ROUTING TECHNIQUES FOR DEEP SUB-MICRON TECHNOLOGIES

by

Sunil P. Khatri, Amit Mehrotra, Mukul R. Prasad,
Robert K. Brayton and Alberto L. Sangiovanni-Vincentelli

Memorandum No. UCB/ERL M99/15

16 March 1999

COVER

ROUTING TECHNIQUES FOR DEEP SUB-MICRON TECHNOLOGIES

by

Sunil P. Khatri, Amit Mehrotra, Mukul R. Prasad,
Robert K. Brayton and Alberto L. Sangiovanni-Vincentelli

Memorandum No. UCB/ERL M99/15

16 March 1999

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Routing Techniques for Deep Sub-micron Technologies

Sunil P Khatri Amit Mehrotra Mukul R Prasad
Robert K Brayton Alberto L Sangiovanni-Vincentelli

16th March 1999

1 Introduction

In this paper we propose a routing methodology which is appropriate for Deep Sub-micron (DSM) technologies, and also survey possible routing techniques that would be applicable in the context of this methodology. In its most general form, the routing problem can be viewed as finding wiring connections between electrically equivalent pins, utilizing one or more layers of metal, while avoiding obstacles present possibly on every layer of metal. This general formulation (called *area routing*) has been traditionally used for routing printed circuit boards (PCBs) and multi-chip modules (MCMs). This formulation is general enough to encompass routing in integrated circuits as well. However, more specialized forms of routers have traditionally been used for ICs.

With rapid advances in VLSI processing in the DSM era, many new constraints are being imposed on the detailed routing problem. We discuss these changes and the corresponding new constraints, and review solution techniques that are likely to work in this new scenario. We argue why the traditional routing methodology of first performing a global routing step, then utilizing switchbox or channel routers, is inadequate in this new scenario. Rather, area routers, which have been useful in MCM and PCB design are more promising for ICs. Area routing techniques have been reviewed as part of several surveys on physical design [KO90, Oht86, DL89]. In these works, however, the focus of study is very general. We also show how some of the routing techniques used for analog systems are well suited to routing in the presence of the new constraints.

The rest of this paper is organized as follows. In Section 2 we discuss DSM technology trends, and in Section 3 we outline the effect of these trends on the routing problem. Section 4 describes our proposed routing methodology, while Section 5 surveys routing techniques that are applicable to our methodology. Area routing techniques from various application domains such as digital ICs, MCMs, PCBs and analog systems have been reviewed in this section, while conventional IC detailed routing techniques such as channel and switchbox routing have been omitted. Section 6 provides a short introduction to accelerated routing techniques, while Section 7 provides some concluding remarks.

2 Trends in Deep Sub-micron Technologies

Recent advances in VLSI processing technology have given cause for us to rethink many CAD algorithms. Due to aggressive advances, especially in the interconnect, many traditional CAD algorithms or methodologies need to be modified or redesigned. In this section we discuss the trends in VLSI processing that provide us with these interesting challenges and opportunities.

Modern fabrication processes usually have several layers of interconnect. In fact it is predicted that fabrication processes will have as many as nine layers of metal within the next seven years [NTR97]. As a result of this increase, it is imperative that new routing techniques be utilized.

Assume that the general process scaling factor is S, and the chip scaling factor is S_C. For a typical process generation (18 months) $S = 1.44$, and $S_C = 1.14$, based on empirical data [Bak90]. Hence the number of transistors that can be packed on a single die increases proportionate to $S^2 \cdot S_C$ with every generation. This results in increasing design complexity.

The above scaling arguments also suggest that interconnect resistance as well as via resistance grows proportionately with S. Since interconnect length (and capacitance) grows proportionately with S_C, we find that the delay for long wires on the chip increases quadratically, limiting the performance of the overall design. This is called the *long wire problem*. It constrains the router to minimize individual net lengths while performing the routing task.

New interconnect models are discussed in [Kir97], which account for changes in wire shapes and aspect ratios. It is shown that the coupling capacitance between two wires as a fraction of the total capacitance

increases as wire spacing decreases. In deep-submicron processes, this means that the signal transitions of a wire of interest are affected by those of its immediate neighbors. Hence both signal integrity and delay of the wire of interest are adversely affected. This translates to new constraints on the router.

With process scaling, [Bak90] shows that *electromigration* effects degrade as $(S_C)^2/S^2$. Electromigration is a reliability problem, which occurs due to high current densities in power supply and signal wires. It causes atoms of the wires to migrate in the direction of flow of current.

Another effect that is aggravated with scaling is *self-heat*, which is caused by the Joule heating of the wire. Although the self-heat limit is an order of magnitude less than DC electromigration limits, it scales very poorly. Self heat problems scale proportional to S.

3 Effect of DSM Technologies on Routing

With the advent of deep submicron technologies, interconnect trends are increasingly becoming critical in determining the overall functionality and performance of the design, as discussed in Section 2. In this section we detail how these trends will affect routers in the future.

Until a few generations ago, ICs typically had two layers of metal along with polysilicon, available to realize the routing. Since the functional blocks also used these metal layers to realize their internal wiring, *over-the-cell* routing was not feasible. Thus explicit routing regions needed to be defined. This, coupled with a row-based layout methodology gave rise to a natural partition of the routing space into channels and switchboxes. Therefore, specialized detailed routers, such as *channel* and *switchbox routers* worked well in this scenario. With the availability of three or four layers of interconnect the situation changed only marginally.

However, with the advancement of processing technology, it is predicted that fabrication processes will have as many as nine layers of metal within the next seven years [NTR97]. As a result, a routing methodology which can intelligently handle this extra flexibility will be required. Such a methodology should carefully budget the available routing resources on different metal layers.

Given the large number of metal layers available for routing, explicit routing regions are not required to be defined for routing. The prevalent industry practice for high end designs is to abut rows of cells and use an area router to perform the interconnections. According to [ZW97], channels are no longer accurate models for routing resources in such technologies and area routing is a robust and general approach that can be used. In these respects, the IC routing problem bears a striking similarity to the MCM and PCB routing problems, domains where area routing techniques have been very successful.

The rapidly increasing size and complexity of designs requires major modifications to CAD algorithms and data structures, and in some cases, suggests altogether new CAD approaches and methodologies. In particular, this is true for the routing problem as well.

New constraints will be placed on the routing tools of the deep-submicron era. The long wire problem will require that the routing tools ensure that individual route lengths are less than a threshold value. Also, strict constraints on crosstalk noise will require that certain critical nets are not routed too close to each other. Such constraints will need to be passed along to the routers of the future.

Electromigration and self-heating problems require that current densities in wires not exceed a threshold value. In case a route is much longer than the manhattan distance between its terminals, the increased capacitance leads to high current densities. This requires the use of a wider metal wire. Since the exact length of each route is only known during the routing phase, this constraint is best enforced by a router.

Hence, for high performance deep sub-micron designs, routers will need to respect multiple and strict constraints of delay, crosstalk noise, electromigration and thermal effects. These constraints can easily be incorporated in area routers; there are examples of existing maze type area routers which respect one or more of these constraints. For example, in analog settings, maze routers have been used to perform routing while respecting such constraints. It is predicted that analog effects will limit the performance of high end designs in deep submicron technologies. Thus, area routers would seem to be the routers of choice for these applications.

So we define the DSM routing problem as an area routing problem.

3.1 Problem Definition

The area routing problem can be defined as follows:

Given m available metal layers, and n signal nets (each of which comprises a set of electrically equivalent pins) find an interconnection between the pins of each signal net, subject to

- arbitrary sized obstacles present on each layer i.
- individual pins of each signal net present on possibly different metal layers.
- various geometric and electrical constraints

In the remainder of this paper, we survey promising software and hardware solutions to the DSM routing problem in ICs. For reasons described above, we review area routing techniques from application areas such as digital ICs, MCMs, PCBs and analog systems, while traditional detailed routing schemes such as channel and switchbox routing have been omitted.

4 Routing Methodology

Given the complexity of the routing problem and the strict performance constraints, it is difficult to envision that a flat or bottom-up approach will be very successful. A **top-down approach** seems to be more feasible. In such an approach, the design is hierarchically decomposed into smaller components. In a typical high performance design, these components can be the different macro blocks that constitute the design and the interconnect. High level performance specifications such as delay are pre-assigned to each of the components in such a way that if all the components can be designed to meet their respective specifications, the overall specification of the design is met by construction. If, on the other hand, the specification for certain components is not met, the constraints are reassigned. Such a methodology has been successfully applied for analog and mixed signal designs [CCC+97].

For the design of interconnect, i.e., routing, the constraints on performance were mapped on to the constraints on the layout geometry [CSV90] using performance sensitivities. The router was required to find the routes for each component respecting its geometric constraints. This constraint driven routing scheme was proposed as a part of the overall top-down constraint driven design approach for analog circuits [CCC+97]. This is a promising methodology but needs modifications if it is to be applied in a digital setting where the problem size is much larger than typical analog problems. For instance, given the complexity of digital circuits and the fact that factors such as crosstalk noise critically affect the performance of DSM design, the number of routing constraints generated will be extremely large. A router, even if it is hierarchical, which has to respect all these constraints will be excessively slow and in many cases may not be able to find a feasible solution even if one exists. Hence the set of constraints needs to be pruned to make the routing algorithm robust.

There are two attractive techniques which prune the cross-talk constraints. In [Kir97] the concept of **digital sensitivity** was introduced. It was observed that the set of constraints can be further pruned by observing that all signal pairs in a design do not interact because either their transitions are sufficiently staggered in time or the effect of their interactions cannot be observed at the output due to the structural characteristics of the logic that is implemented. This analysis is performed at the logic level using functional and timing analysis. The number of constraints is thus greatly reduced and the router has a better chance of finding a feasible solution.

For high performance designs, design styles other than static CMOS are usually employed for implementing high speed logic. These design styles typically sacrifice the noise immunity of static CMOS for performance gains. This further aggravates their sensitivity to crosstalk. Hence the number of constraints for the router will increase significantly. A novel way of tackling this problem is to use a **pre-characterized layout style or *fabric***. Here the cross-coupling capacitance is pre-characterized and hence can be controlled at the time of layout. One such approach was proposed in [KMB+98] where every signal line is shielded with power supply and ground lines to effectively eliminate the cross-coupling capacitance. For DSM technologies, it was shown that the delay variation due to cross-coupling capacitance was 2% as against 98% for a non-fabric layout style. This can also be viewed as effectively pruning the set of constraints, thereby enabling the router to find a feasible solution quickly. This uniform mesh of wires also results in very small power supply and ground resistance. Further since the current return path (through the supply lines) is right next to the signal lines, signal line inductance is small and predictable. This increased predictability is obtained at the expense of layout area.

5 Routing Techniques

The most general method of solving the routing problem is to find a method that routes all the nets simultaneously. These techniques have the advantage of generating very high quality routes. The approach of [SSF93] views the routing problem as a multi-commodity network flow (MCNF) problem and formulates it as an integer linear programming problem with constraints on wire capacity. Performance degradation is incorporated in the cost function instead of adding it as a constraint. A region adjacency graph is constructed for the MCNF whose vertices are the routing regions and the edges represent the adjacency relationship between routing regions. Each edge has two weights, wire capacity and Manhattan distance between the center of the routing regions. The routing step consists of finding paths for the wires connecting the corresponding vertices of the graph. Even though the technique was originally proposed for the global routing problem, it can be used for detailed routing as

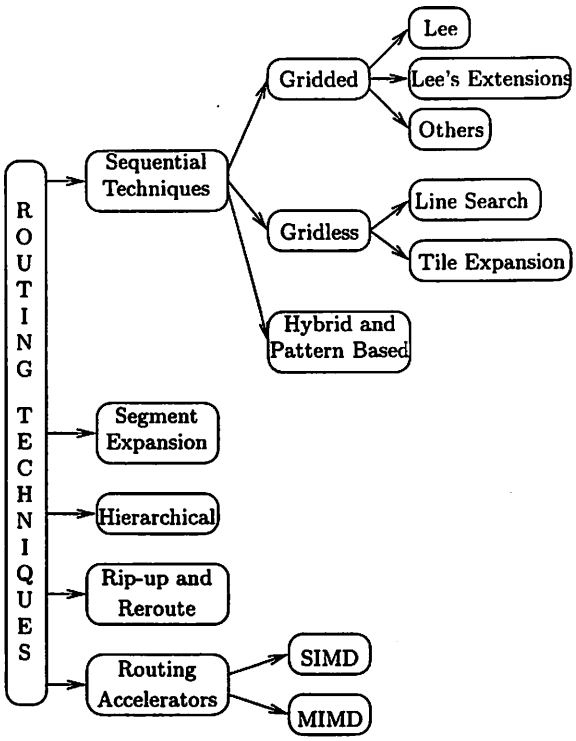


Figure 1: Routing Techniques

well.

Li and Carothers [LC96] tackle this problem by finding a small number of candidate routes for each net. They construct a *route compatibility graph* with edges representing geometric or electrical incompatibilities between different routes and determine the routes by heuristically selecting a compatible set of routes from the graph.

Nagamatu et al. [NISY95] view this as a decision problem of allocating grid points to different nets (or keeping the grid point vacant). An actual legal routing is produced by constraining this allocation to ensure:

- Continuity of the path from source to sink for each net
- No grid point is allocated to more than one net
- Each net route is a tree rather than a DAG

The problem is formulated as a nonlinear and non-convex continuous optimization problem with total wire length as the cost function. This is then numerically solved with the method of *Lagrangian Multipliers* using *Euler's method.*

Given the projected complexity of the DSM routing problem, it is difficult to see if these techniques will be feasible. There are a host of other approaches which try to cope with the complexity of the problem using some innovative techniques which we now describe.

5.1 Sequential Routing Techniques

The traditional and most common way of dealing with the complexity of the routing task is to serially route the nets one at a time, under some heuristic pre-ordering of the nets. The completion rate of these techniques would therefore appear to be sensitive to the order in which the nets are routed. However, for digital designs [Abe72] found that no ordering scheme works well for all instances of the routing problem. He concluded that routing results are statistically independent of the ordering of nets. Even for some restricted wire density range, no ordering demonstrated a substantially consistent superiority. Most of the techniques in this section address the problem of routing a single two terminal net. It is implicit that there is a suitable net ordering heuristic complementing the scheme, producing an ordered set of pin-pair routing instances.

The techniques are discussed under the three broad categories of *gridded*, *gridless* and *hybrid*[1] schemes. Gridded routers discretize the routing space into a *routing grid* and operate with a single grid point as the atomic unit of storage and computation. They are capable of operating under complicated routing scenarios, with arbitrary cost functions and can usually guarantee finding a route if one exists. However, they are limited by the size of problems they can handle. Gridless routers, on the other hand, formulate the routing problem on a continuous plane and operate on path segments rather than grid points, thereby providing a fast and memory efficient alternative to their gridded counterparts. They are, however, neither as exhaustive nor as flexible as the latter and hence provide weaker guarantees, if any, on the cost functions they can handle and the quality of solutions they can produce. The category of hybrid techniques includes those techniques which either combine aspects of both gridded and gridless schemes or are general enough to be used with both gridded and gridless frameworks.

5.1.1 Gridded Routing Techniques

Of all the grid based area routers proposed in literature, Lee's approach [Lee61] based on *maze running* is by far the most popular one. In addition to being the first work in this area, it is also the basis for a majority of the proposed techniques on gridded area routing. In the following we briefly describe Lee's algorithm and then review a number of other published techniques which are based on the general framework of Lee's al-

[1]The categorization into gridded and gridless schemes is not unique to serial routing schemes. However, it so happens that schemes reported in other categories viz. segment expansion, are primarily gridded.

gorithm. We also survey some fundamentally different formulations of the gridded area routing problem.

5.1.1.1 Lee's Algorithm Lee's algorithm operates on a *routing grid* onto which routed objects and free routing space have been mapped. There is a *cost function* defined on paths on the routing grid. The objective of the algorithm is to find a minimum cost route between two specific points on this grid which are distinguished as *source (s)* and *target (t)* points respectively. The cost function is allowed to be a vector of scalar cost functions as long as there is a total order defined on the components of the vector. Additionally, each of these scalar cost functions should be monotonically non-decreasing in the path length (i.e. a sub-path of a path cannot have cost more than that of the path). Under these constraints, Lee's algorithm can be viewed as a generalization of *Dijkstra's single-source shortest path algorithm* [Dij59] for non-negative edge weight graphs. In Lee's formulation the routing grid corresponds to the graph and the monotonic, multi-dimensional cost function represents a convenient generalization of the non-negative edge weights on the graph. A *neighborhood function* is used to implicitly represent the edges of the graph. This function gives for each grid-point the set of grid points reachable from it in a single step. Thus, in finding a source to target path Lee's algorithm essentially performs a shortest path search from s until a shortest path to t is found. This search is termed *wavefront expansion* since it can be seen as a set of "explored" grid points (also called *cells*) progressively expanding outwards from the source towards the target. The cells on the boundary of this explored set constitute the *frontier*. The path optimality of Lee's algorithm follows from the proof of optimality of Dijkstra's algorithm.

From the above description of the algorithm we see that different variants of this algorithm can be realized simply by changing the cost function and the neighborhood function. In the following we describe several such schemes that have been proposed in the literature.

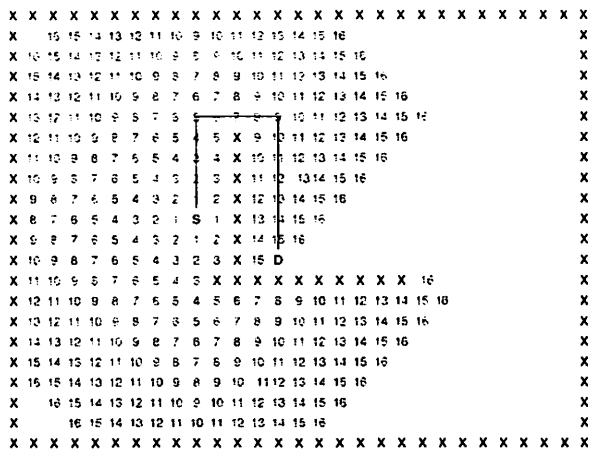


Figure 2: Lee's Algorithm (X denote obstacles) and the numbers indicate the distance of the cells from the source

5.1.1.2 Lee's Extensions

Reducing Memory Requirement: Some of the first optimizations to Lee's algorithm were introduced for reducing the memory requirement, since memory would appear to be an obvious bottleneck of any gridded approach. As proposed by Lee, cells in the frontier are assigned the same label at each step of wavefront expansion and the storage space required for the label of each cell would be $\lceil \log n \rceil$ bits where n is the maximum label number. This was reduced to two [Hoe76, Ake67] and three [Rub74b] by modifying the labeling scheme appropriately. Hightower [Hig83] demonstrated that Lee's algorithm (along with it's speedup techniques) can be used to perform two layer maze routing with almost the same memory requirement as one layer. In recent work, Soukup [Sou92] describes various methods for further reducing the memory requirement. His scheme requires storing only the cost of frontier (*active*) and vicinity (*passive*) cells. This cost is stored in binary heaps. Cells with the same cost are stored in a circular list. Since the set of active costs at an instance is small, the number of lists is small. Obstacles are stored as rectangles rather than bit maps. He also proposes an improved paging algorithm for efficient memory access.

Cost Function: There are a host of techniques which use different cost function to obtain routes with certain characteristics. In order to perform a more directed search towards the target, Rubin [Rub74b] used the A* algorithm, where the cost function is modified by adding the lower bound of the cost of reaching the target from the current position $f(c)$=cost(s,c)+LB(c,t) where s is the source, c is the current cell and t is the target cell. This modification results in expansions away from the target being penalized more heavily. This idea is also used in [MSV93] in an analog setting. Additionally their cost function takes into account localized congestion and parasitics and their sensitivities to performance.

Korn [Kor82] modified Rubin's cost function by scaling the lower bound by a factor (called *over-pull*) to further prune the search space. However, the optimality of the routes is lost as the cost function is no longer monotonic. This idea was also used by [WOYT98] along with the capability to handle variable width and spacing for

nets, as required by high speed applications. The approach described in [Het98] is an adaptation of Rubin's algorithm to *interval labeling*. Storage of and operations on the interval are much more efficient than operating on the grid. The author uses a modified form of this algorithm to compute the minimal weighted rip-up set for the subsequent rip-up and reroute phase. The global cost estimator in [AS88] takes into account the added cost of going around obstacles when computing the lower bound.

Lee [Lee61] demonstrated the ability of his scheme by defining different cost functions and thus obtaining paths with minimal crossing, minimal edge-effect (i.e. routes avoid those cells which are neighbors of obstacles) and a combination of these two properties. The algorithm proposed in [Had76] uses the detour number (number of times the path moves away from the target) as the cost function. It is shown that length of the path is the sum of the Manhattan distance and twice the detour number. A modified version of Lee's algorithm is used in [Hei69] for 2D maze routing with a cost function that minimizes via count. The cost function in [Pug78] includes the cost of vias, channel congestion and penalizes the direction orthogonal to the preferred direction on the given layer etc. The multi-layer maze routing scheme by Hightower [Hig83] also uses dynamic cost schemes to encourage or discourage nets from getting routed close to other nets or obstacles. More complicated cost functions have been successfully used in maze routers for analog applications. LIBRA [OKK92, OKK93] incorporates performance (DC, AC and transient) sensitivities to process parameter variations, stray wire resistance and capacitance and device heat generation in the cost function. Several other routers for analog applications use Lee's maze router [zBMK+93, SSA89, MSKH93, SKK+90]. The single layer maze router of [BDRV92] uses a cross between *best-first* and *breadth-first* path search.

Sequentially routed solutions are dependent on routing order, frequently a cause of incomplete routing. The method proposed in [Sup82] uses a heuristic cost function to discourage candidate routes from using grid points close to terminals of unrouted nets. Routes are determined by solving the shortest path problem on a modified graph instead of the usual Lee grid graph which is shown to be inadequate in representing the proposed cost function. The approach in [SSU85] sacrifices the optimality of the length of routes for the ease of completing subsequent routes. The routes tend to adhere to the obstacles as much as possible thereby leaving large empty spaces for subsequent nets. Here the cost function is modified to minimize the maximum of the difference of the x and y coordinates of the current vertex and the target. A line search algorithm with built-in avoidance of concave obstacle area is used in [KU85]. It also uses sub-targets to improve the quality of results.

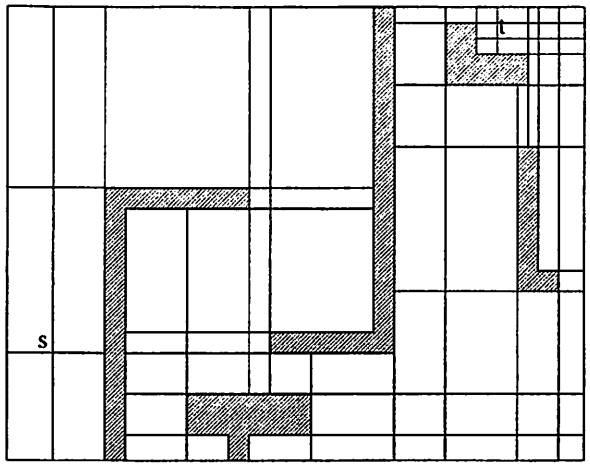


Figure 3: The connection graph (shaded regions are obstacles)

Neighbourhood Function: Some techniques modify the neighbourhood function to restrict the search space. Line search schemes[2] can be viewed as schemes which allow all the cells in the straight line from a cell to be in it's neighbourhood. Soukup's [Sou78] maze router performs a line search towards the target and uses Lee's wavefront expansion to "bubble" around an obstacle till it has a clear path towards the target. The approach in [TYKS80] defines a *macro* which is an L shape box encompassing the source(s) and target(s). At each iteration, the maze router is constrained to find a route in the macro. The size of the macro is increased after each iteration thus enabling failed nets to be routed. The sea-of-gates router in [BW90] uses the idea of preferred half plane for expansion before trying the whole grid.

Zheng et al. [ZLI93] construct a connection graph for the current routing problem and use it for propagating the wavefront. The connection graph is formed by the intersection of maximal horizontal and vertical extensions of the obstacles in the region and maximal line segments from the source and the target (Figure 3). They show that if a path exists from source to target in the original graph, then the length of the shortest path in the connection graph is the same as the length of the shortest path in the original graph. This graph can be constructed in $O(n_c \log n_c + e)$ time where n_c is the number of corner points and e is the total number of edges in the connection graph. This notion can be

[2]These can be gridded as well as gridless. Gridless schemes will be described later.

generalized to rectilinear minimum spanning trees.

Adler and Scheible [AS88] describe an interactive router where they use the concept of *magnets* to pull the routes in the preferred directions. Obstacles are expanded to avoid design rule violations. They use a shifting window to bound the wavefront expansion in a pre-specified area and move this window from source to target with time. Watanabe et al. [WKS87] propose an algorithm for finding the shortest path which also has minimum number of bends. By wavefront expansions from source and target the set of all possible shortest paths from source to destination is identified. A line expansion algorithm then finds the minimum bend path among these. Vai et al. [VAC90] propose a different labeling technique for this which requires only one wavefront expansion. The approach of [MAU86] allows multi-layer wiring and diagonal wiring. The cost assigned to each wiring layer can be controlled. The approach in [AA81] uses net ordering and Lee's maze router with a spiraling search. The neighbourhood function can also be suitably modified to handle 45^0 lines [AS98, SFH+91].

Huijbregts et al. [HEJ94] address the question of maze routing avoiding illegal wiring patterns, i.e. design rule violations. The authors show that for a layout modeled as a grid graph finding a legal path between source and destination and finding a legal path with cost less than a specified value are both NP–complete. They also present two heuristics to guide the wavefront expansion so as to find valid paths.

Multi-terminal nets: A popular approach for routing multi-terminal nets is to intelligently decompose the problem to that of routing several two terminals nets. For instance, [WOYT98] use a guided variable cost maze router where the wavefront expansion happens in the direction of the nearest terminal in case the cost of expanding into more cells is the same. On the other hand, [HJ93] start wavefront expansions from each of the k terminals. When two wavefronts meet, a minimum cost net segment is found and a new wavefront is started. They also use the information from a failed search to predict the success of the remain routes. Pugh [Pug78] uses a minimum spanning tree approach to find the optimal connection for multi-terminal nets.

Multi-layer VLSI Area Routing For modern multi-layer VLSI technologies, pitch at different layers of metal is usually different and therefore different grids need to constructed for these metal layers. The multi-layer detailed router Echelon [GW96] constructs virtual grid graphs for each of the layer. Lee's maze router is used to determine the approximate paths of each nets without violating the grid capacity. A layer assignment step determines which segment goes on which layer. This is formulated as a graph coloring problem and solved using simulated annealing and tabu search.

5.1.1.3 Other Gridded Techniques In this section we discuss some techniques for routing single nets which operate on a grid graph but do not use maze running. A novel formulation, suggested by Cheng et al. [CTY91], solves the problem of routing a net on a routing grid by solving for potentials in a resistive network. The routing grid is modeled as a *unity resistive network (URN)*, with grid points as nodes and unit resistances connecting adjacent grid points. The net source and sink nodes are connected to the highest and lowest potentials respectively. Obstacles are modeled by missing resistances, or by setting the obstacles' potentials to the highest value in the URN. A route is constructed by following the local maximum current direction at each successive node, starting from the source node. Buses and multi-pin nets are also handled.

Another gridded technique which uses an iterative framework tightly coupled with re-routing is the "supply-demand" router proposed by Rosenberg [Ros87]. It uses a Lagrangian relaxation based discrete optimization formulation for routing each net. A cost is computed for each grid point, based on its "demand" (i.e. the number of nets routed through the point) in the previous iteration. In each iteration, the current set of unrouted nets are routed sequentially (allowing routing violations) under their minimum cost solutions. The costs of each grid point is updated based on this routing. All non-conflict routes are frozen in position and the remaining ripped up for potential re-routing in the following iterations.

The first technique incorporating cross-talk constraints in an area routing framework is reported in [ZW97]. Here, the objective is to find the shortest route for a single two-terminal net on a multi-layer grid graph, subject to obstacle constraints and crosstalk constraints from previously routed nets. The problem is formulated as a *Multi-Constrained Shortest Path (MCSP)* problem, under a simple crosstalk model and shown to be NP-Complete. A *convex programming* based approximation algorithm is proposed for *MCSP* which is solved by running Dijkstra's shortest path algorithm inside a Lagrangian multiplier relaxation loop. The authors also suggest modifications to handle multi-terminal nets and a more sophisticated crosstalk model.

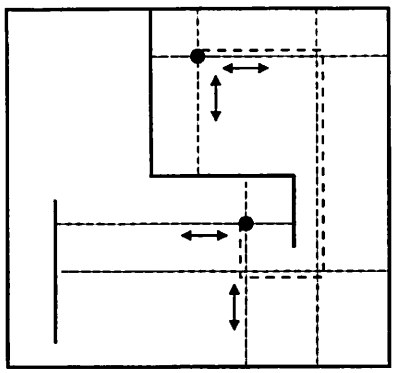


Figure 4: Principle of Hightower's algorithm [Hig69]

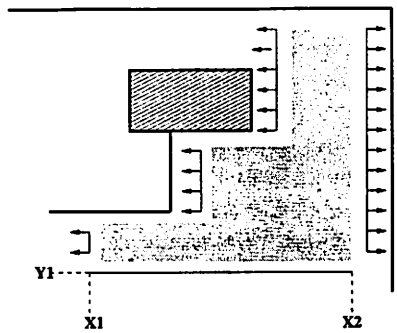


Figure 5: Heyn's modification to Hightower's algorithm. The arrows indicate the direction in which the active lines are being expanded [HSB80]

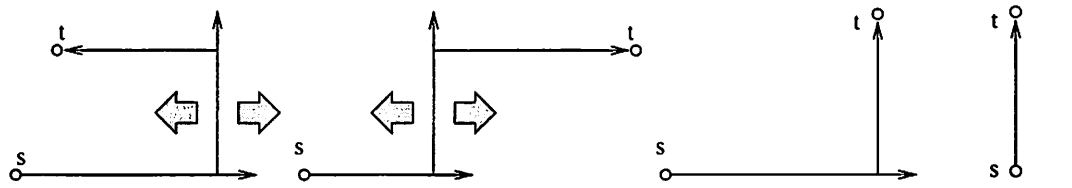
5.1.2 Gridless Routing Techniques

5.1.2.1 Line Search and its Extensions The earliest approaches proposed in this class, are based on a technique called *line search* and were proposed independently by Mikami-Tabuchi [MT68] and Hightower [Hig69] respectively. Since the two are very similar in spirit only the latter is reported here.

Hightower [Hig69] addresses the problem of routing a two terminal net through a two dimensional maze of obstructions. The essential idea of the proposed solution is to alternately grow one segment at a time from the two terminals, till the two growing sub-routes meet. At each *expansion step* the current segment is extended till it hits obstructions. Then, using two expansion techniques, an *escape point* is found on the current segment, from which a perpendicular segment can be drawn to escape the immediate set of obstructions and hence continue the search. Figure 4 illustrates this scheme. Since this approach makes a deterministic choice of the expansion segment at each step without considering or recording all feasible choices, it cannot guarantee finding a route if one exists and even when it does find one it may be sub-optimal. Hence, a post-processing step is added to locally optimize the constructed route.

The *Line Expansion* procedure proposed by Heyns et al. [HSB80] incorporates the idea of *wavefront propagation* of Lee's algorithm into Hightower's gridless routing framework (Figure 5). At each expansion step, instead of generating a single escape line, the approach generates the borders of the zone (the *expansion zone*) that can be reached by the set of all possible escape lines drawn perpendicular to the current escape line (called the *activity line*). The *expansion zone* is obtained by expanding each grid point of the activity line. It is stored in terms of its bounding segments, which are used as activity lines for continuing the search. The method always finds a path if one exists but can be computationally expensive, owing to the grid-point based generation of the expansion zone and also because the number of activity lines can grow very large. An attempt at partially alleviating this shortcoming is the *zone expansion* algorithm proposed by Johns and Hachtel [JH86]. Here, the escape line is not expanded grid-point by grid-point, rather the expansion zone borders are created by an examination of the obstacle boundaries. A similar idea was used in the recursive pattern router (RPS) proposed in [SFH+91] where the algorithm recursively applies search patterns (instead of straight lines) on each of the current search lines till connection is made. "Self-looping" is avoided by checking if any of the generated search lines overlaps with any part of the route from the start pin. The patterns are U, Z, L and I shaped (Figure 6). Most of the above works employ a *k-dimensional binary tree* data structure in their implementation. The first use of this data structure, in a framework for gridless routing, is credited to Lauther [Lau80].

In order to obtain a version of Lee's algorithm which does not require the presence of a grid, Xiong and Kozawa [XK81] make the observation that the wavefront expansion can be emulated if the router keeps track of the irregularities (or *diffraction points*) in the layout. These irregularities can be the beginning of an obstacle (which causes wavefront propagation to stop) or the end of an obstacle (which causes the wavefront to propagate in an additional direction). The algorithm defines four quadrants (four different zones around the source with different propagation rules). Once the wavefront encounters an irregularity, it is diffracted around that point. The first wave that reaches the target is the shortest path. The algorithm needs to store only the diffraction boundaries, the obstacles and the current wavefront. Since the waves travel only in one direction in a quadrant, tracing back requires only the knowledge of the diffraction boundaries. Later [Xio86] extends this method to determine the diffraction points



(a) U Shape (b) Z Shape (c) L Shape (d) I Shape

Figure 6: Pattern Search

and boundaries directly from the geometry of the obstacles without explicitly propagating the wavefront. The auxiliary lines required to determine the diffraction boundaries run along the obstacles and are at a minimum distance from them. Note that this method does not require the presence of a grid. However, this approach only finds the shortest path from source to destination and cannot easily incorporate arbitrary cost functions.

Implementation aspects of some of the above algorithms are discussed in [CI77] while [FMBS85] describe extensions to handle multi-terminal nets and multiple layers of interconnect.

5.1.2.2 Tile Expansion Based Techniques Tile expansion routers use the corner stitching data structure [Ous84] to represent the routing surface. In this data structure, rectangular areas or *tiles*, which represent empty as well as occupied space, are stitched together at their corners like a patchwork quilt. Based on the corner stitching data structure, [Ous84] gave linear or constant time algorithms for common layout tasks like searching, creation and deletion of tiles.

The earliest reported router based on the tile expansion algorithm is [MRG+87]. It finds a solution with a minimum number of jogs, if it exists, but it assumes a 2 layer routing surface. This scheme represents all mask layers on a single tile plane. Consider a route being constructed between source pin s and target pin t. First, a *working tree* is created. The space tile adjacent to the source pin in the horizontal (vertical) direction is the root tile of the working tree. Space tiles adjacent to it in the horizontal or vertical directions are first level children. For each of these tiles in turn, the space tiles adjacent to them in the horizontal or vertical direction are second level children and so on. The working tree is traversed in a breadth first fashion until the target pin is reached. This scheme can be used for incremental routing as well. This type of router is very similar to a line expansion router. The disadvantage of this scheme is that routing and contact layers are all combined on to single tile plane, which is very fractured.

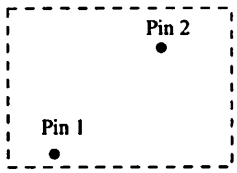
In [TCF92], the work of [MRG+87] is extended to multiple layers. In this work, a H-V alternating router is described. The router performs routes by a sequence of alternating H and V tile expansions, with an A* approach to guide the expansion of the search tree. The expansion has an additional factor in the cost function, which can give rise to minimum manhattan distance paths, or minimum bend paths. Just like [MRG+87], it is guaranteed to find a solution if it exists. Unlike [MRG+87], it uses one tile plane per routing layer. Multi-terminal nets are routed by a *net-forest* technique which expands paths from each pin in parallel towards their center of mass.

Tile expansion routers are conceptually similar to zone expansion routers. Even after the route path is chosen, they provide the some flexibility in choosing wire widths and spacings. The final choice of the route can lead to greater fracturing of the routing space, and must therefore be carefully made.

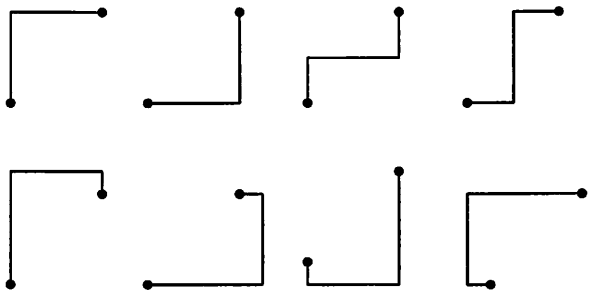
5.1.3 Hybrid Techniques

A hybrid approach proposed by Cohoon and Richards [CR88] uses a combination of gridless line search and grid-based wave expansion for routing a single two-terminal net. Similar to Lee's algorithm, the actual route is found through a shortest path search on a routing graph. However, the "grid graph" conventionally used as the routing graph is replaced by an *intersection graph* formed by the intersection of a set of escape segments. The set of escape segments is generated using a *line sweep technique* borrowed from the field of computational geometry. The modified grid thus constructed is proved to be sufficient to capture at least one *optimal* path. The optimality criterion is a weighted sum of the wire length and the number of bends. The results are extended to multi-layer routing.

5.1.3.1 Pattern Based Routing The general idea behind pattern based routing is to store a small set of pre-defined route shapes (patterns), potentially having parameterizable elements, and then try to create a



(a) Example of a pin-pair



(b) Route configurations allowing two vias

Figure 7: Parametric Pattern Routing [Asa82]

route for a given net by fitting one of the pre-stored patterns to it. Although the schemes in this category have been proposed in a gridded framework they are equally applicable in a gridless setting.

The earliest use of this idea is credited to Pugh [Pug78]. Here, a simple pre-processing step is used to route simple connections first (before invoking the main routing engine) using predefined, non-conflicting path shapes, without actually searching for a path. The connections chosen are those which are likely to create the minimum degree of conflict. The idea of pattern routing was extended by Soukup and Fournier [SF79] into a complete router. They used a small set of patterns with the restriction that each pattern was allowed to have at most two "flexible" (extensible) segments. They observed that pattern routing is very effective for routing regular structures (e.g. memory busses) especially during the early stages of design.

The pattern router reported in [Asa82] constructs its pattern set by enumerating the possible route configurations, allowing a certain maximum number of bends or vias. For example the pin-pair of Figure 7(a) can have at most eight routing configurations allowing at most two vias, as shown in Figure 7(b). The length of each of the segments is parameterized. To find routes the router sequentially processes all pin pairs. For each pin pair, it chooses a pattern serially from a ordered set of patterns. For each pattern it generates all possible routes by varying parameters systematically, and testing feasibility.

Pattern based routers, while incapable of finding connections in a densely populated maze of obstacles, can route *easy* connections really fast, with a judicious choice of the pattern set. Since such easy connections might form an appreciable fraction of the total connections to be routed, we see pattern based routing as a useful front-end step to the core routing scheme but not a complete routing methodology in itself.

5.2 Segment Expansion Techniques

These techniques are motivated by the observation that highest quality routing solutions are obtained when all nets are simultaneously routed. This is accomplished by routing small segments of the nets using a possibly sequential routing algorithm and growing these routed segments in each iteration. One of the earliest techniques along these lines is an algorithm due to Vintr. Vintr's algorithm, as reported in [Sou81], uses a variant of Lee's algorithm to find routes, but once a path is found only a specified fraction (*cutoff fraction* or *tail*) of the net is routed from the source and the target. These tails are not allowed to cross over. At every iteration the tail size is increased and the tails are ripped up and rerouted. This basic algorithm is modified in [ME95] to penalize potential cross overs. One version runs this algorithm up to a certain value of the cutoff fraction and then switches over to the A^* algorithm. A similar idea is used in [Lun88] where a scheduler applies a series of routing strategies to a queue of partially routed nets. Each strategy attempts to increase the routed fraction of the net. Strategies include coarse routing, region routing (straight line wires), rip-up routing, jog routing, push routing, maze route (in congested areas). Initialization is done by running coarse routing step, and ordering nets based on a simple net complexity measure. As a post-processing step point to point maze routing is performed on complex problems, and some routes are simplified if possible. A corner stitched data structure is used to represent obstacles and routes.

The approach described in [DRR90] improves upon Vintr's algorithm in the following way. It assigns different costs to crossing or touching the routed segments of the nets based on the observation that crossing violations are difficult to fix compared to touching violations. They also observe that if the completion of all the nets is scheduled in a single iteration, it is virtually impossible to complete that iteration. To avoid this the completion of the routes is staggered. The nets are prioritized into buckets. At iteration i, $100 \times \frac{i}{j}\%$ $(i \leq j)$ of the nets in the j^{th} bucket get routed.

The techniques described above expand segments

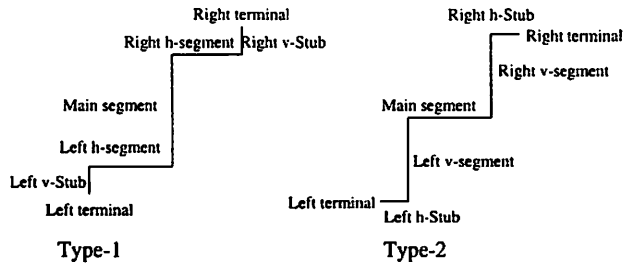


Figure 8: Two "orthogonal" four-via routing topologies [KC95]

from all the source and target pins that are to be interconnected. There are techniques which select a set of nets and perform segment expansion on just that set. These techniques described here were originally proposed for MCM routing but can be readily used for IC detailed routing. V4R [KC95] constructs routes with a maximum of four vias. A four via net can have two possible configurations as shown in Figure 8. The algorithm processes two layers at a time and processes each layer-pair column by column. For each column c the algorithm attempts to

- connect the right terminal of every net, whose left terminal is in c, to an appropriate horizontal track which is free between c and the column containing this right terminal (make *right v-stub*). Every net for which this is possible is labeled type 1 and the remaining ones are labeled type 2 (Figure 8).
- connect each type 1 left terminal in c to an appropriate horizontal track (make *left v-stub*). Then try to assign the main segment of type 2 nets whose left terminal is in c.
- select the maximum subset of the unrouted v-segments (main segments of type 1 nets, left and right v segments of type 2 nets) and route them in c
- extend the remaining left h segments of the unrouted nets to the next column

In each of the above steps, if any section of the net is blocked, the net is ripped up entirely. Once all the columns have been processed, the unrouted nets are then processed in the next layer-pair.

In an earlier work [KC92], a technique was presented where only one layer is processed at a time and a planar solution is generated for each layer. For all column pairs, which have terminals, a maximum weighted non-crossing matching is generated and used for routing. A two layer maze router is called for routing the failed nets. The unrouted terminals are propagated to the next layer for processing.

5.3 Hierarchical Techniques

Hierarchical routing techniques attempt to alleviate the complexity of the routing task by viewing the routing area at different levels of detail. Routes created at the coarser levels are passed down and refined in successive finer levels to yield the final routes at the lowest (finest) level of the hierarchy.

The simplest kind of hierarchical techniques have two levels of hierarchy; a global routing phase to assign an approximate topology to the routes and a detailed routing phase where each global route is turned into a legal physical route by *examining the area of the global route* in full detail. In [SLS89] Lee's algorithm is used in the global routing stage using track capacity to guide the wavefront expansion. In the initial phase of the detailed routing, track reservation is used to restrict the search space. As the nets become difficult to route in the current configuration, the router frees up reserved tracks starting from the least critical nets. Soukup and Royle [SR81] and Pugh [Pug78] use maze running for both global and detailed phases of routing. *Codar* [TS88] uses a similar approach albeit with a different method of performing the global and detailed routing. In the global phase, multiple routes are allowed to occupy the same physical space; these are legalized in the detailed routing phase. A *congestion map* is maintained for the grid (multiply occupied grid-points are given a higher congestion value). Global routes are constructed by performing maze-running on a *coarse grid*. The coarse grid is constructed by selecting a few least congested tracks (in both horizontal and vertical directions) from the routing grid, in a heuristic manner. The detailed routing is accomplished by locally refining the routes through sequences of wire-move operations. This is discussed in further detail in Section 5.4.1. A similar approach is used by the Window Pattern Router described in [TFS84].

Another popular approach to two-level hierarchical routing is to perform global routing and then *partition the routing area* into routing regions to be subsequently detail routed. These detail routed regions are then combined together to yield the complete routing solution. This kind of methodology has been traditionally used in conjunction with channel or switchbox routing for the detailed routing phase. As discussed in Section 3 channel and switchbox routing seem to be less promising techniques for DSM routing. However, the *divide and conquer* methodology in which they are used still seems to be a promising framework for routing. Two schemes which employ this framework are [FW96] and [LTS98]. In [FW96] the routing area is hierarchically decomposed into small regions and nets which lie *entirely* in a region are routed using the V4R [KC95] scheme (described earlier in Section 5.2). In the next

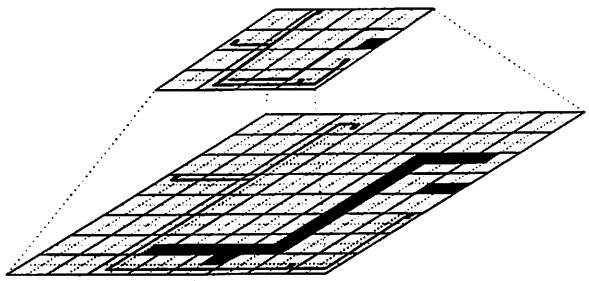


Figure 9: Mapping blockages and preferred routing region down to next lower level [LHT90]. The dark shading represents projected routing blockages and the light shading shows the projected preferred routing region.

iteration, the routing sub-regions are merged and unrouted nets are routed using V4R. In the approach of [LTS98] the chip area is divided into small routing regions by extending the boundaries of blockage areas. A graph depicting this topology is constructed and global routing performed on it using a steiner tree heuristic. In the second phase, the area is partitioned into routing regions (*gboxes*) for detailed routing. Each gbox is a generalized switchbox, with pins and obstructions at arbitrary locations. Detailed routing is performed using a tile-expansion approach based on the A^* algorithm, supported by a rip-up and reroute scheme.

Burstein and Pelavin [BP83] propose a top-down hierarchical scheme which starts with wiring a trivial 2×1 grid at the topmost level, then a 2×2 grid, a 4×2 grid and so on. At each level of hierarchy the routing is accomplished by partitioning the problem into a set of $2 \times N$ grid routing problems. The authors propose two solutions to the problem of routing a set of nets in a $2 \times N$ grid; one based on a linear programming formulation and the other based on a min-cost steiner tree construction. Another approach, proposed in [LHT90] performs an initial bottom-up pass followed by a top-down phase and uses *wave-expansion* based on Lee's algorithm (see Figure 9). A quad tree data structure, linking together a hierarchy of grids of increasing level of coarseness, is used to represent the routing space. The algorithm works in three phases :

1. Starting from the lowest level, perform k steps of wavefront expansion from each of the terminals to make a partial connection

2. Project up to the next higher level and repeat step 1 till the connection is completed

3. Project down the connection path to a region at the lower level. Complete detailed route at the lower level, as far as possible within the projected region and pass this onto the next lower level.

5.4 Re-Routing Techniques

The approaches described in Sections 5.1, 5.3 and 5.2 attempt to cope with the complexity of the routing problem by partitioning it into a set of smaller problems. Owing to the sub-optimality inherent in such a partitioning these techniques often lead to routing violations, incomplete routing and globally sub-optimal routes. Hence some form of re-routing is invariably employed as a back-end to the core routing engine, frequently integrated with the actual router through an iterative refinement scheme.

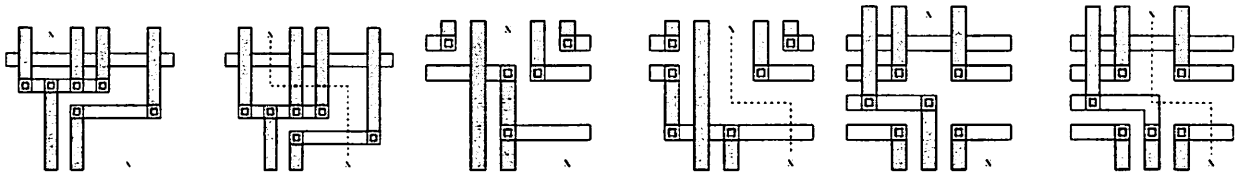
A variety of re-routing techniques have been proposed in literature. These approaches can be broadly differentiated along two axes. Firstly, the techniques can operate under either one of the following two models of routing:

- **Allow routing violations:** These schemes allow intermediate configurations with *routing violations* (i.e. two or more connections routed through the same physical space). They initially route all nets, potentially creating some violations, and then use an iterative rip-up and reroute scheme to successively reduce violations.

- **Allow only legal configurations :** These methods iterate the process of "legally" routing as many connections as possible followed by ripping-up or re-adjusting some of the routed connections in an effort to route the failed nets, till all connections are routed or all pending connections are recognized as unroutable. In such a framework we can define a *blocking situation*, i.e. a situation where the sequential routing of previous nets has rendered the current net unroutable.

Secondly, the schemes differ on aspects of the rip-up and reroute procedure itself, *viz.* (a) the rip-up cost function (b) the algorithm for choosing connections to be rerouted and (c) the method of re-routing the chosen nets. In the following we review the major developments in re-routing, in the light of these three aspects.

In one of the earlier works addressing rip-up and reroute, Rubin [Rub74a] proposed an iterative routing framework which allowed routing violations but kept track of the violations for each net through a scoring function. In each iteration, nets with the highest penalty score were ripped up and rerouted. Later, Linsker [Lin84] extended this approach using *penalty functions*. A similar approach has been suggested in [SCA94]. A more elaborate heuristic measure to select the rip-up candidate has been proposed in [CS89]. Here, a combination of re-routability potential of the net and a measure of the number of grid points of the net reachable from the blocked net is used to select the



(a) unitpush (down) (b) jump-push (down) (c) pointpush (down)

Figure 10: Examples of weak modification [SSV87]

most eligible candidate for rip-up. In [AB77] Agrawal and Breuer suggested a backtracking algorithm which can possibly enumerate all alternate connections for each net.

No single re-routing scheme works best for all situations. Some initial work aimed at modeling and quantifying the efficacy of different re-routing schemes has been done by Dees and Smith [DS81]. The authors use statistical modeling approaches to develop formulas for calculation of *expected completion rate* and *computational effort* for different rip-up and reroute mechanisms.

5.4.1 Shove and Route

A number of works MIGHTY [SSV87], m3D [WLS93], CRACKER [GH89], CODAR [TS88] and [DK82] propose a *shove-aside* or *weak modification* strategy as an alternative to the rip-up version of re-routing. Here, blocking nets (i.e. nets producing routing violations) are locally perturbed or rerouted to alleviate the violation. In [SSV87] the notions of *weak modification* and *strong modification* are introduced to correspond to *shove-aside* and *rip-up and reroute* respectively. A weak modification implies a local pushing of blockages in any one of four directions (E,W,N,S) using one of three local moves, namely *unit push, jump push and point push* (shown in Figure 10). To prevent an oscillatory condition, a history of weak moves is maintained. The same weak move is not recalled unless one of the associated nets has been rerouted or at least one of the blocked connections made. For the *strong modification* phase, a dynamically varying *difficulty index* is maintained for each net (indicative of the re-routing ease of the net). A minimum rip-up cost path is computed for the blocked net, where the rip-up cost function is based on the number of nets ripped up, their associated difficulty indices, length of the path and the number of vias. All blockages on this path are then ripped-up, the blocked net routed, and the ripped-up connections scheduled for routing.

This approach is extended in Codar [TS88]. Here the move set contains a general *wire-pushing operation* that moves a wire segment laterally and a number of more complex primitives such as *layer switching* for moving wire segments in the Z-direction, *jog insertion* for moving only a part of a wire segment and *small-area maze routing* for finding a legal path with many jogs in a restricted area. A recursive search is conducted to find sequences of moves that will reduce the number of routing violations. At each level of the search a violating segment is chosen and a number of local moves tried on it. If this legalizes the violation, the search backtracks; if it creates more violations the search proceeds recursively to those violations.

CRACKER [GH89] performs the initial routing by a quasi-minimal rectilinear Steiner tree heuristic (disregarding all conflicts with other nets). The violations are removed by performing *connectivity preserving transformations* (analogous to the concept of *weak modifications* defined above). Three types of transformations are employed, namely (i) layer change, (ii) segment split (create a node connecting the two split portions) (iii) segment move by one grid point.

5.4.2 Probabilistic methods

The routers presented in [MBE94](PROCORE) and [LHT89] (SILK) propose solving the rip-up and reroute problem in an iterative, probabilistic setting. The idea is to initially route all nets, allowing routing violations, and then probabilistically allow the system to evolve to a configuration without violations. In the tool SILK [LHT89], this is accomplished by a *simulated evolution* based approach. Each net is assigned a cost based on the number of violations, number of vias and net length. It has a *probability of rip-up* proportional to its cost. In each iteration of the "evolution", a set of nets are probabilistically ripped up and rerouted. PROCORE [MBE94] uses a *simulated annealing* framework which is similar to SILK. The cost function here is the number of routing violations. An annealing transformation consists of selecting an intersecting net at

random and trying to route it without obstructions. If this is possible the move is accepted. Otherwise the net is routed, allowing intersections with new nets (not the old ones) and the move accepted with probability $\exp(\Delta/T)$ where Δ is the difference in cost function and T the annealing temperature. In both these approaches, a version of Lee's algorithm is used for the actual routing.

In [SST96] Shirota et al. argue that routing violations can be attributed to :

- Dependence of the solution on the routing order of nets
- Local wire congestion due to concentration of routes in certain regions of the routing space.

They propose a two-level hierarchical framework for rip-up and reroute which permits routing violations. To correct violations caused by routing order, the SILK methodology (described above) is used to perform a local rip-up and reroute. A global rip-up and reroute phase is used to alleviate routing congestion based conflicts. This phase employs a new measure of rip-up cost for the nets, which indirectly reflects the *routing violation congestion* in each region. The re-routing is accomplished by means of a maze router operating on a *fragmentary coarse grid map* (i.e. a coarser grid map where regions have been mapped onto grid points), obtained through a congestion analysis of the routing grid.

5.4.3 Computing a good Rip-up Set

One of the earliest works addressing this aspect is due to Suzuki et al. [SMTO86]. Given a *blocking situation*, the authors define the notion of a *minimal separator*. This is a minimal set of occupied grid-points, at least one of which is used in every potential route of the blocked net. A minimal separator can be easily computed from a modified wavefront propagation phase of Lee's algorithm. This can subsequently be used to decide if ripping-up a given set of nets would be sufficient to route the blocked net.

In more recent work, Raith et al. [RB91], propose a novel solution to the problem of determining an "optimal" set of connections to be ripped up in order to alleviate a *blocking situation*. The optimality of the rip-up set is measured in terms of its minimality and on a heuristic measure of the re-routability potential of the ripped-up nets. The current routing configuration is represented by means of a hypergraph. The nodes represent connected regions on the routing grid and hyperedges are routing connections, the removal of which would connect the associated regions. The source and target net points are also nodes in this graph. Simple source-target paths in this graph characterize "minimal" rip-up sets. Re-routability potential of a net is based on the density of connections in its vicinity. This is ascertained by a scan of each net's surrounding rectangle. A "best" rip-up set is calculated on this basis.

Rosenberg [Ros87] works with an iterative framework allowing routing violations. Given a routing configuration where each grid point may be occupied by several nets and each net is assigned a rip-up cost based on its violations, the objective is to choose a *minimum cost set of nets* to be ripped up in order to remove all violations. This problem is formulated as an integer programming problem and solved using a set-covering heuristic.

6 Accelerated Area Routing

Acceleration of area routing can either be achieved using specialized hardware, or by mapping the routing algorithm on to general purpose parallel computers.

In general, schemes which require the design of specialized routing hardware, like hardware accelerators or systolic acceleration schemes, have limited appeal. These hardware routing systems have a fairly long design time, and given the rapid pace at which faster microprocessors are becoming available, a hardware accelerator is often slower than the fastest available software implementations. As a result, we do not cover such schemes.

Lee's maze routing algorithm offers the most opportunities for acceleration, given its inherent parallelism and regularity. Acceleration schemes for other routing techniques have not been proposed.

Routing acceleration techniques are classified into two broad categories.

6.1 SIMD Routing Accelerators

Single Instruction Multiple Data-stream (SIMD) routing accelerators capture the intrinsic parallelism of the maze routing algorithm, resulting in potentially linear runtime for finding a path. In such schemes, each Processing Element (PE) handles a subset of the grid cells.

Some of these techniques proposed in this section utilize massively parallel SIMD machines, utilizing lightweight PEs. Others require one PE per grid point. Yet others utilize a coarse-grained notion of parallelism, with less than one PE per grid point. However, since the first two configurations use PEs with minimal functionality, it is difficult to incorporate complex cost functions like those which would be encountered in a DSM setting. As a result, we believe that MIMD architectures, or SIMD architectures with coarser grained

parallelism will be favored over these massively parallel SIMD architectures.

SIMD schemes are further classified based on their interconnection network configuration.

Mesh Connected Machines. In most of these schemes, each PE communicates with its four neighbors. Some examples are the Wire Routing Machine [NHLV82], the DAP (Distributed Array Processor) [Ads82], MAPLE [SKS+90], the Iterative State Machine Array (ISMA) [RR87], PRIAP [IV85], and [Mik94] which handles 3-D maze routing. In general this technique to handle 3-D routing can be used by other hardware maze routers as well.

Toroidal Machines. The accelerated router in [SMTO86] has a twisted torus interconnection network. A high utilization of PEs is obtained by interconnecting PEs in a twisted torus arrangement. In [KSS+90] a accelerated maze router is described which is implemented in [SKS+90] and [KFM95] on toroidal SIMD machines.

Hexagonal Array Machines In [VM90] and [VM93], a hexagonal mesh multi-computer with wrap-around connections is described. For a grid of size G^2 and k layers, this configuration has a smaller ($3kG$) number of processors than a full grid machine (kG^2).

6.2 MIMD Routing Accelerators

Multiple Instruction Multiple Data-stream (MIMD) techniques exploit parallelism which is typically coarser in nature than SIMD parallelism.

The earliest MIMD maze routing accelerator is the Wire Routing Machine ([HNS81]), an 8×8 rectangular array of mesh connected processors. In [WS88], [KAE91] and [MP91], maze routing is performed on a machine with a binary hypercube topology, with message passing as the form of inter-processor communication. Strategies for mapping the maze routing algorithm onto hypercube architectures are discussed in [YDB93]. Another important issue for MIMD routing algorithms is *synchronization*, which is detailed in [WS88].

Parallel maze routing on a network of workstations (NOW) is discussed in [SSH+93]. This maze router has a variable cost function, and handles multiple layers. Routing violations are eliminated in a subsequent rip-up and reroute step. Almost linear speedups are obtained with an increasing number of workstations.

A hierarchical routing algorithm is found in [SJA94], which is different in that it partitions the problem hierarchically using a scheme like [BP83] but finally solves each problem using maze routing. The formulation allows parallelization of sub-problems, which are processed on a network of workstations.

We believe that the parallel methods holding most promise are those which map the routing problem on a network of workstations. These methods, which are relatively simple to implement, show impressive speedups, and require no special hardware.

7 Conclusion

We introduced a new routing methodology for DSM technologies, and also reviewed possible techniques for performing routing within this methodology. We contend that area routing would be the most prominent routing technique since it can handle the unique routing constraints in VLSI circuits of the future. We reviewed techniques which have been used in VLSI routing, as well as others from MCM and PCB routing which are general enough to be extended to VLSI routing. Since these techniques were originally proposed in widely different contexts, we have not made any attempt to compare their specific results but have concentrated on the algorithmic and methodological aspects instead.

References

[AA81] S Aranoff and Y Abulaffio. Routing of printed circuit boards. In *Proceedings ACM IEEE Eighteenth Design Automation Conference*, pages 130–6, 1981.

[AB77] P Agrawal and M A Breuer. Some theoretical aspects of algorithmic routing. In *Proceedings of the 14th Design Automation Conference*, 1977.

[Abe72] L C Abel. On the ordering of connections for automatic wire routing. *IEEE Transactions on Computers*, G-21(11):1227–33, November 1972.

[Ads82] H G Adshead. Towards VLSI complexity: The algorithm scaling problem: CAN special hardware help? In *19th ACM/IEEE Design Automation Conference*, pages 339–43, 1982.

[Ake67] S B Akers. A modification of Lee's path connections algorithms. *IEEE Transactions on Electronic Computers*, EC-16:97–8, 1967.

[AS88] M H Anold and W S Scott. An iterative maze router with hints. In *Proceedings of the 25th ACM/IEEE Design Automation Conference*, pages 672–6, 1988.

[AS98] T Adler and J Scheible. An interactive router for analog IC design. In *Proceedings. Design, Automation and Test in Europe*, pages 414–20, 1998.

[Asa82] T Asano. Parametric pattern router. In *ACM IEEE Nineteenth Design Automation Conference Proceedings*, pages 411–7, 1982.

[Bak90] H Bakoglu. *Circuits, Interconnects and Packaging for VLSI.* Addison-Wesley, 1990.

[BDRV92] O Busset, M Declercq, F Rahali, and P Vaucher. A fast, single-layer, area router for semi-custom analogue circuits. *International Journal of Circuit Theory and Applications*, 20(3):283–98, may–jun 1992.

[BP83] M Burstein and R Pelavin. Hierarchical wire routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and System*, CAD-2(4):223–34, October 1983.

[BW90] M Bartholomeus and W Weisenseel. A routing concept for large sea-of-gates designs. In *Proceedings, Euro ASIC*, pages 225–29, 1990.

[CCC+97] H Chang, E Charbon, U Chowdhury, A Demir, E Liu, E Malavasi, A Sangiovanni-Vincentelli, and I Vassiliou. *A Top-Down Constraint-Driven Design Methodology for Analog Integrted Circuits.* Kluwer Academic Publishers, 1997.

[CI77] W G Cage and R J Smith II. A rectangle probe router for multi-layer p.c. boards. In *Proceedings of the 14th Design Automating Conference*, pages 13–23, 1977.

[CR88] J P Cohoon and D S Richards. Optimal two-terminal alpha - beta wire routing. *Integrating. The VLSI Journal*, 6(1):35–57, May 1988.

[CS89] W Choi and G Sobelman. Hardware rip-up router with concurrent wavefront propagation. *Electronics Letters*, 25(6):373–4, March 1989.

[CSV90] U Chowdhury and A Sangiovanni-Vincentelli. Constraint generation for routing of analog circuits. In *Proceedings IEEE/ACM Design Automation Conference*, pages 561–6, 1990.

[CTY91] G-X Cheng, M Tanaka, and M Yamada. A parallel routing technique based on local current comparison. In *1991 International Symposium on Circuits and Systems*, volume 5, pages 3114–7, 1991.

[Dij59] E W Dijkstra. A note on two problems in connection with graphs. *Numerical Mathematics*, 1:269–71, 1959.

[DK82] W A Dees, Jr. and P G Karger. Automated rip-up and reroute techniques. In *Proceedings of the ACM/IEEE Nineteenth Design Automation Conference*, pages 432–9, 1982.

[DL89] M S Damnjanović and V B Litovski. A survey of routing algorithms in custom IC design. *Journal of Semicustom ICs*, 7(2):10–9, 1989.

[DRR90] R Dutta, A Roy, and R Rao. Multilayer area routing algorithm as an optimization problem. In *Proceedings of the IEEE 1990 Custom Integrated Circuits Conference*, pages 27.4.1–4, 1990.

[DS81] W A Dees, Jr. and R J Smith. Performance of interconnection rip-up and reroute strategies. In *Proceedings of the ACM/IEEE Eighteenth Design Automation Conference*, pages 382–90, 1981.

[FMBS85] A C Finch, K J Mackenzie, G J Balsdon, and G Symonds. A method for gridless routing of printed circuit boards. In *2nd ACM/IEEE Design Automation Conference Proceedings*, pages 509–15, 1985.

[FW96] T Fujii and T Watanabe. A hierarchical mcm routing using four-via routing. In *IEEE Asia Pacific Conference on Circuits and Systems*, pages 389–92, 1996.

[GH89] S H Gerez and O E Herrmann. Cracker: A general area router based on stepwize reshaping. In *Proceedings of the ICCAD*, pages 44–47, 1989.

[GW96] M Guruswamy and D F Wong. Echelon: a multilayer detailed area router. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(9):1126–36, September 1996.

[Had76] F O Hadlock. A shortest path algorithm for grid graphs. *Networks*, 7:323–34, 1976.

[Hei69] S Heiss. A path connection algorithm for multi-layer boards. In *Proceedings of the Eigth annual 1969 IEEE Region III Convention*, pages 86–96, 1969.

[HEJ94] E P Huijbregts, J T S Van Eijndhoven, and J A G Jess. On design rule correct maze routing. In *Proceedings The European Design and Test Conference*, pages 407–11, 1994.

[Het98] A Hetzel. A sequential detailed router for huge grid graphs. In *Proceedings. Design, Automation and Test in Europe*, pages 332–8, 1998.

[Hig69] D W Hightower. A solution to routing problems on the continuous plane. In *Proceedings of the 6th Design Automation Workshop*, June 1969.

[Hig83] D Hightower. The Lee router revisited. In *Proceedings of the International Conference on Computer-Aided Design*, 1983.

[HJ93] E P Huijbregts and J A G Jess. General gate array routing usign a k-terminal net routing algorithm with failure prediction. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 1(4):473–81, December 1993.

[HNS81] S J Hong, R Nair, and E Shapiro. A physical design machine. In *VLSI 81*, pages 257–66, London, 1981. Academic Press.

[Hoe76] J H Hoel. Some variations of Lee's algorithms. *IEEE Transactions on Computers*, C-25(1):19–24, January 1976.

[HSB80] W Heyns, W Sansen, and H Beke. A line-expansion algorithm for the general routing problem with guaranteed solution. In *Proceedings of the Seventeenth Design Automation Conference*, pages 243–9, 1980.

[IV85] A Iosupovicz and A Vahidsafa. Parallel routing on a hardware array router. In *Proceedings of the International conference on Computer-aided design*, pages 136–38, November 1985.

[JH86] J F Johns and G D Hachtel. A zone expansion algorithm for gridless maze routing on a continuous plane (VLSI). In *1986 IEEE International Symposium on Circuits and Systems*, volume 1, pages 331–4, 1986.

[KAE91] T M Kurc, C Aykanat, and F Ercal. Parallelization of Lee's routing algorithm on a hypercube multicomputer. In F Bode, editor, *Distributed Memory Computing. 2nd European Conference, EDMCC2 Proceedings*, pages 244–53. Springer–Verlag, 1991.

[KC92] K-Y Khoo and J Cong. A fast multilayer general area router for MCM designs. *IEEE Transactions on Circuits and System II: Analog and Digital Signal Processing*, 39(11):841–51, November 1992.

[KC95] Kei-Yong Khoo and Jason Cong. An efficient multilayer MCM router based on four-via routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(10):1277–90, October 1995.

[KFM95] K Kawamura, S Fueki, and H Miwatari. Generalized touch and cross router. *Fujitsu Scientific and Technical Journal*, 31(2):208–14, 1995.

[Kir97] D A Kirkpatrick. *The implications of deep sub-micron technology on the design of high performance digital VLSI systems.* PhD thesis, University of California at Berkeley, 1997.

[KMB+98] S P Khatri, A Mehrotra, R K Brayton, R H M J Otten, and A Sangiovanni-Vincentelli. A noise-immune VLSI layout methodology with highly predictable parasitics. Technical report, Electronics Research Laboratory, University of California at Berkeley, 1998.

[KO90] E S Kuh and T Ohtsuki. Recent advances in VLSI layout. *Proceedings of the IEEE*, 78(2):237–63, February 1990.

[Kor82] R K Korn. An efficient variable-cost maze router. In *Proceedings ACM IEEE Nineteenth Design Automation Conference*, pages 425–31, 1982.

[KSS+90] K Kawamura, T Shindo, T Shibuya, H Miwatari, and Y Ohki. Touch and cross router. In *Proceedings International Conference on Computer-Aided Design*, pages 56–9, 1990.

[KU85] H Kitazawa and K Ueda. A look-ahead line search algorithm with high wireability for custom VLSI design. In *1985 International Symposium on Circuits and Systems*, volume 3, pages 1035–8, 1985.

[Lau80] U Lauther. A data structure for gridless routing. In *Proceedings of the 17th Design Automation Conference*, pages 603–9, 1980.

[LC96] D Li and J D Carothers. Mcm routing algorithm based on a compatibility graph approach. *Electronics Letters*, 32(1):5–6, jan 1996.

[Lee61] C Y Lee. An algorithm for path connections and its applications. *IRE Transactions on Elcetionic Computing*, EC-10:346–65, September 1961.

[LHT89] Y-L Lin, Y-C Hsu, and F-S Tsai. SILK: A simulated evolution router. *IEEE Transactions on Computer-Aided Design*, CAD-8(10):1108–14, 1989.

[LHT90] Y-L Lin, Y-C Hsu, and F-S Tsai. Hybrid routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 9(2):151–7, February 1990.

[Lin84] R Linsker. An iterative-improvement penalty-function-driven wire routing system. *IBM Journal of Research and Development*, 28, September 1984.

[LTS98] L-C E Liu, H-P Tseng, and C Sechen. Chip-level area routing. In *International Symposium on Physical Design*, pages 197–204, 1998.

[Lun88] R E Lunow. A channelless, multilayer router. In *Proceedings of the 25th Design Automation Conference*, pages 667–671, 1988.

[MAU86] H Miyashita, T Adachi, and K Ueda. An automatic cell pattern generation system for CMOS transistor-pair array LSI. *Integration. The VLSI Journal*, 4(2):115–33, June 1986.

[MBE94] Z Moosa, M Brown, and D Edwards. An application of simulated annealing to maze routing. In *Proceedings EURO-DAC 94*, pages 434–9, 1994.

[ME95] Z Moosa and D Edwards. An investigation of iterative routing algorithms. In *Proceedings EURO-DAC '95 European Design Automation Conference*, pages 91–6, 1995.

[Mik94] Y Miki. A fast vectorized maze routing algorithm on a supercomputer. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science*, E77-A(12):2067–75, December 1994.

[MP91] R Mall and L M Patnaik. Parallel maze routing on hypercube computers. *Computer Aided Design*, 23(6):454–9, July 1991.

[MRG+87] A Margarino, A Romano, A De Golria, F Curatelli, and P Antognetti. A tile expansion router. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, CAD-6(4):507–17, July 1987.

[MSKH93] M Mogaki, Y Shiraishi, M Kimura, and T Hino. Cooperative approach to a practical analog LSI layout system. In *30th Design Automation Conference*, pages 544–9, 1993.

[MSV93] E Malavasi and A Sangiovanni-Vincentelli. Area routing for analog layout. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(8):1186–97, August 1993.

[MT68] K Mikami and K Tabuchi. A computer program for optimal routing of printed circuit connectors. In *IFIPS Proceedings*, volume H47, pages 1475–78, 1968.

[NHLV82] R Nair, S J Hong, S Liles, and R Villani. Global wiring on a wire routing machine. In *Proceedings of the 19th Design Automation Conference*, pages 225–31, 1982.

[NISY95] M Nagamatu, S Ismail, R Shinji, and T Yanaru. Wire routing by lagrangian method. In *Proceedings of the Asia and South Pacific Design Automation Conference*, pages 837–43, 1995.

[NTR97] The National Tecnology Roadmap for Semiconductors. `http://notes.sematech.org/97melec.htm`, 1997.

[Oht86] T Ohtsuki. *Layout Design and Verification*, chapter Maze-running and line-search algorithms, pages 99–131. North-Holland, Amsterdam, 1986.

[OKK92] T Ohtsuka, H Kunieda, and M Kaneko. LIBRA: automatic performance-driven layout for analog LSIs. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E75-C(3):312–21, March 1992.

[OKK93] T Ohtsuka, N Kurosawa, and H Kunieda. The improvement in performance-driven analog LSI layout system LIBRA. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E76-A(10):1626–35, October 1993.

[Ous84] J Ousterhout. Corner-stitching: A data-structuring technique for VLSI layouts. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, CAD-3(1):87–100, Jan 1984.

[Pug78] L S Pugh. An improvement in printed circuit board routability using a maze-running algorithm. *Electronics Letters*, 14(1):8–9, January 1978.

[RB91] M Raith and M Bartholomeus. A new hypergraph based rip-up and reroute strategy. In *Proceedings of the 28th ACM/IEEE Design Automation Conference*, pages 54–9, 1991.

[Ros87] E Rosenberg. A new iterative supply/demand router with rip-up capability for printed circuit boards. In *Proceedings of the 24th Design Automation Conference*, pages 721–6, 1987.

[RR87] T Ryan and E Rogers. An ISMA Lee router accelerator. *IEEE Design and Test of Computers*, 4(5):38–45, 1987.

[Rub74a] F Rubin. An iterative technique for printed wire routing. In *Proceedings of the 11th Design Automation Workshop*, pages 308–13, 1974.

[Rub74b] F Rubin. The Lee path connection algorithm. *IEEE Transactions on Computers*, C-25:907–14, September 1974.

[SCA94] N K Sehgal, C Y R Chen, and J M Acken. A gridless multi-layer area router. In *Proceedings. Fourth Great Lakes Symposium on VLSI. Design Automation of High Performance VLSI Systems GLSV '94*, pages 158–61, 1994.

[SF79] J Soukup and S Fournier. Pattern router. In *Proceedings of the 1979 International Symposium on circuits and systems*, pages 486–9, 1979.

[SFH+91] Y Sekiyama, Y Fujihara, T Hayashi, M Seki, J Kusuhara, K Iijima, M Takakura, and K Fukatani. Timing-oriented routers for PCB layout design of high-performance computers. In *IEEE International Conference on Computer-Aided Design. Digest of Technical Papers*, pages 332–5, 1991.

[SJA94] H Spruth, F Johannes, and K Antreich. PHIroute: A parallel hierarchical sea-of-gates router. In *IEEE International Symposium on Circuits and Systems*, volume 1, pages 487–90, 1994.

[SKK+90] Y Shiraishi, M Kimura, K Kobayashi, T Hino, M Seriuchi, and M Kusaoke. A high-packing density module generator for bipolar analog LSIs. In *1990 IEEE International Conference on Computer-Aided Design. Digest of Technical Papers*, pages 194–7, 1990.

[SKS+90] T Shibuya, K Kawamura, T Shindo, H Miwatari, and Y Ohki. Application specific massively parallel machine. In *Proceedings of Frontiers '90*, pages 274–7, 1990.

[SLS89] E Shragowitz, J Lee, and S Sahni. *Progress in Computer-Aided VLSI*, volume 2: Techniques, chapter Placer-router for sea-of-gates design style, pages 43–92. Ablex Publishing, 1989.

[SMTO86] K Suzuki, Y Matsunaga, M Tachibana, and T Ohtsuki. A hardware maze router with applications to interactive rip-up and reroute. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, CAD-5(4):466–76, October 1986.

[Sou78] J Soukup. Fast maze router. In *Proceedings of the Fifteenth Annual Design Automation Conference*, pages 100–2, 1978.

[Sou81] J Soukup. Circuit layout. *Proceedings of the IEEE*, 69(10):1281–304, 1981.

[Sou92] J Soukup. Maze router without a grid map. In *1992 IEEE/ACM International Conference on Computer Aided Design. Digest of Technical Papers*, pages 382–5, 1992.

[SR81] J Soukup and J C Royle. On hierarchical routing. *Journal of Digital System*, 5(3):265–89, 1981.

[SSA89] Y Sone, S Suzuki, and K Asada. A gate matrix deformation and three-dimensional maze routing for dense MOS module generation. In *Proceedings of the IEEE 1989 Custom Integrated Circuits Conference*, pages 3.5.1–4, 1989.

[SSF93] Y Shiraishi, J Sakemi, and K Fukuda. A global routing algorithm based on the multi-commodity network flow method. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E76-A(10):1746–54, October 1993.

[SSH+93] T Shimamoto, I Shirakawa, H Hane, N Yui, and N Nishiguchi. A distributed routing system for multilayer SOG. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E76-A(3):370–6, month 1993.

[SST96] H Shirota, S Shibatani, and M Terai. A new rip-up and reroute algorithm for vary large scale gate arrays. In *Proceedings of the IEEE 1996 Custom Integrated Circuits Conference*, pages 171–4, 1996.

[SSU85] T Shimamoto, A Sakamoto, and A Ushida. A maze running router and its wirability for single-layer printed wiring boards. *Electronics and Communications in Japan I*, 68(5):21–30, May 1985.

[SSV87] H Shin and A Sangiovanni-Vincentelli. A detailed router based on incremental routing modifications: Mighty. *IEEE Transactions on Computer-Aided Design*, CAD-6(6):924–55, 1987.

[Sup82] K J Supowit. A minimum impact routing algorithm. In *Proceedings of the 19th Design Automation Conference*, pages 104–11, 1982.

[TCF92] Chia-Chun Tsai, Sao-Jie Chen, and Wu-Shiung Feng. An h-v alternating router. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(8):976–91, August 1992.

[TFS84] K Toyoda, K Fujimori, and Y Suzuki. Window pattern router (PCB routing). In *1984 International Conference on Industrial Electronics, Control and Instrumentation*, volume 1, page B23, 1984.

[TS88] Pin-San Tzeng and C Sequin. Codar: a congestion-directed general area router. In *IEEE International Conference on Computer-Aided Design. Digest of Technical Papers*, pages 30–3, 1988.

[TYKS80] F Tada, K Yoshimura, T Kagata, and T Shirakawa. A fast maze router with iterative use of search space restriction. In *Proceedings of the 17th Design Automation Conference*, pages 250–4, 1980.

[VAC90] M-K Vai, G Aybay, and S-M Chu. A maze routing method of combined unity and infinity expansion distances. In *Proceedings of the IEEE 1990 Custom Integrated Circuits Conference*, pages 27.3.1–4, 1990.

[VM90] R Venkateswaran and P Mazumder. A hexagonal array machine for multi-layer wire routing. *IEEE Transactions on Computer-Aided Design*, 9:1096–112, October 1990.

[VM93] R Venkateswaran and P Mazumder. Coprocessor design for multilayer surface-mounted PCB routing. *IEEE Transactions of Very Large Scale Integration (VLSI) Systems*, 1(1):31–45, March 1993.

[WKS87] T Watanabe, H Kitazawa, and Y Sugiyama. A parallel adaptable routing algorithm and its implementation on a two-dimensional array processor. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, CAD-6(2):241–50, March 1987.

[WLS93] C Wiley, K M Lau, and S A Szygenda. m3D: A multidimensional dynamic configurable router. In *Proceedings 1993 IEEE International Symposium on Circuits and Systems*, volume 3, pages 1857–60, 1993.

[WOYT98] T Watanabe, Y Ohtomo, K Yamakoshi, and Y Takei. An effective routing

methodology for Gb/s LSIs using deep-submicron technology. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science*, E81-A(4):677–84, April 1998.

[WS88] Youngju Won and S Sahni. Maze routing on a hypercube multicomputer. *Journal of Supercomputing*, 2(1):55–79, sept 1988.

[Xio86] J G Xiong. A gridless maze router: DBM (diffraction boundary method). In *IEEE International Conference on Computer-Aided Design. Digest of Technical Papers*, pages 192–5, 1986.

[XK81] J G Xiong and T Kozawa. An algorithm for searching shortest path by propagating wavefronts in four quadrants. In *Proceedings of the 18th Design Automation Conference*, pages 29–34, 1981.

[YDB93] I-L Yen, R M Dubash, and F B Bastani. Strategies for mapping Lee's maze routing algorithm onto parallel architectures. In *Proceedings of the Seventh International Parallel Processing Symposium*, pages 672–9, 1993.

[zBMK+93] V Meyer zu Bexten, C Moraga, R Klinke, M Brockherde, and K-G Hess. ALSYN: flexible rule-based layout synthesis for analog IC's. *IEEE Journal of Solid-State Circuits*, 28(3):261–8, March 1993.

[ZLI93] S-Q Zheng, J SJ Lim, and S S Iyengar. Efficient maze-running and line-search algorithms for VLSI layout. In *Proceedings IEEE Southeastcon '93*, page 648, 1993.

[ZW97] Hai Zhou and D F Wong. Crosstalk-constrained maze routing based on lagrangian relaxation. In *Proceedings International Conference on Computer Design*, pages 628–33, 1997.