METHODS FOR HIERARCHICAL AUTOMATIC LAYOUT OF CUSTOM LSI CIRCUIT MASKS*

B. T. Preas and C. W. Gwyn
Sandia Laboratories
Albuquerque, NM 87185

ABSTRACT

A new automatic IC mask layout code is described which avoids most of the problems inherent in the present generation of layout codes such as lack of flexibility, inefficient use of area, and restricted design complexity. The structured hierarchical layout approach, construction graphs, and placement and routing algorithms are outlined.

INTRODUCTION

Computer aids have been used for a number of years in the design and layout of custom integrated circuit (IC) masks (Refs. 1-5). These aids range from the use of interactive graphics systems to sets of programs designed to run in the batch mode on a large computer. Semiconductor manufacturers designing IC's with large production volumes have usually employed manual layout methods and have restricted the use of computer aids to digitizing and editing in the final phase of the design sequence. Although this approach usually leads to high design costs and long design cycles, the final circuit can be inexpensively manufactured because of the small chip size. An intermediate approach is the semiautomatic design aid. Typically, these programs require manual assistance to complete an IC design and have placement and routing algorithms which can be invoked by the user for sections of the total layout problem. At the opposite end of the design spectrum, designers involved with low production volumes have often relied on automatic design aids which produce circuit masks based on the use of a predefined library of circuit modules or cells and a circuit connection description. With the increasing emphasis on the use of custom IC's for low volume applications, the use of computer aids for the automatic layout of IC masks becomes increasingly important in minimizing the design cycle and development cost.

For circuits containing approximately 500 gates and few restrictions regarding chip area, several automatic IC layout programs can rapidly produce error free designs at low cost. However, there are a number of deficiencies in the present codes. These include: lack of flexibility, inefficient use of silicon area, and limitation in circuit complexity. The mask layouts produced using computer aids are characterized by the use of standard cells placed in rows. This ordered arrangement does not allow easy incorporation of design requirements such as special cell or pad placement or use of special macrocells such as ROMs and RAMs. These non-standard layouts usually require the manual addition and interconnection of the special cells using an interactive graphics system which is time consuming and can result in the introduction of errors. The inefficient use of silicon area produced by the regularity and generality required in cell design and the restriction of cell placement to rows can lead to long interconnection distances and use of additional silicon area. Another important deficiency is the limitation on chip complexity imposed by the IC layout programs. The present limit is on the order of 1000 equivalent gates. With the continuing decrease in design rule tolerances made possible by improved circuit fabrication methods, the production of Very Large Scale Integrated (VLSI) circuits in the range of 10,000 to 40,000 equivalent gates is possible and the present design aids and methods are becoming outdated. The present layout codes also tend to inhibit the design process rather than encouraging successive refinement in a design. This does not prevent the automatic layout of a low complexity chip but may prevent the successful design of a VLSI circuit.

The goal for the new automatic layout code, SICLOPS[6] (Sandia Integrated Circuit Layout Program System) is to avoid the problems noted above, provide flexibility for designing IC's using a variety of different technologies, and provide the basis necessary to design highly complex chips bordering on VLSI complexity. The code has been designed to provide automatic circuit layout with manual override to provide various degrees of user interaction with the layout process and to run within the batch mode for a variety of computer systems.

This paper outlines the important algorithms used in the SICLOPS code and is divided into three major sections. The first section describes the architecture and building blocks for the layout program. The second section describes layout methods using standard cells and the third section describes the general cell assembly design.

STRUCTURED DESIGN ARCHITECTURE

To achieve the goals outlined in the previous section and to provide the capability for designing highly complex chips, an ordered design process must be introduced. Circuits containing tens of thousands of equivalent gates are several orders of magnitude larger than circuits which can be easily comprehended. For circuits of this size, design techniques are required to reduce the level of detail considered during each step of the design to a manageable size. In drawing an analogy between the design of a large software program and the layout of the VLSI circuit, structured design methods must be used (Refs. 7-9). For example, hierarchical decomposition or partitioning must be used to reduce the system function into tractable module sizes. Basically, this method reduces to top-down circuit partitioning and initial layout and a bottom-up circuit layout implementation at each level of hierarchy. By considering only the subfunction to be implemented and interface interconnections, the degree of circuit complexity for each module design can be reduced to a level consistent with the available design capability.

Modular Layout Approach

Although the present IC layout programs are generally restricted to the use of standard cell building blocks placed in rows, the SICLOPS layout philosophy

*This work was supported by the U.S. Dept. of Energy, Contract AT(29-1)-789.

departs from this rigid order and approachs the flexibility provided by manual chip layout where module placement is not restricted to rows or columns. This approach allows arbitrary size rectangular modules to be placed in an optimum layout configuration and allows module interconnection via a number of irregularly shaped interconnection regions.

One of the key generalizations incorporated in SICLOPS is the recursive use of modules to solve the problems of design complexity and actual design process. In the context used in SICLOPS, a module usually performs a specific circuit function and is defined in terms of lower level modules and interconnection regions. The modules are nested to implement a design of any complexity. By considering only the function to be implemented and the sub functions along with interface connections, the degree of complexity during each module design can be reduced to a level consistent with the available design capability and designer understanding. This capability permits many of the same algorithms to be used at each recursion level and encourages the designer to use an orderly design process such as hierarchical decomposition and successive layout approximations.

Building Blocks

A number of fundamental building blocks and assemblies are required to implement a general IC layout. For the fundamental blocks consisting of standard cells, macro cells, and bonding pads, the layout program requires only the outline dimensions and locations of the input and output connections. Fundamental blocks are grouped together by the program and interconnected to form assemblies.

One fundamental block is the standard cell. Standard cells are rectangular and may have input and output connections on one or opposite sides. The cells are uniform in height and vary in width as required to implement the required circuit function. For cells with connections on two sides, an individual connection may, but need not, be repeated on both connection sides. Power and phase connections are routed to the ends of the cell row by the layout program; the distribution of these signals in the interior cells of a row is provided by the cell design.

Standard cells are grouped together and interconnected to form a standard cell assembly. One or more cell rows are contained within the assembly and in general, each assembly contains input/output terminals on all four sides of the module. For an assembly consisting of a single row of cells, the position of the terminals is determined by the cell placement in the row. For assemblies containing several rows of cells, the terminals contained on the sides parallel to the cell rows are determined by the terminals on the outer cell rows. For the other two sides, the terminals are located at the termination of the tracks in the interconnection routing channels. In general, an assembly is rectangular in shape.

Bonding pads are special standard cells. The bonding pad cells range from simple metal shapes used to provide external connections to the chip circuit to complex input pads containing static charge protection and output pads containing buffer devices to provide the necessary signal drive for use in circuitry external to the IC.

Bonding pads are grouped together in rows to form a bonding pad assembly. The number of bonding pad assemblies placed end to end on the side of a chip is limited only by the physical dimension of the chip. The spacing between pads may be varied as required to match the external package bonding requirements.

Another fundamental cell is the macro cell. Typically, the macrocells consists of a custom layout (usually handcrafted) designed to perform a specific function. These cells may have an arbitrary rectangular size and shape and may contain input/output connections on one to four sides. Power and phase signals must be provided as part of the standard interconnection specification for the larger assembly containing the macrocell.

General cell assemblies are the highest level blocks supported by the layout code. The general cell assembly may consist of combinations of standard cell assemblies, macrocells, standard cells, or lower level general cell assemblies. Use of the general cell assembly in a recursive sequence allows very large assemblies to be defined.

An IC chip is a complete circuit layout. The chip contains a general cell assembly, one or more bonding pad assemblies and a number of chip support cells and entities such as masking borders, alignment keys, test structures and identification markings. All of these chip support functions are placed by the layout code.

The SICLOPS input language used to describe the above modules is outlined in Reference 6.

Examples of Structured Layout

A general cell assembly is shown in Figure 1. Each of the modules in the assembly can correspond to macrocells, standard cell assemblies or other smaller general cell assemblies. In general, terminals may be located on all four sides of each module. The modules can assume any orientation within the assembly as determined by the placement algorithms. The routing channels between modules may be input by the user or determined by the program and are used in constructing position and channel intersection graphs described later.

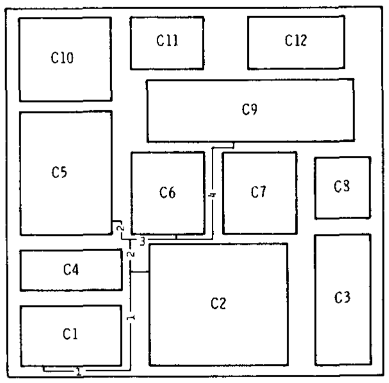


Figure 1. General Cell Assembly Containing 12 Modules. One net has been routed as shown.

A complete layout for a chip, following the structured format, is shown in Figure 2. This chip contains one general cell assembly, several bonding pad assemblies and the associated support entities. The general cell assembly, in turn, contains a smaller general cell assembly (containing a macrocell and two standard cell assemblies), a standard cell assembly, and a macrocell. The nesting of assemblies can be implemented, theoretically, to any depth, thus permitting the design of an arbitrarily complex chip.

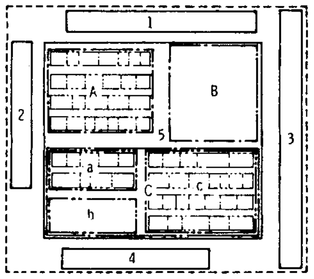


Figure 2. Entire Layout for an IC. The layout contains one general cell assembly and four bonding pad assemblies.

In each of the above designs, the designers can interface with the layout program at a variety of levels. At the most detailed level of interaction, the designer may implement the logic in several blocks of custom macrocells and use the computer aids to perform the interconnection routing. For this problem, the designer specifies the interconnection and approximate size of the macrocells and makes several SICLOPS runs to determine the best shape for the chip and the approximate placement of the macrocells. After determining the optimum shapes for the macrocells, the cells can be designed on an interactive graphics system, and a final SICLOPS run executed to incorporate the macrocells into the complete chip layout.

For extremely complex chip layouts such as those encountered in VLSI circuits, the designer uses recursive hierarchical decomposition to reduce the system to small modules. For example, the designer partitions the total system or circuit into first level subsystems, specifies the interconnection of the subsystems and makes several SICLOPS runs to obtain the general placement and shape for each subsystem. The subsystems are, in turn, partitioned into smaller modules until the entire system is defined in terms of either standard cells or macrocells. At this point, a bottom up implementation process can be used, since block sizes and input/output connection locations are known. The chip design is complete when the top level is reached.

Structure Tree

The implementation of the hierearchical structure in the design of an IC is accomplished using a structure tree. A structure tree for the chip layout in Figure 2 is shown in Figure 3. The largest module being designed is represented as the trunk and successive submodules are represented as smaller and smaller branches. For an IC chip, the trunk represents the entire chip while the branches are denoted by general cell assemblies, macrocells, or standard cell assemblies.

STANDARD CELL ASSEMBLY IMPLEMENTATION

The standard cell assembly in SICLOPS is composed of standard cells placed in rows with routing surfaces between the rows. The placement algorithm for standard cells uses constructive initial placement followed by an iterative improvement step. Routing is performed using channel routers. Although this approach is typically used by the current generation of IC layout codes, several new features are incorporated in SICLOPS. These features provide a high degree of user control over cell placement for an automatic code, extension of channel routing algorithms, and use a figure of merit to optimize the layout based on module width and height.

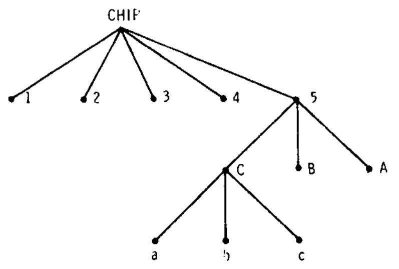


Figure 3. Structure Tree for IC Chip of Figure 2.

Cell Placement

The SICLOPS placement algorithm allows the user to specify (1) inital cell positions (suggestion to algorithm for a starting placement), (2) final positions (corresponding to traditional fixed placement), or (3) permissible locations for cells using logical FORTRAN statements. The logical FORTRAN statement is TRUE for a permissible cell location and FALSE otherwise. The following functions may be used: ROWOF (cell) returns the row of the argument cell; POSOF (cell) returns the position of the argument cell within a row; and ORENOF(cell) returns the orientation of the argument cell.

The following example illustrates the flexibility provided to allow the designer to control placement of cells in an assembly and consists of a three-bit shift register which is to be included in a standard cell assembly (shown in Figure 4). The three-shift register cells do not need to be placed side by side in the module but should be placed on the same row and in sequential order. Figure 4 lists the input restrictions and shows the relative cell placement. For each of the three shift register cells, the first line of input information indicates the cell name followed by output and input signal names. The next two lines indicate logical restrictions to be satisfied during cell placement. In each case, the logical variables THROW and THPOS are true when the cells are in the allowed locations. Because of the symmetrical wiring for the shift register, other placement algorithms based on minimizing wire length, track count, or area may not provide the same layout.

```
SRBIT1     DFLPFP  SRINP  SHFT1  SRCLK
           THSROW = ROWOF(SRBIT1) .EQ. ROWOF(SRBIT2)
           THSPOS = POSOF(SRBIT1) .LT. POSOF(SRBIT2)

SRBIT2     DFLPFP  SHFT1  SHFT2  SRCLK
           THSROW = ROWOF(SRBIT1) .EQ. ROWOF(SRBIT2)
           THSPOS = POSOF(SRBIT1) .LT. POSOF(SRBIT2)
                    .AND. POSOF(SRBIT2) .LT. POSOF(SRBIT3)

SRBIT3     DFLPFP  SHFT2  SROUT  SRCLK
           THSROW = ROWOF(SRBIT1) .EQ. ROWOF(SRBIT3)
           THSPOS = POSOF(SRBIT2) .LT. ROWOF(SRBIT3)
```

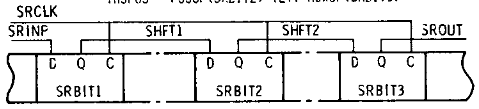


Figure 4. Shift Register Layout Using Standard Cells. The logical FORTRAN statements are satisfied to produce the layout shown in the lower portion of the figure.

Routing

Many standard channel routing algorithms (Ref. 10-12) are used in the program. However, since the program is automatic, manual intervention to break constraint loops which prevent the completion of routing paths is not available and the algorithms have been modified to resolve constraint loops. Two techniques are used to break constraint loops: (1) doglegs or jogs are permitted at points intermediate to the cell pin loations, and (2) the routing is not restricted to lie within the area specified by the rectangle defined by the two outside pin locations for the interconnection net. In some cases, the constraint loops are broken by increasing the number of tracks in the channel.

The routing algorithms also permit the use of non-uniform channel widths and the use of wide vias to provide connections between different interconnect levels. Non-uniform channels usually result because of non-standard height cells or irregular boundaries between modules or assemblies of different size.

Figure of Merit

A new figure of merit is used in placement based on extensions of previous techniques (Ref. 10). During the iterative improvement step, a new placement is accepted only if the total standard cell assembly area is decreased. The area is calculated as follows:

Area = height x width

$$\text{height} = \sum_{i=1}^{\text{nrows}} \text{row height } i$$

$$+ \sum_{i+1}^{\text{nrows} + 1} \begin{pmatrix}\text{max track density}\\ \text{for channel i}\end{pmatrix} \times \begin{pmatrix}\text{track}\\ \text{height}\end{pmatrix}$$

width = max (row width i) + $\left[(\text{max left side track density}) + (\text{max right side track density})\right]$ x track width.

To obtain standard cell assemblies with a specified aspect ratio, restrictions can be introduced in the above calculation.

GENERAL CELL ASSEMBLY IMPLEMENTATION

A general cell assembly is a rectangular block of circuitry composed of subassemblies or blocks separated by routing channels. SICLOPS automatically places the block, determines and positions the channels, and routes the interconnections.

Design Representation

Three graphs are used to represent a general cell assembly; these graphs are similar to those described in Reference 5 and consist of a channel intersection graph, a horizontal channel position graph and a vertical channel position graph.

The channel intersection graph defines the topology of the general cell assembly. This graph represents the non-uniform rectangular grid on which interconnection routing is performed. Each node of the graph is the intersection of two channels and each arc represents a self-contained routing channel or 'sub-channel.' Weights are assigned to the arcs of the graph to accomplish the loose routing described below. Figure 5 shows the channel intersection graph for the general cell assembly shown in Figure 1.

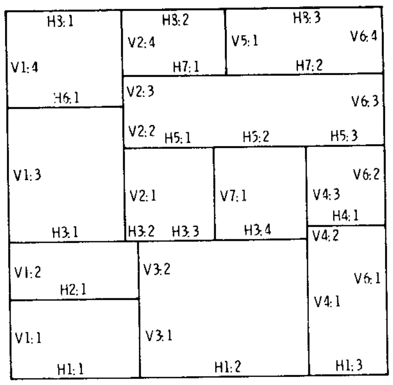
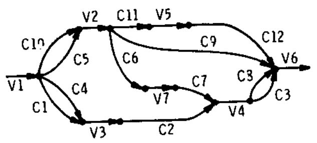
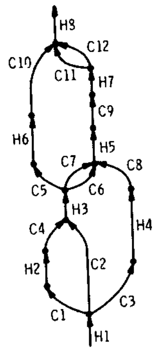


Figure 5. Channel Intersection Graph for the General Cell Assembly Shown in Figure 1.

The horizontal channel position graph and the vertical channel position graph are directed graphs defining the positional constraints of the channels in relation to each other. A node on these graphs represents the boundary between a block and a channel and an arc corresponds to a physical dimension on the layout. Numerical values associated with each arc are weights representing the width of a routing channel or the dimension of a block. The width and height of the general cell assembly may be obtained by finding the longest path through the horizontal and vertical channel position graphs respectively. Critical channels and critical blocks are arcs of these graphs that lie on any longest path. The channel position graphs for the general cell assembly shown in Figure 1 are outlined in Figure 6.



(a) Horizontal Channel Positon Graph



(b) Vertical Channel Position Graph

Figure 6. Channel Position Graphs for the General Cell Assembly Shown in Figure 1.

Placement

The solution to the placement problem consists of determining the locations for blocks of arbitrary size and shape which will minimize the interconnection area and the area within a rectangle enclosing the blocks. The placement algorithm consists of a constructive initial placement phase and an iterative improvement phase. The initial positions of both the inner blocks and the channels are floating to avoid channel capacity constraints. During placement, channel definitions corresponding to the selected module placement are defined.

The placement algorithms have been designed to provide the following four levels of interfacing with the designer:

1. Completely free placement--The placement algorithm determines an initial starting placement, and then iteratively improves the placement.

2. User defined initial placement--The placement algorithm iteratively improes the initial layout.

3. User defined final placement--The relationship of the modules is fixed by the designer and the placement function simply determines the channel positions.

4. User defined final placement and the channel definitions--The layout is rigidly specified and the placement algorithm simply converts the input data into internal form.

Constructive initial placement within an assembly consists of the following steps. The size of the general cell assembly is estimated based on the size of the modules and used to construct a temporary outline of the general cell assembly. All input/output terminals with locations specified are placed around the periphery of this temporary outline. The modules with highest connectivity to the placed input/output terminals are located first as closely as possible to the fixed terminals. The remaining modules are selected one at a time based on highest connectivity to the modules and input/output terminals already placed and added as close to the placed modules as possible. Holes left within the area of placed cells are filled if possible.

Once an initial placement is found, the next step determines the location of the channels corresponding to the given placement. The first step consists of surrounding each module and the entire general cell assembly with channels. Channels are combined which overlap in the direction parallel to the channels. Any channels intersecting the combined channels are either stretched or shrunk to maintain an intersection with the combined channel. The four channels originally surrounding the entire general cell assembly assure the channels are completely connected after the combining process is finished. Once the channels and their intersections have been identified, the three graphs described earlier are constructed.

The iterative improvement phase consists of a loop which attempts to improve the current placement. The first step identifies the critical modules (modules corresponding to arcs on a longest path through either channel position graph). Next, each of the critical modules is input to orientation, movement and exchange improvement operations. These operations search for an improved placement by altering the position of the critical modules.

The orientation improvement operation tries to improve the placement by altering the rotation or reflection of the given module. This step modifies the arc weights corresponding to this module in the channel position graphs.

The movement operation searches for a new vertical or horizontal position for the module near other temporarily fixed modules. This operation has the effect of deleting the initial arcs corresponding to the moved module in channel position graphs, and creating a series of arcs in one channel position graph and parallel arcs in the other channel position graph from a single arc at the target position.

The improvement operation searches for an exchange partner for the specified critical module. This operation alters the weights of the arcs corresponding to both modules but does not alter the topology of the channel position graph.

If any of these operations successfully determine a placement with a smaller area, the change is accepted, the new general cell assembly area is determined, and this new placement is used as the basis for further improvement operations. Figure 7 outlines the placement algorithm.

```
PLACE
   INITIAL PLACE
      ESTIMATE SIZE OF GENERAL CELL ASSEMBLY
      PLACE FIXED I/O PINS
      FOR EACH CELL DO
         FIND MAXIMALLY CONNECTED CELL
         PLACE THIS CELL
         END DO
      END INITIAL PLACE
   PLACE IMPROVEMENT
      FIND CHANNELS, DETERMINE SIZE, IDENTIFY CRITICAL
                  MODULES AND CHANNELS
      ESTIMATE TRACK DENSITY FOR INTERCONNECTIONS
      REPEAT UNTIL STOPPING CONDITION DO
         FOR EACH CRITICAL MODULE DO
            SEARCH FOR ALTERNATE ORIENTATION
            SEARCH FOR ALTERNATE LOCATION (MOVE)
            SEARCH FOR EXCHANGE PARTNER
            IF GENERAL CELL ASSEMBLY IS SMALLER DO
               MAKE THIS PLACEMENT CORRECT
               BREAK
               END DO
            ELSE DO
               RESTORE PREVIOUS PLACEMENT
               END DO
            END REPEAT
      END PLACE IMPROVEMENT
   END PLACE
```

Figure 7. Placement Algorithm for Locating Modules in a General Cell Assembly.

Routing

Several features combine to place very severe requirements on the routing function in the general cell assembly. First, the program must work automatically, since there is no interaction at this level in the design sequence. Second, 100% routing completion is a requirement. This is especially important in VLSI where there are a large number of interconnection nets. In fact, this requirement separates the IC layout and the printed circuit board layout problem. For printed circuit boards, the board area is specified and the objective is to optimze the completion rate. For IC's, the completion rate is specified (100%) and the objective is to minimize the area.

Many techniques are available for routing. The most important methods fall into two classes. Search methods (primarily derivations of the Lee and

Hightower methods) attempt to serially connect nets to build up all interconnections. In general, these methods suffer from completion problems because previous interconnections can block a succeeding one. Methods have been developed to enhance completion rates; these methods in general, consist of rip-up and re-route and adding extra tracks (Refs. 13 & 14). For complex topologies, 100% route completion is difficult to obtain and the result can be wasted area because there is no global optimization step. Channel routing techniques have been successfully used in systems which require 100% completion; however, the cost is a restriction in layout topology. Layouts using channel routers have taken the form of modules placed in horizontal rows and vertical columns.

The routing approach used for a general cell assembly finds a loose route specification for all interconnection nets and then assigns individual tracks using channel routing techniques. This two-step procedure approximates a parallel router.

The loose routing phase finds a strategic route on the subchannel intersection graph for each interconnection net and includes an explicit optimization step to reduce area. Further, since nets are assigned to channels, but not to specific tracks, rip-up and re-route is easy. The channels have infinite track capacity and 100% route completion is guaranteed. Loose route determination is a complex problem for a general cell assembly, since there are usually many acceptable ways to route a net. However, the best path cannot be determined without considering the interaction with other nets. The longest route in geometric length may not add any incremental area to the chip because it does not pass through a position of maximum track density on a critical channel.

The loose routing phase uses the subchannel intersection graph. Weights are assigned to the arcs of this graph to indicate the incremental area that would be added to the general cell assembly if an additional track was used in the subchannel associated with the arc. All noncritical subchannels and subchannels of a critical channel which do not contain a position of maximum track density are assigned a weight of zero because no area would be added to the general cell assembly if a track on this subchannel was used. Subchannels of critical channels that contain a position of maximum track density are assigned a large weight because an extra track would add to the area of the general cell assembly. Pins of the net are added as nodes to the subchannel intersection graph and the weights are adjusted for the affected subchannel arcs. The remaining problem consists of determining an interconnection tree which connects the nodes corresponding to the pins of the given interconnection net to minimize the sum of the weights on the arcs used.

Implementation of the loose routing algorithm uses a constructive initial solution followed by iterative improvement. The constructive routing step locates an initial route for each interconnection net by: (1) selecting the net to be routed from a selected subset, (2) determining the weights for the subchannel intersection graph, and (3) adding pins to the subchannel intersection graph as described above. The pins for each net are connected as a sequence of two point paths using shortest path algorithms. When all nets of the current set have been routed, a new set is selected and the process is repeated until an initial route has been found for all nets. Iterative improvement is accomplished as follows: First, the size of the general cell array is determined and the critical channels are identified using the channel position graphs. Weights are assigned to the arcs of the subchannel intersection graph and a segment of a net is identified which contributes to the size of the general cell array. The identified segment is removed from the data array and the remaining portion of the net is added to the subchannel intersection graph with the connected vertices fused. The two parts of the net are then interconnected using a shortest path algorithm, assuming the other nets and area of the assembly are fixed. If an area reduction is produced, the new loose route for the net is accepted; otherwise, the old route is restored to the data array. The loose route algorithm is summarized in Figure 8.

```
LOOSE ROUTE DETERMINATION
  INITIAL LOOSE ROUTE
    FOR EACH INTERCONNECTION NET DO
      DETERMINE LOOSE ROUTE FOR THIS NET
      END DO
    END INITIAL LOOSE ROUTE
  LOOSE ROUTE IMPROVEMENT
    FIND SIZE AND CRITICAL CHANNELS
    REPEAT UNTIL STOPPING CONDITION DO
          FIND POSITIONS OF MAX TRACK DENSITY
          CHOOSE NET TO RE-ROUTE
          REMOVE NET FROM DATA
          ADD PINS OF THIS NET TO SUB-CHANNEL
                INTERSECTION GRAPH
          ASSIGN WEIGHTS TO SUB-CHANNEL INTERSECTION
                GRAPH
          FIND LOOSE ROUTE FOR THIS NET
          FIND SIZE AND CRITICAL CHANNELS WITH
                MODIFICATION
          IF SIZE IS SMALLER DO
                ACCEPT CHANGE TO THIS NET
                END DO
          ELSE DO
                REJECT CHANGE TO THIS NET
                END DO
          END REPEAT
    END LOOSE ROUTE IMPROVEMENT
  END LOOSE ROUTE DETERMINATION
```

Figure 8. Routing Algorithm for Determining Loose Routing Paths for Nets in a General Cell Assembly.

Following completion of loose route, the individual nets are assigned specific tracks within the channels. The extended channel routing discussed previously is used to complete the detailed routing within each channel.

SUMMARY

The methods employed in SICLOPS to design complex masks for integrated circuits have been described. The code is being implemented on a DEC 10 computer and has been designed to provide a structured approach for the layout of complex IC's with the flexibility to design IC mask layouts ranging from very simple circuits containing a small number of standard cells to very large IC's involving the nesting of large numbers of assemblies containing standard cells and macrocells. Since the system of programs is being implemented in stages, an entire chip design has not been completed. However, the algorithms used for placement and routing have been verified for typical general layout configurations.

Although specific improvements in the layout efficiency are difficult to quantify, the initial goal for the program is to generate a CMOS IC layout using a silicon gate technology which is approximately a factor of five higher in device density than a metal gate technology for combinational logic circuits. This improvement is produced by the combined effects of using new cell design concepts, improved design tolerances, and improved placement and routing algorithms.

ACKNOWLEDGMENT

The authors would like to thank Professors R. W. Dutton and W. M. vanCleemput of Stanford University for their helpful discussions and suggestions concerning the architecture and development of a general IC layout code and guidance during the initial phases of the code development.

REFERENCES

1. A. Feller, "Automatic Layout of Low-Cost Quick-Turnaround Random-Logic Custom LSI Devices," Proceedings of the Design Automation Conference, p. 79, June 1976.

2. R. L. Mattison, "Design Automation of MOS Artwork," Computer, p. 21, January 1974.

3. J.G.M. Klomp, "CAD for LSI-Production's Interest in Its Economics," 13th Design Automation Conference, June 1976.

4. G. Persky, D. N. Deutsch, and D. G. Schweikert, "LTX-A System for the Directed Automatic Design of LSI Circuits,"Proceedings of the Design Automation Conference, p. 399, June 1976.

5. H. Kato, H. Kawanishi, S. Goto, T. Oyamada, and K. Kani, "On Automated Wire Routing for Building-Block MOS LSI," Proceedings ISCAS, p. 309, April 1974.

6. B. T. Preas and C. W. Gwyn, "Architecture for Contemporary Computer Aids to Generate IC Mask Layouts," Eleventh Annual Asilomar Conference on Circuits, Systems, and Computers, November 1977.

7. C. W. Gwyn, "A View of Integrated Circuit Design," Conference on Industrial Integrated Circuit Design," California Institute of Technology, May 1977.

8. W. M. vanCleemput, "An Hierarchical Language for the Structural Description of the Digital Systems," Proceedings of the Design Automation Conference, p. 377, June 1977.

9. W. M. vanCleemput and E. Slutz, "InitialDesign Considerations for a Hierarchical IC Design System," Eleventh Annual Asilomar Conference on Circuits, Systems, and Computers, November 1977.

10. B. W. Kernighan, D. G. Schweikert, and G. Persky, "An Optimum Channel Routing Algorithm for Polycell Layouts of Integrated Circuits," Design Automation Workshop Proceedings, p. 50, June 1973.

11. T. Asano, T. Kitahashi, K. Tanaka, H. Horino, and T. Amano, "A Graph-Theoretical Approach to the Routing Problems," Electronics and Communications in Japan, Vol. 56-A, No. 12, 1973.

12. D. N. Deutsch, "A Dogleg Channel Router," Proceedings of the Design Automation Conference, p. 425, June 1976.

13. R. L. Mattison, "A High Quality, Cost Router for MOS/LSI," 9th Design Automation Workshop Proceedings, p. 94, June 1972.

14. F. Rubin, "An Iterative Technique for Printed Wire Routing," 11th Design Automation Workshop Proceedings, p. 308, June 1974.