THE BERKELEY BUILDING-BLOCK LAYOUT SYSTEM FOR VLSI DESIGN

by

N. P. Chen, C. P. Hsu and E. S. Kuh

Memorandum No. UCB/ERL M83/10

14 February 1983

(cover)

THE BERKELEY BUILDING-BLOCK LAYOUT SYSTEM FOR VLSI DESIGN

by

N.P. Chen, C.P. Hsu and E.S. Kuh

Memorandum No. UCB/ERL M83/10

14 February 1983

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

The Berkeley Building-Block Layout System for VLSI Design*

N. P. Chen, C. P. Hsu and E. S. Kuh

Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley, California 94720, USA
Telephone: (415) 642-2689

Abstract**

Automatic layout design of custom VLSI circuits depends on the building-block hierarchical approach in which macrocells of arbitrary shapes and sizes are given together with the net list. Although many building-block layout systems have been proposed in recent years, almost none is in production use. Our aim is to design an intelligent and practical automatic layout system (BBL) which will interphase with other design aids at Berkeley. The BBL system has a general purpose database, a smart global router which can dynamically adjust placement and efficient detailed routers, namely: the channel router and the switch-box router.

The System incorporates several novel ideas and is based on a number of graph-theoretical algorithms. Experimental results indicate that the System is extremely effective. An example from AMI with 33 functional blocks, 132 nets and 440 pins has been tested. It takes 5.5 minutes cpu time and 2.5 megabytes on a VAX 11/780 under the Berkeley UNIX operating system. The chip size is about 8.5% less than the original AMI layout.

*Research sponsored by National Science Foundation Grant ECS-8201580 and the Air Force Office of Scientific Research (AFSC) United States Air Force Contract F49620-79-C-0178.

**If accepted, the paper will be presented by one of the authors at the Conference.

The Berkeley Building-Block Layout System for VLSI Design*

N. P. Chen, C. P. Hsu and E. S. Kuh

Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley, California 94720, USA
Telephone: (415) 642-2689

1. Introduction**

The present status of automatic layout design for integrated circuits is limited to the gate array and the polycell approach. Although papers have been written describing automatic hierarchical, building-block layout [1-10], to the best knowledge of the authors no such system has been adopted for production use. One main reason is that in comparison with manual design the chip size is anywhere between 30% and 50% larger. Needless to say automatic layout for future custom chips design is essential. Our present approach is sufficiently different in philosophy from others, and we believe that our System will compete favorably with manual design in terms of area usage.

The BBL System is a hierarchical, automatic layout system for IC design. It allows component modules with rectilinear boundary of arbitrary size as building blocks. A software configuration of the BBL System is shown in Fig. 1. The programs of the System are classified into the following categories: input/output processors, placement

*Research sponsored by National Science Foundation Grant ECS-8201580 and the Air Force Office of Scientific Research (AFSC) United States Air Force Contract F49620-79-C-0178.

**If accepted, the paper will be presented by one of the authors at the Conference.

programs, global routing programs, detailed routing programs, interactive editors, and display utilities. The database is constructed in such a way that different placements, feedthrough assignments and routing algorithms can be implemented and tested by using the BBL System. The users can input the initial configuration either by a text file or by using the interactive graphics editors developed at the University of California, Berkeley.

A constructive placement program is under development and is not reported in this paper. The automatic routing system takes relative placement and the signal netlist as input. The signal netlist specifies all terminals to be connected. The current version of the system assumes two layers available for routing and will not route signals over the blocks. In the system, a prerouting analysis is carried out first. It estimates the expected routing density around each block, and routing space is then allocated accordingly. The routing order of signals is determined by the approximate routing space available for each signal. The global routing is formulated in terms of the problem of Steiner tree on graphs, and signals are then routed one by one.

The System will next define regions for detailed routing and call the channel router and the switch-box router to do the track assignment [11,12]. Finally, the global router will output all the routing information to the database. The flowchart is shown in Fig. 2. The System has the ability to move blocks at any moment throughout the routing process. Thus, better placement, hence better control of distribution of wires on the layout plane is achieved. This capability of moving blocks dynamically is one of the key features of the System and is missing in almost all other existing building-block layout systems.

2. The Database

A general purpose database is used in the BBL System. It is based on a hierarchical structure and serves as both the input and the output for any placement and routing programs. Logical signal information and geometrical layout descriptions of the entire chip are represented hierarchically according to the user specified hierarchy.

The geometrical information is represented by modules (blocks). Each module may be a functional block or a circuit component. The boundary of a module can be any rectilinear polygon, and a module may have any number of different boundaries on different layers (sometimes called protection frames [13]). Different modules may have the same geometry which is represented by the same geometric structure. A cell library may be created for different geometric structures. Two lists of terminals are associated with a module, one for logical terminals and the other for physical terminals. Electrical equivalence and logical equivalence information of terminals are properly represented. The interconnection information is provided by a netlist. Every net is hierarchically decomposed into a set of signals, where every signal represents the net information on one level of the hierarchy. A sophisticated structure is used for routing. It has the ability to represent any routing results based on straight line connections.

The communication between the BBL System and other software tools developed at the University of California, Berkeley is established through this database.

3. Placement Adjustment and Global Routing

The principal concept involved is the creation of a set of "bottlenecks" pertaining to the relative placement. The complete set of "bottlenecks" gives the critical regions of the routing plane where congestion of routing is most likely to occur and hence needs to be carefully controlled. It also serves as a link between blocks whereby all information is easily updated when some blocks must move to a new position. A "bottleneck" is created in the routing region between two neighboring blocks if, and only if, there exists no other block between them. The bottleneck thus becomes a data structure which stores current information on routing and geometry of blocks. The "active region" of a bottleneck is defined as the common region between the two parallel edges of the neighboring blocks. In Fig. 3 we give an illustration of a chip which contains four blocks. Horizontal and vertical bottlenecks are marked by dash lines. A bottleneck is called the horizontal type if it exists between two horizontal edges and is called the vertical type if it exists between two vertical edges. In this example there are altogether seven horizontal-type bottlenecks and seven vertical-type bottlenecks.

Each edge of a block has a pointer to the bottlenecks associated with it. The bottlenecks associated with a block are arranged in a counterclockwise direction around the block. When a block needs to be moved to achieve better placement, it is easy to trace through the bottlenecks around the block and its nearby area to update the routing and geometrical information. Bottlenecks could disappear or could be generated as placement changes. However, this rarely occurs and it only takes place when the active region of a bottleneck becomes empty. By and large, the complete set of bottlenecks changes very little throughout the placement adjustment and routing process.

The prerouting analysis consists of two passes, one for the horizontal and another for the vertical. Only the horizontal analysis is discussed here because the vertical analysis is essentially the same except for the direction. First, a signl span is defined as the minimal rectangular region containing all its terminals. For each signal, the horizontal prerouting analysis starts with the finding of bottlenecks within its signal span. Next, the probability of a particular signal going through a particular bottleneck is calculated by assuming that the signal has equal probability of going through any bottleneck within the signal span. Finally, the expected number of signals going through a bottleneck is obtained by summing up the expected densities for all signals going through this bottleneck. Based on this information the preliminary placement adjustment is made, which allocates routing space between blocks. As a rule, blocks are shifted toward the left as far as possible. The prerouting analysis will reduce significantly subsequent block movements in the routing process. This new placement is then used as the initial placement for global routing. The System will work on absolute coordinates from this point on. A complete set of bottleneck is created for this placement and maintained throughout the global routing and the detailed routing processes to follow.

The signal routing order is determined by the routing space available for each signal span with the smallest one assigned first. The idea behind this is simply that it is more difficult to route signals with smaller routing space available because they will more likely cause congestion. The routing space pertaining to a given signal is approximated by the sum of available tracks of all bottlenecks within the signal span.

A "global routing graph" is tentatively defined as follows: A node corresponds to a routing region between two neighboring bottlenecks. An edge represents a bottleneck which connects two nodes if and only if the two nodes represent two neighboring regions of the bottleneck. The global routing graph for the example in Fig. 3 is shown in Fig. 4 with the edge numbers corresponding to the bottleneck numbers. Each edge is given a weight which reflects the congestion of the associated bottleneck and the length of its active region. It is updated every time a signal is routed. Furthermore, the global routing graph is slightly modified for each signal being routed according to the position of the terminals which belong to the signal. If a terminal resides in an active region of a bottleneck, a new node is added on the edge associated with the bottleneck and hence splits the edge in two. This will be helpful in controlling the routing density within that bottleneck. If a terminal does not reside in an active region, it is simply represented by a node of that region.

A newly developed "Steiner Tree on Graphs" (STOG)-algorithm is used for the global routing of each signal [14]. Because the problem of "Steiner Tree on Graphs" is NP-complete, STOG is a heuristic algorithm based on a 3-point "Steiner Tree on Graphs" algorithm. This algorithm has a reasonable computational complexity and yields very good results from our experimental tests. During the global routing phase special emphasis is placed on the control of signals going through bottlenecks. If there is overcrowdedness within a bottleneck, then the global router will either try not to assign signals through that crowded bottleneck or move some blocks to create more room. The System has the capability of minimizing the number of blocks to be moved, while, in the meantime, not increasing the size of the chip dimension. All information is updated with the help of the set of bottlenecks associated with the placement as mentioned earlier.

4. Detailed Routing

In the detailed routing phase the existing channel router and switch-box (2-dimensional) router are called for track assignment [11,12]. Both routers use two layers for routing. The channel router can handle rectangular regions with fixed terminals on two sides of the channels and floating terminals on the remaining two sides. After channel routing, floating terminals become fixed in position. Each active region of a bottleneck defines a channel routing region. The switch-box router can handle an arbitrary rectilinear region with fixed or floating terminals on all sides. Each region which corresponds to a node in the global routing graph defines a switch-box routing region. Figure 5 shows the channel-router routing region and the switch-box router routing region of the example in Fig. 3.

The System first calculates the actual number of tracks needed for each bottleneck region and removes the unneeded space by two passes of compaction in the horizontal and vertical directions. After the compaction most bottlenecks will have all their available tracks occupied. The switch-box routing regions are routed following the order from the lower left corner of the chip to the upper right corner. The bottleneck regions around each switch-box are routed by channel router first in order to fix the terminals on the boundary of the switch box. The switch-box router is next called for the region at hand. The router will beedback the actual space required to complete the routing. If the routing space allocated is not enough, then some blocks must be moved in the direction upward and to the right so as to guarantee that regions already routed in the lower left directions will not be disturbed.

After completion of the BBL routing programs, we discovered that switch-box routing is critical in enhancing the quality and reducing the iteraton times of detailed routing. The following improvements have been made to increase the completion rate of switch-box routing.

The first one is to enlarge the routing space for switch-box. In the original design the entire active region of a bottleneck is used for channel routing. But some active regions have no terminals near their sidewalls. This kind of sidewalls will be pushed toward the center of the associated active regions until they meet a terminal. This sidewall pushing operation has no effects on channel routing but results in some relief of those crowed switch-boxes.

The second improvement is terminal alighment on channel sidewalls. The idea behind is to add a little global sense into channel routing which has been routed independent of each other so far. Instead of letting channel router decide terminal positions on the channel sidewall arbitrarily, we try to force terminals to be put on favorable positions for the future switch-box routing. Two way forcing has been implemented. If a terminal on a channel sidewall will be connected to a terminal across a switch box facing each other, then their positions will be forced to be aligned. The signal can then be connected as a straight line in that switch-box to reduce a jog, thus the routing space required. If a terminal on a channel sidewall will be connected to a termnal in certain direction, say north, then the channel router will try to put it on the north side of the channel sidewall. The advantage is that the signal of routes will be shorter and unnecessary crossing of signals in the switch box could be removed. The nearby terminal distribution information can be obtained by traversing bottlenecks and blocks in the neighborhood of the channel sidewall under consideration.

The third improvement is routing region combination. Whenever switch-box router fails to complete routing in a switch box, this switch box will be combined with two of its neighboring channel regions to form a bigger switch box. With this bigger region the switch-box has a better chance to complete the routing because the router has at its command more freedom to arrange the nets in the channel regions in a more favorable way. However, caution should be taken not to allow the region to be too large in size because the speed of the switch-box router has a second order time dependence on the box size.

A test example achieves about 10% reductionin chip size after implementing the above mentioned improvements.

5. Power and Ground Routing

The power and ground nets (P&G) will be allowed different wire widths according to power consumption information provided by the user. P&G has priority to use the metal layer which can provide higher current densities and lower resistance. BBL handles P&G routing in a similar way as signal routing but with some modification in the global and detailed routing.

In the global routing part, P&G will be given higher priorities than signal nets so that they may get minimum length global routes.

Also the edge weights of global routing graph will be adjusted to force crossings of P&G to a minimum. Given the tree of a P&G from the global routing, it is possible to calculate the width of each segment of the net. Starting from power consumption information of each terminal which is a leaf of the tree, the width associated with an edge in the tree is the sum of widths of all those terminals below it in the tree. This bottom up process computes widths for all edges which correspond to bottlenecks. The width information will be maintained in the bottleneck as well as P&G terminals throughout the entire routing process.

In the detailed routing part, existing channel router and switch-box router are modified for P&G routing. A preprocessor for channel router will route the P&G near the channel boundaries and put them on the metal layer. Channel boundaries are then modified to exclude P&G routing regions. Signal terminals shielded by the P&G routing will be propagated to the new channel boundaries. A channel router that can handle irregular boundaries is called for the signal routing. In the switch-box router the P&G nets will be routed first. Protection frames will be put around the P&G routing on the metal layer so that later signal routing which utilizes metal layer will avoid these regions.

6. Conclusion and Future Work

The System is implemented in C language on a VAX 11/780 under the Berkeley UNIX operating system. The example of above with 26 nets has been routed and is shown in Fig. 6. A real chip with 11 blocks and 73 nets has been routed by the System and is shown in Fig. 7. The CPU time for this example is 145 seconds and the memory is about 1M bytes. In comparison with the Siclop Layout System using the same placement [2,3], BBL uses 16% less space. An example from AMI with 33 functional blocks, 132 nets and 440 pins has been tested. It takes 5.5 minutes cpu time and 2.5 megabytes on a VAX 11/780 under Berkeley UNIX operating system. The chip size is about 8.5% less than the original AMI layout. The result is shown in Fig. 8.

Fugure work to be completed includes a constructive placement algorithm, the power/ground routing and further refinements on detailed routing. We expect the complete system to be up and running within the next few months.

References

[1] K. Kani, H. Kawanishi, and A. Kishimoto, "ROBIN: A building block LSI routing program," Proc. IEEE Int. Symp. on Circuits and Systems, 1976, pp. 658-661.

[2] B. T. Preas and C. W. Gwyn, "Methods for hierarchical automatic layout of custom LSI circuit masks," Proc. 15th Design Automation Conference, 1978, pp. 206-212.

[3] B. T. Preas, "Placement and routing algorithms for hierarchical integrated circuit layout," Technical Report 180, Computer System Laboratory, Stanford University, August 1979.

[4] Ronald L. Rivest, "The 'PI' (Placement and Interconnect) system," Proc. 19th Design Automation Conference, 1982, pp. 475-481.

[5] J. Soukup and J. Royle, "Cellmap representation for hierarchical layout," Proc. 17th Design Automation Conference, 1980, pp. 591-594.

[6] B. W. Colbry and J. Soukup, "Layout aspect of the VLSI microprocessor design," Proc. IEEE Int. Symp. on Circuits and Systems, 1982, pp. 1214-1228.

[7] J. Soukup, "Circuit layout," Proc. of the IEEE, Vol. 69, No. 10, 1981, pp. 1281-1304.

[8] M. Wiesel and D. A. Mylnski, "An efficient channel model for building block LSI," Proc. IEEE Int. Symp. on Circuits and Systems, 1981, pp. 118-121.

[9] Horng, C. S., and M. Lie, "An automatic/interative layout planning system for arbitrarily-sized rectangular building blocks," Proc. 18th Design Automation Conference, 1981, pp. 293-300.

[10] U. Lautter, "A min-cut placement algorithm for general cell assemblies

based on graph representation," Proc. 16th Design Automation Conference, 1979, pp. 1-10.

[11] Y. Yoshimura and E. S. Kuh, "Efficient algorithms for channel routing," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. CAD-1, No. 1, January 1982, pp. 25-35.

[12] C. P. Hsu, "A new two-dimensional routing algorithm," Proc. 19th Design Automation Conference, 1982, pp. 46-50.

[13] K. Y. Keller, A. R. Newton and S. Ellis, "A symbolic design system for integrated circuits," Proc. 19th Design Automation Conference, 1982, pp. 460-466.

[14] N. P. Chen, "New algorithms for Steiner tree on graphs," submitted to IEEE Int. Symp. on Circuits and Systems, 1983.

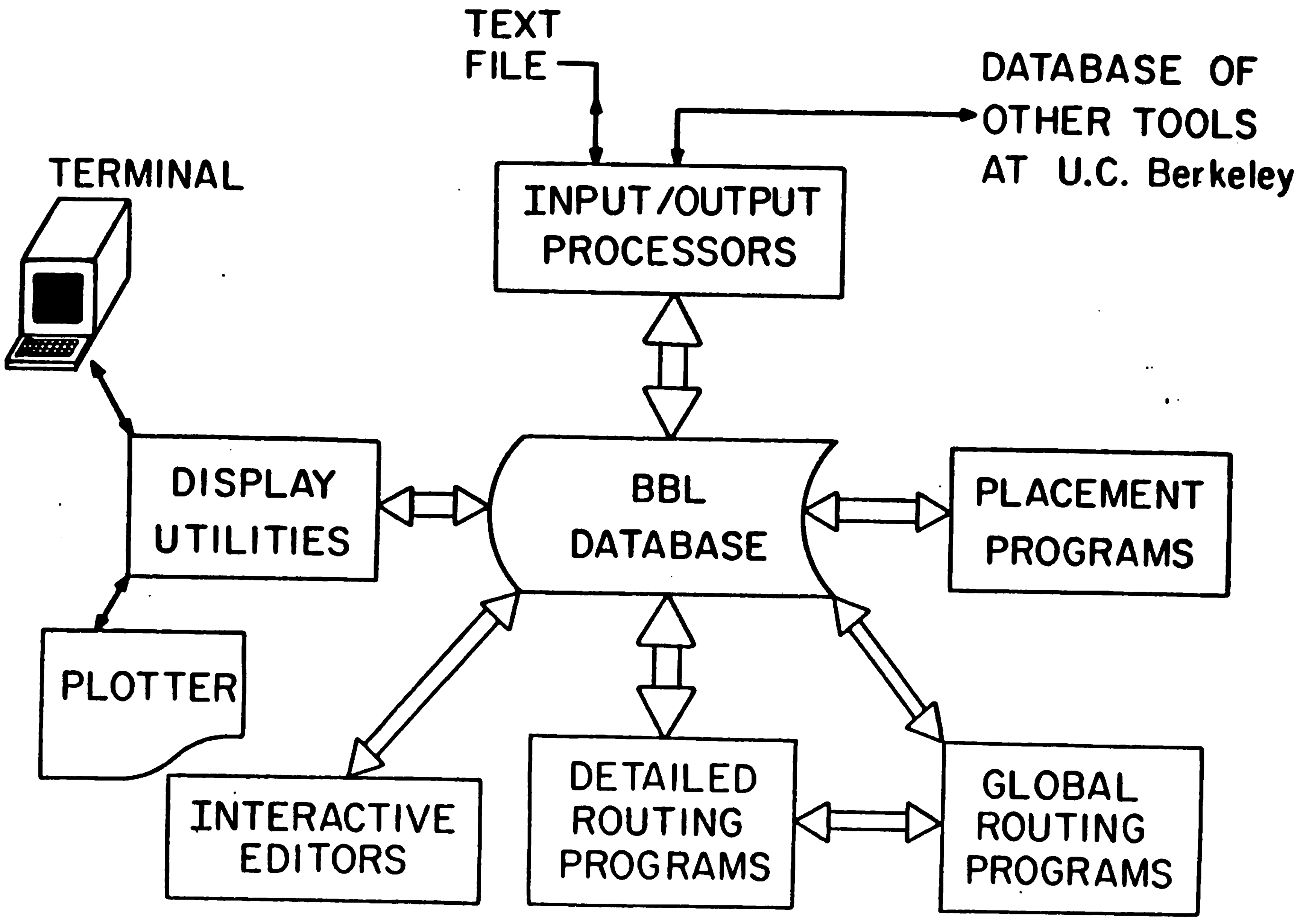
TEXT FILE
DATABASE OF OTHER TOOLS AT U.C. Berkeley
TERMINAL
INPUT/OUTPUT PROCESSORS
DISPLAY UTILITIES
BBL DATABASE
PLACEMENT PROGRAMS
PLOTTER
INTERACTIVE EDITORS
DETAILED ROUTING PROGRAMS
GLOBAL ROUTING PROGRAMS

Fig. 1

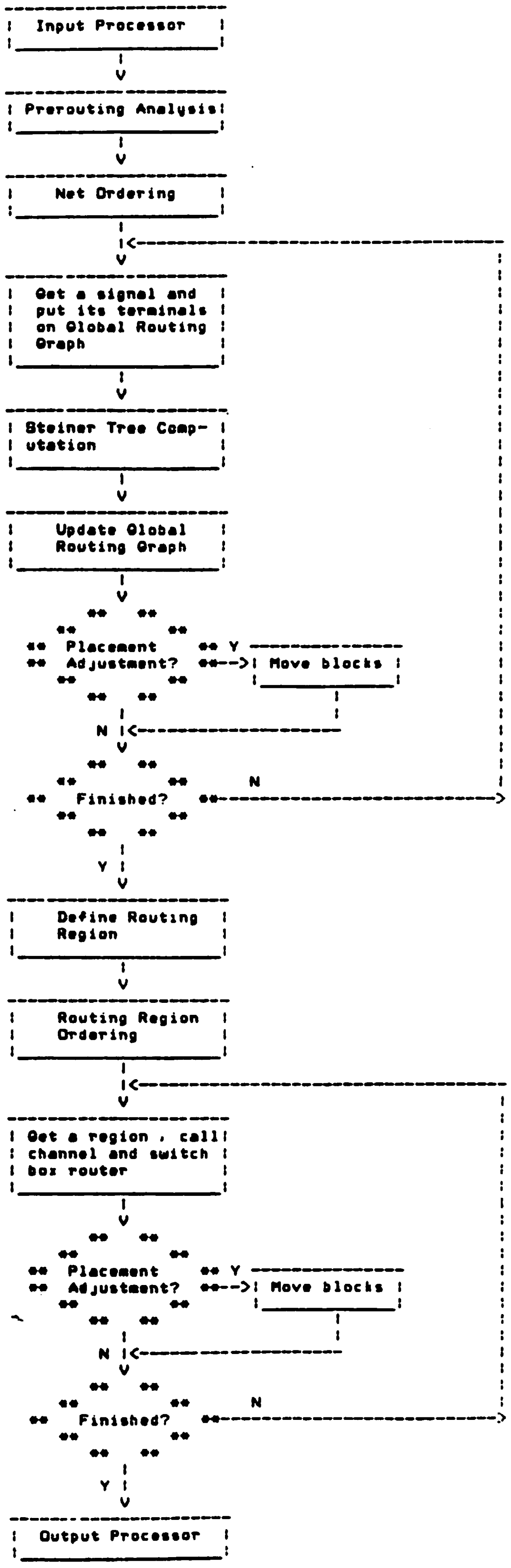

Fig 2 The routing system flowchart

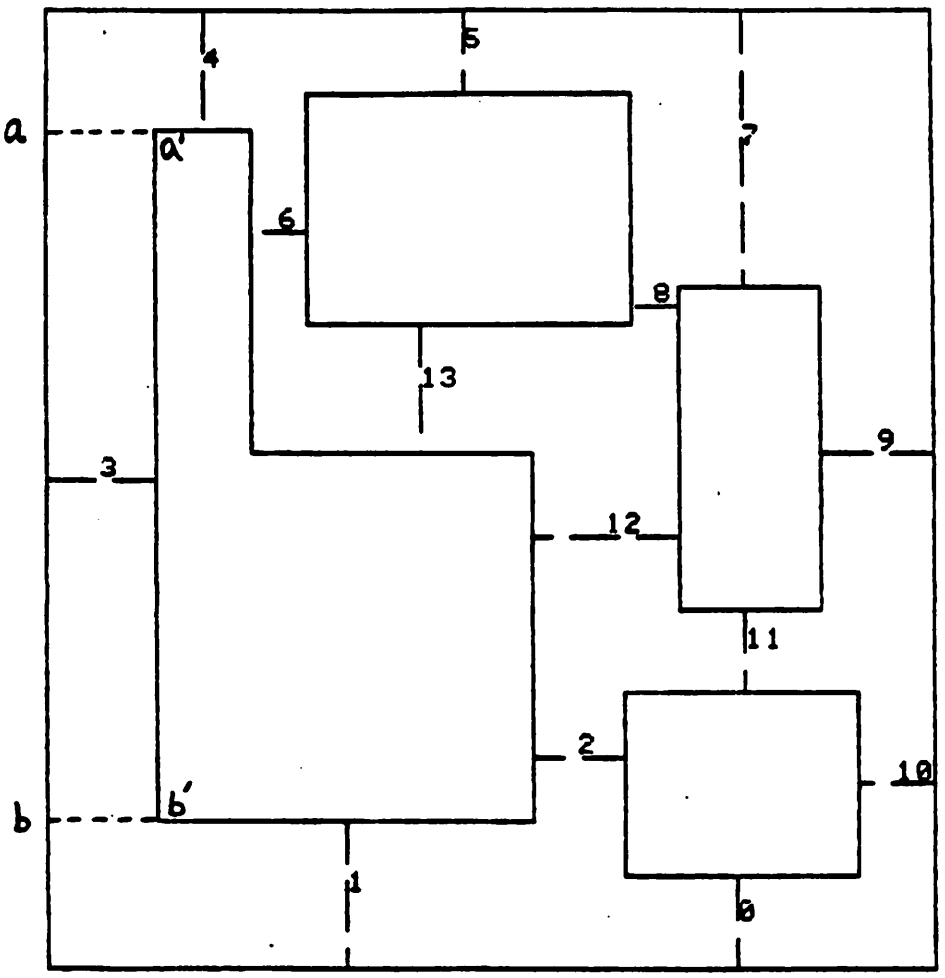

Fig. 3 Bottlenecks. aa' - bb' defines the active region for bottleneck 3.

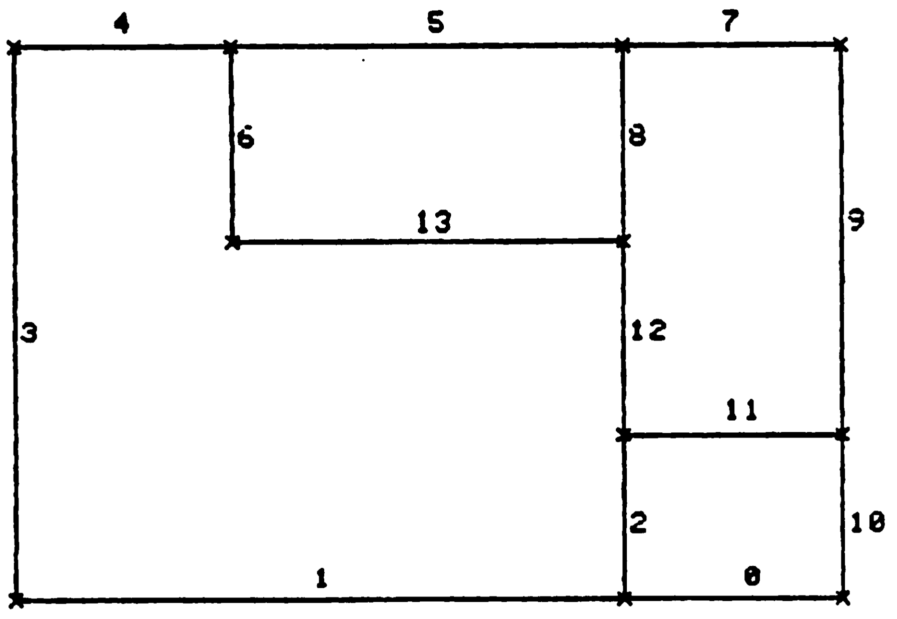

Fig. 4 Global Routing Graph of the example in Fig. 3. Edge number corresponding to bottleneck number.

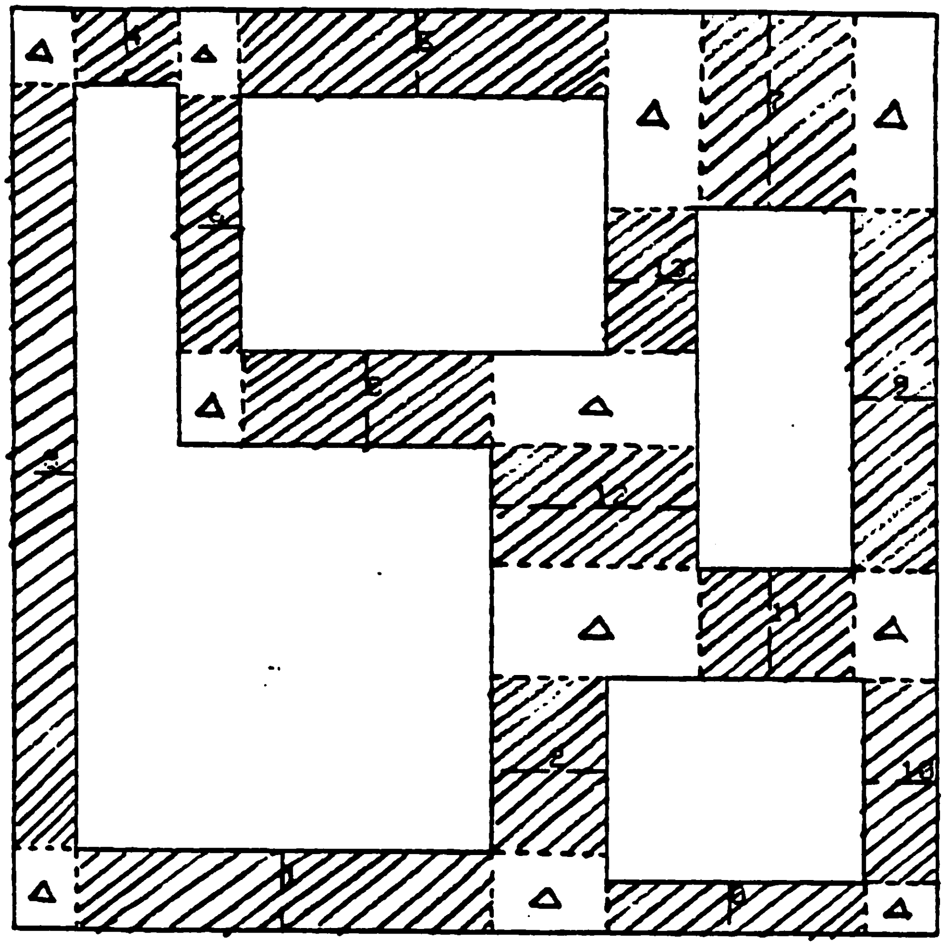

Fig. 5 Routing regions of the example in Fig. 3.
Δ : subox router routing region,
※ : channel router routing region.

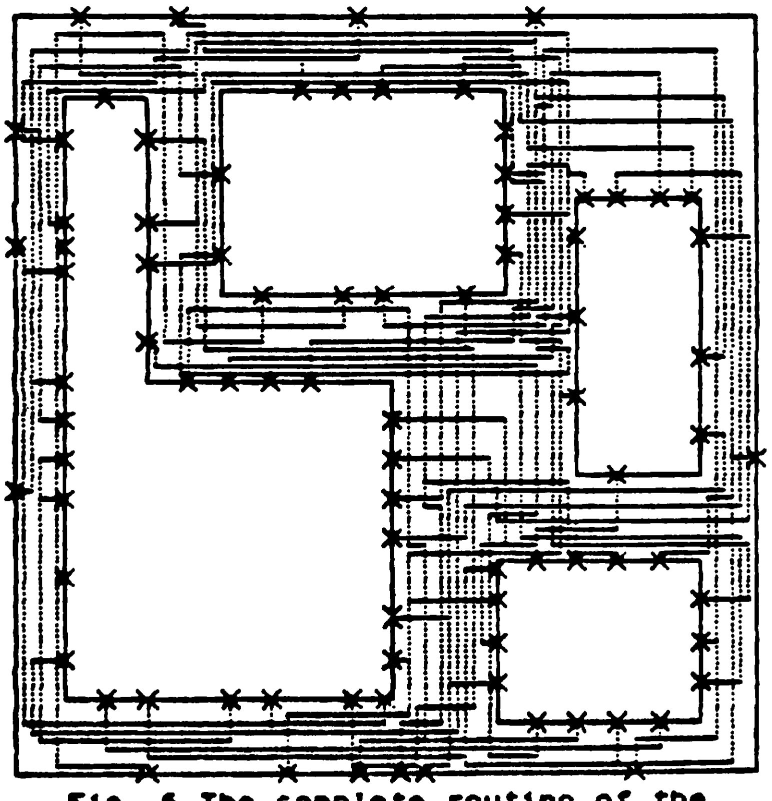

Fig. 6 The complete routing of the example in Fig.3.

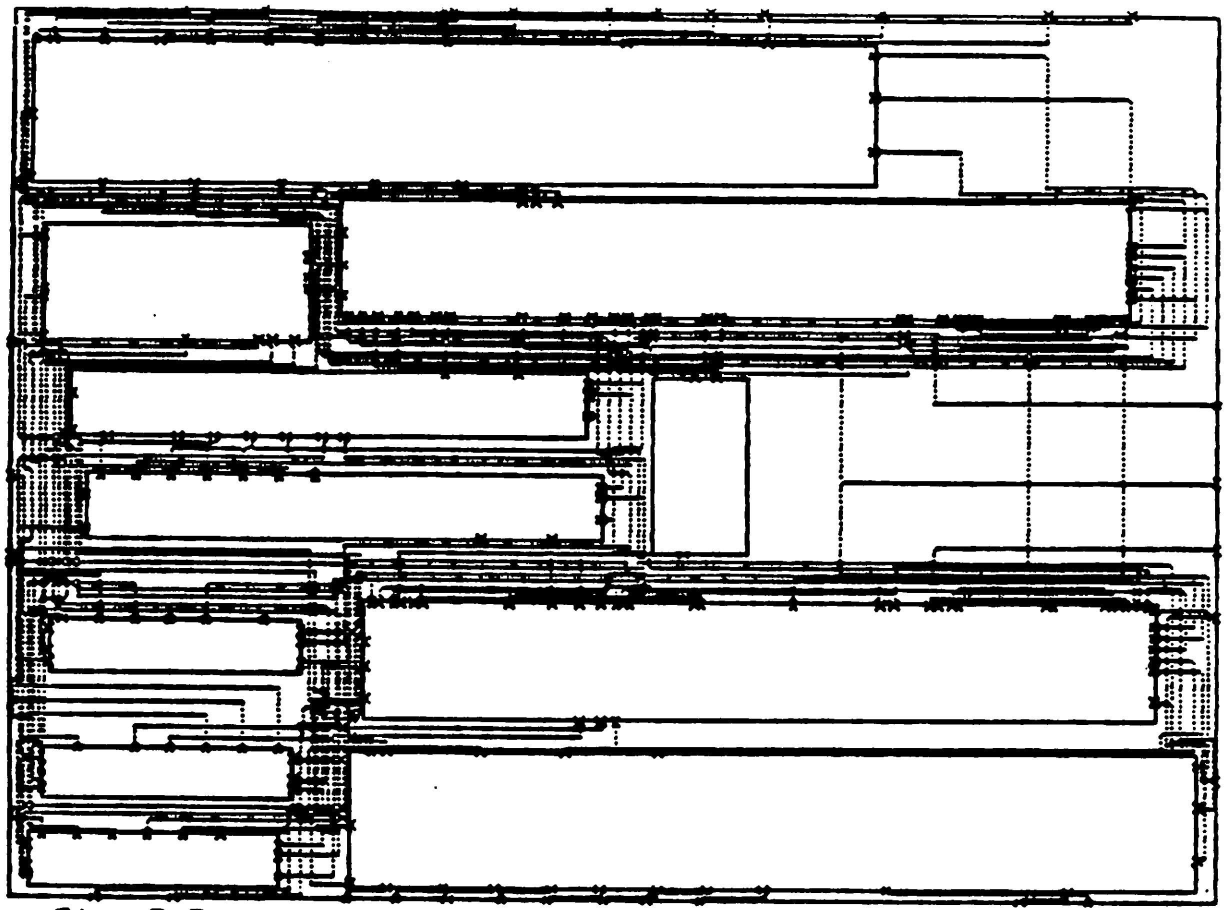

Fig. 7 Example of routing on a real chip

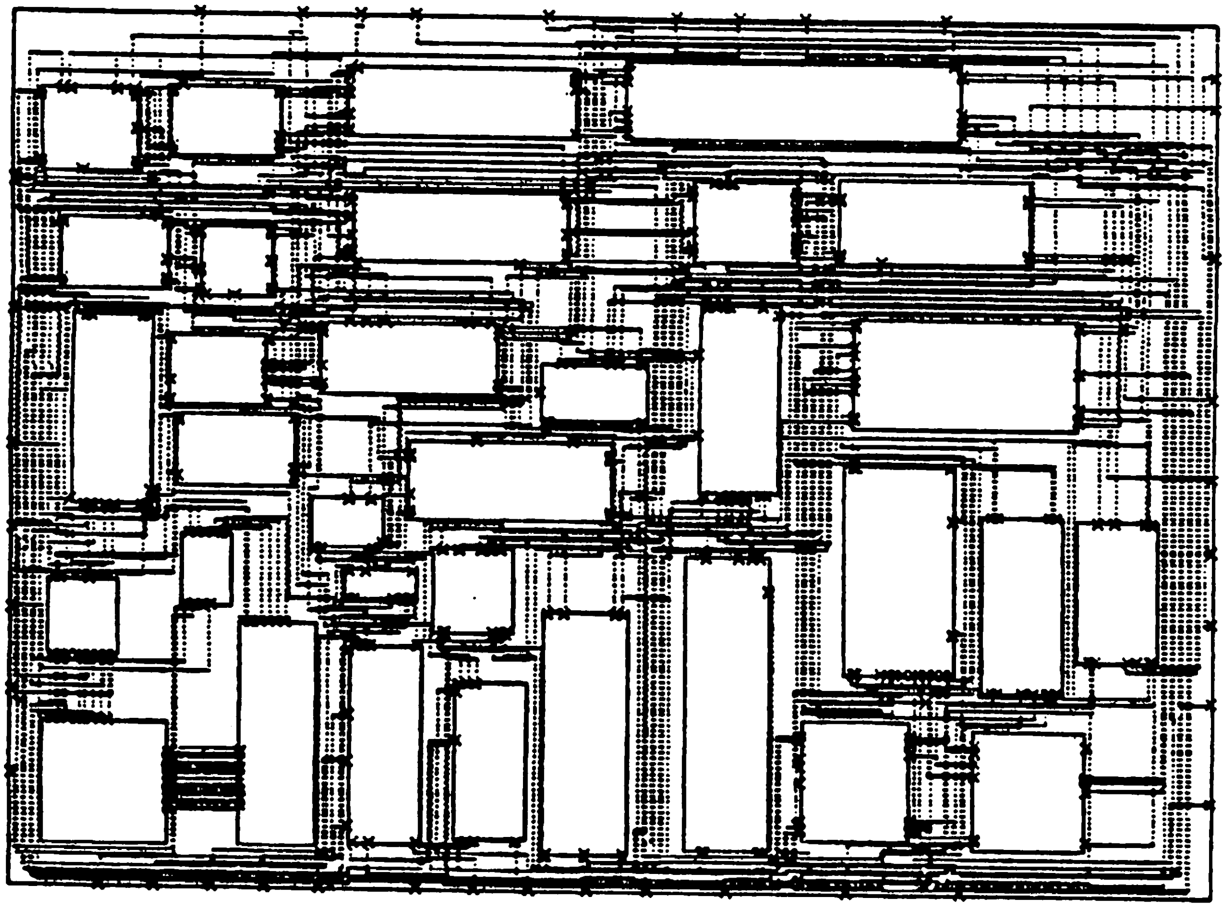

Fig. 8 AMI example with 33 blocks, 132 nets and 440 pins.