INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

MCG – A MUTILAYER GENERAL AREA MCM ROUTING ALGORITHM

by

Donghui Li

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

1 9 9 5

THE UNIVERSITY OF ARIZONA ®
GRADUATE COLLEGE

As members of the Final Examination Committee, we certify that we have read the dissertation prepared by Donghui Li

entitled MCG - A Multilayer General Area MCM Routing Algorithm

and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy

Jo Dale Carothers	11/17/95
Jo Dale Carothers	Date
Fredrick J. Hill	17 Nov 1995
Fredrick J. Hill	Date
Hal S. Tharp	11/17/95
Hal S. Tharp	Date
	Date
	Date

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copy of the dissertation to the Graduate College.

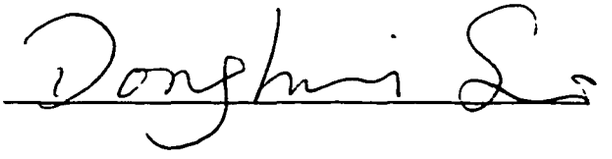
I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

Jo Dale Carothers	11/17/95
Dissertation Director	Date
Jo Dale Carothers	

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under the rules of the Library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of the source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the copyright holder.

SIGNED: 

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor, Dr. Jo Dale Carothers, for the suggestion of this research topic and for her guidance and support throughout this project. Thanks are also extended to Dr. Fredrick J. Hill, Dr. Hal S. Tharp, Dr. Richard D. Schlichting and Dr. Mary L. Bailey for their participation on the committee and for their helpful comments. My fellow graduate students, Mr. Tom Hameenanttila, Mr. Kusnadi, Ms. Carol Mackey and Mr. Richard Mackey, deserve many thanks for their friendship and help during my graduate studies. I would like to thank the members of my family, who have always supported my education with their love, encouragement, patience, and support.

To my wife, Hong Zou,
my parents, Chuanzhou Li and Jingxiu Ren,
whose love and encouragement
made this possible

TABLE OF CONTENTS

LIST OF FIGURES 9

LIST OF TABLES 10

ABSTRACT 13

1 INTRODUCTION 14

2 BACKGROUND 18

2.1 3-D Maze Router 19

2.2 M^2R Router 19

2.3 SLICE Router 20

2.4 V4R Router 21

2.5 V4CMRA Router 22

2.6 SURF Router 22

2.7 Layer Assignment Before Routing 23

2.8 Summary 23

3 THE MCG ROUTING ALGORITHM 25

3.1 Problem Formulation 25

3.2 Overview of the MCG Routing Algorithm 26

3.3 The Candidate Routes 27

3.3.1 Type-0 Routes 28

3.3.2 Type-1 Routes 29

3.3.3 Type-2 Routes 31

3.3.4 Type-3 Routes 32

3.3.5 Type-4 Routes 34

3.3.6 Type-5 Routes 35

3.3.7 Type-X Routes 37

3.3.8 Type-Z Routes 39
3.4 Multi-Terminal Nets 42
3.5 The Compatibility Graph and Its Reduction 42
3.6 X-Y Layer Pair Routing 48
3.7 Single-Layer Routing 52
3.8 Post-Routing Processing 54
3.8.1 Jog Removal 55
3.8.2 Steiner Point Insertion 55
3.9 Summary 56

4 EXPERIMENTAL RESULTS AND ANALYSIS 57
4.1 Test Examples 57
4.1.1 MCC Benchmarks 57
4.1.2 Test Data of M^2R 60
4.1.3 Flip-Chip Example 61
4.2 Test Results for X-Y Layer Pair Routing 61
4.2.1 Test Results for MCC1 62
4.2.2 Test Results for MCC2-75 67
4.2.3 Test Results for MCC2-45 72
4.2.4 Test Result for the Flip-Chip Example 74
4.2.5 Comparison with Other Routers 76
4.3 Test Results for Single-Layer Routing 80
4.4 Some Implementation Issues 81
4.4.1 The Data Structure Representing the Compatibility Graph . 81
4.4.2 The Storage Organization for Obstacles 82
4.5 Complexity Analysis 82
4.5.1 The Space Complexity 83
4.5.2 The Time Complexity 83
4.6 Selection of the Number of Candidate Routes 85

5 LAYER BALANCING AND RE-ROUTING 93

5.1 Layer Balancing 93
5.2 Re-Routing 98
5.3 Test Results of Layer Balancing 99
5.4 Test Results of Re-Routing 101

6 CONCLUSION 102

REFERENCES 104

LIST OF FIGURES

3.1 Routing Grid and Obstacles 28
3.2 Type-0 Route 29
3.3 Type-1 Route 30
3.4 Type-2 Route and Its Construction 32
3.5 Type-3 Route and Its Construction 33
3.6 Type-4 Route and Its Construction 35
3.7 Type-5 Route and Its Construction 36
3.8 Examples of Type-X Routes 37
3.9 A Net Routed by Line-Probe Algorithm 41
3.10 Type 1 Jog and its Removal 54
3.11 Type 2 Jog and its Removal 55
3.12 An Example of Steiner Point Insertion 56

4.1 Run Time vs. Number of Candidates(M^2R-100) 87
4.2 Run Time vs. Number of Candidates(M^2R-200) 87
4.3 Percentage Routed vs. Number of Candidates 88
4.4 Run Time vs. Number of Candidates(MCC1) 89
4.5 Run Time vs. Number of Candidates(MCC2) 91

LIST OF TABLES

4.1 MCC1 Net Count 58
4.2 MCC2 Net Count 59
4.3 Characteristics of the MCC Benchmarks 59
4.4 Test Examples of M^2R 60
4.5 M^2R Net Count 60
4.6 Routing Results for MCC1(w/o P&G) 62
4.7 Test Results of Jog Removal on MCC1(w/o P&G) 63
4.8 Test Results of Steiner Point Insertion on MCC1(w/o P&G) 63
4.9 Percent Completed Routes After x Layers 64
4.10 Via Usage of Test Results for MCC1(w/o P&G) 64
4.11 Final Results for MCC1(w/o P&G) 65
4.12 Routing Results for MCC1(P&G) 65
4.13 Test Results of Jog Removal on MCC1(P&G) 65
4.14 Test Results of Steiner Point Insertion on MCC1(P&G) 66
4.15 Percent Completed Routes After x Layers 66
4.16 Via Usage of Test Results for MCC1(P&G) 66
4.17 Final Results for MCC1(P&G) and Comparison 67
4.18 Routing Results for MCC2-75(6) 68
4.19 Test Results of Jog Removal on MCC2-75(6) 68
4.20 Test Results of Steiner Point Insertion on MCC2-75(6) 68
4.21 Percent Completed Routes After x Layers 69
4.22 Via Usage of Test Results for MCC2-75(6) 69
4.23 Final Results for MCC2-75(6) and Comparison 69
4.24 Routing Results for MCC2-75(4) 70
4.25 Test Results of Jog Removal on MCC2-75(4) 70
4.26 Test Results of Steiner Point Insertion on MCC2-75(4) 71
4.27 Percent Completed Routes After x Layers 71

4.28 Via Usage of Test Results for MCC2-75(4) 71
4.29 Final Results for MCC2-75(4) and Comparison 72
4.30 Routing Results for MCC2-45 72
4.31 Test Results of Jog Removal on MCC2-45 73
4.32 Test Results of Steiner Point Insertion on MCC2-45 73
4.33 Percent Completed Routes After x Layers 73
4.34 Via Usage of Test Results for MCC2-45 73
4.35 Final Results for MCC2-45 and Comparison 74
4.36 Routing Results for FLIP 75
4.37 Test Results of Jog Removal on FLIP 75
4.38 Test Results of Steiner Point Insertion on FLIP 75
4.39 Percent Completed Routes After x Layers 75
4.40 Via Usage of Test Results for FLIP 76
4.41 Final results for FLIP 76
4.42 Comparison with Other Routers on Number of Layers 77
4.43 Comparison with Other Routers on Total Number of Vias 77
4.44 Comparison with Other Routers on Total Wire Length 78
4.45 Comparison with Other Routers on Ratio to Lower Bound 78
4.46 Percentage Routed on First Layer Pair 79
4.47 Platforms On Which the Routers Were Implemented 79
4.48 Comparison with Other Routers on Run Time 79
4.49 Comparison on Single-Layer Routing 80
4.50 Test Results of Different Number of Candidates for M^{2}R-100 85
4.51 Test Results of Different Number of Candidates for M^{2}R-200 86
4.52 Test Results of Different Number of Candidates for MCC1 89
4.53 Test Results of Different Number of Candidates for MCC2 90

5.1 Test Results of Layer Balancing on MCC1 100
5.2 Test Results of Layer Balancing on MCC2 100
5.3 Comparison with and without Layer Balancing 100

5.4 Test Results of Re-routing on MCC1 101
5.5 Test Results of Re-routing on MCC2 101

ABSTRACT

A new multilayer, general-area, multichip module(MCM) routing algorithm, called MCG, is introduced. The algorithm differs from other MCM routers in the way that the routes interconnecting the nets are constructed. Some routers perform the routing net-by-net and others extend the routes piece by piece during the routing process. The MCG router takes a more global approach by constructing a small number of candidate routes for each net, building a compatibility graph for the candidate routes and reducing the graph to yield a routing solution. Due to this unique way of performing the routing, the MCG router offers several outstanding features. First, it performs the routing of the nets simultaneously. Therefore it is not subject to the net ordering problem. Second, it can give the designer the flexibility of selecting the topology of routes for the nets to be routed, which is almost impossible for other algorithms. Third, it offers a natural way to incorporate the electrical constraints into the routing process, which is absent or hard to handle in other algorithms. The timing constraint can be incorporated easily by constructing the candidate routes of the net according to the requirements of the design; and the compatibility test of the candidate routes can be used to estimate the crosstalk between the routes from different nets, which make the crosstalk estimation more sophisticated than the other techniques. Compared with other MCM routers, the MCG router produces better quality routing solutions in terms of number of layers, number of vias, total wire length and routing density. In addition to the application in the area of MCM routing, the MCG algorithm can also be applied to high density PCB routing problems.

CHAPTER 1

INTRODUCTION

The integrated circuit (IC) has evolved from small scale integration (SSI) in the 1960's to today's very large scale integration (VLSI). The number of transistors on a single chip has grown from 30 to more than 3 million. The performance of the IC's has been increasing at an exponential rate together with the decreasing of their cost. All of these rapid improvements in integration technology have been made possible by the improvements of the various techniques involved in design and fabrication of VLSI chips.

However, the fast speed of improvement in performance of the IC's has not been matched by the packaging and interconnection technologies. In conventional packaging technology, bare dies are packaged into plastic or ceramic packages. These packages are then mounted onto printed circuit boards (PCBs). The interconnection between chips is made on the boards. It has been shown that the interconnect density on PCBs is at least one order of magnitude lower than the interconnect density at the chip level. Therefore, the interconnection length between the chips is long, which means long propagation delays. For high performance systems, long propagation delays are unacceptable.

A higher performance packaging and interconnection approach is needed to meet the requirements of high performance systems. One alternative approach is the multichip module (MCM) technology, which eliminates the single chip package by mounting the bare chips (dies) directly onto the routing substrate. The interconnections between the chips are routed on this common substrate. The pitch size of the substrate in MCM technology is much smaller than that of PCBs, so a much higher density of interconnection wires can be achieved. Since the chips typically occupy only one tenth of the area of single chip packages, they can be easily placed closer together in an MCM. This provides for both higher density assemblies as

well as shorter and faster interconnects.

The MCM technology offers advantages over the conventional PCB technology in the following ways:

1. System size and weight are greatly reduced due to the elimination of the single-chip package.

2. Multichip packaging contributes to reduced interchip signal delays because of the much shorter length of the interchip connection, which helps to increase the clock speeds of the system, hence improve the performance of the system.

3. MCMs can provide reduced power consumption. Typically, a large portion of the total power consumption of a chip is due to the off-chip drivers. These drivers are typically large, since they are required to charge or discharge the large capacitive loads presented by interchip connections. The sizes of off-chip drivers can be reduced, since the loads to be driven are smaller in MCMs. These modifications at the chip level can lead to considerable reduction in the overall power consumption.

While MCMs provide a number of significant advantages over conventional packaging, they also introduce some difficulties in design, manufacturing and testing. Physical design is a very significant part of the system design process. Broadly speaking, physical design is the process of laying out the components of a system and establishing electrical connections between them, subject to various physical, electrical and thermal constraints.

VLSI and PCB physical design algorithms have been a very important field of research in the past 20-30 years. These algorithms have contributed significantly to the rapid improvements of VLSI chips and the systems.

Even though the steps in the physical design cycle of MCMs are similar to those in the PCB and IC design cycle, the design tools for PCBs and ICs cannot be directly used for MCMs. This is mainly due to the fact that MCM layout problems are different from IC layout or PCB layout problems. The existing PCB design tools cannot handle the dense and complex wiring of MCMs. On the other

hand, IC layout tools are inadequate to solve the MCM layout problems due to the fact that they have different geometrical and electrical constraints. As a result, the lack of CAD tools for MCMs is hindering further development. In order to maximize the benefit of using MCMs, CAD tools developed specifically for MCMs are needed. In fact, a number of CAD vendors and researchers at universities are now involved in developing MCM-specific approaches and design tools. Some progress has been achieved in the area.

The physical design cycle of an MCM has been divided into three steps:

1. Partitioning: The first task is to partition the given circuit into subcircuits, such that each subcircuit can be fabricated on a single chip and the number of subcircuits is less than or equal to the number of chips that the MCM can carry.

2. Placement: The placement step is concerned with mapping the chips to the chip sites on the MCM substrate. The placement affects the efficiency of the next step – routing.

3. Routing: After the chips have been placed on the chip sites, the next phase of the MCM physical design process is to connect these chips as specified by the net list. The objective of minimizing routing area in IC design is no longer valid in the MCM routing environment. Instead, the objective of MCM routing is to minimize the number of layers, because the cost of an MCM depends on the number of layers used. Due to the fact that the interconnection wires involved in MCMs are much longer than that of IC's, crosstalk between the wires becomes an important consideration which is not of as much concern in IC layout. The MCM routing problem can be viewed as a truly three-dimensional problem because the number of layers used for routing can range from only a few to as many as several tens.

The goal of this research was to develop a new algorithm for the MCM routing problem and provide the mechanism for the incorporation of the electrical constraints into the routing process. The new routing algorithm named MCG (Multi-candidate Compatibility Graph) has been shown to be able to produce high quality routing solutions for some of the very challenging MCM benchmarks, and it

has the ability to offer several outstanding features such as the flexibility to choose topologies for the nets and a natural way to incorporate the electrical constraints into the routing process, which are crucial to the success of the routing algorithm. And this work can also be applied to high density PCB routing problems.

Prior routing algorithms proposed for MCMs are discussed in the next chapter. The details of the new routing algorithm are described in Chapter 3. Chapter 4 presents the test results of the algorithm on a number of MCM examples including the industrial benchmarks from MCC (Microelectronics and Computer Technology Corporation). One of the benchmarks contains 37 VHSIC gate arrays, over 7000 nets, and over 14000 pins. The test results have shown that the newly developed algorithm can produce better routing solutions than the other algorithms. A new layer balancing algorithm which distributes the nets uniformly among the layers is introduced in Chapter 5. Finally, the conclusion and discussions of future work are presented in Chapter 6.

CHAPTER 2

BACKGROUND

Unlike the VLSI routing environment where the routing area is naturally decomposed into channels and switchboxes, the MCM routing environment consists of the entire multilayer substrate beneath the chips with the exception of some obstacles such as thermal vias, distribution vias, etc., which make the MCM routing problem truly three-dimensional. The objective is minimization of the total number of layers since the cost is directly related to the number of layers used. Also minimization of the total number of vias and the total wire length is required. In addition, electrical constraints such as crosstalk effect, which are usually ignored in VLSI routing, must be taken into consideration because of the length of interconnects in MCMs. In general, the VLSI routing tools are inadequate for MCM routing. Meanwhile, most existing PCB routing tools cannot handle the MCM routing problem because of the higher packaging density of MCMs. As a result of the above, there is a need for new routing algorithms to meet the specific requirements of the MCM routing problem.

The overall quality of a routing solution can be measured in terms of the number of layers, the number of vias, the total wire-length, and the satisfaction of electrical constraints such as timing constraints and crosstalk avoidance. The number of layers, the number of vias and the wire-length affect the cost and performance of the package. Vias tend to degrade the performance of the system since they introduce discontinuities. In addition, long interconnects cause delay problems which may result in violation of the timing constraints. The delay model of the transmission line can be used to estimate the time-domain response of the interconnects. The crosstalk effect can be addressed by limiting the parallel length of two signal wires which run next to each other in the same layer or in adjacent layers.

In the following sections, we discuss previously proposed algorithms including

the 3-D Maze router, the M^2R router, the SLICE router, the V4R router, the V4CMRA router, the SURF router and the layer assignment before routing approach. Their characteristics, merits and drawbacks are discussed.

2.1 3-D Maze Router

The Maze routing algorithm was first introduced by Lee in 1961 [28]. The algorithm can be used to find an optimal path between two points in a rectangular grid while avoiding the obstacles along the way. It is basically a breadth first search algorithm. It has been widely used in the global routing phase in single-chip VLSI routing.

The 3-D Maze routing algorithm [18] extends the Maze routing algorithm from planar space to three dimensional space to fit the MCM routing environment. It is relatively easy to implement and commonly used by commercial routing tools. However, it suffers from several problems. First, the algorithm routes net-by-net, so the quality of the whole routing solution is subject to the ordering of the nets, and there is no known effective algorithm for determining a good ordering for the nets. Second, the computation time is long due to the need for searching all of the available grid points and it has large memory requirements to store the information about the entire 3-D routing grid. Third, it tends to use a large number of vias because some nets are routed using many different layers. Finally, the methodology of routing net-by-net makes global optimization difficult, if not impossible.

2.2 M^2R Router

M^2R was developed by Cho, et al. [6] [35]. It first uses a few layers to uniformly redistribute the closely positioned pins to relieve the congestion around the chips. It then routes the signal nets in additional layers using a combination of single-layer and x-y routing. The single-layer routing approach places the wires for the entire

route for the net on one routing plane; the x-y routing approach routes the nets on two adjacent layers at a time, with one layer used for horizontal wires and the other used for vertical wires. The critical nets are routed on single-layers and the less critical nets are then routed on x-y layer pairs. The routing is performed on a net-by-net basis and the order of the nets is determined by a linear combination of the estimated congestion, net length and priority. The routing technique used is called the density algorithm. The basic motivation of the density algorithm is to find the shortest path for the net, while avoiding routing it through congested areas by forcibly introducing a detour in the path.

In cases where the the spacing of the pins of the chips is different from the spacing of the grid points in the signal distribution layers, pin redistribution can be used to match the pin and grid spacing. While in other cases the pin redistribution may help to reduce the number of layers and number of vias. The routing algorithm of M^2R is still a version of maze routing. Hence, it is subject to the problems of maze routing.

2.3 SLICE Router

SLICE was developed by Khoo and Cong [26]. The SLICE router performs the routing on a layer by layer basis, with primarily planar (single-layer) routing in each layer. In the planar routing process, SLICE tries to connect as many nets as possible in the layer. It scans the whole routing region column by column from left to right. In each adjacent column pair, the nets which have one terminal on the left hand side and the other one on the right hand side of the current column pair is considered to assign a path which extends from the left column to the right column on the condition that paths from different nets do not cross each other. It calculates a maximum weighted non-crossing matching to select paths among all the possible paths for the nets under consideration to maximize the number of connections. The planar routing process is repeated until the rightmost column on

the layer is reached. After routing on one layer is finished, the nets which cannot be routed in the current layer are brought down to the next layer. This process repeats until all the nets are routed.

SLICE is 6 times faster and uses 29% fewer vias, compared with the 3-D Maze router on the MCC benchmarks. Because the planar routing cannot route enough nets, SLICE still uses a two-layer maze router after each planar routing phase to route the unrouted nets, so as not to use too many layers. Thus, it suffers some of the problems that the maze router faces such as long computation time, large memory requirement and additional vias.

2.4 V4R Router

The V4R router was also developed by the authors of SLICE, Khoo and Cong [24] [25]. Unlike SLICE, which mainly performs the routing in a single-layer, V4R takes the x-y routing approach. For each layer pair, the router divides the routing region into channels and processes channel by channel from the left to the right. Inside each channel, the router extends the routes of the nets under consideration by assigning them to appropriate horizontal or vertical tracks. It uses some optimization techniques, such as maximum weighted matching and maximum weighted k-cofamily, to maximize the track assignment. Once the rightmost channel is reached, unrouted nets are propagated to the next layer pair and the process is repeated until all nets are routed. One impressive characteristic of V4R is that it only allows four or fewer vias for each two-terminal net. The authors state that the four-via routing offers sufficient flexibility to complete the route, and yet avoid using too many vias.

V4R, like SLICE, is independent of net ordering. It runs 3.5 faster, uses 9% fewer vias, 2% less wire length and fewer routing layers than SLICE on the MCC benchmarks. The problem with V4R is that it is difficult to incorporate the electrical constraints into the optimization process. Also, it will be shown that the

number of vias, the total wire length, and the number of layers can be further reduced.

2.5 V4CMRA Router

V4CMRA was developed by Miyoshi et al. [32], in an attempt to bring crosstalk consideration into a V4R-type of router by limiting the parallel length of two lines running next to each other. It quickly produces competitive results with the price of sacrificing the optimal track assignment. Its track assignment is based on the predetermined ordering of nets and it is not an easy task to determine a good net ordering. Also, it is not clear how well it will perform in cases where all layers are densely routed since the optimal track assignment approach was eliminated. Although, they have incorporated crosstalk avoidance measure by limiting the length of parallel lines, this is a simplistic manner for dealing with crosstalk and does not guarantee an optimal noise tolerant design [17].

2.6 SURF Router

SURF was developed by Staepelaere et al. [42] [10]. It performs the routing in three phases: the global routing, the local routing and the transformation from topological sketches to geometrical layout. In the global routing phase, it uses hierarchical top-down partitioning to iteratively cut the whole routing region into small areas called bins until the routing problem inside each bin can be handled by the local router. During the partitioning, it produces rough routes for every net simultaneously. The local router processes a bin at a time and generates efficient topological sketches, called Rubber-band sketches, based on the rough routes of the nets. If the rough route of a net to be routed is not confined inside a single bin, it only generates a partial sketch for the net, and the partial sketch is extended in the processing of other relevant bins to form a total sketch. After all the bins

are processed, the Rubber-band sketches are finally transformed into a geometrical layout of the wires.

SURF allows arbitrary angle interconnects and variable width and spacing for the interconnection wires. It is difficult, however, for SURF to handle electrical constraints because only very simplistic electrical models can be applied to topological sketches and it is the final geometrical topology which determines the electrical properties. It has been reported that SURF is only effective in deposited thin-film MCMs where the number of layers for single distribution is usually small (2 layers in most cases). The effectiveness of using this router to solve multilayer general area MCM routing problems remains unknown.

2.7 Layer Assignment Before Routing

One method for MCM routing is first to assign the nets to a specified number of layers or layer pairs and then perform the routing in each layer or each layer pair [21] [5] [39]. The layer assignment is based on the geometrical locations of the nets, and the goal is to minimize the overall interference between the nets. This objective can greatly reduce the complexity of the original routing problem since the problem of routing on each layer or each layer pair is much smaller. The main problem with this method is that the number of layers to be used in the layer assignment must be predetermined. There is no good algorithm to estimate the number of layers needed. Another problem is that it does not guarantee the routability of the nets because the information contained in the geometrical locations of the nets is far from enough to tell us which way the nets can be routed.

2.8 Summary

The majority of the routing algorithms described above perform the routing in such a way that the route for a given net is extended incrementally piece by

piece without knowing the whole shape of the route before the completion of the routing process. Thus, it is difficult to address electrical constraints with any level of sophistication during the routing process. Then the designer has to relay on the post-routing electrical analysis followed by rip-up and re-route. This process may iterate many times, which leads to long turn-around times.

One possible way to solve the above problem is the constrained multilayer routing. It first generates a two-dimensional route for each net individually without considering the other nets. The route can be generated based on the design rules and optimization models such as minimum-length Steiner trees [22] and A-search trees [41] [5]. After this initial phase, the routes are assigned to different layers to solve the conflicts among the routes. This method can produce maximum performance, but at the same time, may cause too many layers to be used to resolve the conflicts and increase the cost. It is usually used to route the critical nets such as clock nets to guarantee higher performance and it is hard to use for general nets.

It is desirable to know the shape of the route for the net beforehand without the need to increase the number of layers. In the next chapter, a new multilayer MCM routing algorithm will be introduced which has been shown to be able to produce high quality routing solutions, while providing the flexibility for integrating electrical characteristics into the routing process.

CHAPTER 3

THE MCG ROUTING ALGORITHM

Given a net, there exist thousands of possible routes which could interconnect the terminals of the net. Only a small portion of the possible routes are acceptable (not necessarily optimal) by some standard, which we call candidate routes. The candidate routes for one net may not all be compatible with candidate routes for all the other nets in terms of geometrical and/or electrical constraints. Only a subset of all the candidate routes may co-exist in the space available for routing. An algorithm is needed to find this subset. The goal of this research was to develop an efficient routing algorithm based on the consideration of the compatibility of a set of candidate routes, which allows a more global approach than the algorithms described in the previous chapter. In addition, this approach will allow the incorporation of more accurate delay models and crosstalk models.

In the following discussion of the algorithm, a formal description of the problem formulation is given in Section 3.1. An overview of the algorithm is presented in Section 3.2. The remaining sections will discuss the ways to construct the candidate routes, handle multi-terminal nets, construct and reduce the compatibility graph (which is used to find the compatible subset of the candidate routes), and some post-routing processing techniques used to improve the quality of the routing solutions.

3.1 Problem Formulation

The MCM routing problem consists of a set of chips, C, a set of nets, N, a set of I/O terminals, P, and a multilayer routing substrate for the signal distribution. The chips are mounted on the top of the substrate. The pins (pads) of the chips and the I/O terminals are brought down to the first signal distribution layer using

distribution vias. It is assumed that a Manhattan grid is superimposed on each routing layer. The separation of grid points is determined by the design rules. The goal of the router is to complete the interconnection of all the nets using the signal distribution layers in the substrate. The output of the routing problem is a set of line segments and vias. The vias are used to connect two line segments in adjacent layers and to distribute the pins and the terminals to proper layers where routing can be performed. The former are called interconnection vias and the latter are called distribution vias. The distribution vias can be stacked vias, that is they can be stacked on top of each other to let the wire cross a number of layers. However, the use of vias should be limited because of the impedance discontinuity they introduce in the signal paths. The number of layers required for the routing, which is not predefined before routing, should also be minimized to decrease the manufacturing cost. The wire length used for interconnection of the net should be equal (or as close as possible) to the lower bound of the wire length for the net [26] to avoid long propagation delays in the signal paths and to allow increased routing density. Finally, the crosstalk among the routes of different nets should be addressed in such a way that it is guaranteed that the signal lines will not be falsely triggered.

3.2 Overview of the MCG Routing Algorithm

The routing is performed on two adjacent layers (x-y layer pairs) at a time, with all horizontal wires running on one layer and all vertical wires on the other layer. Adjacent wires on different layers are connected by vias. On each layer pair, the router first generates a small number of candidate routes for each net and then performs the compatibility test for every pair of candidate routes which results in a compatibility graph. Finally, the compatibility graph is reduced to yield a subset of routes which are fully compatible with each other. This process may be repeated a few times to route as many nets as possible on the given layer pair. Once no more nets can be completed on a given layer pair, the next layer pair is used.

until all the nets are routed. After the routing is completed, some post-processing techniques, which include jog removal and Steiner point insertion can be used to further improve the quality of the routing solutions in terms of the number of vias and the wire length.

For convenience, this algorithm is referred to as the MCG routing algorithm, which stands for multi-candidate compatibility graph routing algorithm.

Although the primary focus has been on routing on x-y layer pairs, the MCG router is not confined to x-y routing. The MCG routing algorithm is flexible enough to perform single-layer routing with only minimal modification.

3.3 The Candidate Routes

In this section, the topologies used for the construction of the candidate routes are discussed.

Several terms must be defined before the description of the types of candidate routes are presented. The bounding box for a net is defined as the smallest rectilinear box containing all terminals in the net. Assume a grid point in the routing area is labeled as (row-coordinate, column-coordinate). The upper, left corner is labeled $(0,0)$, the row-coordinates increase from top to bottom, and the column-coordinates increase from left to right. Given a two-terminal net, the terminal closest to the left and top is called the source terminal and referenced by (x_s, y_s). The other terminal is referred to as the target and referenced by (x_t, y_t). Starting from any grid point (x, y), (x, y_left) is the leftmost point reachable prior to reaching the border of the routing grid or an obstacle such as a via. In the same way, (x, y_right) is defined as the rightmost point reachable from (x_t, y_t) in the row x, (x_up, y) is the top-most point and (x_down, y) is the bottom-most point reachable from (x_t, y_t) in the column y. Given a point (x_p, y_p), y_p_left, y_p_right, x_p_up and x_p_down can be found as illustrated in Fig. 3.1. Assume the route can only be rectilinear. A route is represented by a series of points such as $\{ (x_s, y_s),$

(x_1, y_1), (x_2, y_2), ... (x_n, y_n), (x_t, y_t) }. The point (x_i, y_i) $i < n$ is a turning point of the route, which represents a jog in single-layer routing and a via in x-y layer pair routing. There are n jogs or vias in this example route.

There are two possible relative locations for the source and target terminals: one is that the source terminal is located at the top-left corner of the bounding box and the target terminal is at the bottom-right position; another is that the source terminal is located at the bottom-left corner of the bounding box and the target terminal is at the top-right position. In the following discussion, only the former case is discussed since the later case is the mirror image of the former. Therefore, without loss of generality, it is assumed that for the source terminal (x_s, y_s) and the target terminal (x_t, y_t) $x_s <= x_t$ and $y_s <= y_t$.

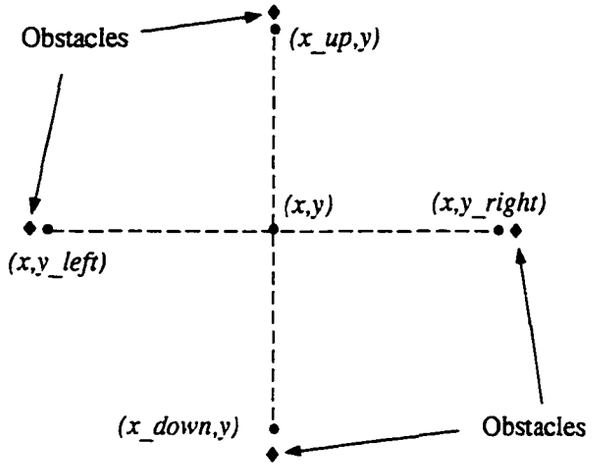


Figure 3.1: Routing Grid and Obstacles

3.3.1 Type-0 Routes

This type of route is defined as connecting two terminals of a given net without any vias. Type-0 routes obviously can only occur if the two terminals are on the same horizontal or vertical lines and there are no obstacle in between. Only one

such route may exist for a given two-terminal net. An example of a type-0 route is shown in Fig. 3.2.

Algorithm Type-0_Routes
Inputs: (x_s, y_s) and (x_t, y_t) – terminals
Outputs: Type0Route – possible type-0 route
Begin
routeNumber = 0
If $x_s == x_t$
If$y_s\_right >= y_t$
Type0Route = { (x_s, y_s), (x_t, y_t) }
routeNumber = 1
EndIf
Else If $y_s == y_t$
If $x_s\_down >= x_t$
Type0Route = { (x_s, y_s),(x_t, y_t) }
routeNumber = 1
EndIf
EndIf
End

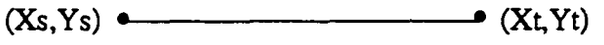


Figure 3.2: Type-0 Route

3.3.2 Type-1 Routes

Type-1 routes use only one via to connect the two terminals of a given net. This type of route can be constructed along the bounding box perimeter. At most two such routes exist. An example of a type-1 route is shown in Fig. 3.3.

Algorithm Type-1_Routes

Inputs: (x_s, y_s) and (x_t, y_t) - terminals

Outputs: Type1Route[] - possible type-1 routes

Begin

routeNumber = 0

If $x_s\_down >= x_t$

If $y_s >= y_t\_left$

Type1Route[routeNumber] =

$\{ (x_s, y_s), (x_t, y_s), (x_t, y_t) \}$

routeNumber = routeNumber + 1

EndIf

EndIf

If $y_s\_right >= y_t$

If $x_s >= x_t\_up$

Type1Route[routeNumber] =

$\{ (x_s, y_s), (x_s, y_t), (x_t, y_t) \}$

routeNumber = routeNumber + 1

EndIf

EndIf

End

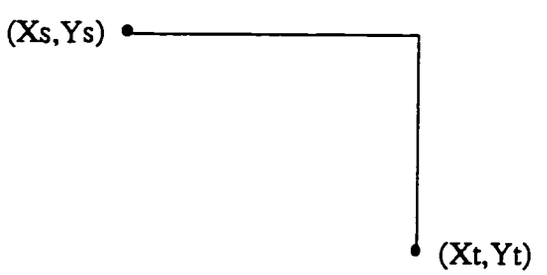


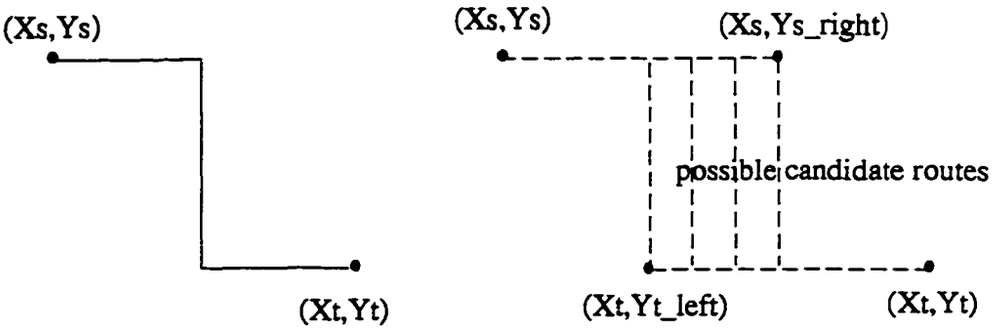
Figure 3.3: Type-1 Route

3.3.3 Type-2 Routes

Two vias are used to connect the two terminals of a given net using type-2 routes. An example of a type-2 route is shown in Fig. 3.4(a). Assuming the length and width of the bounding box of the two-terminal net are m and n respectively, there are at most $(m + n - 2)$ possible type-2 routes. The construction of type-2 routes is illustrated in Fig. 3.4(b).

Algorithm Type-2_Routes
Inputs: (x_s, y_s) and (x_t, y_t) - terminals
Outputs: Type2Route[] - possible type-2 routes
Begin
routeNumber = 0
If $y_t\_left <= y_s\_right$
For $y_p = y_t\_left$ to y_s_right
If the grid points from (x_s, y_p) to (x_t, y_p) are unoccupied
Type2Route[routeNumber] =
$\{ (x_s, y_s), (x_s, y_p), (x_t, y_p), (x_t, y_t) \}$
routeNumber = routeNumber + 1
EndIf
EndFor
EndIf
If $x_t\_up <= x_s\_down$
For $x_p = x_t\_up$ to x_s_down
If the grid points from (x_p, y_s) to (x_p, y_t) are unoccupied
Type2Route[routeNumber] =
$\{ (x_s, y_s), (x_p, y_s), (x_p, y_t), (x_t, y_t) \}$
routeNumber = routeNumber + 1
EndIf
EndFor
EndIf

End



(a) An Example of Type-2 Route (b) Construction of Type-2 Routes

Figure 3.4: Type-2 Route and Its Construction

3.3.4 Type-3 Routes

Type-3 routes use three vias to connect the two terminals. The routes are limited to be constructed within the bounding box of the net and have length equal to the Manhattan distance between the terminals. An example of a type-3 route is shown in Fig. 3.5(a). There are at most $2 \times (m-1) \times (n-1)$ possible type-3 routes. The type-3 routes are constructed by first selecting a point, (x_p, y_p), either between $(x_s + 1, y_s)$ and (x_s_down, y_s) or between $(x_s, y_s + 1)$ and (x_s, y_s_right), and then attempting to find type-2 routes between the point (x_p, y_p) and the target (x_t, y_t) as shown in Fig. 3.5(b).

Algorithm Type-3_Routes

Inputs: (x_s, y_s) and (x_t, y_t) - terminals

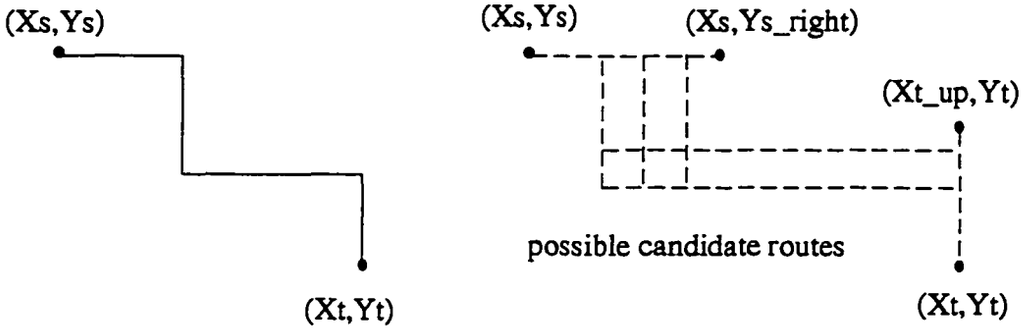
Outputs: Type3Route[] - possible type-3 routes

Begin

routeNumber = 0

For $y_p = y_s + 1$ to y_s_right

```
        Type2Route[] = Type-2_Routes((x_s, y_p), (x_t, y_t))
        For  i = 0 to number_of_type-2_routes_found - 1
            Type3Route[routeNumber] = { (x_s, y_s), Type2Route[i] }
            routeNumber = routeNumber + 1
        EndFor
    EndFor
    For  x_p = x_s + 1 to x_s_down
        Type2Route[] = Type-2_Routes((x_p, y_s), (x_t, y_t))
        For  i = 0 to number_of_type-2_routes_found - 1
            Type3Route[routeNumber] = { (x_s, y_s), Type2Route[i] }
            routeNumber = routeNumber + 1
        EndFor
    EndFor
End
```



(a) An Example of Type-3 Route (b) Construction of Type-3 Routes

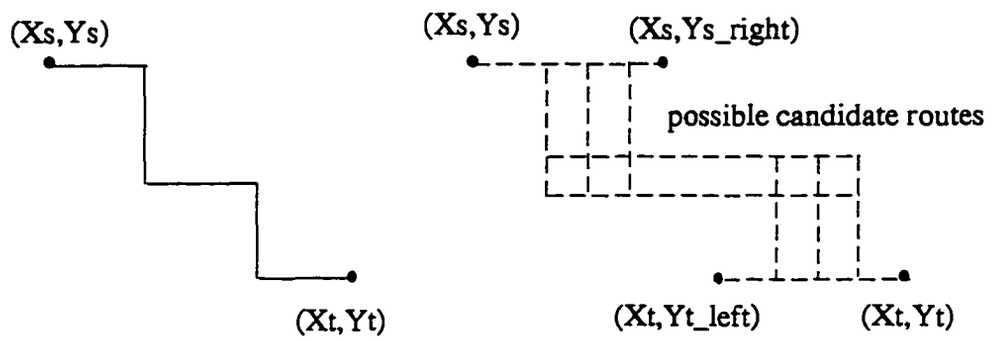
Figure 3.5: Type-3 Route and Its Construction

3.3.5 Type-4 Routes

Four vias are used to construct type-4 routes. These routes are of minimum length and constructed within the bounding box. The type-4 routes are constructed by selecting a pair of points with either the first point, (x_p, y_p), between $(x_s + 1, y_s)$ and (x_s_down, y_s) and the second point, (x_q, y_q), between (x_t_up, y_t) and $(x_t - 1, y_t)$ or the first point, between $(x_s, y_s + 1)$ and (x_s, y_s_right) and the second point, between (x_t, y_t_left) and $(x_t, y_t - 1)$. Then the algorithm attempts to find type-2 routes between the point (x_p, y_p) and the point (x_q, y_q). An example of a type-4 route and the construction of type-4 routes are shown in Fig. 3.6.

Algorithm Type-4_Routes
Inputs: (x_s, y_s) and (x_t, y_t) - terminals
Outputs: Type4Route[] - possible type-4 routes
Begin
routeNumber = 0
For $y_p = y_s + 1$ to y_s_right
For $y_q = y_t\_left$ to $y_t - 1$
If $y_p < y_q$
Type2Route[] = Type-2_Routes($(x_s, y_p), (x_t, y_q)$)
For i = 0 to number_of_type-2_routes_found - 1
Type4Route[routeNumber] =
{ (x_s, y_s), Type2Route[i], (x_t, y_t) }
routeNumber = routeNumber + 1
EndFor
EndIf
EndFor
EndFor
For $x_p = x_s + 1$ to x_s_down
For $x_q = x_t\_up$ to $x_t - 1$
If $x_p < x_q$

```
        Type2Route[] = Type-2_Routes((x_p, y_s), (x_q, y_t))
        For i = 0 to number_of_type-2_routes_found - 1
            Type4Route[routeNumber] =
                { (x_s, y_s), Type2Route[i], (x_t, y_t) }
            routeNumber = routeNumber + 1
        EndFor
    EndIf
  EndFor
EndFor
End
```



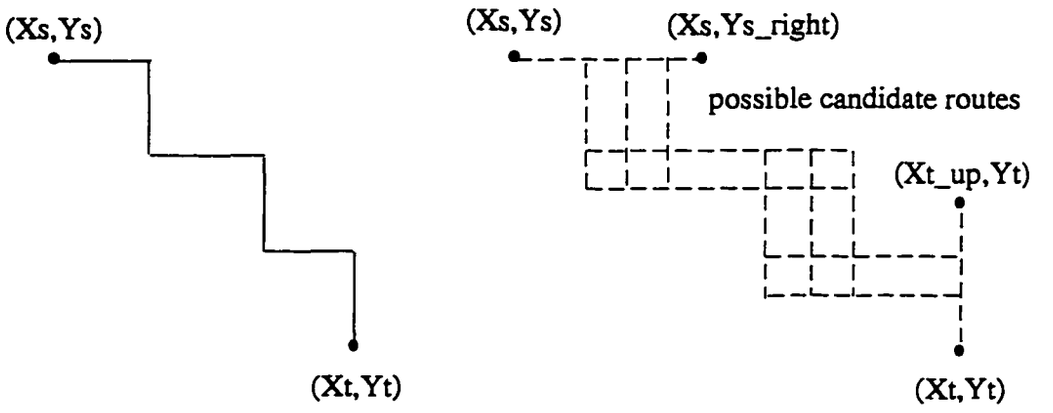
(a) An Example of Type-4 Route (b) Construction of Type-4 Routes

Figure 3.6: Type-4 Route and Its Construction

3.3.6 Type-5 Routes

Type-5 routes use five vias to construct minimum length routes inside the bounding box of the net. This type of route is constructed by selecting a point, (x_p, y_p), inside the bounding box, and then attempting to find type-2 routes from (x_s, y_s) to (x_p, y_p) and from (x_p, y_p) to (x_t, y_t). An example of a type-5 route and the construction of type-5 routes are shown in Fig. 3.7. The use of type-5 routes

is limited because they use more vias than other types. To further reduce vias, type-5 routes can be omitted entirely.



(a) An Example of Type-5 Route (b) Construction of Type-5 Routes

Figure 3.7: Type-5 Route and Its Construction

Algorithm Type-5_Routes

Inputs: (x_s, y_s) and (x_t, y_t) - terminals

Outputs: Type5Route[] - possible type-5 routes

Begin

routeNumber = 0

For $x_p = x_s + 1$ to $x_t - 1$

For $y_p = y_s + 1$ to $y_t - 1$

Type2Route1[] = Type-2_Routes($(x_s, y_s), (x_p, y_p)$)

Type2Route2[] = Type-2_Routes($(x_p, y_p), (x_t, y_t)$)

For i = 0 to number_of_type-2_routes_found1 - 1

For j = 0 to number_of_type-2_routes_found2 - 1

Type5Route[routeNumber] =

{ Type2Route1[i], Type2Route2[j] }

routeNumber = routeNumber + 1

EndFor

EndFor

EndFor

EndFor

End

3.3.7 Type-X Routes

This type of route uses less than four vias to connect two terminals of the given net. These routes are allowed to go outside of the bounding box. Therefore, they are not of minimal length. The type-X routes are constructed by selecting a pair of points with the first point, (x_p, y_p), either between (x_s_up, y_s) and (x_s_down, y_s) or between (x_s, y_s_left) and (x_s, y_s_right) and the second point, (x_q, y_q), either between (x_t_up, y_t) and (x_t_down, y_t) or between (x_t, y_t_left) and (x_t, y_t_right) and

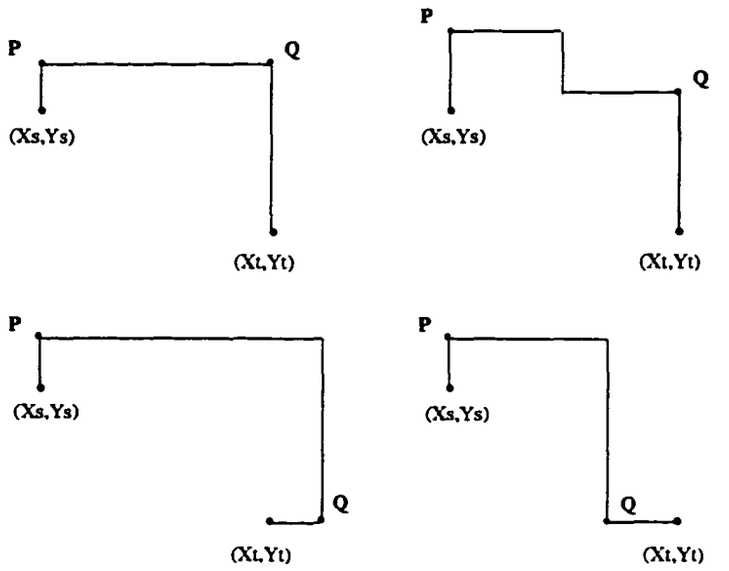


Figure 3.8: Examples of Type-X Routes

then attempting to find type-2 routes between the point (x_p, y_p) and the point (x_q, y_q). Examples of type-X routes are shown in Fig. 3.8.

Because type-X routes are not guaranteed to have minimum length, they are only used in the final phase of routing after the routes with minimum length have been considered. The length of type-X route can be reduced by limiting the range of the points (x_p, y_p) and (x_q, y_q).

Algorithm Type-X_Routes

Inputs: (x_s, y_s) and (x_t, y_t) - terminals

Outputs: TypeXRoute[] - possible type-X routes

Begin

routeNumber = 0

For $x_p = x_s\_up$ to x_s_down

For $y_q = y_t\_left$ to y_t_right

Type1Route[] = Type-1_Routes($(x_p, y_s), (x_t, y_q)$)

For i = 0 to number_of_type-1_routes_found - 1

TypeXRoute[routeNumber] =

{ (x_s, y_s), Type1Route[i], (x_t, y_t) }

routeNumber = routeNumber + 1

EndFor

EndFor

EndFor

For $x_p = x_s\_up$ to x_s_down

For $x_q = x_t\_up$ to y_t_down

If $x_p == x_q$

Type0Route = Type-0_Route($(x_p, y_s), (x_q, y_t)$)

If number_of_type-0_routes_found == 1

TypeXRoute[routeNumber] =

{ $(x_s, y_s), (x_p, y_s), (x_q, y_t), (x_t, y_t)$ }

routeNumber = routeNumber + 1

EndIf

EndIf

Type2Route[] = Type-2_Routes($(x_p, y_s), (x_q, y_t)$)

For i = 0 to number_of_type-2_routes_found - 1

TypeXRoute[routeNumber] =

{ (x_s, y_s), Type2Route[i], (x_t, y_t) }

routeNumber = routeNumber + 1

EndFor

EndFor

EndFor

The code below is the mirror image of the above code

.....

End

3.3.8 Type-Z Routes

The types of routes previously discussed are all simple in topology and easy to construct. Other types of candidate routes can be constructed although an exhaustive description will not be included. However, in some cases it is necessary to find a route for the given net no matter the number of vias and the wire length. Hence, the type-Z routes are used, which actually have no fixed topology and no restriction on the number of vias and wire length. Such a route can take any shape as long as it connects the terminals of the given net. The use of the type-Z routes are very limited because it is time consuming to construct and they tend to use more vias and extra wire length. Type-Z routes are based on the line-probe algorithm [31]. In most designs these routes will not be used at all in x-y routing. And they will be used only if it results in the elimination of an entire layer pair.

The line-probe algorithm searches the path from the source to the target by generating a series of line segments which do not pass through any obstacle. The line segments generated start from the source terminal and are kept in a list. The

operation of generating line segments is repeated until a line segment in the list reaches the target or no more line segments can be generated. The path can be formed by retracing the line segments in the list from the target to source. The line segments are generated from the escape points on the existing line segments in the list. Every grid node on the line segment is an escape point which generates a new perpendicular line segment. Initially, there is only one escape point, the source terminal. The algorithm is guaranteed to find a path if one exists. Fig. 3.9 shows an example of a net routed by the line-probe algorithm.

Algorithm Type-Z_Route
Inputs: (x_s, y_s) and (x_t, y_t) - terminals
Outputs: TypeZRoute - a possible type-Z route
Begin
newLineList = line segment generated form (x_s, y_s)
While newLineList is not empty
insert every line segment in newLineList into lineList
For each line segment s in lineList
If s reaches (x_t, y_t)
pathExist = TRUE
exit while
EndIf
EndFor
set newLineList empty
For each line segment s in lineList
For each escape point e on s
newLine = generateNewLine(e, s)
insert newLine into newLineList
EndFor
EndFor
EndWhile
If pathExist = TRUE

TypeZRoute = retrace(lineList)

Else

a path cannot be found

EndIf

End

The procedure generateNewLine generates a new line segment which is perpendicular to line segment s and passes through the escape point e. The procedure retrace traverses the list of the line segments and produces a type-Z route for the net. The procedure only produces one type-Z route. If more than one type-Z route is needed, the procedure can be modified to find multiple type-Z candidate routes, if they exist.

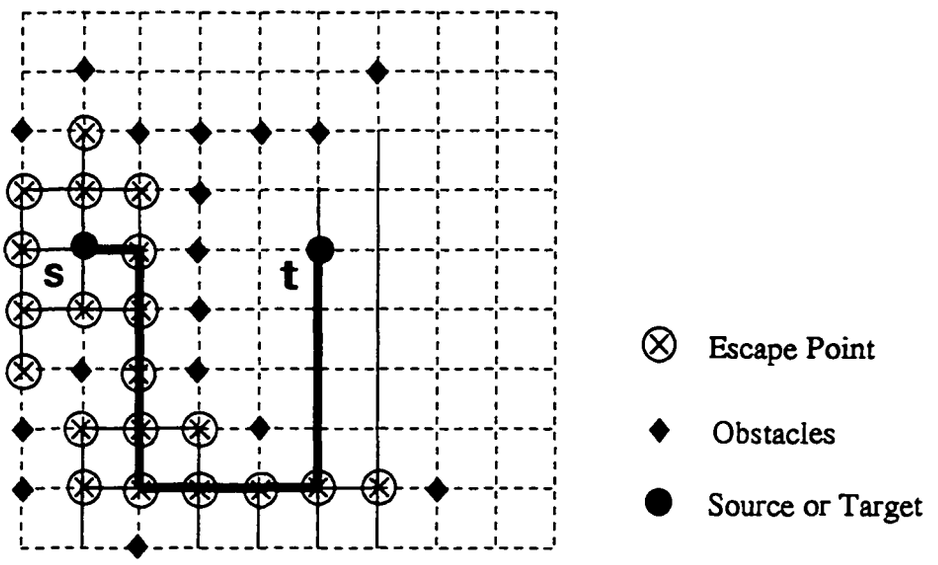


Figure 3.9: A Net Routed by Line-Probe Algorithm

3.4 Multi-Terminal Nets

The types of routes described above are only for two-terminal nets. Although most nets are two-terminal nets, it is also necessary to handle multi-terminal nets. The candidate routes for multi-terminal nets can be generated by a minimum-length Steiner tree algorithm [22] or an A-search tree algorithm [41], [5]. However, these algorithms are very costly to implement, hence only suitable for a few critical nets. For other non-critical multi-terminal nets, the technique used is based upon the traditional minimal spanning tree approach. For each multi-terminal net, the minimal spanning tree is determined. For a t terminal net, the minimal spanning tree will contain $t-1$ edges. The t terminal net is then decomposed into the corresponding $t-1$ two terminal nets. In the routing algorithm these routes are then treated as two terminal nets with the property that they are never seen as obstacles to other nets that arose from the same multi-terminal net. This effectively allows the insertion of Steiner points for some nets during the routing process, and thus can reduce the total wire length of multi-terminal nets below that of the minimal spanning tree. It is still possible to insert more Steiner points during the post-processing phase which follows the routing process. The decomposition of the multi-terminal nets occurs during initialization. Therefore, for the purpose of discussing the details of the routing algorithm, it is assumed that all nets have two terminals.

3.5 The Compatibility Graph and Its Reduction

The candidate routes of one net may not be compatible with candidate routes of other nets due to geometrical and/or electrical constraints. Therefore, a compatibility test is performed, which results in the compatibility graph. This graph is then reduced in polynomial time to yield a subset of the candidate routes which are fully compatible. The candidate routes can be constructed as discussed in earlier sections. However, it is not reasonable to consider all the possible candidate routes

even using the simple topologies. Therefore, the number of candidate routes for each net must be restricted. Let c be the predetermined number. A maximum of c candidate routes will be allowed for each net (Selection of the value for c is discussed in the next chapter).

Now let us define the route compatibility graph (RCG). An RCG is a graph $G = (V, E)$ where V is a set of vertices representing candidate routes and E is a set of edges such that edge e_{ij} exists if and only if routes v_i and v_j cannot be routed on the same layer in single-layer routing or on an x-y pair of layers in x-y layer pair routing due to either geometrical or electrical constraints. Two vertices v_i and v_j in the graph are considered compatible if no edge (v_i, v_j) exists in G. As a result, the corresponding routes can be compatibly routed on the same layer or same x-y layer pair. If E = 0 (the empty set), then all the routes corresponding to the vertices in the graph can be compatibly routed on the same layer or same x-y layer pair. However, the possibility of having such a graph originally is very small. The graph, however, can be reduced to obtain a graph with no edges (a null graph). A selected vertex can be deleted from the graph, which means it will not be considered as a candidate route any more. Then all the edges incident to it are also removed. This process is repeated until no edges exist in the graph. All the routes corresponding to the remaining vertices are compatible with each other. The problem with this approach is how to select the vertex to be deleted so that as many vertices as possible remain because our goal is to find the largest subset of the candidate routes that are compatible. This problem is an NP hard problem.

Theorem: The problem of finding the largest compatible subset of candidate routes is NP hard.

Proof: Let $G_1 = (V_1, E_1)$ be the complement of G. G_1 has the same set of vertices and two vertices are connected in G_1 if and only if they are not connected in G. There is an edge incident to two vertices, if their corresponding routes are compatible. The subset of such compatible routes corresponds to a clique in G_1. Finding the largest compatible subset of candidate routes is equivalent to finding

the clique with largest size in G_1. So the problem is NP hard [30].

Because of the NP hardness of the problem, an approximation algorithm is needed to find a good solution within reasonable computation time. A simple heuristic technique has been taken to find the desired subset. The technique is first to find the vertex with largest degree in the graph, delete it, then update the graph, repeat the process until no edges remain. Each time when the vertex with largest degree is deleted, more edges are eliminated from the graph than would be for any other vertex. This hopefully will delete several edges and retain as many vertices as possible. This simple technique has been very effective. The results of testing the algorithm are given in the next chapter.

After the reduction, all the routes corresponding to the remaining vertices can be compatibly routed on the same layer or the same x-y layer pair. It is possible that multiple candidates remained for some nets. In this case, only one candidate route needs to be selected for each of these nets. The selection can be either arbitrary or be decided based on priority, number of vias, other electrical considerations, etc.

For the nets with no candidates remaining, they can be routed on the next layer or next layer pair. Since not all the possible candidate routes are used in the construction of the graph and the technique used in the graph reduction is not guaranteed to find the largest subset of candidate routes, there may exist some candidate routes for these nets which are not included in the subset that was found and are actually compatible, hence the nets can be routed in the current layer or x-y layer pair. Therefore, another iteration is tried in the same layer or x-y layer pair to get as many nets routed as possible before another layer or pair of layers is added. We construct candidate routes for unrouted nets, build the compatibility graph, reduce it, and route additional nets (previously routed nets become obstacles). The process can be repeated until all nets are routed or no more nets can be routed. If there are still some nets unrouted after this process, another layer or layer pair must be added. The unrouted nets are then propagated to the new layer(s), and the routing process is repeated until all the nets are completed.

Routing Algorithm Based On Compatibility Graph

Inputs:

netNumber – number of net

Net[] – the net list

c – number of candidate routes allowed for each net

Outputs:

Route[] - routing solution

Begin

Construct_Candidate_Routes

Build_Compatibility_Graph

Reduce_Compatibility_Graph

Select_Route

End

Procedure Construct_Candidate_Routes

Inputs:

netNumber – number of net

Net[] – the net list

c – number of candidate routes allowed for each net

Outputs:

Vertex[][] - vertices associated with candidate routes

Begin

For i = 0 **to** netNumber - 1

If Net[i] is not routed

CandRoute[] = construct_candidate_routes(Net[i], c)

For j = 0 **to** number_of_routes_constructed - 1

Vertex[i][j].route = CandRoute[j]

EndFor

Else

Vertex[i][0].route = the previously routed route of Net[i]

EndIf

EndFor

End

The vertices are grouped by nets. The vertices in Vertex[i] are all associated with candidate routes for Net[i]. For convenience, we refer to them as the vertex-group of Net[i].

Procedure Build_Compatibility_Graph

Inputs:

Vertex[][] – vertices with associated candidate routes

Outputs:

Begin

G(V, E) – the compatibility graph

V – the vertices in G

E – the edges in G

Begin

V = all the vertices in Vertex[][]

For each vertex v_i in G

For each vertex v_j in G

compatible = compatibleTest(v_i.route, v_j.route)

If not compatible

add edge (v_i, v_j) into E

EndIf

EndFor

EndFor

End

In the compatibility test, if the two routes to be tested are from the same multi-terminal net, they obviously are compatible.

Procedure Reduce_Compatibility_Graph

Inputs:

G(V, E) – compatibility graph

Outputs:

Gprime – reduced graph with no edge

Begin

While there exist edge in E

For each vertex v in V

compute degree of v

EndFor

find v_{max} with maximum degree

delete v_{max} and all the edges incident to v_{max} from G

EndWile

End

In the procedure Reduce_Compatibility_Graph, there may exist multiple vertices with maximum degree. Ties will be broken by the number of vertices remaining in the vertex-group. Since it is best to route as many nets as possible, the vertex in the vertex-group with more remaining vertices is selected and then deleted.

Procedure Select_Route

Inputs:

Gprime – the reduced graph

Outputs:

Route[] – the routes which can be routed compatibly

Begin

For i = 0 **to** netNumber - 1

If Net[i].status = UNROUTED

If there exists vertex in the vertex-group of Net[i]

Net[i].status = ROUTED

Route[i] = select one route from the group

EndIf

EndIf

EndFor

End

If there is no vertex remaining in the vertex-group, the algorithm will attempt to route the net in the next iteration. If there are one or more vertices in the vertex-group, one and only one should be selected to be the actual route for the net. The selection can be arbitrary or can be decided by certain rules such as electrical characteristics of the route. All the selected routes will become obstacles in the future iterations.

The algorithm described above performs the routing in such a way that the nets are routed simultaneously instead of routing net-by-net, hence it is not subject to the problem of ordering of the nets; and the candidate routes are constructed as a whole instead of extending the routes piece by piece, which makes the incorporation of timing constraints (delay models) easy by simply constructing the candidate routes for the nets according to the requirements of the design; and the compatibility test of the candidate routes can be used to estimate the crosstalk between the routes from different nets instead of between the line segments of the routes, which make the crosstalk estimation more sophisticated than the other technique. In general, the algorithm offers some unique features such as taking a more global approach to handling the routing problem and it is easy to incorporate the electrical constraints into the routing process, which is absent or hard to handle in other algorithms. It gives the designer the flexibility of selecting the topology for routes for the nets, which is almost impossible for other algorithms.

3.6 X-Y Layer Pair Routing

X-y layer pair routing performs the routing on two adjacent layers at a time; with all horizontal wires running on one layer and all vertical wires on the other layer. Adjacent wires on different layers are connected by vias. On each layer pair, the routing algorithm described in the previous section is used to produce the

routing solution.

Due to the iterative nature of the algorithm, different types of routes can be used as the candidate routes in different iterations. The preferred types can be used in the initial iterations and the use of alternative types can be delayed until the preferred options have been exhausted. Among the types of routes described before, the preferred types are type-0 and type-1 routes because they use the minimum number of vias and minimum wire length. However these two types of routes are not always available. Even if they do exist, the number of such routes is very small; there are at most only one for type-0 route and two for type-1 routes. More candidates are certainly needed for the algorithm to produce high quality results. On the other hand, the type-2 routes only use one extra via and still uses minimum wire length. Also, it is easy to find more type-2 routes. Therefore, type-0, type-1 and type-2 routes are selected to be the candidate routes in the initial iterations of the algorithm. This process is called Phase 1 of the routing process.

After Phase 1, we use type-3, type-4 and type-5 routes as the candidate routes for Phase 2 of the routing process. The types of routes used in Phase 2 are still inside the bounding box with minimum wire length. They use a few more vias to interconnect nets. Since type-3 routes use fewer vias than other types of routes, the candidate routes are first constructed using type-3 routes. For a given net, if c candidate routes are found, no more routes will be constructed. Otherwise, type-4 routes are constructed. If there are still not enough candidate routes, type-5 routes may be used for candidates. If we do not want to introduce too many vias into the routing solution, the step where type-5 routes are constructed can be omitted.

Phase 3 of the routing process uses type-X routes as candidate routes. Since this type of route is not confined to the bounding box of the net, the route can virtually go anywhere on the plane; thus it may introduce additional wire length and increase the total wire length of the routing solution. However, we can reduce the wire length by limiting the size of the extended bounding box.

Each of the three phases can have a number of iterations. One phase can

be run until no more additional nets can be routed. before going into the next phase. However, it may not be efficient practically, because after a few iterations the additional nets routed in the subsequent iterations diminishes greatly. A lot of time may be spent without getting many additional nets routed with this approach. Therefore, a number ϵ can be specified so that the current phase stops if the additional nets routed is less than ϵ. The suggested range of ϵ is 0.5 - 1.0% of the total number of nets.

Phase 1 of the Routing Process
Begin
  Repeat
    For each net
      Construct c candidate routes using type-0,
        type-1 and type-2 routes
    EndFor
    Build compatibility graph
    Reduce the graph
    Select routes
  Until number of additional nets routed $< \epsilon$
End

Phase 2 of the Routing Process
Begin
  Repeat
    For each net
      Construct c candidate routes using type-3
      If number of routes constructed $< c$
        Construct the rest candidate routes using type-4
      EndIf
      If number of routes constructed $< c$
        Construct the rest candidate routes using type-5

EndIf

EndFor

Build compatibility graph

Reduce the graph

Select routes

Until number of additional nets routed $< \epsilon$

End

Phase 3 of the Routing Process

Begin

Repeat

For each net

Construct c candidate routes using type-X route

EndFor

Build compatibility graph

Reduce the graph

Select routes

Until number of additional nets routed $< \epsilon$

End

If there are still some nets that are unrouted after the three routing phases, another layer pair is added, the unrouted nets are propagated to the new layer pair, and the routing process is repeated until all the nets are completed. If the number of unrouted nets is very small (less than 1% of the total number of nets) after three routing phases, type-Z routes can be used to route them on the current layer pair instead of using another layer pair. Thus, a pair of routing layers may be saved at the expense of a few additional vias and extra wire length. The reduction in the number of layers may well justify the cost of the vias and extra wire. This optional process is called Phase 4.

Phase 4 of the Routing Process

Begin

```
If number of unrouted nets < 0.01× netNumber
    For i = 0 to netNumber - 1
        If Net[i].status = UNROUTED
            Route[i] = type-Z_Route()
            If type-Z route found
                Net[i].status = ROUTED
            EndIf
        EndIf
    EndFor
EndIf
End
```

3.7 Single-Layer Routing

Single-layer routing is even more difficult than x-y routing since all the wire must be laid out on the same plane and one wire could possibly block the way of a number of other routes. So single-layer routing is not a common practice in MCM routing. However there may exist some cases where single-layer routing is useful, such as, some critical nets are preferred to be routed on a single plane due to electrical considerations.

The MCG algorithm described in Section 3.5 is general enough to perform single-layer routing. The difference between single-layer and x-y routing is the compatibility test since the routes compatible on the x-y layer pair may not be compatible at all on a single-layer (plane). The other issue in single-layer routing is the construction of candidate routes. In x-y layer pair routing, type-0, type-1, type-2, type-3, type-4, type-5 and type-X routes are used and type-Z routes are only used to route a few nets if it reduces the total number of layers. These types of routes can still be used in single-layer routing. The single-layer routing can also be performed in the same way as x-y layer pair routing. However, some types of

routes, e.g. type-X routes, may not be as useful as in the case of x-y layer pair routing, because after type-0 through type-5 routes have been tried the chance of finding type-X routes is very small on the single-layer(plane). On the other hand, the type-Z routes, which are rarely used in x-y layer pair routing, may be more useful than type-X routes because they are much more flexible.

The algorithm for finding type-Z routes described in Section 3.3.8 must be modified for single-layer routing. First, the search area should be restricted because it is unnecessary and inefficient to search the whole plane. A rectangular area, which may be a little larger than the bounding box of the net, can be specified to be the platform for searching. Second, the algorithm described in Section 3.3.8 finds only one type-Z route if it exists. For the purpose of constructing candidate routes, more than one such route is certainly needed. To find multiple type-Z routes, the retracing part of the algorithm needs to be changed and the searching part untouched. In retracing, the list of line segments should be traversed multiple times and each time a different set of line segments is used to construct the route.

The process of single-layer routing is similar to the process of the x-y layer pair routing with three phases. Phases 1 and 2 are the same while Phase 3 is different. In Phase 3 of the single-layer routing process, we use type-Z routes as candidate routes instead of type-X routes. Therefore, there is no Phase 4 in single-layer routing.

Phase 1 of the Routing Process

Same as in Section 3.6

Phase 2 of the Routing Process

Same as in Section 3.6

Phase 3 of the Routing Process

Begin

Repeat

For each net

Construct c candidate routes using type-Z routes

EndFor

Build compatibility graph

Reduce the graph

Select routes

Until number of additional nets routed $< \epsilon$

End

3.8 Post-Routing Processing

After all the nets are completed by the routing process, some post-routing processing techniques can be used to further improve the quality of the routing solution in terms of the number of vias and wire length. These techniques include jog removal, Steiner point insertion, etc.

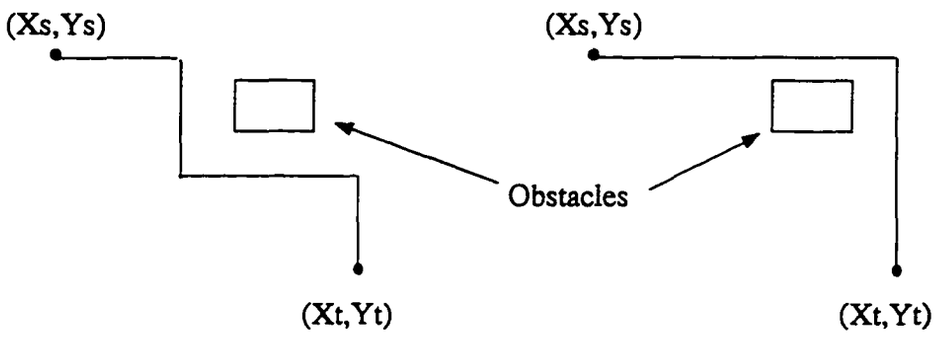


Figure 3.10: Type 1 Jog and its Removal

3.8.1 Jog Removal

Jog removal is a simple technique to eliminate extra vias and make the interconnection shorter. Two types of jogs are easily identified and they can be removed by changing the path of the route. Figs. 3.10 and 3.11 show the jogs (vias) removal.

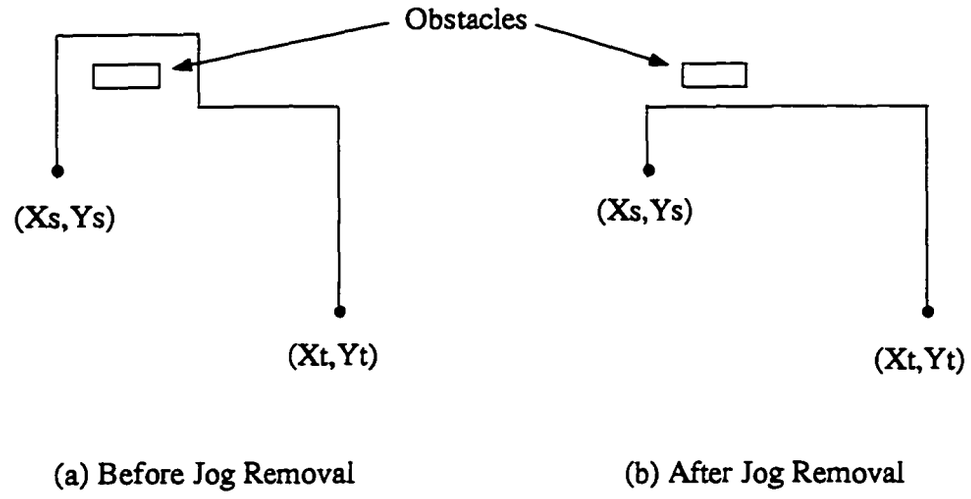


Figure 3.11: Type 2 Jog and its Removal

3.8.2 Steiner Point Insertion

Before the routing process, multi-terminal nets are split into several simple two-terminal nets which are routed independently during the routing process. After all the nets are routed, it is found that the routes which interconnect the multi-terminal net may not be in optimal positions because there is no interaction between the two-terminal nets during the routing process. For example, two wires may run parallel without knowing about each other's existence. These cases will be identified after the routing process, and a Steiner point can be introduced in the proper position. Then one of the routes involved is changed and re-routed through this Steiner point. Thus, the parallel wires in the same net will be eliminated. The

selection criterion for the route to be changed is the reduction of vias and wire length. The route which can eliminate more vias and use less wire for the net is selected. Fig. 3.12 shows the route of a three-terminal net before and after Steiner point insertion and re-routing.

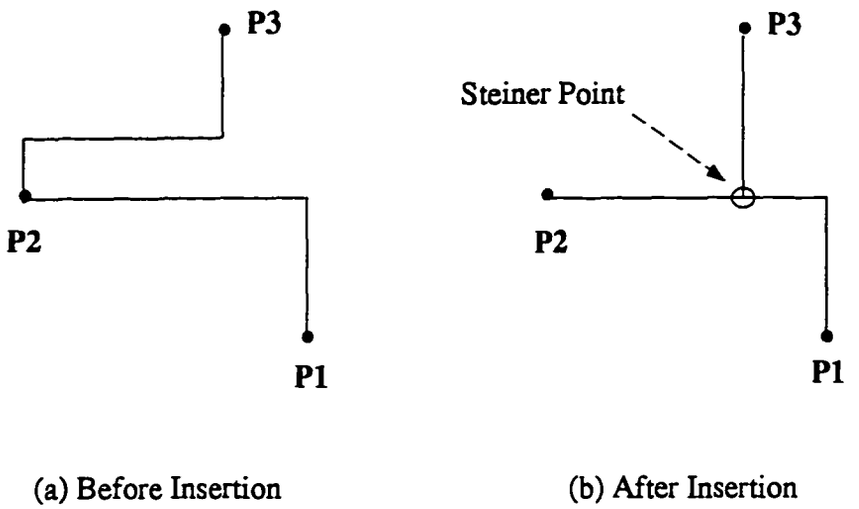


Figure 3.12: An Example of Steiner Point Insertion

3.9 Summary

In this chapter, the MCG routing algorithm was discussed in detail. Various types of routes used to construct the candidate routes were defined. The compatibility graph for the candidate routes and its reduction was discussed. The algorithm enables us to perform the routing simultaneously and incorporate the electrical constraints easily.

The algorithm is very effective and it yields high quality routing solutions. The experimental results and analysis will be presented in the next chapter.

CHAPTER 4

EXPERIMENTAL RESULTS AND ANALYSIS

The MCG routing algorithm was implemented on a Sun Microsystems workstation (Sparc-10) using the C++ language. The algorithm was tested on a number of examples, which include the three industrial designs provided by MCC [19]. The test results for these examples are given and compared with the performance of other routers introduced in Chapter 2. The complexity of the algorithm both in terms of time and space is analyzed. In addition, some implementation issues, such as the data structures used and the selection of the number of candidate routes. are also discussed.

4.1 Test Examples

The test examples include three standard industrial benchmarks provided by MCC and test data used by the authors of M^2R.

4.1.1 MCC Benchmarks

The MCC benchmarks include three designs namely MCC1-75, MCC2-75 and MCC2-45, which were provided by the Microelectronics and Computer Technology Corporation (MCC). The last two digits represent the routing pitch. For example, 75 means 75-micron. These designs were used as MCM routing benchmarks for the 4th ACM/SIGDA Physical Design Workshop because they are real designs from industry instead of artificial examples, and they are large in size which provide very challenging tests for MCM routing algorithms. They are available via anonymous ftp from *mcnc.org* in the directory pub/benchmark/PDWorkshop93/benchmarks. These designs have been widely accepted as test benchmarks for MCM routing

and were used as test examples in a number of research papers including those presenting SLICE, V4R, V4CMRA.

1. MCC1

This design consists of 6 chips, 765 I/O pins, 799 signal nets, two power nets and one ground net. There are 2496 pins total, 2043 of which are signal pins. There are numerous 3 to 7 terminal nets. The power and ground nets are not routed on the x and y layers, but are tied directly to the power and ground planes. There are two chip footprints: 550 × 550 mils with 448 pins and 330 × 330 mils with 272 pins. The SUBSTRATE footprint distributes the I/O pins around the perimeter of the substrate. Table 4.1 gives a net count for MCC1. The first column shows the number of terminals of the net, and the second column shows the number of nets that have the specified number of terminals. The routing pitch for MCC1 is 75-micron.

Table 4.1: MCC1 Net Count

No. of Terminals in the Net	No. of Nets
2	608
3	87
4	50
5	3
6	6
7	45
72 (power)	1
176 (power)	1
205 (ground)	1
Total No. of Nets	802

2. MCC2

This routing example models a next generation supercomputer on a 6 × 6 inch substrate with 37 Honeywell VHSIC gate arrays and 18 high density connectors. The chips are 1.5 × 1.5 cm with 35 mil TAB leads on a 4 mil pitch. The connectors are placed around the perimeter of the substrate. The net list contains 7118 signal nets and 14659 pins. Table 4.2 gives a net count for MCC2. The net list does not include the power and ground nets, and there are not many multi-terminal nets in the design. There are two routing pitches for MCC2, 75-micron and 45-micron, respectively. We refer to them as MCC2-75 and MCC2-45.

Table 4.2: MCC2 Net Count

No. of Terminals in the Net	No. of Nets
2	6698
3	415
4	5
Total No. of Nets	7118

Table 4.3: Characteristics of the MCC Benchmarks

Example	No. of Chips	No. of Nets	No. of Pins	Size of Substrate (mm^2)	Grid Size
MCC1-75	6	802	2496	45 × 45	599 × 599
MCC2-75	37	7118	14659	152.4 × 152.4	2032 × 2032
MCC2-45	37	7118	14659	152.4 × 152.4	3386 × 3386

Table 4.3 gives the characteristics of all the MCC designs. From the table, we can see that MCC2 is a very large design which has a large number of chips, nets and pins and a very large routing area. So it is a challenging test example for the

routing algorithms and some algorithms do fail to route it due to their inability to handle such large routing problems.

4.1.2 Test Data of M^2R

This set of examples came from the authors of [35]. Although these data are randomly generated and not real designs, they can be used to compare the results in the case of single-layer routing. There are two examples in the set; one has 100 nets and a 60 × 60 routing grid and the other has 200 nets and a 120 × 120 routing grid. They are referred to as M^2R-100 and M^2R-200, respectively. Table 4.4 gives the characteristics of the examples, and Table 4.5 lists the net count of the examples.

Table 4.4: Test Examples of M^2R

Example	No. of Nets	No. of Pins	Grid Size
M^2R-100	100	303	60 × 60
M^2R-200	200	505	120 × 120

Table 4.5: M^2R Net Count

No. of Terminals in the Net	No. of Nets	
	M^2R-100	M^2R-200
2	48	127
3	16	41
4	21	32
5	15	0
Total No. of Nets	100	200

4.1.3 Flip-Chip Example

To test the algorithm's ability to handle the flip-chip bounding technology, another example was generated from MCC1. Flip-chip bounding is a relatively new technology to mount the chip on top of the substrate. Flip-chip bonding uses small solder balls on the I/O pads of the chip to both physically attach the chip and make the required electrical connections. This procedure is also called face-down bonding. The location of the I/O pads are on one face of the chip instead of being around the perimeter of the chip. We generated the flip-chip example by manually moving the locations of all the pads of the chips in MCC1 from the perimeter of the chips to inside the footprint of the chips. All the other specifications of MCC1 remained the same. This test example is referred to as FLIP.

4.2 Test Results for X-Y Layer Pair Routing

In this section, the test results of the x-y layer pair routing are presented. The main examples are the MCC benchmarks. And the results are compared with that of other routers, including the V4R, SLICE, V4CMRA, and 3D-Maze algorithm that were introduced in Chapter 2 and a commercial Router X.

Router X is a PCB/MCM router from a major commercial CAD vendor, which is a grid based router using sophisticated multi-pass routing, including maze routing, rip-up and re-routing, shove-aside routing, etc. These techniques were developed and refined over a number of years. It is considered to be a state-of-the-art industrial PCB/MCM router. (Note that X is not the real name of this router. The real name cannot be disclosed due to contractual agreements [27].)

First, the equation for determining the lower bound of the total wire length for the nets is given. This lower bound can be computed as in [25]. The equation for computing this lower bound, $b(i)$, for the wire length of net_i is

$$b(i) = \max\{m(i), \frac{2}{3}\mathrm{MST}(i)\} \tag{4.1}$$

where $m(i)$ is 1/2 the perimeter of the smallest bounding box containing all terminals of net_i, and MST(i) is the length of the minimal spanning tree for net_i. The 2/3 factor is derived from the fact that in Manhattan routing the wire length of the minimum spanning tree is at most 1 1/2 times the wire length of the minimum Steiner tree [22]. The lower bound on total wire length,B, is given by

$$B = \sum_{i \in N} b(i) \tag{4.2}$$

4.2.1 Test Results for MCC1

MCC1 was tested in two different settings; one is without the power and ground nets and the other is with the power and ground nets. Although it is specified that power and ground nets are not routed on the x and y layers, but are tied directly to the power and ground planes, some other research papers accidentally routed the power and ground nets with the signal nets on the x-y layer pair. For the convenience of comparison, the power and ground nets were also routed with the signal nets on the x-y layer pair. The test example for MCC1 without the power and ground nets is referred to as MCC1(w/o P&G) and the other as MCC1(P&G).

Table 4.6: Routing Results for MCC1(w/o P&G)

No. of Layers	No. of Vias	Total Wire Length
4	4797	371959

MCC1(w/o P&G)

Table 4.6 gives the routing results for the the MCC1(w/o P&G) by the MCG router before using the post-processing techniques. The first column shows the total number of layers used by the MCG router. The second column shows the total number of vias used by the MCG router, which includes both the number of stacked vias used for bringing the terminals to their proper routing layers and

the number of vias used for connecting the net. The third column shows the wire length used by the MCG router.

Table 4.7 shows the results of applying the technique of jog removal for MCC1(w/o P&G). It lists the total number of vias and total wire length both before and after the processing and the percentage of improvement. The total number of vias is 2.19% fewer and the total wire length is 0.65% less after the jog removal, which means the technique does help to improve the quality of the routing solution.

Table 4.7: Test Results of Jog Removal on MCC1(w/o P&G)

No. of Vias			Wire Length		
Before	After	% Improved	Before	After	% Improved
4797	4694	2.19%	371959	369555	0.65%

Table 4.8 shows the results of applying the technique of Steiner point insertion for MCC1(w/o P&G). It lists the total number of vias and total wire length both before and after the processing and the percentage of improvement. The total number of vias is 0.32% fewer and the total wire length is 1.29% less after the Steiner point insertion. Although the improvement is small, the quality of the routing solution is better than before.

Table 4.8: Test Results of Steiner Point Insertion on MCC1(w/o P&G)

No. of Vias			Wire Length		
Before	After	% Improved	Before	After	% Improved
4694	4679	0.32%	371959	367214	1.29%

Table 4.9 shows the percentage of nets routed on each layer pair for MCC1(w/o P&G) by the MCG router. We can see from the table that most of the nets (more

than 89%) are routed on the first layer pair, which is an indication of the high routing density achieved by the MCG router.

Table 4.9: Percent Completed Routes After x Layers

Example	1-2	3-4
MCC1(w/o P&G)	89.47%	100.00%

Table 4.10 shows the via usage (excluding the stacked vias used to bring the terminals to their routing layers) for MCC1(w/o P&G) by the MCG router. We can see from the table that more than 73% of the nets use 2 or fewer interconnection vias to complete the routing and only a very small percentage (5.71%) of the nets use 4 interconnection vias. Furthermore, no more than 4 interconnection vias are used for the nets. The last column shows the average number of interconnection vias used for net connection. The average number of vias per net is 2.13, which shows that the MCG router uses a very small number of interconnection vias to complete the routing.

Table 4.10: Via Usage of Test Results for MCC1(w/o P&G)

Number (percentage) of Nets That Use					Avg. Vias
0 Vias	1 Vias	2 Vias	3 Vias	4 Vias	per Net
3(0.24)	234(18.81)	678(54.50)	258(20.74)	71(5.71)	2.13

Table 4.11 shows the data for the final routing solution for MCC1(w/o P&G) produced by the MCG router. The third column shows the lower bound of the total wire length as computed by Eq. 4.2. The fourth column shows the wire length actually used by the MCG router. The fifth column shows the ratio of the actual wire length to the lower bound. The MCG router used 8% more wire length than the lower bound. However, since there are many multi-terminal nets in MCC1, the

lower bound computed by Eq. 4.2 may actually be lower than the optimal wire length.

Table 4.11: Final Results for MCC1(w/o P&G)

No. of Layers	No. of Vias	Wire Length		
		Lower Bound	Actual Length	Ratio
4	4679	338955	367214	1.08

MCC1(P&G)

Table 4.12 gives the routing results for the the MCC1(P&G) by the MCG router before using the post-processing techniques. Tables 4.13, and 4.14 show the results of applying the technique of jog removal and Steiner point insertion for MCC1(P&G). Table 4.15 shows the percentage of nets routed on each layer pair for MCC1(P&G) by the MCG router.

Table 4.12: Routing Results for MCC1(P&G)

No. of Layers	No. of Vias	Total Wire Length
4	5964	383537

Table 4.13: Test Results of Jog Removal on MCC1(P&G)

No. of Vias			Wire Length		
Before	After	% Improved	Before	After	% Improved
5964	5747	3.64%	383537	378707	1.26%

Table 4.16 shows the via usage for MCC1(P&G) by the MCG router. We can see from the table that more than 80% of the nets use 2 or fewer interconnection

Table 4.14: Test Results of Steiner Point Insertion on MCC1(P&G)

No. of Vias			Wire Length		
Before	After	% Improved	Before	After	% Improved
5747	5742	0.09%	378707	376478	0.59%

Table 4.15: Percent Completed Routes After x Layers

Example	1-2	3-4
MCC1(P&G)	86.0%	100.00%

vias to complete the routing and only a very small percentage (3.60%) of the nets use 4 interconnection vias. And no more than 4 interconnection vias are used for the nets. The average number of interconnection vias per net is 1.90, which again shows that the MCG router uses a very small number of interconnection vias to complete the routing.

Table 4.16: Via Usage of Test Results for MCC1(P&G)

Number (percentage) of Nets That Use					Avg. Vias
0 Vias	1 Vias	2 Vias	3 Vias	4 Vias	per Net
132(7.79)	291(17.18)	940(55.49)	270(15.94)	61(3.60)	1.90

Table 4.17 shows the data for the final routing solution for MCC1(w/o P&G) produced by the MCG router and the results for the same example produced by other routers. The second column shows the number of layers used for the routing. The third column shows the number of vias, and the fourth column shows the ratio of the number of vias used by the router to the minimum number of vias among all the routers. The fifth column shows the wire length actually used by the routers,

and the sixth column shows the ratio of the actual wire length to the lower bound. The MCG router used the minimum number of layers among all the router for this example. Compared with the V4R router, the MCG router used 21.8% fewer vias and 5% less wire length. Compared with the V4CMRA router, the MCG router used 17.7% fewer vias and 1% less wire length. Compared with the SLICE router, the MCG router used 11.2% fewer vias and 7% less wire length. Compared with the 3D-Maze router, the MCG router used 53.2% fewer vias and 6% less wire length. In comparison with the Router X, the MCG router used 3.6% fewer vias and 1% less wire length. As shown in the table, MCG requires fewer vias and less wire length than that of all the other routers.

Table 4.17: Final Results for MCC1(P&G) and Comparison

Router	No. of Layers	No. of Vias	Ratio to Minimum	Total Wire Length	Ratio to Lower Bound*
MCG	4	5742	1.000	376478	1.10
V4R	4	6993	1.218	394272	1.15
V4CMRA	4	6757	1.177	381592	1.11
SLICE	5	6386	1.112	402258	1.17
3D Maze	5	8794	1.532	397221	1.16
X	4	5949	1.036	382151	1.11

* The lower bound for this example is 343767

4.2.2 Test Results for MCC2-75

There are two sets of test results for MCC2-75. There were fewer than 0.5% of the nets unrouted when Phase 3 of the routing process finished on the second layer pair. At that point, there were two choices; one was to add another layer pair and route the unrouted nets on them; the other was to try Phase 4 (type-Z routes). Both options were tested, which resulted in two different solutions. The

one without using type-Z routes ended up with 6 layers for the routing. And the other used only 4 layers with slightly more interconnection vias and longer wire length. The reduction of a pair of layers can significantly reduce the manufacturing costs. Therefore, the use of type-Z routes may be well justified. For convenience, the two routing solutions are referred to as MCC2-75(6) and MCC2-75(4).

MCC2-75(6)

Table 4.18 gives the routing results for the the MCC2-75(6) by the MCG router before using the post-processing techniques. Tables 4.19, and 4.20 show the results of applying the technique of jog removal and Steiner point insertion for MCC2-75(6). Table 4.21 shows the percentage of nets routed on each layer pair for MCC2-75(6) by the MCG router.

Table 4.18: Routing Results for MCC2-75(6)

No. of Layers	No. of Vias	Total Wire Length
6	33782	5512347

Table 4.19: Test Results of Jog Removal on MCC2-75(6)

No. of Vias			Wire Length		
Before	After	% Improved	Before	After	% Improved
33782	33301	1.42%	5512347	5461155	0.93%

Table 4.20: Test Results of Steiner Point Insertion on MCC2-75(6)

No. of Vias			Wire Length		
Before	After	% Improved	Before	After	% Improved
33301	33279	0.07%	5461155	5457063	0.07%

Table 4.21: Percent Completed Routes After x Layers

Example	1-2	3-4	5-6
MCC2-75(6)	67.86%	95.65%	100.00%

Table 4.22 shows the via usage for MCC2-75(6) by the MCG router. We can see from the table that more than 57% of the nets use 2 or fewer interconnection vias to complete the routing, and no more than 4 interconnection vias are used. The average number of interconnection vias per net is 2.45.

Table 4.22: Via Usage of Test Results for MCC2-75(6)

Number (percentage) of Nets That Use					Avg. Vias per Net
0 Vias	1 Vias	2 Vias	3 Vias	4 Vias	
8(0.11)	520(6.90)	3798(50.36)	2513(33.32)	702(9.31)	2.45

Table 4.23: Final Results for MCC2-75(6) and Comparison

Router	No. of Layers	No. of Vias	Ratio to Minimum	Total Wire Length	Ratio to Lower Bound
MCG	6	33279	1.000	5457063	1.02
V4R	6	36438	1.095	5559479	1.04
V4CMRA	6	33974	1.021	5429010	1.01
SLICE	7	47864	1.438	5902818	1.10
3D Maze	—	—	—	—	—
X	—	—	—	—	—

* The lower bound of this example is 5362181

Table 4.23 shows the data for the final routing solution for MCC2-75(6) produced by the MCG router and the results of the same example produced by other

routers. The 3D-Maze router and Router X failed to produce a routing solution for MCC2-75 because of the large memory requirement of these routers for this large example. The MCG router used the minimum number of layers among all the routers for this example. Compared with the V4R router, the MCG router used 9.5% fewer vias and 2% less wire length. Compared with the V4CMRA router, the MCG router used 2.1% fewer vias and 0.52% more wire length. Compared with the SLICE router, the MCG router used 44% fewer vias and 8% less wire length. The MCG router uses fewer vias than that of all the other routers. The MCG router is comparable to or better than all the routers with respect to wire length.

MCC2-75(4)

Table 4.24 gives the routing results for the the MCC2-75(4) by the MCG router before post-processing. Tables 4.25, 4.26 show the results of applying the technique of jog removal and Steiner point insertion for MCC2-75(4). Table 4.27 shows the percentage of nets routed on each layer pairs.

Table 4.24: Routing Results for MCC2-75(4)

No. of Layers	No. of Vias	Total Wire Length
4	34311	6106485

Table 4.25: Test Results of Jog Removal on MCC2-75(4)

No. of Vias			Wire Length		
Before	After	% Improved	Before	After	% Improved
34311	34290	0.06%	6106485	5696861	6.71%

Table 4.28 shows the via usage for MCC2-75(4) by the MCG router. We can see from the table that more than 50% of the nets use 2 or fewer interconnection vias to complete the routing, and only a very small percentage of the nets (1.32%)

use more than 4 interconnection vias. There are a few nets which use more than 5 and less than 9 vias because of using the type-Z routes. The average number of interconnection vias per net is 2.58.

Table 4.26: Test Results of Steiner Point Insertion on MCC2-75(4)

No. of Vias			Wire Length		
Before	After	% Improved	Before	After	% Improved
34290	34279	0.03%	5696861	5692953	0.07%

Table 4.27: Percent Completed Routes After x Layers

Example	1-2	3-4
MCC2-75(4)	72.65%	100.00%

Table 4.28: Via Usage of Test Results for MCC2-75(4)

Number (percentage) of Nets That Use					Avg. Vias
0 Vias	1 Vias	2 Vias	3 Vias	4 Vias	per Net
3(0.04)	430(5.70)	3510(46.55)	2513(33.32)	983(13.04)	2.58
5 vias	6 vias	7 vias	8 vias	9 vias	10+ vias
80(1.06)	7(0.09)	3(0.04)	12(0.16)	0(0.00)	0(0.00)

Table 4.29 shows the final routing solutions for MCC2-75(4) produced by the MCG router and the results of the same example produced by other routers. The MCG router used fewer layers than all the other routers for this example. Compared with the V4R router, the MCG router used 6.4% fewer vias and 2% more wire length. Compared with the V4CMRA router, the MCG router used 0.9% more vias and 5% more wire length. Compared with the SLICE router, the MCG router used 40% fewer vias and 4% less wire length. The vias usage of the MCG router is

comparable to the minimum produced by the V4CMRA router (only 0.9% more). Slightly more wire length is used than some of the other routers when MCC2-75 is compacted into 4 layers. This is quite acceptable since it is preferable to reduce the number of layers even if that results in a slight increase in wire length.

Table 4.29: Final Results for MCC2-75(4) and Comparison

Router	No. of Layers	No. of Vias	Ratio to Minimum	Total Wire Length	Ratio to Lower Bound*
MCG	4	34279	1.009	5692953	1.06
V4R	6	36438	1.073	5559479	1.04
V4CMRA	6	33974	1.000	5429010	1.01
SLICE	7	47864	1.409	5902818	1.10
3D Maze	—	—	—	—	—
X	—	—	—	—	—

* The lower bound of this example is 5362181

Table 4.30: Routing Results for MCC2-45

No. of Layers	No. of Vias	Total Wire Length
4	33873	9768083

4.2.3 Test Results for MCC2-45

Table 4.30 gives the routing results for MCC2-45 by the MCG router before using the post-processing techniques. Table 4.31 shows the results of applying the technique of jog removal for MCC2-45. Table 4.32 shows the results of applying the technique of Steiner point insertion for MCC2-45. Table 4.33 shows the percentage

of nets routed on each layer pairs for MCC2-45 by the MCG router. More than 90% of the nets are routed on the first layer pair.

Table 4.31: Test Results of Jog Removal on MCC2-45

No. of Vias			Wire Length		
Before	After	% Improved	Before	After	% Improved
33873	33308	1.67%	9768083	9139958	6.43%

Table 4.32: Test Results of Steiner Point Insertion on MCC2-45

No. of Vias			Wire Length		
Before	After	% Improved	Before	After	% Improved
33308	33283	0.08%	9139958	9129743	0.11%

Table 4.33: Percent Completed Routes After x Layers

Example	1-2	3-4
MCC2-45	90.37%	100.00%

Table 4.34: Via Usage of Test Results for MCC2-45

Number (percentage) of Nets That Use					Avg. Vias
0 Vias	1 Vias	2 Vias	3 Vias	4 Vias	per Net
5(0.07)	409(5.42)	3888(51.55)	2540(33.68)	699(9.27)	2.45

Table 4.34 shows the via usage for MCC2-45 by the MCG router. We can see from the table that more than 57% of the nets use 2 or fewer interconnection vias to complete the routing, and no more than 4 interconnection vias were used. The average number of interconnection vias per net is 2.45.

Table 4.35 shows the data for the final routing solution for MCC2-45 produced by the MCG router and the results of the same example produced by other routers. The 3D-Maze router and X router failed to produce a routing solution for MCC2-45 because of the large memory requirement of these algorithms for large examples. For the same reason, the SLICE router failed to complete the routing of MCC2-45, because SLICE uses a maze routing algorithm in order to reduce the need for too many layers to complete the routing. The MCG router used the same number of layers as the other routers for this example. Compared with the V4R router, the MCG router used 9.6% fewer vias and about the same wire length. Compared with the V4CMRA router, the MCG router used 2.2% fewer vias and 1% more wire length. In general, the MCG router uses fewer vias than the other routers. The wire usage is comparable to other routers (very close to the lower bound).

Table 4.35: Final Results for MCC2-45 and Comparison

Router	No. of Layers	No. of Vias	Ratio to Minimum	Total Wire Length	Ratio to Lower Bound*
MCG	4	33283	1.000	9129743	1.02
V4R	4	36473	1.096	9130705	1.02
V4CMRA	4	34026	1.022	9039996	1.01
SLICE	—	—	—	—	—
3D Maze	—	—	—	—	—
X	—	—	—	—	—

* The lower bound of this example is 8935372

4.2.4 Test Result for the Flip-Chip Example

Table 4.36 gives the routing results for the FLIP by the MCG router before using the post-processing techniques. Table 4.37 shows the results of applying the technique of jog removal for FLIP. Table 4.38 shows the results of applying the

technique of Steiner point insertion for FLIP.

Table 4.36: Routing Results for FLIP

No. of Layers	No. of Vias	Total Wire Length
4	5168	365645

Table 4.37: Test Results of Jog Removal on FLIP

No. of Vias			Wire Length		
Before	After	% Improved	Before	After	% Improved
5168	5104	1.24%	365645	362202	0.94%

Table 4.38: Test Results of Steiner Point Insertion on FLIP

No. of Vias			Wire Length		
Before	After	% Improved	Before	After	% Improved
5104	5088	0.31%	362202	359683	0.70%

Table 4.39: Percent Completed Routes After x Layers

Example	1-2	3-4
FLIP	83.52%	100.00%
MCC1	89.47%	100.00%

Table 4.39, 4.40, and 4.41, show the percentage of nets routed on each layer pair, the via usage, and the final routing solutions for FLIP. For comparison, the test result for MCC1(w/o P&G) are also listed because the FLIP example is created by modifying the terminal locations of MCC1(w/o P&G). Table 4.41 shows that the test results for FLIP are consistent with the test results for MCC1(w/o P&G), although the FLIP case uses a few more interconnection vias to complete the

routing. The average interconnection vias per net for FLIP is 2.37 whereas it is 2.13 for MCC1(w/o P&G). However, there is no significant performance decrease in the FLIP example, which indicates that the MCG algorithm can handle the flip-chip bounding technology.

Table 4.40: Via Usage of Test Results for FLIP

Example	Number (percentage) of nets that use					
	0 vias	1 vias	2 vias	3 vias	4 vias	5 vias
FLIP	1(0.08)	43(3.46)	795(63.91)	306(24.60)	94(7.56)	5(0.40)
MCC1	3(0.24)	234(18.81)	678(54.50)	258(20.74)	71(5.71)	0(0.00)

Table 4.41: Final results for FLIP

example	no. of layers	no. of vias	wire length		
			lower bound	actual length	ratio
FLIP	4	5088	329731	359683	1.09
MCC1	4	4679	338955	367214	1.10

4.2.5 Comparison with Other Routers

The comparison is made in terms of the number of layers used by the routers, total number of vias, total wire length and the running time.

1. Number of Routing Layers

Table 4.42 shows the number of layers used by MCG, V4R, SLICE, 3D-Maze, V4CMRA and Router X to complete the routing of MCC1, MCC2-75 and MCC2-45. The MCG router always used the minimum number of layers among all the routers for all the examples, especially in the case of MCC2-75 where MCG used 4 layers while the others used 6 or more layers.

Table 4.42: Comparison with Other Routers on Number of Layers

Example	Number of Layers					
	MCG	V4R	V4CMRA	SLICE	3D Maze	X
MCC1	4	4	4	5	5	4
MCC2-75	4	6	6	7	—	—
MCC2-45	4	4	4	—	—	—

2. Total Number of Vias

Table 4.43 shows the total number of vias used by MCG, V4R, SLICE, 3D-Maze, V4CMRA and X to complete the routing of MCC1, MCC2-75 and MCC2-45. The total number of vias includes both the number of stacked vias used for bringing the terminals to their proper routing layers and the number of vias used for connecting the net. The MCG router always used fewer vias than the others.

Table 4.43: Comparison with Other Routers on Total Number of Vias

Example	Total Number of Vias					
	MCG	V4R	V4CMRA	SLICE	3D Maze	X
MCC1	5742	6993	6757	6386	8794	5949
MCC2-75	33279*	36438	33974	47864	—	—
MCC2-45	33283	36473	34026	—	—	—

* It will be slightly more if only 4 layers are used

3. Total Wire Length

Tables 4.44 and 4.45 show the total wire length used by MCG, V4R, SLICE, 3D-Maze, V4CMRA and X to route MCC1, MCC2-75 and MCC2-45 and the ratio of the total wire length to the lower bound of the wire length.

For MCC1, the MCG router uses less wire than other routers. For MCC2-75

and MCC2-45, the wire length used by MCG is very close to the lower bound and comparable to the minimum produced by V4CMRA.

Table 4.44: Comparison with Other Routers on Total Wire Length

Example	Total Wire Length					
	MCG	V4R	V4CMRA	SLICE	3D Maze	X
MCC1	376478	394272	381592	402258	397221	382151
MCC2-75	5457063*	5559479	5429010	5902818	—	—
MCC2-45	9129743	9130705	9039996	—	—	—

* It will be slightly more if only 4 layers are used

Table 4.45: Comparison with Other Routers on Ratio to Lower Bound

Example	Ratio to Lower Bound					
	MCG	V4R	V4CMRA	SLICE	3D Maze	X
MCC1	1.10	1.15	1.11	1.17	1.16	1.11
MCC2-75	1.02	1.04	1.01	1.10	—	—
MCC2-45	1.02	1.02	1.01	—	—	—

4. Percentage of Nets Completed On First Layer Pair

The percentage of nets routed on the first layer pair is a reflection of the routing density that a router can achieve. Table 4.46 show the percentage achieved by the MCG router together with that of the V4R router. The data about the routed percentage on the layer pairs of the V4R router was reported in [25] and was absent in their later papers [24][27]. Therefore, this is the only data available for comparison.

The MCG router can route a high percentage of total number of nets on the first layer pair, which can help to reduce the total number of layers for the routing in cases such as MCC2-75.

Table 4.46: Percentage Routed on First Layer Pair

Example	Percentage Routed	
	MCG	V4R
MCC1	86.00%	74.30%
MCC2-75	72.73%	56.66%
MCC2-45	90.37%	N/A*

* Data not provided

5. Run Time

The run time of the routers is difficult to compare because they were implemented on different platforms. Table 4.47 shows the platforms which the routers were implemented on and Table 4.48 shows the run time of the routers to complete the routing for the MCC designs.

Table 4.47: Platforms On Which the Routers Were Implemented

Router	MCG	V4R	V4CMRA	SLICE	3D Maze	X
Platform	Sparc-10	Sparc-2	Sparc-2	Sparc-2	Sparc-2	HP735

Table 4.48: Comparison with Other Routers on Run Time

Example	Run Time (hr:min:sec)					
	MCG	V4R	V4CMRA	SLICE	3D Maze	X
MCC1	0:08:00	0:03:00	0:00:38	0:12:00	0:59:00	41:14
MCC2-75	1:52:00*	1:06:00	0:11:54	8:15:00	—	—
MCC2-45	1:48:00	1:37:00	0:11:48	—	—	—

* It will be more if only 4 layers is used

The MCG router is not as fast as V4R and V4CMRA. It is at about the same

speed as SLICE. And it is certainly faster than the routers based on maze routing algorithms. The routing algorithm of V4CMRA described in [32] is a revised version of the V4R algorithm which assigns the tracks based on a predetermined ordering of nets instead of the optimized track assignment found in V4R, which may contribute to the speed of the algorithm. The elimination of the optimized track assignment may result in lower routing density which in turn may increase the number of layers needed for the routing in cases where all layers are densely routed. Although MCG is not as fast as V4R type routers, the improved quality of the routing solutions and the flexibility for incorporating electrical constraints [16] make it well worth the additional computation time. Another fact worth mention is that only the MCG router produced a better routing solution for MCC1 than that of the state-of-the-art industrial PCB/MCM Router X.

4.3 Test Results for Single-Layer Routing

The MCG router can also perform single-layer routing with minor changes in the algorithm. The single-layer version of the MCG router was tested with the examples M^{2}R-100 and M^{2}R-200. The comparison can be made with the results of the M^{2}R router. Table 4.49 shows the test results for the MCG and M^{2}R routers on M^{2}R-100 and M^{2}R-200. It only lists the routing results on the first layer because only the test results of the M^{2}R router on the first layer were available.

Table 4.49: Comparison on Single-Layer Routing

Examples	% Routed		No. of Jogs		Wire Length	
	MCG	M^{2}R	MCG	M^{2}R	MCG	M^{2}R
M^{2}R-100	43%	38%	135	191	1111	1420
M^{2}R-200	39%	34%	221	483	3255	3715

Compared with the M^{2}R router, the MCG router routed 5% more nets on the

first layer while using on average 80% fewer jogs and 21% less wire length for M²R-100 and M²R-200. The MCG router performs the routing simultaneously while the M²R takes a net-by-net approach. The above results show the advantage of the simultaneous routing methodology over the net-by-net routing technique.

4.4 Some Implementation Issues

In this section, some of the implementation issues such as the data structure representing the compatibility graph, the storage organization for obstacles on the routing layers and the impact of different implementation methods on the performance of the MCG router are described.

4.4.1 The Data Structure Representing the Compatibility Graph

There are several commonly used data structures for representing graphs. Adjacency matrices and adjacency lists are two such representations. Adjacency lists are usually used in sparse graphs and adjacency matrices are more often used for dense graphs. For the implementation of the MCG algorithm, either approach can be used.

For the MCG router, it needs to store the compatibility graph for the routing problems. Assume the total number of nets is n and each net has c candidate routes, then there are $c \times n$ vertices in the graph. For the edges in the graph, the worst case is a complete graph, where the number of edges would be $1/2c^2n^2$. The storage requirement for adjacency matrices form can be estimated accurately given the number of nets, n, and the number of candidate routes for each net, c. However, the memory requirements for the storage of the graph in the form of an adjacency list is not easy to estimate. It depends on the actual number of edges in the graph, which is variable in different iterations of the routing process.

4.4.2 The Storage Organization for Obstacles

The obstacles on the routing plane are the vias used for bringing the terminals to their proper routing layers and the line segments of routes which are routed in the previous iterations. The storage of the obstacles can be in two forms; one is a linked list form and the other is a table form.

The linked list form is the linked lists of line segments indexed by the row or column coordinates of the line segments (a via can be seen as a short line segment starting at one grid point and ending at the same grid point). To be specific, all the line segments on a particular row or column are linked to form a list and the list is indexed by the coordinate of the row or column. If we want to find if a certain grid point is currently open or not, we have to traverse the list with the corresponding index. The memory requirement is linear to the number of line segments, which in turn is linear to the number of nets because the number of line segments for each route is no more than 6 (disregarding the type-Z routes which are rarely used). However, the time required for the lookup of the open points depends on the length of the linked list.

On the other hand, we can store the obstacles in a table with each grid point on the routing plane corresponding to an entry in the table. Therefore, the time needed for the lookup is just 1, which could speed up the routing process. But the memory requirement for the table is large, because it requires storing an entire layer of the routing grid.

Both forms of the storage of the obstacles were implemented. The table lookup method cut the run time in half.

4.5 Complexity Analysis

In this section, the space and time complexity of the MCG routing algorithm are analyzed.

4.5.1 The Space Complexity

The space needed for running the algorithm includes the storage for the nets, the candidate routes, the compatibility graph and the obstacles. The space for storage of the nets is proportional to the total number of nets. The storage for the candidate routes is the number of bytes of memory for each route times the total number of candidate routes. Therefore, the space for the storage of candidate routes is also proportional to the total number of nets. For the storage of vertices in the compatibility graph, the total number of vertices is equal to the total number of candidate routes, which again is proportional to the total number of nets. However, the memory requirement for the edges is proportional to the square of the total number of vertices in the worst case. Therefore, the space requirement for the edges is $O(n^2)$ of the total number of nets. The memory requirement for the storage of obstacles depends on the forms in which the obstacles are stored. If the obstacles are stored in the form of a linked list, then the space for the storage of obstacles is proportional to the total number of line segments, thus proportional to the total number of nets. On the other hand, the table form for the storage of obstacles requires storing an entire layer of the routing grid which is $O(L \times W)$ where L is the number of rows and W is the number of columns in the routing plane.

4.5.2 The Time Complexity

The time needed for the routing process includes the time for construction of candidate routes, the time for building the compatibility graph, and the time for reduction of the graph.

1. Time for Construction of Candidate Routes

The time for the construction of candidate routes depends on the type of route to be constructed. For the worst case, assume there is no obstacle inside the bounding box of the net and that every possible candidate route must be constructed. (This is an extreme and highly unlikely case but serves as a worst-case analysis.)

The time needed to construct every possible candidate route is the following. For type-0 and type-1 routes, the time is constant. For type-2 routes, it is proportional to $h + w$ where h is the length and w is the width of the bounding box of the net. For the type-3 routes, it is $O(hw)$. For the type-4 routes, it is proportional to $hw(h+w)$. For type-5 routes, it is $O(h^2w^2)$. For type-X, it is similar to the type-4 route only with a larger bounding box. For the type-Z routes, it is proportional to the number of line segments generated by the line-probe algorithm. Except for the type-Z routes, type-5 routes are the most time consuming to construct and they use the largest number of vias for the interconnection of the net. Therefore, the use of type-5 routes should be limited.

The time complexity described above is the worst case with the assumption of no obstacle in the bounding box and that all the possible routes are constructed, which is far from reality. First, there are obstacles which will cut the number of possible routes significantly. Secondly, there is no need to find all the possible routes because the algorithm only needs a small number of candidates (2 - 20) to find high quality solutions. Therefore, the average time complexity would be significantly less.

2. Time for Building the Compatibility Graph

The time complexity of building the compatibility graph is obviously on the order of $O(n^2)$ because every pair of candidate routes are to be tested for compatibility. The number of possible combination of pairs of candidate routes is $1/2cn(cn-1)$. Considering that the candidate routes from the same net should not be tested, the number of possible combinations will drop to $1/2c^2n(n-1)$. But it is still $O(n^2)$.

3. Time for the Reduction of the Graph

The worst case for the reduction of the graph is that the graph is a complete graph which requires $O(n^2)$ time to reduce the graph.

4.6 Selection of the Number of Candidate Routes

In the previous section, it is shown that the complexity of the MCG routing algorithm is directly linked to the number of candidates, c. Therefore, the number of candidate routes for each net plays an important role in the memory requirements and run time of the algorithm. Intuitively, it is expected that a large value for c would be desirable. However, a large value of c will dramatically increase the memory requirement and the run time. Counter to intuition, it will be shown that it is not necessary.

Since the MCG algorithm is iterative, it can be run with a small value for c for multiple iterations, which may yield a solution as good as that of a large number of c, meanwhile it does not require a large amount of memory and extra long runtime. Phase 1 of the routing process was tested on M^2R-100, M^2R-200, MCC1 and MCC2-75 with several different values for c. In each case, the process was repeated until no more nets could be routed.

Table 4.50: Test Results of Different Number of Candidates for M^2R-100

No. of Candidate	No. of Iteration	Run Time (second)	No. of Nets Routed	Percent Routed
2	2	0.69	132	65.02%
4	1	0.70	134	66.01%
8	1	1.25	138	67.98%
12	1	1.49	138	67.98%
16	1	1.76	138	67.98%
32	1	3.22	138	67.98%

Tables 4.50 and 4.51 show the test results for the different number of candidate routes on M^2R-100 and M^2R-200, respectively. The first column shows the number of candidate routes. The second column shows the number of iterations of the

routing process. The third column shows the run time, and the fourth and fifth columns show the number of nets routed and the percentage of the total number of nets.

Table 4.51: Test Results of Different Number of Candidates for M^2R-200

No. of Candidate	No. of Iteration	Run Time (second)	No. of Nets Routed	Percent Routed
2	2	2.17	237	77.70%
4	2	3.03	249	81.64%
8	1	4.39	254	83.28%
12	1	6.63	255	83.61%
16	1	8.92	256	83.93%
32	1	17.04	256	83.93%

The number of candidate routes for M^2R-100 and M^2R-200 ranges from 2 to 32. The routing process with a larger number of candidate routes tends to require fewer iterations before exhaustion. As the number increases, the run time increases. The routing process with a small number uses less time, and the routing process with a large number needs a lot more time to complete. The memory requirement, of course, also increases with the number of candidate routes. However, there is not a big difference for the number of nets routed with a different number of candidate routes. For M^2R-100, the largest number of candidate routes, 32, only routes 2.96% more nets than that of the smallest number, 2; compared with $c = 12$, it does not route any more. For M^2R-200, the largest number of candidate routes, 32, routes 6.23% more nets than that of the smallest number, 2; while compared with $c = 12$, it only routes 0.32% more. The run time and percentage routed with different number of candidates are shown in Figs. 4.1, 4.2 and 4.3.

Table 4.52 shows the test results of different number of candidate routes on MCC1. The number of candidate routes ranges from 2 to 32. The routing process

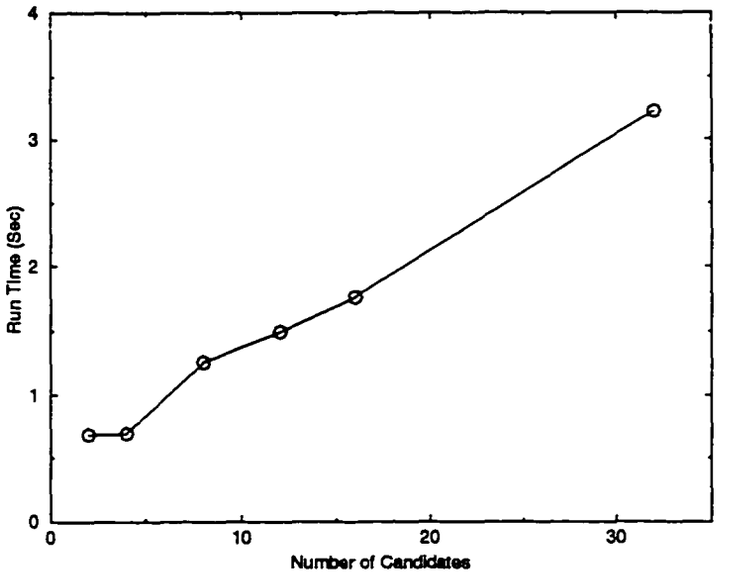


Figure 4.1: Run Time vs. Number of Candidates(M^2R-100)

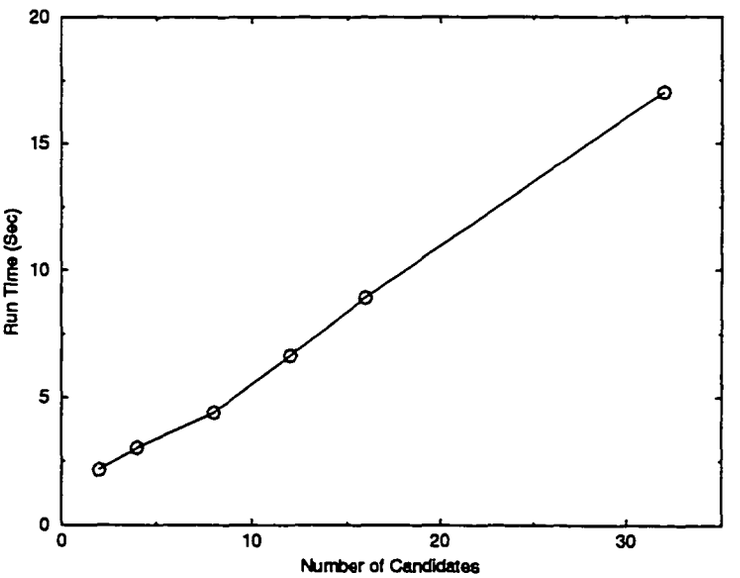


Figure 4.2: Run Time vs. Number of Candidates(M^2R-200)

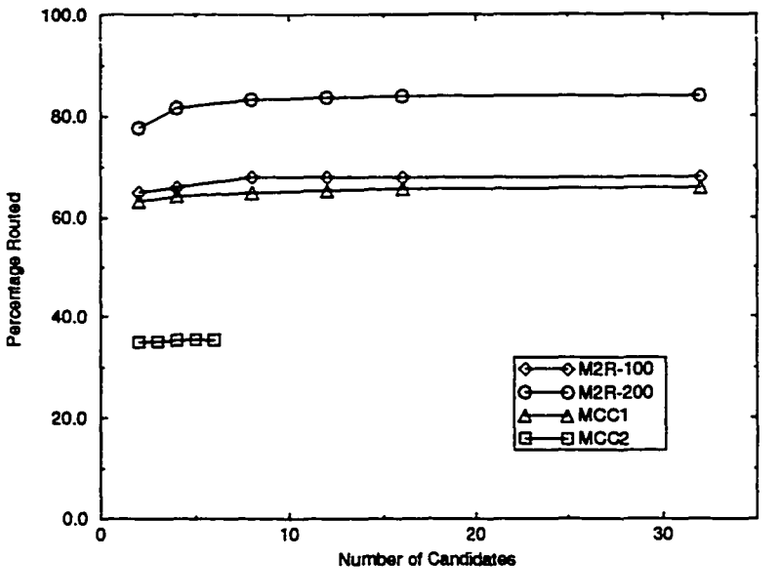


Figure 4.3: Percentage Routed vs. Number of Candidates

using a larger number of candidate routes tended to require fewer iterations before exhaustion. As the number increases, the run time decreased and then began increasing. The routing process using a small value for c used less time per iteration, but it involved more iterations. Therefore, the total time used tended to be more. On the other hand, for a large value of c there were fewer iterations, but each iteration required a lot more time to complete. The routing process with a medium value of c, such as 8, achieved a compromise and thus lower run time. Note, there is not a big difference for the number of nets routed with different numbers of candidate routes used. The largest number of candidate routes, 32, only routed 2.58% more nets than that of the smallest number, 2. Compared with the medium number, 12, it only routed 0.49% more. And this large number uses much more memory and runs about 4 times slower. Therefore, there is hardly any advantage using a large number of candidate routes on this benchmark. Similar results were seen for other examples as well. The run time and percentage routed with different number of candidates are shown in Fig. 4.4 and 4.3.

Table 4.52: Test Results of Different Number of Candidates for MCC1

No. of Candidate	No. of Iteration	Run Time (second)	No. of Nets Routed	Percent Routed
2	12	185	787	63.26%
4	7	138	800	64.31%
8	3	133	808	64.95%
12	3	197	813	65.35%
16	3	284	818	65.76%
32	2	732	819	65.84%

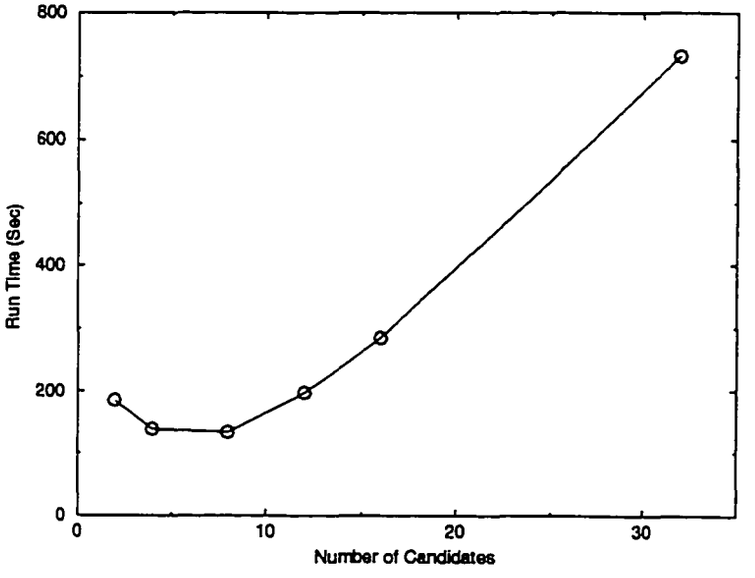


Figure 4.4: Run Time vs. Number of Candidates(MCC1)

Table 4.53: Test Results of Different Number of Candidates for MCC2

No. of Candidate	No. of Iteration	Run Time (second)	No. of Nets Routed	Percent Routed
2	17	5007	2635	34.94%
3	12	3782	2646	35.08%
4	9	3173	2666	35.35%
5	8	2910	2670	35.41%
6	6	2564	2669	35.39%

Due to the time and memory requirements, a smaller range of values (from 2 to 6) was studied for MCC2-75. From the test results, we can still see similar behavior to that for MCC1. The routing process using a larger number of candidate routes tended to have less iterations before exhaustion. The routing process using a small value for c involved more iterations. Therefore, the total time used tended to be more. Reasonably, we can estimate that the routing process with a large number of candidate routes needs more time because each iteration needs a large amount of time to complete, even though it involves fewer iterations. Here again, we see that there is not a big difference for the number of nets routed with different numbers of candidate routes. The largest number of candidate routes, 6, only routed 0.45% more nets than that of the smallest number, 2. Again, we hypothesize that a number larger than 6 might route only a few more nets with the cost of a larger amount of memory and significantly longer run time. The run time and percentage routed with different numbers of candidates are shown in Fig. 4.5 and 4.3.

From the above test data, it is seen that the performance of the MCG algorithm is not very sensitive to the number of candidate routes and a relatively small value for c can produce solutions with quality as high as when larger values for c are used. In addition, the routing process using relatively small values for c avoids large memory requirements and saves computation time. Note, there is no strict

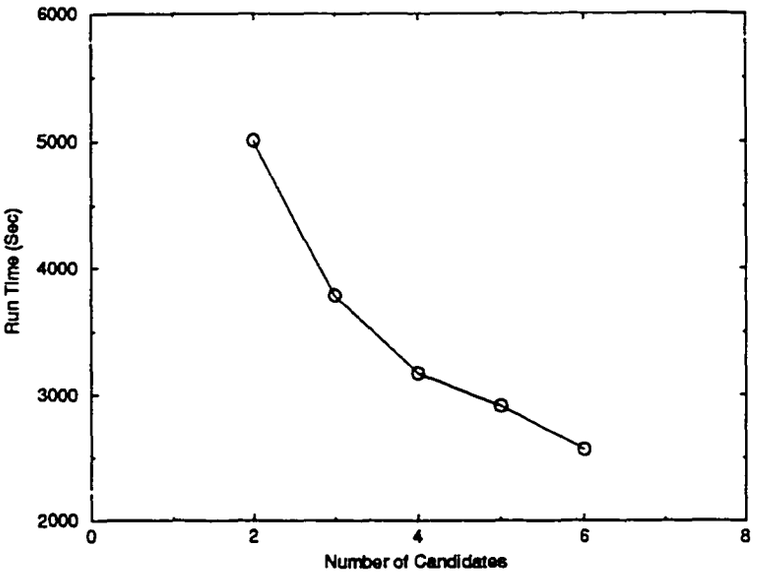


Figure 4.5: Run Time vs. Number of Candidates(MCC2)

rule for the selection of the number of candidate routes. The selection could be based on the size of the problem (number of nets, etc.) and the underlying system architecture (memory, cpu, etc). Again, note that the routing solution is not very sensitive to the number of candidate routes, therefore, any reasonable number will work fine. This robustness is an important feature of the algorithm.

It is proposed that the best way to perform the routing process is to start with a relatively small number of candidate routes. As the process goes on, more and more nets get routed and the nets unrouted become fewer, then the number of candidate routes can be gradually increased. Therefore, we can avoid the large memory requirement meanwhile reducing the number of iterations at the same time. Actually, when the routing for MCC1 and MCC2 was performed, a variable number of candidate routes was used (5 - 20 candidate routes were used for MCC1 and 2 - 10 candidate routes were used for MCC2).

Another strategy to reduce the run time is to stop the iteration of the routing

process when the number of nets routed in the iteration is very small (0.5% of the total number of nets), except for the last layer or layer pair. The reason for this is that it is observed that the vast majority of nets are routed in the first few iterations and each additional iteration thereafter only adds a small number of routed nets. By stopping it, we can save a lot of run time with only a small decrease in routing density for this layer pair. Again, this would not in general cause the use of more layers to complete the routing because of the fact that most of the nets are routed on the top layer pairs and the bottom layer pair is usually very sparse. Therefore, there is room available for this small number of unrouted nets on the last layer pair. However, in the situation where the goal is to pack the routing solution into fewer layers, it is advantageous to run the process until as many nets are routed as possible on each layer pair. By doing this, we may reduce the number of layers used for the routing in exchange for the cost of run time. It depends on the objective of the designer. The algorithm itself is flexible to facilitate the different design goals.

CHAPTER 5

LAYER BALANCING AND RE-ROUTING

5.1 Layer Balancing

After the routing is finished, the nets have been routed in a number of layer pairs. It can be noted that the nets are usually very unevenly distributed among the layer pairs; the top layer pairs tend to have many more nets routed on them than that of the lower layer pairs. Most of the nets are routed on the first layer pair (90% in one example and 73% in another). There are only a few nets routed on the last layer pair (10% and 0.5% respectively). Hence the top layer pairs have much higher routing density and the bottom layer pairs have a lot of unoccupied open area. High density routing helps to reduce the number of layers needed for routing, thus reducing the total cost. On the other hand, it may introduce some electrical and manufacturing problems. Also, it is more likely to cause potential crosstalk and certainly is more difficult to fabricate, resulting in lower yields.

Actually it is possible to make use of the unoccupied open area on the bottom layer pairs by deleting some of the nets routed on the top layer pairs and routing them on the bottom layer pairs. In this way, we will relieve the high density of the top layer pairs and distribute the wiring more uniformly over all the layers, without increasing the total number of layers. The process of moving nets from the top layers to bottom layers is referred as layer balancing. We introduce a new algorithm for the problem of layer balancing as described below.

Step 1: Routing

This step is the same as the routing process described in Section 3.5 and Section 3.6, except that the locations of all the terminals of nets are seen as obstacles in all the layer pairs. In the routing process described previously, only the unrouted

nets are propagated to the next layer pair and the terminals of the nets which are routed in the previous layer pairs are not seen as obstacles. The reservation of the position of the terminals in all layers will make the next step possible.

Step 2: Redundant routing

After Step 1, all the nets have been routed in various layer pairs. In this step, all nets routed on a particular layer pair are seen as obstacles. The algorithm then attempts to route all other nets on this layer pair. The same algorithm is repeated for each layer pair. After this redundant routing process, we have multiple routes for some nets but on different layer pairs, which provides a basis for the next step – balancing.

Step 3: Balancing

After Step 2, some of the nets will have more than one route but on different layer pairs. For each of those nets with multiple routes, we need to select only one route for the final implementation. The criteria for selection is reducing the routing density.

Before we introduce the selection algorithm, we define the weighted congestion graph G = (V, E) where V is a set of vertices representing the routes of the nets on the given layer pair and E is a set of weighted edges. The weight w_{ij} for edge e_{ij} is determined as follow:

Procedure Weight_of_Edge

Input:

route r_i and route r_j

Output:

w_{ij} – the weight associated with edge e_{ij}

Begin

$w_{ij} = 0$

For each line segment s_i in r_i

For each line segment s_j in r_j

If s_i and s_j are parallel to each other

Ovlp = overlap between s_i and s_j

Dist = distance between s_i and s_j

w_{ij} += Ovlp / Dist

EndIf

EndFor

EndFor

End

The weight w_{ij} is referred to as the congestion weight, which reflects the closeness between the two routes. For each vertex v_i in V, there is a weight w_i associated with it, which is the total of the congestion weights of all the edges incident to v_i. The w_i is determined by the following procedure:

Procedure Weight_of_Vertex

Input:

r_i – route associated with vertex v_i

Output

w_i – the weight associated with vertex v_i

Begin

$w_i = 0$

For each e_{ij} incident to v_i

w_i += w_{ij}

EndFor

End

The congestion graph is constructed as follow:

Procedure Congestion_Graph

Input:

all the routes on a given layer pair from Step 2

Output:

the congestion graph

Begin

For each route r_i

associate a vertex v_i with r_i

EndFor

For each vertex v_i

For each vertex v_j

add edge e_{ij}

w_{ij} = Weight_of_Edge(r_i, r_j)

EndFor

w_i = Weight_of_Vertex(r_i)

EndFor

End

The balancing algorithm first constructs a congestion graph for each layer pair, and then finds the vertex with the maximum weight among vertices in the congestion graphs of all the layer pairs. If the vertex is the only one or last one remaining for the net, the route associated with the vertex is selected for the net. Otherwise, it is deleted and the graph is modified accordingly. Ties are broken by the layer pair on which the route was originally located. The process is repeated until all the nets are routed.

Algorithm Layer_Balancing

input:

routing result from step 2

Output:

final routing solution

Begin

For each layer pair

construct a congestion graph Gi = (Vi, Ei)

EndFor

Repeat

find the vertex v_{max} with the maximum weight

If v_{max} is the only one or last one left for the net

The route associated with v_{max} is selected for the net

Else

Delete the v_{max}

Modify the graph (adjust the weights)

EndIf

Until every net is routed

End

Previous research on layer assignment before routing was discussed in Chapter 2, and it was pointed out that it suffered two drawbacks. First, the number of layers needed for the routing must be estimated for the layer assignment, and the exact number of layers needed for the routing is not known before the routing is finished. The estimation may not be accurate; if more than needed, it increases the cost, on the other hand, if less, there will be routability problems. Second, after the nets are assigned to a certain layer, it does not guarantee that the net can be routed on that layer. In the worst case, the nets are not all routable. So the process has to be re-run all over again with a different number for the layers needed. Our algorithm does not increase the number of layers, it is not subject to the routability problem and it yields a high degree of uniformity of distribution of nets among the layers and a great reduction in the congestion of wiring on the top layer pairs.

The down side of using the layer balancing algorithm is that the routing solutions produced by the algorithm use more vias and wire length than that of the routing solutions without using layer balancing because of the reservation of the positions of all pads in all layers. This introduces more obstacles in the routing process. The computation time is certainly longer because of the time needed for the redundant routing on each layer pair.

5.2 Re-Routing

Layer balancing distributes the nets evenly among a number of layer pairs. However, in a given layer pair, the wires are not necessarily well distributed. Some areas may have more wires and others have less. The congestion graph can be used to adjust the wires on a given layer pair to further reduce the congestion and reduce the total number of vias and total wire length. We call this process re-routing. The algorithm works as follows. It constructs the congestion graph for the given layer pair as described in the previous section. The weight of a vertex in the congestion graph reflects the overall closeness of the route associated with the vertex to all the other routes. Higher weights means that the route is closer to others. Now the goal is to adjust the route to reduce the weight and hence reduce the congestion. For a given net routed on the layer pair, candidate routes can be constructed in the unoccupied area with all the routes on the layer pair as obstacles. The types of route used for candidate routes this time are limited to type-0, type-1, type-2, and type-3 routes, provided the candidate routes use less than or equal to the number of vias that the original route for the net uses. For example, if the original route for the net has two vias, we can only use type-0, type-1 and type-2 routes to construct the candidate routes. The reason for this restriction is that we want to reduce vias and wire length while reducing the congestion. These types of routes will help us to achieve the goal because they are all minimum length routes and they use a small number of vias. After we find the candidate routes, we can compute the congestion weight for all these candidate routes and find the one with minimum weight. Ties are broken using the number of vias since the goal is to reduce the total number of vias. We can compare the weight of the new route with the weight of the old route. If it is smaller than the old one, replace the old route with the new route. This procedure is repeated for every net. In the end, some nets are re-routed with less congestion weight and/or fewer vias; others remain unchanged. The overall effect will be less congestion, fewer vias, and shorter wire length for the routing solution.

Algorithm Re-Routing

input:

routing solution for a given layer pair

Output:

re-routed routing solution

Begin

construct a congestion graph Gi = (Vi, Ei)

For each net routed on the layer pair

Construct the candidate routes with the preferred types of routes

For each candidate route r_i

Calculate the congestion weight w_i

EndFor

Find the r_{min} with the minimum weight w_{min} for all the r_i

If w_{min} < the congestion weight of the route for the net

Replace the route with r_{min}

Modify the graph (adjust the weights)

EndIf

EndFor

End

5.3 Test Results of Layer Balancing

Layer balancing is a new technique which tries to uniformly distribute the nets among the routing layers and reduce the wiring congestion while not increasing the number of routing layers. Tables 5.1 and 5.2 show the results of applying the layer balancing algorithm on MCC1 and MCC2-75, respectively. The solution with three layer pairs was used for MCC2-75 to test the quality of the layer balancing algorithm on multiple (more than two) layer pairs. The tables list the number of nets routed on certain layer pairs and their percentage of the total number of nets both before and after applying the layer balancing technique. From the tables, it is seen that the algorithm distributes the nets quite evenly among the layer pairs.

Table 5.1: Test Results of Layer Balancing on MCC1

Layer	Nets Routed			
	before	percent	after	percent
1 - 2	1116	89.71%	633	50.88%
3 - 4	128	10.29%	611	49.12%

Table 5.2: Test Results of Layer Balancing on MCC2

Layer	Nets Routed			
	before	percent	after	percent
1 - 2	5479	72.73%	2555	33.88%
3 - 4	1732	22.89%	2498	33.13%
5 - 6	330	4.38%	2488	32.99%

As expected, the routing solutions produced by the layer balancing algorithm used slightly more vias and wire length than without using it. Table 5.3 shows the comparison between the results with and without applying the layer balancing algorithm on MCC1 and MCC2-75.

Table 5.3: Comparison with and without Layer Balancing

Example	Total No. of Vias			Total Wire Length		
	with	without	% increase	with	without	% increase
MCC1	5090	4880	4.30%	377660	372594	1.36%
MCC2	36025	33301	8.18%	5673032	5461155	3.88%

5.4 Test Results of Re-Routing

The technique of re-routing introduced in this chapter tries to place the routes more evenly on the given layer. In this section, the test results of applying the re-routing technique on MCC1 and MCC2 after layer balancing are given. It is difficult to show how much more evenly the routes are placed after the re-routing process than before because there is no quantitative measure for the distribution of the routes on a layer. We can show the total number of vias and the wire length. The improvement of re-routing on MCC1 and MCC2 is on average 3.32% fewer vias and 0.15% less wire length.

Table 5.4: Test Results of Re-routing on MCC1

Layer	No. of Vias			Wire Length		
	before	after	% improved	before	after	% improved
1 - 2	2508	2441	2.67%	185374	185228	0.08%
3 - 4	2488	2336	6.11%	184512	184123	0.21%

Table 5.5: Test Results of Re-routing on MCC2

Layer	No. of Vias			Wire Length		
	before	after	% improved	before	after	% improved
1 - 2	12007	11672	2.79%	1803997	1750554	2.96%
3 - 4	12241	11940	2.46%	2082979	2037120	2.20%
5 - 6	11777	11472	2.59%	1786056	1745719	2.26%

The effectiveness of re-routing is dependent on the routing density of the given layer. If the given layer is routed very densely, there is not much room left for the re-routing to adjust the routes, and the improvement of re-routing will be minimal. Therefore, it is more effective to apply it after layer balancing due to the lower routing density.

CHAPTER 6

CONCLUSION

A new multilayer, general-area, multichip module routing algorithm named MCG was presented. The algorithm differs from other MCM routers in the way that the routes interconnecting the nets are constructed. Some routers perform the routing net-by-net and others extend the routes piece-by-piece during the routing process. The MCG router takes a more global approach by constructing a certain number of candidate routes for each of the nets, building a compatibility graph for the candidate routes and reducing the graph to yield a routing solution. Due to this unique way of performing the routing, the MCG router offers several outstanding features. First, it performs the routing of the nets simultaneously. Therefore it is not subject to the net ordering problem. Second, it can give the designer the flexibility of selecting the topology of routes for the nets to be routed, which is almost impossible for other algorithms. Thirdly, it offers a natural way to incorporate the electrical constraints into the routing process, which is absent or hard to handle in other algorithms. Compared with other MCM routers, the MCG router produced better quality routing solutions in terms of number of layers, number of vias, total wire length and routing density with significantly improved computation times over a well known commercial router.

A unique post-routing processing technique, named layer balancing, is also introduced, which can distribute the nets uniformly among the routing layers without increasing the number of routing layers. The layer balancing can reduce the congestion of wiring, therefore reducing the potential for crosstalk and the difficulties of fabricating high density wiring which may result in low fabrication yields. The re-routing technique can also be used to improve the quality of routing solutions.

There are several areas for future work with regard to this project. For example, the MCG router handles multi-terminal nets in such a way that they are decom-

posed into several two terminal nets before the actual routing process and the two terminal nets are then routed independently during the routing process. Although applying the technique of Steiner point insertion in the post-processing may help to improve the routing solution for the multi-terminal nets, this method is obviously not the optimal way to handle multi-terminal nets. Ideally, multi-terminal nets would be treated as one net and the candidate routes would be constructed taking this into account. However, the construction of candidate routes for multi-terminal nets is not as simple as that for two terminal nets. Different multi-terminal nets have different numbers of terminals, and certainly the topology of candidate routes would be different. The relative location of the terminals in multi-terminal nets again makes a difference for the topology of the candidate routes. The problem of finding an efficient way to construct candidate routes for multi-terminal nets is a well justified research topic.

Currently, the MCG router requires that all the wires for the interconnection of the nets run on the grid and the separation between the wires is fixed. However, there are times when off-grid wiring and variable pitch are desirable.

Another important research topic is the incorporation of electrical constraints into the MCG router. This has been a parallel investigation along with the development of MCG itself and some findings were published in [16]. In the paper, the authors used a precise model to check the crosstalk between the routes in the compatibility test during the routing process, and therefore a certain degree of noise avoidance is achieved, which will reduce the rework time needed for electrical analysis followed by re-route.

These ongoing research projects will certainly enrich the characteristics of the MCG router and make it more powerful and even more attractive.

REFERENCES

[1] Bondy, J. A. and U. S. R. Murty, *Graph Theory with Applications*, Macmillan, London, 1976.

[2] Carothers, J. D. and D. Li, "A Multilayer MCM Autorouter Based on the Correct-by-Design Approach," *Proc. 1995 IEEE International ASIC Conf.*, pp 139-142, Sept. 1995.

[3] Carothers, J. D. and D. Li, "The MCG Autorouter for Multichip Modules," *IEEE Transactions on Circuits and Systems*, submitted.

[4] Cho, J.-D., K.-F. Liao, M. Sarrafzadeh, "Multilayer Routing Algorithm for High Performance MCMs", *Proc. 1992 IEEE International ASIC Conf.*, pp. 226-229, 1992.

[5] Cho, J.-D., M. Sarrafzadeh, M.. Sriram, and S.-M. Kang, "High Performance MCM Routing", *IEEE Design & Test of Computers*, pp. 27-37, vol. 10, Dec. 1993.

[6] Cho, J.-D., K.-F. Liao, S. Raje, and M. Sarrafzadeh, "M^2R: Multilayer Routing Algorithm for High-Performance MCMs", *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, pp. 253-265, vol. 41, April, 1994.

[7] Cong, J., K. S. Leung, and D. Zhou, "Performance-Driven Interconnection Design Based on Distributed RC Delay Model," *UCLA Computer Sci. Dept. Tech. Report CSD-920043*, 1992.

[8] Cormen, Thomas H., C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, McGraw-Hill, New York, 1990.

[9] Dai, W. W.-M., "Performance Driven Layout of Thin-Film Substrates for Multichip Modules," *Proc. of the 1991 IEEE International Symposium on Circuits and Systems*, pp. 2308-2311, 1991.

[10] Dai, W. M., T. Dayan, and D. Staepelaere, "Topological Routing in SURF: Generating a Rubber-Band Sketch," *Proc. of the ACM/IEEE 28th Design Automation Conference*, pp. 41-44, 1991.

[11] Dai, W. M., R. Kong, and M. Sato, "Routability of a Rubber-Band Sketch," *Proc. of the ACM/IEEE 28th Design Automation Conference*, pp. 45-48, 1991.

[12] Dai, W. W.-M., R. Kong, J. Jue and M. Sato, "Rubber Band Routing and Dynamic Data Representation," *Proc. IEEE International Conf. on Computer-Aided Design*, pp. 52-55, 1990.

[13] Deo, Narsingh, *Graph theory with applications to engineering and computer science*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[14] Devaraj, G. and D. Bhatia, "Crosstalk Driven MCM Router," *Journal of Microelectronic Systems Integration*, vol. 2, pp. 65-80, 1994.

[15] Frye R. C. et al., "Trends in Silicon-on-Silicon Multichip Modules," *IEEE Design & Test of Computers*, pp. 8-17, December 1993.

[16] Hameenanttila, T. and J. D. Carothers, "Fast Coupled Noise Calculation for Crosstalk Avoidance in MCM Autorouting," *Proc. 1995 IEEE International ASIC Conf.*, pp 11-14, Sept. 1995.

[17] Hameenanttila, T., "Fast Coupled Noise Estimation for Crosstalk Avoidance in MCM Routing," *Master Thesis, Dept. Electrical and Computer Engineering, The University of Arizona*, 1995.

[18] Hanafua, A., Y. Yamashita, and M. Yasuda, "Three-Dimensional Routing for Multilayer Ceramic Printed Circuit Boards," *Proc. IEEE International Conf. on Computer-Aided Design*, pp. 386-389, Nov. 1990.

[19] Herrell, D., "Multichip Module Technology at MCC," *IEEE International Symposium on Circuits and Systems*, vol. 3, pp. 2099-2103, 1990.

[20] Hightower, D. W., "A Solution to the Line Routing Problem on a Continuous Plane," *Proc. 6th Design Automation Workshop,* 1969.

[21] Ho, J. M. et al., "Layer Assignment for Multichip Modules," *IEEE Transactions on Computer-Aided Design*, vol. 9, pp. 1272-1277, 1990.

[22] Hwang, F. K. "On Steiner Minimal Trees with Rectilinear Distance," *SIAM Journal on Applied Mathematics*, Vol. 30, pp. 104-114, 1976.

[23] Khoo, K.-Y., and J. Cong, "A Fast Multilayer General Area Router for MCM Designs", *Proc. of the European Design Automation Conference*, pp. 292-297, 1992.

[24] Khoo, K.-Y., and J. Cong, "An Efficient Multilayer MCM Router Based on Four-Via Routing," *Proc. of the 1993 ACM/IEEE Design Automation Conference*, pp. 590-595, 1993.

[25] Khoo, K.-Y., and J. Cong, "A Fast Four-Via Multilayer MCM Router," *Proc. of the 1993 IEEE Multi-Chip Module Conference*, pp. 179-184, 1993.

[26] Khoo, K.-Y., and J. Cong, "A Fast Multilayer General Area Router for MCM Designs", *IEEE Transactions on Circuits and Systems*, vol. 39, pp. 841-851, 1992.

[27] Khoo, K.-Y., and J. Cong, "An Efficient Multilayer MCM Router Based on Four-Via Routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, in press.

[28] Lee, C. Y., "An Algorithm for Path Connections and Its Applications," *IRE Transactions on Electronics Computers*, 1961.

[29] Licari, James J., *Multichip Module Design, Fabrication, & Testing*, McGraw-Hill, Inc., New York, 1995.

[30] Manber, Udi., *Introduction to Algorithms : a Creative Approach*, Addison-Wesley, Reading, Mass. 1989.

[31] Mikami, K., and K. Tabuchi, "A Computer Program for Optimal Routing of Printed Circuit Connectors," *IFIPS Proc.*, H47, pp. 1475-1478, 1968.

[32] Miyoshi, T. et al., "An MCM Routing Algorithm Considering Crosstalk," *Proc. IEEE International Symposium on Circuits and Systems*, pp. 211-214, 1995.

[33] Preas, B., M. Pedram, and D. Curry, "Automatic Layout of Silicon-on-Silicon Hybrid Packages," *Proc. ACM/IEEE 26th Design Automation Conference*, pp. 393-399, 1989.

[34] Sait, S. M., H. Youssef, *VLSI Physical Design Automation*, Institute of Electrical and Electronics Engineers, 1995.

[35] Sarrafzadeh, M., K.-F. Liao, and C. K. Wong, "Single-Layer Global Routing," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 13, pp. 38-47, Jan. 1994.

[36] Sherwani, N., *Algorithms for VLSI Physical Design Automation*, Kluwer Academic Publishers, Boston, 1993.

[37] Sherwani, N., Bhingarde, Siddharth. and Panyam, Anand., *Routing in the Third Dimension : from VLSI Chips to MCMs*. Institute of Electrical and Electronics Engineers, 1995.

[38] Sriram, M. and S. M. Kang, "iPROMIS: An Interactive Performance Driven Multilayer MCM Router," *Proc. of the 1993 IEEE Multi-Chip Module Conference*, pp. 170-173, 1993.

[39] Sriram, M. and S. M. Kang, "Detailed layer assignment for MCM routing," *Proc. International Conf. on Computer-Aided Design*, pp. 386-389, 1992.

[40] Sriram, M. and S. M. Kang, *Physical Design for Multichip Modules*, Kluwer Academic Publishers, Boston, 1994.

[41] Sriram, M. and S. M. Kang, "Performance-Driven MCM Routing Using a Second-Order RLC Tree Delay Model," *Proc. IEEE International Conf. Wafer Scale Integration*, pp. 262-267, 1993.

[42] Staepelaere, D., J. Jue, T. Dayan, and W. W.-M. Dai, "Surf: Rubber-Band Routing System for Multichip Modules," *IEEE Design & Test of Computers*, vol. 10, pp. 18-26, Dec., 1993.

[43] Zhou, D. et al., "A Simplified Synthesis of Transmission Lines with a Tree Structure," *Analog Integrated Circuits and Signal Processing*, pp. 19-30, 1994.