An Overview of Placement and Routing Algorithms for PCB, VLSI, and MCM Designs with a Proposal for a New MCM Routing Algorithm

A Technical Report submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Engineering

By

Charles N. Frisbee

May 1996
Department of Computer Systems Engineering
University of Arkansas

Table of Contents

ABSTRACT

Placement and routing algorithms are an important means for achieving fast, high quality layouts of high density circuit chips. Packaging of Multi-Chip Modules (MCM) places further demands on these placement and routing algorithms than on VLSI or PCB layouts. However, many of the existing techniques and algorithms for VLSI and PCB placement and routing are well-suited for use in MCMs and can be combined together or modified to help solve the MCM layout problem. This paper presents a broad overview of the different kinds and categories of placement and routing algorithms for PCB, VLSI, and MCM designs in use today. Many specific algorithms are also presented. This paper also addresses the application issues of using different placement and routing algorithms. This is done by comparing and contrasting these algorithms in order to confront the issues of advantages, disadvantages, and trade-offs of different algorithms.

Finally, this paper proposes a strategy for solving the MCM routing problem using existing layout techniques and presents some important considerations needed to turn this strategy into a working algorithm. This paper's primary intent is to show the paradigm of combining existing research of layout algorithms with

new ideas in order to develop new algorithms for higher demanding circuit packages.

1.0 Introduction.

As new algorithms are sought to satisfy the layout demands of high density circuit packages, research into existing layout techniques is necessary. Layout of integrated circuits is the process of designing the physical representation of the circuit scheme and its connectivity between circuit modules. The two phases of the layout process are called *placement* and *routing*. Study into the many different and varied placement and routing algorithms and approaches provides assistance in designing new algorithms, improving old algorithms, avoiding endeavors leading to unavailing results, and adapting existing algorithms to meet current demands.

The purpose of this paper is to contribute a single assembled resource of information on the different kinds of placement and routing algorithms for PCB, VLSI, and MCM algorithms. Also, this paper proposes a strategy for ultimately achieving a new, higher functional MCM routing algorithm. Strategies and ideas from existing routing and placement methods are gathered together and discussed to suggest another solution to the MCM routing problem.

2.0 Placement and Placement Algorithms

Placement is the process of physically arranging electronic functions (library cell circuits, integrated circuit (I.C.) chips, module components, etc.), on a planar surface (silicon substrate, printed circuit board (PCB), multichip module (MCM)) in a desired manner. Placement is one of two phases in the layout process with the second phase being the routing of the placed components. Thus, placement is a process ranging from the macrolevel (i.e. I.C. chips) to the microlevel (i.e. multichip modules).

A placement algorithm can be defined as a procedure for physically arranging all electronic functions on a planar surface such that minimum wiring and minimum area result. Therefore, the algorithm tries to satisfy a desired optimization. Other optimization criteria include minimizing delay, minimizing parasitic capacitance and inductance couplings [0] (in the case of a PCB), and meeting thermal considerations such as the designer's heat dissipation requirements. It should be noted that no one placement algorithm achieves a complete optimization, and the choice of a placement algorithm depends on the designer's objectives and goals. Typically, algorithms are combined or adapted to interact with one another to gain further improvement. The choice of a placement algorithm can also affect the level of circuit testing quality [0].

There are basically two main classes of placement algorithms: iterative and constructive. Iterative placement algorithms operate by creating a very crude placement and

improving it in a step-by-step fashion [0]. Constructive placement algorithms, in complete contrast to iterative placement algorithms, try to place modules one at a time in a position that is usually not far from the final solution [0]. The significant part of a constructive algorithm spends time calculating the position of each module and is, thus, computationally intensive. Most algorithms available today adapt known methods to achieve the designer's goals, or they are optimized to increase efficiency of characteristics such as computation time or better placement. Other kinds of algorithms exist, and these will be introduced.

Placement algorithms can also be described as being rigor or heuristic [0]. Rigor consists of step-by-step computations until a solution or optimal solution is found. However, these computations can be too costly when time is a limitation. Therefore, heuristic algorithms are implemented. Heuristic algorithms have evolved from inductive reasoning based on past experience instead of mathematical rigor. The heuristic process is designed to capitalize on the statistical properties of "real" circuits.

2.1 Iterative Algorithms.

Many iterative-based algorithms exist for placement and typically use a simulation of a natural process to perform the optimization. Criteria (i.e. design goals) also play an important role. The force-directed method is a common iterative approach [0]. The algorithm treats each module as a point mass,

and the interconnecting nets as springs, of weighted force constant. A few modules are assumed fixed and the rest are allowed to move around until they assume a minimum "energy" configuration. The term "energy" is defined as a function of intermodule spacing and connectivity. The process of "relaxing" the system repeats itself until the module movements become negligible. Another method similar to the preceding is called the attractive and repulsive force method (AR method) [0].

The "scatter-and-gather" method is an example of combining several algorithms and techniques together to accomplish floor planning of IC chips [0]. The "scatter" phase uses the force-directed method and also unconstrained cluster growth (UCG) and constrained cluster growth (CCG). CCG considers size and aspect ratio of the package as fixed, and the center of the package always coincides with the center of mass of the cluster. The "gather" phase tries to solve a geometric problem called minimal box embedding (MBE) which consists of two solution techniques: GBS (growing box scheme) and RBS (reducing box scheme). Speedup techniques called incremental transform/sorting and quadrant folding are also used.

2.2 Simulated annealing.

Simulated annealing (SA) algorithms fall under the category of iterative placement algorithms [0]. Simulated annealing tries to minimize the overall "energy" of the system by mathematical methods that closely resemble the way systems in nature relax. The "energy" is a measure of the total wire length congestion,

which can be computed with the use of a *net-crossing histogram*, and is represented with an objective function [0]. The package surface is divided into natural boundaries, and a histogram is used to represent the number of nets crossing each boundary. Both horizontal and vertical net-crossing histograms are constructed. The information in each histogram is combined into an objective function by first introducing a threshold level for each histogram - an amount of wire that will nearly exhaust the available wire capacity. Next, we sum for all histogram elements that exceed the threshold the square of the excess over the threshold. Finally, we obtain the objective function by adding this quantity to the lower bound wire length provided by the peak of the histograms.

The SA algorithm starts with a random layout of modules and then performs random module interchange (usually pairwise). The change in "energy" is computed, and if the new configuration has a lower energy than the previous one, it is used as the basis for further interchange. However, there is still a probability of acceptance of the new configuration if it is higher in "energy" than the previous one. This acceptance is termed an "uphill climb", and it is used to allow the algorithm the possibility to leave an area of a local minima and search other areas of the solution space. Many placement algorithms use simulated annealing as a basis for either improving the SA algorithm or adapting it to their needs.

2.2.1 Improvements to simulated annealing.

One attempt at improving the SA algorithm involves exploiting the probabilistic nature of the algorithm to reduce computation time by doing cost calculations approximately instead of exactly [0]. Another attempt at improvement is the reduction of the search space of the placement problem and preventing the overlap between modules [0]. The placement by this algorithm is comparable to conventional SA algorithms, but the improvement is manifested in the form of reduced computation time. Other improvements to SA include a cell clustering technique to improve computation time.

The SA algorithm has also been implemented with parallel processors with capability of integrated error control [0]. An example of the parallel processor approach is an algorithm called heuristic spanning [0]. It's used to replace the "high-temperature" portion of simulated annealing. The "low-temperature" portion is sped up by the section annealing technique. Each processor is assigned a separate section to optimize and communicates its move with the other processors. Parallel processors can also achieve a hierarchial simulated annealing (HSA) method. This method divides a given problem into sub-problems applying the SA method at each stage. This feature makes it possible to automatically choose suitable parameters in the cost functions at each stage of computation [0].

2.2.2 A detailed look into simulated annealing.

The simulated annealing technique has become a foundation for further improvement into the placement and layout problem.

Examples stated earlier have shown that improvement is in the form of faster computation time and better placement optimization. Typically, these two criteria are in direct conflict with each other. That is, as one condition is improved the other is worsened. This section describes the SA algorithm in further detail and also describes an improvement called simulated sintering (SS) [0].

Simulated annealing, in the general case, exhibits two main weaknesses in trying to achieve fast computation time and good placement optimization. First, the computation time is slowed down by starting with an initial random placement. Second, the lack of an optimal cooling schedule for the problem prohibits the algorithm from producing a desired tradeoff between cost (in time) and optimization. Simulated sintering is used to solve these two problems and consists of two parts, starting with a "good" initial placement to improve computation speed and the inclusion of an optimal cooling schedule to control termination criteria. The "good" initial improvement is usually obtained by a fast heuristic algorithm. The choice of the algorithm depends on the problem at hand, but the time taken by the heuristic algorithm should be less than the time taken by the general SA algorithm to achieve the same result or no improvement will take place.

The cooling schedule controls the probability of accepting an uphill move in the SA technique and gives direction to the search through the solution space. Two types of cooling

schedules can be used, *goal-directed* and *extended goal-directed*. Goal-directed scheduling uses the cost of a minimal solution to a problem to determine uphill acceptance. Extended goal-directed scheduling is a variation on goal-directed scheduling which allows temperature to increase or decrease. Changing the temperature during the optimization process controls the speed and direction of the search (determined by the acceptance of an uphill climb). This is what the cooling schedule is designed for. A high acceptance rate (increasing the temperature) for an uphill climb is desirable during the initial stages of the optimization process, and a low acceptance rate (decreasing the temperature) is desirable during the final stages.

The success of simulated sintering hinges on the use of a good cooling schedule and the identification of an appropriate initial temperature. If the initial temperature is too high, then the SS algorithm deteriorates to simulated annealing and no improvement is achieved. If the initial temperature is too low, it results in a non-minimal solution. Thus, what is required is the ability to relate configuration costs (time and optimization quality) to temperature values and come up with a convergence criteria (determination of best solution). One possible solution is to base it on experience, but this attempt restricts itself to certain problem types and doesn't try to find a global solution process. Three convergence criteria used in simulated annealing can be controlled by simulated sintering. These criteria are listed as follows:

1. Converge when the number of accepted configurations, expressed as a percentage of the total number of configurations generated at a particular temperature, falls below a predetermined value.

2. Converge when the cost of current best configuration remains unchanged for a number of consecutive temperature values.

3. The ability to accept uphill moves is no longer required when all configurations accepted at a given temperature value have similar cost. Consequently, the SA algorithm can terminate or converge whenever the similar cost condition is detected.

Simulated sintering has been shown to be a good improvement to the SA method. It allows placement optimization to be based on chosen convergence criteria, but most importantly steers the search direction for a solution. This allows a tradeoff between desired cost and desired optimization.

2.3 Genetic Algorithms.

Genetic algorithms are an iterative approach to placement. These types of algorithms attempt to achieve better and better optimization based on the biological evolution process. The algorithm starts with an initial placement and uses genetic operators to do the optimization. Three common operators are used by genetic algorithms: crossover, mutation, and inversion. Crossover, the main genetic operator, is used to combine two current configurations to produce a new configuration. Mutation, a background operator, is used to produce spontaneous random changes in various configurations. Inversion involves taking a random segment in a solution representation and flipping it. The rates of these genetic operators can be changed, automatically

improving efficiency. Each of these types of genetic operators may be used in various ways to explore the solution space in a more efficient manner [0,0,0,0,0].

2.4 Miscellaneous placement algorithms.

The following algorithms are various other methods for placement that have been studied and will be briefly described. Techniques and procedures used in placement are also listed and explained.

1. An algorithm that applies bin-packing to the building block placement problem based on classification of blocks [0].
2. A placement method using fuzzy set theory based on a combination of fuzzy similarity relation, and a modified c-means clustering algorithm [0].
3. An algorithm that optimizes placement by using "statistical cooling" [0].
4. An algorithm based on the divide-and-conquer paradigm that divides sets of logic modules into small clusters, generates an optimal placement for each cluster, and then combines each smaller solution to the original placement problem [0].
5. An algorithm that assumes placement has been done, but pin assignments can still be set to minimize wire connections [0,0].
6. An algorithm for the placement of macrocells in VLSI based on the blackboard model that is suitable for parallel processor implementation [0].
7. An O(n)-time algorithm on the constrained multistage (CSMG) model. The algorithm uses the line sweep method [0].
8. An algorithm based on simulated surface tension that assigns cohesion and adhesion forces to the cells and their surroundings [0].
9. An algorithm based on the self-organization process proposed by T. Kohonen which is a learning algorithm for neural networks that adjusts the weights of links connecting nodes and inputs so that nodes connected closely topologically are sensitive to inputs having similar properties. This algorithm is well suited for parallel implementation [0].
10. An algorithm based on the neural somatotopical mapping [0].
11. An algorithm to determine whether a placement of N rectangles can be represented by a slicing tree and then

determines the minimum height. This is used in the top-down approach to placement which consists of floorplanning and global wiring [0].

12. An algorithm for placing rectangular blocks in the Euclidean plane based on the Lennard-Jones 6-12 potential equation and minimizing the sum of the squares of the Euclidean distances of the block interconnections [0].
13. An algorithm that formulates the problem as a relaxed integer linear max-flow problem which is an NP -complete problem. This algorithm is used in multilayer printed circuit board (MPCB) layout. The solution can be obtained by solving a linear programming problem [0].
14. The bipartitioning method by Kernighan and Lin (KLM) and an algorithm using neural networks to improve the behavior. The KLM method generates a balanced 2-way partition of the blocks such that the wiring across the partition is minimum [0].
15. A parallel algorithm for tiling with polyominoes. The tiling problem is to pack polyominoes in a finite checkerboard [0].
16. An iterative algorithm based on eigenvector decomposition which gradually reduces the search space [0].
17. An algorithm using adaptive and look-ahead procedures with constraints on routability, area, and timing [0,0,0].
18. An algorithm that repeatedly solves sparse linear equations using successive over relaxation (SOR) to solve the linear equations. Also, a BGS (block Gauss-Seidel) iteration scheme is used to achieve global optimum results [0].
19. A placement algorithm based on analysis of the manual design process. The algorithm functions by using knowledge gained from manual design experience [0].

2.5 Application issues in using placement algorithms.

One of the first considerations in choosing a placement algorithm is that of looking at its strengths and weaknesses to solve the particular problem in a desired fashion. No one algorithm can satisfy all design, implementation, and solution constraints. Thus, comparison and contrast measurements of different algorithms are an important and valuable resource for

the designer to have. The following section performs a compare and contrast of the major types of placement algorithms which includes discussion of trade-off and performance issues.

All circuit placement problems are optimization problems of some kind and size, and thus all placement algorithms attempt to achieve a desired degree of optimization ranging from global to some lesser, acceptable degree. The major trade-off is between increasing computation time for an optimum solution and reasonable time for a less optimum (but perhaps acceptable) solution.

Constructive placement algorithms provide the ability to obtain highly optimal solutions at the cost of high computation time. Some speed improvement can be gained through faster hardware, such as parallel processors, and modest programming techniques. Obviously, these kinds of algorithms should only be used when a design demands an extremely strict placement of modules.

Iterative placement algorithms are generally of greater interest than constructive algorithms. An iterative-type algorithm should be used when time is a factor and a less than optimal solution is acceptable. The strategies used in an iterative algorithm employ techniques or a combination of techniques from other iterative algorithms in order to improve certain constraints. Since most of the specific iterative placement algorithms presented in literature only claim to gain improvements over a select few (or single) placement algorithms,

the choice of a specific algorithm warrants separate research into what improvements are offered by the algorithm and what criteria is satisfies. Simulated annealing algorithms will be used in this report as a "gauge standard" for comparison and contrast of placement algorithms.

The basic advantages of simulated annealing and the advantages over other placement algorithms are:[0]

1. Ability to jump out of local minima by accepting uphill climbs and potentially falling into a more promising downhill path.
2. Highly suitable to problems having an increasing order of magnitude on the number of modules to be placed.
3. Ability to optimize a desired set of parameters through the use of a cost function.
4. Greater control of moving through the search space by use of a SA cooling schedule.
5. Only an initial random placement needed to obtain a solution.
6. Alleviates the need for complex mathematical calculations to place a module during the solution process.
7. May be applied to existing placement solutions for even further improvement.

The disadvantages of simulated annealing are:

1. Undeterministic solution. The probabilistic nature of choosing moves causes the solution to change during every execution.
2. Optimal cost functions may be very sophisticated, especially for large scale problems.
3. Smaller scale placement problems may be less efficiently solved (i.e. require more time) with SA than with other iterative algorithms.
4. SA algorithms still require large amounts of computation time for problems of high order.
5. A good cooling schedule is needed to obtain a good solution.

Another favorable optimization technique for module placement is the genetic algorithm (GA). These algorithms, like

SA, are very useful in large-scale combinatorial placement problems. Genetic algorithms are normally very comparable to simulated annealing in both solution quality and speed of execution. Moreover, each algorithm has its general strengths and weaknesses with specific algorithms exhibiting better results over the other kind. The advantages of genetic algorithms over simulated annealing are as follows [0,0,0]:

1. Concurrent search mechanism. GA works with a population of solutions and searches a large number of configurations from this population as opposed to SA which works with only one configuration at a time.
2. GA can learn from past trials of searching through the solution space. SA does not have this ability.
3. GAs allow the good features of candidate solutions to remain in helping to form better solutions.
4. GA can process inferior configurations without compromising the best ones.

The disadvantages are:

1. GA may require more memory space.
2. GA must have a good starting "genetic code" representation of initial layout or poor results may occur.
3. The crossover operator, the main operator in GA, must be able to avoid conflicts in combining 2 different configurations.
4. GA are unable to perform hill-climbing that might result in escaping local minima.

Despite the above listed advantages and disadvantages of GA and SA algorithms, knowing which algorithm is best to use for any one problem is difficult. Furthermore, each kind of algorithm has many variations that exploit certain strengths. GA may use any combination of the three different genetic operators (crossover, mutation, and inversion) to aid in efficient search.

SA algorithms may use partitioning or optimization of the cooling schedule.

Finally, even after a designer has found a very optimal placement algorithm for their problem, the final decision will be based on one or more limiting factors. These factors may include the following:

1. Module shapes.
2. Hardware available (i.e. memory space).
3. Electromagnetic (EM) factors.
4. Thermal considerations.
5. Modification concerns.
6. Testability requirements.
7. Obstacles.

Typically, these factors may necessitate the need for a less optimal algorithm in order to satisfy some limitations. In conclusion, wisely applying a placement algorithm ultimately relies upon obtaining a solution that meets design criteria, but consideration for the "right" placement algorithm can result in a solution in less time and with lower manufacturing costs.

3.0 Routing and Routing Algorithms.

Routing is the process of finding a path between a set of points around a set of blocks on a two-dimensional plane without any path crossing another path on the same layer. Multilayer routing uses vias which are used to connect the same net on different layers. The routing problem usually consists of many constraints and optimization criteria, all of which depend on the designer's goal. These constraints and optimization criteria may include such factors as minimizing total wire length, minimizing

the number of routing layers, and minimizing the longest wire interconnection [0]. Other factors exist, but they are all devised to achieve a lower cost. The purpose of a routing algorithm is to achieve the criteria or constraints chosen by the designer. The routing problem is generally broken down into two subproblems, *global* and *detailed routing*, and thus each may use a different algorithm [0].

Routing is usually done on at least two levels of metallization. Each level is composed of a number of routing channels capable of supporting a limited number of wire nets. This limit is known as the *channel capacity*. A routing channel is defined as the rectangular area set aside for routing nets (or wires) between different functional blocks on the layout surface [0]. Fixed pins are located on two parallel sides of the rectangular area while the other two sides have transient pins, pins that do not have a fixed location initially [0]. The "switch-box" channel is the same as the routing channel but has fixed pins on all four sides. Figure 1 shows an example of each kind of channel routing area. In special cases, however, routing paths may enter into the module placement area.

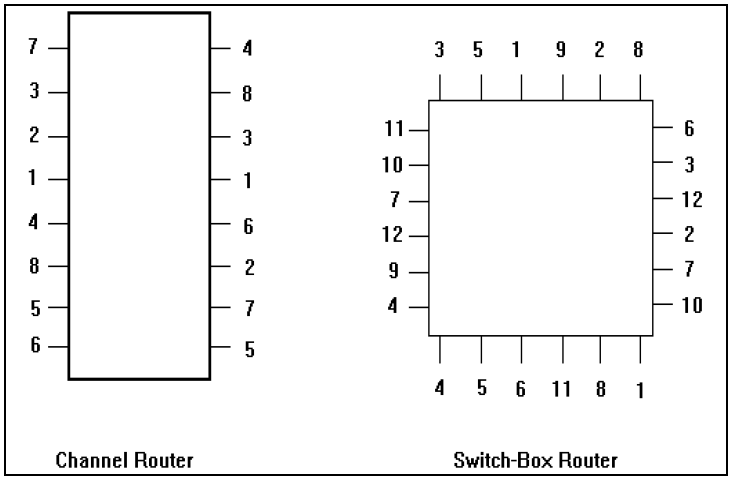


Figure 1

3.1 Global routing.

Global routing is concerned with choosing which set of routing channels a net will occupy [0]. In other words, what side of the rectangular area of the routing channel the net will enter and leave by. This step also decides on the order in which channels should be routed [0]. Global routing is also concerned with the placement of the routing channels on the layout surface (which is normally determined by or limited by the placement of the circuit modules).

3.2 Detailed routing.

Detailed routing is concerned with exactly where on each face of the rectangle (i.e. which fixed pin) the net will cross and how the net is routed across to the other side of the channel [0]. Thus, detailed routing follows the global routing procedure. The essential distinction is that detailed routing operates on each channel *in isolation* from the rest of the system, which makes it simpler and cheaper to process [0].

3.3 Routing Algorithms and Routing Approaches.

Numerous algorithms and techniques exist for the routing problem. The most general algorithms are those called *maze-routers* and *line routers* [0]. Each of these techniques route one net at a time which means that unrouted nets may become blocked by the already routed nets. This requires manual intervention to complete the routing process. The next class of routing techniques are *channel* and *switch-box routers* [0] (used in the detailed routing phase). These routers consider interaction between the nets before routing, but they are only able to route one row or column at a time. Thus, like the previously discussed routers, manual intervention may be required if routing is not done correctly.

There are three characteristics common to known routing approaches [0]. Each of these show the need for further research and improvement into the routing process. The three common characteristics are listed and explained as follows:

1. Brute force - this approach tries to solve the routing problem without knowledge of the way human designers successfully route. Because of this 100% wiring is not met, and more intelligence must be programmed into the routing algorithm.

2. User interaction - most algorithmic approaches do not allow user interaction before, during, or after the routing process. Users should be able to modify already routed nets or guide the routing search space in a desired direction.

3. Unnecessary constraints - most routing algorithms impose unnecessary constraints such as assignment of different layers to different

directions. This imposition reduces the routing quality. Having the ability to relax this constraint or other constraint, at times, can improve the quality of routing.

3.4 Miscellaneous routing algorithms.

The following is a list of miscellaneous routing algorithms or approaches to implementing a certain routing algorithm:

1. A constructive parallel routing method based on selecting local maximum current in a unity resistive network whose goal is to obtain the 'field' expressed with a Poisson equation [0].
2. A routing algorithm to take into account relevant timing information and crosstalk noise requirements [0].
3. A new construct called connection graph, G//c, generated by a geometric algorithm has been proposed to design a class of time and space efficient minimum spanning tree algorithms. These tree algorithms can be used in maze-running and line-search algorithms [0].
4. The use of hardware accelerators such as a reduced array architecture (RAA) [0] and a ARCO architecture [0] to speed up already existing routing algorithms.
5. A parallel algorithm to solve the top-bottom routing problem, which is an important subproblem of routing wires around a rectangle in two layers. The routing problem is no harder than the prefix minima problem for inputs drawn from the range of integers [1..s] and input of size n [0].
6. A routing algorithm based on an iterative routing process in which an initial layout is gradually improved. The initial layout is obtained by constructing a minimum distance Steiner tree for each net. This algorithm is quite general and could be applied to both printed circuit boards and integrated circuit chip wiring [0].
7. A parallel routing algorithm for multi-layer channel routing problems on the HVH model which minimize wiring areas in VLSI circuits and PCBs [0].
8. A routing method called Floating Track Method which ensures a 100 percent connection ratio. A line-search algorithm automatically provides a connection path, and the problem of unconnected pin pairs is solved by adding extra wiring tracks. Prevention rules and correction procedures are provided for the wiring shorts and disconnections caused by the track addition. A fast algorithm is developed for determining the location of additional tracks [0].
9. A routing procedure which recursively cuts the area of the chip into smaller and smaller regions until the

routing problem within a region can be handled by the Dantzig-Wolfe decomposition method. After this, the adjacent regions are pasted together to obtain the routing of the whole chip [0].

10. A routing algorithm that considers topological relationship among nets. The algorithm consists of two routing phases: topological routing and geometrical routing. The aims are to minimize net intersections, number of vias used, and space required for routing completion [0].
11. A line search algorithm with a look-ahead strategy. A special flagging and backtracking strategy guarantees that a solution is found if one exists [0].
12. A zone expansion algorithm for routing on a gridless plane. The algorithm finds a solution for the problem of finding a path to connect a set of points on a plane which contains a maze of obstacles [0].
13. An MCM performance-driven routing algorithm based on a new second-order propagation delay model for RLC interconnection trees [0].

3.5 Applications issues in using routing algorithms.

The choice of a routing algorithm demands the same basic considerations as that of placement algorithms such as the design, implementation, and constraint criteria. Section 3.0 has already discussed some of the issues of choosing a routing algorithm (i.e. global routing algorithms or detailed routing algorithms). Also, since most routing algorithms are generally written for a specific kind of packaging (i.e. PCB, VLSI, or MCM), this will obviously narrow the number of choices for the designer. This next section presents various routing requirements, performs a compare and contrast of major types of routing algorithms in terms of their strengths and weaknesses, and considers the choice of a specific routing algorithm.

Three of the most sought after characteristics of any routing algorithm are quick performance, minimizing total wire

length, and minimizing the number of routing layers. However, many other characteristics are normally desired, all of which are usually only partially satisfied by any one routing algorithm. Furthermore, it must be stated that certain routing requirements can conflict with others, and, thus, design trade-offs are necessary. The following shows a number of routing requirements for MCMs [0], but also pertain to PCB and VLSI routing:

1. Minimize delay, given a priority weight and maximum/ fixed delay assignment.
2. Equalize delay for signal groups, with specified tolerance.
3. Produce minimum bends.
4. Handle stacked and unstacked vias and to control the number of vias.
5. Assign layers constrained with maximum number of vias allowed to a net.
6. Regulate lengths and delays to meet timing and noise margins.
7. Be easily extensible as technologies change.
8. Constrain routes to specified layers (e.g., designated power, ground, or signal layers.
9. Pick the right kind of via based on layer change and technology used.
10. Specify route ordering.
11. Accept preferred directions on specific layers or nets.

Other possible routing requirements (with references to algorithms incorporating them) include crosstalk considerations (includes mixing of analog and digital nets) [0,0], handling routing around arbitrary obstacles [0,0], handling wiring density [0,0], minimizing the longest interconnection [0], arbitrary angle routing [0], minimizing clock skew [0], and variable width traces [0].

Almost all routing algorithms can be categorized as being based on *single-layer routing (SLR)* or *xy plane-pair routing*

(XYR), and each one is used in certain ways to exploit its strengths. SLR assigns an entire net to a layer and attempts to route as many other whole nets on one layer as possible. XYR assigns a net to different layers by splitting the net into one or more layers. SLR is normally used for time critical nets, power, and ground nets. XYR is used to reduce the number of layers required to complete all routing. SLR reduces the number of staircase vias but greatly increases the number of layers. However, XYR reduces the number of layers, but increases the number of staircase vias.

In conclusion, the final choice of a routing algorithm may depend on one or more limitation factors. These factors are as follows:

1. Timing considerations.
2. EM factors (i.e. crosstalk and skin effect).
3. Hardware available (i.e. memory space).
4. Obstacles.
5. Testability requirements.
6. Modification concerns.
7. Wiring technology.

4.0 Simultaneous placement and routing.

Sometimes placement and routing are done simultaneously. This is known as the hierarchial layout method [0,0]. Several schemes can be employed to achieve a hierarchial layout method. Timing information can be used to influence the placement and wiring processes [0,0,0,0,0,0]. The min-cut technique, used in partitioning networks, and its variations try to balance the

minimization of interconnection length and area [0,0,0]. The min-cut algorithm recursively subdivides a placement while minimizing the number of wire crossings at each division line. One variation of the min-cut technique uses spiral-cuts to minimize and/or optimize the placement [0]. After the layout has been optimized, an adaptive correction procedure can be implemented for further improvement.

5.0 Proposition of a new MCM routing strategy.

Most new MCM routing algorithms are based on previous research of what approaches have satisfied the design criteria and what approaches have failed. The MCM routing problem is more difficult than VLSI or PCB routing problems because of the higher packing density in MCM designs. Moreover, MCM designs introduce more performance issues and more interconnection layers than VLSI or PCB designs. However, the techniques for solving these can still be applied to MCMs. Typically, strategies from several different routing algorithms are pieced together and/or modified to come up with a new routing algorithm. The following sections illustrate how existing ideas and approaches can be used to achieve a new MCM routing strategy, and how this might be transformed into a new algorithm for solving the MCM routing problem.

5.1 Overview of the MCM package design.

The MCM packaging technology is described as follows:[0] the top layer called the chip layer consists of the placement of all chips. Below the chip layer, there is a stack of pin

redistribution layers, whose purpose is to redistribute the pins under the chips, uniformly on the last pin redistribution layer. Below the last redistribution layer, there is a stack of signal distribution layers. These layers are used to complete the routing of the nets that connect the redistributed pins of the chips. The signal distribution layers are usually paired together into an x-y plane-pair. The x-plane has wiring channels only in the x direction, and the y-plane has wiring channels only in the y direction.

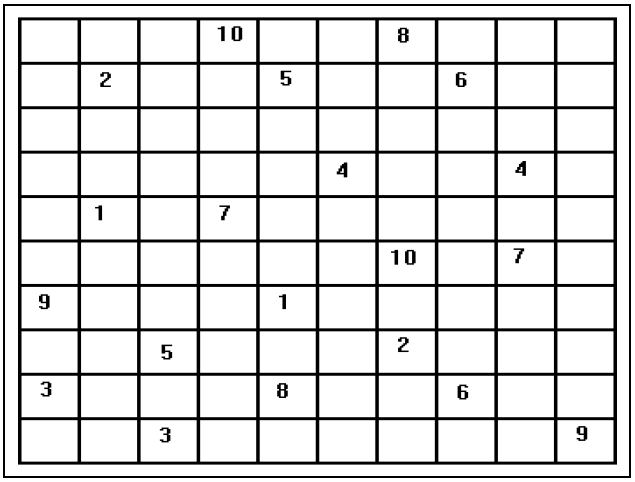


Figure 2 - Last pin redistribution layer

5.2 Definition of the MCM routing problem.

The goal for the proposed algorithm is to complete the routing of the signal distribution layers given the set of pins on the last redistribution layer and achieve a desired total routing cost. Figure 2 shows an example of a final redistribution layer. The user assigns routing costs (via costs)

to each net whose purpose is to allow the algorithm to work towards a total minimum routing cost and obtain a desired circuit performance. Two kinds of via types exist, stacked vias and staircase vias [0]. A stacked via is referred to as the via when a net changes a layer at the same point it started with. A staircase via is referred to as the via introduced when a net changes layers at a new position. Figure 3 shows the difference between a stacked via and staircase via.

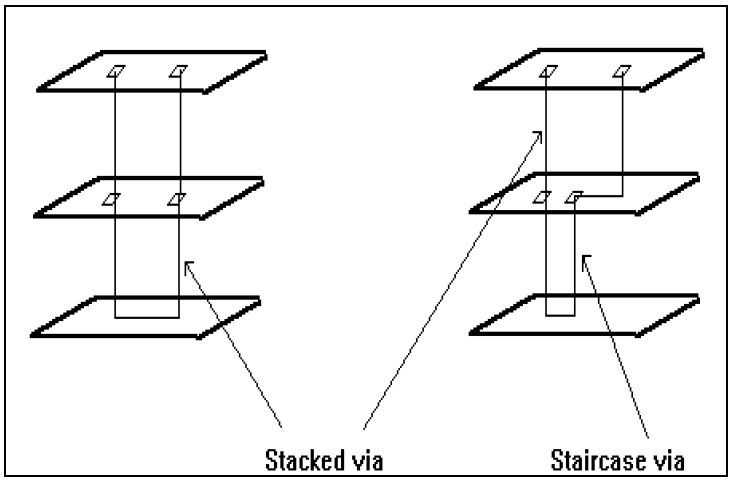


Figure 3 Stacked vias vs. Staircase vias

The following equation summarizes the cost of MCM routing for the proposed algorithm: [0] Cost = w_k x number of layers + w_{strv} x number of staircase vias + w_{stkv} x number of stacked vias, where w_k, w_{strv}, and, w_{stkv} are constants that control the relative importance of each item. As can be seen from the equation, the lower the number of layers and vias the lower the cost. The use of one via type or the other has a direct impact on the number of layers required for routing. That is, limiting the number of staircase vias, the number of layers required for routing all

nets is greatly increased, and thus the number of stacked vias will also increase. However, in high-speed applications (> 10GHz) staircase vias may introduce a critical delay effect on the z-direction bends [0], and thus another limitation is introduced. Therefore, due to the large number of nets assigned different costs and the above mentioned conflicting effects between via types and layers, some means of optimizing the total cost is desired. This is the whole motivation behind proposing the new MCM routing strategy.

The proposed strategy restricts itself to two-terminal nets only. Also, for discussion purposes, completed nets may only consist of straight and single bend (90 degrees) connections.

5.3 Overall routing strategy.

The procedure of the proposed strategy consists of two major elements referred to as *constraint routing* and *compaction*. Constraint routing, the first phase, creates a finished but unminimized routing. The compaction phase, based on simulated annealing, completes the process by taking the results of the first phase and optimizing them according to the given routing costs of each net. Each of the two elements use ideas similar to previous research, although somewhat modified or changed and are brought together in a new way to solve the MCM routing problem.

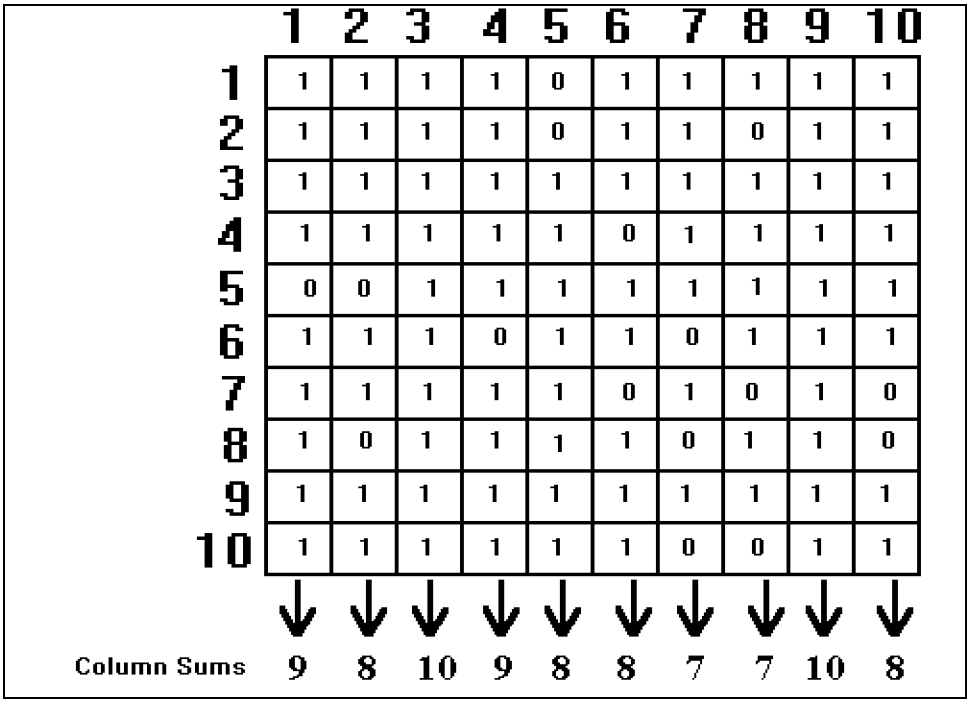
5.3.1 Constraint routing.

The constraint routing phase performs the task of single-layer routing. Single-layer routing involves routing as many entire nets on a single plane as possible. When no more nets can

be placed on the plane another layer must be used. Therefore, some means of maximizing the number of nets placed on a layer is desired in order to keep the layer cost down. The constraint routing strategy accomplishes this goal very practically.

The first step in constraint routing is to create a constraint table matrix based on the data structure in [0]. The matrix represents all the constraining relationships between every pair of nets. The matrix size is n x n, where n is the number of nets to be routed. Figure 4 shows the matrix representing the constraints in Figure 2, and Figure 5 gives the pseudocode for filling the matrix. Each number across the top and down the side of the matrix corresponds to a numbered net. A '1' inside the matrix means that it is possible to route net i (i is a row #) if net j (j is a column #) is routed. A '0' means that it is not possible to route net i if net j is routed.

The second step will determine the number of constraints for each net. Each column is summed, and the total is used to identify which nets have the least number of constraints. The sum of each column is used in the last step to determine where each net is routed. Figure 4 shows how this step is performed.



	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	0	1	1	1	1	1
2	1	1	1	1	0	1	1	0	1	1
3	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	0	1	1	1	1
5	0	0	1	1	1	1	1	1	1	1
6	1	1	1	0	1	1	0	1	1	1
7	1	1	1	1	1	0	1	0	1	0
8	1	0	1	1	1	1	0	1	1	0
9	1	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	0	0	1	1
Column Sums	9	8	10	9	8	8	7	7	10	8

Figure 2 – Constraint Matrix

```
for i = 1 to NUMBER_OF_PINS

{ for j = i to NUMBER_OF_PINS

  { if ( x_i1 < x_j1 AND x_i2 > x_j1 AND

         x_i1 < x_j2 AND x_i2 > x_j2 AND

         y_i1 > y_j1 AND y_i1 < y_j2     )

    { Matrix[i][j] = 0 }

    else
      Matrix[i][j] = 1
  }
}
```

Figure 5 - Pseudocode for Constructing Constraint Matrix

The final step will perform the actual routing of all the nets. The algorithm finds the next net with the least number of constraints (largest column sum) and tries to route that net. A net can be routed if it does not interfere (i.e. crossover) with an already routed net or pin. An array holds the list of nets

already routed and the layer they reside on. The constraint matrix is referred to when checking whether a net can be routed. When no more nets can be routed on a layer, a new layer is introduced. This process insures that the maximum number of nets are routed per layer using the single-layer routing technique. Wire placements will be described one of two ways depending on whether it's a straight or bent connection. Straight connections consist of two coordinates, the locations of the two pins. Bent connections consist of a third coordinate identifying where the wire bend is located. Once all nets are routed, the compaction phase takes control and optimizes the routing layout. Figure 6 shows the routing layout after the constraint routing phase.

5.3.2 Compaction operation.

The compaction phase performs optimization of the completed routing layout using layer costs and the weights assigned to each net. The assigned weights correspond to the desired maximum via count for each routed net. Many nets have high priority weights assigned to them that conflict with the priority weights of other nets, and thus some compromise will have to be made. Weights are used to supply cost-calculation numbers during the compaction phase. It also allows certain nets (i.e. time critical) to have higher circuit performance.

The compaction phase operates by using the simulated annealing (SA) technique to accomplish the optimization and compromise on the conflicting priority weights. Simulated annealing has been successfully employed for chip placement but

can also be used in routing since routing can also be thought of as a problem of placement optimization (i.e. placement of nets and vias within layers). The simulated annealing technique has also been used as a framework for controlling rip-up and rerouting transformations [0]. The compaction phase employs a very similar tactic and applies it to the MCM routing problem.

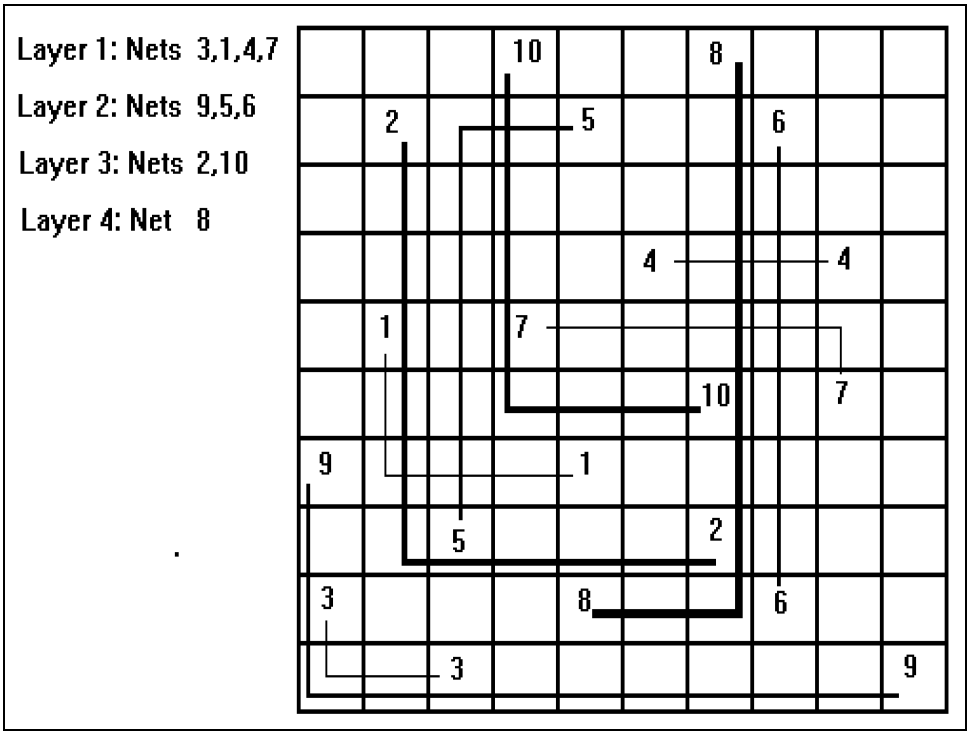


Figure 6 – Completed routing after constraint routing phase

5.3.2.1 The annealing process.

In order to proceed with the annealing process, we must establish a set of allowable moves that will permit us to change from one legal configuration to another. These moves follow the conventional operation of component interchange. For the defined MCM routing problem the annealing process can be thought of as a *rip-up and reroute* procedure. One or more nets are removed (rip-up) allowing another net or nets to be routed, and then the ripped-up nets are rerouted, resulting in a new routing configuration. The following interchanges can be performed:

1. Rip-up 1 net that allows another net to be rerouted on that layer and then reroute the first net on the same layer following a different path. Figure 7 shows an example.

2. Rip-up 1 net that allows another net to be rerouted on that layer and then reroute the first net on the previous layer of the second net.

3. Rip-up 1 net segment that allows 2 other nets to be rerouted on that layer and then reroute the first net segment on previous layer of rerouted nets. Figure 8 shows an example.

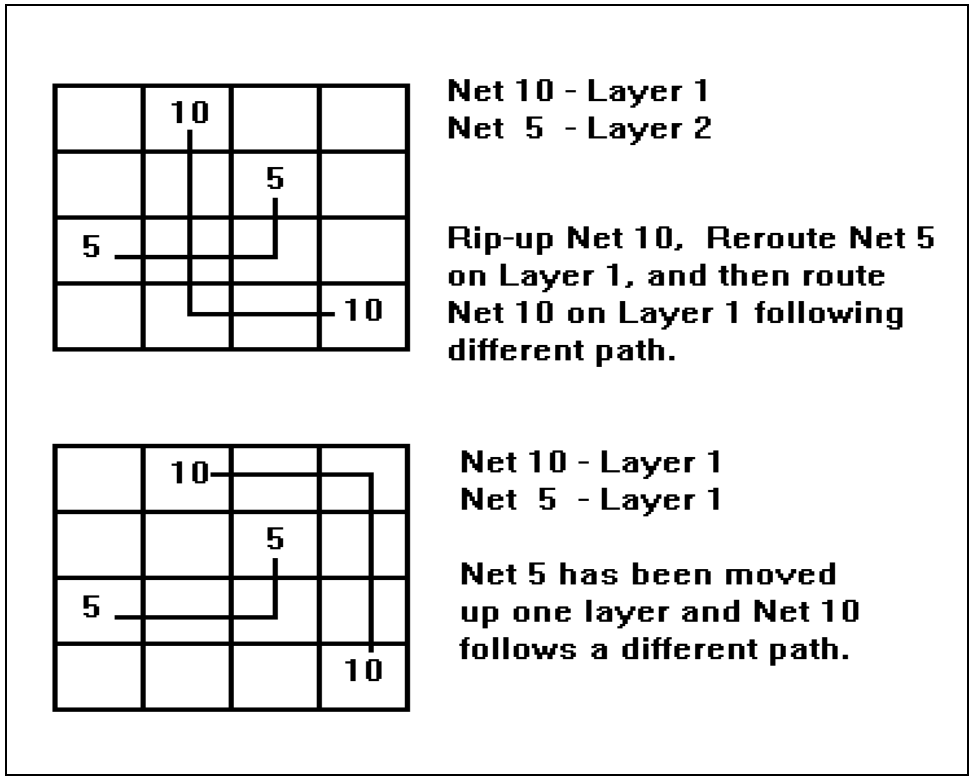

10
5
5
10
Net 10 - Layer 1
Net 5 - Layer 2
Rip-up Net 10, Reroute Net 5 on Layer 1, and then route Net 10 on Layer 1 following different path.
10
5
5
10
Net 10 - Layer 1
Net 5 - Layer 1
Net 5 has been moved up one layer and Net 10 follows a different path.

Figure 7

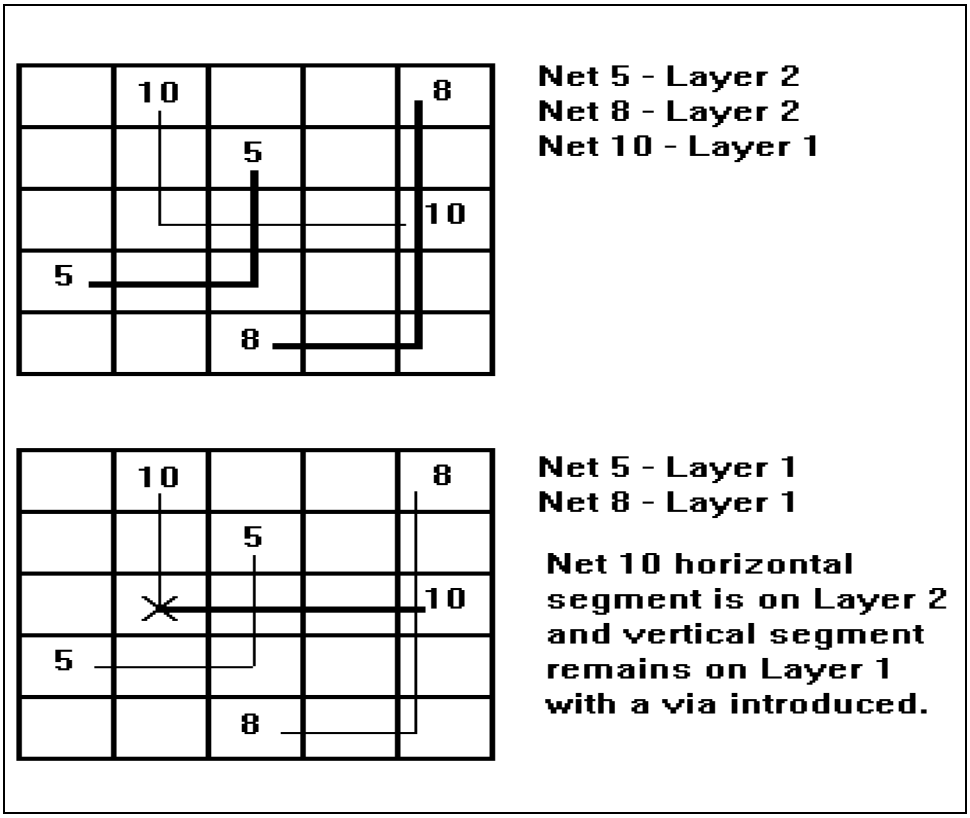

10
8
5
10
5
8
Net 5 - Layer 2
Net 8 - Layer 2
Net 10 - Layer 1
10
8
5
10
5
8
Net 5 - Layer 1
Net 8 - Layer 1
Net 10 horizontal segment is on Layer 2 and vertical segment remains on Layer 1 with a via introduced.

Figure 8

For each of the three specified interchanges, respectively,

the following changes in cost can occur:

1. Decrease cost - The total number of vias decreases by 2 and the second net's priority weight is decreased more.

2. Increase cost - The total number of layers and vias remains the same but the second net's priority weight is raised. This increase cost, however, may open a way to further improvements.

   Decrease cost - The total number of layers and vias remains the same but the second net's priority weight meets or falls under its maximum limit while the first net still meets or falls under its maximum limit.

3. Increase cost - The total number of stacked vias is increased by 2 x # of layers moved down if the two rerouted nets are placed on a lower layer. Total number of staircase vias is increased by 1.

   Decrease cost - The total number of stacked vias is decreased by 2 x # of layers moved up if the two rerouted nets are placed on a higher layer. A removal of a layer may also occur. Total number of staircase vias is increased by 1, but unless weight is high for staircase vias, there should still be a decreased cost.

The interchanges causing increased cost are allowable since it may allow the solution to escape a local minimum. The next question is how do we decide which interchanges should be performed when and in what order? Also, how do we control the progress of compaction? These questions are addressed by the *cooling schedule*.

5.3.2.2 The cooling schedule.

The cooling schedule is a very important task in the annealing process since it controls the improvement of the solution, the speed of obtaining a good solution, and when annealing should be terminated. The success of trying to improve the routing layout and perhaps the speed of obtaining an optimal solution depends on the order we implement the possible routing configuration changes stated previously. Due to the multitude of different MCM routing problems, it will be difficult to determine the order and frequency to implement the described configuration changes. Interchange possibilities can be done repetively in a specified order or certain interchanges can be phased in or out as time progresses. For example, we can initially make it highly probable that complete nets be rerouted and make it less probable that net segments be rerouted. As the annealing progresses, we can make it less probable that complete nets be rerouted and more likely that net segments be rerouted. These efforts are what are required in finding and implementing an optimum cooling schedule.

In addition to determining the order of the interchange rules, we must also determine which nets are to be used for

interchange. These determinations can be based on desired criteria as annealing progresses. The list of cost changes stated above can be referred to when deciding on this element of the cooling schedule. The following shows some suggested strategies that will lead to a routing improvement:

1. Choose nets that are on the lowest layers. Since the lower layers contain the fewest nets, this strategy makes it more likely that a layer will be deleted, as nets are rerouted on higher layers.

2. Choose nets to be rerouted that have no constraints with the ripped-up net, thus allowing an improvement. This information is stored in the constraint matrix. An additional constraint check will have to be made for ripped-up net segments.

3. Choose net segments from highest layers that when ripped-up will allow the most number of other nets to be rerouted on those higher layers, thus lowering the MCM routing cost.

In order for the algorithm to progress toward an optimum routing layout, the cooling schedule should allow the annealing process to continually improve the solution using these strategies and also allow for increased cost moves so that local minima may be escaped.

5.3.2.3 The convergence criterion.

To achieve some desired degree of routing optimization through simulated annealing, some type of convergence criterion must be defined. The convergence criterion should be defined such that it will result in a routing configuration that satisfies or at least compromises on the priority weights of all nets. Therefore, the convergence criterion is another important

item since it determines when a final solution will be obtained and the quality of that solution.

An approach for obtaining a convergence criterion for this particular MCM routing problem should deal with the tradeoffs between layers and vias as discussed earlier. The following shows two suggested approaches:

1. Use the MCM cost function for routing. When the total cost has reached the desired level, the annealing algorithm should terminate.

2. When a certain percentage of priority weights for all nets have been satisfied, the annealing algorithm should terminate.

5.4 Further work or improvements.

The proposed strategy to solve the MCM routing problem is open for further improvements and additions. First, the constraint routing phase can be complemented with an additional routing technique that might create a better initial routing configuration before compaction takes place. Also, relaxing the restriction on straight and single bend connections to allow multiple bends may contribute to further improvement. Second, additional interchanges may be used. These interchanges include ripping up and rerouting smaller net segments.

This routing strategy is a good candidate for being implemented with parallel processors. Several simulated annealing based algorithms have employed parallel processing as a means to achieve speed improvements. For this problem, each processor could be assigned a different set of layers to improve

the routing layout of and allow each processor to communicate moves with the other processors.

6.0 Conclusion.

A proposed strategy for solving the MCM routing problem has been presented. The strategy is very applicable in situations where the routing cost or quality needs to be precisely controlled. The strategy also guarantees a complete routing solution. Finally, it also shows how previous layout approaches may be used and manipulated to solve a particular objective of the MCM routing problem.

Alon, Amir, and Ascher, Uri, "Model and Solution Strategy for Placement of Rectangular Blocks in the Euclidean Plane," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 7, n. 3, March 1988, pp. 378-386.

Andrews, D. L, Glover, M. D., and Conrad, J. M., "Advanced Electronic Packaging: With Emphasis on Multi-Chip Modules."

Anonymous, "Timing-influenced Layout Design," *IBM Technical Disclosure Bulletin*, v. 28, n. 11, April 1986, pp. 4981-4988.

Aude, J. S., Lopes Filho, E. P., Martins, M. F., and Pinto, S. B., "ARCO: A cost-effective and flexible hardware maze router," *19th EUROMICRO Symposium on Microprocessing and Microprogramming*, Barcelona, Spain, 1993.

Berkcan, E., and Kinnen, E., "Optimal placement and network partitioning algorithm," *Conference Proceedings - 28th Midwest Symposium on Circuits and Systems*, Louisville, KY, August 19-20 1985.

Berkman, Omer, Jaja, Joseph, Krishnamurthy, Sridhar, Thurimella, Ramakrishna, and Vishkin, Uzi, "Top-bottom routing around a rectangle is as easy as computing prefix minima," *SIAM Journal on Computing*, v. 23, n. 3, June 1994, pp. 449-465.

Brown, A. D., "Automated Placement and routing," *Computer Aided Design*, v. 20, n. 1, Jan-Feb 1988, pp. 39-44.

Brown, A. D., and Zwolinski, M., "Lee router modified for global routing," *Computer Aided Design*, v. 22, n. 5, June 1990, pp. 296-300.

Brown, Andrew, "VLSI: Circuits and Systems in Silicon," McGraw-Hill Book Company, London, England, 1991.

Burstein, Michael, and Youssef, Mary N., "Timing influenced layout design," *Proceedings 1985 - 22nd ACM/IEEE Design Automation Conference*, Las Vegas, NV, June 23-26 1985.

Chan, Heming, Mazumder, P., and Shahookar, K., "Macro-cell and module placement by genetic adaptive search with bitmap-represented chromosome," *Integration, the VLSI Journal*, v. 12, n. 1, November 1991, pp. 49-77.

Chandrasekharam, R., "Genetic algorithm for node partitioning problem and applications in VLSI design," *IEEE Proceedings, Part E, Computers and Digital Techniques*, v. 140, September 1993, pp. 255-260.

Chao, Ting-Hai, Hsu, Yu-Chin, Ho, Jan-Ming, Boese, Kenneth D., and Kahng, Andrew B., "Zero skew clock routing with minimum wirelength," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, v. 39, n. 11, November, 1992, pp. 799-814.

Chen, Chun-hong, and Liu, Mei-lun. "Application of bin-packing to building block placement," *1987 IEEE International Symposium on Circuits and Systems*, Philadelphia, PA, May 7 1987.

Cheng, Gui-Xin, Tanaka, Mamoru, and Yamada, Minoru, "A parallel routing technique based on local current comparison," *1991 IEEE International Symposium on Circuits and Systems*, Singapore, Singapore, June 11-14, 1991.

Cho, H. G., and Kyung, C. M., "O(n)-time standard cell placement algorithm using constrained multi-stage graph model," Espoo, Finland, June 7-9 1988.

Cho, Jun Dong, Liao, Kuo-Feng, Raje, Salil, and Sarrafzadeh, Majid, "M^{2}R: Multilayer Routing Algorithm for High-Performance MCMs," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 41, no. 4, April 1994.

Cohoon, James P, Paris, William D., "Genetic placement," *IEEE International Conference on Computer-Aided Design*, Santa Clara, CA, November 11-13, 1986.

Cohoon, James P., "Distributed genetic algorithms for the floorplan design problem," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 10, April 1991, pp. 483-492.

Feltham, Derek, Khare, Jitendra, and Maly, Wojciech, "Design for testability view on placement and routing," *European Design Automation Conference*, Hamburg, Germany, September 7-10 1992.

Fujita, Tomoyuki, and Kuh, Ernest S., "New detailed routing algorithm for convex rectilinear space," *IEEE International Conference on Computer-Aided Design*, Santa Clara, CA, November 12-15, 1984.

Funabiki, Nobuo, and Takefuji, Yoshiyasu, "Parallel multi-layer channel router on the HVH model," *Parallel Computing*, v. 19, n. 1, January 1993, pp. 63-77.

Gonzalez, Teofilo F., and Kurki-Gowdara, Shashishekhar, "Approximation algorithm for the via placement problem," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 8, n. 3, March 1989, pp. 219-228.

Grover, Lov K., "New Simulated Annealing Algorithm for Standard Cell Placement," *IEEE International Conference on Computer-Aided Design*, Santa Clara, CA, November 11-13 1986.

H. Moreno, "OSS physical design cad tools specifications," *MCC Technical Report P/I-250-90*, 1990.

Hartoog, Mark R., "Analysis of Placement Procedures for VLSI Standard Cell Layout," *23rd ACM/IEEE Design Automation Conference - Proceedings 1986*, Las Vegas, NV, June 29 - July 2 1986.

Ho, Jan Ming, Sarrafzadeh, Majid, Vijayan, Gopalakrishnan, and Wong, C.K., "Layer Assignment for Multichip Modules," *IEEE Transactions on Computer-Aided Design*, vol. 9, no. 12, December 1990.

Hsu, Yu-Chin, and Kubitz, William J., "Automatic placement using wavefront compaction," *Proceedings - IEEE International Conference on Computer Design: VLSI in Computers*, Port Chester, NY, October 7-10 1985.

Hu, T. C., and Shing, M. T., "Decomposition algorithm for circuit routing," *Mathematical Programming Study*, n. 24, October 1985, pp. 87-103.

Jeen, Jzan-Ching, Gyurcsik, Ronald S., and Liu, Wen-Tai, "Two-layer channel routing algorithm for mixed analog and digital signal nets," *Proceedings of the IEEE 1988 Custom Integrated Circuits Conference*, Rochester, NY, May 16-19, 1988.

Forrest, Stephanie, "What makes a problem hard for a genetic algorithm? Some anomalous results and their explanation," *Machine Learning*, v. 13, November/December 1993, pp. 285-319.

Johns, J. F., and Hachtel, G. D., "Zone expansion algorithm for gridless routing on a continuous plane," *IEEE International Symposium on Circuits and Systems*, San Jose, CA, May 5-7, 1986.

Joobbani, Rostam, "An Artificial Intelligence Approach to VLSI Routing," Kluwer Academic Publishers, Norwell, MA, 1986.

Khoo, Kei-Yong, and Cong, Jason, "A Fast Multilayer General Area Router for MCM Designs," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 39, no. 11, November 1992.

Kim, Sung-Soo, and Kyung, Chong-Min, "Circuit placement in arbitrarily-shaped region using self-organization," *IEEE International Symposium on Circuits and Systems 1989*, Portland, OR, May 8-11 1989.

Kitazawa, Hitoshi, and Ueda, Kazuhiro, "Look-ahead line search algorithm with high wireability for custom VLSI design," *International Symposium on Circuits and Systems*, Kyoto, Japan, June 5-7, 1985.

Kumar, C. P., Ravi, and Sastry, Sarma, "Parallel placement on reduced array architecture," *25th ACM/IEEE Design Automation Conference*, Anaheim, CA, June 12-15 1988.

Kumar, H., Kalyan, R., Bayoumi, M., Tyagi, A., and Ling, N., "Parallel implementation of a cut and paste maze routing algorithm," *1993 IEEE International Symposium on Circuits and Systems*, Chicago, IL.

Kyung, C. M., Lee, P. H., Yang, Y. Y., and Park, I. C., "Efficient Algorithm for Two- and Three-Dimensional IC Floor Planning", *International Journal of Circuit Theory and Applications*, v. 16, n. 4, Oct 1988, pp. 425-445.

Lee, C. Y., "An algorithm for path connections and its applications," *IRE Transactions on Electronic Computers*, September 1961, pp. 346-365.

Leu, M. C., Wong, H., and Ji, Z., "Planning of component placement/insertion sequence and feeder setup in PCB assembly using genetic algorithm," *Journal of Electronic Packaging, Transactions of the ASME*, v. 115, n. 4, December 1993, pp. 424-432.

Lin, Youn-Long, Hsu, Yu-Chin, and Tsai, Fur-Shing, "Hybrid routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 9, n. 2, February 1990, pp. 151-157.

Mathew, J.M., "Parallel Search Algorithms for Solving Constraint Satisfaction Problems," Master's Thesis, Dept. of Comp. Sys. Eng., Univ. of Arkansas, Fayetteville, AR, Dec. 1993.

Miriyala, S., Hashmi, J., and Sherwani, N., "Switchbox Steiner tree problem in presence of obstacles," *1991 IEEE International Conference on Computer-Aided Design*, Santa Clara, CA, November 11-14, 1991.

Mirzaian, Andranik, "River routing in VLSI," *Journal of Computer and System Sciences*, v. 34, n. 1, February 1987,

pp. 43-57.

Moosa, Zahir, Brown, Mike, and Edwards, Douglas, "Application of simulated annealing to maze routing," *Proceedings of the 1994 European Design Automation Conference*, Grenoble, France, 1994.

Odawara, Gotaro, Iijima, Kazuhiko, and Wakabayashi, Kazutoshi, "Knowledge-based placement technique," *Printed Circuit Design*, v. 3, n. 1, January 1986, pp. 20-27.

Patel, A. M., "Wirability Placement Algorithm for Hierarchical VLSI Layout," *Proceedings - IEEE International Conference on Computer Design: VLSI in Computers*, Port Chester, NY, October 8-11 1984.

Patel, Ash M., Soong, Norman L., and Korn, Robert K., "Hierarchical VLSI routing - an approximate routing procedure," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. CAD-4, n. 2, 1985, pp. 121-126.

Razaz, M., and Gan, J., "Novel placement method for VLSI design," *1987 Proceedings - Fourth International IEEE VLSI Multilevel Interconnection Conference*, Santa Clara, CA, June 15-16 1987.

Rose, Jonathan S., Snelgrove, W. Martin, and Vranesic, Zvonko G., "Parallel Standard Cell Placement Algorithms with Quality Equivalent to Simulated Annealing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 7, n. 3, Mar 1988, pp. 387-396.

Rutenbar, Rob A., "Simulated annealing algorithms: An overview," *IEEE Circuits and Devices Magazine*, v. 5, n. 1, January 1989, pp. 19-26.

Sargent, Jeff S., and Banerjee, Prith, "Parallel row-based algorithm for standard cell placement with integrated error control," *26th ACM/IEEE Design Automation Conference*, Las Vegas, NV, June 25-29 1989.

Sastry, L. V. P., and Zargham, Mehdi R., "Macro-cell placement in a multiprocessor environment," *Energy and Information Technologies in the Southeast*, Columbia, SC, April 9-12 1989.

Scheible, Jurgen, and Mlynski, Dieter A., "A high density placement algorithm based on simulated surface tension," *1991 IEEE International Symposium on Circuits and Systems*, Singapore, Singapore, June 11-14 1991.

Schiele, W. L., Kruger, T., Just, K. M., and Kirsch, F. H., "A gridless router for industrial design rules," *27th ACM/IEEE Design Automation Conference*, Orlando, FL, June 24-28, 1990.

Sengupta, Maitreya, Lipa, Steve, Franzon, Paul, and Steer, Michael, "Crosstalk driven routing advice," *Proceedings of the 1994 IEEE 44th Electronic Components & Technology Conference*, Washington, DC.

Sengupta, Maitreya, Lipa, Steve, Franzon, Paul, and Steer, Michael, "Crosstalk driven routing advice," *Proceedings of the 1994 IEEE 44th Electronic Components & Technology Conference*, Washington, DC.

Shahookar, Khushro, and Mazumder, Pinaki, "A Genetic Approach to Standard Cell Placement Using Meta-Genetic Parameter Optimization," *IEEE Transactions on Computer-Aided Design*, vol. 9, no. 5, May 1990, pp. 500-512.

Sone, Tadashi, and Nakabayashi, Kiyoshi, "Floating Track Method for Complete Routing," *Electronic Communication in Japan Part 2*, v. 69, n. 8, 1986, pp. 20-29.

Sreenivasa Rao, D., and Patnaik, L. M., "Neural Network Based Approach to Standard Cell Placement," *Electronics Letters*, v. 25, n. 3, Feb 2 1989, pp. 208-209.

Sriram, M., and Kang, S. M., "Performance driven MCM routing using a second order RLC tree delay model," *Proceedings of the 5th Annual IEEE International Conference on Wafer Scale Integration*, San Francisco, CA.

Storer, J. A., Nicas, A. J., and Becker, J., "Uniform Circuit Placement," *VLSI: Algorithms and Architectures, Proceedings of the International Workshop on Parallel Computing and VLSI*, Amalfi, Italy, May 23-25 1984.

Sugai, Yasuo, and Hirata, Hironori, "Hierarchical algorithm for a partition problem using simulated annealing: Application to placement in VLSI layout," *International Journal of Systems Science*, v. 22, n. 12, Dec 1991, pp. 2471--2487.

Sutanthavibul, Suphachai, and Shragowitz, Eugene, "An adaptive timing-driven layout for high speed VLSI," *27th ACM/IEEE Design Automation Conference*, Orlando, FL, June 24-28 1990.

Sutanthavibul, Suphachai, Shragowitz, Eugene, and Lin, Rung-Bin, "Adaptive timing-driven placement for high performance VLSI's," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 12, n. 10, October 1993, pp. 1488-1498.

Sutanthavibul, Suphachai, Youssef, Habib, and Shragowitz, Eugene, "Cell-based physical design under timing constraints," *1990 IEEE International Symposium on Circuits and Systems*, New Orleans, LA, May 1-3 1990.

Takefuji, Yoshiyasu, and Lee, Kuo-Chun, "Parallel algorithm for tiling problems," *IEEE Transactions on Neural Networks*, v. 1, n. 1, March 1990, pp. 143-145.

Tamminen, M, Luk, W. K., Sipala, P., Woo, L. S., and Wong, C. K., "Constructing Maximal Slicings from Geometry," *Acta Informatica*, v. 23, n. 3, June 1986, pp. 267-288.

Terai, Masayuki, Nakajima, Kazuo, Takahashi, Kazuhiro, and Sato, Koji, "New approach to over-the-cell channel routing with three layers," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 13, n. 2, February 1994, pp. 187-200.

Terai, Masayuki, Takahashi, Kazuhiro, and Sato, Koji, "A new min-cut placement algorithm for timing assurance layout design meeting net length constraint," *27th ACM/IEEE Design Automation Conference*, Orlando, FL, June 24-28, 1990.

Tsay, Ren-Song, and Kuh, Ernest, "A unified approach to partitioning and placement," *IEEE Transactions on Circuits and Systems*, v. 38, n. 5, May 1991, pp. 521-533.

Tsay, Ren-Song, and Kuh, Ernest, "Module placement for large chips based on sparse linear equations," *International Journal of Circuit Theory and Applications*, v. 16, n. 4, October 1988, pp. 411-423.

Ueda, Kazuhiro, Kitazawa, Hitoshi, and Harada, Ikuo, "CHAMP: Chip Floor Plan for Hierarchical VLSI Layout Design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. CAD-4, n. 1, January 1985, pp. 12-22.

Wawryn, Krzysztof, "Layout including parasitics for printed circuit boards," *International Journal of Circuit Theory and Applications*, v. 16, n. 2, April 1988, pp. 107-128.

Vai, Man-Kuan, and Shanblatt, Michael A., "Improved Macrocell Placement Algorithm using Simulated Annealing," *Proceedings*

of the IEEE 1987 Custom Integrated Circuits Conference, Portland, OR, May 4-7 1987.

Walsh, P. A., and D. M. Miller, "Goal-directed simulated annealing and simulated sintering," *MicroElectronics Journal*, v. 25, n. 5, August 1994, pp. 363-382.

Wong, H., and Leu, M. C., "Adaptive genetic algorithm for optimal printed circuit board assembly planning," *CIRP Annals*, v. 42, n. 1, 1993, pp. 17-20.

Yao, Xianjin, Yamada, Masaaki, and Liu, C. L., "New approach to the pin assignment problem," *IEEE Transactions on Computer- Aided Design of Integrated Circuits and Systems*, v. 8, n. 9, September 1989, pp. 999-1006.

Yao, Xianjin, Yamada, Masaki, and Liu, C. L., "New approach to the pin assignment problem," *25th ACM/IEEE Design Automation Conference*, Anaheim, CA, June 12-15 1988.

Zhang, Chen-Xiong, and Mlynski, Dieter A., "Neural somatotopical mapping for VLSI placement optimization," *1991 IEEE International Joint Conference on Neural Networks*, Singapore, Singapore, November 18-21 1991.

Zheng, Si-Qing, Lim, Joon Shik, and Iyengar, Sitharama, "Efficient maze-running and line-search algorithms for VLSI layout," *Proceedings of the IEEE Southeastcon*, Charlotte, NC, 1993.

Kirkpatrick, S., Gelatt, Jr., C. D., and Vecchi, M. P., "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, May 13, 1983, pp. 671-680.