Mem. Fac. Sci. Eng. Shimane Univ.
Series A
32, pp. 237-245 (1998)

A Fast Maze Router Algorithm

Masahiko SUMI

Department of Mathematics and Computer Science,
Interdisciplinary Faculty of Science and Engineering, Shimane University
(Received September 18, 1998)

Abstract

A 100–1,000 times faster maze router was studied for a 2 or 3 metal layer ULSI layout. Similarly to Lee's router, it always guarantees a solution, if one exists. Also, it usually gives the same path length and number of bends. On the contrary to Lee's router, connectivity checking and path improvements are accomplished separately. High speed was attained by limiting routing path possibilities to boundaries of carefully chosen obstructions.

1. Introduction

So far, LSI layout has been carried out mainly by channel routers, because only one metal layer was available before. Recently, however, a three metal layer process has become available and a more general two dimensional router is requested in order to control size and configuration for the layout blocks in a chip, as well as to obtain higher density.

Historically, two dimensional routers were studied earlier than channel routers. Lee's maze router appeared 30 years ago, and it was modified by Mikami-Tabuchi[Mik68], Hightower[Hig69], Soukup[Sou78], Heynes[Hey80] and others.

Two dimensional routers deal with the interconnection of 2 points on a plane with irregularly shaped obstructions, as shown in Fig. 1.

Major requirements for the solution are:

(1) To guarantee a solution, if one exists.
(2) Shorter path.
(3) Fewer bends.
(4) Reasonable CPU memory space.
(5) Reasonable CPU time.

(In addition, determining an appropriate connection order is another difficult problem to be solved.)

Lee's algorithm fully solved reguirements (1) and (2) at the price of (3), (4) and (5). Both (4) and (5) were fatal problems 30 years ago. However, (4) may not be so today, as the memory space is expanding by about 1,000 times, even in a low cost system, as compared with a main frame system at that time. Item (5) is still fatal, as it takes about 1,000 seconds to connect only one net across a $10{,}000 \times 10{,}000$ grid whith a 10 MIPS CPU. Since ULSI net number

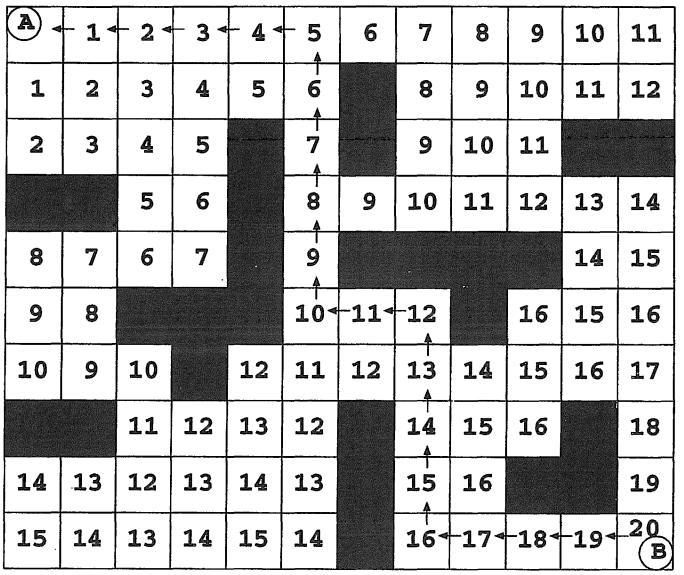

Fig. 1. Original Lee's router algorithm.

reaches 10^5 today, 10^8 second (1,000 days!) time is roughly estimated for Lee's algorithm.

The routing time for one net amounts to $O(n^2)$ for an $n \times n$ grid system. What makes things worse is that the LSI net number is doubling every 2 years. Hence, the routing time must increase 4 times in two years, if the MIPS value to remain the same. Therefore, a low ordered, extremely fast maze router is strongly required.

Soukup's algorithm increased the labeling speed 10–50 times. However, it did not increase the speed when there is no path.

Hightower's line-search algorithm looks faster, because lines are searched for in place of dots. However, one disadvantage is that the algorithm may not find a path, when one exists (P 14), i.e., achieving item (1) is not guaranteed.

Mikami-Tabuchi's line search algorithm guarantees item (1). However, it generates for too many search lines.

Heyns' line-expansion algorithm generates much fewer search lines, but the speed decreases rapidly, when the component number increases.

In practical cases, "line search algorithms" are rather slower than dot search algorithms[Oht86]. The paradox comes from data structures. The dots can be stored in an array, and intersection test and data modification is fast. On the other hand, a line search algorithm requires $O(\log N)$ time in N component system, to pick one intersection and to modify one "line". This holds true, even if lines are stored in a well designed tree structure database.

For the above reason, the present router uses Lee's cell model. In order to obtain an extremely high speed, (1) cell labeling number is reduced from $O(n^2)$ to $O(n)$ and (2) label clear-

ance procedure was carefully removed. Therefore, O(n) speed up is expected. Even if a factor of 10 is reduced, 10^3 times speed up is expected in a $10{,}000 \times 10{,}000$ grid system.

As for item (3) and (4), the present router gives good solutions in most cases, but not always. However, at least it provides a flexible and high speed framing (routing area limiting) tool for other routers.

Only four bits are necessary to represent one cell and 50M bytes are needed for a 10^8 grid system. One second/net routing speed for the ULSI with a 10 MIPS CPU can be expected (for incremental chip refinement).

2. Principle

Lee's router and most routers examine connectivity and search for a good solution at one time. This is still correct in a small size problem. In a big system, the problem is usually divided into global and detail routers. However, the present approach enhances the speed in a big grid system by separating the problem into two parts as follows.

The present first and second routers only check the connectivity and the reasonable routing area at a minimum effort. Only when a solution is guaranteed is the present third router run to find a good solution.

Before explaining the first router, a quick review of Lee's router is given in Fig. 1. To connect points A and B on a plane with obstructions, increasing number labels are given from A to neighboring cells successively until B is found. Then, the numbers are reversely traced back from B to A. This is an excessive operation when it is only desired to check whether or not there is a solution.

2.1 Connectivity Inspection Router

Figure 2 shows the present first router, which only inspects connectivity, before the main routing process is operated. The whole LSI is considered to be the routing area in this step.

Connectivity inspection steps are:

(1) Connect A and B by an arbitrary guide-line. (marked gray in Fig. 2)
(2) Move from A toward B on the guide-line.
(3) If the guide-line hits an obstacle, turn it clockwise (or counterclockwise) completely until the hit point is recognized again.
(4) Jump to the nearest cell (to B) which is neighboring to the obstacle and on the guide-line.
(5) Go to (2) and continue the movement, until B is found.

Path 12–21 and path 34–38 are apparently redundant but very essential. If simply counterclockwise (or clockwise) inspection is done until the guide-line is found again, connectivity is not found out in an example shown as Fig. 3. The guide-line may be the solution, if there is no obstacle. The guide-line length is 2n and the first router's movement is, at most, several times the guide-line length. During the movement, XY coordinates can be recorded on a stack, as the step number is as small as O(n). No marks are given to cells and no clearance is necessary. Note

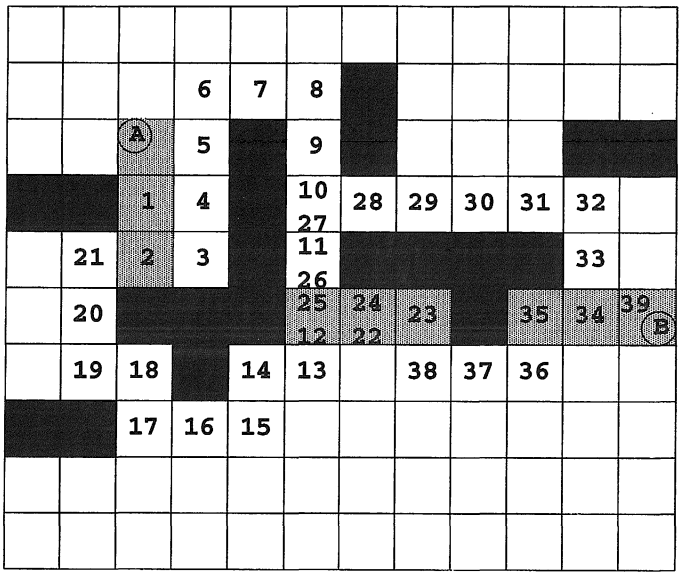

Fig. 2. Present connectivity checking router.
Connect the two points A and B by a guide-line (marked gray). Turn around each obstacle to find the nearest cell to the target. Restart from the nearest cell.

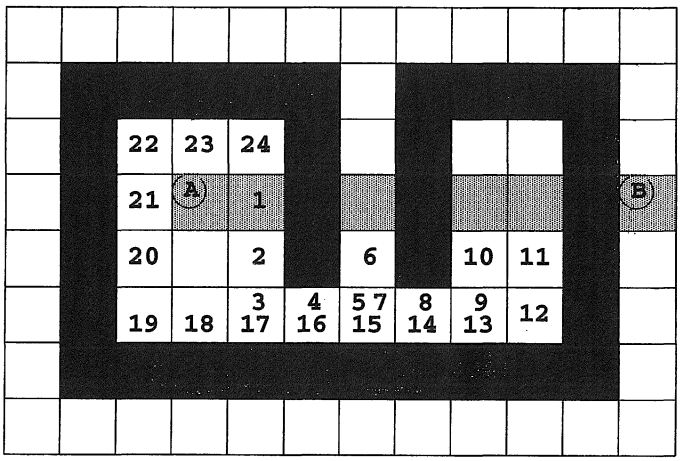

Fig. 3. A difficult connectivity checking example.
Simple counterclockwise (or even clockwise) turning causes falure at 24th pint. While, the present algorithm gives a different path at 7th cell and the path can be escaped.

that obstacles may be penninsulars from the world boundary and that the principle can be applied to a non-bit-mapped (gridless) model.

2.2 Framing Router

Once connectivity is guaranteed at least in the whole chip, the second stage to be undertaken is "framing" (limiting a routing area between A and B), as shown in Fig. 4. To simplify the description, points A and B locations are assumed to be on the framing boundary. The first boundary, consisting of "R" letters, is obtained as follows.

Right boundary routing steps are:

(1) Start from A and go along the right-hand-side wall, cell by cell, until B is found.

(2) If there is an obstacle on the way, turn it clockwise until the wall is found. Go to (1).

If B is not found, framing area will be spread for further trials.

Similarly, the second boundary, consisting of "L" letters, is obtained by turning obstacles counterclockwise. In contrast with the first router, this second router marks "R" and "L" (inclusively) on the bit map for later use. The two boundaries are initial solutions. The refinement is accomplished by the third (main) router within the two solutions. It is clear that the second router's path length is also O(n).

Practically, the first router may be used after the second router to change the routing area, after the second router failure is recognized. The first and the second routers can be used incrementally to dynamically change the framing area according to the congestion as shown in Fig. 5.

2.3 Main Router

When the routing area is defined by the second router, The third (main) router is run, as

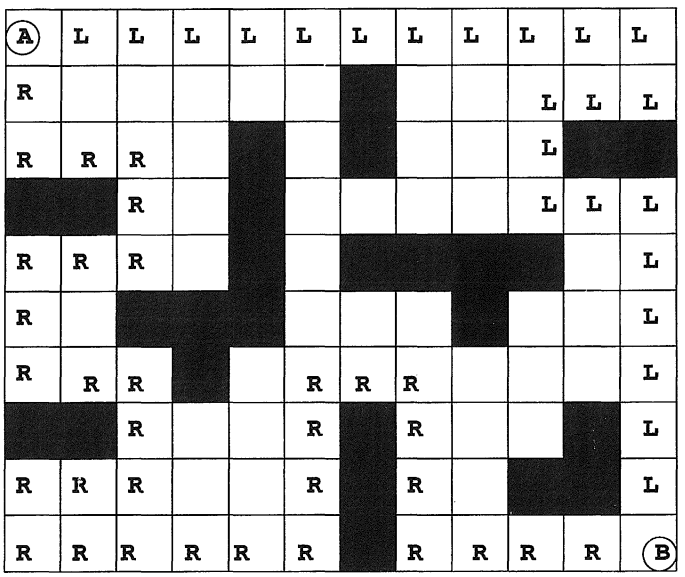

Fig. 4. Present framing router.

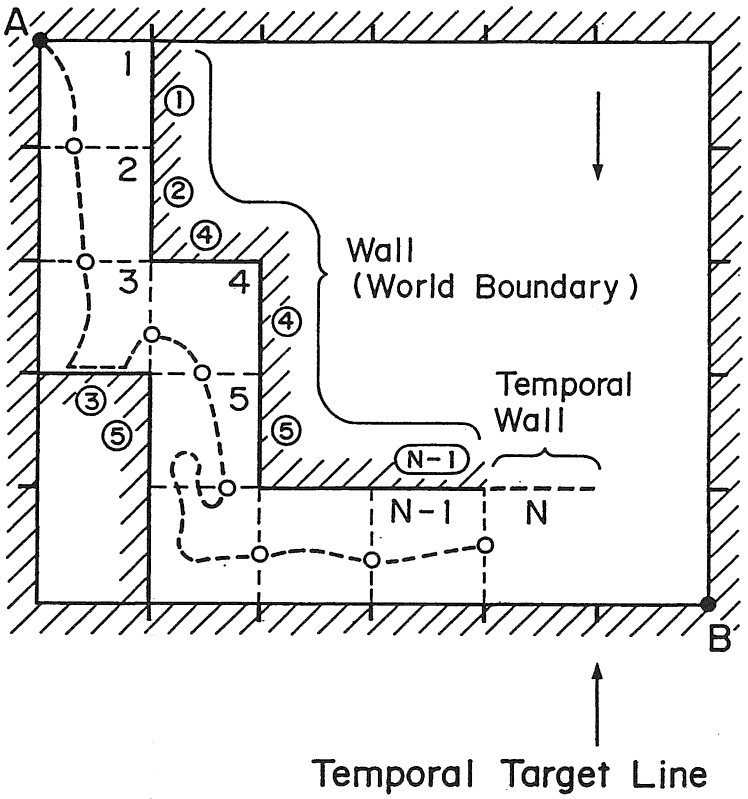

Fig. 5. Present dynamic framing by temporal walls.
Framing area is defined step by step, checking by the present router. When the area is fixed. The third router or Lee's router is used to remove path bends.

shown in Fig. 6. The main router's motion is similar to the second router. However, it is modified to prevent making too redundant bends. For this purpose, it slides and leaves obstacles when limited conditions are meet, while the second router never leaves obstacles.

For example, at the 4th cell in Fig. 6, it stops turning the obstacle clockwise and slides to the south. Therefore, the routing path, after the 6th cell, differs from that for the second router. At 12th and 20th cells, similar similar situations occur and the path slides from 12th to 20th or 25th to 28th, respectively. When the path is obtained, the old "R" boundary line is updated to this new path.

There are two redundant paths in the example. One is from 2nd to 9th and the other is 18th to 23th. The next stage, refinement is carried out by moving backwards, from B to A. In the backward movement, it stops turning counterclockwise and slides obstacles toward the West at 25th and 12th points. The backward line, intersects the new "R" boundary at 17th and 1st points. In this way, the final route is obtained as follows:

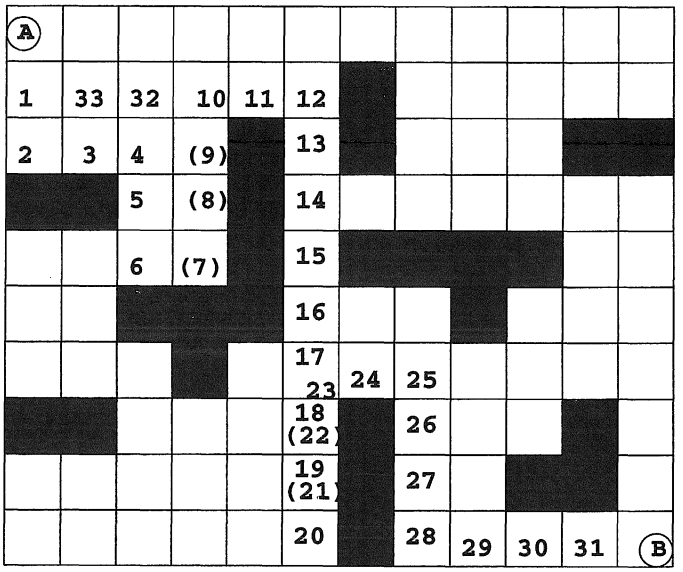

Fig. 6. Present O(n) main router.
The first path is: A 1 2 3.....29 30 31 B
The final path is: A 1 33 32 10 11 12 13 14 15 16 17 24 25 26 27 28 29 30 31 B.

A 1 33 32 10 11 12 13 14 15 16 17 24 25 26 27 28 29 30 31 B.

It is important that only one direction among two target directions is allowed for sliding.

Only one slide direction is: (Assume A is upper and more left to B.)

(1) South, when going A to B along R boundary.
(2) West, B A R .
(3) East, A B L .
(4) North, B A L .

The above "turning" and "sliding" rules are not yet complete. The path may be trapped in a endless loop without the following additional rules.

Rules to avoid path trapping:

(1) At "sliding", examine if the point has been previously recorded as a starting point for sliding. If so, continue "turning"
the obstacle. (This situation occurs seldom.)
(2) If a path intersects during "sliding", with an obstacle and finds L(R) in clockwise (counterclockwise) mode, i.e., in R(L) mode, go along L(R) boundary until it can slide toward its allowed direction.

3. Experimental Results and Discussions

The principle was implemented to a two layer CAD model. This is sufficient to the three metal process, as one metal layer is mostly used for library cells.

To represent one routing cell status, 4 bits are used. The 1st and the 2nd bits are used to show if metal layer 2 and metal layer 3 are available. The 3rd and 4th bits are used to label boundaries "R" and "L" for the present router. When Lee's router is run, alternatively, these 2 bits are used for "123123..." scheme labeling. Both present router and Lee's router were run to same (A to B) sets, to compare lengths and bends.

In the first experiment, a 100×100 grid system was used. Obstacles were prepared by random pre-routing. Sets of point A and B were also obtained from random numbers. Instead of placing A and B at corners, a horizontal margin is given for the routing area. This is to enable routing even when A and B are on a vertical line.

In most ($>90\%$) cases, the minimum length paths were obtained by the two routers, however, the paths are different. Note that in a 2 layer system, extremely many bends other than those described in the last section are generated, when the path is turning thrugh a region, where a already used one of the metal layers locates next to a complete obstacle. Such bends are optimized by simply removing the repeating coordinates from the path.

The reason for the unexpectedly good solutions is considered as follows. At Hightower's router, target direction is not used and the path length quality is insufficient. Generally, there are two directions toward the target. Soukup's router utilized both two directions, however, while the present router utilized only one direction at one time (from A to B). This rule enables a better backward (B to A) direction optimization. Recursive optimization between A and B can be also possible. In many ($>90\%$) cases, the present router gives the same path length to Lee's by one backward moving optimization.

A similar program in a $1{,}000 \times 1{,}000$ grid system was estimated that Lee's router took 188 sec, while the present router took 3 seconds for one path.

So far, the framing (done by the present second router) has been accomplished in a rectangular shape. However, the present approach also enables arbitrary shaped framing. Incremental dynamic framing as shown in Fig. 5. is effective to improve routing speed and to avoid local congestion.

Conclusion

1. An $O(n)$, in a $n \times n$ grid system, fast maze router was proposed and investigated. The router consists of (1) connectivity checking router, (2) a framing router and (3) an optimization router.

2. Experiments in a 100×100 grid system showed 1.007 path length ratio to Lee's router.

3. It is estimated to be 1,000 times faster than Lee's router can be expected in a $10{,}000 \times 10{,}000$ ULSI grid system.

References

[Lee61] C. Y. Lee, "An Algorithm for Path Connections and Its Applications." IRE Trans. on Electronic Computers, Vol. EC–10, No. 3 (September, 1961), pp. 346–365.

[Mik68] K. Mikami and K. Tabuchi, "A Computer Program for Optimal Routing of Printed Circuit Connectors, "IFIPS Proc., Vol. H47, 1968.

[High69] D. W. Hightower, "A Solution to Line-Routing Problem on the Continues Plane," proc. 6th Design Automation Workshop, pp. 1–24, 1969.

[Sou78] J. Soukup, "Fast Maze Router", Proc. 15th Design Automation Conf., pp. 100–102, 1978.

[Hey80] W. Heyns, W. Sansen, and H. Beke, "A Line-Expansion Algorithm for the General Routing Problem with a Guaranteed Solution, "Proc. 17th Design Automation Conf., pp. 243–249, 1980.

[Oht86] T. Ohtsuki, (Ed) "Layout, Design and Vertification" in "Advances in CAD for VLSI" vol. 4, North-Holland, 1986.