A Switchbox Router with Obstacle Avoidance

Gordon T. Hamachi
John K. Ousterhout

Computer Science Division
Department of Electrical Engineering
and Computer Sciences
University of California
Berkeley, California 94720
(415) 642-9716, 642-0865

ABSTRACT

This paper presents a new switchbox router developed as part of the Magic layout system. Based on Rivest and Fiduccia's "greedy" channel router, the Magic router is capable of routing channels containing obstacles such as preexisting wiring. It jogs nets around large obstacles and multi-layer obstacles such as contacts. Where unable to avoid large single-layer obstacles, it river-routes through them. It combines the effectiveness of traditional channel routers with the flexibility of net-at-a-time routers.

Keywords and Phrases: channel routing, physical design aids, layout, VLSI.

1. Introduction

Previously placed wires such as power and ground routing form obstacles in routing areas. We have developed a new switchbox router as part of the Magic layout system [OHM], capable of routing channels containing such obstacles. The router's novel aspect is its ability to both avoid obstacles and consider interactions between nets as channels are routed. It thus combines good features from net-at-a-time routers and traditional channel routers.

The Magic router is an extension of Rivest and Fiduccia's "greedy" channel router [RiF]. It performs a column by column scan of a rectangular routing region. At each column it applies a series of rules controlling the placement of vertical jogs within the column.

Figure 1 shows the solution to a simple routing problem. Figure 2 shows the same routing problem with an obstacle in the routing area. It illustrates some basic principles of obstacle avoidance. As the router extends nets from left to

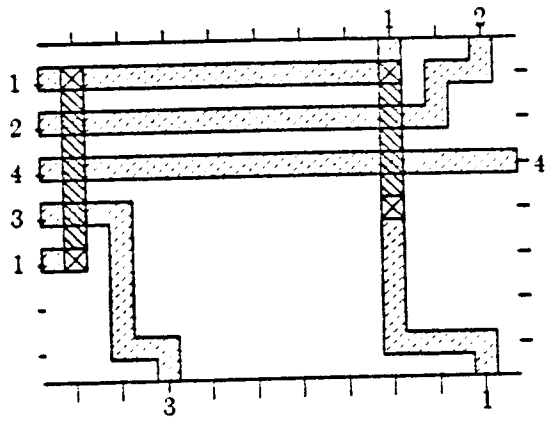

Figure 1. A simple channel routing problem. Numbers around the border of the channel represent pins associated with signal nets. Pins with identical net numbers are connected by the router using a left to right, column by column scan.

right, it tries to avoid large obstacles in the columns ahead by jogging around them. If nets can not jog around large single layer obstacles they *river route* through the obstacles, switching layers if necessary.

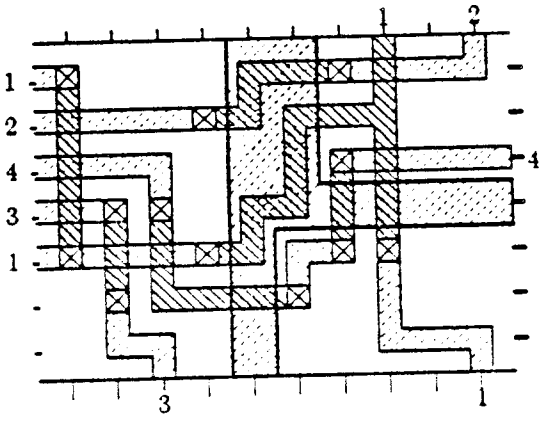

Figure 2. The problem from Figure 1 with obstacles in the channel (drawn with heavy outlines). The router tries to cross obstacles at narrow points. If necessary it river-routes through obstacles.

Section 2 motivates the problem of obstacle avoidance and describes the Magic router's goals. Section 3 summarizes the "Greedy" router, upon which our work is based. Section 4 presents our solutions to a number of problems encountered in adapting the greedy channel router to avoid obstacles. In section 5 we present extensions for routing switchboxes. Section 6 provides a detailed view of the router. Section 7 describes a channel splitting mechanism. The paper

concludes with a discussion of the router's implementation and performance.

2. Motivation

Automated routing systems typically divide the routing of chips into three steps: *channel definition*, *global routing*, and *channel routing*. In the channel definition step, empty areas between cells are divided into non-overlapping rectangular *channels*. The global routing step selects the sequence of channels through which each signal net will be routed to make the desired connections. The channel routing step assigns physical locations to the wires in each channel, realizing the signal routings specified in the global routing step.

A standard model for channel routing assumes a grid of two independent layers of minimum width wiring. Horizontal *tracks* are wired in one of these layers, while vertical *columns* are wired in the other layer. Connections between layers are made with *contacts*; where no contacts appear, layers may cross over each other.

Routers generally assume that channels start off completely free of wiring. Thus, it is impossible to use an automatic router with pre-routed wires. This is a serious limitation since certain signals such as power, ground, and clock lines have special restructions on width, layer, and length which existing channel routers fail to handle.

Because channel routers do not tolerate the presence of obstacles, designers must either accept inferior results generated by automatic routing systems or try to hand patch the router output. Hand patching is difficult because automatic routers leave little room to add wires or to move wires to different layers. Also, if the chip has to be rerouted, the hand patching must be completely redone.

Channel routing systems that do handle obstacles have done so in restricted ways. The PI system [Riv] has the notion of "covered channels" -- areas wired with metal for power and ground routing, through which other signals may be river routed in polysilicon. It fragments large channels containing metal power and ground wiring into many smaller covered and uncovered channels each of which must be routed individually. The problem of routing one large channel is thereby reduced to the problem of routing several smaller, more constrained

channels.

The BBL system [Che], [CHK] handles prewiring from a separate power and ground routing phase. It routes power and ground signals near the edges of channels. BBL then ignores the power and ground routing areas except to bridge other signals across them. It routes all other signals using only the clear parts of the channels. BBL does not allow any hand routing.

Routers using maze [Lee][Hig] routing methods are able to avoid obstacles on multiple layers. The problem with these routers is that they consider only one net at a time. Since they completely route a single net before considering the next net, they cannot consider interactions between nets as a channel is routed. For this reason these routers are inferior to true channel routers for channel routing of general layouts [Sou].

The Magic router provides a general obstacle avoidance capability that combines the advantages of the above approaches. It allows designers to prewire critical nets, putting them at any position and in any layer. The Magic router routes around these prewired nets.

The router considers interactions between nets. Routing decisions are based on an overall strategy rather than on a net-at-a-time basis. Considering tradeoffs between alternatives improves the overall quality of the resulting wiring.

Magic uses single-layer obstructed areas to do useful routing. Since large, loosely constrained areas are easier to route, it avoids fragmenting these obstructed areas into small, highly constrained, hard to route areas.

Particularly in interactive design environments nearly optimal results obtained quickly are more useful than optimal results obtained after long computation. Our router is fast, and produce good results.

3. The Greedy Router

Since the greedy router algorithm is the starting point from which our algorithm was developed, we start with a brief overview of its operation. Three features of the greedy router are of particular importance. First, the greedy router makes a column by column scan of the routing area. It completely wires the current column before extending active tracks into the next column. Second,

it uses a list of rules to control the placement of vertical wiring in a column. Rules are applied in order of importance, to a) avoid "getting stuck"; and b) to make subsequent columns easier to route (Figure 3). The list of rules can easily be modified. Third, unlike constraint graph approaches, the greedy router allows *split nets*, nets that occupy more than one track at a time. Split nets give the router the flexibility to evaluate alternatives and choose the one that is best for the overall routing problem.

Column wiring begins by bringing the nets of a column's top and bottom pins (if any) into the first tracks that are either vacant or already assigned to the nets. Deferring this to a later step might allow vertical wiring to block a net, preventing it from being brought into a vacant track.

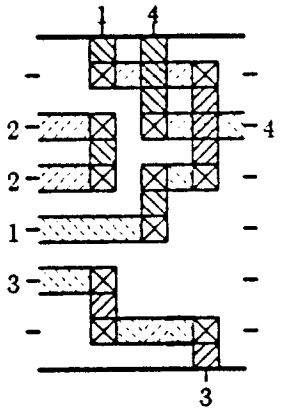

Figure 3. Three columns wired by the greedy router. In the first column net 2 makes a collapsing jog and net 3 makes a falling jog. In the second column net 4 enters the channel, preventing net 1 from making a collapsing jog; however, net 1's lower track makes a jog to reduce the range of tracks assigned to this split net.

Bringing a net into the first available track may leave it *split* on multiple tracks. Split nets can fill up the channel, making it impossible to bring in additional nets. The greedy router thus makes collapsing split nets its next priority. Since conflicting vertical wiring can make it impossible to collapse all split nets in a particular column, the router collapses split nets in the pattern that frees up the most empty tracks for use in the next column.

Vertical wiring conflicts may prevent the router from collapsing all split nets. The router simplifies the routing of these remaining split nets by reducing the range of tracks occupied by these nets. It jogs each split net's highest occupied

track downward and its lowest occupied track upwards. The remaining problem is easier because collapsing can be done with shorter jogs.

Next, unsplit *rising* and *falling* nets are jogged upward or downward toward the edge of the channel with their next pin. This step anticipates the split nets that will be created when upcoming pins' nets are brought into the channel. It attempts to reduce the range of these split nets before they are created. This step prevents split nets if the rising or falling net can be jogged into what would otherwise be the first vacant track seen by a net as it enters the channel from a top or bottom pin.

The handling of split nets and rising and falling nets are examples of decisions based on interactions between nets. Among conflicting alternatives (a jog to raise a rising net may block a jog to lower a falling net) the router chooses the one that does the best job of simplifying the remaining overall problem.

4. Extending the Greedy Router

In modifying the greedy router to avoid obstacles we had to solve a number of problems. The result was an augmented set of rules for placing horizontal and vertical wiring. In the following discussion, an area with a single layer obstacle is called an *obstructed* area. The Magic router river-routes through obstructed areas. An area is *blocked* if it contains a double layer obstacle. No routing may pass through blocked areas.

As it scans a channel from left to right, the greedy router expects that it can always extend a track into the next column if necessary. The router must avoid extending tracks into *blocked* areas (Figure 4).

We solve this problem by anticipating upcoming obstacles and attempting to jog nets out of their way. We do this by identifying areas near obstacles; these areas are called *obstacle thresholds*. A preprocessing step searches the routing area, marking obstacle thresholds. Tracks extending into these marked areas make *vacating* jogs to tracks outside these areas.

Another important issue is the tradeoff between horizontal and vertical wiring. Magic has to decide whether to route horizontal wires or vertical wires over single layer obstacles. It can not do both of these, since an obstacle and a wire

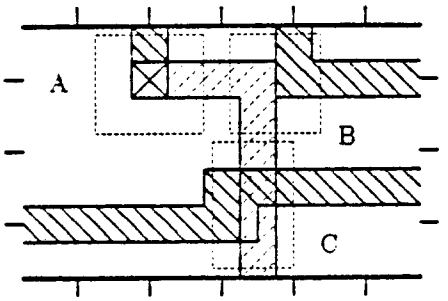

Figure 4. Tracks can not extend into blocked areas (drawn in dotted lines). Note that two adjacent areas of different layers (B) form blocks because there is no place to put contacts to bridge from one area to the other.

crossing it block both routing layers. A thin vertical wire should be bridged horizontally by tracks. Likewise, a thin horizontal wire should be bridged vertically by columns. Intermediate cases are harder to classify (Figure 5).

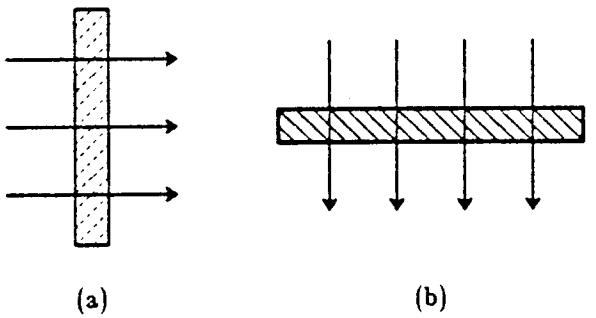

Figure 5. Thin width vertical wires should be bridged horizontally by tracks (a). Thin width horizontal wires should be bridged vertically by columns (b). Intermediate cases are harder to classify.

We solve this problem by always giving priority to horizontal wiring. If vertical wiring is not done in the current column it may be done in some later column. Horizontal wiring is more important: if the router needs to extend tracks but can not, it fails.

Although horizontal wiring gets priority over vertical wiring, we attempt to avoid extending tracks into large single layer obstacles. When tracks do extend into single layer obstacles the Magic router tries to jog them out of these areas, into unobstructed tracks. It is important to do this because a single track running through an obstructed area blocks all columns that might cross the obstructed area (Figure 6).

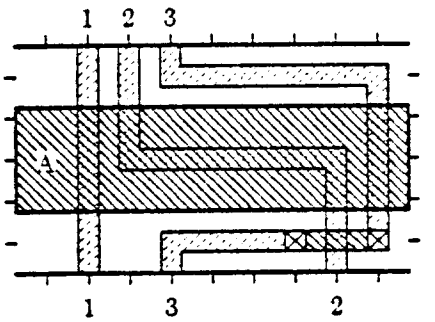

Figure 6. Nets avoid obstructed tracks wherever possible. Failure to do so may create blocked areas. Since net 2 is in an obstructed area, net 3 is forced to make a long detour.

The greedy router assumes that it can make vertical column wiring anywhere the channel is not blocked by vertical wiring it previously placed. The Magic router has to know not only when to place vertical column wiring, but also how to do this. It has to know when areas are blocked, and when to place contacts to switch layers.

Given our wiring model, contact placement is simple. If a contact needs to be placed to allow a layer switch, there is only one place where that contact can go: immediately adjacent to the obstacle. For vertical wiring contacts may be placed immediately above or below the obstacle. For horizontal wiring the locations are immediately to the left and right of the obstacle.

Our wiring model allows horizontal and vertical wiring in either layer; however, only one layer of horizontal wiring and one layer of vertical wiring is allowed at any point. There is a preferred layer in each direction; horizontal tracks and vertical column wires may run in the opposite layer only to bridge an obstacle. Since poly is the preferred vertical layer, a vertical run may bridge a metal obstacle without placing contacts, but contacts need to be placed to bridge a poly obstacle. If the track immediately above the poly obstacle is vacant, then the contact can be placed. If the track is occupied by horizontal wiring, the preferred layer policy says that it must be in metal. The metal/poly boundary blocks the vertical run, since there is no space to bridge the metal track in poly and place a contact before running over the poly obstacle (Figure 7).

The greedy router assumes that channels can be arbitrarily expanded and that terminals on the left and right edges of the channel can "float" up and down

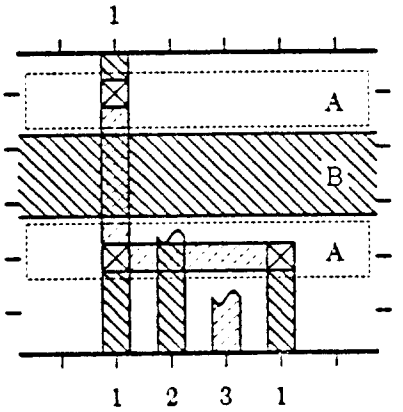

Figure 7. The outlined areas (A) above and below the obstacle (B) are reserved for column contacts necessary if the obstacle is to be bridged vertically. The router tries to keep the areas clear of wiring. Note that the horizontal metal run prevents both the poly (2) and the metal (3) vertical runs from bridging the obstacle, because there is no room to place contacts.

as long as their relative positions remain the same. Tracks may be inserted wherever the router gets "stuck". The Magic router assumes that channels have a fixed number of tracks and that terminals have fixed positions on the edges of the channels. Accordingly, the Magic router omits the greedy router's channel widening step, reporting failure if a net could not be brought into the channel from some top or bottom pin.

5. Routing Switchboxes

The greedy channel router handles pins on at most the top, left, and bottom sides of a channel. To make it a switchbox router, the Magic router contains additional rules to make connections on the right edge of the channel. Furthermore, the Magic router removes the assumption that nets have at most one pin on each end of the channel.

The Magic router deals with switchbox connections by introducing the notion of *reserved* tracks. A track is reserved if it is needed by some net to make a connection on the right edge of the channel. When approaching the end of the channel the router makes vacating jogs to clear reserved tracks and then jogs the appropriate nets into these tracks (Figure 8). Additionally, after nets with only one right edge pin have made their last top and bottom pin connections, their right edge tracks become reserved, other nets vacate these tracks, and the router

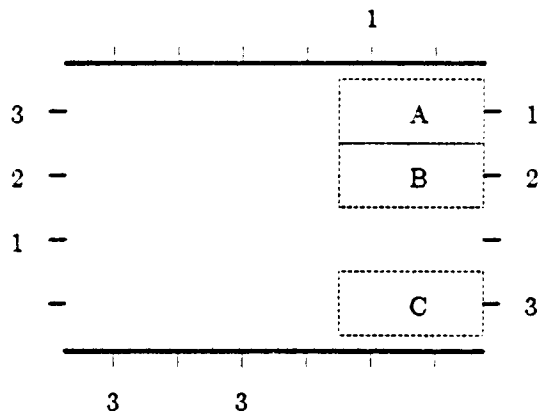

Figure 8. The outlined areas are reserved for nets making connections at the end of the channel. Any other nets entering these areas make vacating jogs, allowing the required nets to occupy the tracks.

tries to jog nets into their final tracks. Vacating reserved tracks uses the same mechanism provided to vacate obstructed tracks.

If a net has more than one pin on the right edge of the channel, the router needs to split the net to connect to them. Split nets occupy tracks that could otherwise be used to help route the channel. Therefore splitting to make multiple end connections is only done when the router gets close to the end of the channel. *Close* is a parameter the user sets to control net splitting. A typical value is two columns.

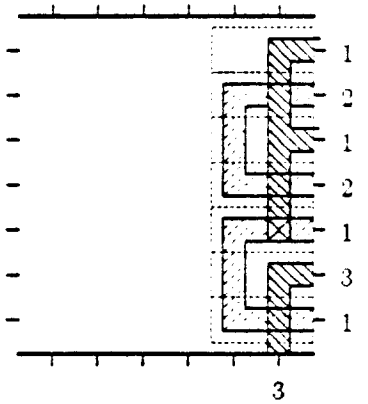

Figure 9. As the router approaches the end of the channel, nets with all of their pins on the right edge of the channel require tracks to be assigned to the nets. This is done if at least two tracks can be allocated and joined with vertical wiring.

Nets with all of their pins on the right edge edge of the channel are another

complication. As the router nears the right edge of the channel it has to decide when to first assign tracks to these *right edge* nets. Since there are no connections to previous pins, a right edge net is introduced into the channel only if it can be assigned to at least two tracks that can be joined by vertical wiring (Figure 9).

We carry this one step further. Groups of two or more tracks for a particular right edge net may be introduced into the channel, even if the groups themselves can not immediately be joined. The task of joining these groups is easier, since the top track of one group need only be connected to the bottom track of a net's higher group.

6. The Magic Routing Algorithm

The Magic router operates in three phases. It begins by making a pre-routing scan of the routing area, identifying obstacle thresholds. After identifying obstacle thresholds, the router extends nets from left edge pins into the routing area and routes it using the column-by-column scan. After routing the channel the Magic router invokes a post processing step to maximize metal and reduce vias.

6.1. Finding Obstacle Thresholds

Obstacle thresholds are generated for all multi-layer obstacles and some single layer obstacles. Multi-layer obstacles such as contacts, crossings, and poly/metal edges must always be avoided as tracks extend from left to right, since it is not possible to bridge these obstacles in any layer. Single layer obstacles extending horizontally for more than one column's width also generate threshold areas. Single layer obstacles extending horizontally for only one column's width do not generate thresholds since the vertical wiring gained in the obstructed area is offset by the vertical wiring wasted in jogging around the obstacle.

Depending on the height of the obstacle, many nets may have to be jogged around it. Not all nets can make vacating jogs in the same column because the vertical wiring for one vacating jog blocks another net from making its vacating jog. On the other hand, vacating tracks long before they near obstacles wastes channel routing area. In recognition of this, the Magic router makes vacating jogs

around an obstacle depending on how far away and how high the obstacle is. Higher obstacles, which block more tracks, cause nets to start vacating jogs farther away, while shorter obstacles can be approached more closely before vacating jogs commence. The width of the threshold is the product of a parameter, *obstacle threshold constant*, and the height of the obstacle. This parameter allows some control over how soon the router attempts to vacate obstructed tracks. A typical value for this parameter is 1.

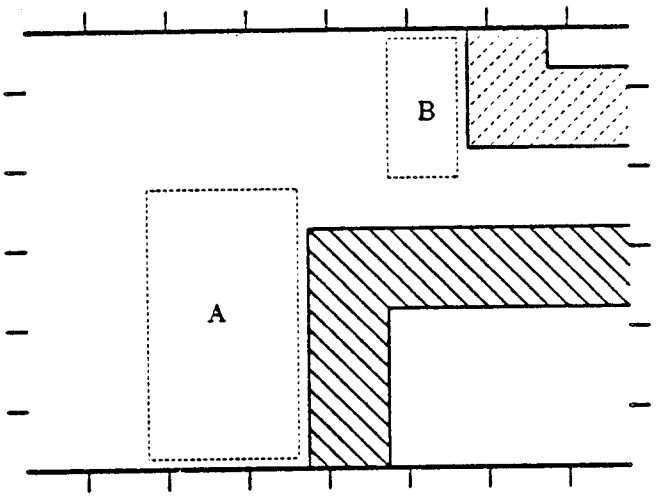

Figure 9. Taller obstacles may require more nets to vacate their thresholds; therefore taller regions have wider thresholds.

The obstacle threshold also extends one track above and below the obstacle. Nets do not get assigned to these tracks unless no other track is free. This allows contacts to be placed if vertical wiring has to switch layers to bridge the obstacle (Figure 7).

6.2. Wiring Rules

This section presents the set of rules the Magic router uses to control the placement of contacts and vertical jogs. The following discussion omits details that are identical in the greedy router. The rules are:

a. *Place Track Contacts*: As the first step in wiring a column, place a contact in each unobstructed track, if either the next column or the previous column has an obstruction in the preferred horizontal track layer. The contact serves one of three purposes: (a) it switches the net from the preferred

horizontal track layer (metal) to the alternate layer (poly) when the net enters a river-routed region; (b) it switches the net from the alternate layer back to the preferred horizontal layer when the net leaves a river-routed region; or (c) it switches the track to the preferred vertical layer in preparation for jogging the net to another track.

b. *Make Minimal Top and Bottom Connections*: Do not bring a net into an unobstructed track that is blocked in the next column. This step may bring a net into an obstructed track. If this occurs, step (f) will attempt to jog the net to an unobstructed tracks. Report failure if some net could not be brought into the channel.

c. *Collapse Split Nets.*

d. *Reduce the Range of Tracks Assigned to Split Nets*: Do not move a net from a free track to a track that needs to be vacated.

e. *Raise Rising Nets and Lower Falling Nets*: Do not jog from a free track to one that needs to be vacated.

f. *Vacate Obstructed Tracks*: Identify tracks from which nets should be vacated. These are tracks which are either in the threshold of an obstacle or are reserved to make some end connection. Try to vacate to the nearest empty, unobstructed track. Do not vacate to another obstructed or reserved track unless the source track is blocked (ie. runs into a multi-layer obstacle) and the destination track is not blocked. Give preference to vacating jogs that move rising and falling nets closer to their next pin.

g. *Split Nets to Make Multiple End Connections*: If within *channel end constant* columns of the end of the channel, attempt to split nets to make multiple connections at the end of the channel. This is the opposite of the collapsing step c above. The best pattern is that which splits the most tracks.

h. *Extend Active Tracks to the Next Column*: Report an error if some track is prevented from extending into the next column by the presence of a multi-layer obstacle that could not be avoided.

6.3. Metal Maximization

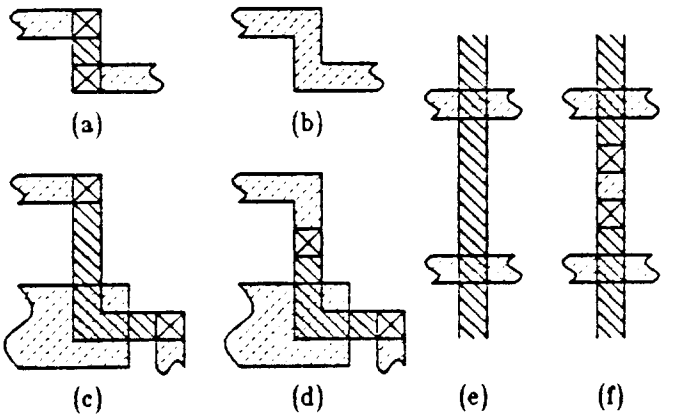

Figure 11. A postprocessing step maximizes metal. This may delete or move vias. It may also introduce vias.

After the subchannels are routed, the Magic router concludes with a metal maximization step. (Figure 11). Since the router already routes metal horizontally wherever possible, this step replaces vertical wiring in polysilicon with vertical wiring in metal, subject to constraints imposed by obstacles in the channel. Vias are deleted wherever they become unnecessary.

7. Channel Splitting

The Magic router also extends the greedy router by including a channel splitting feature. It splits a channel in two at a point of maximum density, assigns tracks to nets crossing the split, then routes both subchannels outwards from the column of the split. The intent of channel splitting is to improve the routability of the two resulting subproblems by (1) assigning tracks to the nets crossing the split to remove conflicts between vertical wiring, and (2) removing split nets at the column where the channel is divided, to guarantee that there are enough available tracks to accommodate the nets that must cross this column Channel splitting is done if the length in columns of each of the resulting subproblems is greater than or equal to the parameter *minimum channel size*, and if the density of the routing problem is close to the size of the channel. If the channel can not be split, then the router routes it from left to right or from right to left, at the discretion of the user.

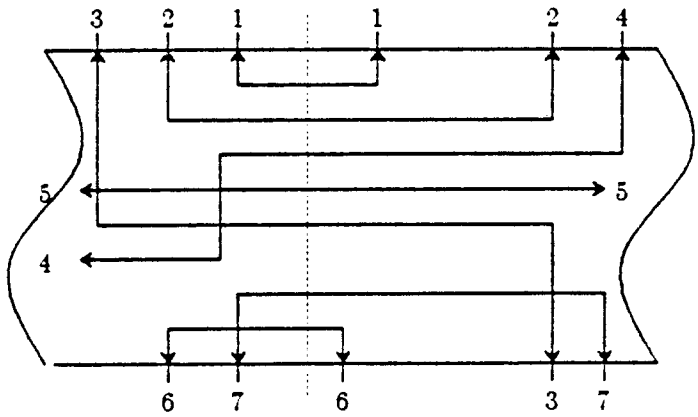

Figure 12. To increase the routability of the two subchannels the router assigns tracks to nets crossing the split. Nets are ordered based on their rising/falling status and the distance to their closest left and right pins.

Channel splitting is not recursive -- it is done at most once. The idea is to route away from the point of maximum density. Splitting each subchannel at its point of maximum density would result in subchannels routing from one highly constrained region to another.

After deciding where to split the channel, the Magic router assigns tracks to the nets crossing the split. The ranking procedure assigns each net a ranking number which is the average of the distance from the center track of the channel to the net's target tracks in the left and right subchannels. The top tracks go to nets which rise to pins on the top edge of both subchannels. The bottom tracks are assigned to nets which fall to pins on the bottom edge of both subchannels. All other nets, including those rising or falling an intermediate distance, and those steady in both subchannels, get distributed between the first two groups.

Another discriminator is used among nets rising to the top or falling to the bottom of both subchannels. A net *a* ranks above another net *b* if both *a*'s nearest left pin and its nearest right pin are closer to the split column than *b*'s corresponding pins. If the distances overlap (ie. *a*'s left pin is closer than *b*'s, and *b*'s right pin is closer than *a*'s), then the net with the smaller sum of distances is placed above the other. A similar procedure is used for falling nets. The intent is to order the nets to eliminate crossings wherever possible. If nets must cross, this procedure favors the net traveling the shorter distance.

8. Implementation and Performance

For channels without obstacles the Magic router produces results similar to those produced by other good channel routers such as the hierarchical router [BuP], the greedy router [RiF], and Algorithm #2 [YoK]. In spite of omitting the track insertion step from the greedy algorithm, it routes Deutsch's difficult in the same number of tracks as the the greedy router. The results are summarized in Table 1.

Router	Tracks	Vias	Wire Length	Time (sec)	Machine
Magic (no obstacles)	20	376	4099	1.5	DEC VAX 11/780
Magic (with obstacles)	20	376	4099	3.0	DEC VAX 11/780
Algorithm #2	20	-	-	2.1	DEC VAX 11/780
Greedy	20	347	4150	7.93	DEC KA-10
Hierarchical	19	270	3983	24	IBM 370/3033

Table 1. Router Results for Deutsch's Difficult Example

Most of the numbers in Table 1 were taken from [BuP]. The first table entry refers to our implementation of a modified greedy switchbox router before obstacle avoidance was added. The reported number of vias for the Magic router does not show the results of metal maximization.

The table shows that the Magic router is competitive with other channel routers on conventional routing problems. It produces nearly optimal solutions quickly, which may be more valuable in practice than programs such as the Hierarchical router which produce slightly better results after significantly greater computation. Adding obstacle avoidance nearly doubled the running time of our router.

Our figures provide a good comparison between Yoshimura and Kuh's Algorithm #2 and Rivest and Fiduccia's greedy router. Rivest and Fiduccia's router

was implemented in LISP on a KA-10. The Magic router without obstacle avoidance (which is almost identical to the greedy router) is implemented in the *C* programming language. Algorithm #2 is implemented in FORTRAN. Both the Magic router (without obstacle avoidance) and Algorithm #2 run on VAX 11/780s running Berkeley Unix. The early version of our router runs faster than the already fast Algorithm #2, and produces a result using the same number of tracks.

Experience with channel splitting has so far been disappointing. It has turned out to be useful mostly for assigning crossings in river routed regions. In other cases splitting the channel typically increases the number of tracks required to route the channel. Better rules for ordering the nets crossing the boundary between the subchannels might change this. Another idea would be to use different criteria to decide where to split the channel.

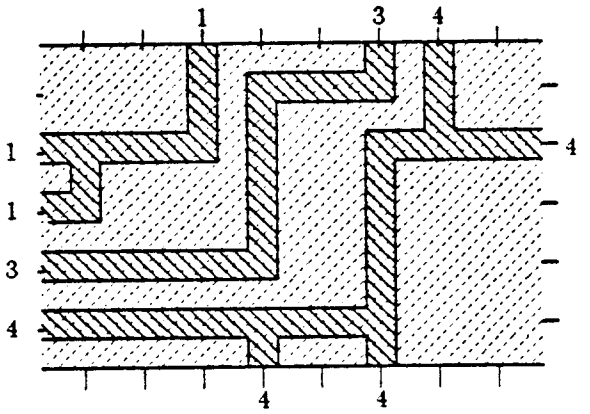

Figure 13. The Magic router river-routes in areas completely blocked in a single layer.

As an example of the range of problems handled by the Magic router, Figure 13 shows a channel completely covered with metal. Our router does a reasonable job of routing this problem.

Postprocessing to increase metal and remove vias appears to significantly improve the quality of the routing.

9. Conclusions

Our obstacle avoiding channel router adds flexibility to our design environment. It allows designers to route critical signals by hand or with separate routing steps. After critical signals are routed, the router makes the remaining connections.

The Magic channel router provides this obstacle avoiding capability, while also considering tradeoffs and interactions between nets. It accomplishes this using a rule based, column sweep routing algorithm which is simple, flexible, and fast. The simplicity of this approach makes it an attractive vehicle for further experimentation.

10. Acknowledgements

Robert Mayo, Walter Scott, and George Taylor all participated in discussions resulting in this work and provided comments on drafts of this paper. Mark Hill, Randy Katz, Carlo Sequin, and David Wallace also reviewed drafts and provided helpful comments. The work described here was supported in part by SRC under grant number SRC-82-11-008.

11. References

[BuP] Burstein, M., and Pelavin, R., "Hierarchical Channel Router", *Proc. 20th Design Automation Conference*, Miami (1983)

[Che] Chen, H., Private communication with authors.

[CHK] Chen, N. P., Hsu, C. P., and Kuh, E. S., "The Berkeley Building-Block Layout System for VLSI Design", ERL memo UCB/ERL M83/10, University of California at Berkeley, (Feb. 1983).

[Lee] Lee, C. Y., "An Algorithm for Path Connections and its Application", *IRE Transactions on Electronic Computers*, pp. 246-365 (September 1961).

[Hig] Hightower, D., "A Solution to the Line Routing Problem on the Continuous Plane", *Proceedings Design Automation Workshop*, pp. 1-24, (1969).

[OHM] Ousterhout, J. K., Hamachi, G. T., Mayo, R. N., Scott, W. S., and Taylor, G. S., "Magic: A VLSI Layout System". In this technical report.

[Riv] Rivest, R. L., "The 'PI' (Placement and Interconnect) System", *Proc. 19th Design Automation Conference*, Las Vegas (1982).

[RiF] Rivest, R. L., and Fiduccia, C. M., "A Greedy Channel Router", *Proc. 19th Design Automation Conference*, Las Vegas (1982), pp. 418-424.

[Sou] Soukup, J., "Circuit Layout", *Proceedings of the IEEE*, Vol. 69, No. 10 (Oct. 1981), 1281-1304.

[YoK] Yoshimura, T., and Kuh, E. S., "Efficient Algorithms for Channel Routing", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*", Vol. CAD-1, No. 1, (Jan 1982).