DESIGN INTEGRITY AND IMMUNITY CHECKING:
A New Look at Layout Verification and Design Rule Checking

Edward J. McGrath
Computer Science Department, California Institute of Technology, Pasadena, California
and
LSI Advanced Development, Digital Equipment Corporation
Hudson, Massachusetts

Telle Whitney
Computer Science Department, California Institute of Technology, Pasadena, California

ABSTRACT

A program implementing a novel approach to layout verification is presented. The approach uses topological and device information to eliminate most false and unchecked errors. This technique, coupled with a hierarchical front end to eliminated redundant checks, is appropriate for layout verification of VLSI designs. Design rules appropriate for this technique, some usage rules in the context of structured design, and a discussion of the future of design rule checking are also presented.

INTRODUCTION

The state of the industry today in Design Rule Checking and Design Rules has been arrived at in an evolutionary manner. The design rules we use today are not much different, except for the numbers, than the ones used back in the days of cutting rubylith. Design rule checking programs serve as a valuable aid to designers in checking their layouts, yet they must still ultimately rely on visual checks for the final verification. Design methodology on the other hand, has undergone drastic changes, through mylar drawing and digitizing, standard cells, gate arrays, as well as the most recent work in chip assembly/silicon compilation. [1,2] This paper attempts to reexamine the role of DRC and verification in light of the changing state of VLSI design and discusses the implementation of some of the results in a program currently under development at Caltech.

There are basically three driving forces behind the work in this area. The first is to develop a methodology to manage the complexity of designs. The second is to reduce the number of both false and unchecked errors. The third is to examine the potential role of layout verification as a driving force in the development of the principles of structured design and further, in steering the directions of design tools being developed to support that design style.

This paper will discuss some of the techniques used to eliminate false and unchecked errors. The role of the design style in verification and the role of DRC in some future design tools supporting the structured design style will be discussed. The techniques for managing the complexity of VLSI designs and reducing checking time will be discussed in a future paper.

WHY DESIGN RULES?

Integrated circuit processes are limited in their ability to fabricate devices. The limiting processes are many and complex. Photolithographic technology limits our ability to resolve small lines and spaces. Mechanical and thermal effects limit our ability to align masks to features on the chip. Chemical processes and our ability to control them limit the accuracy to which we can etch away unwanted portions of the chip. These processes limit the feature sizes which the process can reliably produce.

The integrated circuit designer must design circuits within the process limitations if a working and high yielding chip is desired. A means is required to inform the circuit designer of those limitations. The rules must also communicate the process limitations to those responsible for developing layout verification and layout design tools. Design rules must become increasingly more specific to reflect the changes in expertise of the people using the rules.

SOME DIFFICULT PROBLEMS IN DRC TODAY

One of the most difficult problems in DRC today is that of run time. Handling the complexity of VLSI designs in a layout checker, maintaining run time at an acceptable level is one of the prime reasons for the efforts in this area. An equally important problem however, is the reduction of false and unchecked errors. Most design rule checkers today suffer from a problem which can be illustrated through figure 1.

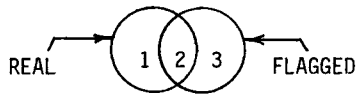

FIGURE 1: Design Rule Errors

Region 2 in this figure represents the successful identification of real layout errors by the design rule checker. Region 1 represents real layout errors which are not identified by the DRC. Region 3 represents the false errors, those places where the DRC has said there were errors when there were not. We believe that the elimination of these false and unchecked errors to be as serious a problem as the run time problem. Experience has shown that the ratio of false to real errors can be 10 to 1 or higher. One result of this is lack of trust in the program, designers may spend large amounts of time doing visual checks or worse, dismiss large numbers of errors as false without thoroughly checking them. One common solution is to remove those checks that are prone to generating false errors. This is not acceptable since it leaves part of the circuit unchecked.

The problem of eliminating unchecked errors is equally difficult and important. The visual checks required on a 100K device chip which has been checked by an 80% effective DRC are as onerous as those required to visually check a 20K device chip with no DRC. The following section discusses some of the most common and difficult problems encountered and some of the reasons for their occurrence.

Geometrical

The first class of problems is what we call geometrical. Some of these problems have been discussed before [3,4,6] but we will review them quickly for completeness. Many proposed programs are "figure based." That is, they operate on an elemental figure which is a simple closed loop of line segments. Some pathologies which can be introduced by this technique are shown below.

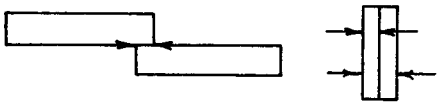

Figure 2: Figure Pathologies

In one case the figures themselves are legal but the composite obtained by unioning them is not. In the other case the figures are too narrow but the composite is legal. One solution to this is to deal with edges rather than figures 1 while another more common solution is to union all figures before performing any checks. The disadvantage to this is that algorithms for general polygon checks can be quite expensive [5] while those for boxes and wires are almost trivial.

Another area of contention is whether Euclidean or Orthogonal expand and shrink is appropriate. 6 As shown below both Euclidean and Orthogonal shrink yield square corners when applied to simple squares, but yield quite different results when expanding. The Orthogonal expand preserves the square corners while the Euclidean expand rounds the corners.

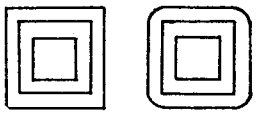

Figure 3: Orthogonal vs Euclidean

Both techniques result in pathologies when used to check geometrical design rules. As shown below, the shrink-expand-compare [7] technique for checking width yields errors at every corner when the Euclidean technique is used, while the expand-check overlap technique for checking spacing yields errors when checking corner to edge spacing.

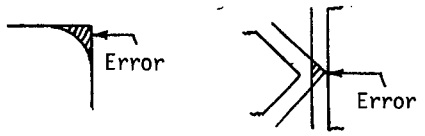

Figure 4: Width & Spacing Pathologies

Neither technique however, accurately reflects the results of processing variations. Some work in this area will be discussed later.

Topological

The next class of difficulties falls into the classification of topological. That is, they have to do with matters of circuit connectivity or components. One problem which is very rarely addressed is that of electrical equivalence. Certainly in figure 5a below, checking spacing between the indicated boxes is unnecessary since they are electrically equivalent. However, if as in figure 5b, the figure in question is a resistor, a spacing check is in order since a short at this point could be critical to the circuit performance.

Figure 5: Topological Pathologies

Almost all DRC and design rules are based on mask levels, although it is possible to make different kinds of devices on the same mask layer. Sometimes the rule is different depending on which kind of device being checked. For example, figure 6a shows the base region of a bipolar transistor shorted to the isolation region around it. This is an error since it destroys the integrity of the device. In figure 6b, a resistor made of the same base diffusion is being connected to the isolation diffusion. This is a common technique to tie one end of a resistor to ground and is quite legal.

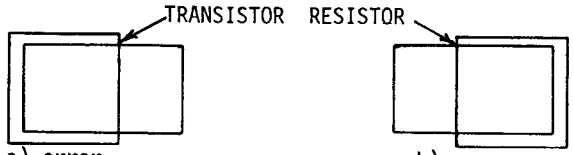

Figure 6: Device Dependent Rules

A similar rule says that a contact is not allowed over the active gate of a silicon gate MOS transistor. This is shown below, in figure 7 where the active gate region is the intersection of the poly and diffusion regions. On the same process a direct contact from poly to diffusion is made by overlapping poly and diffusion, covering the overlap region with a contact and then covering the contact with metal. This would be called out as an error if active gate were defined to be the intersection of poly and diffusion.

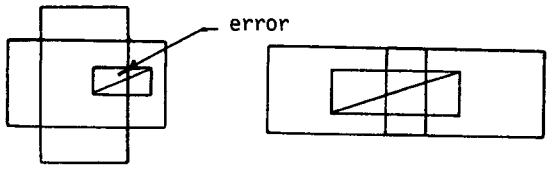

Figure 7: Transistor & Butting Contact

Most design rule checkers today will not recognize the accidental crossing of poly and diffusion as an error since it forms a legal transistor. (see figure below) The difference lies between checking that a layout meets the design rules and checking that the layout implements the circuit intended as well as meeting the design rules.

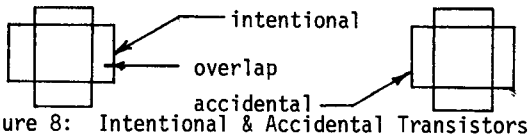

Figure 8: Intentional & Accidental Transistors

Many checkers check the gate overlap on an MOS transistor by isolating it and measuring its width. (see above) If the overlap does not exist it is often not caught as an error.

Often there are devices or special cases that are exceptions to the rules; a device where the rules are intentionally broken or where the rule is so complex as to defy simple checking. (eg. Distance from buried contact to point on N+ where N+ spacing reduces from either 7 microns or 6 microns to 5 microns assuming buried contact to N+ spacing is less than 5 microns.) Most checkers will either ignore this check or flag it everywhere. A technique for flagging specific devices as checked to eliminate large numbers of false errors would be useful.

Non Geometric Design Rules

Finally, there exists a class of composition rules which are design rules in a broad sense but are not geometrical. These rules require symbolic and topological information as well as geometrical information to check them. Some examples of this are:

1.) A net must have at least two "devices" on it.

2.) Power and ground must not be shorted.

3.) A "bus" may not connect to power or ground.

4.) A depletion device may not connect to ground.

Net list generation and non-geometric design rule verification have a lot in common with DRC and should appropriately be handled by a single program.

In summary, we see the problems in DRC today falling into two classes: Geometrical and Topological. We also see layout verification expanding beyond traditional DRC into interconnect verification and verification of other non-geometric construction rules. (Hence the title: Design Integrity and Immunity Checking)

SOME TECHNIQUES FOR SOLVING THESE PROBLEMS

We believe that the key to solving many of these problems as well as the key to managing complexity lies in the internal data structure used by the program. Traditional checkers deal with mask geometry, that is, the geometrical form of the data just before pattern generation, in its fully instantiated form. Any topological or device information about the circuit is discarded. Where the design rule checker needs topological information it must be reconstructed Devices must be recognized before rules regarding them can be checked.

The chips we are checking are not just thousands of pieces of unrelated geometry; they are circuits. At the design rule checking phase in the design cycle much more information is available about the circuit than just the location and layer number of polygons. The data format we choose to use is an extension of CIF (Caltech Intermediate Form). [8] This data form allows symbol definitions, calls (nesting not allowed) to symbol definitions, and primitive geometrical constructs (box, wire, polygons, etc.). The extention to CIF we use allows a net identifier to be attached each primitive element and a device "type" identifier to each primitive symbol (transistor, contact, etc.). Using this format the data representing a chip has a hierarchical structure.

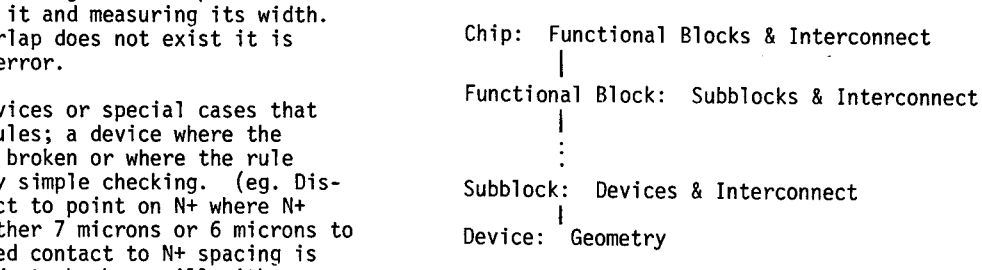

Figure 9: Chip Structure

At the topmost level, there are functional blocks and interconnect, below that, subblocks and more interconnect, and finally, there is a level where there are only primitive symbols and interconnect. Primitive symbols contain only geometry and are the only way a "device" can be defined. Implicit devices, for example poly crossing diffusion to form a transistor, must occur within a primitive symbol definition.

The key difference between the approach described here and that of most other design rule checkers is that the chip is not treated purely as a collection of geometry; the chip is never fully instantiated; the information about what symbol the piece of geometry came from is never lost. This is the key to exploiting the hierarchy in the design and to checking rules which need device or topological information. A flow chart of the technique is shown below:

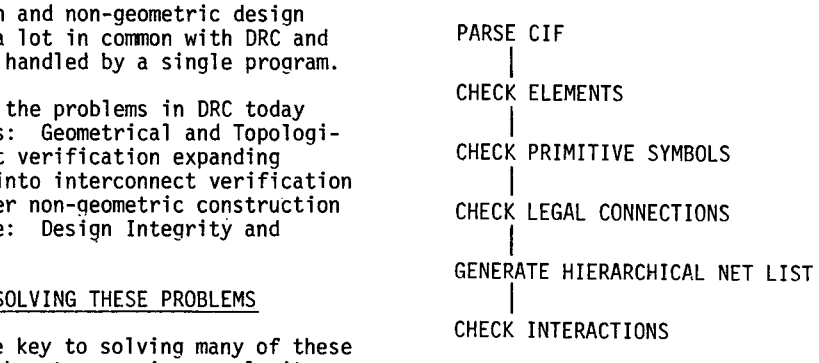

Figure 10: DRC Flowchart

In the box labelled "check elements", the primitive elements of the chip are checked for legal width. This is done in the symbol definition, not in each instance of a symbol. Boxes and wires are trivial to check, polygons require a more general purpose polygon width routine. The only elements which are checked at this stage are interconnect.

Any element which is part of a primitive symbol is treated in the box labelled "check primitive symbols". These checks are the most complicated checks required. These may include enclosure rules, overlap rules, even overlap of overlap rules (buried contact). The process physics involved here is by far the most complex. On the other hand there are not very many different elemental symbols on a given chip (20 to 30). Checking these symbols requires handling small amounts of data but performing quite complex checks on it. This is quite amenable to visual checking and for this reason implementation of this part of the checker is given low priority. Primitive symbols are assumed to be prechecked. In many layout systems (eg. symbolic or device library based systems) this is true already. [9]

In the next box the chip is searched in a hierarchical manner for possible interactions between elements. In doing this, elements which interact and are on the same layer are checked against the connection rules for legal connections. The legal connection criterion used here is that of skeletal connectivity. Two elements are connected if their skeletons touch, overlap, or if one is enclosed within the other. The skeleton of an element is the result of shrinking that element by ½ the minimum width on that layer.

Figure 11: Skeletal Connectivity

The elements on the left side of this figure are skeletally connected while those on the right are not. Note that if two elements are each of legal width and are skeletally connected, then the union of the elements is of legal width. This eliminates using complicated polygon routines to check simple connected elements.

While parsing the design, each element in the design is assigned a unique net identifier using a dot notation to reference elements in an instance from a higher level in the hierarchy (eg. a.b refers to element b in the instance a). With this hierarchical net list available, it is now possible to check electrical construction rules or to check the net list against an input net list for consistency.

The final box on the flow chart is labelled "Check Interactions". At this point all elements are checked, all primitive symbols are checked, connections between the elements and symbols are checked, and net identifiers are available for each element. What remains to be checked are the interactions between elements and/or primitive symbols. The checks which remain are only spacing checks. The possible cases can be enumerated as the elements of an upper triangular matrix as shown below for a Si gate MOS process.

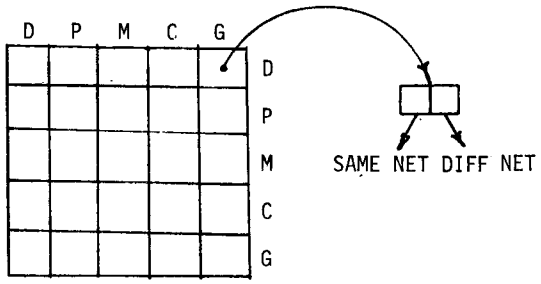

Figure 12: Interaction Rules

Each of these cases can be broken into two subcases, figure 12 , depending on whether or not the elements are on the same net. If the element is part of a transistor, the subcases depend on whether or not the elements are related. (This is because the gate or implant of a transistor cannot be assigned to a net.) Most of these cases are not necessary; either there is no rule between those two mask layers (as in metal and diffusion) or the only rules are relating to primitive symbols which are checked already (as in contact and poly). Usually one of the subcases in each case need not be checked (spacing between elements on the same net is usually not necessary).

2-D Process Modelling for DRC

A technique for modelling the 2-D process variations, which is based on the physics of the process is being investigated for this use. The technique commonly used to check spacing is to expand (either orthogonal or Euclidean) by half the minimum spacing and check for overlap. This technique does not accurately reflect the physics of the process it intends to model. Strictly speaking, there are two different cases to consider. First, and most simple, is the case when the elements are on the same mask layer. The intent of the spacing rule is to insure that under worst case processing the elements do not short together. The effects which might cause this are all "bias effects", that is they are effects which tend to make the geometry on the silicon different in size and shape than the silicon as drawn. The second case is when the geometries are not on the same layer. The intent of the rule here is to insure that the geometries here do not overlap under worst case processing. The worst case processing. The worst case processing in this case consists of both bias effects and mask misalignment.

Modelling of bias and misalignment effects should in general be different. Misalignment can be modelled by a simple translation while bias effects are more complex. Bias effects in fact are not unary. That is, a piece of geometry expands or shrinks differently if there is another piece nearby. This is the so called proximity effect. In the figure below the effect of Euclidean, orthogonal, and proximity effect expand are shown.

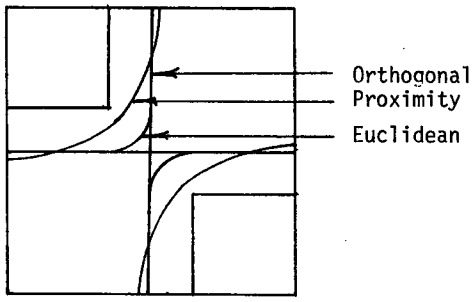

Figure 13: Euclidean, Orthogonal & Proximity Expand

The proximity effect expand shown was calculated by doing a 2-D convolution of a gaussian exposure function representing the exposure and etching variation with a simple binary mask function representing the intended geometry and then clipping to model the photoresist sensitivity and etch time.

$$I(\vec{r})=\iint_{xy} f(\vec{r})M(\vec{r})dxdy=\iint_{xy}Ae^{-r^2/2\sigma^2}M(\vec{r})dxdy \quad (1)$$

The calculation of this exposure function in its general form is quite time consuming. This calculation is quite similar to the proximity effect calculations done to compensate for the gaussian beam profile in E-beam exposure systems. [10,11] Fortunately, some simplifications can be made to speed up the calculations. If the mask function can be simplified to simple boxes or other elemental geometries, then equation (1) for the exposure at each point in the region in question has a closed form solution in terms of an error function.

$$I(\vec{r})=\left[\Phi(\frac{\Delta x1}{\sigma})-\Phi(\frac{\Delta x2}{\sigma})\right]\cdot\left[\Phi(\frac{\Delta y1}{\sigma})-\Phi(\frac{\Delta y2}{\sigma})\right] \quad (2)$$

Calculation of the exposure function at each point is unnecessary since we are only interested in whether the exposure exceeds some critical value at its maximum.

Spacing calculation by this technique now reduces to finding "the line of closest approach"; translating one element along this line (if they are on different layers), finding the maximum of the exposure function (which will lie along this line), and comparing the value at this point against some critical value. This technique, although still slower than the expand-check overlap technique, is more correct and may be feasable to use for design rule checks.

This technique has the advantage that it correctly models relational rules. [6] Relational rules are ones where one dimension of the structure depends on another feature of the same structure. For example, the poly overlap of the gate region on an MOS transistor is a function of the width of the poly in some design rules to account for the "retreat" of the end on narrow wires. (see figure below) The fast way to check this rule using this technique is to translate in the direction to make the overlap smaller, calculate the exposure function for the poly and for the diffusion along the line shown, clip as before, and check if the poly has retreated beyond the diffusion.

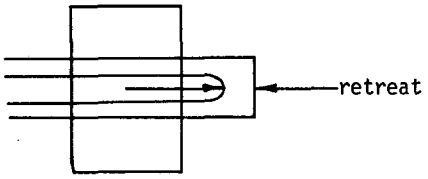

Figure 14: Relational Rule

DESIGN RULES

Design rules serve primarily as the link between design and processing. As such they should reflect the physics limiting the process as well as describe the circuit in terms that are meaningful to the design community. Design rules which are based on mask levels do neither of these very well. The large number of exceptions to and variations of the rules are a testimony to this fact. A more appropriate set are described below. The rules to be described can be broken into four categories:

1.) Legal "devices" and related rules.

2.) Legal interconnect; width and connection rules.

3.) Interaction rules;between devices and interconnect.

4.) Non-Geometric construction rules.

Design rules should describe all potential 2-D failure modes and a layout technique for avoiding the failure. For example, the overlap of poly beyond the active gate of a transistor is to insure that the source and the drain never short together. In addition, each rule should be broken down into its components so that each component can be modelled properly.

Design rules specified in this manner reflect the processing better, make more sense to the designer, and are more appropriate to the approach to DRC described above.

STRUCTURED DESIGN

The principles of structured programming evolved from several considerations. Some of these were ease of understandability, debugging program verification, etc. Layout verification and DRC play the same role in design as program verification did in programming. It is not surprising therefore, to see some structured design principles or good usage rules coming out of the work in layout verification. The concept of structured design [12, 13] is quite new, and many of the ideas are still being developed. Some ideas which have come out of the work in DRC are described below.

Two of the key concepts in structured programming have direct analogs in structured design. These are declarations and typing. In programming languages one is often required to declare all variables explicitly along with their type. Analogously, we are requiring that all "devices" or elemental symbols be called out specifically and their type defined. Implied devices are not allowed. The crossing of poly and diffusion outside of the context of a transistor symbol is an error. This has many benefits. This allows the designer to use special or unusual devices and to mark them checked. It allows the DRC to break the design into devices and interconnect easily before applying the rules. It replaces the need for device recognition with that for device checking. Finally, it greatly simplifies the checking for unintentional devices.

A rule of good usage being enforced by this program is that each symbol or element should be self sufficient at every level of the hierarchy. Butting of two boxes each of half minimum width to form a legal box is called out as an error. This is a common technique used when butting identical symbols as shown below.

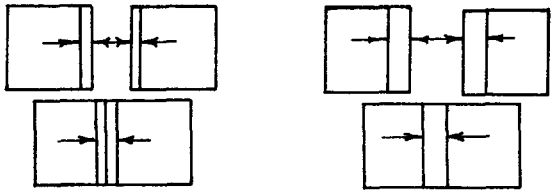

Figure 15: Self Sufficiency

The preferred technique is to include a legal width box in each symbol and to overlap the symbols as shown on the right. Hierarchical checking is nearly impossible without this restriction.

The last rule of good usage is also one from programming technique. That is the principle of locality. It is preferable in almost all cases to use local elements rather than global. This has two benefits; it simplifies the global checking problem and causes a small change to have a small effect.

THE FUTURE OF DESIGN RULE CHECKING

Many times the question has been raised of the future of DRC and layout verification in light of the work on "correctness by construction." Limited success has been achieved in doing without checking on gate array or master slice circuits, but there is clearly a need in the short term in the area of true custom designs for an intelligent hierarchical layout checker. The hierarchical part is especially important for structured VLSI designs where there is a large amount of data but a large degree of regularity in the design.

In the long term the role of DRC is likely to change but not to disappear in the foreseeable future. Several systems exist and/or are in development which may in fact eliminate the need for cell level DRC. [14,15,16] We believe the concepts proposed here to be relevant to the design of those systems. The specification of design rules in terms of devices, interconnect, and interactions is most appropriate for these systems that deal in terms of devices and interconnect either symbolically or abstractly. It is not clear whether the techniques that guarantee correct layouts at the cell level are appropriate at the chip level. These systems, for the most part, are capable of designs of MSI complexity only. Until these systems are capable of handling designs of VLSI there will be a need for chip level DRC, and other layout verification.

There is another class of design systems which are intended to compile or assemble designs of VLSI complexity. [1,2] These systems have the potential of assembling the layout of a large chip in a design rule correct way. Once debugged, these systems may be capable of laying out many images of the same type of chip (eg. a data path) without error. However, chips of a different type will require a significant amount of new code. As much as 30% of the code in these programs is chip dependent and must be changed to accommodate a new floor plan and new circuits. [ref 17] Layout verification clearly still has a role in verifying this new code. It appears unlikely that programs capable of laying out any type of custom chip without errors will appear in the near future. DRC clearly has a role in verification for several years to come.

ACKNOWLEDGEMENT

We would like to express our gratitude to the participants in the Silicon Structures Project at Caltech for all the support and encouragement they gave during this work. Special thanks go out to Ivan Sutherland, Don Oestreicher, Jim Kajiya, Ricky Mosteller and Eric Barton for specific suggestions and help during the course of this work. Credit is due to Don Oestreicher for the exposure function calculations.

REFERENCES

1.) D. Johannsen, "Bristle Blocks: A Silicon Compiler Silicon Structures Project Display File #2587, California Institute of Technology, January 1979.

2.) G. Tarolli, Digital Equipment Corporation, private communication, February 1980.

3.) H.S. Baird, "Fast Algorithms for LSI Artwork Analysis", Proc. of 14th D.A. Conf., 303-311, June 1977.

4.) C.R. McCaw, "Unified Shapes Checker- A Checking Tool for LSI," Proc. of 16th D.A. Conf., 81-87, June 1979.

5.) H.S. Baird, "A Survey of Computer Aids for IC Mask Artwork Verification", Proc. IEEE Int. Symp. on Circ. and Sys., 441-445, April 1977.

6.) P. Losleben, K. Thompson, "Topological Analysis for VLSI Circuits", Proc. of 16th D.A. Conf., 461-473, June 1979.

7.) B.W. Lindsay, B.T. Preas, "Design Rule Checking and Analysis of IC Mask Designs", Proc. of 13th D.A. Conf., 301-308, June 1976.

8.) R. Sproull, R. Lyon, S. Trimberger, "The Caltech Intermediate Form for LSI Layout Description", Silicon Structures Project Display File #2686, California Institute of Technology, April 1979.

9.) D. Gibson, S. Nance, "SLIC-Symbolic Layout of Integrated Circuits", Proc. of 13th D.A. Conf., 434-440, June 1976.

10.) M. Parikh, "Corrections to Proximity Effects in Electron Beam Lithography I. Theory", J. Appl. Phys., 50(6), 4371-4377, June 1979.

11.) N. Sugiyama, K. Saitoh, K. Shimuzu, "Proximity Effect Correction in EBeam Lithography for VLSI Microfabrication", Trans. of ISSCC, 88-89, February 1979.

12.) C. Mead, L. Conway, "Introduction to VLSI Systems", Addison-Wesley, Reading, Massachusetts, 1980

13.) J.P. Gray, "Structured Design Notes," Silicon Structures Project Display File #3354, California Institute of Technology, December 1979.

14.) J. Williams, "STICKS-A Graphical Compiler for High Level LSI Design", Proc. of 1978 NCC, 289-295, May 1978.

15.) M. Hseuh, "Symbolic Layout and Compaction of Integrated Circuits ", Electronics Research Laboratory Memo #UCB/M79/80, University of California Berkeley, December 1979.

16.) A. Dunlop, "SLIP-Symbolic Layout of Integrated Circuits with Compaction", CAD, 10(6), 387-391, Nov. 1978.

17.) D. Johannsen, California Institute of Technology private communication, October 1979.