Computer Systems for Pattern Generator Control

By A. G. GROSS, J. RAAMOT and MRS. S. B. WATKINS

(Manuscript received May 29, 1970)

Computer systems play a fundamental role in the operation of precision integrated-circuit pattern generators. This paper first describes the XYMASK *system which provides a language for describing the geometric shapes in a set of masks and generates graphical artwork on a number of different pattern generators. The remainder of the paper is devoted to discussions of system-design considerations and algorithms for generating input to the primary pattern generator and the electron beam machine.*

I. INTRODUCTION

Computers are indispensable today in the operation of any sizable mask-making laboratory. Nearly all precision pattern generators are either directly computer controlled or else require input of a form which can be reasonably obtained only through the use of computers. Furthermore, the complexity and sheer volume of masks currently required effectively prohibit nonautomated procedures.

The mask-making laboratory system described in this issue relies heavily on the use of computers. The first part of this paper describes a system of programs which links a circuit designer to the mask-fabrication processes; the next two sections discuss algorithms and programs for generating input to the primary pattern generator (PPG) and the electron beam machine (EBM).

1.1 *Computer-Aided Generation of IC Masks*

Masks are tools required in the fabrication of integrated circuits and other devices. The starting point in mask design is thus an electrical schematic or logic diagram of the desired device. An engineer or technician first allocates scaled geometric shapes to each of the circuit components; he then arranges and rearranges these shapes

on a similarly scaled substrate area. During this placement phase, many criteria are generally involved in evaluating the suitability or desirability of one arrangement over another. Some examples are thermal interaction, packing density, and the ability to realize the required component interconnections. The latter criterion is really applied in the next phase wherein the interconnection pattern is designed in detail. For most cases, several iterations between the placement- and interconnection-design phases are required before a satisfactory layout is obtained. At this point the geometric details of all the required masks are completely known; the next step in the process is mask generation.

The draftsman or engineer is now faced with the problem of transforming the mask layouts into a form suitable for driving a pattern generator. The severity of this problem depends on two factors: the form of input required by the particular pattern generator, and the complexity of the masks. For pattern generators which are concerned solely with the outline of the geometric features, such as an automatic knife coordinatograph cutting rubylith, the solution is tedious but straightforward. Either manually or via a digitizer, the coordinates of the endpoints for each horizontal feature boundary line, followed by the coordinates of each vertical feature boundary line, can be recorded on punched paper tape for each mask level. This tape would then be processed by the coordinatograph, the rubylith master peeled, and the masks obtained after appropriate photographic processing of the rubylith master. However, for more sophisticated pattern generators which operate by filling in the interior of mask features with beams of light or electrons on photographic film, substantial use of computers is necessary to convert the mask geometry into commands acceptable by the pattern generators.

1.2 *The* XYMASK *System*

The system of programs in use at Bell Telephone Laboratories and Western Electric Company for computer-aided production of integrated-circuit masks is known as XYMASK. First operational in late 1967 and subsequently improved and modified, the current version of XYMASK evolved from two earlier generations of mask-making programs. Three of the more important system-design goals may be stated as follows.

(*i*) It should provide a standard user-input language for conveniently and efficiently describing mask-feature geometry.

(*ii*) Insofar as possible, the system should be independent of any particular graphical output device.

(*iii*) The implementation should be highly independent of the host computer to enhance portability of both the system and the mask specifications.

The first of these goals is extremely important. Its realization greatly facilitates the transmittal of device designs not only among Bell Laboratories locations but also between Bell Laboratories and Western Electric Company for production. Moreover, the user-input language is a vital factor in the interface between the mask designer and the system since its convenience and flexibility have a direct bearing on user acceptance and satisfaction.

The second goal is a necessity due to the diversity and number of graphical output devices available at Bell Laboratories locations. In an indirect manner, attainment of this goal also simplifies the addition of new output-device capability as we shall see below.

The third goal arises from the use of different large-scale computers at Bell Laboratories and Western Electric locations and the ever-present possibility of new ones being acquired. The most important user benefit is the complete independence from any particular computer of the mask descriptions encoded in machine-readable form in the input language; the same mask-description input deck will produce identical artwork on different computers. Again indirectly, attainment of this goal has simplified program implementation and maintenance. The implementation is almost exclusively in a subset of FORTRAN IV common to the IBM 360 and GE-635 computers; there is essentially one set of source-language programs which runs on the several different computers.

1.3 *The User-Input Language*

As a preliminary to discussing the system organization of XYMASK, it will be helpful to describe briefly the user-input language. A somewhat more detailed description is given by B. R. Fowler[1]. Basically, the input language provides a vehicle for describing the various geometrical shapes contained in a mask or set of masks in a computer-readable form. As such, the most primitive statements in the language are used to specify the interiors of three basic geometrical shapes: rectangles, polygons, and paths. In this context, rectangles are defined to have their edges parallel to the coordinate axes and are specified by giving the coordinates of the vertices on either diagonal. The statement

label RECT mask, 10, 20, 30, 40

illustrates the format of the primitive statements and defines a rectangle with the lower-left vertex at $X = 10$, $Y = 20$, and upper-right vertex at $X = 30$, $Y = 40$. The label and mask attributes are discussed below. The polygon primitive is used to define generalized polygons having either straight lines or circular arcs as edges. The shape and size are fixed by giving the coordinates of the vertices in the order in which they are encountered in either a clockwise or counterclockwise tour of the periphery. The path primitive is used to specify a path of given finite width. The size and shape are fixed by giving the width and the coordinates of the endpoints and breakpoints of the centerline as they are encountered in a tour along the path. The centerline may contain circular arcs as well as straight-line segments.

The preceding paragraph discussed only the specification of the shapes and sizes of the basic geometrical features. The positions of these features on the masks may be specified in either of two ways. If a label attribute is not specified for the feature, the coordinate values define its position as well as its shape and size. On the other hand, if a label attribute is given, separate input-language statements must be used to specify the position. In addition to position, these statements also permit the orientation of the feature to be altered by reflection about either coordinate axis together with a rotation through an arbitrary angle.

In general, a set of individual but inter-related masks is required in the fabrication sequence for an integrated-circuit device. A transistor, for example, may require geometrical features on a number of different masks for forming collector, base, and emitter regions. The XYMASK user-input language allows specification of all geometrical features occurring in all required mask levels for a device in whatever intermixed order is most convenient for the user. In order to correlate the various features with the appropriate mask levels, a mask-level identification is required as part of the specification of the rectangle, polygon and path primitives.

It is often desirable and useful to treat a group of geometrical shapes as a structural entity; for example, it is far more convenient to position a transistor at the required locations as a structural entity rather than as a set of individual primitive shapes. The user-input language allows this hierarchical nesting of structures to an arbitrary depth. In other words, it is possible to define a structure which contains structures of lower "order" as well as basic geometric shapes. The structure may be positioned on the masks, possibly with reorientation, as described above for simple geometric shapes. This hier-

archical structuring in conjunction with reorientation allows the user to take advantage of repetitions and symmetries in the design in order to reduce the number of statements and effort required to encode the design in the user language.

Statements are also available in the input language to retrieve previously designed structures from XYMASK libraries and to invoke component structure-design routines. Transistor designs are typical library entries. An integrated-circuit designer generally uses transistor designs which have been thoroughly tested and characterized. These designs are stored as library entries which contain the XYMASK language specification in the form of hierarchical groupings of the appropriate primitives. Library retrieval provides a sort of shorthand for the user in that only the particular library and the entry identification need be specified in the input deck in contrast to the equivalent set of XYMASK input statements.

Computer programs have been developed to design certain components and structures used in integrated circuits. Pattern generation for thin-film meander resistors, and the generation of sheafs* of interconnection paths are examples of such programs in current use.

Versions of these programs, called design routines, have been integrated into the XYMASK system. A single statement in the input language allows the user to specify the desired routine together with whatever parameters are required. Output from the routine consists of XYMASK statements specifying the generated design. These statements are automatically incorporated into the user's input.

The final feature of the user language to be discussed deals with the specification of particular graphical devices and output options. Graphical output may be requested either in the form of outline drawings or finished artwork. The outline drawings are generally produced on line plotters and are used to verify that the mask descriptions as encoded in the input language are correct. As implied, only the outlines of the geometrical features are displayed. The finished artwork is the desired end product of the system; for plotters working on photographic film, the interiors of the geometrical features have one tonality (clear or opaque) while the area which is exterior to all figures has the opposite tonality.

A single statement is used to indicate the plotter and any pertinent parameters such as drawing type and scale factor. The user has the

* A sheaf is a family of paths each member of which can be derived from a generic member by translating each of its path segments normally through a given distance and lengthening or shortening it as required to create a nested copy of the generic path.

capability of requesting individual drawings or artwork for any or all masks. He may also request composite drawings of any two or more masks. This latter feature is widely used for error checking.

1.4 XYMASK *System Organization*

A simplified diagram of the XYMASK system is shown in Fig. 1. The major program segments are the input preprocessor, the input processor, the execute processor, and the family of device-dependent output postprocessors. Input to the system is a machine-readable description of the desired masks encoded in the XYMASK user language. This input is free format and may be generated by hand encoding and keypunching, digitizing large-scale layouts, or by other computer programs such as interconnection-routing routines.

The input first passes through the input preprocessor. All input statements other than design-routine invocations or library retrievals are transmitted to the expanded input file without change. When a design-routine invocation is found, control is passed to that design routine, and the generated XYMASK statements together with the invocation are transmitted to the expanded input file. Library retrievals

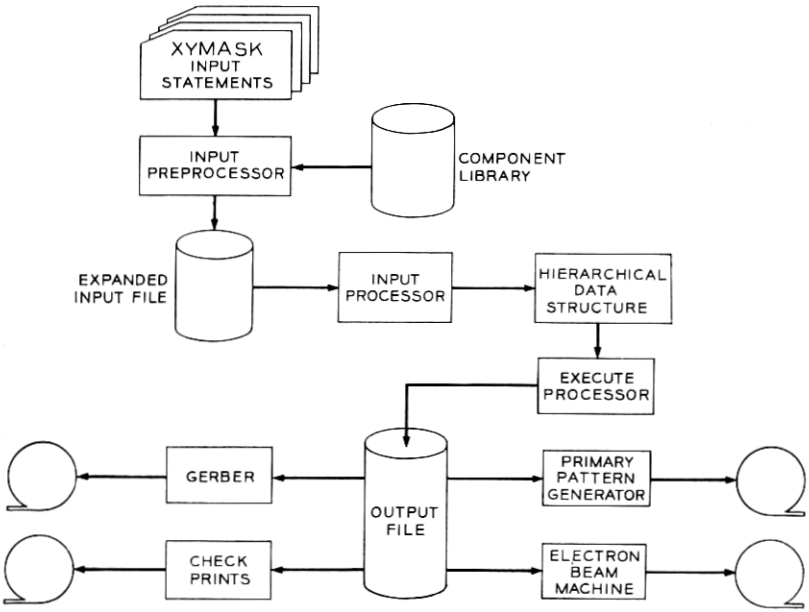

Fig. 1—The XYMASK system.

are treated similarly in that retrieval is made when the statement is encountered in the input deck; the retrieved XYMASK statements along with the retrieval statement are transmitted to the expanded input file. At the conclusion of the preprocessor phase, then, the expanded input file contains the original input statements interpolated with the results of any design-routine invocations or library retrievals.

The system design of the remainder of the XYMASK system was heavily influenced by the desired relative independence from any particular graphical output device. Accordingly, output-device dependence is relegated to a family of postprocessors each of which receives input from a common file referred to as the 'output file'.

This output file contains a representation of each of the masks requested in the XYMASK input deck in a form such that all device-independent processing has already occurred. Each mask is represented by a separate subfile, and each subfile contains only the defining coordinates of individual paths and polygons in their final positions and orientations.

The input and execute processors must then transform the expanded XYMASK input statements into the form required for the output file. The most significant aspects of this transformation are as follows: removal of all hierarchy by generating new copies of the various primitives as required while simultaneously carrying out specified reorientation and positioning; and sorting the resulting primitives into separate sets according to their individual mask-level identifications.

The above aspects of the transformation suggest that detailed descriptions of all required masks be available in memory in a convenient form prior to starting the transformation. Thus the input processor reads the input-language descriptions of the masks, makes extensive error checks, and stores the descriptions in a hierarchical data structure. Upon completion of this process, the execute processor comes into play to generate the output file from the data structure.

When output-file generation is complete, the appropriate postprocessor for the first mask is activated according to the output device specified by the user. Upon completion, processing is initiated on the second, perhaps using a different postprocessor if the user so desired. In like fashion, the remainder of the output file is processed and the job terminates.

Each postprocessor is responsible for the ultimate generation of artwork on a particular graphic-output device. In general, the postprocessor output is a magnetic tape which drives the actual device,

although on-line devices, such as the STARE[2] line-drawing plotter, are easily accommodated. We can again view a postprocessor as a data transformer; it is responsible for reading each path and polygon specification from the output file and generating the proper output-device commands or codes for plotting that figure. The system design is such that all postprocessors are essentially independent programs which receive all of their input from the XYMASK output file. The system is thus open ended in the sense that new postprocessors can be easily and conveniently added.

With regard to execution times for a typical set of masks, the input and execute processors each require on the order of one-minute running time on an IBM 360/65. Postprocessor execution times are generally longer and tend to dominate other costs for the run.

The following two sections of the paper are devoted to detailed discussions of specific postprocessors for the PPG and EBM plotters described elsewhere in this issue. The two differ fundamentally in the manner in which pictures are produced. The EBM is a random-access plotter; the order in which mask features are plotted is immaterial. The PPG, on the other hand, produces pictures using a raster-scan technique. The contributions of all features intersected by each scan line must be determined and transmitted to the device in the order needed to generate the picture.

The PPG postprocessor was developed at Bell Laboratories, Murray Hill, New Jersey, by A. G. Gross. The EBM postprocessor was developed at the Western Electric Engineering Research Center, Princeton, New Jersey, by Mrs. S. B. Watkins and J. Raamot.

II. THE PPG POSTPROCESSOR

The operation and functioning of the PPG together with its control computer are discussed in this issue by A. Zacharias, et al.[3] For convenience, we will briefly review here those aspects which are of importance to the postprocessor.

For our purposes, we can consider the photographic plate plotting surface to be a rectangular lattice of 26,000 × 32,000 addressable points. A writing beam scans the lattice on a line-by-line basis, with the beam turned on at those address points interior to mask features, and off otherwise. The writing beam is controlled by a 26,000-bit display buffer in the control computer with each bit position representing one address along the scan line; the beam is turned on or off

at an address according to whether the content of the corresponding bit position is one or zero. After completing a scan line, the bit configuration in the display buffer must in general be modified to correctly represent the geometric detail in the next scan line. When updating is completed, the bit configuration is again used to modulate the writing beam; this cycle continues until all 32,000 scan lines have been completed.

2.1 *Interface between Postprocessor and Control Computer*

Let us for a moment consider the subsystem comprised of the PPG postprocessor and the control computer program. The postprocessor runs on a large central computer, receiving input from the XYMASK output file discussed previously, and writing output on magnetic tape. The information is read from the magnetic tape by the control computer program and used to load and update the display buffer. The magnetic tape constitutes an interface between two computer programs: the nature of the information on the tape can thus be varied to share, in some sense, the computational load between the two computers.

At one extreme, essentially all computation can be made in the postprocessor. The magnetic tape contains 32,000 records, each representing a complete 26,000-bit display buffer configuration. In this format each mask requires transmission of something like a billion bits between the computers. At the other extreme, the control computer can process the XYMASK output file and develop the display-buffer contents. Far too much computation is relegated to the control computer since display buffer regeneration cannot in general keep up with the pattern generator plotting rate. The result is a severe degradation in plotting time.

A compromise between the above extremes can be reached by considering the basic information required to properly load the display buffer. Let us see what is involved for an extremely simple mask containing a single vertical bar. For all scan lines which do not intersect the bar, the display buffer must contain all zero bits, while the bit configuration for the remaining scan lines is invariant and need only be set once. The basic data needed to load the display buffer involves only details of the changes, if any, in the bit configuration between successive scan lines. This is true even for complex masks since a high degree of similarity generally exists between one scan line and the next. One is thus naturally led to consider a magnetic

tape encoding scheme which takes advantage of these similarities by detailing only the required configuration changes from one scan line to the next.

A complete description of the various commands used in the encoding scheme appears in Ref. 3. The commands fall naturally into three groups. The first group contains commands of an incremental nature for updating the bit configuration in the display buffer. Various combinations of these commands may be used to indicate that strings of one or more bits in the buffer are to be set to zeros or ones as required for the next scan. All bits not referenced in this fashion represent recurring mask detail and are unchanged for the next scan. The second group of commands deals with complete scan-line configurations. Commands are provided for specifying that the bit configuration for the next N scan lines is invariant, contains all one bits, or contains all zero bits. Commands in the final group are used to pass various parameter values to the control computer and are not of interest here.

2.2 *Postprocessor Algorithms*

We turn now to the functioning of the postprocessor. The input data resides on the XYMASK output file in the form of various parameter values and the coordinate specifications for the individual path and polygon geometric features in the mask or masks to be generated. The output is written on magnetic tape and consists of appropriate sequences of the commands discussed above. The necessary data processing can be iteratively characterized as follows: given the set of geometric figures intersected by the previous scan line, determine the set of figures intersected by the current scan line and compare the respective display buffer configurations; the result of this comparison is expressed in the encoding scheme and written onto tape. Iteration commences with a null set of figures in scan-line zero, and terminates when scan-line 32,000 has been processed.

The practical aspects of the above characterization belie its simplicity of statement. A single mask may contain several thousand individual geometric features. Furthermore, the features occur on the XYMASK output file in random order with regard to geometric position in the mask. Finally, it is important to accelerate the scan-line comparison process by quickly detecting sequences of scan lines which have the same display buffer bit configuration. The following paragraphs give a description of the methods and algorithms which were used.

The coordinates of the mask features on the output file represent final device dimensions measured in micrometers from an arbitrary datum point. These coordinates must be scaled up by the appropriate factor to compensate for photographic reductions of the primary pattern, and converted to address units. A coordinate translation is then made to center the mask on the primary pattern plate. The postprocessor is capable, at the user's option, of generating either normal-tone masks having opaque features on a clear background, or reverse-tone masks displaying clear features on an opague background. It is an interesting and perhaps unique characteristic of the system that the two tonalities are produced with equal ease and facility. For simplicity, we will consider only normal-tone processing.

Given the set of individual mask features as input and considering the raster-scan process by which the artwork is created, it is clear that we are primarily interested in the feature boundaries. Returning to the simple mask discussed above, the writing beam is switched on at the left boundary of the bar, remains on in the interior, and is switched off at the right edge. Thus for our purpose the rectangle is totally characterized by its left and right boundary lines together with their respective tonality shifts. More generally, each polygon feature in the mask can be similarly characterized by listing all of its boundary line segments not parallel to the scan-line direction, together with the appropriate tonality transitions. Any arcs which occur are approximated by a sequence of chords and are thus reduced to sets of line segments. Since path features are described on the XYMASK output file by centerline coordinates and width, some additional computation is required. Any arcs in the centerline are first approximated by chords, and path outline then obtained by translating the centerline line segments normally through distances of plus and minus one-half the path width. The path then becomes a polygon and is treated as above.

2.3 *Postprocessor Structure*

A simplified diagram of the postprocessor is shown in Fig. 2. Each mask requires a complete pass through the system. The line segment decomposition routines read the mask-feature descriptions from the XYMASK output file, convert the coordinates into address units, decompose each feature as described above, and write the resulting line segments with their tonality shifts onto the line-segment file. The set of line segments is next sorted into an order convenient for further processing. Each line segment is described by the two coordi-

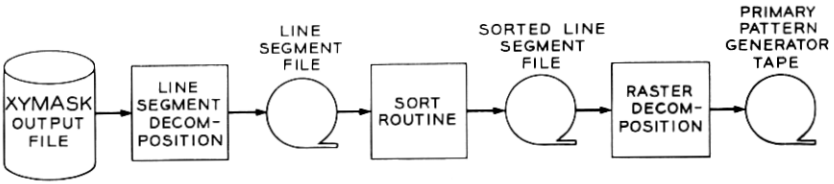

Fig. 2—PPG postprocessor system.

nate pairs of its endpoints. The endpoint which has the lower value for its Y coordinate is termed the lower endpoint. The sort is carried out using the lower endpoint Y value as the primary key, and the lower endpoint X value as secondary key. At the conclusion of the sort, the sorted-line-segment file contains the line segments in the order in which they are encountered by the raster scan. Line segments first encountered by scan N precede those first encountered by scan $N + 1$, and if several line segments are first encountered by scan N, they occur on the file in the order of increasing-scan positions.

The final section of the postprocessor reads the sorted-line-segment file, determines configuration changes between scan lines, and writes the appropriate commands on the PPG tape. This operation is carried out using a 26,000-bit image of the display buffer containing the bit configuration of the previous scan line and a linked list of all line segments contributing to the current scan line. The line-segment representation is compared to the bit-image configuration; any differences are appropriately encoded and written on the tape, and the relevant bits are changed in the bit image. When the comparison has been completed, the list of relevant line segments is updated by deleting those which do not intersect the next scan line and interpolating any new ones which do from the sorted-line-segment file. The scan routines are fairly simple but involve significant computer time. The postprocessor minimizes the number of scan comparisons by examining the line-segment list looking for scan lines which are identical to the previous one, or contain all-zeros or all-ones configurations. When such configurations are found, the scan comparison is bypassed, and the appropriate commands are written on tape. This capability allows very rapid processing for masks containing features having no slant-line boundaries.

The postprocessor execution time varies considerably with the complexity of the mask being generated. A typical interconnection mask

ordinarily requires several minutes on an IBM 360/65 and writes something on the order of one-quarter-million bits on the output tape.

III. EBM POSTPROCESSING AND ALGORITHMS

This section describes a system of programs which interfaces the EBM pattern generator with XYMASK. This system consists of a postprocessor within the XYMASK system and a program for the pattern generator controller. The following short description of the EBM pattern generator will give an insight into the data transformations performed in both the XYMASK postprocessor and the control computer program.

3.1 *The EBM Pattern Generator*

The EBM is similar to a cathode ray tube; in both, a beam of electrons is focused and deflected to form a spot on a target. One difference is that in the EBM, the target is a high-resolution photographic plate, whereas in a cathode ray tube it is a phosphor screen. As the electron beam hits the target, the electrons directly expose the photographic emulsion and thereby produce a fine spot. A detailed description of the EBM pattern generator is given in this issue by W. R. Samaroo, et al.[4]

The EBM pattern generator includes a digital-control computer that drives, through appropriate interface equipment, a set of electrostatic beam deflection plates located within the EBM. The electron beam position on the target is controlled to fill mask features by drawing a sequence of adjacent line segments parallel to one coordinate axis. Fill-line data in the form of position and length are transmitted from the control computer to the interface where the digital fill-line data are converted to a sequence of analog voltages that are applied to the deflection plates.

Since a typical mask pattern may contain an estimated 10^5 fill-lines, it is impractical to read or even to store this data in the control computer. To make data processing more practical, the following strategy is used for the EBM pattern generator: While the interface controls the drawing of one fill-line segment, the control computer calculates the position and length of the adjacent line segment.

Input data to the control computer consists either of paths or of pairs of left-hand and right-hand boundaries specified by the endpoints of straight-line segments or the endpoints and centers of

circular arcs. With this pattern-coding scheme, approximately 4000 words are required to represent a typical mask pattern of 10^5 fill-lines. This small volume of input data facilitates data transfer from the XYMASK postprocessor to the control computer. It is the task of the postprocessor to read the XYMASK output file, transform the data to right-hand and left-hand boundaries for the EBM, and to output this data.

3.2 *The EBM Postprocessor*

The EBM postprocessor is written in the *1 language[5] (read as star one) and in FORTRAN IV for the IBM 360/50 computer. *1 is used because of its inherent power in processing list-data structures and FORTRAN IV is used for input, output, and some of the more complex calculations.

The way the XYMASK output file describes the features of a mask does not conveniently distinguish for the EBM the areas inside and outside the periphery of each feature. Generally speaking, the more automatic the drawing device, the more work has to be done by a computer to obtain this information. Devices such as the coordinatograph and the Calcomp plotter, for example, require data in a form very similar to that of the XYMASK output file because these devices cut or draw along the periphery of each feature. Since the Calcomp plots are part of the "debug" steps and are used for alignment and correction, no further processing is required. In the case of the coordinatograph, an operator must further process the plots by deciding which sections of the rubylith are to remain as part of the mask and which are to be removed and then he manually removes the unwanted pieces. This step in mask making is computerized for the EBM.

The EBM postprocessor must interpret the XYMASK output file to determine which points are inside or outside the periphery of each feature. The EBM postprocessor converts the XYMASK output file data into sets of left-hand and right-hand boundaries whose minimum and maximum Y coordinates, when connected, are parallel to the X axis. The more nonconvex the feature, the more difficult the task becomes.

Since the EBM is a random-access plotter, the postprocessor processes one path or polygon at a time before proceeding to the next feature on the XYMASK output file. The data for a polygon are stored as a linked list in the *1 program. The program determines the lower left-hand and upper right-hand points by comparing the coordinates contained in the list. From this, two routes along the periphery are

established, which eventually yield sets of left-hand and right-hand boundaries. The actual structure of the list-processing algorithm is too complex to be described here in detail.

One of the unique features of the EBM postprocessor is the interpretation of paths. As mentioned above, a path is described on the XYMASK output file as a centerline and a path width normal to the centerline. Postprocessors for drawing devices such as the coordinatograph must translate this path information into a polygon before the feature can be plotted. In other words, the postprocessor must find the periphery points for the path. The EBM postprocessor takes advantage of the form of the output file data by treating the path as the figure formed when a circular tool, having the path width as the diameter, is moved along the centerline. Rather than converting the path into polygon data and then processing the resulting polygon, the postprocessor passes the major portion of path processing onto the EBM's control computer. The description of the control computer algorithms, which follows, will explain how this data is handled.

3.3 *EBM Control Computer Algorithm*

The control computer is capable of calculating the boundary and outline points in less time than it takes for the EBM to draw fill-lines. Thereby, the interface and EBM become the limiting factors in allowing the pattern generator to maintain an average pattern drawing time of one microsecond per addressable point for a significant set of masks. The calculations of the endpoints of fill-lines along the left-hand and right-hand boundaries are based on integer arithmetic.[6] The following example of straight-line-to-arc boundaries illustrates the use of integer arithmetic in this application.

Consider a set of boundaries consisting of the straight line $Y = Y = (A/B)X$ and the circular arc $X^2 + Y^2 = R^2$. The constants A, B, and R^2 are integers calculated from the control computer input data.

In integer arithmetic, the straight line is redefined as:

$$F = BY - AX \tag{1}$$

where F represents a third dimension. Thus, the straight line can be considered as the intersection of two planes in F space, with equation (1) defining one plane, and the XY plane the other.

The introduction of the dimension F results in the following useful properties:

(*i*) F is zero on the straight line and has opposite signs for points X, Y on opposite sides of the straight line.
(*ii*) There exists a single value of F for each point in the XY plane.
(*iii*) F is an integer for all integer points X, Y.
(*iv*) There is no error in a sequence of integer solutions for F.
(*v*) The smallest integer number is 1. If this is the smallest addressable unit in the graphic field, then all points X, Y that are within 1 unit of the true solution represent the true solution in the XY plane.

These properties of F make it easy to form an algorithm for calculating integer points along a straight line. If equation (1) is evaluated at the point (0,0), then the resultant F is 0. Rather than evaluate equation (1) for F at all points, it is easier to calculate a change in F between adjacent integer points. The adjacent integer points (1,0), (0,1), and (1,1), (in the neighborhood of the straight line) have the integer F values of $-A$, $+B$ and $B - A$ respectively. According to property (*i*) the point (0,0) is on the straight line and the points (1,0) and (0,1) are on opposite sides of the line. According to properties (*ii*) and (*iv*), a step-by-step calculation of F values from the point (0,0) to (1,1) will result in the identical F value at the (1,1) point regardless of the steps taken en route. Choosing a sequence of points with the smallest F values guarantees that the points are as close to the straight lines as the address structure of the field allows.

According to property (*v*), there may exist several integer values of X and Y that represent the true solution point of the straight line. This observation is used to form a more practical algorithm where only one addition and one test for sign of F per point is required to find the next integer point along the straight line.

A circular arc is the other boundary considered in the example. The circular arc is redefined in integer arithmetic as

$$F = X^2 + Y^2 - R^2 \tag{2}$$

where again, F represents an added dimension. The circular arc is thus formed in F space by the intersection of the XY plane with a parabaloid. The properties (*i*) through (*v*) also hold true for equation (2).

A sequence of integer points along a circle is computed by taking unit increments parallel to either the X or Y axis and computing the resultant F values; for a change of 1 addressable unit in the X direction, F changes by $2X + 1$. The corresponding computation is shift left, increment, and add. A test of sign of F determines whether the

next step increments X again or decrements Y. The coordinates thus generated are located along the circular arc and form the mask-feature boundary points.

It is also possible to construct an integer arithmetic algorithm to compute points along the outline of a path. According to the path definition, points on each side of the outline represent the envelope generated by a circle moving along the centerline as illustrated in Fig. 3.

The path algorithm finds points along the outline by choosing points along the circle until the normal to the circle is aligned with the normal to the path centerline. The circle is then displaced along the path centerline and the above process is repeated. A separate but identical algorithm is used for finding points on the opposite outline. Fill-lines are drawn parallel to one coordinate axis between these points. While the above algorithm appears to be complicated, surprisingly few calculations are required to find the endpoints of the fill-lines. For example, the slope of a curve in the XY plane is given by the ratio of change of F for changes in the X and Y directions, where the change of F in both directions is already available from the straight-line and circular-arc algorithms. The normal to a curve is the negative inverse of the slope, and thus the only additional computation required in the path algorithm is the comparison of a sequence of two integer ratios.

As is evident from the above discussion, only a few instructions

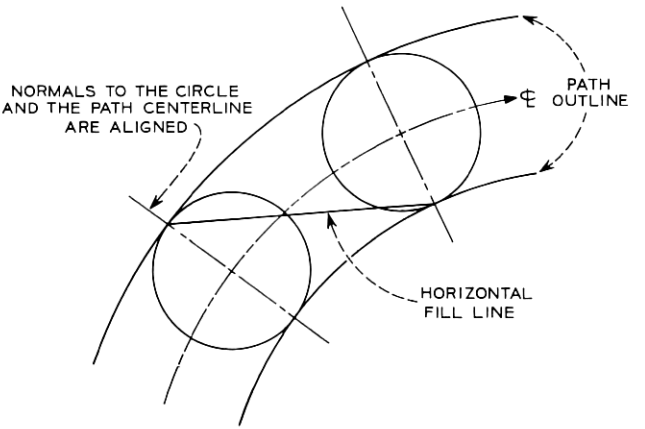

Fig. 3—Construction of a path.

are required in the control computer to calculate the endpoints of a fill line between a set of boundaries. As a result of the redefinition of the problem in integer arithmetic, the calculations in most instances are completed before fill-line generation is finished allowing the EBM pattern generator to maintain the one microsecond per addressable point drawing speed.

The postprocessor execution time varies with the complexity of the mask being generated but to a lesser degree than for the PPG postprocessor. Several minutes on an IBM 360/65 are ordinarily required for a typical interconnection mask.

IV. DISCUSSION

Several computer systems used in the generation of integrated-circuit masks have been described in the preceding sections. The first sections dealt with the XYMASK system which links the circuit designer to the mask-fabrication process. XYMASK provides a computer-independent language for describing the mask configurations, and produces either outline drawings or mask artwork on one or more of a number of different graphical output devices. The majority of all Bell Laboratories and Western Electric Company masks are produced using the XYMASK system.

The next two sections described XYMASK subsystems which generate artwork on the PPG and EBM. These two plotters fundamentally differ in that the first uses a raster-scan technique, while the second is a random-access device. Each is supported by a dedicated control computer. The subsystem descriptions indicate a degree of similarity in postprocessor functions, but different approaches toward the division of the necessary computation between the postprocessor and the control computer.

V. ACKNOWLEDGMENTS

Many persons have contributed to the development of both the XYMASK system and the PPG postprocessor. The efforts of V. A. Fasciano, B. R. Fowler and S. Pardee are singled out as being of particular importance in system design as well as implementation.

We would also like to acknowledge J. E. Gorman, A. D. Janzen, R. Sedgewick, and C. C. Wyckoff for contributions to the EBM control computer program development.

REFERENCES

1. Fowler, B. R., "XYMASK," Bell Laboratories Record, *47*, No. 6 (July 1969), pp. 204–209.
2. Christensen, C., and Pinson, E. N., "Multi-function Graphics for Large Computer System," American Federation of Information Processing Societies (AFIPS), Conference Proceedings, 1967 Fall Joint Computer Conference.
3. Dowd, P. G., Cowan, M. J., Rosenfeld, P. E., and Zacharias, A., "The Primary Pattern Generator: Part III—The Control System," B.S.T.J., this issue, pp. 2061–2067.
4. Samaroo, W., Raamot, J., Parry, P., and Robertson, G., "The Electron Beam Pattern Generator," B.S.T.J., this issue, pp. 2077–2094.
5. Newell, A., Early, J., and Haney, F., **1 Manual,* Carnegie Institute of Technology, Pittsburgh, Pennsylvania, June 26, 1967, Advanced Research Projects Agency No. SD-146.
6. Gorman, J. E., and Raamot, J., "Integer Arithmetic Technique for Digital Control Computers," Computer Design, *9*, No. 7 (July 1970), pp. 51–57.