Beaver: A Computational Geometry-Based Tool for Switchbox Routing†

James P. Cohoon and Patrick L. Heck
Department of Computer Science
University of Virginia

Computer Science Report No. TR–87-01
Revised December 15, 1987

† This report is to appear in the *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.*

BEAVER: A Computational Geometry-Based Tool for Switchbox Routing†

J. P. Cohoon and P. L. Heck

Department of Computer Science
University of Virginia
Charlottesville, VA 22903

ABSTRACT

A new fast switchbox router, BEAVER is presented. BEAVER combines a delayed layering scheme with computational geometry techniques to heuristically produce a switchbox solution that minimizes both via usage and wire length, and maximizes the use of a preferred routing layer. Other important features are its use of priority queues to determine the order that nets are routed and its prioritized control of individual track and column usage to prevent routing conflicts. BEAVER consists of four tools that are run successively—a corner router, a line sweep router, a thread router, and a layerer. The corner router makes single-bend terminal-to-terminal connections. The line sweep router makes straight-line connections, single-bend connections, and two-bend connections. The thread router makes connections of arbitrary form. The layerer completes the switchbox by layering wires that have been assigned a location but not yet a layer. BEAVER has successfully routed all of the classic switchboxes. Its solution quality with respect to wire length was better or comparable to the best previous solutions and its via usage was consistently the minimum. These characteristics make it appropriate for its intended use as the initial router invoked to solve a switchbox.

KEYWORDS

Switchbox Routing, VLSI Circuit Design, Routing, Channel Routing, Computational Geometry

† This work was supported in part by the National Science Foundation under grant DMC 85-05354 and by the Virginia Center for Innovative Technology under grant TDC 86-002-01.

Leave it to Beaver

— Ward Cleaver

1. INTRODUCTION

The two-stage *global-detailed* routing method is an important technique for realizing interconnections on VLSI circuits [HU85, SOUK81]. The method was introduced by Hashimoto and Stevens for determining interconnections for the ILLIAC IV [HASH71]. In its global stage, this method first divides the routing region into a collection of disjoint rectilinear areas. It then determines a sequence of subregions to be used in interconnecting each net. By carefully monitoring and influencing the subregion usage it is possible to evenly distribute connections and thus reduce the likelihood of wire blocking. There are normally two kinds of rectilinear subregions created in the global stage—*channels* and *switchboxes*. Channels are two-layered rectangular regions that limit their interconnection endpoints or *terminals* to one pair of parallel sides. Switchboxes are generalizations of channels that allow terminals on all four sides. In the detailed stage, channel routers and switchbox routers are used to complete the connections. The contribution of this report is a new heuristic method for switchbox routing.

Although some good switchbox routers (e.g., [ENBO87, HAMA84, HO85, HSIE85, KAPL87, LUK85, MARE85, SHIN86]) appeared after Soukup's call for increased research on switchbox routing in his 1981 survey of circuit layout [SOUK81], none of the routers could claim 100% routability. This limitation has caused some state-of-the-art VLSI chip manufacturers to delay using design methods that introduce switchboxes. However, Joobbani and Siewiorek have recently developed a successful knowledge-based expert-system router, WEAVER, that uses a 100% routability expert [JOOB86]. They demonstrated its success by automatically routing the three classic pathological instances: Burstein and Pelavin's Difficult Switchbox [BURS83], Luk's Dense Switchbox [LUK85], and Luk's Terminal Intensive Switchbox [LUK85]. WEAVER was the first router to automatically solve these problems, but there was a price extracted for WEAVER's consistent success—some of its solutions required over a half hour of CPU time to be constructed.

A tool like WEAVER with its consistently deep searching of the solution space is appropriate for the most difficult of switchboxes, but its computing requirements suggest that it is not necessarily the first tool that should be applied to a switchbox. For this purpose, we propose here a new fast switchbox router called BEAVER. Since BEAVER is heuristic-based, it will not route all switchboxes. However, we believe it can route most switchboxes. In particular, it too can automatically route all of the classic pathological instances. Although BEAVER's running time is on the order of several seconds, and WEAVER's running time is on the order of tens of minutes, their wire usage is comparable for the classic pathological instances. With regard to the heuristic routers, BEAVER's running time and wire usage are better or comparable. Finally, we note that BEAVER's via usage is consistently the minimum among all switchbox routers.

To ensure quickness, BEAVER never reroutes a connection once it has been laid. To ensure quality, BEAVER considers several alternatives before assigning any wire to a track or column. In evaluating an alternative, BEAVER considers several criteria such as wire length, number of vias, and the interconnection's impact on other unrealized nets. Although BEAVER is biased toward assigning vertical and horizontal interconnections to different layers, it delays layer assignment as long as possible and will ignore its bias both to ensure the routability of a net and to avoid via introduction. BEAVER's other salient features are its use of priority queues to determine the order that nets are interconnected, its prioritized control of individual track and column usage, and its generalized line sweeping heuristics that are likely to find interconnections with minimal length and with minimal impact on other nets' routability.

In the remainder of this report, we present the BEAVER approach to switchbox routing and statistically compare its performance on several switchbox instances—including the three classic pathological instances—to that of some existing routers.

2. PROBLEM STATEMENT

As stated informally above, a switchbox is a rectangular region with terminals on all four sides. The terminals are grouped into a collection S of disjoint sets or *nets*. The goal of the switchbox router is to make the terminals in each individual net electrically common. To identify which terminals are to be interconnected, a terminal is labeled with a *net identification number*, k, where $1 \le k \le |S|$. Using Luk's terminology [LUK85], a switchbox is formally a region $R = \{0, 1, \ldots, m\} \times \{0, 1, \ldots, n\}$, where m and n are positive integers. Each pair (i,j) in R is a *grid point*. The i-th *column* is the set of grid points $COL(i) = \{(i,j) \mid j \in \{0, 1, \ldots, n\}\}$, $1 \le i \le m$. The j-th *row* or *track* is the set of grid points $ROW(j) = \{(i,j) \mid i \in \{0, 1, \ldots, m\}\}$, $1 \le j \le n$. The zero-th and m-th columns are respectively the *LEFT* and *RIGHT* boundaries of the switchbox. Similarly, the zero-th and n-th rows are respectively the *TOP* and *BOTTOM* boundaries of the switchbox. The connectivity and location of each terminal is represented as $LEFT(i)=k$, $RIGHT(i)=k$, $TOP(i)=k$, or $BOTTOM(i)=k$ depending upon which side of the switchbox it lies on, where i stands for the coordinate of the terminal along the edge and k stands for its net identification number. A small sample switchbox and its solution is given in Figure 1. For this example

$$R = \{0, 1, 2, 3, 4, 5, 6, 7, 8\} \times \{0, 1, 2, 3, 4, 5\},$$
$$LEFT(1, 2, 3, 4) = [5, 2, 5, 1],$$
$$RIGHT(1, 2, 3, 4) = [2, 6, 3, 4],$$
$$TOP(1, 2, 3, 4, 5, 6, 7) = [0, 1, 5, 6, 4, 3, 0],$$
$$BOTTOM(1, 2, 3, 4, 5, 6, 7) = [1, 2, 5, 2, 2, 4, 3].$$

The solution demonstrates several traditional routing conventions [BURS83, SOUK81]. Each connection runs horizontally or vertically along the tracks and columns. A connection can exit a terminal on either layer. A connection can change layers only at a grid point. A *via* or feed-through is the mechanism by which a connection changes layers. Although only a single connection can occupy a given layer of a track or column

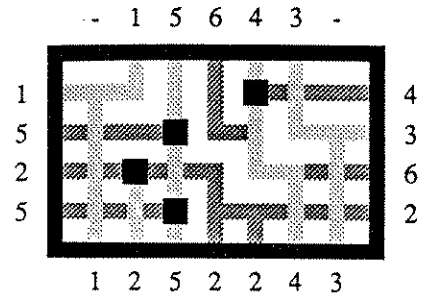


Figure 1. — Switchbox instance with solution.

section, different connections can occupy different layers. For example, net 6's connection lies in part under net 4's connection when entering and exiting grid point (5,2).

3. BEAVER

As seen in the high-level description of Figure 2, BEAVER uses up to three methods to find interconnections for nets—a *corner router*, a *line sweep router*, and a *thread router*. The corner router tries to connect terminals that form a *corner connection*. Such a connection is formed by two terminals if

(1) They belong to same net;

(2) They lie on adjacent sides of the switchbox;

(3) There are no terminals belonging to the net that lie between them on the adjacent sides.

A corner connection is always realized using a wire with a single bend. The line sweep router is an adaptation of the computational geometry technique of plane sweeping [BENT79, PREP85]. Here a minimal length connection that joins two of the current net's disjoint subnets is heuristically sought, where a *subnet* is a set of one or more terminals and wires that are electrically common and where two subnets of a net are *disjoint* if they are not electrically common. Although the BEAVER overview in Figure 2 suggests that the corner routing and line sweep routing phases are independent, the line sweep router does make use of the corner router to complete any corner connections that become realizable during line sweep routing. The thread router is a maze-type router [LEE61, MOOR59] that does not restrict its search for a connection to any preferential form. All three routers, upon their invocation, are given a priority queue of nets to route. During their operation, it may be the case that additional nets are added to the queues. All additions are made at the end of the appropriate queue, since these additions are normally nets being given another chance to route.

BEAVER uses a layering scheme that delays as long as possible the assignment of wires to layers. Whenever a new connection is made by the routers, only those parts of it that cross an existing connection are assigned to a particular layer (the crossed connection is also then assigned to a layer). This delay allows

BEAVER flexibility in routing subsequent wires. Since its three routers typically leave many unlayered wires, a layerer is called to assign unlayered wires. This assignment is done in a manner that minimizes vias and maximizes the use of a preferred layer.

With one partial exception for doglegs, all connections considered by the corner and line sweep routers lie on either those contiguous portions of tracks or columns that are reserved for the current net or on portions that are available to any net. During initialization, a net is given control of the tracks and columns that extend out from its terminals. The extent of this control is determined by a parameter to BEAVER that indicates the percentage of the tracks and columns to be reserved (the default is 50%). If the control parameter is less than 50%, the middle portion of a track or column is available to any net. If instead the control parameter is greater than 50%, there is dual control of the middle section by the nets with terminals on opposite sides of the track or column. A dual-controlled section is used on a first-come-first-served basis by the nets that share it. When a net i is realized, its control of tracks and columns is passed to nets that have *cross terminals* with respect to net i's terminals, where two terminals are cross terminals if they lie on opposite ends of a track or column. If these nets have also been realized, then the sections may be used by the uncompleted nets on a first-come-first-served basis. As stated above, doglegs are given a partial exception to control requirements. The unit-length middle segment or *cross-piece* of the dogleg can use a track or column without its associated net having control. However, the cross-piece is constrained to lie on the layer not normally used by a segment with its orientation.

```
Algorithm
    Initialize control information
    Initialize corner-pq
    Corner route
    if there are unrealized nets then
        Initialize linesweep-pq
        Line sweep route
        if there are unrealized nets then
            Relax control constraints
            Reinitialize linesweep-pq
            Line sweep route
            if there are unrealized nets then
                Initialize thread-pq
                Thread route
            end if
        end if
    end if
    Perform layer assignment
end
```

Figure 2. — BEAVER algorithmic overview.

```
Algorithm
   while corner-pq is not empty do
      Dequeue next net i from corner-pq
      Make net i's realizable corner connections
      if net i is completed then
         Release net i's track and column control
         for all nets j with cross terminals of net i do
            if net i has increased net j's row or column control and net j has an unrealized corner
            then
               Enqueue net j at the end of corner-pq if it is not already in the queue
            else if line sweep routing has begun then
               Enqueue net j at the end of linesweep-pq if it is not already in the queue
            end if
         end for
      end if
   end while
end
```

Figure 3. — Corner router algorithmic overview.

3.1. Corner Routing

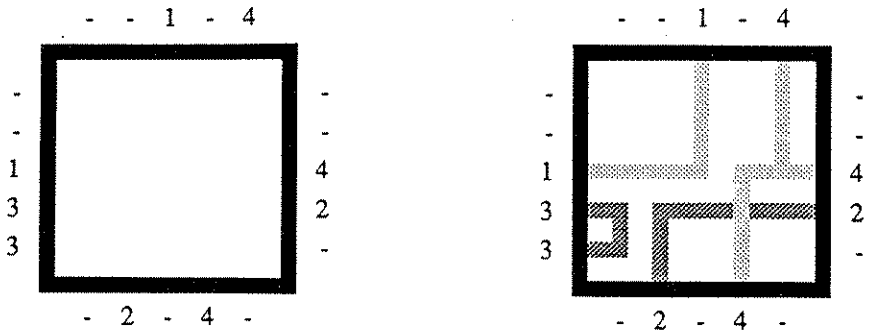
An overview of the corner router's operation is given in Figure 3. Routing with the corner router is preferred by BEAVER for two reasons:

(1) Its connections tend heuristically to be part of the minimal rectilinear Steiner trees for the nets;

(2) It is the fastest of BEAVER's three routers.

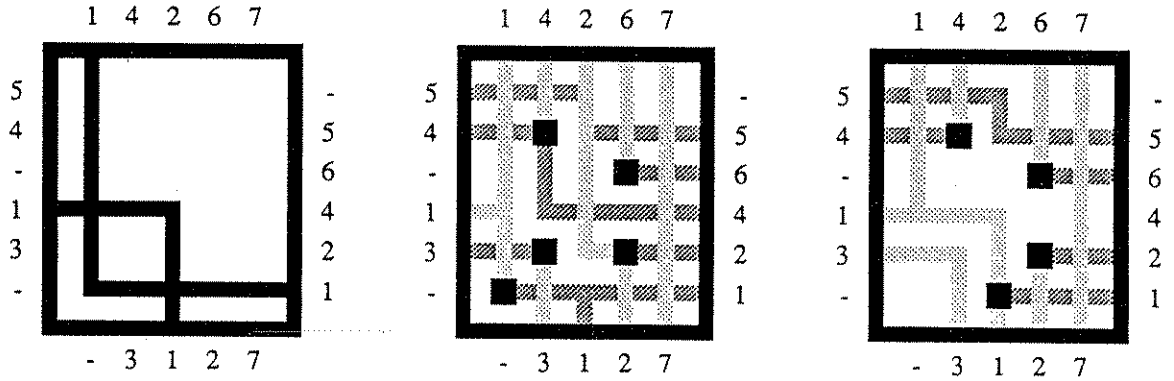
Therefore, as long as it is possible to make corner connections, the corner router's execution continues. In addition, if corner connections become realizable during line sweep routing then the line sweep router's execution is temporarily suspended until the corner connections are routed.

The corner router is fastest because its connections can be made by simply examining the control of the terminals that comprise the corner. If sections of control overlap, then the corner can be realized. Figure 4.a shows a small switchbox. With default control in effect, net 1 is one-corner net (*LEFT*(3)-*TOP*(3)) that is immediately eligible for corner routing. Similarly, net 4 is a two-corner net (*TOP*(5)-*RIGHT*(3), *BOTTOM*(4)-*RIGHT*(3)) that is immediately eligible for corner routing. These nets are both immediately eligible for routing, since their terminals either control or share control of the sections of rows and columns that are necessary for their corner connections. Although net 2 is also a one-corner net (*BOTTOM*(2)-*RIGHT*(2)), it cannot be immediately corner routed, since net 3 has control of grid point (2,2). Once net 3 is routed, net 2's control of *ROW*(2) is expanded to include grid point (2,2) and then net 2's corner connection would be permissible. Figure 4.b shows the same switchbox after it has been routed by BEAVER. In this figure, nets 1, 2, and 4 are realized as corner connections.



(a) Switchbox (b) Routing of switchbox

Figure 4. — A switchbox with corner connections (nets 1, 2, and 4).



(a) Overlapping corners (b) Successful solution (c) Incomplete solution

Figure 5. — Processing a net with an overlap cycle.

During the initialization of *corner-pq*, each net is analyzed to determine if it has corners to be routed. The two terminals that comprise each corner are identified and stored with the net information. If a net is identified as having either one or two corners (terminals on either two or three sides of the switchbox), no additional analysis need take place. However, additional analysis known as *corner ordering* is performed for nets with four corners (a net can never have only three corners). This ordering is performed on four-corner nets since a cycle might otherwise be introduced when routing the corners. In particular, there are two forms of cycles that could be introduced into the routing—*overlap cycles* or *four-terminal cycles*. An overlap cycle occurs when a pair of opposite corners overlap as demonstrated in Figure 5.a. In this figure, net 1's lower left

corner (*BOTTOM*(3)-*LEFT*(3)) overlaps with its upper right corner (*TOP*(1)-*RIGHT*(1)). A four-terminal cycle occurs when a four-terminal net has its terminals positioned so that neither of the two pairs of opposite corners overlap and none of the four terminals share a row or column. This is demonstrated in Figure 6.a.

To prevent an overlap cycle from occurring, BEAVER ensures that only one overlapping corner is corner routed. This is done by marking the two overlapping corners as mutually exclusive. If the net is comprised of only four terminals, then routing the other two corners and one of the mutually exclusive corners will complete the net. Otherwise, the line sweep router or the thread router will be needed to complete the net. As to which of the two mutually exclusive corners will be routed, the corner router prefers the one with least impact on the routability of other nets. To select the preferred one, BEAVER examines the cross terminals of the corners in question to determine whether the routing of the corners will block the cross-terminals' nets from routing. The corner connection with less potential blocks of other nets is preferred. If the preferred corner cannot be realized, the other corner is then used. In Figure 5.a, net 1's lower left corner connection blocks nets 2 and 4, and its upper right corner connection does not block any nets, so the upper right corner connection is preferred. Figure 5.b shows a complete successful routing of the example by BEAVER using the preferred connection. Figure 5.c shows BEAVER's partial routing of the switchbox using the non-preferred connection. In this figure, nets 2 and 4 have not yet been completed. The routing of either one of the two will prevent the other from being routed.

To prevent a four-terminal cycle from occurring, BEAVER merely needs to route three of the corners as this will complete the net and not introduce any cycles. For this case, the analysis determines the order that the corners should be routed to heuristically maximize the routability of subsequent nets. The ordering criteria is similar to the criteria for finding the least preferred corner in the cycle overlap case—the corner *c* with the most

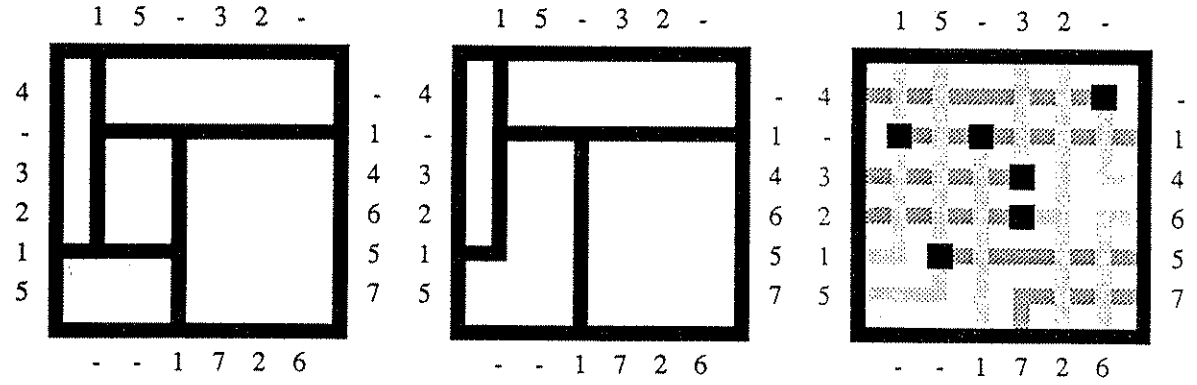


Figure 6. — Processing a net with a four-terminal cycle.

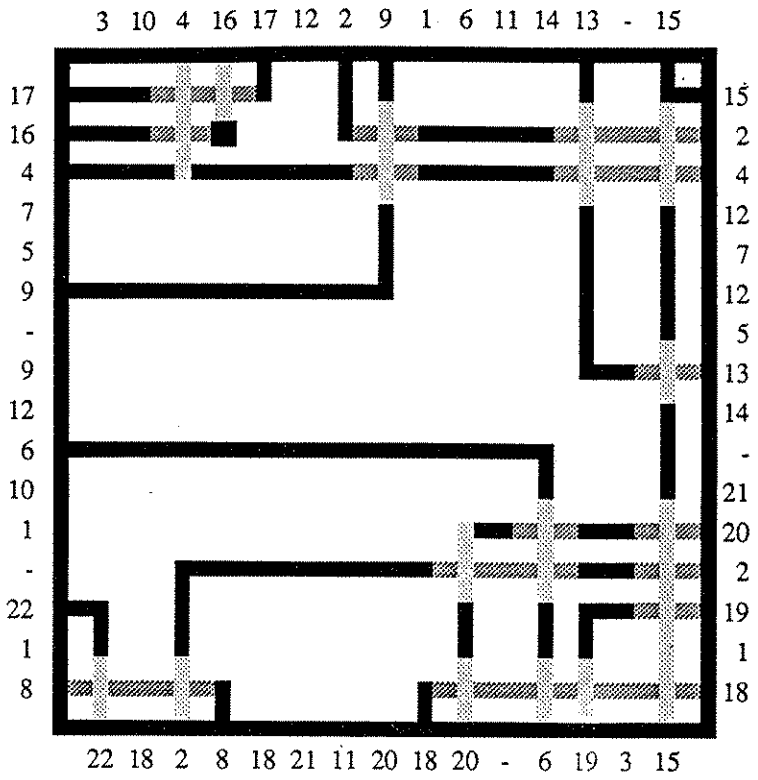


Figure 7. — Corner routing of Pedagogical Switchbox.

potential blocks is preferred least. If any of the other three corners are not completed when corner routing is attempted on them, then corner *c* is tried. In Figure 6.a, net 1's lower left corner (*BOTTOM*(3)-*LEFT*(2)) can potentially block net 5's upper right corner (*TOP*(2)-*RIGHT*(2)) so it is placed last in the ordering (since none of the other three corners have any potential blocks, their order is insignificant). The resulting preferred routing for net 1 is shown in Figure 6.b. A successful solution for the switchbox using the preferred net 1 routing is shown in Figure 6.c. For this instance, if the non-preferred corner is routed, a solution is still possible. However, the solution quality is slightly lower—both an additional via and additional wire are needed.

An example switchbox, the Pedagogical Switchbox (16 rows, 15 columns, 22 nets, 57 terminals) is given in Figure 7 along with BEAVER's working solution after the corner router completed. The corner router found 16 net and subnet connections using 155 units of wire. A little more than 50% of its wire segments still required layering (the black wire segments are ones that have not yet been layered).

```
Algorithm
   repeat
      while linesweep-pq is not empty do
         Dequeue next net i from linesweep-pq
         repeat
            Find and realize the smallest possible interconnection for net i using the line sweep
            heuristics.
         until net i is realized or no connection is found
         if net i is realized then
            Release net i's track and column control
            for all nets j with cross terminals of net i do
               if net i has increased net j's row or column control and net j has an unrealized corner
               then
                  Enqueue net j at the end of corner-pq if it is not already in the queue
               else
                  Enqueue net j at the end of linesweep-pq if it is not already in the queue
               end if
               Corner route
            end for
         end if
      end while
      if connections were made in emptying linesweep-pq then
         Reinitialize linesweep-pq with unrealized nets
      end if
   until linesweep-pq is empty
end
```

Figure 8. — Line sweep router algorithmic overview.

3.2. Line Sweep Routing

An overview of the line sweep router's operation is given in Figure 8. Since the line sweep router is the second fastest of the three routers, it is preferred when there are no corner connections to be made. As noted in the previous section, if some corner connections become realizable after the current net's line sweep realization, then the line sweep router's execution is temporarily suspended until *corner-pq* is emptied.

The line sweep priority queue is initialized with the unrealized nets. These nets are divided into two groups—nets that had corner connections and nets that did not. The nets that did not have corner connections are given higher priority since they have not yet been given a chance to route. Within either group, a net's priority is inversely proportional to its number of disjoint subnets. When examining the current net *i*'s routing possibilities, the line sweep router considers only a wire with a single bend, a single straight-line wire, a dogleg connections with a unit-length cross-piece, and three wires arranged as either a horseshoe or a stair step. A dogleg connection is differentiated from a stair step connection with a unit-length cross-piece, since the dogleg's unit-length cross-piece is allowed to violate track and column control. Examples of the five prototype

connections are shown respectively from left-to-right in Figure 9. BEAVER also checks for connections that are rotations and reflections of the ones given in the figure. In practice, BEAVER also considers connections that join subnet-to-subnet and terminal-to-subnet, rather than just the terminal-to-terminal connections given in the figure. In addition, BEAVER only introduces a via or assigns a wire segment to a layer if it is immediately necessary.

In looking for its connections, the router uses a generalization of the computational geometry technique of plane sweeping [BENT79, PREP85]. This technique customarily involves scanning a geometric space with a single line and observing its intersections in the space. We use it in its natural manner to find straight-line connections that join two disjoint subnets of the net in question. Here, the switchbox is swept both on a row-by-row and a column-by-column basis. To find connections with doglegs or with one and two bends, BEAVER uses a more general sweeping procedure that has up to three scan lines sweeping in tandem. To decrease the computational complexity of the tandem sweeping, *bounding functions* [HORO78] were incorporated into BEAVER that restrict the search of the switchbox. For example, since BEAVER prefers minimal length connections, when a tandem sweep has a cumulative wire length longer than the best solution seen so far, the sweeping is abridged in one direction. As another example, if a net consists of two terminals on the same side of a switchbox, the line sweep router only considers a horseshoe connection.

If there is more than one minimal length connection, BEAVER performs several checks on the possible connections to determine which connection is the preferred one. The first check determines if one of the possible connections is more likely to use less vias than the others. Thus, straight-line connections are preferred over dogleg connections, dogleg connections are preferred over single-bend connections, and single-bend connections are preferred over two-bend connections. If more than one connection meets this criterion, a second check is performed to determine which connection is likely to have the least impact on the remaining

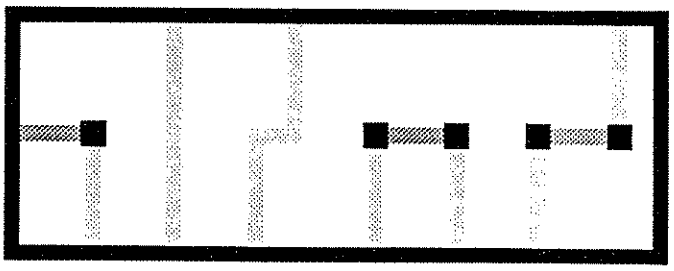


Figure 9. — Prototype line sweep connections.

unfinished nets. Connections that use tracks or columns that contain the terminals of unfinished nets are penalized. The penalty is inversely proportional to the distance of the connection to the terminal of the unfinished net. The connection with least penalty is preferred. If more than one connection meets this criterion, then the connection that was determined first is selected.

Figure 10 shows BEAVER's working solution to our Pedagogical Switchbox instance after the initial line sweep routing has completed. The figure provides working examples of each type of line sweep connection. As an example of a straight-line connection, there is net 1's terminal-to-terminal connection from *LEFT*(2) to *RIGHT*(2). Net 1's terminal-to-subnet connection from *LEFT*(5) to grid point (4,2) is a single-bend connection. Net 7's terminal-to-terminal connection from *LEFT*(13) to *RIGHT*(12) is a dogleg connection. A horseshoe connection is illustrated with net 18's terminal-to-subnet connection from *BOTTOM*(2) to grid point (5,1). Finally, a staircase connection is illustrated with net 11's terminal-to-terminal connection from *BOTTOM*(7) to *TOP*(11).

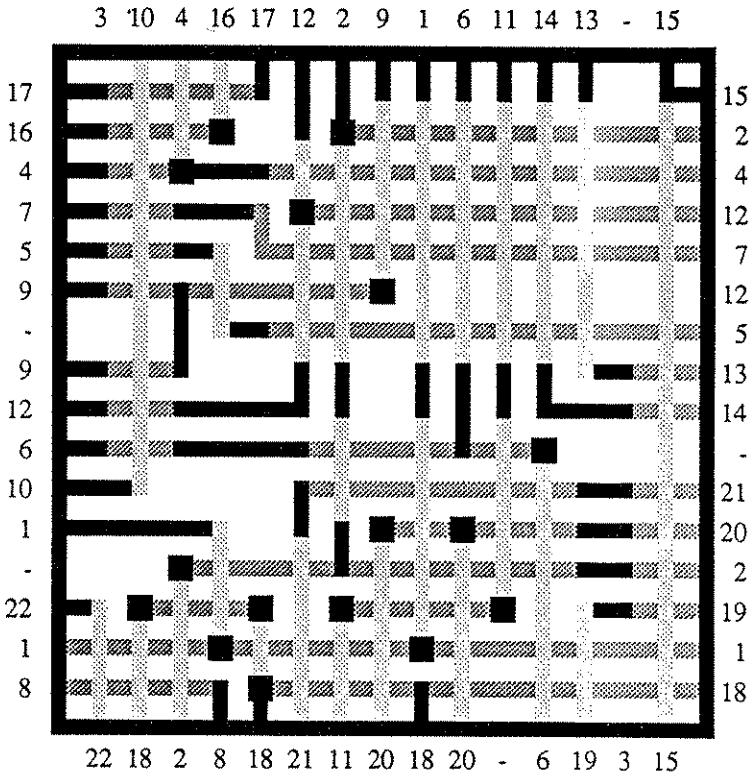


Figure 10. — Line sweep routing of Pedagogical Switchbox with default control limits of 50% in effect.

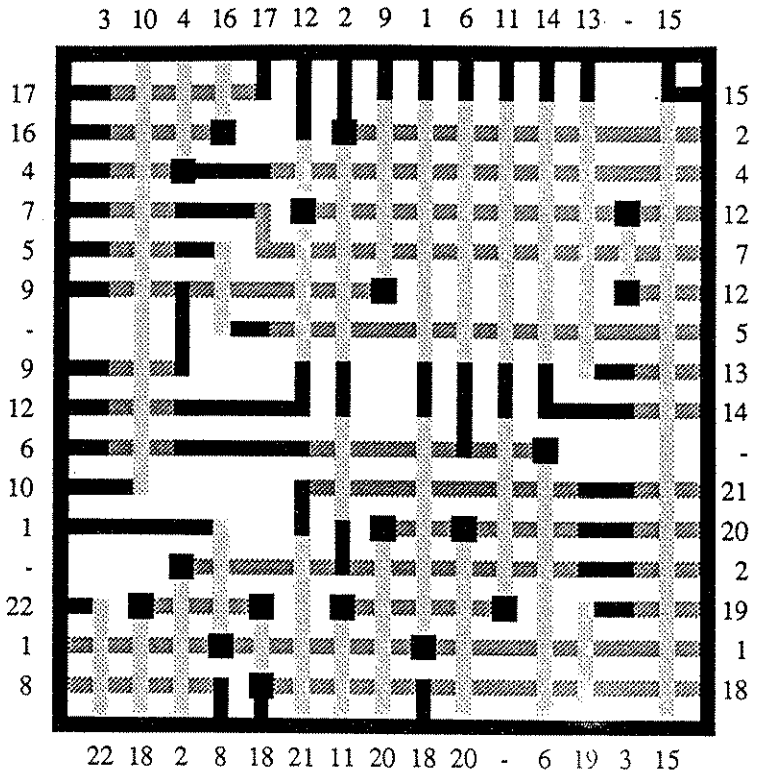


Figure 11. — Line sweep routing of Pedagogical Switchbox with relaxed control in effect.

The line sweep router's Pedagogical Switchbox invocation determined 12 new net and subnet connections using 207 units of wire with 17% of the wire yet to be layered. It assigned fifteen vias (in contrast, the corner router assigned only one via). Twenty of the twenty-two nets in the switchbox have already been completed—net 12 has a loose terminal that must be connected to a three-terminal subnet and net 3 is a two-terminal net with no connections as of yet.

To delay using the thread router, line sweep connections are sought until it is no longer possible to make such connections given the current track and column control. A net's control of a track or column is then reduced to the point where it is first possible for the net to bend away from the track or column. In Figure 10, net 3's terminal *BOTTOM*(14) initially had exclusive control of *COL*(14). After relaxation, the terminal's control is now reduced to grid point (14,7). The line sweep router is then reinvoked with its priority queue containing the unrealized nets, where the priority of an unrealized net is inversely proportional to its number of disjoint subnets. The router again continues until no line sweep connections are possible. Figure 11 shows the working solution to the Pedagogical Switchbox after control was relaxed and the line sweep router was re-invoked. The invocation completed net 12 using a four-unit, two-via, single-bend connection that ran from

terminal *RIGHT*(11) to its subnet at grid point (14,13). This connection was possible as net 3's control of *COL*(14) ended at grid point (14,7).

3.3. Thread Routing

Finally, the thread router is invoked. This router is a maze-type router [LEE61, MOOR59] that seeks minimal-length connections to realize the remaining nets. Among all our test instances, it was necessary to invoke the thread router just four times. For each of these invocations, it was needed only to find a single connection. Since the thread router does not restrict its connections to some preferential forms, it will find a connection for a net if one exists. Thus, the thread router never needs to reconsider or re-enqueue a net.

A high-level description of the thread router is given in Figure 12. As shown there, the thread router processes its priority queue one-net-at-a-time. When processing the current net i, the goal is to use maze expansion to find a minimal length connection that joins i's currently smallest disjoint subnet s to some other disjoint subnet of net i, where the size of a subnet is measured in terms of wire length. The process is repeated until net i is connected or until it is determined that none of i's remaining disjoint subnets can be connected. By selecting s for maze expansion over net i's other subnets there is a two-fold benefit:

(1) The maze expansion is heuristically minimized since it reduces the number of starting points;

(2) The number of possible destination points for the maze expansion is maximized.

Like Soukup's maze router [SOUK78], the thread router determines heuristically the next grid point that should be expanded in its search for a minimal length connection. The grid point that continues the search in its same direction is preferred. This direction is selected because it does not introduce a bend in the wire. If the thread router finds it necessary to backtrack past this grid point, the next direction attempted is one that expands the search towards the "center" of the region where the largest disjoint subnet t of net i can be found. If subnet t is a single terminal, then the center of the subnet is the terminal itself. If instead t is composed of terminals

```
Algorithm
   while thread-pq is not empty do
      Dequeue next net i from thread-pq
      while net i consists of more than one routable disjoint subnet do
         Determine net i's smallest realized subnet s that has not undergone thread routing
         Determine net i's largest realized subnet t
         Perform a maze expansion from subnet s towards subnet t
      end while
   end while
end
```

Figure 12. — Thread router algorithmic overview.

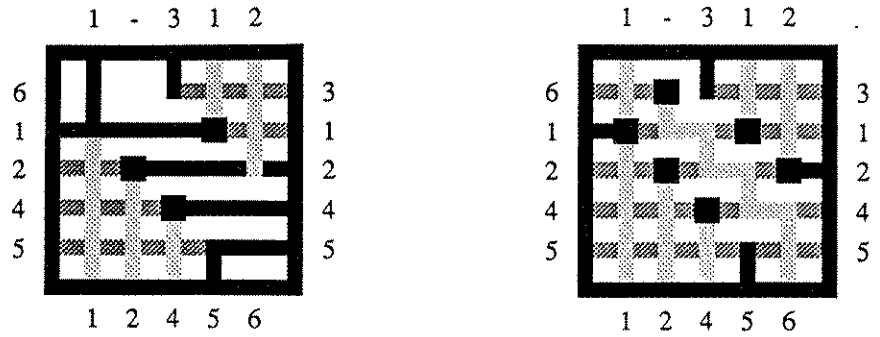
and wires, then t's center is the center of the smallest bounding rectangle that encompasses t's terminals and at least one corner of the switchbox. If the thread router finds it necessary to backtrack past this grid point, then the remaining untried direction is used. Although the thread router biases its search towards subnet t, the router does not demand that subnets s and t be connected in the current maze expansion. Rather, the goal remains to connect s to one of net i's other disjoint subnets.

Since many segments are still typically unlayered when the thread router is invoked, care must be taken in searching for a connection. If the current grid point p being examined is *empty* (i.e. no net is routed through it), the thread router marks the cell as tentatively occupied, but unlayered, by the current net i. If instead point p is *occupied* by another net that has already been layered assigned, the thread router marks the other layer as tentatively occupied by the current net i. If instead point p is occupied by another net that has not yet been layered assigned, then the cell is given special attention. Since it may be the case that a connection for net i can be found by assigning the connection to one of the two layers but not the other, the examination considers both possible layer assignments in turn. The assignment of a connection to a particular layer at a grid point requires that the four adjacent grid points be analyzed to determine if their layering is now necessary. Since this propagation effect does not extend further, it is not necessary to analyze grid points more than one unit from point p.

Choosing one layer for a wire segment over the other layer may lead to a shorter connection. Therefore, both layer assignments are always considered. To reduce the amount of backtracking, bounding functions are again used. For example, the maze expansion is never carried to grid points further from the initial points of expansion than the length of the current best connection. Also, by specifying the exact destinations of the search (there may be more than one grid point of a subnet that can be reached), the searching can be reduced further. As each cell is marked, the current distance traveled to get to the grid point is added to the minimum distance to the closest destination point (assuming there are no obstacles). If this distance is longer than the length of the current best connection, this portion of the search can be abridged.

Solution splicing is also used to reduce the search. This method allows a better solution to be located before the destination of the connection is reached. Rather than always searching for a destination subnet, the thread router recognizes when it is crossing the path of the current best connection. If the current distance traveled by the search to a grid point is shorter than the distance the current best connection traveled to the grid point, it may be possible to replace the head of the best connection so far with the current search path. To ensure the connection can be realized, the path of the tail of the connection is analyzed. This is done to ensure that the two parts can be properly routed since it is possible that the new head of the connection may make layering decisions that could prevent the tail of the connection from being realized.

The result is a maze-type algorithm that can backtrack to ensure that it finds the shortest connection. For almost all instances, the thread router quickly found its connections. For example, its average running time was 0.06 seconds. If the thread router is searching a relatively large area with many unlayered segments, it could



(a) Before thread routing (b) After thread routing

Figure 13. — Thread Routing Net 6.

take the router several minutes to find a connection or to report that it is impossible to find a connection. However, relatively large sparse areas typically increase the routability of a switchbox, thus increasing the probability that the thread router is not needed to complete the routing.

Figure 13 shows how effective the thread router is in realizing difficult connections. The two terminals, *LEFT* (5) and *BOTTOM* (5), that comprise net 6 in Figure 13.a remain unconnected after the corner routing and line sweep routing has completed. Figure 13.b shows the snaking thread router connection (ten units long, seven bends, one via) that joined the two terminals. In realizing this connection, the thread router also layered thirteen other units of wire and assigned 3 other vias.

Figure 14 shows BEAVER's solution to our Pedagogical Switchbox after the thread router has completed. The thread router was required only to route net 3. This 30-unit 2-via connection starts at *TOP*(1) and ends at *BOTTOM*(14). The connection twice violated normal routing conventions. The connection required that 21 other units of wire be layered and 5 other vias be introduced.

3.4. Layer Assignment

BEAVER's final task is layer assignment. It is done in a manner that minimizes the introduction of additional vias. If only one of the two layers is metal, then BEAVER also uses a metal maximization heuristic.

Let a grid point be considered *red* if

(1) It abuts layered and unlayered connections;

(2) All layered connections it abuts are assigned to the same layer.

Let a grid point be considered *black* if

(1) It abuts layered and unlayered wire connections;

(2) It abuts connections assigned to different layers.

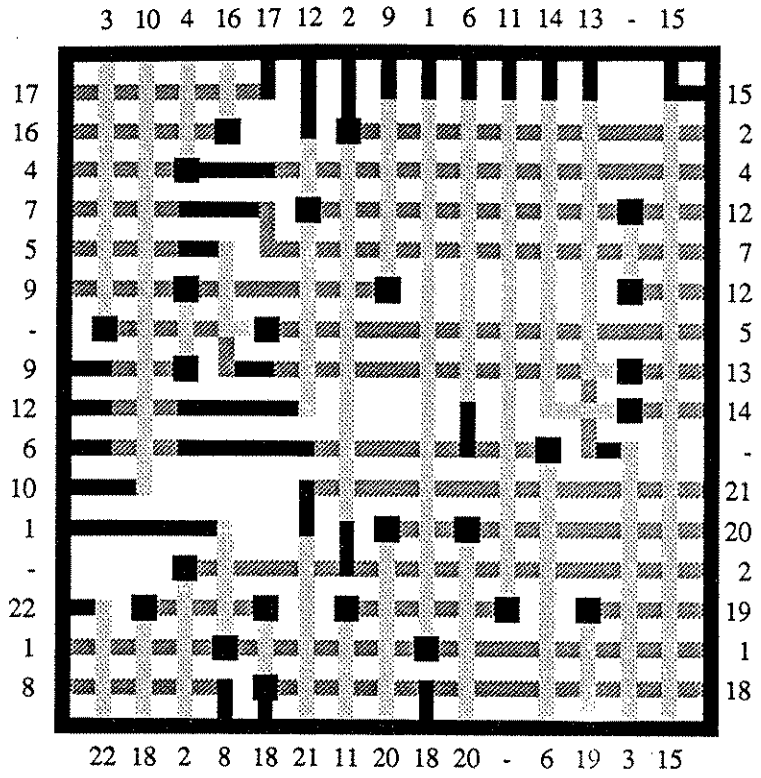


Figure 14. — Thread routing of Pedagogical Switchbox.

It will always be the case, given BEAVER's method of operation by its corner, line sweep, and thread routers, that a black grid point has already been assigned a via. To assign the unlayered connections, BEAVER performs a two-step process. The first step processes those unlayered connections with at least one red endpoint; the second step processes the remaining unlayered connections. For these remaining connections, all endpoints are either black or terminals.

Processing an unlayered connection with only one red endpoint simply requires propagating the layer assignment already in effect at the red endpoint. Processing such connections does not introduce any vias in the switchbox, since when the layer propagation is terminated, the terminating grid point is either a terminal or a via. Processing unlayered connections with multiple red endpoints can be handled similarly if all the red endpoints abut connections on the same layer. Processing an unlayered connection with multiple red endpoints that abut connections on different layers allows BEAVER to assign the connection to either layer. For this case, BEAVER will select the metal layer if only one of the two layers is metal, otherwise the top metal layer is selected. Processing these connections will introduce vias at the red endpoints that previously abutted connections only on the other layer.

Processing connections with endpoints that are either black or terminals is also straight-forward. Either layer can be used for such connections and no additional vias are necessary. BEAVER normally selects the layer in the same manner as it did for connections with red endpoints that abutted connections on different layers. However, for a short wire from a terminal to a black endpoint, preference is given to the layer that is normally assigned a wire with that orientation.

Figure 15 shows BEAVER's complete solution to our sample switchbox after all connections have been layered. For example, since net 12's unlayered connection ran from red grid point (3,8) to red grid point (6,8) in Figure 14, and since the red endpoints abut connections on different layers, the unassigned connection was assigned to the top layer in Figure 15. As a second example, since net 12's connection ran from terminal *LEFT*(8) to red endpoint (1,8) in Figure 14, the layer of the red endpoint's abutting connection was used in Figure 15. As a last example, since net 18's unlayered unit-length connection runs from terminal *BOTTOM*(5) to black grid point (5,1), and since vertical connections normally lie on the top layer, the connection was assigned to the top layer in Figure 15. In completing the Pedagogical Switchbox, BEAVER layered 41 units of wire and added six more vias.

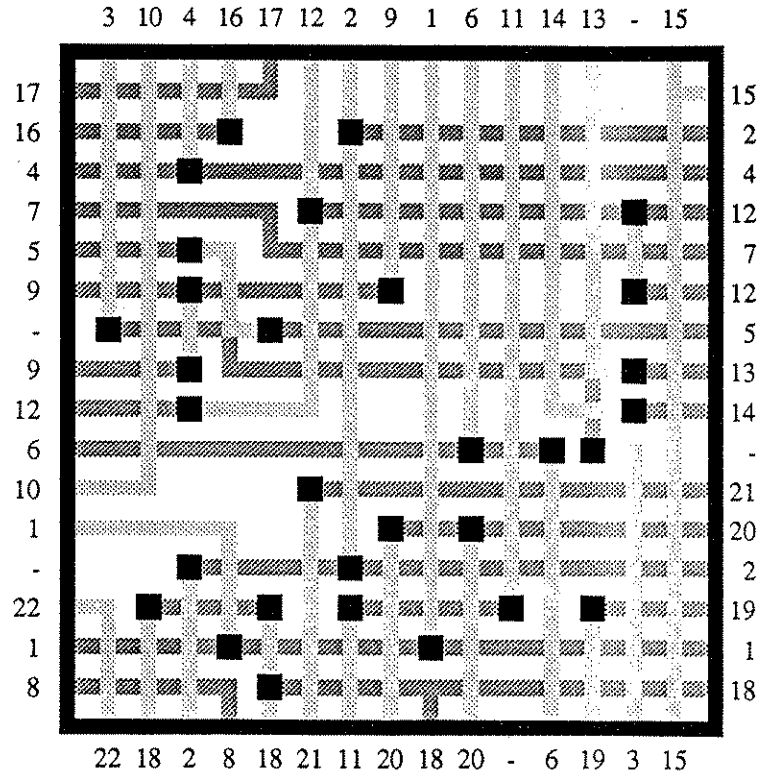


Figure 15. — BEAVER's solution for the Pedagogical Switchbox.

3.5. BEAVER and the Pedagogical Switchbox

BEAVER's default solution to the Pedagogical Switchbox is characterized in Table I. The solution was determined in 1.12 seconds on a Sun 3 workstation. The breakdown of BEAVER's solution was as follows. The corner router determined 39% and layered 19% of the solution in only 3% of the total time. The line sweep router determined 53% and layered 58% of the solution in 88% of the total time. The thread router determined 8% and layered 13% of the solution in 7% of the total. The layerer assigned 10% of the connections in 2% of the total time.

Tool	Vias Assigned	Wire Determined (Units)	Wire Layered (Units)	Time (Seconds)
Corner Router	1	155	76	0.04
Controlled Line Sweep Router	15	207	224	0.80
Relaxed Line Sweep Router	2	4	4	0.18
Thread Router	7	30	51	0.08
Layerer	6	–	41	0.02
Totals	31	396	396	1.12

Performance Summary of BEAVER's Tools on Pedagogical Switchbox
Table I

4. EXPERIMENTAL RESULTS

There are several classic pathological instances available in the literature and BEAVER has successfully routed all of them. These instances include Burstein and Pelavin's Difficult Switchbox [BURS83], two variations of Luk's Dense Switchbox [LUK85], Luk's Terminal Intensive Switchbox [LUK85], Soukup's Sample Switchbox [SOUK81], and Ho, Yun, and Hu's Simple Rectangular Switchbox [HO85]. In the section below, we report on BEAVER's solutions to these problems and statistically compare them to existing routers' solutions. The primary figures of merit are total wire length, number of vias, and running time. Regarding the via figure of merit, we note that not all previous routers were equally concerned with via usage and that some of their solutions can be improved through post-processing with via-reduction algorithms [KAJI80, MARE84]. We note also that BEAVER was implemented in C and consists of approximately 7500 lines of code and comments and that it was run on a Sun 3 workstation using Sun OS 3.3. Although the other routers were typically implemented in C (the most notable exception is Joobbani and Siewiorek's WEAVER implemented in OPS5), they were run on various machines (e.g. Shin and Sangiovanni-Vincentelli's MIGHTY was run on a VAX 11/785 and Enbody and Du's router was run on an Apollo 330). Therefore running time differences on the order of several seconds should be considered insignificant.

BEAVER's solution to the Difficult Switchbox is presented in Figure 16. For this solution, BEAVER was invoked with its track and column control parameter set to 65% (if it is invoked with less control in effect, a solution with slightly greater wire usage is found). All connections were completed by the corner and line sweep router except for net 3's two terminals which were thread routed. We know of six other routers that have solved the switchbox—Hamachi and Ousterhout's switchbox router, DETOUR [HAMA84]; Luk's greedy

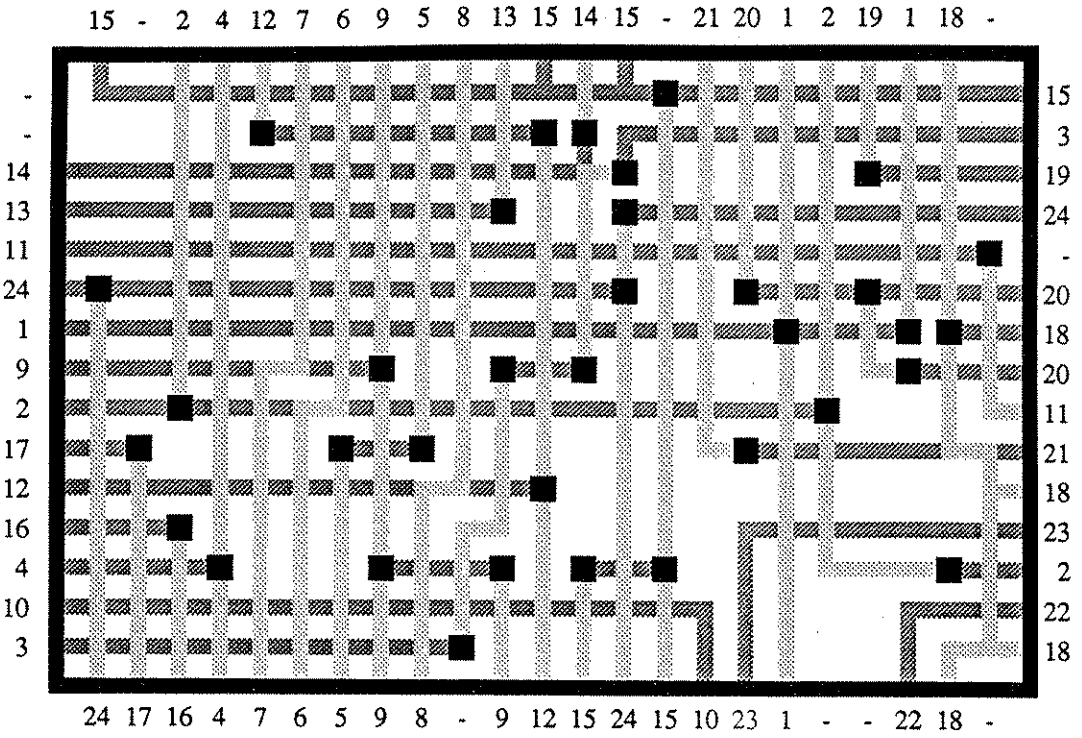


Figure 16. — BEAVER's solution to the Difficult Switchbox.

switchbox router, GSR [LUK85]; Marek-Sadowska's building block layout switchbox router, BBLSR [MARE85]; Hsieh and Chang's DETOUR-like router, MDR [HSIE85]; Joobbani and Siewiorek's WEAVER [JOOB85, JOOB86]; and Kaplan's scanning window router, SWR [KAPL87]. Their solutions along with BEAVER's are characterized in Table II (in this and subsequent tables, routers with entries of '?' did not report their running times). As seen in Table II, BEAVER's solution compares favorably to previous solutions. Two of the other six routers required that the instance be augmented by either adding an extra track or manually pre-wiring a difficult net. BEAVER did not require any such augmentation, its via usage was minimal (15% better than the second best solution), and its wire usage was third best. Although the percentage difference between BEAVER's solution and the best solution was less than 3%, there is a significant difference in the associated running times—WEAVER ran for 1508 seconds and BEAVER ran for one second. This dramatic difference in running times cannot be attributed to different host machines. Rather, WEAVER pays a heavy penalty for its expert system implementation.

Router	Vias	Length	Extra Tracks	Prewired Nets	Time (Seconds)
DETOUR	67	564	–	1	?
GSR	58	577	1	–	1
MDR	63	567	–	–	?
BBLSR	59	560	–	–	5
WEAVER	41	531	–	–	1508
SWR	65	543	–	–	3
BEAVER	35	547	–	–	1

Router Comparison for the Difficult Switchbox
Table II

A restrictive variant of the Difficult Switchbox, the More Difficult Switchbox, was routed by Shin and Sangiovanni-Vincentelli's rip-up router MIGHTY [SHIN86] and Enbody and Du's general purpose router GPR [ENBO87]. The variant removes the switchbox's last column. This column can be removed as it is an overflow column with no terminals. BEAVER's solution is presented in Figure 17 and characterized along with MIGHTY's and GPR's solution in Table III. As indicated in the table, BEAVER's solution used both the least number of vias and the least wire.

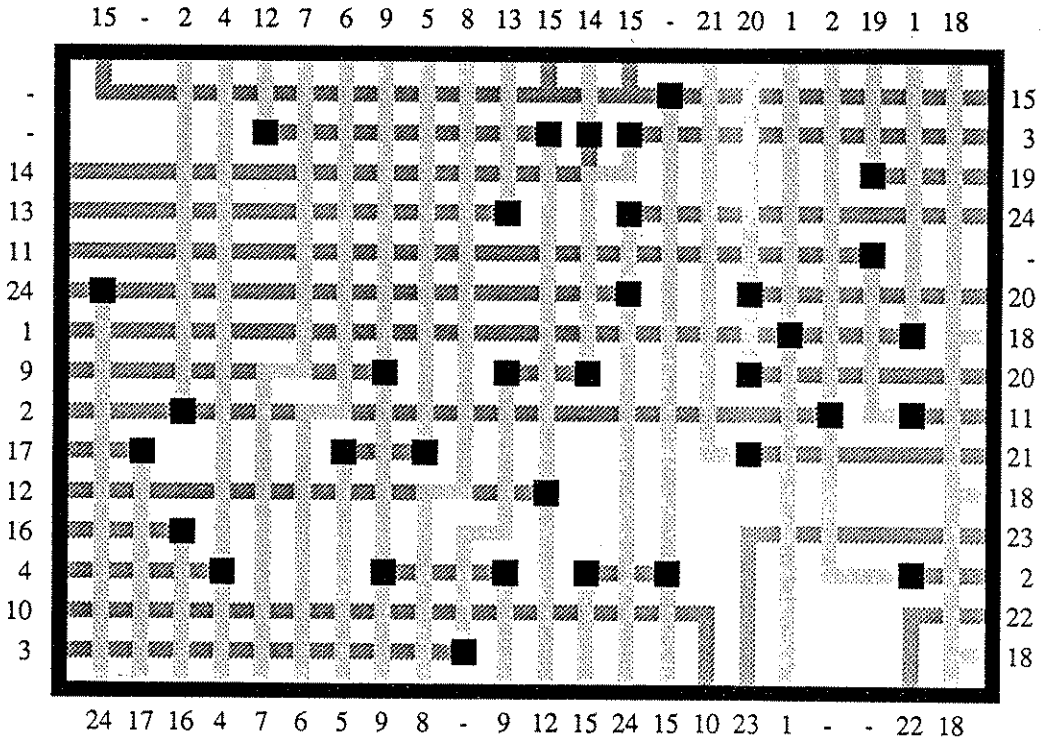


Figure 17. — BEAVER's solution to the More Difficult Switchbox.

Router	Vias	Length	Time (Seconds)
MIGHTY	39	541	4
GPR	64	539	70
BEAVER	34	536	1

Router Comparison for the More Difficult Switchbox
Table III

As a second example, we report on BEAVER's experiences with the Dense Switchbox. It is conjectured that this instance has no solution [LUK85]. However, there are two variants with supplementary routing regions that can be routed. One variant is the Augmented Dense Switchbox that adds an extra column on the right-side of the switchbox and an extra track at the bottom of the switchbox. BEAVER's solution to the Augmented Dense Switchbox is presented in Figure 18. This switchbox has also been routed by GSR and MIGHTY. Their solutions and BEAVER's are characterized in Table IV. As indicated, BEAVER's solution again used the least

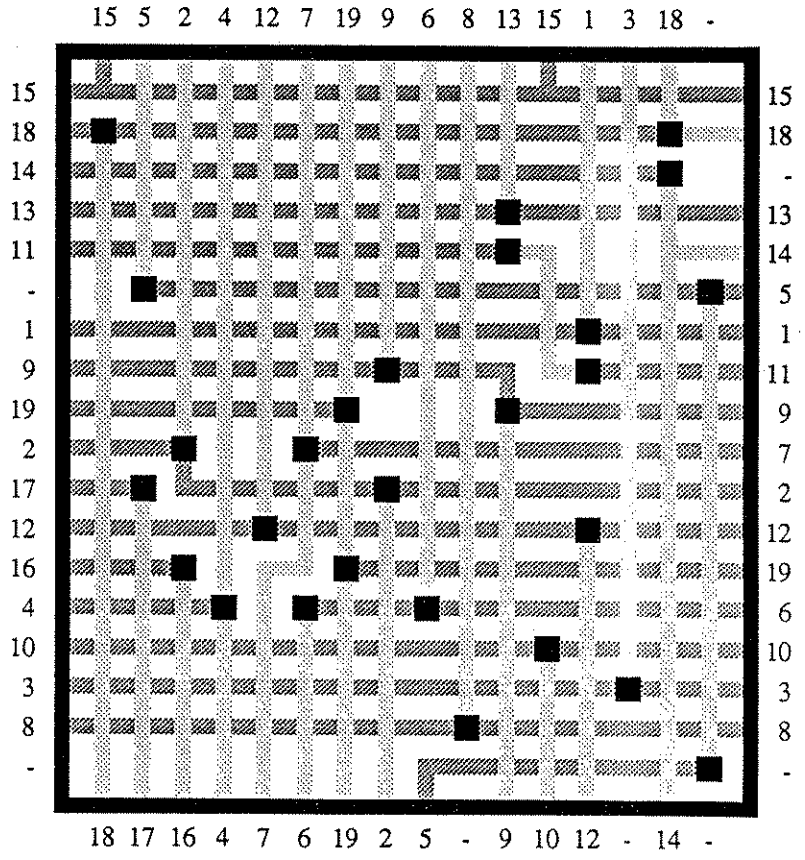


Figure 18. — BEAVER's solution to the Augmented Dense Switchbox.

number of vias and all three routers used virtually the same amount of wire.

Router	Vias	Length	Time (Seconds)
GSR	36	529	?
MIGHTY	32	530	?
BEAVER	27	529	1

Router Comparison for the Augmented Dense Switchbox
Table IV

The other variant of the Dense Switchbox is the Modified Dense Switchbox that adds only an extra column near the middle of the switchbox. BEAVER's solution to this problem is presented in Figure 19. It was produced, as with the Augmented Dense Switchbox, by setting the track and column control parameter to 40%. Neither variant required the use of the thread router. This switchbox has been also routed by MIGHTY and WEAVER. Their solutions and BEAVER's are characterized in Table V. BEAVER's solution again used the

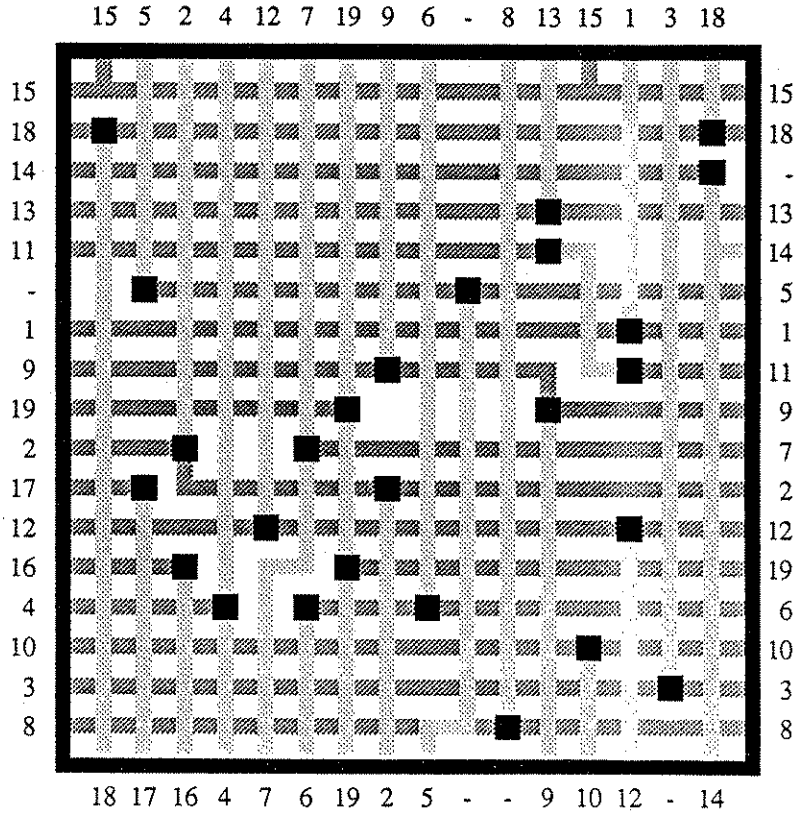


Figure 19. — BEAVER's solution to the Modified Dense Switchbox.

least number of vias and its wire length was the same as the other solutions.

Router	Vias	Length	Time (Seconds)
MIGHTY	29	510	?
WEAVER	29	510	924
BEAVER	26	510	1

Router Comparison for the Modified Dense Switchbox
Table V

As a third example, we report on BEAVER's solution to the Terminal Intensive Switchbox. The solution is presented in Figure 20 and was produced by setting the track and column control parameter to 75%. The thread router was necessary here to connect two subnets of net 2. We know of only three other switchbox routers that are able to produce solutions for the switchbox—GSR, WEAVER, and MIGHTY. Their solutions and BEAVER's are characterized in Table VI. BEAVER's unaugmented solution once again used a minimal number of vias and its wire length was the same as GSR's solution.

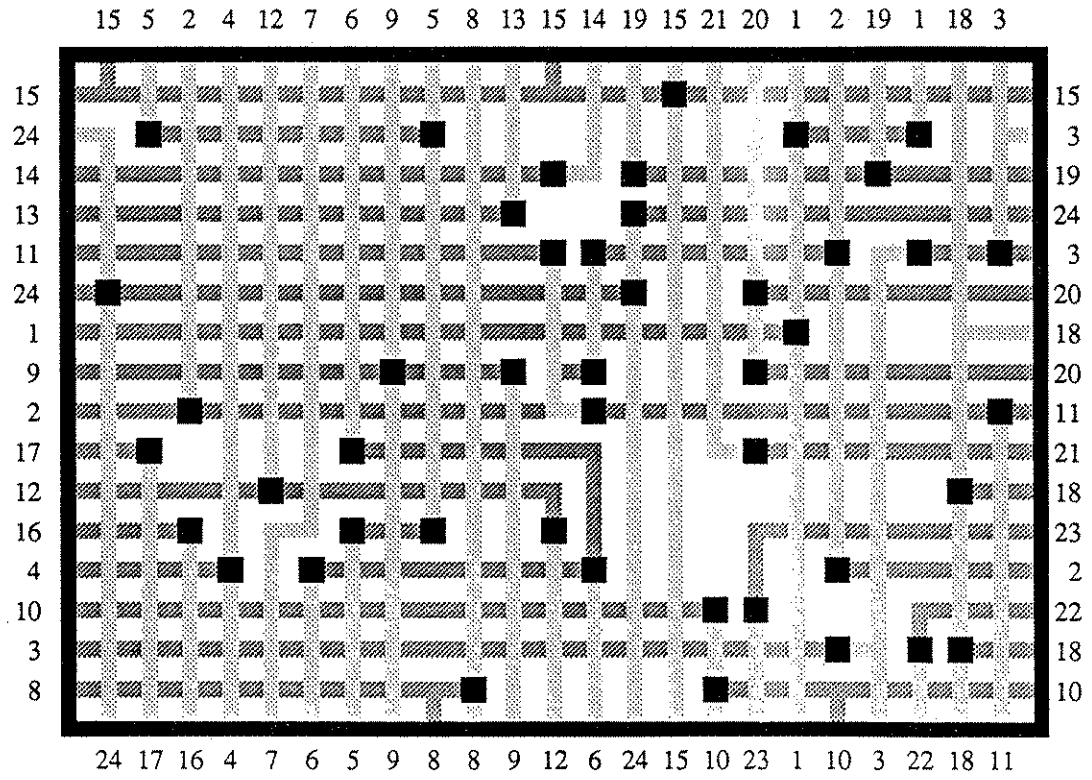


Figure 20. — BEAVER's solution to the Terminal Intensive Switchbox.

Router	Vias	Length	Time (Seconds)
GSR	68	632	?
MIGHTY	50	629	?
WEAVER	49	615	1847
BEAVER	46	632	1

Router Comparison for the Terminal Intensive Switchbox
Table VI

As a fourth example, we report on BEAVER's solution to the Sample Switchbox. This instance was used by Soukup in his survey of physical design to show the relative hardness of the switchbox problem [SOUK81]. BEAVER's solution is presented in Figure 21. It was produced using default track and column control. This instance has also been routed by MIGHTY and WEAVER. Their solutions along with BEAVER's are characterized in Table VII. All three routers used the same amount of wire. However, BEAVER again used the least number of vias.

Router	Vias	Length	Time (Seconds)
MIGHTY	5	60	?
WEAVER	4	60	73
BEAVER	3	60	1

Router Comparison for the Sample Switchbox
Table VII

As a final example, we report on BEAVER's solution to the Simple Rectangular Switchbox. This small instance was used by Ho, Hu, and Yun to demonstrate pedagogically their switchbox router, PLANNER [HO85]. BEAVER's solution used eleven less vias and two more units of wire. The solution is given in Figure 22 and is

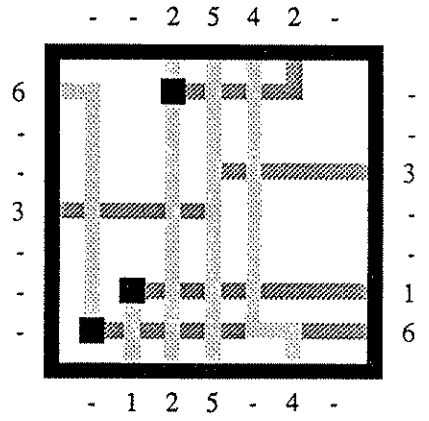


Figure 21. — BEAVER's solution to the Sample Switchbox.

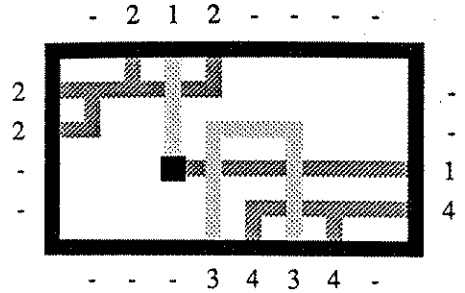


Figure 22. — BEAVER's solution to the Simple Rectangular Switchbox.

characterized along with PLANNER's solution in Table VIII.

Router	Vias	Length	Time (Seconds)
PLANNER	12	29	?
BEAVER	1	31	1

Router Comparison for the Simple Rectangular Switchbox
Table VIII

Table IX supplies several important performance statistics for the various switchbox solutions produced by BEAVER. However, several other statistics concerning these six switchboxes are also worth mentioning. On average, the corner router completed 48% of the wiring solution, the line sweep router completed 48%, and the thread router completed the remaining 4%. In the worst case the corner router was only able to complete 13% of the wiring solution (the Sample Switchbox) while in the best case it was able to complete 76% (the Augmented Dense Switchbox). The line sweep router's lowest completion rate was 24% (the Augmented Dense Switchbox) and its highest was 87% (the Sample Switchbox). The thread router was not required for three of the instances (the Sample Switchbox and both variations of the Dense Switchbox), but in the worst case it was required to complete 8% of the wiring solution (the Pedagogical Switchbox). The layer assigner layered 11% of the wires on average with a low of 4% (the Modified Switchbox) and a high of 25% (the Sample Switchbox). Finally, we note that the solution reported for the Pedagogical Switchbox in Table IX is different than the one reported in Table I. The solution in Table IX uses one less via and four less units of wire. The solutions in Tables I and IX were derived using respectively 50% (default) and 100% as the percentage of a track or column that is to be reserved for an unrealized net with a terminal at the end of the track or column.

5. ALGORITHMIC SENSITIVITY

An algorithm's solution quality is sometimes sensitive to the form of the input. With respect to the switchbox problem, routers may be sensitive to switchbox orientation (i.e. which side of the switchbox is considered

Statistics	Switchbox							
	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
Number of Rows	16	15	15	18	17	16	7	4
Number of Columns	15	23	22	16	16	23	7	8
Number of Nets	22	24	24	19	19	24	6	4
Number of Terminals	58	66	66	60	60	78	13	11
Control Value (Percentage)	100	65	65	65	40	75	50	50
Number of Vias	30	35	34	27	26	46	3	1
Wire Length in Units	392	547	536	529	510	632	60	31
Corner Routing (Seconds)	0.02	0.04	0.04	0.06	0.06	0.05	0.01	0.01
Controlled Line Sweep Routing (Seconds)	0.48	1.03	0.92	0.13	0.17	0.72	0.08	0.08
Relaxed Line Sweep Routing (Seconds)	0.11	0.13	0.11	0.00	0.00	0.16	0.00	0.00
Thread Routing (Seconds)	0.05	0.06	0.06	0.00	0.00	0.07	0.00	0.00
Layering (Seconds)	0.01	0.03	0.03	0.03	0.03	0.04	0.01	0.01
Total Time (Seconds)	0.67	1.29	1.16	0.22	0.26	1.04	0.10	0.10

Label	Name
S_1	Pedagogical Switchbox
S_2	Difficult Switchbox
S_3	More Difficult Switchbox
S_4	Augmented Dense Switchbox
S_5	Modified Dense Switchbox
S_6	Terminal Intensive Switchbox
S_7	Sample Switchbox
S_8	Simple Rectangular Switchbox

Detailed Performance Statistics for Various Solutions Produced by BEAVER
Table IX

"up"). Unlike some previous switchbox routers (e.g. Luk's GSR), BEAVER displays minimal sensitivity with regard to orientation. For each of the eight switchbox instances described in the previous sections, BEAVER was individually run using the four possible orientations. Although a few nets occasionally used slightly different paths, for each problem the derived solutions were constant with respect to the size of the routing

region, total wire length, and via usage.

In contrast to input sensitivity, where an algorithm may or may not be sensitive to its representation, an algorithm is typically sensitive to the values of its parameters—otherwise the existence of the parameters serves no purpose. BEAVER has one parameter and its value does affect solution quality. This was evidenced in the previous section by the reporting of an alternative (better) solution for the Pedagogical Switchbox. The alternative solution was obtained using a control parameter value of 100% rather than using the default value of 50%.

To more thoroughly test BEAVER's control parameter sensitivity with respect to the eight switchboxes described in the previous sections, BEAVER was run on each of these switchboxes with the control parameter value varying from 100% to 0% in decrements of 5%. We found for each of the switchboxes, that BEAVER's best solution was obtained when the control parameter was set to the maximum value for which BEAVER could return a feasible solution. For example, for the Modified Dense Switchbox with the control parameter set to 45% or higher, BEAVER could not find a solution; with the control parameter set to 40%, BEAVER found a solution that was equivalent or better than the ones found with control parameter set to 35% or less. We believe from examining our switchbox solutions that using the maximal value for the control parameter produces the best solutions because it allows a crucial terminal to extend just sufficiently outward to realize a connection that is globally desirable. If BEAVER is run with a higher control parameter value, a different connection is selected that is better for the net in question. However, this new connection extends further outward from the crucial terminal and prevents another net from being realized.

We also found in our experiments that the best solution was found in six of the eight switchboxes with the control parameter set to 65%. Such a value gives a net exclusive control of approximately one-third of the rows and tracks emanating from its terminals and shared control of the middle-third of the rows and tracks emanating from its terminals. This shared control of the middle-third enabled almost all nets to make straightforward, minimal length connections.

It is our recommendation based on the above findings, that a BEAVER user with no prior knowledge of the switchbox first set the control parameter to 65%. If the solution produced with that value is unacceptable then iteratively increase the control parameter by 5% until either an acceptable solution is found or until a value of 100% has been attempted. If an acceptable solution has not yet been determined, then set the control parameter to 60%. If the solution produced with that value is unacceptable then iteratively decrease the control parameter by 5% until either an acceptable solution is found or until a value of 0% has been attempted. We tried such a procedure on each of the eight switchboxes described in the previous sections. The best solution was found on average in less than three seconds and was always found in less than seven seconds.

6. VARIATIONS

Since BEAVER is a switchbox router and since a channel is a degenerate switchbox, by default BEAVER can route channels. However, for channel instances BEAVER is not able to make any use of one of its principal tools—the corner router. Therefore, to make BEAVER more effective on channels we are currently developing and evaluating a pre-processor that decomposes a channel into a collection of sub-problems. The sub-problems are channels with an *L-list* and an *R-list*, where an *L-list* is an ordering of the nets that enters the channel from the left and an *R-list* is an ordering of the nets that exit the channel to the right. Orderings are determined by analyzing the net list's vertical constraint graph. From our preliminary experiments, this decomposition reduces the running time and improves solution quality. We believe these improvements occur because BEAVER is given more guidance on which rows are appropriate for the nets contained in the various *L-list*'s and *R-list*'s.

Another routing situation to which BEAVER can be adapted is a switchbox or channel that contains obstacles (e.g., modules or preplaced wires). This adaptation was simple to incorporate—upon initialization the control values of the various rows and columns that contain obstacles are simply adjusted to reflect the existence of the obstacles. Figure 23 shows BEAVER's solution for a small switchbox that contains two rectilinear obstacles. We believe that with a slightly more generalized decomposition procedure than our channel decomposition procedure, BEAVER can also effectively route irregular-shaped rectilinear regions.

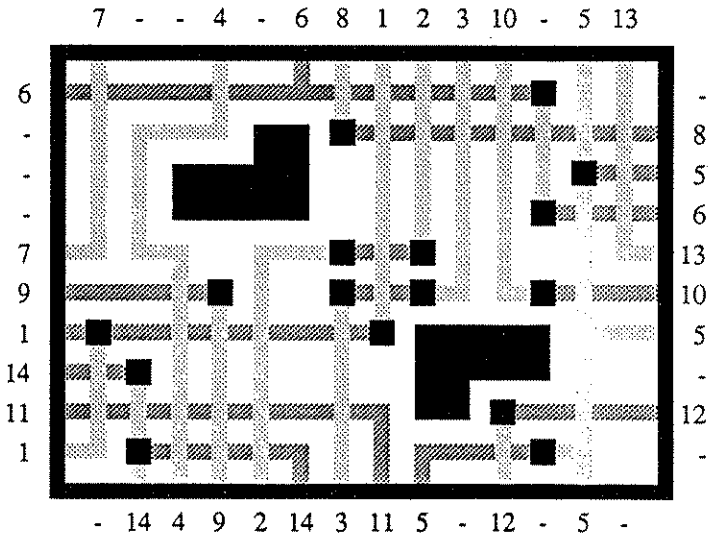


Figure 23. — BEAVER's solution for a switchbox with embedded obstacles.

7. FINAL REMARKS

We have described a new fast router BEAVER that is intended to be the initial router invoked to solve a switchbox. BEAVER has been run successfully on several pathological and pedagogical switchboxes. In all instances, its via usage was better than the previously best known solution and its wire length was either better or comparable to the previously best known solution. BEAVER typically required a second or less on a Sun 3 workstation to solve an instance and never more than two seconds. In addition, through pre-processing BEAVER can effectively handle other routing problems (e.g., channels or routing regions embedded with obstacles).

8. ACKNOWLEDGEMENTS

The comments and suggestions of the referees were greatly appreciated. Their remarks contributed to the quality of discussion in several sections of this paper.

As always, we thank Joanne Cohoon and Cynthia Heck for their support, suggestions, and comments.

This research was supported in part by the National Science Foundation under grant DMC 85-05354 and by the Virginia Center for Innovative Technology under grant TDC 86-002-01. Their support is greatly appreciated.

9. REFERENCES

[BENT79] J. L. Bentley and T. Ottmann, Algorithms for Reporting and Counting Geometric Intersections, *IEEE Transactions on Computers C-28*, 9 (September 1979), 643-647.

[BURS83] M. Burstein and R. Pelavin, Hierarchical Wire Routing, *IEEE Transactions on Computer-Aided Design CAD-2*, 4 (October 1983), 223-234.

[ENBO87] R. J. Enbody and H. C. Du, General Purpose Router, *24th Design Automation Conference Proceedings*, Miami Beach, FL, 1987, 637-640.

[HAMA84] G. T. Hamachi and J. K. Ousterhout, A Switchbox Router with Obstacle Avoidance, *21st Design Automation Conference Proceedings*, Albuquerque, NM, 1984, 173-179.

[HASH71] A. Hashimoto and J. Stevens, Wire Routing by Channel Design, *8th Design Automation Workshop Proceedings*, Atlantic City, NJ, 1971, 155-169.

[HO85] W. P. Ho, D. Y. Y. Yun and Y. H. Hu, Planning Strategies for Switchbox Routing, *Proceedings of the International Conference on Computer Design: VLSI in Computers & Processors*, Port Chester, NY, 1985, 463-467.

[HORO78] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*, Computer Science Press, Potomac, MD, 1978.

[HSIE85] Y. Hsieh and C. C. Chang, A Modified Detour Router, *Proceedings of the International Conference on Computer-Aided Design*, Santa Clara, CA, 1985, 301-303.

[HU85] T. C. Hu and E. S. Kuh, eds., *VLSI Circuit Layout: Theory and Design*, IEEE Press, New York, NY, 1985.

[JOOB85] R. Joobbani, *WEAVER: An Application of Knowledge-Based Expert Systems to Detailed Routing of VLSI Circuits*, Ph.D. Thesis, Department of Electrical and Computer Engineering, Carnegie-Mellon University, April, 1985.

[JOOB86] R. Joobbani and D. P. Siewiorek, Weaver: A Knowleged-Based Routing Expert, *IEEE Design & Test 3*, 1 (February 1986), 12-23.

[KAJI80] Y. Kajitani, On Via Hole Minimization of Routing on a 2-Layer Board, *International Conference on Circuits and Computers Proceedings*, Port Chester, NY, Octover 1980, 295-298.

[KAPL87] D. Kaplan, Routing with a Scanning Window, *24th Design Automation Conference Proceedings*, Miami Beach, FL, 1987, 629-632.

[LEE61] C. Y. Lee, An Algorithm for Path Connections and Its Applications, *IRE Transactions on Electronic Computers EC-10*, 3 (September 1961), 346-365.

[LUK85] W. K. Luk, A Greedy Switch-Box Router, *Integration, the VLSI Journal 3*, (1985), 129-149.

[MARE84] M. Marek-Sadowska, An Unconstrained Topological Via Minimization Problem for Two-Layer Routing, *IEEE Transactions on Computer-Aided Design CAD-3*, 3 (July 1984), 184-190.

[MARE85] M. Marek-Sadowska, Two-Dimensional Router for Double Layer Layout, *22nd Design Automation Conference Proceedings*, Las Vegas, NV, 1985, 117-123.

[MOOR59] E. F. Moore, Shortest Path Through a Maze, *Annals of the Computational Laboratory of Harvard University 30*, (1959), 285-292.

[PREP85] F. P. Preparata and M. I. Shamos, *Computational Geometry*, Springer-Verlag, New York, 1985.

[SHIN86] Y. Shin and A. Sangiovanni-Vincentelli, MIGHTY: A 'Rip-Up and Reroute' Detailed Router, *Proceedings of the International Conference on Computer-Aided Design*, Santa Clara, CA, 1986, 2-5.

[SOUK78] J. Soukup, Fast Maze Router, *15th Design Automation Conference Proceedings*, Las Vegas, NV, 1978, 100-102.

[SOUK81] J. Soukup, Circuit Layout, *Proceedings of the IEEE 69*, 10 (October 1981), 1281-1304.