THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 51, No. 7, September, 1972
Printed in U.S.A.

XYTOLR–A Computer Program for Integrated Circuit Mask Design Checkout

By MICHAEL YAMIN

(Manuscript received March 7, 1972)

XYTOLR is a computer program which checks integrated circuit mask designs for compliance with design rules such as those which require minimum clearances between diffused and metallized areas. The input is the same machine readable mask description that is used for automatic mask art work generation. Clearance violations, if present, are reported in graphic and printed form with pointers to the erroneous statements in the user's input.

I. INTRODUCTION

The increasing size and complexity of large-scale integrated circuit mask designs has created a demand for computer aids to the circuit designer. Without such aids, the design of many circuits would be not merely difficult, but practically impossible. Perhaps the best established design aids in this field are computer art work generation systems, which accept as input numerical descriptions of mask layouts, and drive high-precision graphical output devices which create the mask masters. XYMASK,[1,2] used at Bell Laboratories, is such a system.

The use of a computer art work generation system implies that there exists a complete and unambiguous machine-readable description of the masks required to produce an integrated circuit. In principle, this information, together with the physical properties of the substrate and diffused layers, is sufficient to predict any desired property of the completed circuit. The availability of computer programs to predict and analyze important properties of circuit designs in advance of their fabrication would be a major aid to the circuit designer.

One early phase of the design process for which a computer aid is highly desirable is that of checking a mask set, before it enters production, for compliance with design rules such as required clearances between areas delineated by the same or different masks. Deviations

from these rules may result in loss of manufacturing yield or early device failure. Visual checking of mask art work for such tolerances is not only tedious and time-consuming, but has frequently been found to miss one or two significant errors in the mask design. The exhaustive patience of the computer is better suited for such important but mechanical verification procedures. This paper describes a computer program called XYTOLR, which tests mask designs automatically for tolerance errors. The data structures and other components of XYTOLR can be used to construct other mask analysis programs.

II. USER INTERFACES—INPUT AND OUTPUT

Computer programs, in general, should require the minimum necessary input from the user, and should accept it in a form which is simple and natural for the user to write. The output should include all the information which the user might desire, but indexed and organized so that specific items can be located without extensive searching. An attempt has been made to follow these principles in the design of XYTOLR.

2.1 *Mask Design Input*

The primary input to the tolerance-checking program is the geometrical description of the mask set in XYMASK[1,2] input language. This is the same input deck (except for a few command statements) that is used for mask art work generation via XYMASK. Since such decks can run to several thousand cards, it is unreasonable to ask the user to make any general alterations for the benefit of XYTOLR. To preserve language compatibility, a version of the XYMASK language processor is incorporated in XYTOLR to read this input.

In the XYMASK language, the mask designer defines closed plane figures composed of line and circular arc segments. Each such defined figure is called an "atomic," and its definition includes a symbolic name (mask level name) which assigns the atomic to a specific mask of the set being designed. An atomic may also be given a label (atomic name). By the use of statements referring to this label, an atomic may be translated, rotated, reflected, and repeated many times at different locations in the mask. Groups of atomics may be combined into compound structures called "clumps," which may also be labelled and manipulated like atomics. A "clump" may include atomics assigned to different mask levels. In this way, devices such as transistors which require shapes on several masks can be designed and manipulated as units.

While XYMASK permits atomics to be defined without labels (atomic names), XYTOLR requires a name for each atomic. A name is therefore generated for any atomic not named by the user. Such names are of the form ATnnnnn, where nnnnn is a number pointing to the location (line number) of the atomic definition statement in the XYMASK deck listing. The user thus can find the statement to which the generated name refers.

The structure described by a XYMASK deck consists essentially of many geometrical figures distributed among a number of mask levels. Each figure is called an "occurrence." XYTOLR assigns a serial number to each occurrence. A single atomic definition may result in many occurrences, as a result of repetitions.

2.2 *Tolerance Test Input*

XYTOLR, like XYMASK, is technology-independent, dealing only with the geometry of the mask design, and knowing nothing about design rules or the device functions of the various masks. The user must provide input which describes the "tolerance tests" he wishes the program to conduct. A "tolerance test" asks for a report of all instances in which the clearance between elements on specified mask levels is less than a specified quantity. The operation involved in a tolerance test is as follows: enlarge each occurrence on a designated mask level by some quantity and report all new connections which develop, and old connections which are broken, between these occurrences and the occurrences on another designated mask level. The enlargement quantity is generally a little less than the specified minimum clearance, so that clearances just at the minimum are not reported as errors. If checking of clearances within a mask level is desired, the operation is: enlarge each occurrence by a little less than half the desired clearance and report all new connections within the mask level.

A simple, free-form language is used to specify tolerance tests. For example, suppose one wishes to assure that every emitter in a bipolar integrated circuit lies no closer than 0.5 units to the boundary of the base diffusion which surrounds it. (The units are the same as those used in the XYMASK design deck.) The statement

EMITDIF +.5 BASEDIF

results in every emitter being enlarged by a little less than 0.5 units, and any new contact with the base diffusion boundaries being reported.

Suppose one wishes to check that every occurrence on a metallization mask clears every other occurrence by 0.3 units. The statement

/ THINMET +.15

results in every occurrence on the THINMET level being expanded by a little less than 0.15 units, and any new connections within that level being reported.

A negative number results in contraction of occurrences, and may be used to test for minimum overlaps.

The tolerance test language has the capability of describing quite complicated test patterns within a statement. It includes control words which enable the user to modify the normal handling of a tolerance test by XYTOLR: for instance, expand by exactly the quantity requested rather than a little less. Any number of such statements may be submitted, and each will induce a tolerance test and generate a report.

In practice, a mask designer working on a particular type of circuit will have developed a standard test protocol to be applied to all of his designs. Thus, specifying a tolerance test sequence will consist simply of selecting the appropriate tolerance test input deck for the circuit type being tested.

2.3 *Masklevel Connectivity Input*

The user may know that tolerance tests between certain mask levels will never be made, either because these mask levels can never be in physical contact (for example, a metallization and a buried collector in bipolar technology) or because there is no current interest in such a test. XYTOLR can accept input that tells it which mask levels are to be considered capable of connection, and which are not. It is never necessary, in principle, to supply such input, but, for large mask designs, considerable savings in running time and memory occupancy can result if it is supplied in advance of all tolerance test input.

2.4 *Output*

Both printed and graphical reports are generated by XYTOLR. For each tolerance test, a plot is produced of all occurrences involved in violations of the tolerance rule. Each plotted occurrence is identified with its serial number. Comparison of this plot with drawings of the individual masks (provided on the same scale) facilitates locating the errors in the mask design.

The drawing of tolerance errors is accompanied by printed output identifying each pair of occurrences involved in a violation. The occurrences are identified by atomic name, mask level name, occurrence

serial number, and coordinates. This information makes it possible to find the offending statements in the XYMASK input deck.

Self-intersecting shapes, which are not permitted by some precision plotters although they are accepted by XYMASK, are also reported.

The program has been designed to err on the side of strictness in reporting tolerance violations. Marginal cases are reported as violations, leaving it up to the user to decide whether he considers them acceptable. If no violations are found, that fact is reported explicitly.

Figures 1 and 2 are examples of XYTOLR output. Figure 1 shows a simple metallization pattern. The clearance desired between metallized areas was 0.3 units. A number of occurrences in violation of this rule were deliberately inserted into the XYMASK description of the mask.

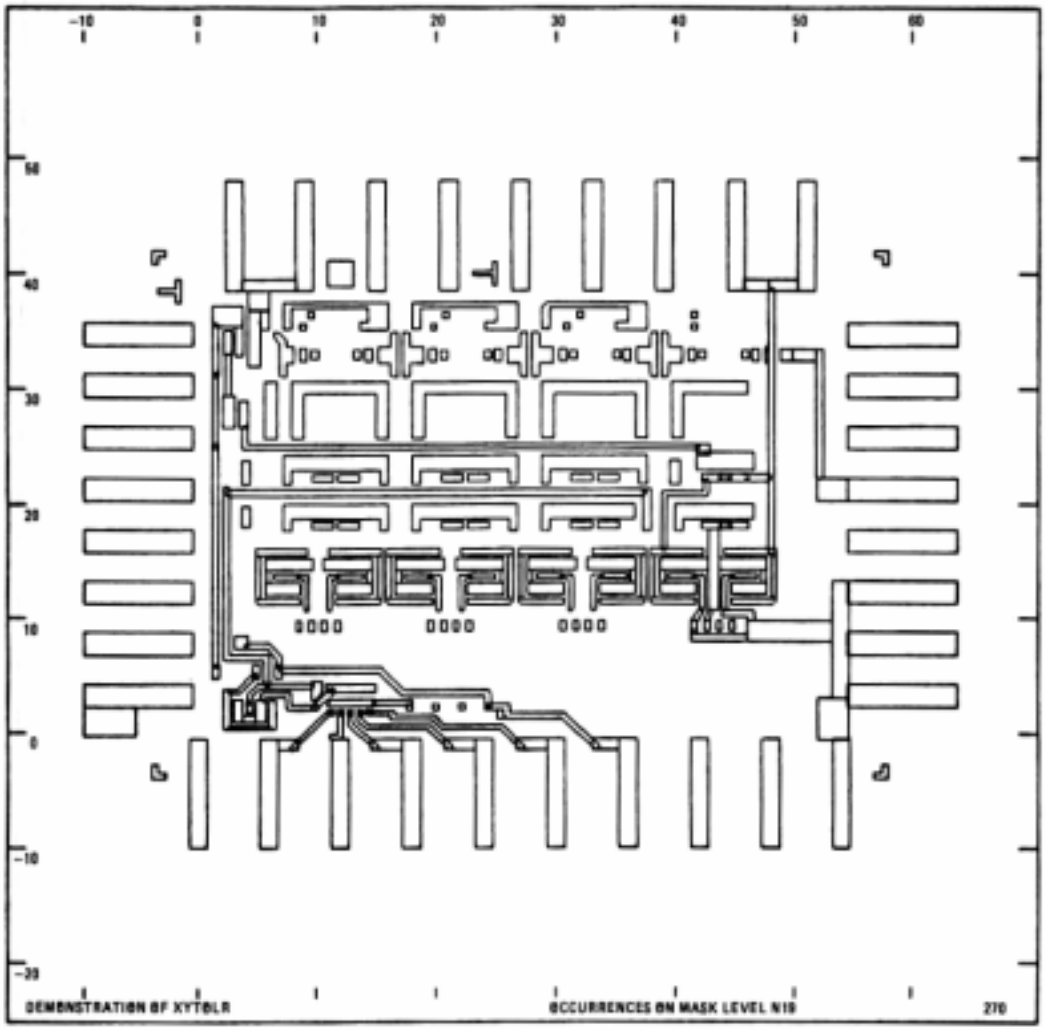

Fig. 1—A simple metallization pattern. Two shapes have been inserted in violation of the required minimum clearance of 0.3 units.

A tolerance test was induced by the input statement

/ N19 +0.15 NODEXX.

NODEXX is a control word requesting that new connections (violations) involving occurrences already indirectly connected through other occurrences (i.e., members of the same node) be plotted separately from new connections between occurrences not so connected (i.e., members of different nodes). Figure 2 is the graphic output illustrating the latter set of violations, which are the ones of interest in this case.

III. PROGRAM STRUCTURE

3.1 *Data Structures*

The output of the XYMASK language processor is a sequential file or tape containing an explicit description of each occurrence in the mask set. Normally, this file is used to drive a precision plotting device. XYTOLR, however, needs parallel access to many occurrence descriptions and therefore must have a randomly accessible data base, in core memory, or on random access disc or drum. Since XYTOLR has been implemented on a computer (HIS 6070) with a large core memory, its data structures are held entirely in core. The principles of operation would not be significantly different for virtual memories or paged memories, or for random access external storage.

The information contained in the XYMASK output file (as generated by a somewhat modified version of the XYMASK language processor) is converted to a data structure, the unit of which is the "occurrence block." Each occurrence block describes one occurrence. The length of each occurrence block is $11 + 2L + 6A$ core locations (36 bit words), where L is the number of line segments, and A is the number of arc segments, composing the occurrence. All geometric information is stored as single precision floating point numbers.

Occurrence blocks are stacked sequentially in memory. Each occurrence block contains, along with line and arc descriptions, a header containing the extreme coordinates of the figure (circumscribed rectangle), its serial number, an index defining its atomic name, and pointers which serve to organize the data structure. By means of these address pointers, each of which is the address of an occurrence, the occurrence block is made a member of three linked lists. By following one list ("main list") from the beginning, every occurrence is encountered in order of increasing x (min), which is the minimum x-coordinate in the figure (including projecting arcs). By following another list, every occurrence of the same atomic name is found. The third list pointer is the "intersection ring pointer," by which all occurrences which

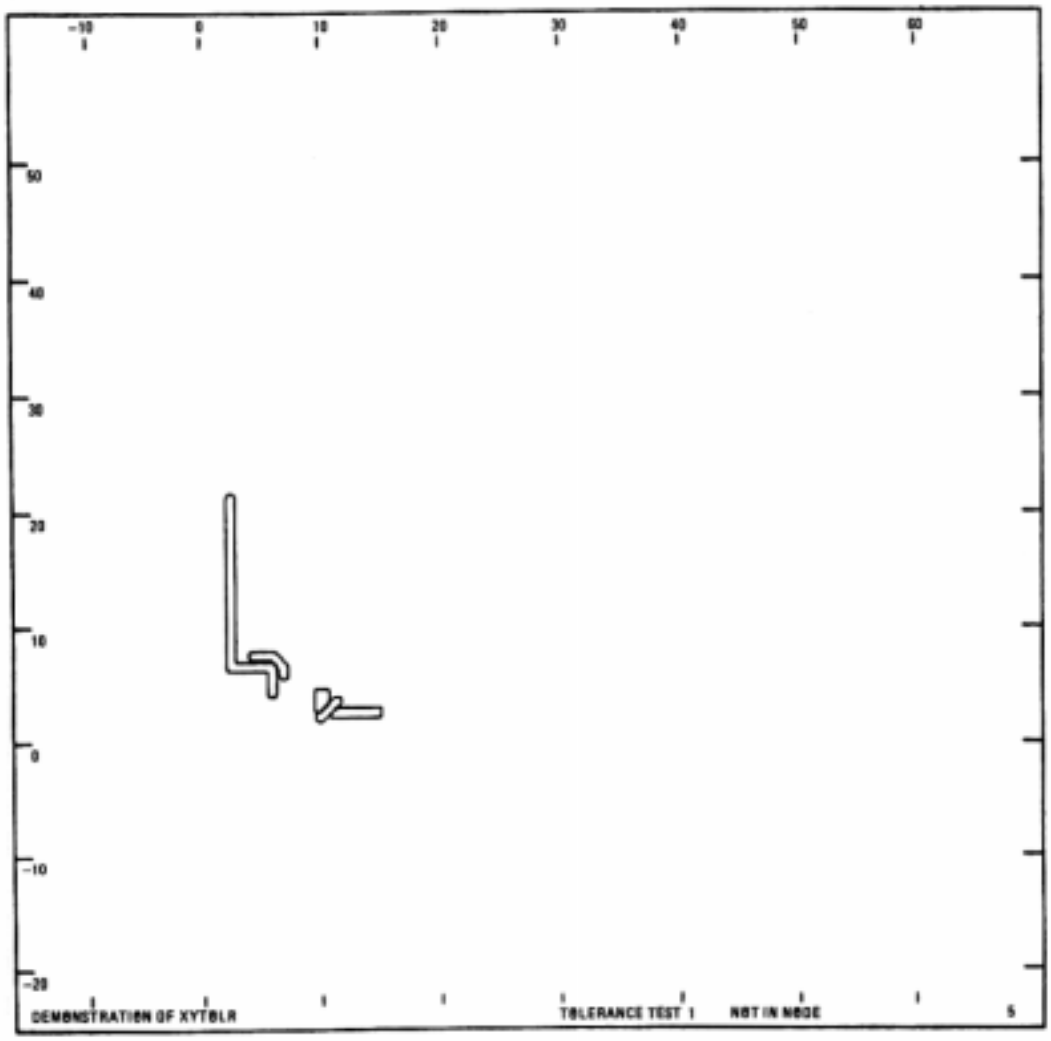

Fig. 2—Output of an XYTOLR test on the pattern of Fig. 1. The input statement was / N19 +0. 15 NODEXX.

contact one other directly or indirectly are ultimately chained into a ring, or circular list, which represents a node.

The data structure also contains, among other things, a table of atomic names and of mask level names. Cross-referencing among these tables and the occurrence blocks facilitates finding the address of a desired occurrence block or the properties of an occurrence the block address of which is known.

One location within the data structure contains an important number known as the "critical distance." Two points separated by less than this distance are considered coincident. The critical distance, while easily variable, is generally set as 0.0001 times the overall mask size. The quantity by which tolerance test enlargement parameters are modified, as described in Section 2.2, is 1.1 times the critical distance.

3.2 *Subroutines*

The working tools required to perform analyses of the data structure are subroutines which generate, manipulate, and display occurrences, given their block addresses. Using these subroutines, along with the crossreferencing provided by the various pointers and tables, a programmer may construct systems which operate on mask levels or on the entire mask set without concerning himself about the detailed structure of occurrence blocks. Many such subroutines are contained in XYTOLR, performing both simple and complex functions. Two of particular importance are CONTAC and NLARGR.

CONTAC tests a pair of occurrence blocks (call them "a" and "b") for connectivity. It returns one of four numerical values, each corresponding to one of the four possible classes of connectivity of a and b: disjoint, a inside b, b inside a, and overlapping (intersecting boundaries). CONTAC first checks the extreme coordinates of the two occurrences to see whether they are clearly disjoint. If not, the test for overlapping boundaries is made: each side of a is solved analytically with a side of b to see if there is an intersection within the bounds of both sides. (This analytical solution is returned by a subroutine called INSECT, which handles lines and arcs interchangeably.) If no such intersection is found, the "inside" test is performed: a line segment is generated from a point on the smaller occurrence (in area) to a point outside the larger one, and the number of intersections of this line segment with the larger occurrence boundary is found. If this number is odd, the smaller occurrence must be within the larger one; if even, they are disjoint.

NLARGR expands or contracts occurrences. Given the address of an occurrence block and a positive or negative expansion distance, it generates a new occurrence block representing the expanded occurrence. In its normal mode of operation ("square mode"), it moves each occurrence side perpendicular to itself by the specified distance and extends it to intersect its relocated neighbors. NLARGR has a more precise, but more expensive, mode of operation ("round mode") which inserts arcs in outside corners so that the new outline truly represents all points equidistant from the original.

IV. OPERATION OF XYTOLR

XYTOLR checks a mask design for tolerance and clearance errors by the systematic application of subroutines NLARGR and CONTAC to the data structure representing the mask design. If the connectivity

value returned by CONTAC for a pair of occurrences is changed when one or both of the occurrences is expanded by the prescribed clearance distance, a violation of the tolerance rule is indicated.

XYTOLR constructs an essentially complete connectivity matrix for all the occurrences in the mask set. The data structure is then modified in accordance with the user's instructions so that all occurrences on the appropriate mask levels are enlarged or contracted; each matrix element is recomputed and those elements which differ from the original are flagged as errors.

In principle, a complete connectivity matrix for a mask set would have as many elements as the square of the number of occurrences in the mask set—a million matrix elements for a thousand-occurrence mask set. The problem of storing such a large matrix is simplified by the following considerations:

(*i*) The matrix is diagonally symmetric and the principal diagonal is of no interest.
(*ii*) Each matrix element may have only four values (the four types of connectivity) and therefore may be represented by a two-bit binary field.
(*iii*) The matrix is generally sparse; that is, most of its elements are zero, and it may usually be reduced to a set of submatrices with a much smaller total number of elements.
(*iv*) The manner in which the tolerance test is conducted permits the matrix to be scanned in a predetermined order rather than randomly; therefore, it may be kept in peripheral storage with only relevant elements in core at any time.

XYTOLR first constructs the occurrence-block data structure from the modified XYMASK output file. At this stage, the main list pointer of each occurrence block points to the next sequential block, and the intersection ring pointer of each occurrence block points at the block itself. A sorting program rearranges the main list pointers so that occurrences can be followed in order of increasing x (min).

Next, the "nodes" of the data structure are identified. A node is a set of occurrences each of which is connected to every other, either directly or indirectly through other members of the set. (Two occurrences are considered "connected" if they are not disjoint.) If the data structure represented a metallization pattern, these nodes would be the electrical nodes; the term is here generalized to mean any set of interconnected occurrences. To find the nodes, each occurrence is tested for contact with other occurrences using CONTAC. The first

occurrence in the main list is tested against each subsequent occurrence, similarly the second, etc. The list need be followed in each case only until an occurrence is found the x (min) of which is greater than the x (max) of the test occurrence. Geometrically, each occurrence is tested against all those which impinge on a vertical strip defined by its minimum and maximum x-coordinate.

When two occurrences are found to be in contact (i.e., not disjoint), their intersection ring pointers are interchanged. This operation is sufficient to combine two isolated occurrences into a ring (circular list), to add an isolated occurrence to an existing ring, or to combine two existing rings. (Occurrences already members of the same ring are not further tested against one another.) When the operation is complete, each node is represented in the data structure as a separate circular list. Isolated occurrences are "one-membered" rings.

Next, the intersection matrix is generated. Since an occurrence must be disjoint with any occurrence not a member of its node, only matrix elements relating members of the same node need be computed. Clearly, the more nodes, and the fewer occurrences per node, the fewer matrix elements need be computed. At this point, the nodes and their submatrices are stored on two sequential files, the "node file" and the "matrix file." The data structure is scanned for nodes; each time one is found, its occurrences are written out sequentially; as each occurrence is written, its connectivity values with subsequent members of its node are determined and written out as a sequence of two-bit fields. The files thus generated are used for all subsequent tolerance tests.

Each tolerance test statement in the user's input is decoded and the appropriate enlargement for each mask level determined. For each tolerance test, the data structure is reconstructed using occurrence blocks read back from the node file. As each occurrence block is read, its mask level is determined and the required enlargement applied, if necessary, using NLARGR. When the new data structure is complete, the connectivity values relating occurrences in each node are recomputed in the same order as that in which they appear on the matrix file and compared with corresponding values read back from that file. Any failure of these values to match indicates a tolerance test violation, and output routines are called to add descriptions of the pair of offending occurrences to the output report and write the occurrence blocks on a graphics file for later plotting. Finally, each occurrence is tested for connectivity with occurrences not members of its node; any contact indicates an error.

The generation of the original noded data structure and of the node and matrix files, which are used in all tolerance tests, incurs a con-

siderable overhead expense for large mask sets. The user is therefore given the option to have all this information saved on a magnetic tape as soon as it is generated. The data structures and files can be reconstructed efficiently from this tape if further tolerance tests are necessary, or in case of abnormal termination of the computer run before all tolerance tests have been completed.

V. EXTENSIONS OF THE XYTOLR SYSTEM

The program XYTOLR is built about a data structure and a set of subroutines which generate, manipulate, and display elements of this data structure. These subroutines can be used to construct mask design and analysis programs other than XYTOLR. The programmer need never concern himself with the detailed structure of the occurrence blocks or with computations involving the fundamental geometric data, but may interact with the data structure almost entirely through subroutine calls which provide the information or perform the actions logically required by his program.

The program ENMASK, assembled mainly from XYTOLR components in this way, creates new mask level descriptions from XYMASK originals. The new descriptions, which are punched in XYMASK input language, represent mask levels on which each original shape has been expanded or contracted by a given quantity, so that it surrounds, or is surrounded by, the original shape. ENMASK can also, if desired, merge connected shapes, replacing each connected set of occurrences by the outer boundary of their union. This program provides a simple solution to certain mask design problems. It utilizes an occurrence-manipulating subroutine called OVRLAP,[3] which, given two occurrence blocks, can generate new occurrence blocks representing each simple closed figure resulting from their superposition.

D. G. Schweikert has used XYTOLR components to construct a system called ICSORT, which can "decode" an XYMASK mask set description representing a bipolar integrated circuit, automatically identify the transistors, diodes, and resistors, and generate a node list topologically equivalent to the circuit diagram. The designer can thus assure himself that the masks he has designed in fact represent the desired circuit.

VI. IMPLEMENTATION

XYTOLR has been implemented on a Honeywell Information Systems (formerly General Electric) 6070 computer, under the GECOS

operating system. The component programs of XYTOLR have been written principally in FORTRAN IV. Assembly language subroutines have been used sparingly to perform functions unique to the HIS 6070. Some FORTRAN code involving character manipulation (for titles, etc.) has been designed on the assumption of a six-character machine word and would have to be modified to execute on a computer with a different internal character representation. Graphic output may be obtained on such devices as the General Dynamics SC-4060 microfilm plotter. Calls to the Bell Laboratories plotting subroutines have been centralized for easy replacement.

XYTOLR occupies a core region the size of which it varies dynamically according to the current size of the occurrence-block data structure. The size of the region is usually limited by system considerations to about 96000 36-bit words. Of this, the program (which is heavily overlaid) occupies about 20000 words, and the rest is available for data. Under these conditions, 2800 to 4000 four-sided figures, or 2200 to 2800 eight-sided figures, can be accommodated, the exact number depending on the manner in which the mask set design is organized.

Running time of XYTOLR depends on the size and complexity of the mask design and on the tolerance tests requested. Running XYTOLR on the simple metallization pattern of Fig. 1 took 62 seconds of processor time, including generation of tape to drive the microfilm plotter. The tolerance test itself took 18 seconds, and additional tests would add about this cost apiece. Extensive checking of large and complex mask sets has on occasion required as much as an hour or more of processor time. The computer charges necessary for such checkout are considered minor compared to the overall cost of the mask set.

VII. CONCLUSIONS

A computer program called XYTOLR tests integrated circuit mask designs, described in the XYMASK input language, for conformity with tolerance rules which require minimum clearances between different regions and minimum overlaps between regions intended to be connected. Tests are specified in a simple input language. Graphical and printed output is produced which identifies all deviations from the specified rules and facilitates their correction. In the HIS-6070 computer environment at Bell Laboratories, mask sets with as many as 3000 to 4000 individual shapes may be processed.

Mask designers have found XYTOLR to be a valuable tool for final

mask checkout before production approval is granted. It is especially impressive when the program finds one or two demonstrable errors in a mask set which had been considered completely checked and ready for production. The program has proved reasonably easy to use and the output reasonably easy to interpret.

The techniques, data structures, and subroutines which constitute the XYTOLR program may be used as the fundamental building blocks from which other mask analysis and design programs may be constructed.

REFERENCES

1. Fowler, B. R., "XYMASK," Bell Laboratories Record, *47*, No. 6 (July 1969), pp. 204–209.
2. Gross, A. G., Raamot, J., and Watkins, Mrs. S. B., "Computer Systems for Pattern Generator Control," B.S.T.J., *49*, No. 9 (November 1970), pp. 2011–2029.
3. Yamin, M., "Derivation of All Figures Formed by the Intersection of Generalized Polygons," B.S.T.J., this issue, pp. 1595–1610.