An iterative-improvement penalty-function-driven wire routing system

by Ralph Linsker

A wire routing system (VIKING) has been developed for interconnection packages. It uses iterative-improvement methods that allow "illegalities" (such as wire crossings within a plane) at intermediate stages of the routing, eliminates some drawbacks of conventional sequential routers, and extends the range of penalty functions with respect to which a wiring configuration can be optimized. Efficient routing in directionally uncommitted planes is provided; specification of preferred-direction (*x* and *y*) planes is optional but not required. Significant reductions in required manual embedding effort, number of vias required, routed wire length, and the number of signal planes required to wire a package, have been found, compared with sequential routers that have been used. Improved automatic control of electrical crosstalk noise has also been provided. In addition to presenting VIKING methods and results, we discuss other issues relating to wiring methods and global optimization. Application of these methods to chip design is also discussed.

©Copyright 1984 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the *Journal* reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied or distributed royalty free without further permission by computer-based and other information-service systems. Permission to *republish* any other portion of this paper must be obtained from the Editor.

1. Introduction

This paper describes some features of a wire routing system (VIKING) that has been developed for interconnection packages, and discusses related strategies for improving wiring density and quality on chips and packages.

Given a description of the rules governing legal interconnections (the board model) and given a net list (each net is a set of locations ("pins") to be mutually connected), the routing problem for a chip or interconnection package consists of finding a "legal" set of paths that accomplishes the required interconnections. More generally, one may wish to near-optimize the wiring configuration with respect to a function that encompasses not only "illegal" but also other undesirable wiring features. The board model typically specifies the required spacings between wires, locations of blockages and of available inter-plane electrical connections ("vias"), number of wiring planes and dimensions of each plane, allowed wiring directions, and so forth.

Automatic routers usually attempt to route wires sequentially, and do not allow illegal configurations at intermediate stages of the routing. The initial routing passes typically leave a set of "failed" (unrouted) connections or "overflows"; specialized algorithms may then be applied to reduce further the number of overflows. Residual overflows need to be either manually routed according to board model rules, or connected by other means (e.g., using discrete wires).

We have used instead an iterative improvement strategy, according to which one generates an initial wiring configuration which is successively improved by optimizing the routing, one connection at a time, with respect to a

sequence of penalty functions. Illegal intermediate wiring configurations are permitted. Independent work on iterative-improvement routers has been described by Moore and Ravitz [1], and earlier by Rubin [2].

Section 2 describes the basic operation of VIKING, including the global router (with automatic assignment of connections to package layers) and the detailed router. A particular application of the methods is to the case of routing in "directionally uncommitted planes"; that is, wiring planes for which no preferential (x or y) routing direction has been specified. This ability, and the related ability to perform extensive "wrong-way" wiring (e.g., x wiring in a predominantly y plane), offer significant wirability advantages in a variety of situations, and are discussed in the section on "free-form" wiring.

Extensions of the basic VIKING methods are described in Section 3. Section 4 describes a class of mazerouter algorithms that, in effect, can calculate an optimal path with respect to a penalty function that can vary nonlinearly with the number of occurrences of a given penalty type along the path (i.e., a "path-history-dependent" penalty function). This is useful when the desired penalty function incorporates crosstalk thresholds, minimum and maximum length constraints, and the like.

Section 5 describes a simplified version of a router (not a part of VIKING proper) that illustrates some aspects of iterative routing with an adjustable penalty function. The use of such an approach, to avoid settling into locally optimal configurations that may be far from globally optimal, is compared with the Monte Carlo simulated annealing method [3].

Section 6 includes results and discussion for several types of VIKING applications. Section 7 gives conclusions.

We note that many of the methods discussed below can be used to near-optimize a routing configuration on an arbitrary graph. In particular, some of these methods may have useful application to the routing of traffic on communications networks and the like, as well as on wiring grids.

2. VIKING system: basic operation

The VIKING wiring system is organized as follows. The board model and connection list are provided as input. To simplify the discussion, let us first consider the following common situation:

- A printed interconnection package is being routed (i.e., any grid point occupied by two paths in the same wiring plane represents a short and is illegal).
- Routing is in the x and y directions on a rectangular grid.
- Each net (i.e., each set of pins to be mutually connected) is specified as a set of two-pin connections.
- A more technical point: If a mazerouter algorithm is being used to route a connection, then we assume (for the present) that the penalty function being minimized does not embody threshold penalties or constraints. It should depend linearly on the number of occurrences of each penalty type along a path. For example, a specified penalty per unit length should not depend upon the total length of the path.

After discussing the operation of the system for the above case, we treat the following situations (see Sections 3 and 4):

- Technologies in which wires may cross each other (subject to constraints) within the same plane.
- Diagonal or "eight-way" wiring on a square grid.
- Dynamic decomposition of multi-pin nets into sets of two-pin connections.
- "T-ing"; i.e., cases in which paths of the same net are allowed to join at points other than pins.
- Path-history-dependent penalty functions. A mazerouter can be used to, in effect, penalize an occurrence differently depending upon whether it is the only such, or one of many such, occurrences along the path. This facilitates the imposition of timing (minimum and maximum length) constraints, crosstalk threshold controls (wherein inter-net adjacency is penalized only if the extent of adjacency exceeds a given value), and the like.

- *Global (coarse) router*

If a global routing is to be done (this is optional), the board model includes a specification of the coordinates (on the detailed or fine grid) of the edges of the rectangular global cells. An edge capacity is inferred for each global cell edge using board model information. This capacity describes the effective number of available horizontal or vertical wiring tracks passing through that global cell. Each layer of the global grid typically, but not necessarily, corresponds to a plane pair of the detailed grid. For package applications in which we have used global routing, each global path is confined to one global layer (though the global router determines dynamically which layer assignment is optimal); but this also is not essential.

The global routing configuration is initialized by assigning to each connection a trial path, which provides a reasonable background of congestion against which to start rerouting connections. A series of iterative passes is then performed. In each pass, all (or some) of the connections are, one at a time, removed from and rerouted on the global grid. The rerouting typically uses a mazerouter algorithm [4] (e.g., of Lee type [5]), but may be by other means (e.g., the "LZU" method described in Section 5).

Each connection is to be optimized in turn, against the background of the others, according to a specified penalty function (for that pass). The penalty function is chosen to have the form

$$PENFUN = \sum [(W_X \text{ or } W_Y) + (BEND, \text{ if bend present}) + CONGESTIONPEN],$$

where the sum is over the links of the path (each link by definition crosses one global cell edge), a cost W_X or W_Y is assigned depending on link direction, and the congestion penalty for each link is a piecewise-linear function depending on the load (the number of paths, other than the one being routed, that pass through the cell edge) and of the edge capacity defined earlier:

$$CONGESTIONPEN = CONG_1 \times \max [0, (LOAD - THRESH_1 \times CAPAC)] + CONG_2 \times \max [0, (LOAD - THRESH_2 \times CAPAC)].$$

Penalty parameters (in particular W_X, W_Y, and $BEND$) can vary with the wiring layer and/or region within a layer, and can differ for different subclasses of connections, if desired. For example, wiring through specified regions can be inhibited by increasing W_X and W_Y or decreasing edge capacities in those regions.

The penalty function parameters are typically chosen by deciding, for the particular problem, on an acceptable general level of wiring congestion. On successive passes, routing through global cell edges having excessive congestion is penalized increasingly (relative to length costs), so that affected connections gradually lengthen in order to avoid congested regions. A satisfactory global routing is thereby accomplished with balanced congestion within and among layers.

Output from the global router includes the coordinates of path segments (on the global grid) for each connection, and diagnostic information describing the degree of congestion imbalance, routed length on the global grid, etc., after each pass. Penalty parameters for the next pass (if any) can be determined based on these diagnostics.

- *Detailed router*

The above description of the global router facilitates discussion of the detailed router, which operates in a similar way. Many global routers have used iterative-improvement techniques to balance congestion, by using information from the trial global routing of all connections after one pass to determine improved routings for the succeeding pass. It is striking that this approach has not generally been applied to the detailed routing problem (Refs. [1] and [2] are exceptions); this may have to do with the idea that intermediate wiring configurations should not embody illegalities such as shorts between different nets. (An imbalance in global congestion, on the other hand, is undesirable but not illegal.)

An initial detailed routing configuration, which makes use of the results of a prior global routing (if one has been performed), is defined. A series of iterative passes is then performed, with penalty parameters specified for each pass, as was done for the global routing. A Lee-type mazerouter or line probe router [6] is used to optimize the rerouting of each connection in turn. The penalty function is the sum of the following terms:

- Length costs (per grid unit) for x and y directions in each plane.
- Node overload costs (for two or more wires, or a wire and a blockage, occupying the same gridpoint in the same plane).
- Via and bend costs.
- Adjacency costs (for each grid unit in which a path is routed adjacent to another path, within the same plane or in adjacent planes).
- Other penalties as required.

Different penalty parameters may be specified for different regions of the board, or for different subsets of connections, as required. The node overload penalty (for example) may be varied depending upon the location of other overloaded nodes on the grid; this can be used to inhibit the clustering of overloaded nodes, for cases in which clustered illegalities may be more difficult to repair automatically or manually.

One may, at each pass, either constrain the routing to lie within the global cells traversed during the final global routing pass (if any), or expand the region available for routing, or eliminate the constraint altogether. An alternative approach is to include a penalty term for each grid unit in which a path is routed outside (or at some distance from) the global swath for that connection.

On later passes, computing time can be reduced by rerouting only a subset of the connections. The penalty function is tailored to improve routing success for these connections.

We indicate some ways in which the flexibility of an iterative-improvement penalty-function-driven router can be exploited to improve wiring success.

- *"Free-form" wiring*

In other work on multi-plane package routing, one typically is advised not to set weights equal for x and y routing within the same region of a plane, and particularly not during the stage in which most routings are being made. One reason is the belief that the resulting "wrong-way wiring" would lead to large numbers of blocked escapes. This can be a significant problem for sequential routers, since earlier-wired connections could block x and y tracks in all planes, impairing the ability of later-routed connections to escape from their source pins or otherwise complete their routings. As we shall illustrate, the present iterative methods

accommodate "free-form" wiring (i.e., wiring in planes for which no preferred wiring direction has been specified) without difficulty. There are no "earlier" or "later" routed connections. Crossings occurring in the initial passes are gradually resolved, and regions of predominantly x or y wiring in each plane can form dynamically, without user intervention. Advantages of free-form or extensive "wrong-way" wiring can include the following, depending on the application:

- Reduction in via count.
- Improved balancing of wiring load among planes, when some planes have extensive blockages (e.g., macro blockages on a chip).
- Improved wirability when component placement dictates regional imbalances between x and y wiring loads.
- Increase in number of wiring channels available for "escape" from pins lying in congested regions (e.g., under dense components on interconnection boards).
- Finally, methods that enhance free-form wirability are of obvious utility for wiring of single-plane boards.

- *Penalty parameter choices*

One strategy [1, 2] is to gradually increase the overloaded-node penalty in relation to the wire-length penalty, as one proceeds from early to later iterative passes. We have also found other types of penalty schedules that aid convergence to a near-optimal configuration while avoiding undesirable metastable configurations. Put another way, one can escape from local minima of a penalty function by allowing "uphill" moves, or (as in the present method) by appropriately changing the parameters in the penalty function itself.

- *Manual embedding simplified*

As an important by-product of the present method, all illegal connections are typically left on the board. This contrasts with sequential routers, which "fail" unrouted connections and require the manual embedder to route them in their entirety against a congested board. The types of illegalities (usually wire crossings) that remain after a VIKING run are usually correctable by strictly local rework, consisting of moving several paths in the vicinity of the crossing in order to create space within which the crossing can be resolved. These corrections are far easier and faster than generating entire paths for unrouted connections.

If it is deemed desirable, wires involved in more than a specified number of crossings can be removed from the board during any pass, and their rerouting can be attempted on the succeeding pass. One purpose is to remove from the board those connections that are so poorly routable that they are expected to require discrete wires for their implementation.

3. VIKING system: extensions of basic methods

- *Technologies allowing intra-plane wire crossings*

VIKING has been applied to the routing of individually insulated discrete wires on a multi-layer grid. The node overload cost is modified so that only those crossings that violate the technology rules are penalized.

- *Diagonal or "eight-way" routing*

Within a Lee-type mazerouting algorithm the grid points to be explored starting from a given grid point are its neighbors in the x, y, and z (if a via is available) directions for orthogonal routing, and also its diagonal neighbors in the x–y plane for diagonal routing. The present methods can thus be used to generate diagonal or directionally unconstrained eight-way wire routings in several planes simultaneously.

The global router was modified in a different way to accommodate problems requiring x–y routing in some planes, and diagonal (45-degree) routing in others. Since rectangular components (on the boards to be wired) are oriented along x and y axes, it is convenient to use global cells aligned with component boundaries. Nearly-square global cells of similar size are used. At each global reroute step, a connection is mazerouted (using x and y wiring directions only) using all layers. Each layer is represented internally as a plane pair consisting of strictly x and y wiring planes. Within the diagonal wiring layers, the primitive moves allowed to the mazerouter consist only of traversing a global cell edge in the x or y direction (depending upon which plane the current grid point lies in) followed by a mandatory via (to the other plane of that layer). Thus each completed global path in the diagonal layers consists of one or more "staircases" (each x or y "step" traversing one global cell). The global swath (used to control the detailed routing) is formed as the union of 45-degree diagonal rectangles, each such rectangle enclosing one of the "staircases."

- *Dynamic decomposition of multi-pin nets into two-pin connections*

Suppose the choice of two-pin connections used to implement the wiring of a multi-pin net is not constrained (e.g., by electrical requirements). One may improve upon a specified net decomposition by removing and rerouting, at each iterative pass, one net (rather than one connection) at a time. Use the mazerouter algorithm to find the lowest-cost path connecting any of a set of source pins to any of a set of target pins; repeat until all pins of that net are connected.

- *Multi-pin nets with "T-ing"*

On chips and some printed-circuit boards, paths of a net can join at points that are not pins (termini) of the net. Consider a net consisting of pins A, B, and C. As is well known, one can route the net by decomposing the problem into the steps

of (for example) mazerouting from A to B, then mazerouting from C to any point on the path AB. To perform net interconnections with "T-ing" (joins at gridpoints that are not pins) using the iterative methods discussed, the same changes would be made to the detailed router as were specified for net decomposition into connections (above). A generalized mazerouter algorithm (for multiple source and/or target points) would be substituted for the usual two-point router; and an entire net, rather than a two-pin connection, would be removed and rerouted at each step of an iterative pass. Alternatively, one may remove and reroute a subset of the paths for a net, provided no remaining paths of that net are "left hanging."

There are other mazerouting algorithms [7] that guarantee a lowest-cost path connecting three pins and generalize heuristically to the N-pin case ($N > 3$). These algorithms can be applied directly at the level of the detailed router. If computing time for these algorithms is excessive, however, one can still use them to fix the desired net topology (and approximate location of "T" points) in the following way. Perform an iterative global routing using these algorithms. (This can be fast since the number of global cells is relatively small.) Require the detailed routing (at least in the early passes) to lie within the union of the global cells determined by the global routing. Use a standard mazerouter (that can route between multiple source and target points) to perform the detailed routing of each part of the net on each detailed pass. (In the previous example, one may specify that pin C is to be connected to the portion of path AB that lies within a prescribed global cell; this preserves the approximate location of the "T" point if desired.)

4. Path-history-dependent penalty functions

A mazerouter algorithm labels gridpoints with the minimum cumulative cost incurred in reaching each such gridpoint from the source point(s), and updates these labels as the "expansion" from source to target proceeds. (The label values should include a "minimum cost to target" term, to reduce computing time [8].) The optimal path is calculated by backtracking from the target to the source, after the labeling process has been completed. Accordingly, the mazerouter is not able to "charge" a different penalty for the nth occurrence of some condition during a path routing than for the first occurrence. The penalty function being minimized must be linear in the number of occurrences of each penalty type, for an incremental cost to be assigned to each link of the grid, independent of the remainder of the path.

Nevertheless, it is important to be able to calculate an optimal routing subject to threshold constraints, or to optimize a penalty function that is nonlinear in the number of occurrences of a given penalty type along a path. We can enumerate all allowed paths if the number of such paths is small enough (see Section 5). However, the great virtue of a mazerouter is its ability to find an optimal path in $O(L^2)$ time, when the number of possible paths is exponential in wire length L. Fortunately, there is a way of optimizing a nonlinear penalty function, or one with threshold constraints, while retaining the advantages of a mazerouter algorithm [9]. The method to be described also fits naturally with the iterative-improvement approach used here, and could indeed recommend such an approach even if iterative-improvement routing had no other virtues.

• *Minimum and maximum length constraints*

Suppose that some of the connections must be routed with lengths lying between specified bounds. We have described the loop structure of the global and detailed routers as follows:

1. Initialize wiring configuration;
2. Start an iterative pass by specifying penalty parameter values;
3. Do for some or all connections:
   a. Remove connection from board;
   b. Reroute using mazerouter to optimize connection path with respect to penalty function;
4. Generate diagnostics and wiring layout; exit if done; else go to 2.

To incorporate length constraints, introduce an additional loop by replacing the "remove and reroute" steps by the following: [C_X and C_Y are the "standard" length penalty parameters (costs per grid unit traversed) in the x and y directions, for the present pass. The possible dependence of the C_X, C_Y values upon plane, or region within a plane, is suppressed for notational simplicity. N_{TRIALS} is a specified maximum number of reroute attempts for the inner loop.]

   a. Remove connection from board;
   b. Set C_{XTEMP} and C_{YTEMP} (e.g., to C_X, C_Y, respectively);
   c. Set $TRIAL = 1$;
   d. Use mazerouter to optimize path with respect to penalty function in which C_{XTEMP} and C_{YTEMP} are the penalty parameters, rather than C_X and C_Y;
   e. If length of optimal path satisfies constraints, or if $TRIAL = N_{TRIALS}$, establish this path on the board and go to (i);
   f. Adjust C_{XTEMP} and C_{YTEMP} (increase if path is too long; decrease if too short);
   g. Increment $TRIAL$ by one;
   h. Go to (d);
   i. Done with this connection (on this pass).

Suppose that the board configuration is such that setting $C_{XTEMP} = C_{YTEMP} = 0$ would give a path length greater than the minimum allowed. Then some choice of (C_{XTEMP}, C_{YTEMP}) will typically yield an optimal path whose length lies within the allowed interval.

If, on the other hand, the only penalties incurred for a particular optimal path are length costs, a minimum length constraint might not succeed in being satisfied for that connection on that iterative pass. In that case, there would be no other costs that could be reduced by lengthening the wire path on that pass, even if C_{XTEMP} and C_{YTEMP} were to be set equal to zero. There are then several options:

- Accept the minimum-cost path (although it violates the length constraint), with the possibility that on a future pass other costs (e.g., wire crossings) will arise that result in a lengthening of the path for that connection. Manual or automatic postprocessing may be required to enforce the minimum-length constraint for those connections violating it at the final iterative pass.
- Introduce a fictitious "barrier" between the connection pins (for that connection on that pass), in order to induce a longer route to be chosen.
- Alternatively but less flexibly, one can decompose the connection (AB) into two connections, requiring that the path go from A to C, thence to B.

When the minimum and maximum allowed lengths are very close to one another and not close to "minimum-Manhattan" length, e.g., wires of clock nets, the algorithm described here may not be practical. (Minimum-Manhattan length of a connection between (x, y) and (x', y') is defined as $|x - x'| + |y - y'|$.) It is then preferable to route the clock wires with their required lengths by other means, then use the present methods to route the other connections iteratively. One may either fix the layout of the clock wires and "route around" them; or provide some flexibility by attempting to reroute the clock wires iteratively, but accepting a different path only if it satisfies the length constraints.

- *Crosstalk constraints*

A similar idea can be used to inhibit adjacency between pairs of nets or connections, when the adjacency distance, or the crosstalk noise resulting therefrom, exceeds a specified threshold. We state the method in a general form that may be more elaborate than required for practical implementations.

After routing connection j, calculate some convenient estimator of electrical crosstalk (e.g., adjacency distance) between j and every other (relevant) path k. Call this function $d(j, k)$.

If crosstalk noise is excessive for this routing, reroute connection j using a penalty function including a cost $f[d(j, k)]$ for every grid unit in which j runs adjacent to k (in the same or adjacent planes). The function f should be chosen such that f increases with d. In particular, f will typically be small for d less than some critical value, and rise rapidly for larger d.

Alternatively, one may use the cost parameter $f[d(j, k)]$ to affect the routing of connection k, when it is next rerouted.

- *Additional comments and applications*

One can delete the inner loop (in which the cost parameter is varied to promote constraint satisfaction), and instead wait until the next iterative pass to reroute connections violating a constraint. In that case, each such connection is then rerouted, using not the standard cost parameter value for that pass, but a value that is greater or less than the standard value depending upon whether the number of occurrences to which the cost applies (e.g., length of path in grid units) was too large or too small on the previous routing of that connection. This approach can reduce the number of reroutings required. However, it can result in a form of "thrashing," in which connections exchange roles as constraint violators from one pass to the next. (This was observed in the case of global routing with maximum-length constraints.)

Constraints represented by inequalities can in general be handled by the methods described above (subject to the type of limitation discussed above for the case of minimum-length constraints). More generally, nonlinear penalty functions can be near-optimized by modifying the cost parameter (cost per penalized occurrence) so as to drive the number of occurrences in the desired direction on successive reroutings of that connection.

Further discussion is given in [10].

5. A fast global router for paths of simple shape ("LZU" paths)

In this section we discuss another type of iterative-improvement router, suitable for global and some simple detailed routing of two-pin connections, and for optimizing assignment of connections to layers of a package. The method is faster than the routing methods discussed in previous sections (no mazerouter is used here), but path complexity is correspondingly limited (only paths with two or fewer bends are considered here). Speed is important if one wishes to use the router to estimate wiring congestion for a particular pin placement, as a guide for improving the placement either manually or automatically. Furthermore, there are applications in which the great majority of wire paths have fewer than three discretionary bends. (Jogs needed to route from a pin to a wiring track are excluded from this count when they entail no significant routing decision.) The method to be described can in such cases be used for detailed routing.

- *Global routing; no layer assignment*

For global routing without respect to layer assignment, first assign to each connection an L-shaped path of random orientation on the rectangular grid. On each succeeding pass, remove and reroute each connection in turn. Rerouting

requires evaluating a penalty function for each path that joins the pins of that connection, has two or fewer bends (hence L-, Z-, or U-shaped paths), and does not exceed a specified length. The penalty function takes into account wiring congestion (including passage through blocked or otherwise specified regions) and length. An example of a simple penalty function is the sum of squares of wiring loads across all global cell edges (assuming equal capacities at all edges).

A path of lowest cost is usually selected for rerouting the connection. Alternatively, a "heat bath" option is provided, wherein the path is chosen randomly with a weighted probability proportional to $\exp(-E/T)$, where E is the cost of the path and T is a positive number (analogous to temperature). This allows moves that increase the value of the penalty function, and is discussed further in the section on "simulated annealing" below.

- *Layer assignment*

Certain interconnection packages require that each connection be routed entirely in one layer (or plane pair). Both the mazerouter methods discussed earlier in this paper and the LZU router can be used to near-optimize the layer assignment for each connection, at the same time that global (or detailed) routing is being done. We have found it useful to perform first a layer-independent route optimization (previous section), then make a random assignment to layers, then optimize the layer assignment (either fixing or simultaneously optimizing the path shape).

- *Comments on simulated annealing and related methods*

The "heat bath" option is a variant (appropriate when a move is to be selected from among several trial alternatives) of the Monte Carlo simulated annealing method of Kirkpatrick, Gelatt, and Vecchi [3]. In their wiring work, one alternative path is considered at each pass, and is accepted as the new path if it is of lower cost, or [with probability $\exp(-\Delta E/T)$] if it is of higher cost (by amount ΔE). To avoid becoming trapped in local minima (of the penalty function) that lie far above the global minimum, their method requires and provides a means for making "uphill" moves. Since we evaluate a large number of possible paths for each connection at each pass (all LZU paths up to some length), we do not require such a means; and in fact using the heat bath option does not significantly improve upon the results using the strict iterative-improvement ("downhill"-only) algorithm, in the cases studied.

The placement and wiring aspects of physical design appear to be different from one another in this respect. For the placement problem (e.g., of components on a package), there can be strong cooperativity: It may be optimal for several components to cluster, but moving any one component to its ultimately preferred location may incur a net cost; "uphill" moves must be allowed to break high-lying metastable equilibria. If primitive moves could consist of moving several components simultaneously, "uphill" moves might become unnecessary; but the number of such primitive moves would be astronomic. For the wiring problem (using simple shapes), in contrast, there is no difficulty in considering a large enough repertoire of primitive moves (e.g., 50–100) to ensure, in the cases studied, that the iterative-improvement method converges to an acceptable solution.

Can "simulated annealing," or other means for breaking metastable equilibria by allowing uphill moves, be used to route paths of greater complexity?

- If each primitive move consists of replacing one segment of path by another, a means is required to ensure that the path does not become absurdly convoluted. (There may be no primitive move converting an Ω-shaped path into a straight line, for example.)
- The following alternative combines the spirit of simulated annealing with the advantages of a mazerouter. At each iterative pass, introduce a random element (e.g., a fictitious fluctuating "blockage" term at some of the grid locations) into the penalty function; then optimize the path of each connection in turn with respect to this function. The fluctuation amplitude corresponds to an annealing "temperature." This should indeed result in breaking otherwise metastable configurations. It has not been attempted because the present methods work well and are probably more efficient. The variation of the penalty function from pass to pass in the present methods (without random fluctuations) serves both to drive the configuration toward greater ultimate legality and to reduce the likelihood of convergence to unsatisfactory (high-lying) local minima.

6. Results and discussion

We give examples of VIKING wiring of printed-circuit cards and boards, and compare wiring results with those obtained by two sequential routers. Some examples illustrate the use of VIKING for free-form wiring (i.e., wiring in directionally uncommitted planes, or extensive "wrong-way" wiring), and its application to

- Routing on via-sparse boards;
- Chips or packages with extensive (e.g., macro) blockages in some planes (a test case is discussed);
- Packages with directionally unbalanced placement (greatly different x and y wiring loads in certain regions); and
- Routing of "escapes" from under congested components.

Results on crosstalk control, reduction in the number of wiring planes required, and reduction in manual embedding effort are also presented.

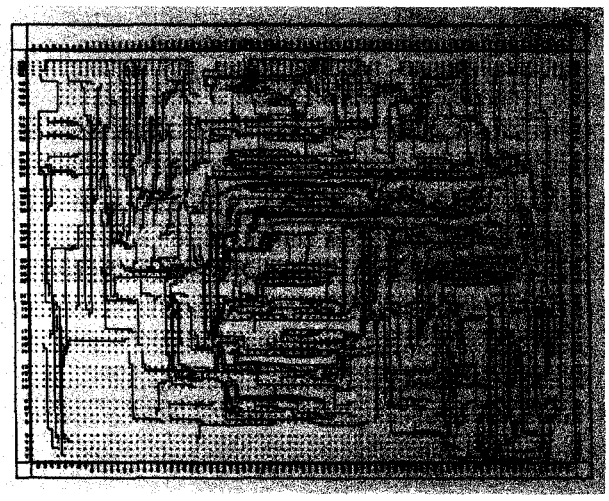

Figure 1

VIKING card routing example. Dots indicate pins and fixed vias. One of six wiring planes is shown; no directional preference was assigned to any plane.

The results of Sections B and H which follow were obtained using the original code developed by the author. The results of Sections A, C–E, and G were obtained using the production code implemented within the FSD package design system by J. F. Cooper and C. N. Lamendola. The global router is identical in both codes. The detailed router of the FSD version of VIKING incorporates (in addition to the basic algorithms) methods for limiting the number of connections requiring reroute after the first few passes, and storage handling methods valuable for large board design. It also uses a line-probe mazerouter rather than the Lee-type mazerouter of the original version; the line-probe method is used to simplify the imposition of crosstalk penalties.

For the runs of Sections A–E and H, the basic detailed router discussed in Section 2 was used. The results of Section G were obtained using the VIKING global and detailed routers, and a crosstalk heuristic differing from, but with similar effect to, that discussed in the section on crosstalk constraints.

For the VIKING results reported, the number of wires left unrouted is zero in each case. (This need not be true in general, since one can optionally have VIKING remove wires having more than a specified number of crossings, after any pass.) Accordingly, we report VIKING results in terms of the number of wires causing crossings at the end of the last pass. Results using other (sequential) routers are reported in terms of the number of remaining unrouted wires.

• *A. Reduction in number of wiring planes, vias, and manual embedding effort*

Twelve printed-circuit cards were routed both by VIKING and by a sequential router (call it "A") used within IBM. Each card contains 5–8 signal wiring planes and typically has 1000–2000 connections. Available vias are fixed and sparse.

Total wire lengths (for each card) were 10–24 percent over minimum-Manhattan length using router "A," and 5–11 percent over minimum-Manhattan length using VIKING; "excess over minimum-Manhattan" length was reduced by a factor of between 1.5 and 3.5. The number of used vias was typically reduced by about 20 percent. Manual embedding effort was reduced by a factor of two to three, and sometimes much more. Based on these and similar results, it was projected that (on average) one wiring plane could be saved per 2–3 cards, a plane savings of about 6 percent.

• *B. Example of iterative-improvement routing; "free-form" wiring*

This was the first case attempted using VIKING. It is a printed-circuit card of the type used in the above study, with 1612 connections to be routed using six planes, and with 860 fixed vias provided. Router "A" had routed this problem with 148 overflows (unrouted connections). (That router has facilities for reducing via usage by assigning connections to planes so as to allow a controlled type of "wrong-way" wiring.) Manual embedding of these overflows took three weeks. VIKING routed the problem (using a component placement that was nonoptimized, in contrast with the placement input to router "A") with 25 crossings (gridpoints occupied by two wires in the same plane).

Since crossings require only local repair (moving of wires in the vicinity of the crossing), manual embedding effort was reduced by even more than a factor of six (=148/25): from three weeks to an estimated one day. Embedding has been substantially eased by use of VIKING, because paths containing illegalities are left on the board rather than being "failed," as by a sequential router.

Figure 1 shows the VIKING routing in one of the planes, for another run (of the same problem) in which no directional preference was at any time assigned to any region of any plane. There were a total of 29 wires causing crossings.

In this example, VIKING's superior performance is due to a combination of two factors: use of iterative-improvement rather than sequential-routing methods, and reduced via requirements (yielding improved wirability in this via-sparse case) due to use of free-form or extensive "wrong-way" wiring.

• *C. Routing in directionally committed planes*

Figure 2 shows one plane of VIKING routing for a different connection list, on a card similar to that described above. In this case, each plane has been assigned an x or y directional

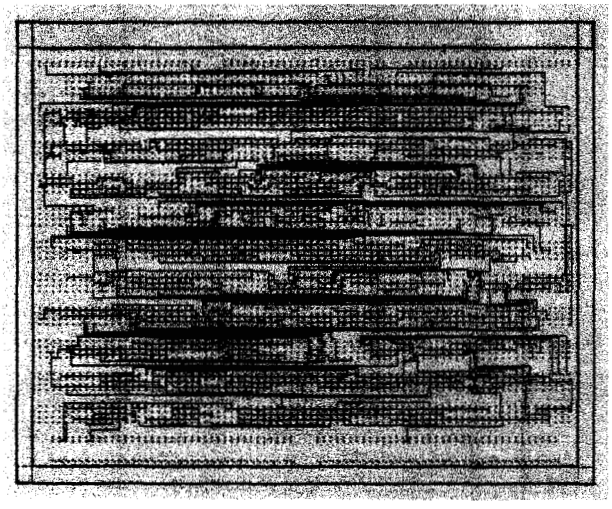

Figure 2

One x plane of a VIKING card routing. Dots indicate pins and fixed vias. No vias are available in the horizontal channels.

preference. The figure shows efficient routing within the wide horizontal channels (where vias are absent), even though VIKING uses no channel routing algorithms. An average of four wires per plane have been manually "repaired" in this case, to resolve crossings left by VIKING.

• *D. Regional imbalance of x and y wiring requirements*

Figure 3(a) shows the connections (to be routed) for a card in which the ratio of x to y wiring demand varies greatly from one region to another. If an equal number of x and y wiring planes are provided, and a capability for extensive "wrong-way" wiring is not available, wirability is impaired. One can explicitly assign regions of some x planes to carry predominantly y wiring, and vice versa, but this step is unnecessary using VIKING. Two x and two y planes were specified in an initial pass. A y domain subsequently developed within an x plane [see **Figure 3(b)**, region left of center], to supplement the capacity of the y planes in that region. This occurred as a result of the iterative-improvement process, and without any user specification of regional directional preferences.

• *E. Potential impact of wiring method on package design decisions*

A choice was to be made between two card package designs, one being less complex but providing less wirability (including less via availability). A wirability study using a sequential router "B" indicated that the less complex package would provide inadequate wirability (too many overflow wires).

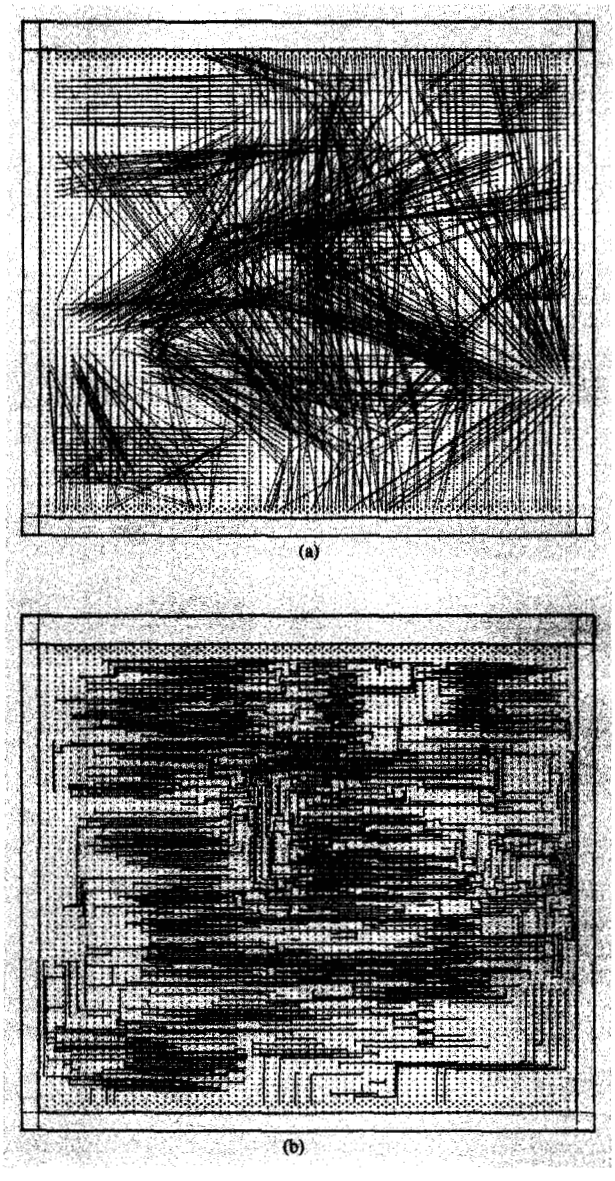

Figure 3

A card with regional imbalance between x and y wiring load (due to component placement). (a) The pin-to-pin connections. (b) VIKING routing in a predominantly x plane, showing extensive "wrong-way" wiring. (The card has two x and two y planes.)

Two cards of the less complex package type were routed using VIKING and router "B." Results were, for one card: Router "B," 62 overflows; VIKING, one crossing. For the second card: Router "B," 94 overflows; VIKING, 39 wires causing crossings (hence requiring local manual rerouting in the vicinity of the crossings).

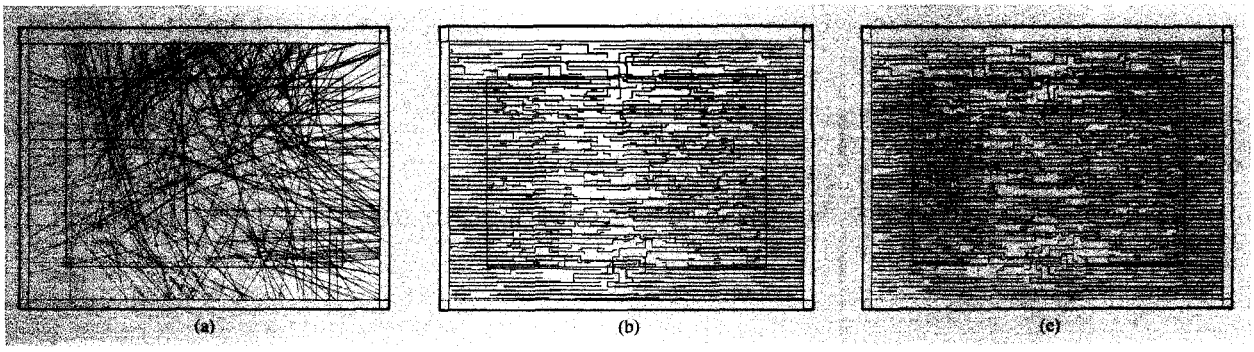

Figure 4

Portion of printed-circuit board, including a congested pin-escape region (rectangle indicates boundary of this region). (a) The pin-to-pin connections passing through this region (for one plane pair); (b) VIKING routing in one plane (diamonds indicate remaining wire crossings; this was the most congested region, and the 16 crossings shown are half of all crossings in the entire plane); (c) Routing in same plane following manual repair of crossings.

• *F. Wirability depends on the wiring program as well as the package*

The previous example illustrates that, apart from the question of whether one wiring program is "better" in a general sense than another, it is important to take into account the specific characteristics of the programs. A few of the issues are as follows:

- Does one program depend more heavily on ample via availability (e.g., because it is less amenable to extensive "wrong-way" wiring)?
- If avoidance of certain regions or wiring configurations (for which, e.g., certain types of manufacturing defects may be more likely) or avoidance of excessive crosstalk is important, is one program able to inhibit such occurrences more effectively? If the wiring program is not sensitive to such occurrences, the user may need to take a more broad-brush approach (e.g., block alternate wiring channels to reduce crosstalk; avoid certain regions of the board entirely), thereby impairing wirability.
- If there are x vs. y wiring load imbalances (as in a previous example), blockages in some planes (to be discussed), or other special situations, does one wiring program offer a specific advantage?
- In some cases, an advantage in computing speed or in ease of manual repair may be a critical determinant of practical wirability.

These and other considerations should be examined when a wiring program is used to evaluate whether a proposed package offers adequate wirability, in order to ensure that additional manufacturing complexity and cost are not incurred unnecessarily.

• *G. Reduction of number of signal planes for a large board; crosstalk and escape considerations*

A printed-circuit board containing about 8000 connections and many congested regions had been routed using a global router, followed by router "B" for detailed routing, in eight wiring planes. An early study using router "B" had found that a large number of overflows resulted when wiring in only six planes was attempted. Moreover, some of the congested regions had too many "escapes" to be routable using the wiring track capacity of three x and three y planes. "Wrong-way" wiring was not allowed in these runs using router "B."

VIKING produced a six-plane routing, incorporating adjacency controls to reduce electrical crosstalk. For a particular heuristic crosstalk criterion that was proposed, VIKING produced a routing with 25 times fewer violations in six planes than router "B" had produced using eight planes. (Router "B" does not incorporate dynamic adjacency control; instead, its user can impose certain artificial constraints in order to inhibit adjacency.) The manual effort of rerouting nets to satisfy the actual crosstalk criteria, to which the heuristic criterion was only an approximation, can accordingly be eased greatly by incorporating a well-chosen adjacency control into the routing, without the need to impair wirability by blocking wiring channels or the like.

As noted above, some of the congested regions were unroutable using three x and three y planes, without "wrong-way" wiring. By choosing the penalty functions appropriately, VIKING resolved this difficulty. One such region is shown in **Figure 4**: (a) shows the pin-to-pin connections passing through this region in one plane pair; (b) shows one plane of VIKING routing in this region before manual repair of residual crossings (shown as diamonds); (c)

shows the same plane following manual repair. In VIKING six-plane routing, the ratio of wire length to total wiring track length was approximately 45 percent. Wire length was 1.12 times minimum-Manhattan connection length.

• *H. Wiring with extensive blockages in one plane; a test case*

Consider a chip with two metal wiring planes, one of which is substantially blocked by macros. That is, much of the area in that plane is unavailable for inter-macro wiring. Clearly, if planes are assigned directional preferences, chip wirability can be severely limited by the low track capacity in the blocked (say the x) plane. In that case, one may wish to optimize device placement so as to reduce x wire load (and increase y load if necessary).

VIKING's capability for wiring in directionally uncommitted planes can provide an alternative solution. **Figure 5** shows a test case, intended to (crudely) model a macro blockage situation. A real chip can have many macro blockages in one metal plane (call it *plane 1*), some of which may have a number of small "holes" (regions lying within macros, in which plane 1 is available for wiring). For simplicity, we have considered a 100 × 100 grid with nine blockages. Each blockage is a square annulus (see dotted outlines), with one large "hole" instead of a number of small ones. Approximately 50 percent of the area of plane 1 is blocked; none of plane 2 is blocked. Sixty-five two-pin connections were chosen, with a Poisson length distribution and a mean length of 75 units.

VIKING free-form wiring is shown in Fig. 5 (solid wiring, plane 1; dashed wiring, plane 2). There were no remaining crossings, and routed length was 3 percent over minimum-Manhattan. When x and y directional assignments were made (and adhered to strictly, for a clean comparison), even a smaller problem with 40 connections could not be wired (there were eight wires causing crossings, and routed length was 13 percent over minimum-Manhattan). This is not surprising, since x wiring capacity is so greatly impaired by the directional plane assignment in this case.

When the 65-connection case was wired using VIKING with a strict rotary-wiring constraint (the rotary method is due to B. Dunham [11]), the advantage of the rotary idea over conventional x, y plane assignment was clear: the 65 connections were routed with 13 wires causing crossings, and routed length 9 percent over minimum-Manhattan.

This example indicates, in a qualitative way, that VIKING free-form wiring can "discover" unaided some of the virtue of the rotary-wiring approach (see the rough similarity of the VIKING wiring style of Fig. 5 to the rotary style, especially for the dashed-plane wiring), and can also provide some flexibility in, for example, the use of "holes" for providing wiring overpasses to improve wirability. Quantitative conclusions should not be inferred from this single example with unrealistic macros.

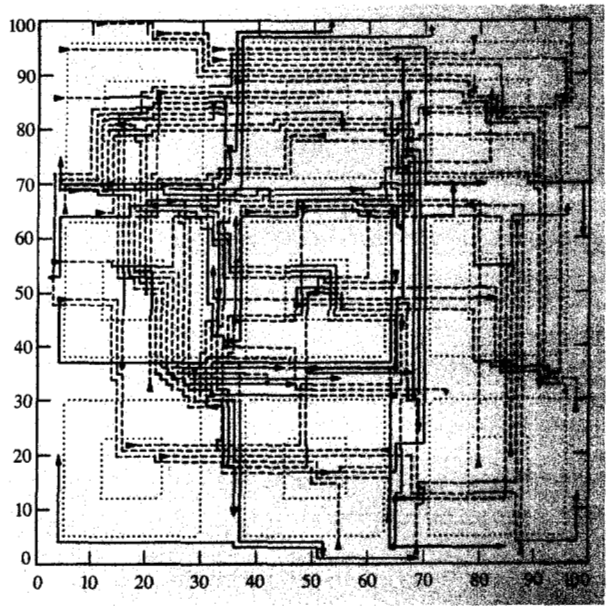

Figure 5

Test case (two wiring planes) with 50 percent blockage (dotted square annuli) in plane 1. VIKING routing of 65 connections on a 100 × 100 grid (mean connection length 75 units) is shown. Solid wiring, plane 1; dashed wiring, plane 2. No directional preference was assigned to any region of either plane. Arrowheads mark pin locations; in this run pins were not treated as blockages (i.e., other wires could pass through them without penalty).

7. Conclusions

The VIKING wiring system generates global routings, layer assignments, and detailed routings for interconnection packages. It has demonstrated advantages with respect to reduced wire length, via count, manual embedding effort, and required number of signal planes, for a variety of applications. An iterative-improvement strategy facilitates global near-optimization by allowing illegal configurations at intermediate stages, and successively routing each connection against the background of all the others. The penalty function being optimized can accommodate a wide range of design tradeoffs and desiderata, and in particular allows fine-tuning of crosstalk control, avoidance of configurations that may be undesirable from the manufacturing standpoint, and some control of minimum and maximum length. Efficient routing in directionally uncommitted planes (free-form wiring) has been demonstrated. A method for mazerouting with nonlinear or path-history-dependent penalty functions has been described. A simple version of a global router has also been presented, with some comments on methods for breaking metastable equilibria and achieving global near-optimization.

Acknowledgments

I thank J. F. Cooper and C. N. Lamendola, who implemented a production version of VIKING, and with whom I enjoyed close collaboration on subsequent VIKING development and applications work. I thank D. E. Eastman, C. D. Gelatt, Jr., S. Kirkpatrick, J. C. McGroddy, J. A. Palmieri, R. S. Rutter, J. R. Sents, D. P. Seraphim, and D. A. Thomas, for very useful discussions and help during this work. I also thank many others at East Fishkill, Endicott, Owego, Poughkeepsie, and Yorktown who have been involved with VIKING-related applications and analyses, or with more general wirability and DA concerns, and from whom I have learned much about DA and manufacturing issues.

References and notes

1. A. Moore and C. Ravitz, "Weighted and Iterative Multi-Wire Routing," *IBM Tech. Disclosure Bull.* **25,** No. 7B, 3619–3628 (1982).
2. F. Rubin, "An Iterative Technique for Printed Wire Routing," *Proceedings,* 11th Design Automation Workshop, 1974, pp. 308–313.
3. S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science* **220,** 671–680 (1983); M. P. Vecchi and S. Kirkpatrick, "Global Wiring by Simulated Annealing," *IEEE Trans. Computer-Aided Design* **CAD-2,** No. 4, 215–222 (1983); and S. Kirkpatrick, "Optimization by Simulated Annealing: Quantitative Studies," *J. Statist. Phys.* **34,** 975–986 (1984).
4. Given a grid or more general graph (consisting of nodes and links), given source and target nodes (or sets of nodes) A and B on the graph, and given a link cost function, a mazerouter algorithm calculates a path (lying on the graph) connecting A and B, such that the path cost (defined as the sum of link costs for all links on that path) is a minimum (over all possible paths connecting A and B).
5. C. Y. Lee, "An Algorithm for Path Connections and Its Applications," *IRE Trans. Electron. Computers* **EC-10,** 316–365 (1961).
6. We use the term *line probe router* to refer to a mazerouter in which the set of points being labeled with cumulative cost information at each step comprises all points along one or more rays emanating from the "current point" of the mazerouting out to some specified distance.
7. J. Hickson and W. Donath, "Connecting Three Points While Minimizing Cost," *IBM Tech. Disclosure Bull.* **25,** No. 7B, 3853–3858 (1982); and "Procedure for Connecting *N* Points With Near-Minimum Cost," *ibid.,* **25,** No. 11A, 5571–5575 (1983).
8. P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. Syst. Sci. & Cybernetics* **SSC-4,** No. 2, 100–107 (1968); F. Rubin, "The Lee Path Connection Algorithm," *IEEE Trans. Computers* **C-23,** No. 9, 907–914 (1974).
9. A different method for optimizing a penalty function subject to constraints, which involves saving multiple scores (cumulated costs) at each gridpoint, is described in R. Linsker, "Optimal Maze Routing Subject to Constraints," *IBM Tech. Disclosure Bull.* **27,** No. 2 (July 1984).
10. R. Linsker, "Wire Routing with Path History Dependent Penalty Functions," *IBM Tech. Disclosure Bull.* **27,** No. 1A, 399–406 (1984).
11. Divide each of the two planes into four sectors defined by the two diagonals of the "chip." Require that plane 2 (which is unblocked) carry *x*-running wires in the north and south sectors, and *y*-running wires in the east and west sectors; require the opposite for plane 1 (which contains the blockages). We imposed a strict rotary wiring constraint (no "wrong-way" routing) for purposes of our comparison.

Received January 9, 1984; revised April 9, 1984

Ralph Linsker *IBM Research Division, P. O. Box 218, Yorktown Heights, New York 10598.* Dr. Linsker is currently manager of system design in the Semiconductor Science and Technology Department. Since he joined IBM in 1981, his work has dealt with design automation, packaging, medical laser applications, and artificial intelligence. He received his A.B. (*summa cum laude*) and Ph.D. in physics from Columbia University, New York, the latter in 1972, and the M.D. degree from Cornell University Medical College, Ithaca, New York, in 1976. Dr. Linsker has worked in medical and physics areas at New York Hospital and Princeton University.