OPTIMIZED ONE-DIMENSIONAL COMPACTION OF BUILDING-BLOCK LAYOUT

by

Xiao-Ming Xiong

Memorandum No. UCB/ERL M87/45

23 May 1987

COVER PAGE

OPTIMIZED ONE-DIMENSIONAL COMPACTION OF BUILDING-BLOCK LAYOUT

by

Xiao-Ming Xiong

Memorandum No. UCB/ERL M87/45

23 May 1987

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

TITLE PAGE

OPTIMIZED ONE-DIMENSIONAL COMPACTION OF BUILDING-BLOCK LAYOUT

by

Xiao-Ming Xiong

Memorandum No. UCB/ERL M87/45

23 May 1987

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Optimized One-Dimensional Compaction of Building-Block Layout

Xiao-Ming Xiong

Electronics Research Laboratory
Department of EECS
University of California, Berkeley
Berkeley, California 94720

ABSTRACT

A new optimized method for compacting a VLSI chip in one direction is presented in this paper. In contrast to existing compaction algorithms, we rely on the geometric method and bypass the constraint graph during the entire compaction process. A systematic way is proposed to enumerate all possible jogs. Therefore, under the given layout topology and design rules, our algorithm yields the optimal one-dimensional compaction by repositioning the layout objects and introducing jogs. In the final layout, only necessary jogs are inserted, and total wire length is minimized. The algorithm is very efficient in both complexity and actual run time. The worst case run time complexity is $O(n\log n + k)$ and space complexity is $O(n + k)$ where n is the number of layout objects in the input chip and k is the number of possible jogs enumerated during the compaction.

Optimized One-Dimensional Compaction of Building-Block Layout

Xiao-Ming Xiong

Electronics Research Laboratory
Department of EECS
University of California, Berkeley
Berkeley, California 94720

1. Introduction and background

Compaction is a term used to describe the process of repeatedly cutting unnecessary spaces from a rough layout. A compactor is an effective tool for cutting the production costs of a VLSI chip because the yield of fabricated chips is strongly dependent on the total chip area. Different methodologies and algorithms for layout compaction have been developed and well described in the literature ([1] and [2]). Due to the huge amount of input data in a layout chip, it is almost impossible to keep and manipulate the global information of the entire chip. This makes chip compaction extremely difficult. Instead of trying to compact the chip simultaneously in both directions, most compactors developed so far alternately compact the chip in the x and y direction using a one-dimensional approach based on the longest path search in the horizontal or the vertical constraint graphs as used by Hsueh and Pederson ([3]). The constraint graph approach offers good flexibility and allows for an efficient implementation. However, this approach does not work well with the compaction problem involving automatic jog introduction. When we look at the

This research is sponsored by the Semiconductor Research Cooperation under SRC 82-11-008.

constraint graph, we can never tell where to introduce the jogs to reduce the chip size further. A common method used to introduce jogs is to first find the critical path(s) in the constraint graph and then go back to the layout plane using some kind of local searching techniques to determine whether there is room for jogs. This kind of improvement methods makes automatic jog introduction difficult and time consuming. Furthermore, it can not guarantee to find all the jog positions even though there is room for jogs due to the nature of the local search technique.

In our new one-dimensional compaction algorithm, we use the plane sweep technique ([4]) and all compaction procedures are done directly on the layout plane. Compared with other one-dimensional compaction algorithms, ours maintains and handles the data only on a scan line. Thus our algorithm is more efficient. During the compaction, we use a systematic way to enumerate all possible jogs. Therefore, under the given layout topology and design rules, our algorithm achieves the lower bound of one-dimensional compaction with automatic jog insertion. Due to the inherent properties of the scan line approach, our algorithm has no problem in handling rectilinear layout objects.

2. Underlying ideas

The idea of using the geometric method to do the compaction and bypass the constraint graph appeared in an earlier paper in channel routing spacing ([5]). With some modifications, we are able to apply it to the general one-dimensional problem.

Without loss of generality, let us assume we compact the chip in the y direction. We call the boundaries of cell blocks, terminals, wires and contacts in the given layout plane *layout objects.* We sort all layout objects in the chip lexicographically by their y, x coordinates and then sweep the chip plane by jumping a scan line along the y coordinates of the layout objects. By *current scan line,* we mean the y coordinate where the scan line is currently located. By *previous boundary,* we mean the front of the layout objects with y coordinates smaller than the current scan line. For each mask layer, we have a previous boundary. *A corner point* is a discontinuous point on the previous boundary. An object *covers* a corner point if the object is immediately above the point and the the x-coordinate of the point is in between the x-coordinates of the two end points of the object.

Our algorithm consists of two phases. In the first phase, we enumerate all possible jog positions of a wire and find out the maximum move for every object. This is the maximal distance an object can be moved down without violating the given design rules, by scanning the chip plane from the bottom to the top. The possible jog positions of a wire are determined by examining the corner point covered by this wire. The maximum move is calculated by comparing the y-coordinate of the object with the previous boundaries corresponding to the mask layers occupied by the object. We call a group of layout objects which must be moved together *a bag.* For example, a cell block, the terminals attached to this block, and the wires connected to these terminals form a bag. The maximum move of a bag is the minimum of all the moves of the layout objects in the bag. The chip height is determined in this phase. Fig. 1 shows a simple example and Fig. 2 explicitly displays the result after the first phase of the algorithm. Then in the second phase, we embed all layout objects in the chip from the top down to the bottom, minimizing the number of jogs and decreasing the wire length obtained in the first phase. In this phase, we keep the front of all the embedded layout objects. This front combined with the maximum movement of an object determines the feasible range of the location of the object. When a wire is embedded, the wire is placed as close as possible, within the feasible range, to the terminals connected in order to minimize the total wire length. Fig. 3 shows the result after a y direction compaction of the example in Fig. 1. Next, we compact in the x direction as shown in Fig. 4. Fig. 5 shows the final result.

3. Complexity Analysis and Experimental Results

Because we simply use the plane sweep algorithm to do the compaction, the complexities of our algorithm are the same as the plane sweep algorithm. Let n be total number of objects in the chip and k be the number of possible jogs enumerated, the worst case time complexity of the algorithm is $O(n\log n + k)$ if a balanced tree is used to maintain the data on a scan line. If a linked list is used to maintain the data on a scan line, the worst case time complexity of the algorithm will be $O(n^2 + k)$. The space complexity for the algorithm is $O(n + k)$.

The algorithm is implemented in C. Different examples have been tested. Fig. 6 shows an example with about 1,000 layout objects. We compact this example first in the y direction and then in the x direction. The chip area reduction is 4.3% (in the y direction) + 2.2% (in the x direction). The total wire length reduction is about 5%. The compaction was done in 40 seconds on a microvax workstation. Fig. 7 shows another example with about 10,000 layout objects. We compact this example first in the y direction and then in the x direction. The chip area reduction is 3.8% (in the y direction) + 1.1% (in the x direction). The total wire length decreased by about 4.5%. The compaction was done in about 33 minutes on a microvax workstation.

In Fig. 8 and Fig. 9, we insert all possible jogs and move every object in the chip by its maximum move in the above two examples to show that the lower bound of the chip height in one-dimensional compaction with automatic jog insertion is achieved by our algorithm. The chip area reduction is strongly dependent on the input chip layout topology and design rules.

References

[1]. Y. E. Cho, “A Subjective Review of Compaction”, Proc. of the 22nd Design Automation Conference, June 1985, pp. 396-404.

[2]. D. D. Mlynski and C. H. Sung, "Layout Compaction", Advances in CAD for VLSI, Vol. 4, T. Ohtsuki Editor, North Holland Publ. Co., 1985.

[3]. M. Y. Hsueh and D. O. Pederson, "Computer-Aided Layout of LSI Circuit Building Blocks", Proc. IEEE International Symposium on Circuits and Systems, 1979, pp. 474-477.

[4]. J. Nievergelt and F. P. Preparata, "Plane-Sweep Algorithms for Intersecting Geometric Figures", Communications of the ACM, Volume 25, Number 10, October 1982, pp. 739-747.

[5]. X.-M. Xiong and E. S. Kuh, "Nutcracker: An Efficient and Intelligent Channel Spacer", Proc. of the 24th Design Automation Conf., June 1987.

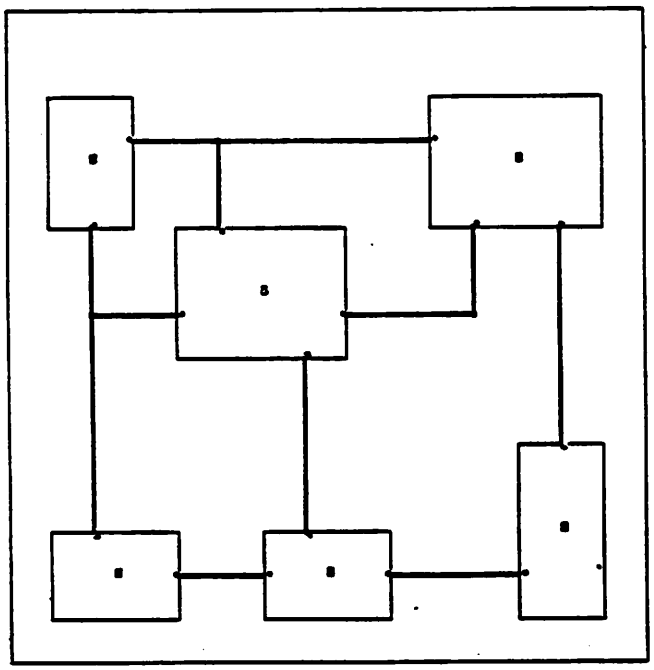

Fig. 1

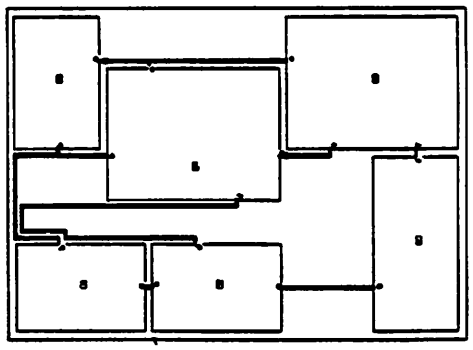

Fig. 4

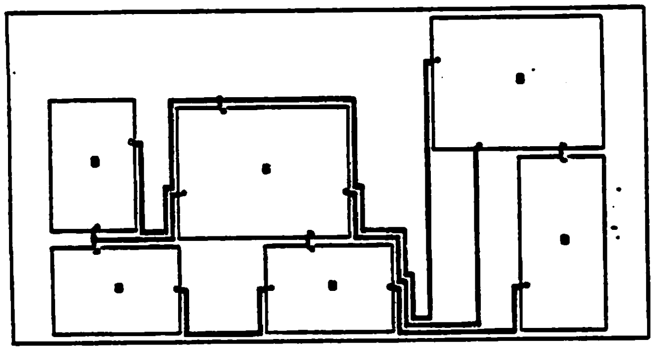

Fig. 2

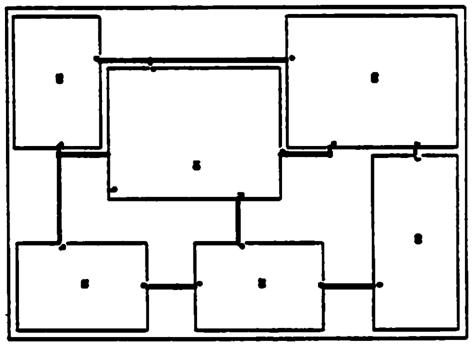

Fig. 5

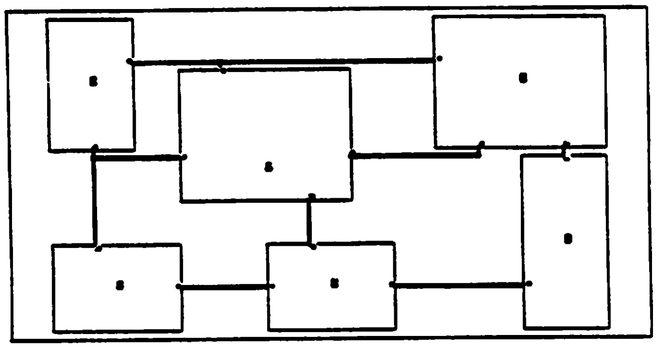

Fig. 3

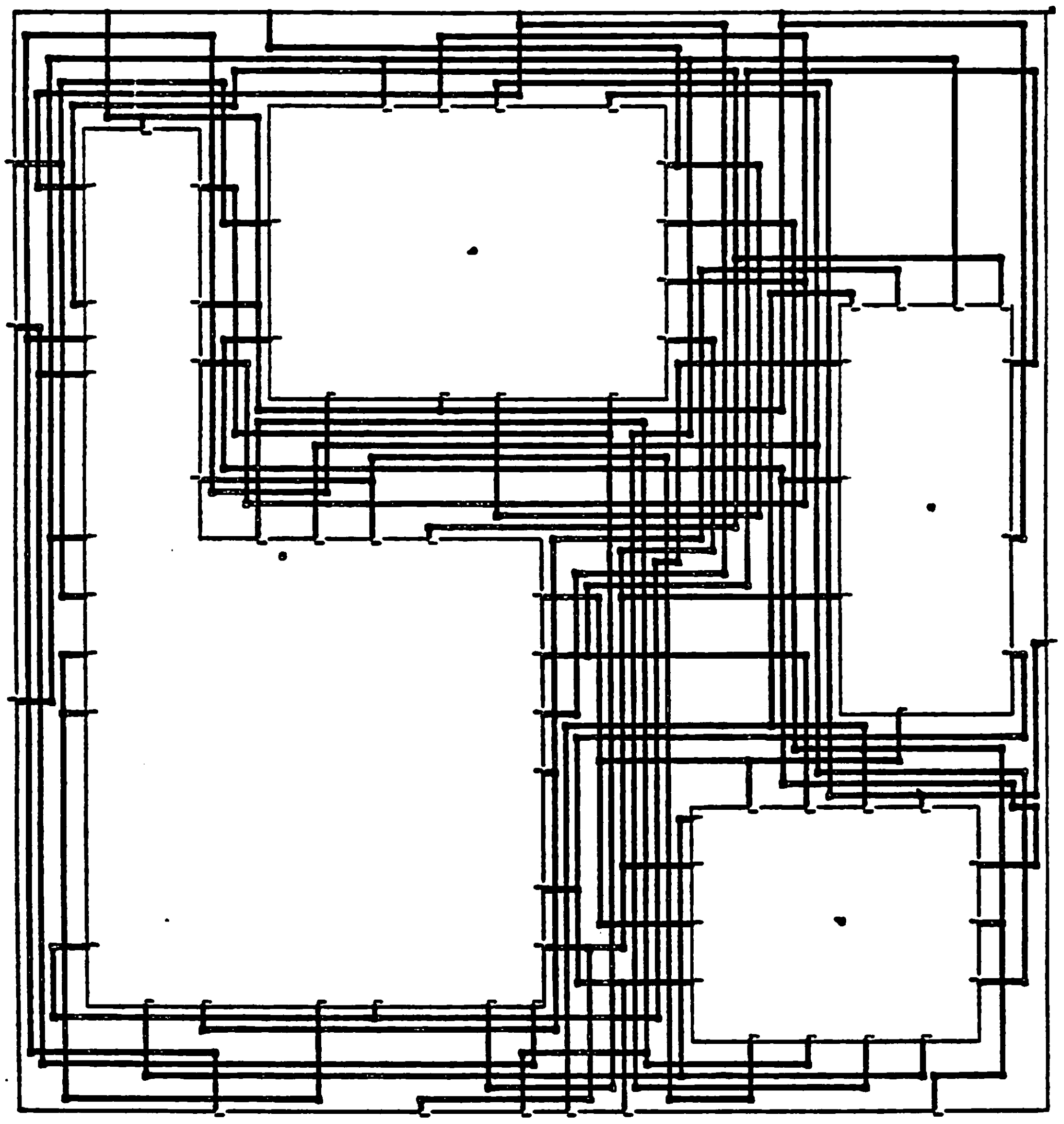

Fig. 6

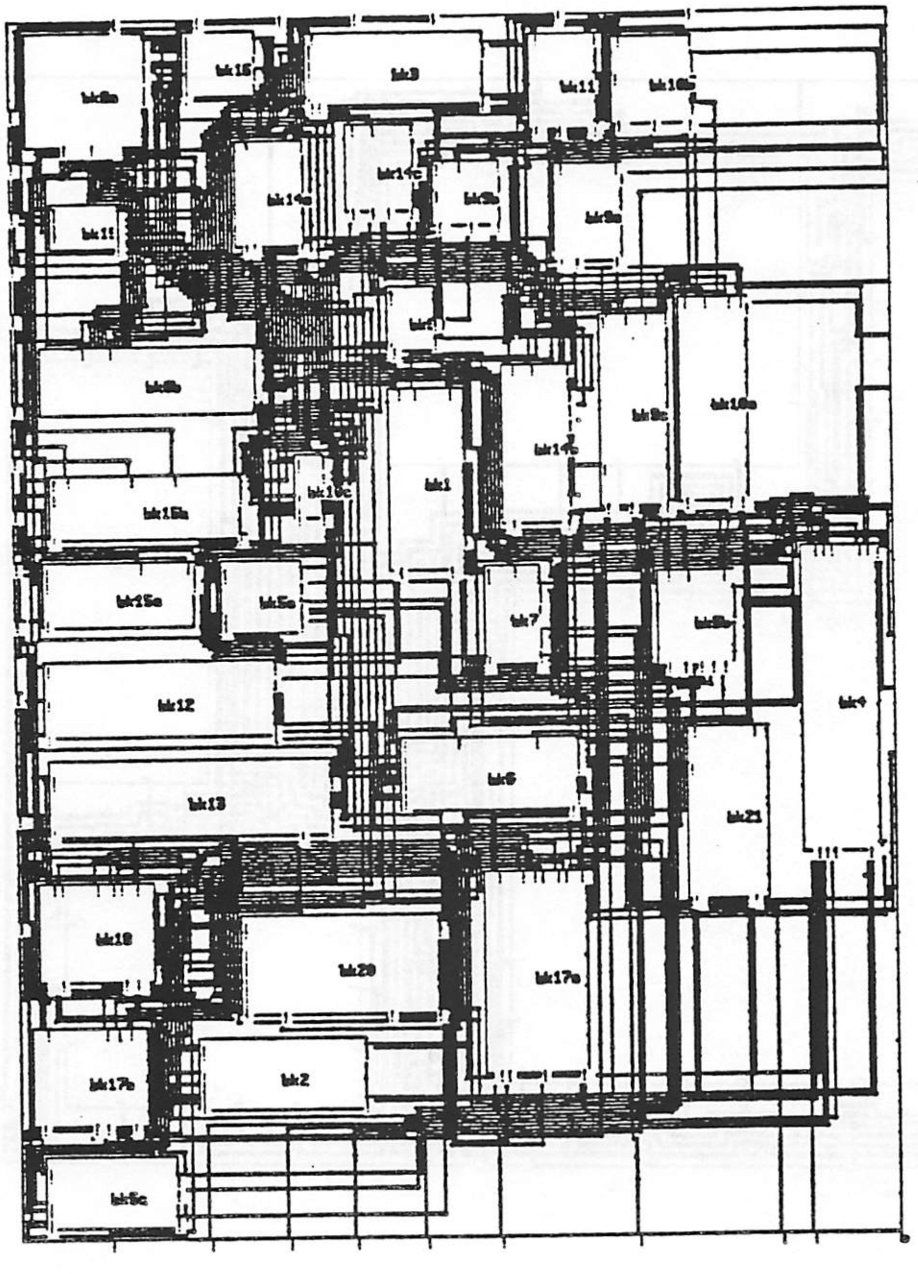
blk8a
blk15
blk3
blk11
blk10b
blk14a
blk14c
blk3b
blk11
blk9a
blk2
blk8b
blk9c
blk10a
blk14b
blk10c
blk1
blk16a
blk15a
blk5a
blk7
blk8b
blk4
blk12
blk6
blk13
blk21
blk18
blk20
blk17a
blk17b
blk2
blk5c

Fig. 7

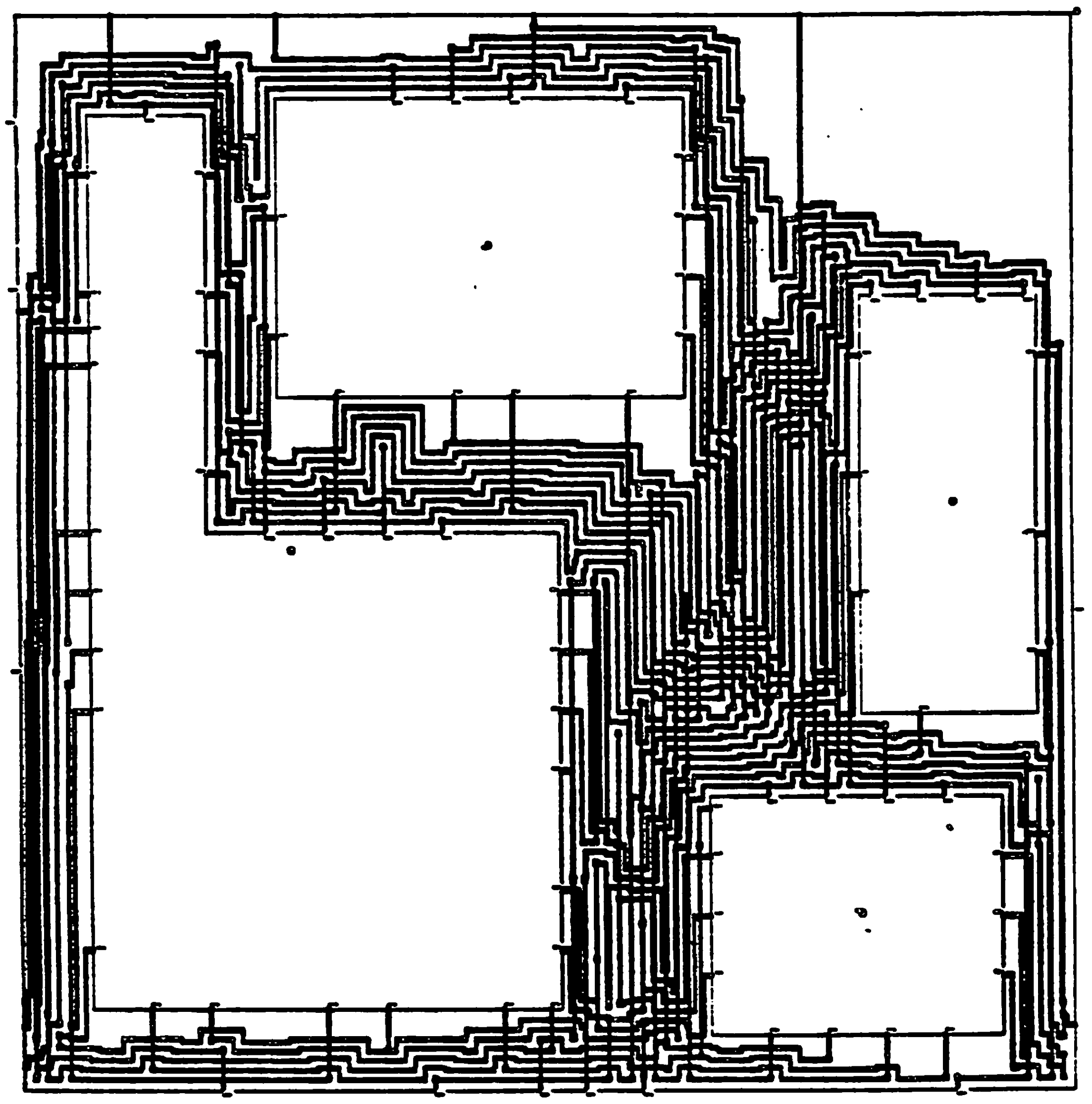

Fig. 8

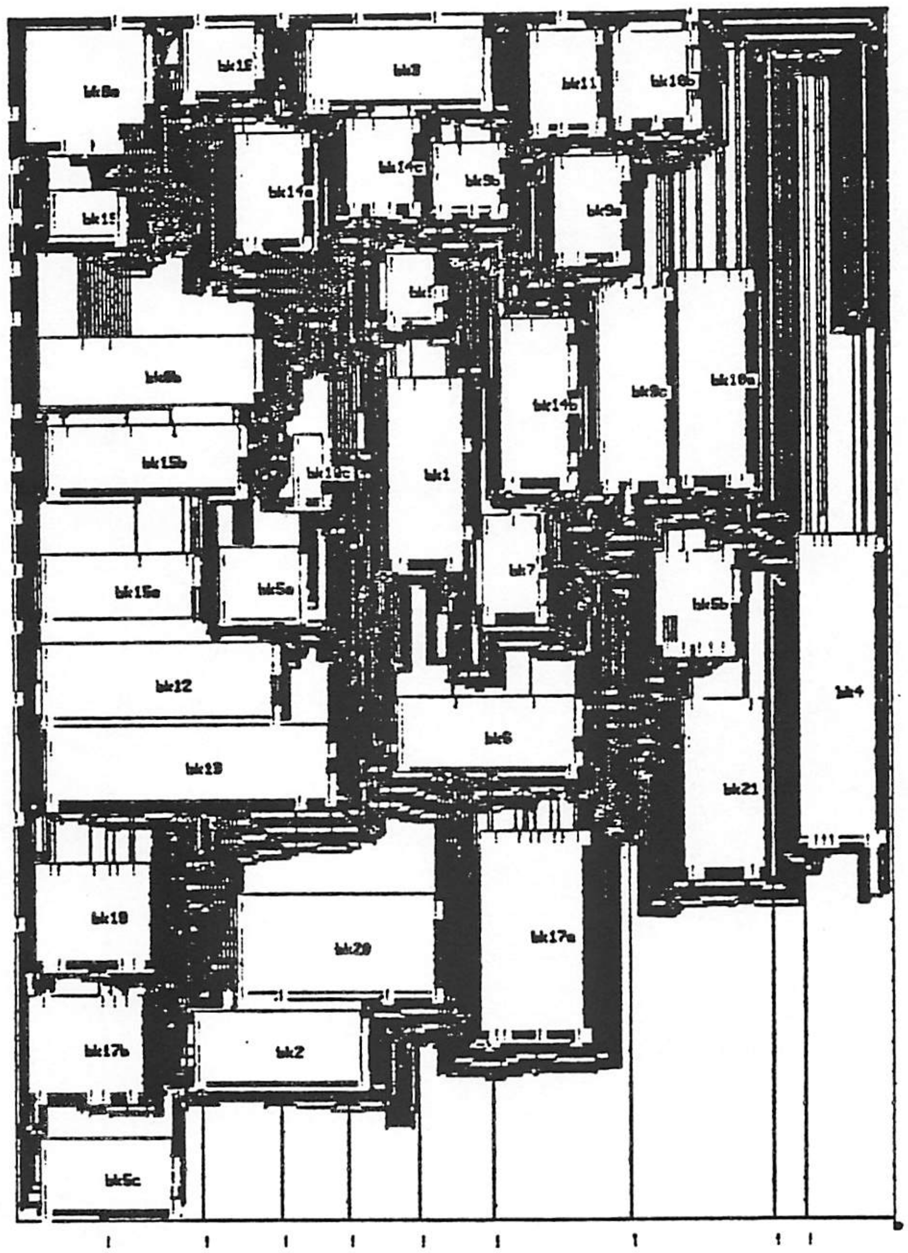
bk8a
bk1f
bk3
bk11
bk16b
bk14a
bk14c
bk9b
bk9a
bk15
bk8b
bk14b
bk8c
bk16a
bk15b
bk16c
bk1
bk7
bk15a
bk5a
bk5b
bk12
bk6
bk4
bk13
bk21
bk18
bk20
bk17a
bk17b
bk2
bk6c

Fig. 9