Magic's Obstacle-Avoiding Global Router

Gordon T. Hamachi and John K. Ousterhout
Cornputer Science Division
Department of Electrical Engineering and Computer Science
University of California at Berkeley

ABSTRACT

This paper describes the global router portion of Magic's obstacle-avoiding routing system. In addition to choosing sequences of channels between terminals, the global router chooses crossing points between channels. A rule-based penalty function considers obstacles and other potential problems to pick crossings that will make channel routing as easy as possible. Where no satisfactory crossing points are available between two channels, the global router will attempt to route nets through different channels in order to avoid the problem area.

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

1. Introduction

The Magic IC layout editor contains an automatic routing system that makes interconnections between subcells in custom layouts [Ousterhout84]. The router's most unusual capability is its ability to work around pre-existing material in the routing channels. We call this material *obstacles*. The obstacle-avoidance feature provides a much more flexible routing environment since it allows designers to pre-route critical nets by hand and then invoke the automatic router to make the

rest of the connections. Similarly, nets with special routing requirements, such as power and ground, may be pre-routed with special-purpose routing tools, and the Magic router can make the remaining connections.

Magic's routing system divides the routing process into three steps. In the first step a channel decomposer divides the areas between subcells into non-overlapping rectangular channels. In the second step a global router determines which sequence of channels each net passes through to connect its terminals. In the final step a channel/switchbox router assigns physical locations to the wires in each channel, realizing the signal routings specified by the global router. Both the global router and the channel router must consider the presence of obstacles. A previous paper has already described the channel router [Hamachi84]; this paper focuses on the global router.

A global router's task is to consider tradeoffs between channels, choosing routings through sequences of channels to simplify the channel router's task. Since long wires occupy space that could be used to route other signals and also introduce resistance and capacitance in circuits, a global router usually chooses the shortest path from one point to another. In some cases, however, it may choose a less direct path to avoid routing through congested channels whose wire capacities are already fully used.

Obstacles complicate the routing problem. In some places, they may occupy all of the routing layers; such areas will have to be avoided completely by the channel router. Single-layer obstacles limit the choice of routing layer; they may make the channel router's job difficult or impossible. The global router must ensure that the channel router will have enough space to route around obstacles, and must take the obstacles into account when computing the capacities of channels. In some cases, it may be better to choose a longer path than to attempt to pass through an area with many obstacles.

To operate in this obstacle-avoiding routing environment, the Magic global router chooses not only the sequence of channels for each net, but also the exact points (and layer) where nets cross from channel to channel. These points are called *crossings* and the selection process is called *crossing placement*. Crossing placement was originally introduced in the PI system [Rivest82] as a separate step occurring after global routing. Unlike the PI system, Magic's routing system performs crossing placement during global routing. As it generates shortest paths for global routing, it evaluates crossing points along channel-to-channel boundaries to assess the the effects of obstacles. This allows the global router to select crossings that result in the simplest channel-routing problems. In cases where all of the available crossings on a channel-to-

channel boundary are undesirable, the global router will consider alternate, "next-shortest" paths of channels to use for the net.

2. Global Router Overview

Magic's global router takes as input a cell placement, a collection of channels, and a net-list specifying desired connections between pins on cell edges. The cell placement is determined by the designer, and is not modified by the routing tools. The channel structure is determined automatically by the channel decomposer. The net-list is provided by the designer, and contains one or more nets, each of which is a set of terminals to be wired together. The global router outputs a set of channels ready to be routed by our switchbox router. After global routing, channel-to-channel crossing points for each net are completely specified. See Figure 1 for an example.

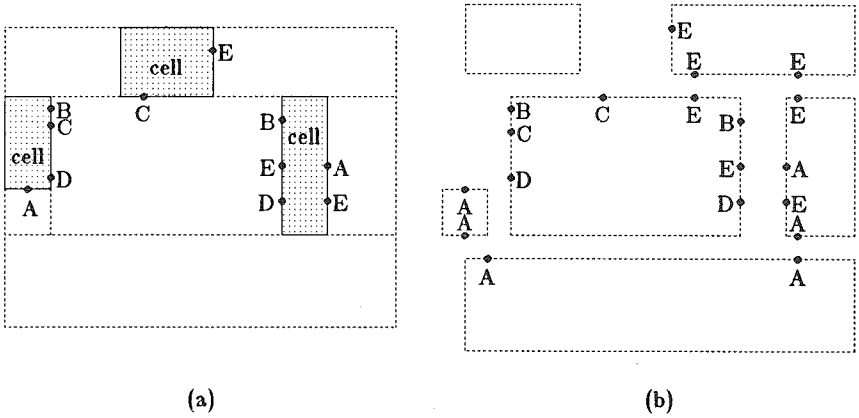

Figure 1. The global router is given a set of cells, a set of channels, and a net-list, as in (a). It produces a set of channel routing problems, as in (b).

The global router processes nets sequentially. It makes a rough estimate of the wire length of each net and sorts the nets according to the estimates (Section 6 discusses the sorting in more detail). Nets are then processed one-by-one, shortest net first. After the global routing for one net is completed, the next net is then considered.

Each net is processed in two steps, involving shortest-path generation and crossing placement. The process is outlined in Figure 2. For most of the discussion that follows, each net is assumed to have two terminals. Section 7 generalizes the algorithm to handle multi-pin nets. The global router begins by finding the sequence of channels through which to route the net in order to minimize wire length. As it creates

```
0.  sort nets;
1.  for each net
2.  { source points = (first pin);
3.    for each additional pin in the net
4.    {     repeat
5.          { find the (next) shortest path to target pin;
6.            if (path length > best total cost so far)
                    break;
7.            adjust crossings;
8.            path total cost = path length + crossing penalties;
9.            if (path total cost < best total cost so far)
                    save new path;
              }
10.         source points+= new path points;
      }
    }
```

Figure 2: Overview of the global router. Lines 3 and 10 are used for multi-pin nets and are explained in Section 7.

this shortest path, it makes an initial crossing placement by choosing crossing points that will minimize the net length.

After the shortest path has been found, the global router re-examines the initial crossing points that were chosen for the path. Each crossing is penalized for nearby obstacles and other undesirable features that will complicate the task of the channel router. The penalties are computed as distances. A penalty of 10 units means that it is preferable to choose a path that is up to 10 units longer in order to avoid this crossing. Where penalties are assessed, the global router will examine other nearby crossings and replace the initial crossing choice with the crossing whose penalty is smallest. Section 5 describes how the penalties are computed.

After finding the best crossings along the path, Magic adds all the penalties for those crossings to the wire length for the path and uses this as the *total cost* of the path. The desirability of a path thus depends on both its length and the quality of its channel-to-channel crossings. The shortest path may not be the best one if it has particularly bad crossings. For this reason, the global router does not stop with the shortest path. Instead it generates more paths, next-shortest-first, in the hope that it will find one whose crossings are attractive enough to give it a lower total cost. Since paths are examined shortest-first, the algorithm can terminate when a path is

found whose length alone is greater than the best total cost seen so far. When the global router finds the lowest cost path for a net, it marks the path's crossing points as "in use" by that net, and then proceeds to the next net. Section 4 gives a more detailed discussion of this approach to crossing placement.

3. Shortest Path

One of the global router's key components is a multiple-source, single-destination shortest-path generator. The shortest-path generator is somewhat like Lee-type algorithms [Lee61] in that it extends a wavefront outwards from source to destination. However, there are two significant differences between the Magic router and the standard Lee approach. First, the Magic router extends the wavefront one channel at a time [Clow84] instead of one grid unit at a time. Second, Magic's shortest-path generator is directed: instead of extending the wavefront uniformly on all sides, it first explores the most direct route to the destination. These two differences result in a dramatic speed-up over the conventional Lee approach. This search technique has been variously characterized as a branch and bound search or as Dijkstra's shortest-path algorithm [Horowitz78] with a path length estimate substituted for the distance traveled.

The shortest-path algorithm constructs a number of partial paths through sequences of channels from the source point toward the destination point, as shown in Figure 3. It stores these partial paths in a heap, sorted by estimated path length. The estimated length for a partial path is the actual distance travelled from the source point plus the Manhattan distance between the path's endpoint and the destination point. The estimated distance is a lower bound on the path length: the actual distance to the destination may be greater than the estimate if there are cells blocking the most direct path (see Figure 3).

On each iteration the shortest-path algorithm removes the most promising partial path (i.e. the one with shortest estimated length) from the heap. If the path's endpoint is in the destination point's channel, then the shortest-path algorithm terminates successfully, returning this path. Otherwise, the algorithm extends the path in all possible directions and estimates the length of each new path. The new partial paths are added back into the heap in sorted order, and the process repeats. If the heap ever becomes empty then no partial path can be extended, so the search terminates unsuccessfully. Figure 4 summarizes the algorithm.

This procedure guarantees that the first path to reach the destination channel is the shortest. To see this, recall that the algorithm always extends the path whose estimated distance (distance

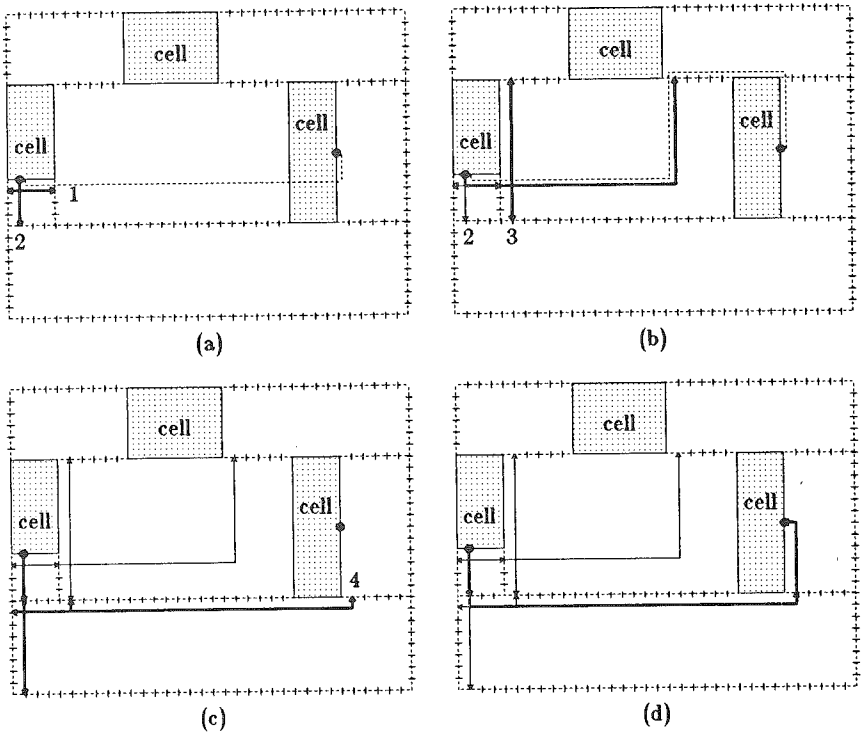

Figure 3. The shortest-path algorithm expands a search wavefront from the source point towards the destination. In each step it extends the most promising path, which is shown with dark lines. In (a), three partial paths are generated from the source point. Path 1 appears to be the most promising: it has the smallest sum of distance traveled from the source and estimated distance to the destination (the dotted line shows the distance used for comparison; in this case the estimate is optimistic since a subcell blocks the most direct path to the destination). In (b), path 1 is extended outward one channel in all directions. The dotted line in (b) shows the distance estimate for one of the new paths. The estimated costs for all of the new paths are at least as great as for path 2, so path 2 is extended in (c) (the choice between it and path 3 is arbitrary).. In step (d) the most promising path reaches the destination, thereby ending the search.

travelled plus Manhattan distance to the destination) is minimum. When a path reaches the destination channel, its length is no longer an estimate: it is exact (there cannot be any cells blocking its path to the destination). All the other points on the heap have estimates at least as great as this, and their estimates are lower bounds on the exact lengths. Thus they cannot end up with shorter path lengths than the one that has already reached the destination channel.

```
1.  for each source point
2.    HeapAdd(pin, ManhattanDistTo(target));
3.  repeat
4.  { P = HeapRemoveTop();
5.    if no more points
            failure;
6.    if channel(target point) contains P
            success;

7.    for each channel C1 adjacent to channel(P)
8.          if !(loop created)
9.          {  P1 = closest available crossing into C1;
10.           HeapAdd(P1, Dist(P) +
                  ManhattanDistTo(P1) +
                  ManhattanDistTo(target));
            }
    }
    until success or failure;
```

Figure 4: The shortest-path algorithm repeatedly extends the most promising partial path toward the target point, one channel at a time.

Crossing locations are chosen by the shortest-path generator and later modified by the crossing placer. In the shortest-path stage of global routing, each new crossing is made as close as possible to the previous crossing in the path, subject to the availability of crossing points. This approach puts off additional wire length as long as possible and guarantees that the path length estimates are lower bounds. At a later stage of global routing the crossing locations will be reconsidered in order to avoid obstacles, eliminate jogs, or otherwise simplify the channel routing problems.

4. Crossing Placement

Magic's global router is unusual in that it does crossing placement during global routing. Other routers assign crossings after all global routing is complete, either by letting the channel router assign them or by executing a separate crossing-placement step before channel routing. Magic's mechanism has advantages over each of these other approaches.

The most common approach to crossing placement is to let the channel router assign crossings. When a given channel is routed, all of its crossings are fixed during the routing of the channel. The channel router is free to place unassigned crossings anywhere along the relevant

channel-channel border, but it must accept any crossings fixed while routing previous channels. This approach has the advantage of leaving the channel router maximum flexibility to choose crossings in the most convenient place. Also, in some situations the channels can be routed in order so that channels have fixed crossings only on their sides and never on their ends. In Magic, the channel router must handle channels with fixed crossings on all four sides.

A disadvantage of choosing crossings during channel routing is that it tends to simplify the problem at hand without considering the solution's global ramifications. The channel router may choose a crossing point that is convenient for the channel being routed but extremely inconvenient for the channel on the other side of the crossing; once the crossing is fixed, it will have to be used when the adjacent channel is routed. For example, if the channel router chooses a crossing adjacent to a large obstacle, it may be difficult or impossible to route the signal around the obstacle in the next channel.

The PI system [Rivest82] and the VTI Composition Editor [Ng84] perform crossing placement as a separate step between global routing and channel routing. They use global information during their crossing placement steps to adjust the channel crossings on the paths established during global routing. The aim is to maximize the number of crossings that face each other across channel edges so that the channel router will not need to insert jogs.

A problem with this approach is that the sequence of channels traversed by each net is fixed before crossing placement occurs. When actual crossing locations are established, it may turn out that many of the crossings are undesirable because of obstacles. The router will not be able to consider alternative global routing paths to avoid the undesirable crossings, so it may have to use some of the bad crossings. This may make channel routing difficult or impossible.

Magic's crossing placement scheme avoids the problems with the above approaches. By assigning crossings during global routing, the characteristics of adjacent channels can be considered in order to choose a crossing that makes each of the channel-routing problems as simple as possible. Since the global router repeatedly generates paths and adjusts crossing until the lowest cost path is found, it can consider alternate channel paths during the global routing for a particular net.

The Magic approach has other advantages over the above approaches. Since it knows where nets cross channel boundaries, it can make better estimates of path length. During global routing, the PI system assumes that nets all cross at the centers of channel-to-channel boundaries. It can result in large overestimates of path length, as shown in Figure 5. [Ng84] reports that this technique produces poor

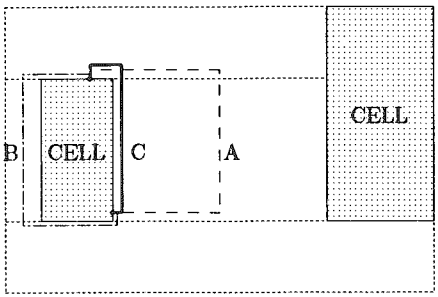

Figure 5. If net lengths are estimated using the centers of channel-to-channel boundaries, unnecessarily long paths may be chosen. In this example, path B appears to be the shortest path if distances are measured using boundary centers. However, path C, which Magic will use, is actually the shortest route between the two points.

global routes.

Another advantage of the Magic approach is that it can avoid overcommitting areas of channels. If actual crossings are assigned, the global router can maintain tables of local density for each column and track within each channel, and refuse to route nets through areas that are overcommitted. Without specific information about channel crossings only rough estimates of density can be made, so it is easier for global router to overcommit an area. Note: we have not yet added local density checks to the Magic router, although we plan to do so in the near future.

One of the main goals in crossing placement is to permit straight-across routing in channels by choosing crossings that face each other. In order to make as many facing crossings as possible, each crossing must be chosen with knowledge about the previous and next crossings in the net. This information isn't available until the shortest-path step is complete (when extending a path to a new crossing, the next crossing's location is not known). Thus, crossing placement cannot be done effectively during the shortest-path step of global routing.

Therefore, crossing placement is a two-step operation. During shortest-path generation, initial crossings are chosen to minimize net length. Once the complete path of channels is established for a net, the router makes an additional scan over the crossings in the path. Each crossing is reconsidered in light of nearby obstacles, the locations of the preceding and following crossings in the net, and other information that is described in Section 5. If the crossing is undesirable, it may be replaced by a nearby crossing that is more desirable, even if the other crossing results in a longer path length. Thus, crossing placement

involves a tradeoff between net length and the difficulty of the resulting channel routing problems.

When reconsidering a crossing, the router first computes a penalty for the initial crossing, with the penalty expressed in units of distance. This penalty gives the maximum amount of additional net length it is worth incurring to avoid the problems at the crossing. If the penalty is not zero, then the global router examines crossings on either side of the initial crossing. Penalties are computed for each of these crossings, which also include the additional distance that will have to be travelled to use the crossings, and the lowest-cost crossing is chosen to replace the initial crossing. The penalty for the initial crossing limits how far to either side the router needs to look to find the best crossing. Figure 6 outlines the algorithm.

```
1.  crossing = closest pin;
2.  penalty = evaluate(crossing);
3.  if (penalty == 0)
        done; /* Nothing else could be better */
4.  best so far = crossing;
5.  current penalty = penalty;
6.  for (dist = 1; penalty <= jog_penalty(dist); dist += 1)
7.  { if(Save_Best(crossing - dist) == 0) done;
8.    if(Save_Best(crossing + dist) == 0) done;
    }

9.  Save_Best(crossing)
10. { new penalty = evaluate(crossing);
11.   if (new_penalty < current penalty)
12.   { best so far = crossing;
13.     if (new penalty == 0)
            return(0); /* Done. Nothing else can be better */
14.     current penalty = new penalty;
      }
15.   return(current penalty);
    }
```

Figure 6: The crossing placement algorithm uses the crossing penalty at its initial crossing to define the limits of its search for a lower cost crossing..

5. The Penalty Function

The global router's penalty function uses a number of heuristics to estimate the effect of a particular crossing upon the overall channel routing problem. The penalty function consists of a collection of rules, each of which assesses a distance penalty for a particular class of undesirable features at crossing points. Larger penalties are assessed for features likely to have the greatest impact upon the routability of specific channels. Casting channel routing complexity into units of distance allows the global router to make tradeoffs between different combinations of length and complexity, and also different types of complexity, as it selects crossing points between channels.

Since the crossing placement algorithm uses the crossing penalty at its initial crossing to define the limits of its search for a lower cost crossing, each of the following penalties may be regarded as a specification of how far to each side of the original crossing the crossing placement algorithm should look in an attempt to find a better crossing. The penalty function for a particular channel-to-channel crossing considers a number of features described in the subsections below: obstacles, jogs, crossing density, and proximity to the corner of a channel.

Although we believe that the general framework is sound, the relative weights of our heuristics, as well as the rules themselves, are tentative. The specific rules and penalty values are likely to change as we get more experience with the system.

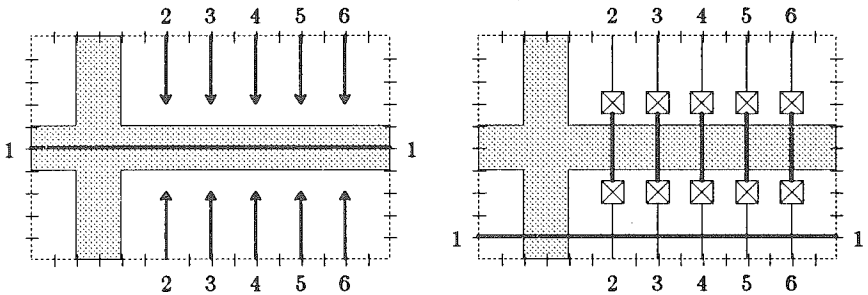

Figure 7. Wherever possible, the routing tools attempt to route across single-layer obstacles in the narrow direction. In this example, if net 1 were routed across the length of the obstacle, as in (a), all of nets 2, 3, 4, 5, and 6 would be blocked. By routing 1 around the obstacle, the other nets can cross the obstacle in the narrow direction and all the nets can be routed.

5.1. Obstacles

Several rules are concerned with avoiding obstacles. Obstacles consist of hand-routing and other mask material in the routing channels. They come in two forms: double-layer and single-layer. Double-layer obstacles occupy both of the routing layers. There is no way to route any additional wires over those areas, so the router must avoid them completely. Single-layer obstacles occupy one of the routing layers; the router can still run wires over single-layer obstacles if it uses the other routing layer. However, the router normally expects to be able to use both routing layers for different signals, one horizontal and one vertical, so single-layer obstacles restrict the router's options: it can only route in a single direction over them.

When choosing which way to route over single-layer obstacles the global and channel routers try to route across the narrowest dimension of the obstacle, as shown in Figure 7. This creates the least amount of two-layer blockage. If a signal would have to cross an obstacle along its long dimension, the routing tools try instead to jog the signal around the side of the obstacle so that other signals can bridge both the jogged signal and the obstacle.

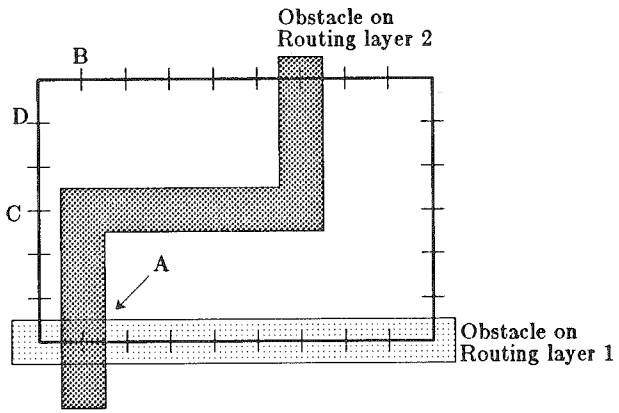

Figure 8. Crossing A is *blocked* by obstacles on both routing layers and cannot be used. All of the other crossings at the bottom of the channel are *covered* by a single-layer obstacle. Crossings B and C are *hazardous*, and crossing D is *obstructed*.

The global router has four rules that deal with obstacles, illustrated in Figure 8. The first rule concerns crossing points that are covered by two-layer obstacles. These are called *blocked* crossings and cannot be used at all. They receive an infinitely high penalty.

The second rule deals with *covered* crossings. These are crossings that lie underneath single-layer obstacles. The goal is to penalize the

crossing a lot if a wire connecting to that crossing would pass over the long dimension of the obstacle, and to use a smaller penalty if the wire would pass across the narrow dimension of the obstacle. The rule assesses a penalty equal to 3 times the width of the obstacle. Among covered crossings, this favors those covered by the narrowest obstacles.

Even if a crossing is not covered, it may be so close to an obstacle that the channel router will not have enough space to jog the wire around the obstacle. Such crossings are called *hazardous*. A crossing is hazardous if a wire running through that crossing is likely to have to cross a two-layer obstacle or the unfavorable dimension of a single-layer obstacle. How close an obstacle has to be to a crossing to make it hazardous is a function of the dimensions of the obstacle. The larger the obstacle, the farther away it can be while still affecting a crossing. Hazardous crossings are assessed a penalty of 3 times the obstacle size minus the distance to the obstacle. Among hazardous crossings this favors ones with the narrowest obstacles, and those whose obstacles are furthest away.

Finally, a crossing point is termed *obstructed* if a straight run through that point and across a channel would cross an obstacle in the channel. These are simply assigned a penalty of 5.

5.2. Jog Avoidance

Another of the most important goals of crossing placement is to allow straight-across routing through channels wherever possible. This involves three rules. The first rule penalizes crossings that are not facing. If a signal passes from one side of a channel to the opposite side, the global router attempts to use crossings that are directly opposite each other. The penalty function assesses a fixed penalty of 5 to a crossing if it is on the opposite side of the channel from the previous crossing but is not directly across from the previous crossing.

The second jog rule preserves facing pairs of crossings when making turns. If a net enters a channel through one side and turns to leave the channel through an adjacent side rather than the facing side, then the router attempts to use crossings whose opposite numbers are already in use. See Figure 9. If a facing pair of crossings is broken up by a turning net, a penalty of 3 units is assessed.

The third jog-elimination rule is also applied when a net turns within a channel. When this occurs, the penalty function looks ahead at the next channel. If the next channel has a straight-through run, the router attempts to enter that channel at a crossing-point whose facing crossing is available, so that there won't be any need to jog within the channel. Any crossing whose opposite number is busy is assessed a penalty of 3 units. See Figure 10.

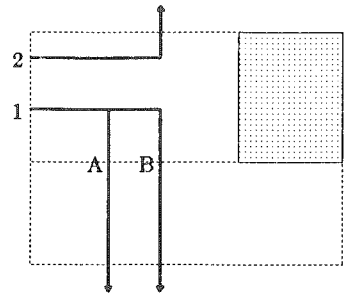

Figure 9. When choosing a crossing for net 1, the penalty function will penalize crossing point A, since it splits up a pair of facing crossings. Since the crossing opposite B is already in use by net 2, there is no penalty for using B.

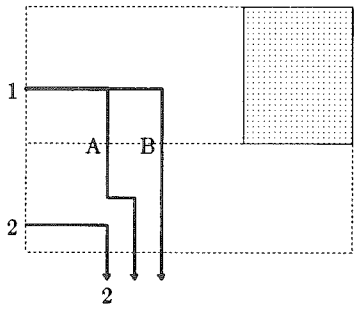

Figure 10. When net 1 makes a turn in the top channel, the penalty function will look ahead to the next channel. If the net runs straight through that channel, it will attempt to find a crossing whose opposite number is still available, as is the case with crossing B. Crossing A will be penalized since it would require a jog in the lower channel.

5.3. Pin Density and Corners

Our channel router, which is derived from Rivest and Fiduccia's greedy router [RiF82], can produce better routing if the occupied crossings are spread out over the length of the channel, rather than bunched together. For example, if only every other crossing is used then the router can virtually guarantee good routing. For this reason, the penalty function contains a rule that penalizes a crossing 5 units if both adjacent crossings are already in use, and 3 units if only one of the adjacent crossings is in use. This rule is most useful when channels are

undercommitted.

Crossing points near the corners of channels also cause problems for Magic's channel router. For example, switchbox connections are harder to make if vertical wiring space in the last few columns must be used to make top and bottom connections. Therefore, a crossing that is the nth closest to the end of the channel is assessed a penalty of 5-n units. Only the five crossings at each end of the channel are penalized in this way.

6. Net Ordering

The global router processes shorter nets first. Before path generation, Magic's global router sorts nets according to the sum of the height and width of the bounding boxes containing their terminals. The height plus width of the bounding box containing a net's terminals approximates the length of the wires needed to connect the terminals. For two-pin nets (the most typical case), this number represents the Manhattan distance between pins.

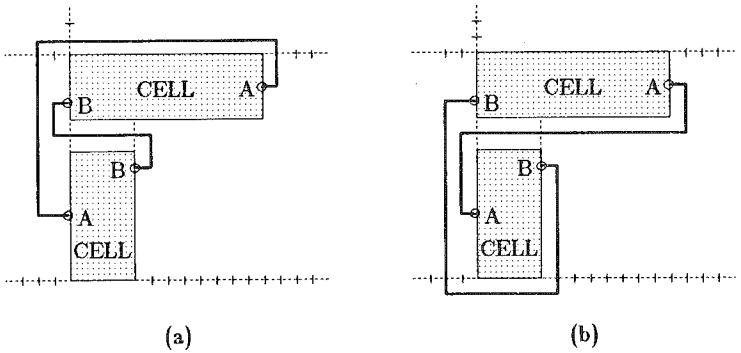

Figure 11. Shortest nets should be routed first, as in (a). If net A is routed first, the result is increased overall wire length (b).

This routing order for nets can have beneficial effects on the resulting wiring. Channel congestion created early during global routing may cause later nets to take less direct global routing paths; the same sized detour is a smaller percent increase for a longer net than for a shorter net. Shorter nets are less likely to interfere with each other; since longer nets cover more area, they are likely to interfere with many other nets. Designers may group cells during placement to reduce interconnection delays for critical paths; by routing the shortest nets first, the global router gives preferential treatment to these paths. The shorter nets will also get first choice of crossing points, while longer

nets will use whichever crossings are available at the time they are routed.

While not addressed directly, Magic's obstacle-avoiding global router provides indirect support for other net orderings. If designers want to give priority to arbitrary nets, they are free to pre-wire them in any manner and then invoke Magic to make the remaining connections. This pre-wiring can be done by hand, with special-purpose routing tools, or even using Magic's automatic routing system.

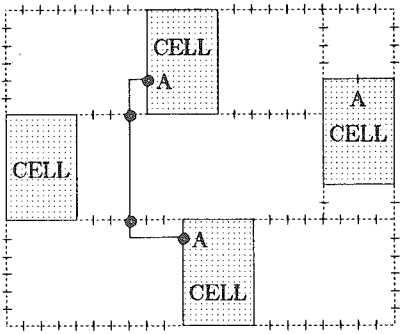

Figure 12. After the first two terminals of net A are connected, those terminals and the channel crossing points are all used as starting points to connect to the third terminal in the net.

7. Multi-Pin Nets

Although the shortest-path algorithm described in Section 5 dealt only with two-pin nets, it extends naturally to handle nets with three or more pins. The resulting algorithm generates a crude approximation to Steiner trees. If a net has more than two pins, the global router picks any two of them at random and applies the algorithm described above to generate a path from the source to the destination. After crossings have been assigned along this path, the algorithm then processes the third pin. That pin is used as the destination for the shortest-path algorithm. However, instead of searching from a single source, the shortest-path algorithm uses the previous pins, plus all of the crossings that have been assigned to this net, as starting points (see Figure 12). Each of these points is inserted in the heap with a zero cost, and the algorithm proceeds exactly as before: a shortest path will be found from one of these points to the new pin, and crossings will be assigned along that path. If there are still more pins in the net, the algorithm processes each of them in the same fashion, using all previous pins and crossings as starting points.

8. Evaluation

The complete routing system has only recently become usable, so our experience routing real chips is limited to a single Master's project designed by a graduate student at U. C. Berkeley. All of the results presented in this section are based on this one design. A thorough evaluation of the system will have to wait until there are more and larger test cases available. In particular, we don't have enough information to evaluate the specific penalties attached to each of the rules. Consequently, our results are preliminary and only indicative of the general performance of our system.

Our results are based on an 8000 transistor full custom NMOS chip called "Tester". We used Magic to make the global interconnections for Tester, consisting of 153 nets and 198 point-to-point connections between 351 terminals. Hand-placed power and ground routing formed obstacles for the global router to work around.

We measured our crossing placement's effectiveness by comparing results with and without crossing placement. When crossing placement was used the subsequent channel routing produced no bad connections; however, with crossing placement disabled the channel routing produced 14 bad connections. Crossing placement dramatically improves the the resulting routing: It is an essential part of our obstacle-avoiding routing system.

Penalty Disabled	Bad Connections	Paths Considered
None (normal)	0	289
All (no penalties)	14	198
All Obstacle Penalties	13	269
Straight Through Current	2	260
Straight Through Next	1	271
Neighboring Pins	1	269
Avoid Corners	1	276
Paired Orphans	0	272

Figure 13. Results of selectively disabling individual components of the crossing penalty function. The difference between the number of paths considered when all penalties are used and the number of paths considered when a penalty is disabled indicates the additional amount of work the global router does as a result of that penalty.

While processing Tester, the global router took advantage of its ability to select slightly longer paths in order to avoid areas with bad crossings. Of the 198 point-to-point connections, 151 could be made through two or more different sequences of channels. Of these, 13 (8.6

%) connections used lowest-cost paths that were better than their initial shortest paths. Crossing placement was useful even when the global router used the initial shortest path to connect a net. On the Tester chip the global router moved 42% of crossings from their initial locations to reduce penalties and simplify channel routing.

We selectively disabled individual components of our crossing penalty function to measure their effectiveness. The results are summarized in Figure 13.

Obstacle penalties are critical to Magic's success in routing chips containing previously placed hand routing. With obstacle penalties disabled, Magic's router generates 13 bad connections while routing Tester; however, when all hand routing is removed, Magic routes Tester with no errors.

The other penalties had much smaller effects on Tester's routability. In particular, the paired orphan penalty had no effect on the number of bad connections. Since it is intended to make "tight" channels easier to route, we expect this rule to be more significant on chips with less free area than Tester.

Path Length	Nets	Cumulative % Nets	Average Expanded Points
1	75	38	1.00
2	37	57	2.14
3	19	66	3.37
4	20	76	6.50
5	16	84	8.69
6	15	92	9.53
7	8	96	11.38
8	4	98	37.50
9	2	99	49.50
10	2	100	26.50

Figure 14. The number of search points examined versus path length. For example, 15 nets had a path length of 6, and 92 percent of all nets had path length 6 or less. On average, paths of length 6 expanded 9.53 search points, or (9.53 - 6 =) 3.53 unnecessary search points.

The shortest-path algorithm appears to be working well. We instrumented the system to count the total number of search points expanded during shortest-path searching (each extension of an existing path into new channels counts as an expansion), and to count the number of channels in the shortest paths. On average, each initial

shortest path crossed 1.94 channel boundaries, and on average 5.17 points were expanded per shortest path found. Since each channel boundary represents one search point expansion and the search origin represents an additional mandatory search point expansion, this indicates that the shortest path algorithm typically examines only 2 search points not on the actual shortest path.

9. Summary

There are three key aspects to the Magic global router. The first aspect is that it combines crossing placement with global routing. This allows the global router to choose more circuitous paths to avoid particularly bad areas of the layout. Initial results indicate that the router does indeed find situations where it is better to choose a longer route. The second key aspect is the use of a rule-based penalty function. By evaluating the crossings with a set of rules, the global router can apply many different criteria in evaluating crossings. As a result, the Magic crossing placer performs jog elimination as in the PI and VTI systems, and also applies different rules to keep wires away from obstacles and the ends of channels. The third key aspect of the system is its use of a directed shortest-path algorithm, which tends to find the shortest path with very low overhead.

Our experience with the penalty function is limited, and the specific numbers in use right now are fairly arbitrary. We need more experience to understand the relative importance of the various penalties, and to extend the rule set with more knowledge about good and bad crossings. Nonetheless, even with this initial rule set we have found crossing placement during global routing to be essential for effective obstacle-avoiding routing.

10. Acknowledgements

Robert N. Mayo, Carlo H. Séquin and David Wallace reviewed drafts of this paper and provided numerous comments. Franky Leung designed the Tester chip which we used to debug the router and measure its performance. The work described here was supported in part by SRC under grant number SRC-82-11-008.

11. References

[Chen77] Chen, K. A., et. al., "The Chip Layout Problem: An Automatic Wiring Procedure", *Proceedings 14th Design Automation Conference*, New Orleans (1977), pp. 298-302.

[Chen83] 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, UC Berkeley, (Feb. 1983).

[Clow84] Clow, Gary W., "A Global Routing Algorithm for General Cells", *Proceedings 21st Design Automation Conference*, Albuquerque (1984), pp. 45-51.

[Hamachi84] Hamachi, G. T., and J. K. Ousterhout, "A Switchbox Router with Obstacle Avoidance", *Proceedings 21st Design Automation Conference*, Albuquerque (1984), pp. 173-179.

[Hightower69] Hightower, D., "A Solution to the Line Routing Problem on the Continuous Plane", *Proceedings Design Automation Workshop* (1969), pp. 1-24.

[Hightower80] Hightower, D., "A Generalized Channel Router", *Proceedings 17th Design Automation Conference*, Minneapolis, (1980) pp. 12-21.

[Horowitz] Horowitz, E., and S. Sahni, *Fundamentals of Computer Algorithms*, Computer Science Press, Inc. (1978), pp. 183-188.

[Lee61] Lee, C. Y., "An Algorithm for Path Connections and its Application", *IRE Transactions on Electronic Computers* (Sept. 1961), pp. 346-365.

[Ng84] Ng, C. H., "A Symbolic-Interconnect Router for Custom IC Design", *Proceedings 21st Design Automation Conference*, Albuquerque (1984), pp. 52-58.

[Ousterhout84] Ousterhout, J. K., Hamachi, G. T., Mayo, R. N., Scott, W. S., and Taylor, G. S., "Magic: A VLSI Layout System", *Proceedings 21st Design Automation Conference*, Albuquerque (1984), pp. 152-159.

[RiF82] Rivest, R. L., and C. M. Fiduccia, "A Greedy Channel Router", *Proceedings 19th Design Automation Conference*, Las Vegas (1982).

[Rivest82] Rivest, R. L., "The 'PI' (Placement and Interconnect) System", *Proceedings 19th Design Automation Conference*, Las Vegas (1982).

[Sou81] Soukup, J., "Circuit Layout", *Proceedings of the IEEE*, Vol. 69, No. 10 (Oct. 1981), 1281-1304.

[Suen81] Suen, L, "A Statistical Model for Net Length Estimation", *Proceedings 18th Design Automation Conference*, Nashville (1981), pp. 769-774.