 25500  Center 24900,-14250 ;
DF;


DS 6;     ( Name: PadSample );
( 15 Items. );
Call 5 Trans 2100,-2400;
( Be sure to connect Vdd );
( and Gnd, if they're not );
( on the same strip );
Call 4 Trans 26100,-2400;
Layer NDif:  Box  Len 600  Wid 9600  Center 33600,-42600 ;
Call 2 Trans 50100,-2400;
Call 3 Trans 74100,-2400;
Layer NPol:  Box  Len 4500  Wid 600  Center 95850,-37800 ;
Call 3 Trans 98100,-2400;
Layer NDif:  Box  Len 600  Wid 9600  Center 99300,-42600 ;
( Notice that this symbol );
( is not symmetric );
Layer NPol:  Box  Len 600  Wid 9900  Center 117600,-42450 ;
Call 1 Trans 122100,-2400;
DF;

End
```

## Documentation for Inverters: InverterPair -- BackwardInverterPair InverterQuad -- BackwardInverterQuad

| | | | |
|---|---|---|---|
| Date | July 25, 1978 | Status | used in summer 1978 MPC |
| Designer | Bob Baldwin | Address/Phone | PARC SSL |
| Info File | Inverters.LibDoc | CIF File | Inverters.cif |
| Design Rules | Mead/Conway | Scale | $\lambda = 3\ \mu m$ |

Dimensions and Replication

| | | | | |
|---|---|---|---|---|
| InverterPair | X: $41\lambda$ | Y: $16\lambda$ | DX: no | DY: no |
| Back...Pair | X: $37\lambda$ | Y: $16\lambda$ | DX: no | DY: no |
| InverterQuad | X: $41\lambda$ | Y: $28\lambda$ | DX: no | DY: $30\lambda$ |
| Back...Quad | X: $37\lambda$ | Y: $28\lambda$ | DX: no | DY: $30\lambda$ |

Further Info — See section 7.3 of *A Guide to LSI Implementation.*

Function/Use — Produces the inverted and non-inverted value of a signal that has been routed though pass transistors (i.e. the first inverter has $Z_{pu}/Z_{pd} = 8$). For InverterPair, the input comes from the left, and both the true and complement output are available in red and green on the right. BackwardInverterPair has the input on the right, and output on the left.

...Quads are pairs of ...Pairs, sharing a ground line between them.

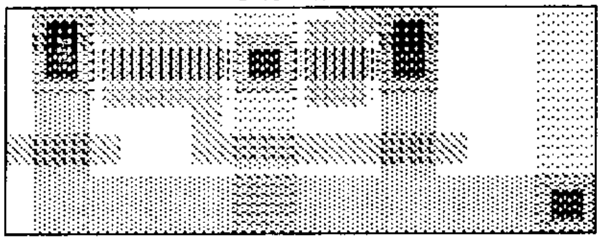
Connections — Vertical Vdd, Ground, and clocks in metal $4\lambda$ wide. Input is on the left (right) side in red $2\lambda$ wide. Both the true and complement output are avialible in red and green on the right (left) side.

Included Cells — None.

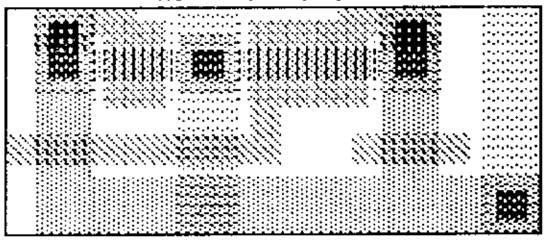
Loadings — Input load: $8\lambda^2$ gate = .03 pf (fanout of 1)
The complement output drives the second inverter, so it already has a load of 1 fanout.

Performance — True output time constant: $\sim(20k\Omega/2)\cdot C_{load}$ ~10ns/pf.
Internal delay: output rises $\sim((f_{comp}+1)+4f_{true})\tau$, falls $\sim(8(f_{comp}+1)+f_{true})\tau$.
Complement output time constant: $\sim(20k\Omega/4)\cdot(C_{load}+.03pf)$ ~5ns/pf.
Internal delay: output rises $\sim8(f_{comp}+1)\tau$, output falls $\sim(f_{comp}+1)\tau$

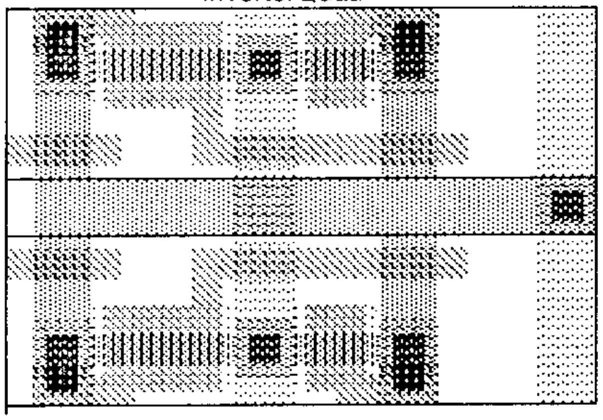
InverterPair



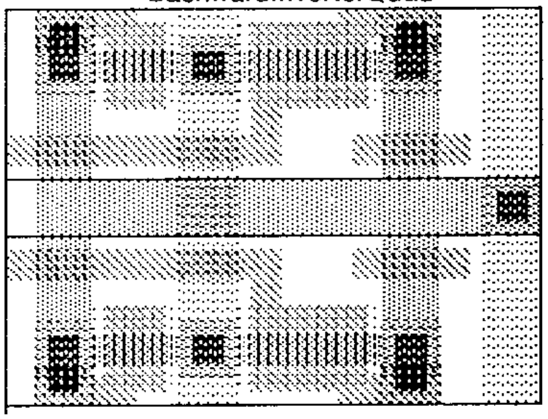
BackwardInverterPair



InverterQuad



BackwardInverterQuad



file: Inverters.cif

```
( Created by Sif from Inverters.ic );

DS 1:    ( Name: InverterPair );
( 27 Items. );
Layer NPol:  Box  Len 2400  Wid 600  Center 1200,-3000 ;
Layer NPol:  Box  Len 1200  Wid 900  Center 1200,-450 ;
Layer NDif:  Box  Len 1200  Wid 3000  Center 1200,-2100 ;
Layer NDif:  Box  Len 11700  Wid 1200  Center 6450,-4200 ;
Layer NMet:  Box  Len 1200  Wid 1800  Center 1200,-900 ;
Layer NCut:  Box  Len 600  Wid 600  Center 1200,-600 ;
Layer NCut:  Box  Len 600  Wid 600  Center 1200,-1200 ;
Layer NPol:  Box  Len 1500  Wid 600  Center 1950,-300 ;
Layer NDif:  Box  Len 3600  Wid 600  Center 3300,-1200 ;
Layer NImp:  Box  Len 3000  Wid 1800  Center 3300,-1200 ;
Layer NPol:  Box  Len 2400  Wid 1800  Center 3300,-1200 ;
Layer NPol:  Box  Len 600  Wid 1200  Center 4200,-2100 ;
Layer NPol:  Box  Len 5700  Wid 600  Center 6750,-3000 ;
Layer NMet:  Box  Len 1200  Wid 4800  Center 5400,-2400 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 5400,-1200 ;
Layer NCut:  Box  Len 600  Wid 600  Center 5400,-1200 ;
Layer NDif:  Box  Len 2700  Wid 600  Center 6750,-1200 ;
Layer NImp:  Box  Len 1800  Wid 1800  Center 6900,-1200 ;
Layer NPol:  Box  Len 1200  Wid 1800  Center 6900,-1200 ;
Layer NPol:  Box  Len 1200  Wid 600  Center 7500,-300 ;
Layer NDif:  Box  Len 1200  Wid 3000  Center 8400,-2100 ;
Layer NPol:  Box  Len 1200  Wid 900  Center 8400,-450 ;
Layer NMet:  Box  Len 1200  Wid 1800  Center 8400,-900 ;
Layer NCut:  Box  Len 600  Wid 600  Center 8400,-600 ;
Layer NCut:  Box  Len 600  Wid 600  Center 8400,-1200 ;
Layer NMet:  Box  Len 1200  Wid 4800  Center 11700,-2400 ;
Layer NCut:  Box  Len 600  Wid 600  Center 11700,-4200 ;
DF;

DS 2:    ( Name: InverterQuad );
( 4 Items. );
Call 1 Trans 0,0;
Call 1 Mir Y  Trans 0,-8400;
Layer NMet:  Box  Len 1200  Wid 3600  Center 5400,-6600 ;
Layer NMet:  Box  Len 1200  Wid 3600  Center 11700,-6600 ;
DF;

DS 3:    ( Name: BackwardInverterPair );
( 27 Items. );
Layer NPol:  Box  Len 5700  Wid 600  Center 2850,-3000 ;
Layer NDif:  Box  Len 1200  Wid 3000  Center 1200,-2100 ;
Layer NPol:  Box  Len 1200  Wid 900  Center 1200,-450 ;
Layer NMet:  Box  Len 1200  Wid 1800  Center 1200,-900 ;
Layer NDif:  Box  Len 10500  Wid 1200  Center 5850,-4200 ;
Layer NCut:  Box  Len 600  Wid 600  Center 1200,-600 ;
Layer NCut:  Box  Len 600  Wid 600  Center 1200,-1200 ;
Layer NDif:  Box  Len 2700  Wid 600  Center 2850,-1200 ;
Layer NPol:  Box  Len 1200  Wid 600  Center 2100,-300 ;
Layer NImp:  Box  Len 1800  Wid 1800  Center 2700,-1200 ;
Layer NPol:  Box  Len 1200  Wid 1800  Center 2700,-1200 ;
Layer NMet:  Box  Len 1200  Wid 4800  Center 4200,-2400 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 4200,-1200 ;
Layer NCut:  Box  Len 600  Wid 600  Center 4200,-1200 ;
Layer NDif:  Box  Len 3600  Wid 600  Center 6300,-1200 ;
Layer NImp:  Box  Len 3000  Wid 1800  Center 6300,-1200 ;
Layer NPol:  Box  Len 2400  Wid 1800  Center 6300,-1200 ;
Layer NPol:  Box  Len 600  Wid 1200  Center 5400,-2100 ;
Layer NPol:  Box  Len 1500  Wid 600  Center 7650,-300 ;
Layer NPol:  Box  Len 2400  Wid 600  Center 8400,-3000 ;
Layer NPol:  Box  Len 1200  Wid 900  Center 8400,-450 ;
Layer NDif:  Box  Len 1200  Wid 3000  Center 8400,-2100 ;
Layer NMet:  Box  Len 1200  Wid 1800  Center 8400,-900 ;
Layer NCut:  Box  Len 600  Wid 600  Center 8400,-600 ;
Layer NCut:  Box  Len 600  Wid 600  Center 8400,-1200 ;
Layer NMet:  Box  Len 1200  Wid 4800  Center 10500,-2400 ;
Layer NCut:  Box  Len 600  Wid 600  Center 10500,-4200 ;
DF;
```

```
DS 4;    ( Name: BackwardInverterQuad );
( 2 Items. );
Call 3 Trans 0,0;
Call 3 Mir Y  Trans 0,-8400;
DF;

End
```

## Documentation for Pullups: HPullup4:1 -- HPullup2:1 VPullup4:1 -- VPullup2:1

| | | | |
|---|---|---|---|
| Date | July 26, 1978 | Status | used in summer 1978 MPC |
| Designer | Bob Baldwin | Address/Phone | PARC SSL |
| Design Rules | Mead/Conway | Scale | $\lambda$ = 3 $\mu$m |
| Info File | Pullups.LibDoc | CIF File | Pullups.cif |

Dimensions and Replication

| | | | | |
|---|---|---|---|---|
| HPullup4:1 | X: 7$\lambda$ | Y: 18$\lambda$ | DX: no | DY: 9$\lambda$ |
| HPullup2:1 | X: 7$\lambda$ | Y: 14$\lambda$ | DX: no | DY: 9$\lambda$ |
| VPullup4:1 | X: 7$\lambda$ | Y: 18$\lambda$ | DX: 9$\lambda$ | DY: no |
| VPullup2:1 | X: 7$\lambda$ | Y: 14$\lambda$ | DX: 9$\lambda$ | DY: no |

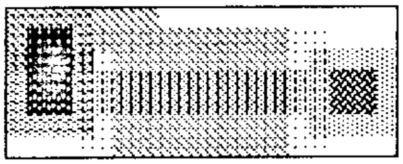
Further Info

Function/Use General purpose pullups with given L/W ratios.

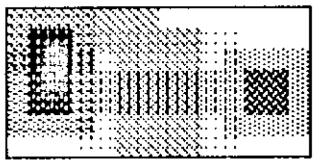
Connections The symbols contain diffusion and cut flashes for Vdd, without metal, and complete butting contacts for connection to the circuit.

Included Cells None

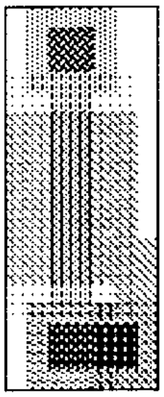
HPullup4:1



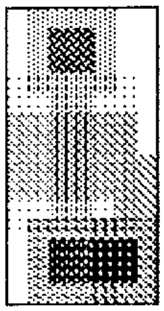
HPullup2:1



VPullup4:1



VPullup2:1



file: Pullups.cif

```
( Created by Sif from Pullups.ic );

DS 1;    ( Name: HPullup4:1 );
( 11 Items. );
Layer NMet:  Box  Len 1200  Wid 1800  Center 600,-900 ;
Layer NPol:  Box  Len 1200  Wid 900  Center 600,-450 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 600,-1200 ;
Layer NCut:  Box  Len 600  Wid 600  Center 600,-600 ;
Layer NCut:  Box  Len 600  Wid 600  Center 600,-1200 ;
Layer NPol:  Box  Len 1500  Wid 600  Center 1350,-300 ;
Layer NDif:  Box  Len 3600  Wid 600  Center 2700,-1200 ;
Layer NImp:  Box  Len 3400  Wid 1800  Center 2700,-1200 ;
Layer NPol:  Box  Len 2400  Wid 1800  Center 2700,-1200 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 4800,-1200 ;
Layer NCut:  Box  Len 600  Wid 600  Center 4800,-1200 ;
DF;

DS 2;    ( Name: VPullup4:1 );
( 11 Items. );
Layer NImp:  Box  Len 1800  Wid 3400  Center 900,-2700 ;
Layer NPol:  Box  Len 1800  Wid 2400  Center 900,-2700 ;
Layer NMet:  Box  Len 1800  Wid 1200  Center 1200,-4800 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 900,-4800 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 900,-600 ;
Layer NCut:  Box  Len 600  Wid 600  Center 900,-4800 ;
Layer NDif:  Box  Len 600  Wid 3600  Center 900,-2700 ;
Layer NCut:  Box  Len 600  Wid 600  Center 900,-600 ;
Layer NPol:  Box  Len 900  Wid 1200  Center 1650,-4800 ;
Layer NCut:  Box  Len 600  Wid 600  Center 1500,-4800 ;
Layer NPol:  Box  Len 600  Wid 1500  Center 1800,-4050 ;
DF;

DS 3;    ( Name: VPullup2:1 );
( 11 Items. );
Layer NImp:  Box  Len 1800  Wid 2200  Center 900,-2100 ;
Layer NPol:  Box  Len 1800  Wid 1200  Center 900,-2100 ;
Layer NMet:  Box  Len 1800  Wid 1200  Center 1200,-3600 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 900,-3600 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 900,-600 ;
Layer NDif:  Box  Len 600  Wid 2700  Center 900,-1950 ;
Layer NCut:  Box  Len 600  Wid 600  Center 900,-3600 ;
Layer NCut:  Box  Len 600  Wid 600  Center 900,-600 ;
Layer NPol:  Box  Len 900  Wid 1200  Center 1650,-3600 ;
Layer NCut:  Box  Len 600  Wid 600  Center 1500,-3600 ;
Layer NPol:  Box  Len 600  Wid 1200  Center 1800,-2700 ;
DF;

DS 4;    ( Name: HPullup2:1 );
( 11 Items. );
Layer NMet:  Box  Len 1200  Wid 1800  Center 600,-900 ;
Layer NPol:  Box  Len 1200  Wid 900  Center 600,-450 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 600,-1200 ;
Layer NCut:  Box  Len 600  Wid 600  Center 600,-600 ;
Layer NCut:  Box  Len 600  Wid 600  Center 600,-1200 ;
Layer NPol:  Box  Len 1200  Wid 600  Center 1500,-300 ;
Layer NDif:  Box  Len 2700  Wid 600  Center 2250,-1200 ;
Layer NImp:  Box  Len 2200  Wid 1800  Center 2100,-1200 ;
Layer NPol:  Box  Len 1200  Wid 1800  Center 2100,-1200 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 3600,-1200 ;
Layer NCut:  Box  Len 600  Wid 600  Center 3600,-1200 ;
DF;

End
```

## Documentation for PLA: PlaCellPair -- PlaConnect -- PlaGround PlaIn -- PlaInPair -- PlaOut PullupPair -- PlaProgFlash -- PLA-2-4-4

| | | | |
|---|---|---|---|
| Date | August 29, 1978 | Status | used in summer 1978 MPC |
| Designer | Dick Lyon | Address/Phone | PARC SSL x4325 |
| Design Rules | Mead/Conway | Scale | $\lambda = 3\ \mu m$ |
| Info File | PLA.LibDoc | CIF File | PLA.cif |

Dimensions and Replication

| | | | | |
|---|---|---|---|---|
| PlaCellPair | X: $14\lambda$ | Y: $14\lambda$ | DX: $14\lambda$ | DY: $14\lambda$ |
| PlaConnect | X: $16\lambda$ | Y: $14\lambda$ | DX: no | DY: $14\lambda$ |
| PlaGround | X: $14\lambda$ | Y: $10\lambda$ | DX: $14\lambda$ | DY: no |
| PlaIn | X: $16\lambda$ | Y: $53\lambda$ | DX: no | DY: no |
| PlaInPair | X: $28\lambda$ | Y: $53\lambda$ | DX: $28\lambda$ | DY: no |
| PlaOut | X: $14\lambda$ | Y: $68\lambda$ | DX: $14\lambda$ | DY: no |
| PullupPair | X: $29\lambda$ | Y: $14\lambda$ | DX: no | DY: $14\lambda$ |
| PlaProgFlash | X: $4\lambda$ | Y: $4\lambda$ | DX: no | DY: no |
| PLA-2-4-4 | X: $114\lambda$ | Y: $139\lambda$ | DX: no | DY: no |

Further Info — See [Mead 1978] chapter 3, section "The Programmable Logic Array."

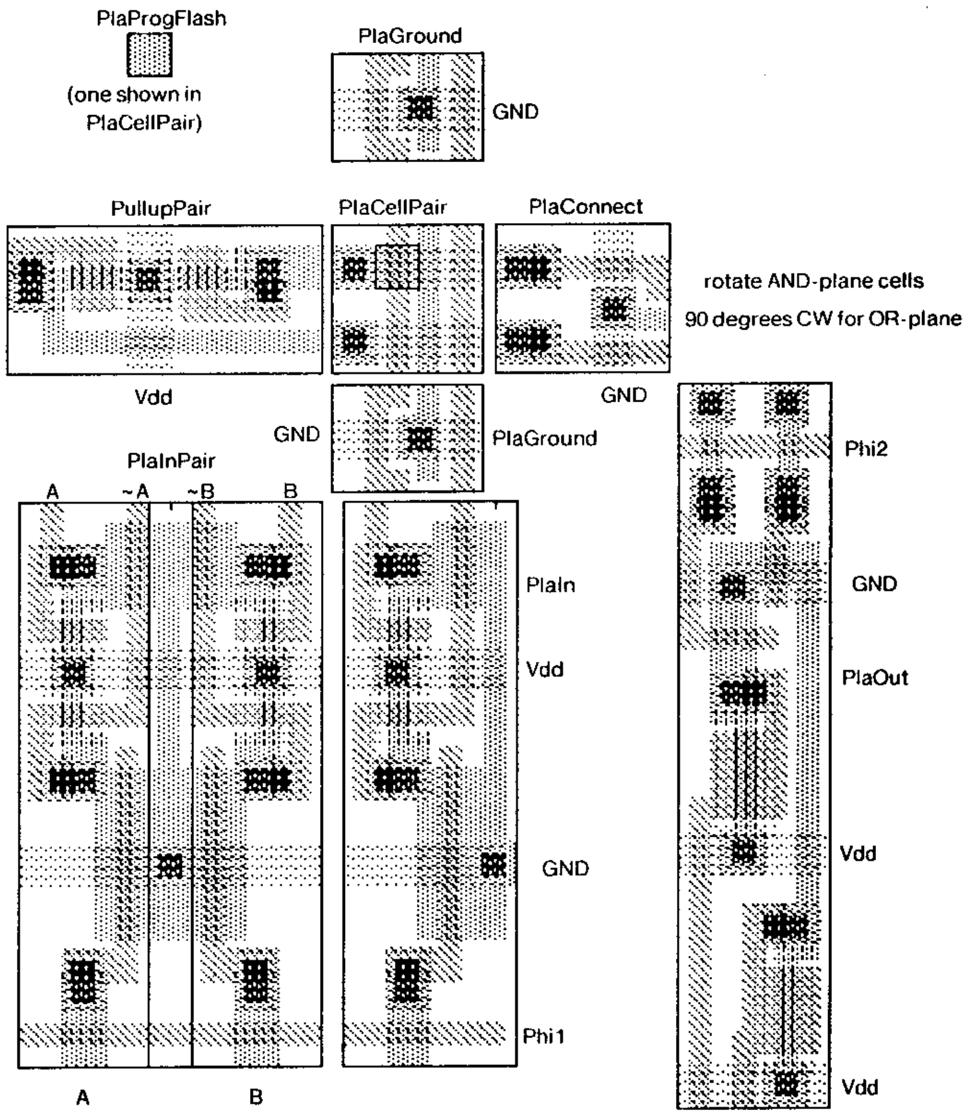
Function/Use — General logic and state machines.

Connections — See the checkplots of various cells, and of PLA-2-4-4, which has one output connected back to an input, for one bit of state.

Loadings — Each input gate is $32\lambda^2$.
Phi1 drives $8\lambda^2$ per input.
Phi2 drives $4\lambda^2$ per output.
Outputs have $Z=4$ pullups.

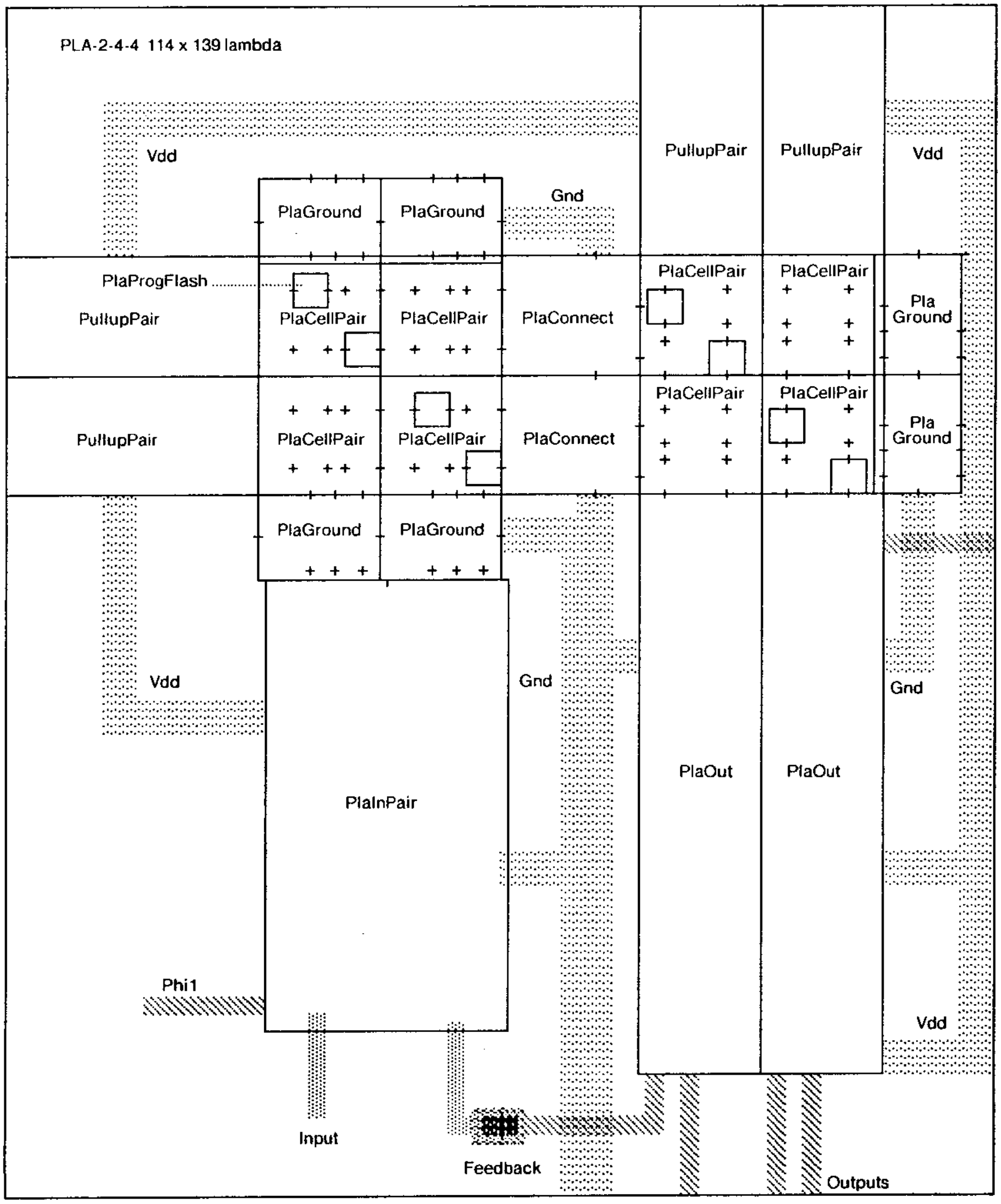
Performance — Not computed, but limited by the $Z=2$ pullups on the minterm lines, which may have large fanouts.

How it works — The AND-plane is made of NOR gates, so programming flashes "PlaProgFlash" should be placed to form a path across the "~A" poly line to include "A" in a minterm, and vice versa. The OR-plane is also made of NOR gates, and since the minterms are non-inverted, just place a "PlaProgFlash" to include a minterm in an output. The output of the OR-plane is thus inverted, but PlaOut fixes it.



PlaProgFlash
(one shown in PlaCellPair)
PlaGround
GND
PullupPair
PlaCellPair
PlaConnect
rotate AND-plane cells
90 degrees CW for OR-plane
Vdd
GND
GND
PlaGround
PlaInPair
A
~A
~B
B
PlaIn
Vdd
GND
Phi1
A
B
Phi2
GND
PlaOut
Vdd
Vdd

PLA Cells



Checkplot of PLA-2-4-4

file: PLA.cif

```
( Created by Sif from Pla.ic );

DS 1:     ( Name: PlaCellPair );
( 9 Items. );
Layer NMet:  Box  Len 4200  Wid 1200  Center 2100,-1200 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 600,-1200 ;
Layer NMet:  Box  Len 4200  Wid 1200  Center 2100,-3300 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 600,-3300 ;
Layer NCut:  Box  Len 600  Wid 600  Center 600,-1200 ;
Layer NCut:  Box  Len 600  Wid 600  Center 600,-3300 ;
Layer NPol:  Box  Len 600  Wid 4200  Center 1800,-2100 ;
Layer NDif:  Box  Len 600  Wid 4200  Center 2700,-2100 ;
Layer NPol:  Box  Len 600  Wid 4200  Center 3600,-2100 ;
DF;

DS 2:     ( Name: PullupPair );
( 22 Items. );
Layer NDif:  Box  Len 1500  Wid 1200  Center 750,-1800 ;
Layer NPol:  Box  Len 1200  Wid 1200  Center 600,-900 ;
Layer NMet:  Box  Len 1200  Wid 1800  Center 600,-1500 ;
Layer NCut:  Box  Len 600  Wid 1200  Center 600,-1500 ;
Layer NDif:  Box  Len 7800  Wid 600  Center 4800,-3300 ;
Layer NDif:  Box  Len 2700  Wid 600  Center 2250,-1500 ;
Layer NDif:  Box  Len 600  Wid 1500  Center 1200,-2850 ;
Layer NImp:  Box  Len 2400  Wid 1800  Center 2400,-1500 ;
Layer NPol:  Box  Len 1800  Wid 600  Center 2100,-600 ;
Layer NPol:  Box  Len 1200  Wid 1800  Center 2400,-1500 ;
Layer NMet:  Box  Len 1200  Wid 4200  Center 3900,-2100 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 3900,-1500 ;
Layer NCut:  Box  Len 600  Wid 600  Center 3900,-1500 ;
Layer NCut:  Box  Len 600  Wid 600  Center 3900,-1500 ;
Layer NDif:  Box  Len 2700  Wid 600  Center 5550,-1500 ;
Layer NImp:  Box  Len 2400  Wid 1800  Center 5400,-1500 ;
Layer NPol:  Box  Len 1200  Wid 1800  Center 5400,-1500 ;
Layer NPol:  Box  Len 1800  Wid 600  Center 5700,-2400 ;
Layer NDif:  Box  Len 2400  Wid 1200  Center 7500,-1200 ;
Layer NPol:  Box  Len 1200  Wid 1200  Center 7200,-2100 ;
Layer NMet:  Box  Len 1200  Wid 1800  Center 7200,-1500 ;
Layer NCut:  Box  Len 600  Wid 1200  Center 7200,-1500 ;
DF;

DS 3:     ( Name: PlaConnect );
( 17 Items. );
Layer NMet:  Box  Len 1800  Wid 1200  Center 900,-1200 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 600,-1200 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 600,-3300 ;
Layer NMet:  Box  Len 1800  Wid 1200  Center 900,-3300 ;
Layer NCut:  Box  Len 600  Wid 600  Center 600,-1200 ;
Layer NCut:  Box  Len 600  Wid 600  Center 600,-3300 ;
Layer NCut:  Box  Len 600  Wid 600  Center 1200,-1200 ;
Layer NCut:  Box  Len 600  Wid 600  Center 1200,-3300 ;
Layer NPol:  Box  Len 900  Wid 1200  Center 1350,-1200 ;
Layer NPol:  Box  Len 900  Wid 1200  Center 1350,-3300 ;
Layer NPol:  Box  Len 3600  Wid 600  Center 3000,-3600 ;
Layer NPol:  Box  Len 3000  Wid 600  Center 3300,-1200 ;
Layer NMet:  Box  Len 1200  Wid 4200  Center 3300,-2100 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 3300,-2400 ;
Layer NCut:  Box  Len 600  Wid 600  Center 3300,-2400 ;
Layer NDif:  Box  Len 1500  Wid 600  Center 4050,-2700 ;
Layer NPol:  Box  Len 600  Wid 1200  Center 4500,-1500 ;
DF;

DS 4:     ( Name: PlaGround );
( 8 Items. );
Layer NMet:  Box  Len 4200  Wid 1200  Center 2100,-1500 ;
Layer NPol:  Box  Len 600  Wid 2400  Center 1200,-1500 ;
Layer NPol:  Box  Len 1200  Wid 600  Center 1500,-300 ;
Layer NPol:  Box  Len 1200  Wid 600  Center 1500,-2700 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 2400,-1500 ;
Layer NCut:  Box  Len 600  Wid 600  Center 2400,-1500 ;
Layer NDif:  Box  Len 600  Wid 2700  Center 2700,-1350 ;
Layer NPol:  Box  Len 600  Wid 3000  Center 3600,-1500 ;
```

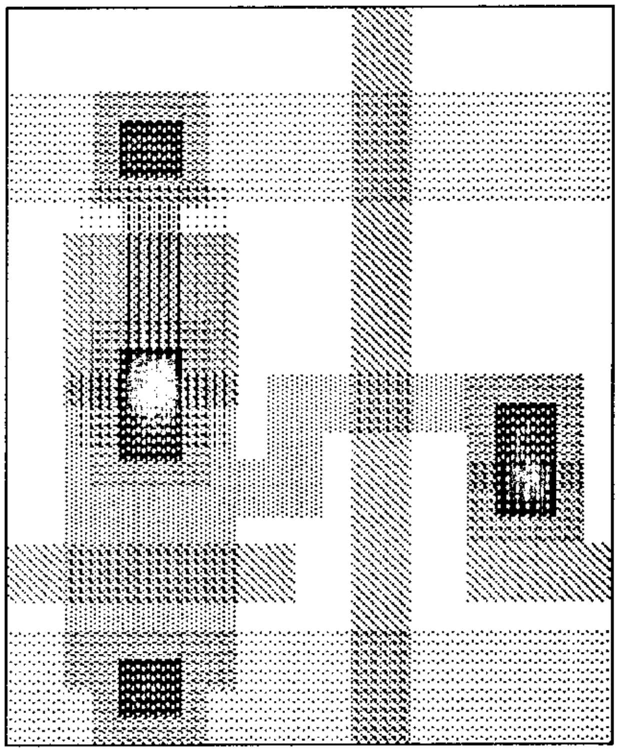
```
DF:


DS 5;     ( Name: PlaIn );
( 35 Items. );
Layer NMet:  Box  Len 4500  Wid 1200  Center -1950,-4800 ;
Layer NMet:  Box  Len 4500  Wid 1200  Center -1950,-10200 ;
Layer NPol:  Box  Len 4500  Wid 600  Center -1950,-15000 ;
Layer NPol:  Box  Len 1200  Wid 1200  Center -3300,-7800 ;
Layer NPol:  Box  Len 600  Wid 1800  Center -3600,-6600 ;
Layer NPol:  Box  Len 600  Wid 1800  Center -3600,-3000 ;
Layer NPol:  Box  Len 1200  Wid 1200  Center -3300,-1800 ;
Layer NMet:  Box  Len 1800  Wid 1200  Center -2700,-7800 ;
Layer NImp:  Box  Len 1800  Wid 1800  Center -2700,-6000 ;
Layer NPol:  Box  Len 2700  Wid 600  Center -2250,-6000 ;
Layer NImp:  Box  Len 1800  Wid 1800  Center -2700,-3600 ;
Layer NPol:  Box  Len 1800  Wid 600  Center -2700,-3600 ;
Layer NMet:  Box  Len 1800  Wid 1200  Center -2700,-1800 ;
Layer NPol:  Box  Len 600  Wid 1500  Center -3300,-750 ;
Layer NCut:  Box  Len 1200  Wid 600  Center -2700,-7800 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center -2700,-4800 ;
Layer NCut:  Box  Len 1200  Wid 600  Center -2700,-1800 ;
Layer NDif:  Box  Len 1200  Wid 1800  Center -2400,-7500 ;
Layer NDif:  Box  Len 600  Wid 1800  Center -2700,-6000 ;
Layer NDif:  Box  Len 600  Wid 300  Center -2700,-6750 ;
Layer NCut:  Box  Len 600  Wid 600  Center -2700,-4800 ;
Layer NDif:  Box  Len 600  Wid 1800  Center -2700,-3900 ;
Layer NDif:  Box  Len 600  Wid 300  Center -2700,-2850 ;
Layer NDif:  Box  Len 1200  Wid 1800  Center -2400,-2100 ;
Layer NDif:  Box  Len 1200  Wid 2700  Center -2400,-14550 ;
Layer NMet:  Box  Len 1200  Wid 1800  Center -2400,-13500 ;
Layer NPol:  Box  Len 2100  Wid 900  Center -1950,-13050 ;
Layer NCut:  Box  Len 600  Wid 1200  Center -2400,-13500 ;
Layer NDif:  Box  Len 2400  Wid 4800  Center -900,-9900 ;
Layer NDif:  Box  Len 2100  Wid 2400  Center -750,-1800 ;
Layer NPol:  Box  Len 600  Wid 6000  Center -1200,-9900 ;
Layer NPol:  Box  Len 600  Wid 6300  Center -900,-3150 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 0,-10200 ;
Layer NDif:  Box  Len 600  Wid 6000  Center 0,-4800 ;
Layer NCut:  Box  Len 600  Wid 600  Center 0,-10200 ;
DF;


DS 6;     ( Name: PlaInPair );
( 2 Items. );
Call 5 Trans 0,0;
Call 5 Mir X  Trans 0,0;
DF;


DS 7:     ( Name: PlaOut );
( 50 Items. ):
Layer NPol:  Box  Len 4200  Wid 600  Center 2100,-1800 ;
Layer NPol:  Box  Len 600  Wid 3600  Center 300,-5400 ;
Layer NPol:  Box  Len 2700  Wid 600  Center 1350,-7200 ;
Layer NMet:  Box  Len 4200  Wid 1200  Center 2100,-5700 ;
Layer NMet:  Box  Len 4200  Wid 1200  Center 2100,-13200 ;
Layer NMet:  Box  Len 4200  Wid 1200  Center 2100,-19800 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 900,-600 ;
Layer NMet:  Box  Len 1200  Wid 1200  Center 900,-600 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 900,-3000 ;
Layer NPol:  Box  Len 1200  Wid 900  Center 900,-3750 ;
Layer NMet:  Box  Len 1200  Wid 1800  Center 900,-3300 ;
Layer NPol:  Box  Len 600  Wid 8700  Center 600,-16050 ;
Layer NCut:  Box  Len 600  Wid 600  Center 900,-600 ;
Layer NDif:  Box  Len 600  Wid 1800  Center 900,-1800 ;
Layer NCut:  Box  Len 600  Wid 1200  Center 900,-3300 ;
Layer NDif;  Box  Len 1200  Wid 3600  Center 1500,-6300 ;
Layer NMet:  Box  Len 1800  Wid 1200  Center 1800,-8700 ;
Layer NDif:  Box  Len 1200  Wid 1500  Center 1500,-8850 ;
Layer NPol:  Box  Len 1800  Wid 2400  Center 1800,-11100 ;
Layer NImp;  Box  Len 1800  Wid 3600  Center 1800,-11100 ;
Layer NCut:  Box  Len 600  Wid 600  Center 1500,-5700 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 1800,-8700 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 1800,-13200 ;
Layer NDif:  Box  Len 600  Wid 3900  Center 1800,-11250 ;
Layer NCut:  Box  Len 600  Wid 600  Center 1800,-13200 ;
Layer NPol:  Box  Len 600  Wid 2100  Center 1800,-19350 ;
Layer NDif:  Box  Len 2100  Wid 1200  Center 2850,-5100 ;
Layer NPol:  Box  Len 1200  Wid 1200  Center 2400,-8700 ;
Layer NPol:  Box  Len 1200  Wid 1200  Center 2400,-15300 ;
Layer NPol:  Box  Len 600  Wid 3600  Center 2100,-17100 ;
Layer NMet:  Box  Len 1800  Wid 1200  Center 3000,-15300 ;
Layer NImp:  Box  Len 1800  Wid 3600  Center 3000,-17700 ;
Layer NPol:  Box  Len 1800  Wid 2400  Center 3000,-17700 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 3000,-600 ;
Layer NMet:  Box  Len 1200  Wid 1200  Center 3000,-600 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 3000,-3000 ;
Layer NPol:  Box  Len 1200  Wid 900  Center 3000,-3750 ;
Layer NMet:  Box  Len 1200  Wid 1800  Center 3000,-3300 ;
Layer NPol:  Box  Len 600  Wid 2400  Center 2700,-5100 ;
Layer NPol:  Box  Len 600  Wid 3600  Center 2700,-10500 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 3000,-15300 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 3000,-19800 ;
Layer NCut:  Box  Len 600  Wid 600  Center 3000,-600 ;
Layer NDif:  Box  Len 600  Wid 1800  Center 3000,-1800 ;
Layer NCut:  Box  Len 600  Wid 1200  Center 3000,-3300 ;
Layer NDif:  Box  Len 1200  Wid 1500  Center 3300,-15450 ;
Layer NDif:  Box  Len 600  Wid 3900  Center 3000,-17550 ;
Layer NCut:  Box  Len 600  Wid 600  Center 3000,-19800 ;
Layer NCut:  Box  Len 600  Wid 600  Center 3000,-19800 ;
Layer NDif:  Box  Len 600  Wid 10200  Center 3600,-10500 ;
DF;


DS 8;    ( Name: PLA-2-4-4 );
( 42 Items. );
Call 2 Trans 0,-8700;
Call 2 Trans 0,-12900;
Layer NMet:  Box  Len 1200  Wid 5400  Center 3900,-6000 ;
Layer NMet:  Box  Len 1200  Wid 8400  Center 3900,-21300 ;
Layer NMet:  Box  Len 4500  Wid 1200  Center 6750,-24900 ;
Layer NMet:  Box  Len 17400  Wid 1200  Center 13200,-3900 ;
Layer NPol:  Box  Len 4200  Wid 600  Center 6900,-35100 ;
Call 1 Trans 8700,-8700;
Call 1 Trans 8700,-12900;
Call 1 Trans 12900,-8700;
Call 1 Trans 12900,-12900;
Call 4 Trans 8700,-17100;
Call 4 Trans 12900,-17100;
Call 4 Trans 8700,-6000;
Call 4 Trans 12900,-6000;
Call 6 Trans 13200,-20100;
Layer NDif:  Box  Len 600  Wid 3600  Center 10800,-37200 ;
Layer NDif:  Box  Len 600  Wid 3600  Center 15600,-37500 ;
Layer NDif:  Box  Len 1200  Wid 600  Center 15900,-39300 ;
Layer NMet:  Box  Len 1800  Wid 1200  Center 17100,-39300 ;
Layer NDif:  Box  Len 1200  Wid 1200  Center 16800,-39300 ;
Layer NCut:  Box  Len 600  Wid 600  Center 16800,-39300 ;
Call 3 Trans 17100,-8700;
Call 3 Trans 17100,-12900;
Layer NMet:  Box  Len 3900  Wid 1200  Center 19050,-18600 ;
Layer NMet:  Box  Len 3900  Wid 1200  Center 19050,-7500 ;
Layer NMet:  Box  Len 3300  Wid 1200  Center 18750,-30300 ;
Layer NCut:  Box  Len 600  Wid 600  Center 17400,-39300 ;
Layer NPol:  Box  Len 900  Wid 1200  Center 17550,-39300 ;
Layer NPol:  Box  Len 5100  Wid 600  Center 20250,-39300 ;
Layer NMet:  Box  Len 1800  Wid 22500  Center 20100,-30450 ;
Layer NMet:  Box  Len 1200  Wid 1500  Center 20400,-7950 ;
Layer NMet:  Box  Len 1200  Wid 1200  Center 20400,-17700 ;
Layer NMet:  Box  Len 1500  Wid 1200  Center 21150,-22800 ;
Call 1 Rot 0,-1  Trans 30300,-8700;
Call 1 Rot 0,-1  Trans 30300,-12900;
Call 1 Rot 0,-1  Trans 26100,-8700;
Call 1 Rot 0,-1  Trans 26100,-12900;
Call 2 Rot 0,-1  Trans 30300,0;
Call 2 Rot 0,-1  Trans 26100,0;
Call 7 Trans 21900,-17100;
Call 7 Trans 26100,-17100;
Layer NPol:  Box  Len 600  Wid 1800  Center 22500,-38400 ;
Layer NPol:  Box  Len 600  Wid 4200  Center 23700,-39600 ;
Layer NPol:  Box  Len 600  Wid 4200  Center 26700,-39600 ;
Layer NPol:  Box  Len 600  Wid 4200  Center 27900,-39600 ;
Call 4 Rot 0,-1  Trans 33000,-8700;
Call 4 Rot 0,-1  Trans 33000,-12900;
Layer NPol:  Box  Len 3900  Wid 600  Center 32250,-18900 ;
Layer NMet:  Box  Len 1800  Wid 1200  Center 31200,-22800 ;
Layer NMet:  Box  Len 2700  Wid 1200  Center 31650,-3900 ;
Layer NMet:  Box  Len 3000  Wid 1200  Center 31800,-30300 ;
Layer NMet:  Box  Len 3000  Wid 1200  Center 31800,-36900 ;
Layer NMet:  Box  Len 1200  Wid 6000  Center 31500,-20100 ;
Layer NMet:  Box  Len 1200  Wid 34200  Center 33600,-20400 ;
DF;

End
```

## Documentation for SRCELL

| | | | |
|---|---|---|---|
| Date | August 1, 1978 | Status | Layout only |
| Designer | Lynn Conway | Address/Phone | PARC SSL |
| Design Rules | Mead/Conway | Scale | $\lambda = 3\ \mu m$ |
| Info File | SRCELL.LibDoc | CIF File | SRCELL.cif |
| Dimensions | X: 21λ Y: 26λ | Replication<br>Mirrored | DX: 21λ DY: 26λ<br>DX: 21λ DY: 19λ* |

| | |
|---|---|
| Further Info | See Chapter 4 of Mead & Conway |
| Function/Use | Shift Register. Each cell contains an inverter and a pass gate control line. |
| Connections | Horizontal power and ground in 4λ metal. Vertical clock/control line in 2λ poly. The input and output are horizontal in 2λ poly. The cell can be stacked in X and Y to form a shift register of arbitrary length and width. *The cells can be packed closer in the Y direction if every other cell is mirrored about the X axis, thus allowing adjacent cells to share power and/or ground buses. |
| Included Cells | None |
| Loadings | Input load: $12\lambda^2$ gate = .045pf. $Z_{pu}/Z_{pd} = 8$.<br>The output passes through a minimum pass transistor. |
| Performance | Fall time for output: $RC+f_T$ Rise time: $RC+3f_T$. Where RC = time constant for the pass transistor. |

SRCELL



file: SRCELL.cif

```
( Created by Sif from SRCELL.ic );

DS 1;    ( Name: SRCELL );
( 24 Items. );
Layer NMet;  Box  Len 6300  Wid 1200  Center 3150,-7200 ;
Layer NMet;  Box  Len 6300  Wid 1200  Center 3150,-1500 ;
Layer NPol;  Box  Len 3000  Wid 600  Center 1500,-6000 ;
Layer NDif;  Box  Len 1800  Wid 3300  Center 1500,-5550 ;
Layer NPol;  Box  Len 1800  Wid 1800  Center 1500,-3300 ;
Layer NImp;  Box  Len 1600  Wid 2800  Center 1500,-3300 ;
Layer NDif;  Box  Len 1200  Wid 1200  Center 1500,-7200 ;
Layer NDif;  Box  Len 1200  Wid 1200  Center 1500,-1500 ;
Layer NMet;  Box  Len 1200  Wid 1800  Center 1500,-4200 ;
Layer NCut;  Box  Len 600  Wid 600  Center 1500,-7200 ;
Layer NCut;  Box  Len 600  Wid 600  Center 1500,-1500 ;
Layer NCut;  Box  Len 600  Wid 600  Center 1500,-3900 ;
Layer NCut;  Box  Len 600  Wid 600  Center 1500,-4500 ;
Layer NDif;  Box  Len 600  Wid 2100  Center 1500,-3150 ;
Layer NDif;  Box  Len 600  Wid 600  Center 2700,-5100 ;
Layer NDif;  Box  Len 600  Wid 1200  Center 3000,-4800 ;
Layer NDif;  Box  Len 2100  Wid 600  Center 3750,-4200 ;
Layer NPol;  Box  Len 600  Wid 7800  Center 3900,-3900 ;
Layer NPol;  Box  Len 1500  Wid 600  Center 5550,-6000 ;
Layer NPol;  Box  Len 1200  Wid 900  Center 5400,-5250 ;
Layer NDif;  Box  Len 1200  Wid 1200  Center 5400,-4500 ;
Layer NMet;  Box  Len 1200  Wid 1800  Center 5400,-4800 ;
Layer NCut;  Box  Len 600  Wid 600  Center 5400,-4500 ;
Layer NCut;  Box  Len 600  Wid 600  Center 5400,-5100 ;
DF;

Call 1 Trans 0,0;
End
```

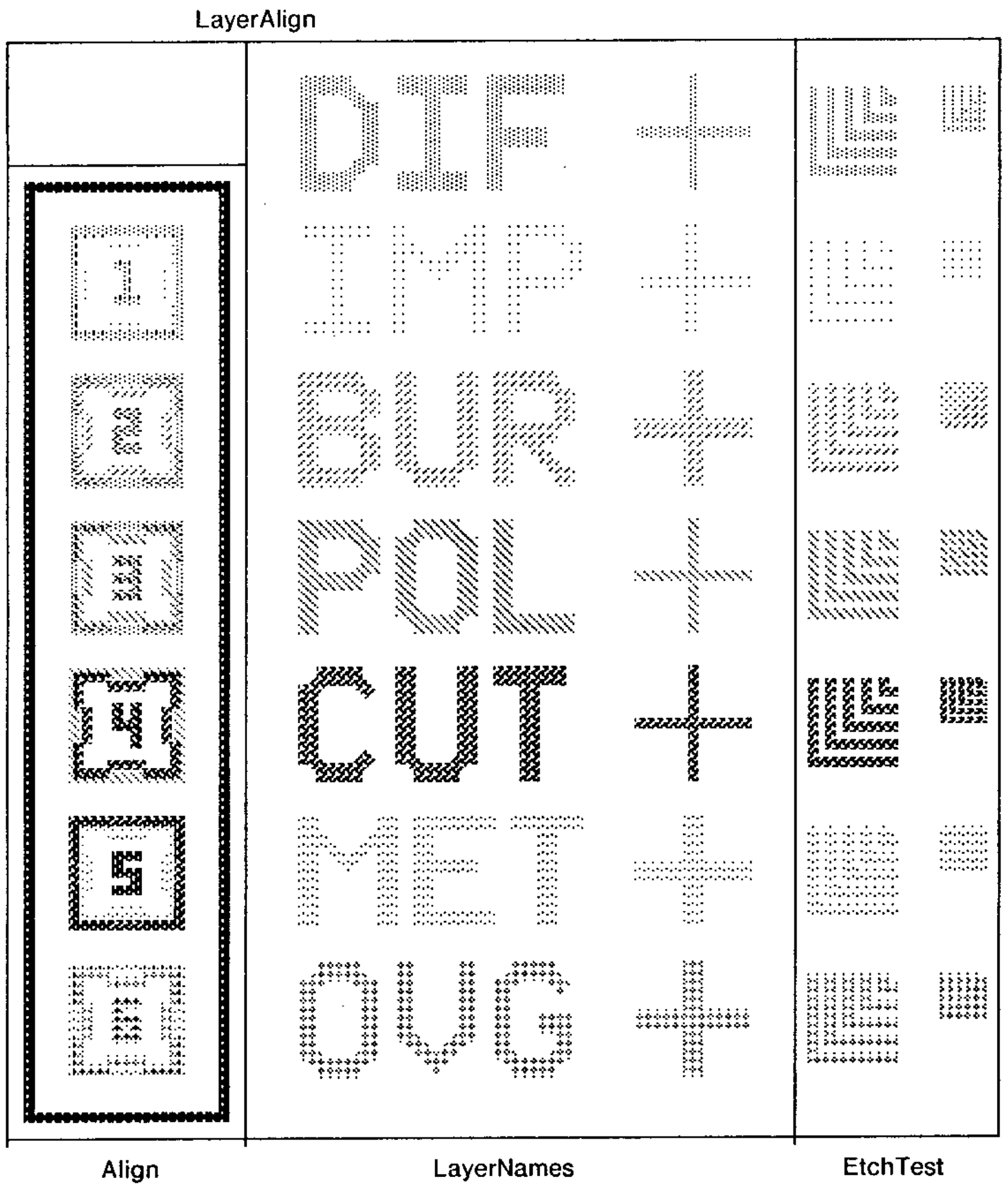
## Documentation for AlignMarks6: Align -- LayerNames EtchTest -- LayerAlign

| | | | |
|---|---|---|---|
| Date | August 29, 1978 | Status | similar to summer 1978 MPC |
| Designer | Hon, Lyon, and Davies | Address/Phone | PARC SSL x4325 |
| Design Rules | Mead/Conway | Scale | $\lambda$ = 3 $\mu$m |
| Info File | AlignMarks6.LibDoc | CIF File | AlignMarks6.cif |

Dimensions and Replication

| | | | | |
|---|---|---|---|---|
| Align | X: 54$\lambda$ | Y: 224$\lambda$ | DX: no | DY: no |
| LayerNames | X: 124$\lambda$ | Y: 232$\lambda$ | DX: no | DY: no |
| EtchTest | X: 46$\lambda$ | Y: 232$\lambda$ | DX: no | DY: no |
| LayerAlign | X: 224$\lambda$ | Y: 232$\lambda$ | DX: no | DY: no |

| | |
|---|---|
| Further Info | See *A Guide to LSI Implementation*, chapter 7. |
| Function/Use | Starting frame artifacts for project chips. See attached figure. |
| Connections | none |



AlignMarks6

file: AlignMarks6.cif

```
( Created by Sif from AlignMarks6.ic );

DS 1:     ( Name: EtchTest );
( 140 Items. );
Layer NGls:  Box  Len 600  Wid 5400  Center 1200,8400 ;
Layer NGls:  Box  Len 6000  Wid 600  Center 3900,5400 ;
Layer NMet:  Box  Len 600  Wid 5400  Center 1200,18600 ;
Layer NMet:  Box  Len 6000  Wid 600  Center 3900,15600 ;
Layer NCut:  Box  Len 600  Wid 5400  Center 1200,28800 ;
Layer NCut:  Box  Len 6000  Wid 600  Center 3900,25800 ;
Layer NPol:  Box  Len 600  Wid 5400  Center 1200,39000 ;
Layer NPol:  Box  Len 6000  Wid 600  Center 3900,36000 ;
Layer NBur:  Box  Len 600  Wid 5400  Center 1200,49200 ;
Layer NBur:  Box  Len 6000  Wid 600  Center 3900,46200 ;
Layer NImp:  Box  Len 600  Wid 5400  Center 1200,59400 ;
Layer NImp:  Box  Len 6000  Wid 600  Center 3900,56400 ;
Layer NDif:  Box  Len 600  Wid 5400  Center 1200,69600 ;
Layer NDif:  Box  Len 6000  Wid 600  Center 3900,66600 ;
Layer NGls:  Box  Len 600  Wid 4200  Center 2400,9000 ;
Layer NGls:  Box  Len 4800  Wid 600  Center 4500,6600 ;
Layer NMet:  Box  Len 600  Wid 4200  Center 2400,19200 ;
Layer NMet:  Box  Len 4800  Wid 600  Center 4500,16800 ;
Layer NCut:  Box  Len 600  Wid 4200  Center 2400,29400 ;
Layer NCut:  Box  Len 4800  Wid 600  Center 4500,27000 ;
Layer NPol:  Box  Len 600  Wid 4200  Center 2400,39600 ;
Layer NPol:  Box  Len 4800  Wid 600  Center 4500,37200 ;
Layer NBur:  Box  Len 600  Wid 4200  Center 2400,49800 ;
Layer NBur:  Box  Len 4800  Wid 600  Center 4500,47400 ;
Layer NImp:  Box  Len 600  Wid 4200  Center 2400,60000 ;
Layer NImp:  Box  Len 4800  Wid 600  Center 4500,57600 ;
Layer NDif:  Box  Len 600  Wid 4200  Center 2400,70200 ;
Layer NDif:  Box  Len 4800  Wid 600  Center 4500,67800 ;
Layer NGls:  Box  Len 600  Wid 3000  Center 3600,9600 ;
Layer NGls:  Box  Len 3600  Wid 600  Center 5100,7800 ;
Layer NMet:  Box  Len 600  Wid 3000  Center 3600,19800 ;
Layer NMet:  Box  Len 3600  Wid 600  Center 5100,18000 ;
Layer NCut:  Box  Len 600  Wid 3000  Center 3600,30000 ;
Layer NCut:  Box  Len 3600  Wid 600  Center 5100,28200 ;
Layer NPol:  Box  Len 600  Wid 3000  Center 3600,40200 ;
Layer NPol:  Box  Len 3600  Wid 600  Center 5100,38400 ;
Layer NBur:  Box  Len 600  Wid 3000  Center 3600,50400 ;
Layer NBur:  Box  Len 3600  Wid 600  Center 5100,48600 ;
Layer NImp:  Box  Len 600  Wid 3000  Center 3600,60600 ;
Layer NImp:  Box  Len 3600  Wid 600  Center 5100,58800 ;
Layer NDif:  Box  Len 600  Wid 3000  Center 3600,70800 ;
Layer NDif:  Box  Len 3600  Wid 600  Center 5100,69000 ;
Layer NGls:  Box  Len 600  Wid 1800  Center 4800,10200 ;
Layer NGls:  Box  Len 2400  Wid 600  Center 5700,9000 ;
Layer NMet:  Box  Len 600  Wid 1800  Center 4800,20400 ;
Layer NMet:  Box  Len 2400  Wid 600  Center 5700,19200 ;
Layer NCut:  Box  Len 600  Wid 1800  Center 4800,30600 ;
Layer NCut:  Box  Len 2400  Wid 600  Center 5700,29400 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 4800,40800 ;
Layer NPol:  Box  Len 2400  Wid 600  Center 5700,39600 ;
Layer NBur:  Box  Len 600  Wid 1800  Center 4800,51000 ;
Layer NBur:  Box  Len 2400  Wid 600  Center 5700,49800 ;
Layer NImp:  Box  Len 600  Wid 1800  Center 4800,61200 ;
Layer NImp:  Box  Len 2400  Wid 600  Center 5700,60000 ;
Layer NDif:  Box  Len 600  Wid 1800  Center 4800,71400 ;
Layer NDif:  Box  Len 2400  Wid 600  Center 5700,70200 ;
Layer NGls:  Box  Len 600  Wid 600  Center 6000,10800 ;
Layer NGls:  Box  Len 1200  Wid 600  Center 6300,10200 ;
Layer NMet:  Box  Len 600  Wid 600  Center 6000,21000 ;
Layer NMet:  Box  Len 1200  Wid 600  Center 6300,20400 ;
Layer NCut:  Box  Len 600  Wid 600  Center 6000,31200 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 6300,30600 ;
Layer NPol:  Box  Len 600  Wid 600  Center 6000,41400 ;
Layer NPol:  Box  Len 1200  Wid 600  Center 6300,40800 ;
Layer NBur:  Box  Len 600  Wid 600  Center 6000,51600 ;
Layer NBur:  Box  Len 1200  Wid 600  Center 6300,51000 ;
Layer NImp:  Box  Len 600  Wid 600  Center 6000,61800 ;
Layer NImp:  Box  Len 1200  Wid 600  Center 6300,61200 ;
Layer NDif:  Box  Len 600  Wid 600  Center 6000,72000 ;
Layer NDif:  Box  Len 1200  Wid 600  Center 6300,71400 ;
Layer NGls:  Box  Len 300  Wid 2700  Center 10050,9750 ;
Layer NGls:  Box  Len 3000  Wid 300  Center 11400,8250 ;
Layer NMet:  Box  Len 300  Wid 2700  Center 10050,19950 ;
Layer NMet:  Box  Len 3000  Wid 300  Center 11400,18450 ;
Layer NCut:  Box  Len 300  Wid 2700  Center 10050,30150 ;
Layer NCut:  Box  Len 3000  Wid 300  Center 11400,28650 ;
Layer NPol:  Box  Len 300  Wid 2700  Center 10050,40350 ;
Layer NPol:  Box  Len 3000  Wid 300  Center 11400,38850 ;
Layer NBur:  Box  Len 300  Wid 2700  Center 10050,50550 ;
Layer NBur:  Box  Len 3000  Wid 300  Center 11400,49050 ;
Layer NImp:  Box  Len 300  Wid 2700  Center 10050,60750 ;
Layer NImp:  Box  Len 3000  Wid 300  Center 11400,59250 ;
Layer NDif:  Box  Len 300  Wid 2700  Center 10050,70950 ;
Layer NDif:  Box  Len 3000  Wid 300  Center 11400,69450 ;
Layer NGls:  Box  Len 300  Wid 2100  Center 10650,10050 ;
Layer NGls:  Box  Len 2400  Wid 300  Center 11700,8850 ;
Layer NMet:  Box  Len 300  Wid 2100  Center 10650,20250 ;
Layer NMet:  Box  Len 2400  Wid 300  Center 11700,19050 ;
Layer NCut:  Box  Len 300  Wid 2100  Center 10650,30450 ;
Layer NCut:  Box  Len 2400  Wid 300  Center 11700,29250 ;
Layer NPol:  Box  Len 300  Wid 2100  Center 10650,40650 ;
Layer NPol:  Box  Len 2400  Wid 300  Center 11700,39450 ;
Layer NBur:  Box  Len 300  Wid 2100  Center 10650,50850 ;
Layer NBur:  Box  Len 2400  Wid 300  Center 11700,49650 ;
Layer NImp:  Box  Len 300  Wid 2100  Center 10650,61050 ;
Layer NImp:  Box  Len 2400  Wid 300  Center 11700,59850 ;
Layer NDif:  Box  Len 300  Wid 2100  Center 10650,71250 ;
Layer NDif:  Box  Len 2400  Wid 300  Center 11700,70050 ;
Layer NGls:  Box  Len 300  Wid 1500  Center 11250,10350 ;
Layer NGls:  Box  Len 1800  Wid 300  Center 12000,9450 ;
Layer NMet:  Box  Len 300  Wid 1500  Center 11250,20550 ;
Layer NMet:  Box  Len 1800  Wid 300  Center 12000,19650 ;
Layer NCut:  Box  Len 300  Wid 1500  Center 11250,30750 ;
Layer NCut:  Box  Len 1800  Wid 300  Center 12000,29850 ;
Layer NPol:  Box  Len 300  Wid 1500  Center 11250,40950 ;
Layer NPol:  Box  Len 1800  Wid 300  Center 12000,40050 ;
Layer NBur:  Box  Len 300  Wid 1500  Center 11250,51150 ;
Layer NBur:  Box  Len 1800  Wid 300  Center 12000,50250 ;
Layer NImp:  Box  Len 300  Wid 1500  Center 11250,61350 ;
Layer NImp:  Box  Len 1800  Wid 300  Center 12000,60450 ;
Layer NDif:  Box  Len 300  Wid 1500  Center 11250,71550 ;
Layer NDif:  Box  Len 1800  Wid 300  Center 12000,70650 ;
Layer NGls:  Box  Len 300  Wid 900  Center 11850,10650 ;
Layer NGls:  Box  Len 1200  Wid 300  Center 12300,10050 ;
Layer NMet:  Box  Len 300  Wid 900  Center 11850,20850 ;
Layer NMet:  Box  Len 1200  Wid 300  Center 12300,20250 ;
Layer NCut:  Box  Len 300  Wid 900  Center 11850,31050 ;
Layer NCut:  Box  Len 1200  Wid 300  Center 12300,30450 ;
Layer NPol:  Box  Len 300  Wid 900  Center 11850,41250 ;
Layer NPol:  Box  Len 1200  Wid 300  Center 12300,40650 ;
Layer NBur:  Box  Len 300  Wid 900  Center 11850,51450 ;
Layer NBur:  Box  Len 1200  Wid 300  Center 12300,50850 ;
Layer NImp:  Box  Len 300  Wid 900  Center 11850,61650 ;
Layer NImp:  Box  Len 1200  Wid 300  Center 12300,61050 ;
Layer NDif:  Box  Len 300  Wid 900  Center 11850,71850 ;
Layer NDif:  Box  Len 1200  Wid 300  Center 12300,71250 ;
Layer NGls:  Box  Len 300  Wid 300  Center 12450,10950 ;
Layer NGls:  Box  Len 600  Wid 300  Center 12600,10650 ;
Layer NMet:  Box  Len 300  Wid 300  Center 12450,21150 ;
Layer NMet:  Box  Len 600  Wid 300  Center 12600,20850 ;
Layer NCut:  Box  Len 300  Wid 300  Center 12450,31350 ;
Layer NCut:  Box  Len 600  Wid 300  Center 12600,31050 ;
Layer NPol:  Box  Len 300  Wid 300  Center 12450,41550 ;
Layer NPol:  Box  Len 600  Wid 300  Center 12600,41250 ;
Layer NBur:  Box  Len 300  Wid 300  Center 12450,51750 ;
Layer NBur:  Box  Len 600  Wid 300  Center 12600,51450 ;
Layer NImp:  Box  Len 300  Wid 300  Center 12450,61950 ;
Layer NImp:  Box  Len 600  Wid 300  Center 12600,61650 ;
Layer NDif:  Box  Len 300  Wid 300  Center 12450,72150 ;
Layer NDif:  Box  Len 600  Wid 300  Center 12600,71850 ;
DF;


DS 2;     ( Name: LayerNames );
( 123 Items. );
Layer NPol:  Box  Len 1200  Wid 7800  Center 4200,38700 ;
Layer NCut:  Box  Len 1200  Wid 5400  Center 4200,28500 ;
Layer NMet:  Box  Len 1200  Wid 7800  Center 4200,18300 ;
Layer NBur:  Box  Len 1200  Wid 7800  Center 4200,48900 ;
Layer NImp:  Box  Len 4800  Wid 1200  Center 6000,62400 ;
Layer NImp:  Box  Len 4800  Wid 1200  Center 6000,55800 ;
Layer NDif:  Box  Len 1200  Wid 7800  Center 4200,69300 ;
Layer NGls:  Box  Len 1200  Wid 5400  Center 4200,8100 ;
Layer NPol:  Box  Len 3600  Wid 1200  Center 6000,42000 ;
Layer NPol:  Box  Len 3600  Wid 1200  Center 6000,38400 ;
Layer NCut:  Box  Len 1697  Wid 1200  Center 4800,31200 Direction 1,1;
Layer NCut:  Box  Len 1697  Wid 1200  Center 4800,25800 Direction 1,-1;
Layer NMet:  Box  Len 4243  Wid 1200  Center 5700,20100 Direction 1,-1;
Layer NBur:  Box  Len 3600  Wid 1200  Center 6000,52200 ;
Layer NBur:  Box  Len 3600  Wid 1200  Center 6000,49200 ;
Layer NBur:  Box  Len 3600  Wid 1200  Center 6000,45600 ;
Layer NDif:  Box  Len 3000  Wid 1200  Center 5700,72600 ;
Layer NDif:  Box  Len 3000  Wid 1200  Center 5700,66000 ;
Layer NGls:  Box  Len 1697  Wid 1200  Center 4800,5400 Direction 1,-1;
Layer NGls:  Box  Len 1697  Wid 1200  Center 4800,10800 Direction 1,1;
Layer NCut:  Box  Len 3000  Wid 1200  Center 6300,31800 ;
Layer NCut:  Box  Len 3000  Wid 1200  Center 6300,25200 ;
Layer NGls:  Box  Len 3000  Wid 1200  Center 6300,4800 ;
Layer NGls:  Box  Len 3000  Wid 1200  Center 6300,11400 ;
Layer NImp:  Box  Len 1200  Wid 6600  Center 6000,59100 ;
Layer NMet:  Box  Len 4243  Wid 1200  Center 8100,20100 Direction 1,1;
Layer NDif:  Box  Len 2546  Wid 1200  Center 7500,71700 Direction 1,-1;
Layer NDif:  Box  Len 2546  Wid 1200  Center 7500,66900 Direction 1,1;
Layer NPol:  Box  Len 1697  Wid 1200  Center 7800,41400 Direction 1,-1;
Layer NPol:  Box  Len 1697  Wid 1200  Center 7800,39000 Direction 1,1;
Layer NCut:  Box  Len 1697  Wid 1200  Center 7800,31200 Direction 1,-1;
Layer NCut:  Box  Len 1697  Wid 1200  Center 7800,25800 Direction 1,1;
Layer NBur:  Box  Len 1697  Wid 1200  Center 7800,46200 Direction 1,1;
Layer NBur:  Box  Len 1697  Wid 1200  Center 7800,48600 Direction 1,-1;
Layer NBur:  Box  Len 1697  Wid 1200  Center 7800,49800 Direction 1,1;
Layer NBur:  Box  Len 1697  Wid 1200  Center 7800,51600 Direction 1,-1;
Layer NGls:  Box  Len 1697  Wid 1200  Center 7800,5400 Direction 1,1;
Layer NGls:  Box  Len 1697  Wid 1200  Center 7800,10800 Direction 1,-1;
Layer NPol:  Box  Len 1200  Wid 2400  Center 8400,40200 ;
Layer NBur:  Box  Len 1200  Wid 2400  Center 8400,47400 ;
Layer NBur:  Box  Len 1200  Wid 1800  Center 8400,50700 ;
Layer NDif:  Box  Len 1200  Wid 4200  Center 8400,69300 ;
Layer NGls:  Box  Len 1200  Wid 5400  Center 8400,8100 ;
Layer NMet:  Box  Len 1200  Wid 7800  Center 9600,18300 ;
Layer NImp:  Box  Len 1200  Wid 7800  Center 10200,59100 ;
Layer NPol:  Box  Len 1200  Wid 5400  Center 10800,38700 ;
Layer NCut:  Box  Len 1200  Wid 6600  Center 10800,29100 ;
Layer NBur:  Box  Len 1200  Wid 6600  Center 10800,49500 ;
Layer NImp:  Box  Len 4243  Wid 1200  Center 11700,60900 Direction 1,-1;
Layer NDif:  Box  Len 4800  Wid 1200  Center 12600,72600 ;
Layer NDif:  Box  Len 4800  Wid 1200  Center 12600,66000 ;
Layer NGls:  Box  Len 1200  Wid 5400  Center 10800,9300 ;
Layer NPol:  Box  Len 1697  Wid 1200  Center 11400,36000 Direction 1,-1;
Layer NPol:  Box  Len 1697  Wid 1200  Center 11400,41400 Direction 1,1;
Layer NCut:  Box  Len 1697  Wid 1200  Center 11400,25800 Direction 1,-1;
Layer NBur:  Box  Len 1697  Wid 1200  Center 11400,46200 Direction 1,-1;
Layer NGls:  Box  Len 3394  Wid 1200  Center 12000,6000 Direction 1,-1;
Layer NPol:  Box  Len 3000  Wid 1200  Center 12900,42000 ;
Layer NPol:  Box  Len 3000  Wid 1200  Center 12900,35400 ;
Layer NCut:  Box  Len 3000  Wid 1200  Center 12900,25200 ;
Layer NMet:  Box  Len 1200  Wid 7800  Center 12000,18300 ;
Layer NBur:  Box  Len 3000  Wid 1200  Center 12900,45600 ;
Layer NMet:  Box  Len 4800  Wid 1200  Center 14400,15000 ;
Layer NMet:  Box  Len 4800  Wid 1200  Center 14400,21600 ;
Layer NMet:  Box  Len 3600  Wid 1200  Center 13800,18600 ;
Layer NDif:  Box  Len 1200  Wid 6600  Center 12600,69300 ;
Layer NImp:  Box  Len 4243  Wid 1200  Center 14100,60900 Direction 1,1;
Layer NGls:  Box  Len 3394  Wid 1200  Center 13800,6000 Direction 1,1;
Layer NPol:  Box  Len 1697  Wid 1200  Center 14400,36000 Direction 1,1;
Layer NPol:  Box  Len 1697  Wid 1200  Center 14400,41400 Direction 1,-1;
Layer NCut:  Box  Len 1697  Wid 1200  Center 14400,25800 Direction 1,1;
Layer NBur:  Box  Len 1697  Wid 1200  Center 14400,46200 Direction 1,1;
Layer NPol:  Box  Len 1200  Wid 5400  Center 15000,38700 ;
Layer NCut:  Box  Len 1200  Wid 6600  Center 15000,29100 ;
Layer NBur:  Box  Len 1200  Wid 6600  Center 15000,49500 ;
Layer NGls:  Box  Len 1200  Wid 5400  Center 15000,9300 ;
Layer NImp:  Box  Len 1200  Wid 7800  Center 15600,59100 ;
Layer NDif:  Box  Len 1200  Wid 7800  Center 16800,69300 ;
Layer NPol:  Box  Len 1200  Wid 7800  Center 17400,38700 ;
Layer NCut:  Box  Len 4800  Wid 1200  Center 19200,31800 ;
Layer NBur:  Box  Len 1200  Wid 7800  Center 17400,48900 ;
Layer NDif:  Box  Len 4800  Wid 1200  Center 19200,72600 ;
Layer NDif:  Box  Len 3600  Wid 1200  Center 18600,69000 ;
Layer NGls:  Box  Len 1200  Wid 5400  Center 17400,8100 ;
Layer NPol:  Box  Len 4800  Wid 1200  Center 19800,35400 ;
Layer NBur:  Box  Len 3600  Wid 1200  Center 19200,48600 ;
Layer NBur:  Box  Len 3600  Wid 1200  Center 19200,52200 ;
Layer NImp:  Box  Len 1200  Wid 7800  Center 18000,59100 ;
Layer NGls:  Box  Len 1697  Wid 1200  Center 18000,5400 Direction 1,-1;
Layer NGls:  Box  Len 1697  Wid 1200  Center 18000,10800 Direction 1,1;
Layer NMet:  Box  Len 4800  Wid 1200  Center 20400,21600 ;
Layer NImp:  Box  Len 3600  Wid 1200  Center 19800,62400 ;
Layer NImp:  Box  Len 3600  Wid 1200  Center 19800,58800 ;
Layer NGls:  Box  Len 3000  Wid 1200  Center 19500,4800 ;
Layer NGls:  Box  Len 3000  Wid 1200  Center 19500,11400 ;
Layer NCut:  Box  Len 1200  Wid 7200  Center 19200,28200 ;
Layer NBur:  Box  Len 4243  Wid 1200  Center 20100,47100 Direction 1,-1;
Layer NMet:  Box  Len 1200  Wid 7200  Center 20400,18000 ;
Layer NGls:  Box  Len 2400  Wid 1200  Center 21000,7800 ;
Layer NBur:  Box  Len 1697  Wid 1200  Center 21000,51600 Direction 1,-1;
Layer NBur:  Box  Len 1697  Wid 1200  Center 21000,49200 Direction 1,1;
Layer NGls:  Box  Len 1697  Wid 1200  Center 21000,5400 Direction 1,1;
Layer NGls:  Box  Len 1697  Wid 1200  Center 21000,10800 Direction 1,-1;
Layer NBur:  Box  Len 1200  Wid 2400  Center 21600,50400 ;
Layer NBur:  Box  Len 1200  Wid 1200  Center 21600,45600 ;
Layer NImp:  Box  Len 1697  Wid 1200  Center 21600,59400 Direction 1,1;
Layer NImp:  Box  Len 1697  Wid 1200  Center 21600,61800 Direction 1,-1;
Layer NGls:  Box  Len 1200  Wid 2400  Center 21600,6600 ;
Layer NImp:  Box  Len 1200  Wid 2400  Center 22200,60600 ;
Layer NDif:  Box  Len 7800  Wid 600  Center 30300,69300 ;
Layer NImp:  Box  Len 7800  Wid 1200  Center 30300,59100 ;
Layer NBur:  Box  Len 7800  Wid 1200  Center 30300,48900 ;
Layer NPol:  Box  Len 7800  Wid 600  Center 30300,38700 ;
Layer NCut:  Box  Len 7800  Wid 600  Center 30300,28500 ;
Layer NMet:  Box  Len 7800  Wid 1200  Center 30300,18300 ;
Layer NGls:  Box  Len 7800  Wid 1200  Center 30300,8100 ;
Layer NImp:  Box  Len 1200  Wid 7800  Center 30300,59100 ;
Layer NBur:  Box  Len 1200  Wid 7800  Center 30300,48900 ;
Layer NMet:  Box  Len 1200  Wid 7800  Center 30300,18300 ;
Layer NGls:  Box  Len 1200  Wid 7800  Center 30300,8100 ;
Layer NDif:  Box  Len 600  Wid 7800  Center 30300,69300 ;
Layer NPol:  Box  Len 600  Wid 7800  Center 30300,38700 ;
Layer NCut:  Box  Len 600  Wid 7800  Center 30300,28500 ;
DF;


DS 3;    ( Name: Align );
( 170 Items. );
Layer NDif:  Box  Len 600  Wid 64200  Center 1500,33900 ;
Layer NImp:  Box  Len 600  Wid 64200  Center 1500,33900 ;
Layer NPol:  Box  Len 600  Wid 64200  Center 1500,33900 ;
Layer NCut:  Box  Len 600  Wid 64200  Center 1500,33900 ;
Layer NMet:  Box  Len 600  Wid 64200  Center 1500,33900 ;
Layer NBur:  Box  Len 600  Wid 64200  Center 1500,33900 ;
Layer NImp:  Box  Len 13200  Wid 600  Center 7800,1500 ;
Layer NDif:  Box  Len 13200  Wid 600  Center 7800,1500 ;
Layer NPol:  Box  Len 13200  Wid 600  Center 7800,1500 ;
Layer NCut:  Box  Len 13200  Wid 600  Center 7800,1500 ;
Layer NMet:  Box  Len 13200  Wid 600  Center 7800,1500 ;
Layer NBur:  Box  Len 13200  Wid 600  Center 7800,1500 ;
Layer NDif:  Box  Len 13200  Wid 600  Center 8400,65700 ;
Layer NImp:  Box  Len 13200  Wid 600  Center 8400,65700 ;
Layer NPol:  Box  Len 13200  Wid 600  Center 8400,65700 ;
Layer NCut:  Box  Len 13200  Wid 600  Center 8400,65700 ;
Layer NMet:  Box  Len 13200  Wid 600  Center 8400,65700 ;
Layer NBur:  Box  Len 13200  Wid 600  Center 8400,65700 ;
Layer NMet:  Box  Len 7800  Wid 600  Center 8100,11700 ;
Layer NMet:  Box  Len 7800  Wid 600  Center 8100,4500 ;
Layer NMet:  Box  Len 600  Wid 6600  Center 4500,8100 ;
Layer NCut:  Box  Len 7800  Wid 600  Center 8100,21900 ;
Layer NCut:  Box  Len 7800  Wid 600  Center 8100,14700 ;
Layer NCut:  Box  Len 600  Wid 6600  Center 4500,18300 ;
Layer NPol:  Box  Len 7800  Wid 600  Center 8100,32100 ;
Layer NPol:  Box  Len 7800  Wid 600  Center 8100,24900 ;
Layer NPol:  Box  Len 600  Wid 6600  Center 4500,28500 ;
Layer NDif:  Box  Len 7800  Wid 600  Center 8100,42300 ;
Layer NDif:  Box  Len 7800  Wid 600  Center 8100,35100 ;
Layer NDif:  Box  Len 600  Wid 6600  Center 4500,38700 ;
Layer NDif:  Box  Len 7800  Wid 600  Center 8100,52500 ;
Layer NDif:  Box  Len 7800  Wid 600  Center 8100,45300 ;
Layer NDif:  Box  Len 600   Wid 6600  Center 4500,48900 ;
Layer NDif:  Box  Len 7800  Wid 600   Center 8100,55500 ;
Layer NDif:  Box  Len 7800  Wid 600   Center 8100,62700 ;
Layer NDif:  Box  Len 600   Wid 6600  Center 4500,59100 ;
Layer NGls:  Box  Len 2400  Wid 600   Center 5700,11400 ;
Layer NGls:  Box  Len 2400  Wid 600   Center 5700,4800 ;
Layer NGls:  Box  Len 600   Wid 1800  Center 4800,10200 ;
Layer NGls:  Box  Len 600   Wid 1800  Center 4800,6000 ;
Layer NMet:  Box  Len 2400  Wid 600   Center 5700,21600 ;
Layer NMet:  Box  Len 2400  Wid 600   Center 5700,15000 ;
Layer NMet:  Box  Len 600   Wid 1800  Center 4800,20400 ;
Layer NMet:  Box  Len 600   Wid 1800  Center 4800,16200 ;
Layer NCut:  Box  Len 2400  Wid 600   Center 5700,31800 ;
Layer NCut:  Box  Len 2400  Wid 600   Center 5700,25200 ;
Layer NCut:  Box  Len 600   Wid 1800  Center 4800,30600 ;
Layer NCut:  Box  Len 600   Wid 1800  Center 4800,26400 ;
Layer NPol:  Box  Len 2400  Wid 600   Center 5700,42000 ;
Layer NPol:  Box  Len 2400  Wid 600   Center 5700,35400 ;
Layer NPol:  Box  Len 600   Wid 1800  Center 4800,36600 ;
Layer NPol:  Box  Len 600   Wid 1800  Center 4800,40800 ;
Layer NBur:  Box  Len 2400  Wid 600   Center 5700,52200 ;
Layer NBur:  Box  Len 600   Wid 1800  Center 4800,51000 ;
Layer NBur:  Box  Len 600   Wid 1800  Center 4800,46800 ;
Layer NBur:  Box  Len 2400  Wid 600   Center 5700,45600 ;
Layer NImp:  Box  Len 2400  Wid 600   Center 5700,62400 ;
Layer NImp:  Box  Len 2400  Wid 600   Center 5700,55800 ;
Layer NImp:  Box  Len 600   Wid 1800  Center 4800,61200 ;
Layer NImp:  Box  Len 600   Wid 1800  Center 4800,57000 ;
Layer NGls:  Box  Len 600   Wid 2400  Center 5400,8100 ;
Layer NMet:  Box  Len 600   Wid 2400  Center 5400,18300 ;
Layer NCut:  Box  Len 600   Wid 2400  Center 5400,28500 ;
Layer NPol:  Box  Len 600   Wid 2400  Center 5400,38700 ;
Layer NBur:  Box  Len 600   Wid 2400  Center 5400,48900 ;
Layer NImp:  Box  Len 600   Wid 2400  Center 5400,59100 ;
Layer NGls:  Box  Len 2400  Wid 600   Center 8100,10800 ;
Layer NGls:  Box  Len 2400  Wid 600   Center 8100,5400 ;
Layer NMet:  Box  Len 2400  Wid 600   Center 8100,21000 ;
Layer NMet:  Box  Len 2400  Wid 600   Center 8100,15600 ;
Layer NCut:  Box  Len 2400  Wid 600   Center 8100,31200 ;
Layer NCut:  Box  Len 2400  Wid 600   Center 8100,25800 ;
Layer NPol:  Box  Len 2400  Wid 600   Center 8100,41400 ;
Layer NPol:  Box  Len 2400  Wid 600   Center 8100,36000 ;
Layer NBur:  Box  Len 2400  Wid 600   Center 8100,51600 ;
Layer NBur:  Box  Len 2400  Wid 600   Center 8100,46200 ;
Layer NImp:  Box  Len 2400  Wid 600   Center 8100,61800 ;
Layer NImp:  Box  Len 2400  Wid 600   Center 8100,56400 ;
Layer NImp:  Box  Len 1800  Wid 600   Center 8100,57900 ;
Layer NDif:  Box  Len 1800  Wid 600   Center 8100,57900 ;
Layer NImp:  Box  Len 600   Wid 600   Center 7500,60300 ;
Layer NDif:  Box  Len 600   Wid 600   Center 7500,60300 ;
Layer NBur:  Box  Len 1800  Wid 600   Center 8100,47700 ;
Layer NDif:  Box  Len 1800  Wid 600   Center 8100,47700 ;
Layer NBur:  Box  Len 600   Wid 600   Center 7500,48300 ;
Layer NDif:  Box  Len 600   Wid 1200  Center 7500,48600 ;
Layer NBur:  Box  Len 1800  Wid 600   Center 8100,48900 ;
Layer NBur:  Box  Len 1200  Wid 600   Center 7800,50100 ;
Layer NDif:  Box  Len 1200  Wid 600   Center 7800,50100 ;
Layer NPol:  Box  Len 1200  Wid 600   Center 7800,37500 ;
Layer NDif:  Box  Len 1200  Wid 600   Center 7800,37500 ;
Layer NPol:  Box  Len 1200  Wid 600   Center 7800,38700 ;
Layer NDif:  Box  Len 1200  Wid 600   Center 7800,38700 ;
Layer NPol:  Box  Len 1200  Wid 600   Center 7800,39900 ;
Layer NDif:  Box  Len 1200  Wid 600   Center 7800,39900 ;
Layer NCut:  Box  Len 1200  Wid 600   Center 7800,28500 ;
Layer NPol:  Box  Len 600   Wid 1800  Center 7500,29100 ;
Layer NCut:  Box  Len 600   Wid 1200  Center 7500,29400 ;
Layer NMet:  Box  Len 1800  Wid 600   Center 8100,17100 ;
Layer NCut:  Box  Len 1800  Wid 600   Center 8100,17100 ;
Layer NMet:  Box  Len 1200  Wid 600   Center 7800,18300 ;
Layer NCut:  Box  Len 1800  Wid 600   Center 8100,18300 ;
Layer NMet:  Box  Len 600   Wid 1200  Center 7500,19200 ;
Layer NCut:  Box  Len 600   Wid 1200  Center 7500,19200 ;
Layer NGls:  Box  Len 1200  Wid 600   Center 7800,6900 ;
Layer NMet:  Box  Len 600   Wid 3000  Center 7500,8100 ;
Layer NGls:  Box  Len 600   Wid 2400  Center 7500,8400 ;
Layer NImp:  Box  Len 600   Wid 2400  Center 8100,59400 ;
Layer NDif:  Box  Len 600   Wid 2400  Center 8100,59400 ;
Layer NDif:  Box  Len 600   Wid 600   Center 8100,48900 ;
Layer NPol:  Box  Len 600  Wid 600  Center 8100,28500 ;
Layer NMet:  Box  Len 1200  Wid 600  Center 8400,19500 ;
Layer NCut:  Box  Len 1200  Wid 600  Center 8400,19500 ;
Layer NMet:  Box  Len 600  Wid 600  Center 8100,6900 ;
Layer NGls:  Box  Len 1200  Wid 600  Center 8400,8100 ;
Layer NMet:  Box  Len 600  Wid 600  Center 8100,8100 ;
Layer NGls:  Box  Len 1200  Wid 600  Center 8400,9300 ;
Layer NMet:  Box  Len 1200  Wid 600  Center 8400,9300 ;
Layer NDif:  Box  Len 600  Wid 1800  Center 8700,49500 ;
Layer NBur:  Box  Len 600  Wid 1200  Center 8700,49800 ;
Layer NPol:  Box  Len 600  Wid 3000  Center 8700,38700 ;
Layer NDif:  Box  Len 600  Wid 3000  Center 8700,38700 ;
Layer NCut:  Box  Len 600  Wid 3000  Center 8700,28500 ;
Layer NPol:  Box  Len 600  Wid 3000  Center 8700,28500 ;
Layer NMet:  Box  Len 600  Wid 1200  Center 8700,18000 ;
Layer NCut:  Box  Len 600  Wid 600  Center 8700,17700 ;
Layer NGls:  Box  Len 600  Wid 1200  Center 8700,7200 ;
Layer NMet:  Box  Len 600  Wid 1800  Center 8700,7500 ;
Layer NGls:  Box  Len 2400  Wid 600  Center 10500,11400 ;
Layer NGls:  Box  Len 2400  Wid 600  Center 10500,4800 ;
Layer NMet:  Box  Len 2400  Wid 600  Center 10500,21600 ;
Layer NMet:  Box  Len 2400  Wid 600  Center 10500,15000 ;
Layer NCut:  Box  Len 2400  Wid 600  Center 10500,31800 ;
Layer NCut:  Box  Len 2400  Wid 600  Center 10500,25200 ;
Layer NPol:  Box  Len 2400  Wid 600  Center 10500,42000 ;
Layer NPol:  Box  Len 2400  Wid 600  Center 10500,35400 ;
Layer NBur:  Box  Len 2400  Wid 600  Center 10500,52200 ;
Layer NBur:  Box  Len 2400  Wid 600  Center 10500,45600 ;
Layer NImp:  Box  Len 2400  Wid 600  Center 10500,62400 ;
Layer NImp:  Box  Len 2400  Wid 600  Center 10500,55800 ;
Layer NGls:  Box  Len 600  Wid 2400  Center 10800,8100 ;
Layer NMet:  Box  Len 600  Wid 2400  Center 10800,18300 ;
Layer NCut:  Box  Len 600  Wid 2400  Center 10800,28500 ;
Layer NPol:  Box  Len 600  Wid 2400  Center 10800,38700 ;
Layer NBur:  Box  Len 600  Wid 2400  Center 10800,48900 ;
Layer NImp:  Box  Len 600  Wid 2400  Center 10800,59100 ;
Layer NGls:  Box  Len 600  Wid 1800  Center 11400,10200 ;
Layer NGls:  Box  Len 600  Wid 1800  Center 11400,6000 ;
Layer NMet:  Box  Len 600  Wid 1800  Center 11400,20400 ;
Layer NMet:  Box  Len 600  Wid 1800  Center 11400,16200 ;
Layer NCut:  Box  Len 600  Wid 1800  Center 11400,30600 ;
Layer NCut:  Box  Len 600  Wid 1800  Center 11400,26400 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 11400,40800 ;
Layer NPol:  Box  Len 600  Wid 1800  Center 11400,36600 ;
Layer NBur:  Box  Len 600  Wid 1800  Center 11400,51000 ;
Layer NBur:  Box  Len 600  Wid 1800  Center 11400,46800 ;
Layer NImp:  Box  Len 600  Wid 1800  Center 11400,57000 ;
Layer NImp:  Box  Len 600  Wid 1800  Center 11400,61200 ;
Layer NMet:  Box  Len 600  Wid 6600  Center 11700,8100 ;
Layer NCut:  Box  Len 600  Wid 6600  Center 11700,18300 ;
Layer NPol:  Box  Len 600  Wid 6600  Center 11700,28500 ;
Layer NDif:  Box  Len 600  Wid 6600  Center 11700,38700 ;
Layer NDif:  Box  Len 600  Wid 6600  Center 11700,48900 ;
Layer NDif:  Box  Len 600  Wid 6600  Center 11700,59100 ;
Layer NDif:  Box  Len 600  Wid 64200  Center 14700,33300 ;
Layer NImp:  Box  Len 600  Wid 64200  Center 14700,33300 ;
Layer NPol:  Box  Len 600  Wid 64200  Center 14700,33300 ;
Layer NCut:  Box  Len 600  Wid 64200  Center 14700,33300 ;
Layer NMet:  Box  Len 600  Wid 64200  Center 14700,33300 ;
Layer NBur:  Box  Len 600  Wid 64200  Center 14700,33300 ;
DF;

DS 4;    ( Name: LayerAlign );
( 3 Items. );
Call 3 Trans 0,0;
Call 2 Trans 16200,0;
Call 1 Trans 53400,0;
DF;

End
```

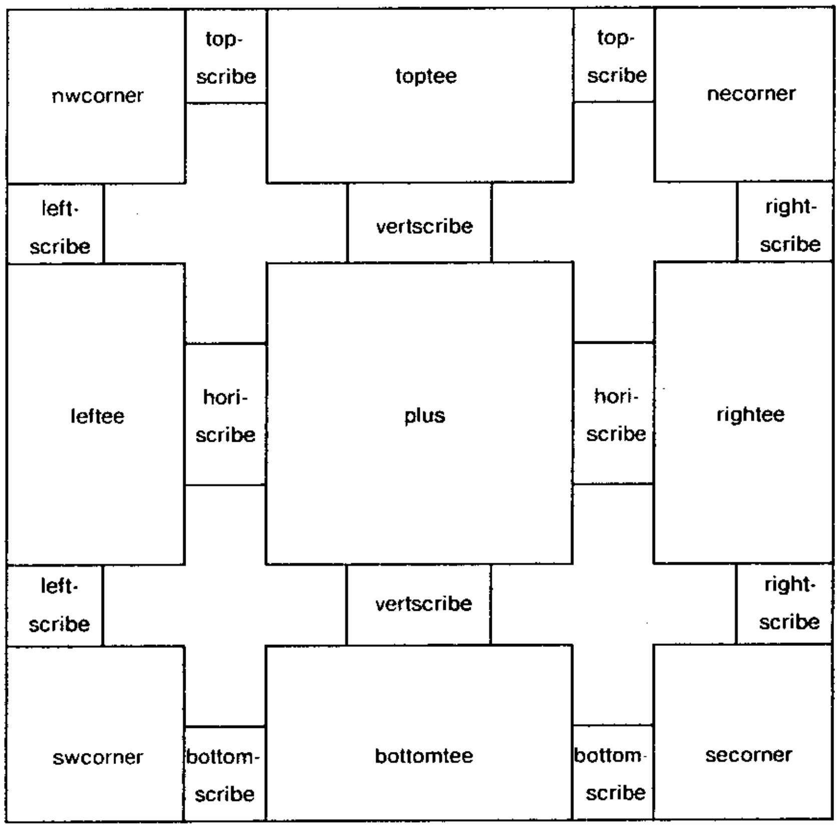
## Documentation for ScribeLines: vertscribe -- horiscribe -- plus rightscribe -- leftscribe -- topscribe -- bottomscribe rightee -- leftee -- toptee -- bottomtee nwcorner -- necorner -- secorner -- swcorner

| | | | |
|---|---|---|---|
| Date | August 29, 1978 | Status | used in summer 1978 MPC |
| Designer | Hon, Johannsen | Address/Phone | PARC SSL |
| Design Rules | Mead/Conway | Scale | λ = 3 μm |
| Info File | ScribeLines.LibDoc | CIF File | ScribeLines.cif |

Dimensions and Replication

| | | | | |
|---|---|---|---|---|
| vertscribe | X: 60λ | Y: 34λ | DX: no | DY: 34λ |
| horiscribe | X: 34λ | Y: 60λ | DX: 34λ | DY: no |
| plus | X: 128λ | Y: 128λ | DX: no | DY: no |
| rightscribe | X: 40λ | Y: 34λ | DX: no | DY: 34λ |
| leftscribe | X: 40λ | Y: 34λ | DX: no | DY: 34λ |
| topscribe | X: 34λ | Y: 40λ | DX: 34λ | DY: no |
| bottomscribe | X: 34λ | Y: 40λ | DX: 34λ | DY: no |
| rightee | X: 74λ | Y: 128λ | DX: no | DY: no |
| leftee | X: 74λ | Y: 128λ | DX: no | DY: no |
| toptee | X: 128λ | Y: 74λ | DX: no | DY: no |
| bottomtee | X: 128λ | Y: 74λ | DX: no | DY: no |
| nwcorner | X: 74λ | Y: 74λ | DX: no | DY: no |
| necorner | X: 74λ | Y: 74λ | DX: no | DY: no |
| secorner | X: 74λ | Y: 74λ | DX: no | DY: no |
| swcorner | X: 74λ | Y: 74λ | DX: no | DY: no |

| | |
|---|---|
| Further Info | See *A Guide to LSI Implementation*, chapter 7. |
| Function/Use | Starting frame scribelines for multi-project chips. See attached figure. |
| Connections | none |



Scribe Lines

file: ScribeLines.cif

```
( Created by Sif from ScribeLines.ic );

DS 1;     ( Name: vertscribe );
( 5 Items. );
Layer NMet;  Box  Len 3000  Wid 10200  Center 1500,-5100 ;
Layer NDif;  Box  Len 16200  Wid 10200  Center 9000,-5100 ;
Layer NCut;  Box  Len 14400  Wid 10200  Center 9000,-5100 ;
Layer NGls;  Box  Len 10200  Wid 10200  Center 9000,-5100 ;
Layer NMet;  Box  Len 3000  Wid 10200  Center 16500,-5100 ;
DF;


DS 2;     ( Name: horiscribe );
( 1 Items. );
Call 1 Rot 0,-1  Trans 10200,0;
DF;


DS 3;     ( Name: rightscribe );
( 1 Items. );
Call 15 Rot 0,-1  Trans 12000,0;
DF;


DS 4;     ( Name: bottomscribe );
( 1 Items. );
Call 15 Rot -1,0  Trans 10200,-12000;
DF;


DS 5;     ( Name: leftscribe );
( 1 Items. );
Call 15 Rot 0,1  Trans 0,-10200;
DF;


DS 6;     ( Name: nwcorner );
( 9 Items. );
Call 5 Trans 0,-12000;
Layer NDif;  Box  Len 11100  Wid 12000  Center 5550,-6000 ;
Layer NGls;  Box  Len 8100  Wid 12000  Center 4050,-6000 ;
Layer NCut;  Box  Len 10200  Wid 12000  Center 5100,-6000 ;
Layer NGls;  Box  Len 4200  Wid 8100  Center 9900,-4050 ;
Layer NMet;  Box  Len 3000  Wid 3000  Center 10500,-10500 ;
Layer NCut;  Box  Len 1800  Wid 10200  Center 11100,-5100 ;
Layer NDif;  Box  Len 1200  Wid 11100  Center 11400,-5550 ;
Call 15 Trans 12000,0;
DF;


DS 7;     ( Name: necorner );
( 1 Items. );
Call 6 Rot 0,-1  Trans 22200,0;
DF;


DS 8;     ( Name: secorner );
( 1 Items. );
Call 6 Rot -1,0  Trans 22200,-22200;
DF;


DS 9;     ( Name: swcorner );
( 1 Items. );
Call 6 Rot 0,1  Trans 0,-22200;
DF;


DS 10;     ( Name: plus );
( 17 Items. );
Call 2 Trans 0,-10200;
Call 1 Trans 10200,0;
Call 1 Trans 10200,-28200;
Layer NMet;  Box  Len 3000  Wid 3000  Center 11700,-11700 ;
Layer NMet;  Box  Len 3000  Wid 3000  Center 11700,-26700 ;
Layer NGls:  Box  Len 18000  Wid 10200  Center 19200,-19200 ;
Layer NDif:  Box  Len 18000  Wid 16200  Center 19200,-19200 ;
Layer NCut:  Box  Len 18000  Wid 14400  Center 19200,-19200 ;
Layer NDif:  Box  Len 16200  Wid 900  Center 19200,-10650 ;
Layer NDif:  Box  Len 16200  Wid 900  Center 19200,-27750 ;
Layer NCut:  Box  Len 14400  Wid 1800  Center 19200,-11100 ;
Layer NCut:  Box  Len 14400  Wid 1800  Center 19200,-27300 ;
Layer NGls:  Box  Len 10200  Wid 3900  Center 19200,-12150 ;
Layer NGls:  Box  Len 10200  Wid 3900  Center 19200,-26250 ;
Layer NMet:  Box  Len 3000  Wid 3000  Center 26700,-11700 ;
Layer NMet:  Box  Len 3000  Wid 3000  Center 26700,-26700 ;
Call 2 Trans 28200,-10200;
DF;


DS 11:    ( Name: toptee );
( 11 Items. );
Call 15 Trans 0,0;
Call 1 Trans 10200,-12000;
Layer NGls:  Box  Len 18000  Wid 8100  Center 19200,-4050 ;
Layer NMet:  Box  Len 3000  Wid 3000  Center 11700,-10500 ;
Layer NDif:  Box  Len 18000  Wid 11100  Center 19200,-5550 ;
Layer NCut:  Box  Len 18000  Wid 10200  Center 19200,-5100 ;
Layer NDif:  Box  Len 16200  Wid 900  Center 19200,-11550 ;
Layer NCut:  Box  Len 14400  Wid 1800  Center 19200,-11100 ;
Layer NGls:  Box  Len 10200  Wid 3900  Center 19200,-10050 ;
Layer NMet:  Box  Len 3000  Wid 3000  Center 26700,-10500 ;
Call 15 Trans 28200,0;
DF;


DS 12:    ( Name: rightee );
( 1 Items. );
Call 11 Rot 0,-1  Trans 22200,0;
DF;


DS 13:    ( Name: bottomtee );
( 1 Items. );
Call 11 Rot -1,0  Trans 38400,-22200;
DF;


DS 14:    ( Name: leftee );
( 1 Items. );
Call 11 Rot 0,1  Trans 0,-38400;
DF;


DS 15:    ( Name: topscribe );
( 4 Items. );
Layer NMet:  Box  Len 10200  Wid 3000  Center 5100,-10500 ;
Layer NDif:  Box  Len 10200  Wid 11100  Center 5100,-5550 ;
Layer NCut:  Box  Len 10200  Wid 10200  Center 5100,-5100 ;
Layer NGls:  Box  Len 10200  Wid 8100  Center 5100,-4050 ;
DF;

Call 14 Trans -32400,0;
Call 6 Trans -32400,32400;
Call 5 Trans -32400,10200;
Call 15 Rot 0,1  Trans -32400,-48600;
Call 6 Rot 0,1  Trans -32400,-70800;
Call 2 Trans -10200,-10200;
Call 15 Mir Y  Trans -10200,-70800;
Call 5 Rot 0,1  Trans -10200,-70800;
Call 15 Trans -10200,32400;
Call 10 Trans 0,0;
Call 11 Trans 0,32400;
Call 13 Trans 0,-48600;
Call 1 Trans 10200,10200;
Call 1 Trans 10200,-38400;
Call 2 Trans 38400,-10200;
Call 5 Rot 0,-1  Trans 48600,32400;
Call 15 Rot -1,0  Trans 48600,-70800;
Call 12 Trans 48600,0;
Call 6 Rot 0,-1  Trans 70800,32400;
Call 6 Rot -1,0  Trans 70800,-70800;
Call 15 Rot 0,-1  Trans 70800,10200;
Call 5 Rot -1,0  Trans 70800,-48600;
End
```

## Appendix F. References

[Auto 1966] "Precision Artwork Language (PAL), Language Specification Manual," Automation Technology, Inc. 1966.

[Ayres 1978] R. Ayres, "ICL Reference Manual." Caltech internal document, SSP File #1364, January 1978.

[Dennard 1974] R. H. Dennard et al, "Design of Ion-Implanted MOSFETS with Very Small Physical Dimensions," *IEEE J. Solid-State Circuits*, SC-9, Oct. 1974, pp.256-269.

[Fairbairn 1978] D. Fairbairn and J. Rowson, "ICARUS: An Interactive Integrated Circuit Layout System", *Proceedings of the 15th Annual Design Automation Conference*, June, 1978.

[Gibson 1976] D. Gibson and S. Nance, "SLIC - Symbolic Layout of Integrated Circuits", *Proceedings of the 13th Annual Design Automation Conference*, June, 1976.

[Glaser 1977] A. B. Glaser and G. E. Subak-Sharpe, *Integrated Circuit Engineering*, Addison-Wesley, 1977.

[Grove 1967] A. S. Grove, *Physics and Technology of Semiconductor Devices*, John Wiley and Sons, 1967.

[Larsen 1978] R. P. Larsen, "Symbolic Layout System Speeds Mask Design for IC's", *Electronics*, Vol 51, No. 15, July 20, 1978.

[Locanthi 1978] B. Locanthi, "A Simula Package for IC Layout", Display File #1862, Computer Science Department, California Institute of Technology.

[McWilliams 1978] T. M. McWilliams and L. C. Widdoes, Jr., "SCALD: Structured Computer-Aided Logic Design", *Proceedings of the 15th Annual Design Automation Conference*, June, 1978.

[Mead 1978] C. A. Mead and L. A. Conway, *Introduction to VLSI Systems*, textbook in preparation.

[Newman 1973] W. M. Newman and R. F. Sproull, *Principles of Interactive Computer Graphics*, McGraw-Hill, 1973.

[Rowson 1978] J. A. Rowson, "A Data Structure for Interactive Integrated Circuit Design", Submitted to the 15th Annual Design Automation Conference, June, 1978.

[Stone 1978] M. Stone, "IC Design Under ICL, Version 1.0." Caltech internal document, SSP File #1336, February 1978.

[Wang 1978] P. P. Wang, "Device Characteristics of Short-Channel and Narrow-Width MOSFETS, *IEEE Trans. on Electron Devices*, ED-25, July 1978, pp.779-786.

[Williams 1977] J. Williams, "Sticks -- A New Approach to LSI Design", Master's Thesis, Massachusetts Institute of Technology, June, 1977.

The following articles are not referenced directly in the main document but are included as background for those who are interested in pursuing individual topics further.

[Bell 1978] A. T. Bell, "An Introduction to Plasma Processing", *Solid State Technology*, April, 1978.

[Beyerlein 1978] F. W. Beyerlein, "New Developments in Automatic Wire Bonding Equipment", *Electronic Packaging and Production*, January, 1978.

[Blais 1977] P. D. Blais, "Edge Acuity and Resolution in Positive Type Photoresist Systems", *Solid State Technology*, August, 1977.

[Bollinger 1977] L. D. Bollinger, "Ion Milling for Semiconductor Production Processes", *Solid State Technology*, November, 1977.

[Cuthbert 1977] J. D. Cuthbert, "Optical Projection Printing", *Solid State Technology*, August, 1977.

[Duchynski 1977] R. J. Duchynski, "Ion Implantation for Semiconductor Devices", *Solid State Technology*, November, 1977.

[Frey 1978] D. W. Frey and E. B. Hryhorenko, "Photoresists for Electronics", *Electronic Packaging and Production*, January, 1978.

[Gupta 1972] A. Gupta and J. W. Lathrop, "Yield Analysis of Large Integrated-Circuit Chips", *IEEE Journal of Solid-State Circuits*, vol. sc-7, no. 5, October, 1972.

[Henriksen 1977] G. M. Henriksen, "Automatic Photo-Composition of Reticles", *Solid State Technology*, August, 1977.

[Hughes 1977] G. P. Hughes, "X-Ray Lithography for IC Processing", *Solid State Technology*, May, 1977.

[Keyes 1977] R. W. Keyes, "Physical Limits in Semiconductor Electronics", *Science*, March, 1977.

[Levinthal 1977] D. J. Levinthal, "Photoresist Trends in Semiconductor Processing", *Electronic Packaging and Production*, November, 1977.

[Markstein 1977] H. W. Markstein, "Trends In Semiconductor Photolithography", *Electronic Packaging and Production*, March, 1977.

[Markstein 1978] H. W. Markstein, "Wafer Sawing Update", *Electronic Packaging and Production*, March, 1978.

[Marshall 1974] J. F. Marshall, "Scribing and Breaking of Semiconductor Wafers", *Solid State Technology*, September, 1974.

[Meindl 1977] J. D. Meindl, "Microelectronic Circuit Elements", *Scientific American*, Sept., 1977.

[Oldham 1977] W. G. Oldham, "The Fabrication of Microelectronic Circuits", *Scientific American*, Sept., 1977.

[O'Malley 1975] A. J. O'Malley, "Technological Implications in the Photomasking Process", *Solid State Technology*, June, 1975.

[Rajchman 1977] J. A. Rajchman, "New Memory Technologies", *Science*, March, 1977.

[Roussel 1978] J. Roussel, "Step-and-Repeat Wafer Imaging", *Solid State Technology*, May, 1978.

[Stapper 1976] C. H. Stapper, "LSI Yield Modeling and Process Monitoring", *IBM Journal of Research and Development*, May, 1976.