FULL-CHIP NANOMETER ROUTING TECHNIQUES

ANALOG CIRCUITS AND SIGNAL PROCESSING SERIES

Consulting Editor: Mohammed Ismail. *Ohio State University*

Titles in Series:

ANALOG CIRCUIT DESIGN TECHNIQUES AT 0.5V
Chatterjee, S., Kinget, P., Tsividis, Y., Pun, K.P.
ISBN-10: 0-387-69953-8

IQ CALIBRATION TECHNIQUES FOR CMOS RADIO TRANCEIVERS
Chen, Sao-Jie, Hsieh, Yong-Hsiang
ISBN-10: 1-4020-5082-8

LOW-FREQUENCY NOISE IN ADVANCED MOS DEVICES
Haartman, Martin v., Östling, Mikael
ISBN-10: 1-4020-5909-4

THE GM/ID DESIGN METHODOLOGY FOR CMOS ANALOG LOW POWER INTEGRATED CIRCUITS
Jespers, Paul G.A.
ISBN-10: 0-387-47100-6

PRECISION TEMPERATURE SENSORS IN CMOS TECHNOLOGY
Pertijs, Michiel A.P., Huijsing, Johan H.
ISBN-10: 1-4020-5257-X

CMOS CURRENT-MODE CIRCUITS FOR DATA COMMUNICATIONS
Yuan, Fei
ISBN: 0-387-29758-8

RF POWER AMPLIFIERS FOR MOBILE COMMUNICATIONS
Reynaert, Patrick, Steyaert, Michiel
ISBN: 1-4020-5116-6

IQ CALIBRATION TECHNIQUES FOR CMOS RADIO TRANCEIVERS
Chen, Sao-Jie, Hsieh, Yong-Hsiang
ISBN: 1-4020-5082-8

ADVANCED DESIGN TECHNIQUES FOR RF POWER AMPLIFIERS
Rudiakova, A.N., Krizhanovski, V.
ISBN 1-4020-4638-3

CMOS CASCADE SIGMA-DELTA MODULATORS FOR SENSORS AND TELECOM
del Río, R., Medeiro, F., Pérez-Verdú, B., de la Rosa, J.M., Rodríguez-Vázquez, A.
ISBN 1-4020-4775-4

SIGMA DELTA A/D CONVERSION FOR SIGNAL CONDITIONING
Philips, K., van Roermund, A.H.M.
Vol. 874, ISBN 1-4020-4679-0

CALIBRATION TECHNIQUES IN NYQUIST A/D CONVERTERS
van der Ploeg, H., Nauta, B.
Vol. 873, ISBN 1-4020-4634-0

ADAPTIVE TECHNIQUES FOR MIXED SIGNAL SYSTEM ON CHIP
Fayed, A., Ismail, M.
Vol. 872, ISBN 0-387-32154-3

WIDE-BANDWIDTH HIGH-DYNAMIC RANGE D/A CONVERTERS
Doris, Konstantinos, van Roermund, Arthur, Leenaerts, Domine
Vol. 871 ISBN: 0-387-30415-0

METHODOLOGY FOR THE DIGITAL CALIBRATION OF ANALOG CIRCUITS AND SYSTEMS: WITH CASE STUDIES
Pastre, Marc, Kayal, Maher
Vol. 870, ISBN: 1-4020-4252-3

HIGH-SPEED PHOTODIODES IN STANDARD CMOS TECHNOLOGY
Radovanovic, Sasa, Annema, Anne-Johan, Nauta, Bram
Vol. 869, ISBN: 0-387-28591-1

LOW-POWER LOW-VOLTAGE SIGMA-DELTA MODULATORS IN NANOMETER CMOS
Yao, Libin, Steyaert, Michiel, Sansen, Willy
Vol. 868, ISBN: 1-4020-4139-X

DESIGN OF VERY HIGH-FREQUENCY MULTIRATE SWITCHED-CAPACITOR CIRCUITS
U, Seng Pan, Martins, Rui Paulo, Epifânio da Franca, José
Vol. 867, ISBN: 0-387-26121-4

DYNAMIC CHARACTERISATION OF ANALOGUE-TO-DIGITAL CONVERTERS
Dallet, Dominique; Machado da Silva, José (Eds.)
Vol. 860, ISBN: 0-387-25902-3

ANALOG DESIGN ESSENTIALS
Sansen, Willy
Vol. 859, ISBN: 0-387-25746-2

Full-Chip Nanometer Routing Techniques

By

TSUNG-YI HO

YAO-WEN CHANG

SAO-JIE CHEN

Graduate Institute of Electronics Engineering
and Department of Electrical Engineering
National Taiwan University, Taipei, Taiwan, ROC

Springer

A C.I.P. Catalogue record for this book is available from the Library of Congress.

ISBN 978-1-4020-6194-3 (HB)
ISBN 978-1-4020-6195-0 (e-book)

Published by Springer,
P.O. Box 17, 3300 AA Dordrecht, The Netherlands.

www.springer.com

Printed on acid-free paper

All Rights Reserved
© 2007 Springer
No part of this work may be reproduced, stored in a retrieval system, or transmitted
in any form or by any means, electronic, mechanical, photocopying, microfilming, recording
or otherwise, without written permission from the Publisher, with the exception
of any material supplied specifically for the purpose of being entered
and executed on a computer system, for exclusive use by the purchaser of the work.

Contents

List of Figures

List of Tables

Preface

As Moore's Law continues unencumbered into the nanometer era, chips are reaching 1,000 M gates in size, process geometries have shrunk to 90 nm and below, and engineers have to face compounded design complexity with every new design. These nanometer-scale designs require a new generation of physics-aware and manufacturing-aware routing. At 90 nm and below, there are so many signal-integrity issues that design teams cannot manually correct them all. At 90 nm, wires account for nearly 75% of the total delay in a circuit. Even more insidious, however, is that among nearly 40% of these nets, more than 50% of their total net capacitance is attributed to the cross-coupling capacitance between neighboring signals. At this point a new design and optimization paradigm based on real wires is required. Nanometer routers must prevent and correct these effects on-the-fly in order to reach timing closure. From a manufacturability standpoint, nanometer routers must explicitly deal with the ever-increasing design complexity, and be capable of adapting to the constraint requirements of timing, signal integrity, process antenna effect, and new interconnect architecture such as X-architecture. In the nanometer era, we must look into new-generation routing technologies that combine high performance and capacity with the integration of congestion, timing, SI prevention, and DFM algorithms as the best means of getting to design closure quickly. In this book, we present a novel multilevel full-chip router, namely *m*SIGMA for SIGnal-integrity and MAnufacturability optimization. And these routing technologies will ensure faster time-to-market and time-to-profitability.

Tsung-Yi Ho, Yao-Wen Chang, and Sao-Jie Chen
Graduate Institute of Electronics Engineering
and Department of Electrical Engineering
National Taiwan University, Taipei, Taiwan, ROC

Acknowledgments

To our parents and families

Tsung-Yi Ho
Yao-Wen Chang
Sao-Jie Chen

Graduate Institute of Electronics Engineering
and Department of Electrical Engineering
National Taiwan University,Taipei, Taiwan, ROC

Chapter 1

INTRODUCTION

As Moore's Law continues unencumbered into the nanometer era, chips are reaching 1,000 M gates in size, process geometries have shrunk to 90 nm and below, and engineers have to face compounded design complexity with every new design. These nanometer-scale designs require a new generation of physics-aware and manufacturing-aware routing. At 90 nm and below, there are so many signal-integrity issues that design teams cannot manually correct them all. Nanometer routers must prevent and correct these effects on-the-fly in order to reach timing closure. From a manufacturability stand-point, nanometer routers must explicitly deal with the ever-increasing design complexity, and be capable of adapting to the constraints of timing, signal integrity, process antenna effect, and new interconnect architecture such as X-architecture. In this book, we present a multilevel full-chip router, namely *m*SIGMA for SIGnal-integrity and MAnufacturability optimization.

1. DOWN TO THE WIRE

In nanometer design, wiring delay accounts for the vast majority of overall delay. It is well known that delay has been shifting from gates to wires for quite some time. As shown in Figure 1-1, wiring delay exceeds gate delay at 0.18 μm and below in aluminum processes, and at 0.13 μm and below in copper. By 90 nm, wiring delay will account for some 75% of the overall delay [1]. And nearly 40% of the nets in a design at that technology node will attribute more than 50% of their total net capacitance to the cross-coupling between neighboring signals. At smaller geometries, the situation grows more severe, and traditional design methodologies breakdown (see Figure 1-2). As a result, design teams need to shift their focus from logic optimization to wire optimization.

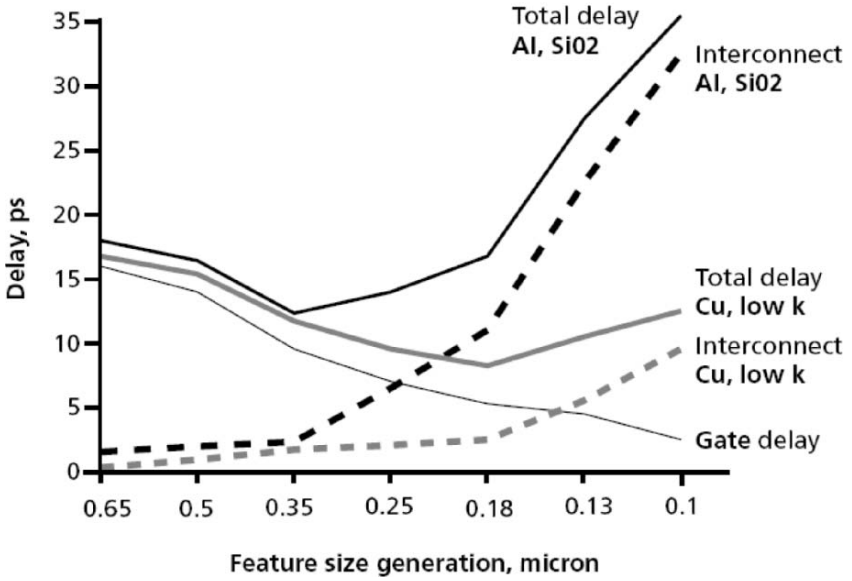

Figure 1-1. Wire and Gate Delay in Al and Cu.

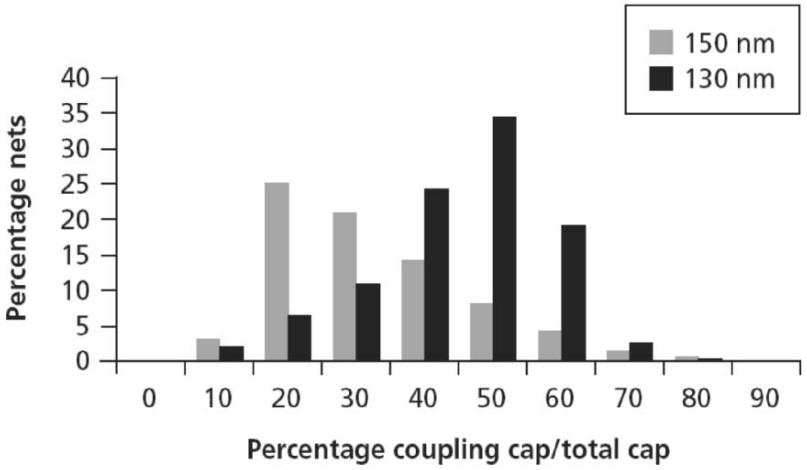

Figure 1-2. Cross-Coupling Between Neighboring Signal Affects Total Net Capacitance.

2. ROUTING PROBLEMS

Routing is the process of finding the geometric layouts of all the nets. The input of a routing problem is shown in the following:

(1) Routing area: the dimensions of the rectangular routing region and the number of available routing layers
(2) Netlist: required interconnects as sets of connection points ("pins") on placed objects
(3) Design rules: specifications of the minimum and/or maximum values allowed for wire width, wire spacing, via width, via spacing, etc.
(4) Pin locations: as determined by the placement or the floorplanning process
(5) Obstacles: objects which wires cannot pass through, including placed cells, IP blocks, pads, pre-routed wires, etc.
(6) Constraint-related parameters: such as the electrical parameters (e.g., RC constants) for performance and signal integrity constraints, thermal parameters for the thermal constraints, etc.

The constraints of a general routing process usually include connection rules and design rules [35]. Connection rules stipulate that (1) to prevent open circuits, wires of the same net must be connected together and (2) to avoid short circuits, wires of different nets must not overlap with each other at the same layer. Design rules specify the sizes of the wires and vias (connections between layers) and the minimum spacing between them according to the available manufacturing technology. The objective of the routing problem depends on the circuit design. For general purpose chips, the objective is usually the total wire length of all nets. For high-performance chips, the delay of the circuit is minimized.

The continuous increasing system-on-chip (SoC) design complexity imposes severe challenges for modern router design. As pointed out in Cong et al. [31], a 2.5 × 2.5 cm^2 chip in the 70 nm technology may have over 360,000 horizontal and vertical routing tracks. In addition, the 90 nm technology node has design rules in the high hundreds to low thousands, whereas the forthcoming 65 nm node may have several thousand design rules. To tackle the challenges, the routing frameworks are evolving from the *flat* framework to the *hierarchical* and *multilevel* frameworks. We detail the three routing frameworks in the following.

2.1 Flat Routing Framework

Research in VLSI routing has received much attention in the literature. It is typically a very complex combinatorial problem. In order to make it manageable, the routing problem is usually solved using a two-stage approach consisting of global routing followed by detailed routing. Global routing first partitions the routing area into tiles and decides tile-to-tile paths for all nets while attempting to optimize some given objective functions (for example, the overall wiring length and the timing constraints). Then, guided by the results obtained in global routing, detailed routing assigns actual tracks and vias for nets. Traditionally, many routing algorithms adopt a flat framework that finds paths for nets directly. These algorithms can be classified into sequential and concurrent approaches.

2.1.1 Sequential Approach

Perhaps the most straightforward strategy for routing is to select a specific order and then to route nets sequentially in that order. The main advantage of this approach is that the congestion information for previously routed nets can be taken into consideration while routing a given net. The drawback of sequential approach is that the quality of the routing solution greatly depends on the order, and it is hard to find a good net ordering. Abel [7] has concluded that there is no single net-ordering technique that performs better than any other ordering method in all routing problems. Since the net-ordering problem may cause unroutable nets, a rip-up/reroute procedure is often used to refine the solution.

One basic subproblem in sequential routing is to find a path connecting two pins in the presence of wiring blockages. Many algorithms have been proposed for this subproblem, and these algorithms can be classified into *maze-searching* and *line-searching* approaches.

(1) Maze Searching

Lee [68] proposed the first maze-searching algorithm, which adopts a two-phase approach of *wave propagation* followed by *retrace*. In the wave propagation phase, starting from the source vertex A, the accumulated length from the source to each vertex is labeled one by one according to the wavefront until the target vertex B is reached. The shortest length path is then traced back from B to A in the retrace phase. Figure 1-3 illustrates the process of Lee's maze-search algorithm. The Lee's algorithm guarantees to find a connection between two terminals if it does exist and the connection is the shortest path. However, in practice, the maze-searching algorithm is slow and has large memory requirements; therefore, it cannot be applied to large designs directly.

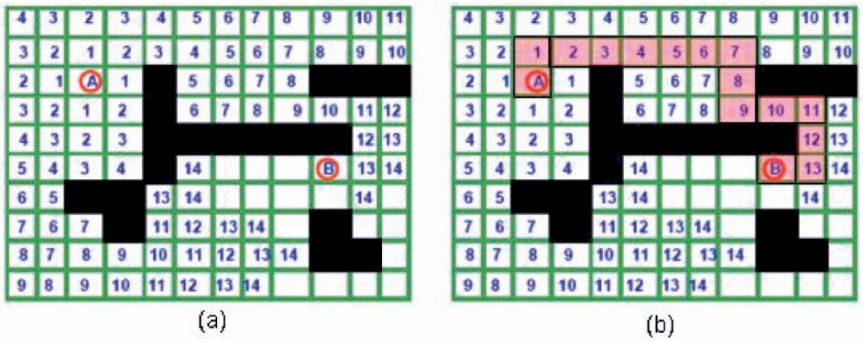

Figure 1-3. Maze Searching Example: (a) Wave Propagation and (b) Backtracking.

Many efforts have attempted to improve its speed and memory usage. Akers [8] proposed a *coding sequence* technique to reduce memory requirements. Instead of wavefront numbers, Hadlock [44] used detour numbers for wave labeling to substantially reduce the search space and running time. Soukup [92] combined breadth-first search and depth-first search approaches to the wave propagation; with this approach, the maze-searching algorithm can speed up 10–50 times than Lee's algorithm, but the disadvantage is that it does not guarantee to find the shortest path. Some techniques such as *starting point selection*, *double fanout*, and *framing* are proposed to reduce the search space of wave propagation and therefore to speed up the running time considerably [87].

Although there exist some disadvantages in this maze-search method, the maze-search approach still plays an important role and is usually incorporated into other existing routing algorithms. For example, Cong and Madden [34] integrated maze routing and the *iterative deletion* technique to develop a performance-driven multilayer area router for printed circuit board (PCB) and multichip block (MCM) designs.

(2) Line Searching

As mentioned earlier, the major drawbacks of a maze-searching algorithm are its high amount of memory required and long running time. The line-search algorithm overcomes these drawbacks by using line segments to represent the routing space and paths. Mikami and Tabuchi [81] proposed the first line-search algorithm. As opposed to the maze-searching algorithm, which mainly proceeds in a breadth-first manner, the line-searching algorithm performs a depth-first search. The line-searching algorithm initially sets the source S and the target T as *base points*, and then generates four (two horizontal and two vertical) line segments passing through these base points. These line segments are extended until they hit the design boundary or

obstacles. Then, the intersections of these line segments are iteratively set as new base points, and four new line segments are generated for these new base points. This process repeats until a segment generated from *S* intersects a segment generated from *T*, and a connection can be found by tracing from this intersection point to both *S* and *T*. Figure 1-4 gives an example of the Mikami–Tabuchi's line-searching algorithm. Like Lee's maze-searching algorithm, Mikami–Tabuchi's line-searching algorithm also guarantees to find a path if one exists, but it may not always be the shortest. The line-searching technique significantly reduces both memory requirement and execution time.

Later, Hightower [50] proposed another line-searching algorithm, which is similar to Mikami–Tabuchi's algorithm. The difference is that Hightower's algorithm only considers those line segments that are extendable beyond obstacles, and each line segment has at most two base points. Figure 1-4(b) illustrates Hightower's line-searching algorithm. Because fewer line segments are considered, Hightower's algorithm has dramatic memory saving than Mikami–Tabuchi's algorithm. However, Hightower's algorithm might fail to find a path even if one exists. To remedy the deficiencies, it needs *back-tracking* procedures to choose the right base points, and therefore, the running time does not improve very much than Lee's maze-searching algorithm in practice.

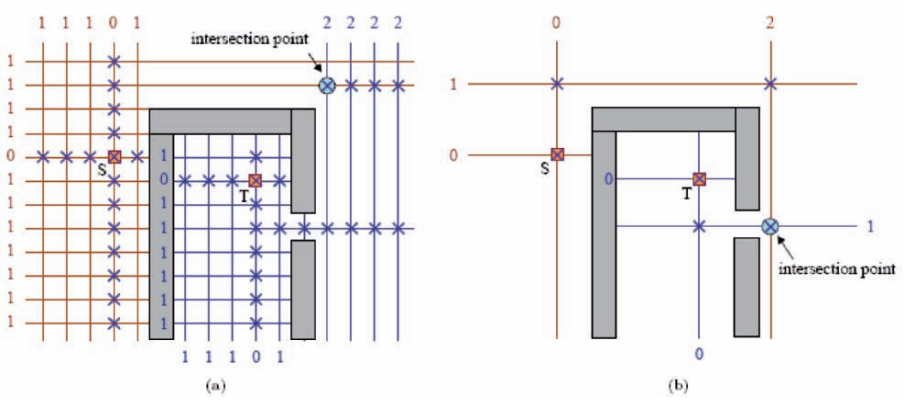

Figure 1-4. Line-Searching Algorithms: (a) Mikami–Tabuchi's Algorithm and (b) Hightower's Algorithm. The Crossing Points Denote the Base Points, and the Numbers Denote the Sequence of the Search Process.

2.1.2 Concurrent Approach

The major drawback of the sequential approach is that it suffers from the net-ordering problem. Under any net ordering, it is more difficult to route the nets that are considered later because they are subject to more blockages. In

addition, if the sequential routing fails to find a feasible solution, it is not clear whether this is because of no feasible solution existed or because of a bad selection of net order. Moreover, when the sequential routing does find a feasible solution, we do not know whether or not this solution is optimal, or how far it is from the optimal solution. These questions will be answered if we can solve the routing problem with the concurrent approach.

One common concurrent approach is to formulate global routing as a 0–1 *integer linear programming* (0–1 ILP) problem. The layout is first modeled as a routing graph $G(V, E)$, where each node represents a tile and each edge denotes the boundary between two adjacent tiles. Each edge $e \in E$ is assigned a capacity, denoted by c_e, which represents the number of tracks belonging to that boundary. Given a net, all of its possible routing patterns can be enumerated. Let the variable $x_{i,j} \in \{0, 1\}$ indicates if the routing pattern $R_{i,j}$ is selected from the set of routing patterns R_i of net N_i. Consequently, for a routing graph $G(V, E)$ with netlist N, the congestion-driven global routing can be formulated as a 0–1 ILP problem as follows:

$$\begin{aligned}
&\text{Minimize } \hat{\lambda} \\
&\text{Subject to } \sum\nolimits_{R_{i,j} \in R_i} x_{i,j} = 1, \quad \forall N_i \in \mathrm{N} \\
&x_{i,j} = \{0,1\}, \quad \forall N_i \in \mathrm{N}, \forall R_{i,j} \in R_i \\
&\sum\nolimits_{i,j:e \in R_{i,j}} x_{i,j} \le \hat{\lambda} c_e, \quad \forall e \in E
\end{aligned}$$

The first and the second constraints require that only one routing pattern can be chosen for each net, and the third constraint with the objective together ensure to minimize the maximum congestion. If a solution of $\lambda \le 1$ exists, an optimal global routing solution (the maximum congestion is minimized) can be achieved.

Since the 0–1 ILP is NP-complete, the high time complexity greatly limits the feasible problem size. An alternative approach to this problem is to first solve the continuous linear programming (LP) relaxation, obtained by replacing the second constraint with $x_{i,j} = [0, 1]$, because LP problems can be solved in polynomial time. Then, the fractional solution obtained may be transformed to integer solutions through rounding techniques such as randomized rounding [86]. However, this approximation would inevitably lose the optimality.

In practice, the 0–1 ILP concurrent routing technique is often embedded into a larger overall global routing strategy with a divide-and-conquer manner, such as solving a subproblem, where the complexity of computing the optimal solution is manageable.

2.2 Hierarchical Routing Framework

The major problem of the flat frameworks lies in their scalability for handling larger designs. To cope with the increasing complexity, researchers proposed to use hierarchical frameworks to handle the problem. The hierarchical routing frameworks use systematic divide-and-conquer approach by transforming a large and complicated routing problem into a series of smaller and simpler subproblems and then proceed in a *top-down*, *bottom-up*, or *hybrid* manner.

2.2.1 Top-Down Hierarchical Approach

Burstein and Pelavin [14] proposed the first prominent top-down hierarchical global routing framework. They recursively divide the routing regions into successively smaller subregions, named *super cells*, and nets at each hierarchical level are routed sequentially or concurrently and are refined in the subsequent levels. An example of global routing by the top-down hierarchical approach is illustrated in Figure 1-5. Figure 1-5(a) gives a global-routing instance with a 3-pin net. Figure 1-5(b) depicts the process of top-down hierarchical global routing, in which the routing region is recursively bisected into smaller super cells, and at each level, the net is routed in terms of these super cells at that level. This process is performed in a top-down manner until the super cells reduce to the actual global routing cells.

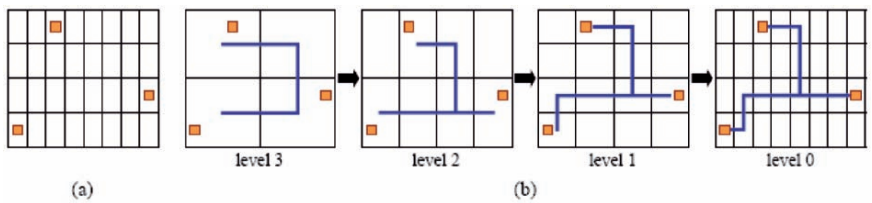

Figure 1-5. An Example of Global Routing Using the Hierarchical Top-Down Approach: (a) A Global Routing Instance with a 3-Pin Net and (b) The Level-by-Level Top-Down Hierarchical Global Routing.

Marek-Sadowska [79] proposed another top-down hierarchical framework based on bisection and *linear assignment* techniques. When a super cell is bisected by a cut line c, any net n that must cross c is then partitioned into two subnets n_1 and n_2 by inserting a *pseudo pin* on c, such that if n crosses c through this pseudo pin, no capacity overflow would occur and the wire length is minimized. Then, the subnets n_1 and n_2 can be solved independently in the subsequent levels. This bisection and insertion process

is performed recursively in the subregions until the smallest subregions is manageable for global routing. In Marek-Sadowska [79], the problem of finding a pseudo pin for each crossing net is formulated and solved as a linear assignment problem. For the global-routing instance in Figure 1-5(a), Figure 1-6 shows the bisection and pseudo-pin insertion process at each hierarchical level.

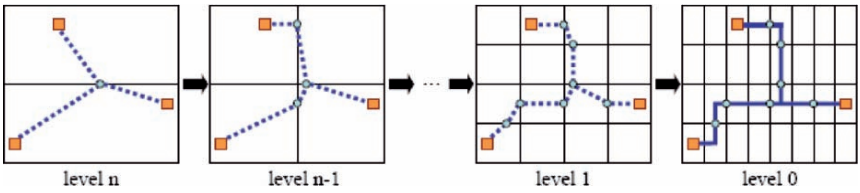

Figure 1-6. An Example of Top-Down Hierarchical Global Routing by the Bisection and Pseudo-Pin Insertion Process. The Dots Represent the Inserted Pseudo Pins.

Heisterman and Lengauer [49] presented a hierarchical ILP approach for global routing. When we formulate a global routing problem as an integer program, the size of routing graph could be too large since each net may have many choices of route, which makes the method ineffective. Therefore, they reduced the problem size by introducing the structure of small integer programs, routing graphs H_4 containing 2×2 blocks. This method identifies nets that can be routed optimally in H_4 without loss of generality and routes them in advance.

Then the algorithm deletes variables and constraints related to them from the integer program.

Later Wang and Kuh [99] proposed a hierarchical (α, β)* algorithm for timing-driven multilayer MCM/IC routing. Their algorithm, MLR, assigns all nets into routing layers layer-pair by layer-pair based on a layer assignment algorithm. During each layer-pair routing, the timing-driven Steiner area routing algorithm (SOAR) is used to generate a Steiner tree for each net while minimizing the Elmore delay of the net. For the two nodes in a net being routed to be connected, an optimal path from one node to the other is created by the (α, β)* algorithm.

Chang et al. [18] applied linear assignment to develop a hierarchical, concurrent global and detailed router for FPGA's. At each hierarchical level, they recursively decided a cut line to divide the circuit into two parts, and then assigned routing sections to connections by a linear assignment method. Finally, they redistributed delay bound for all connections. A new cost function derived from the special properties of FPGA routing architectures is used to consider timing constraints.

2.2.2 Bottom-Up Hierarchical Approach

The first bottom-up hierarchical global routing method is described by Marek-Sadowska [79]. Initially, the routing region is partitioned into an array of 2 × 2 super cells. At each hierarchical level, the global routing is restrained within each super cell individually. When the routing at the current level is finished, every four super cells are merged to form a new larger super cell at the next higher level. This process continues until the top level containing only one 2 × 2 array is reached. Figure 1-7 illustrates this bottom-up approach. Figure 1-7(a) gives a global-routing instance with a 7-pin net. Figure 1-7(b) depicts the process of bottom-up hierarchical global routing, in which the solid rectangles represent the super cells, and the dots denote the merging points where two routing subsolutions of the previous level are merged together. Hu and Shing [57] formulated the problem of finding merging points as a linear programming problem.

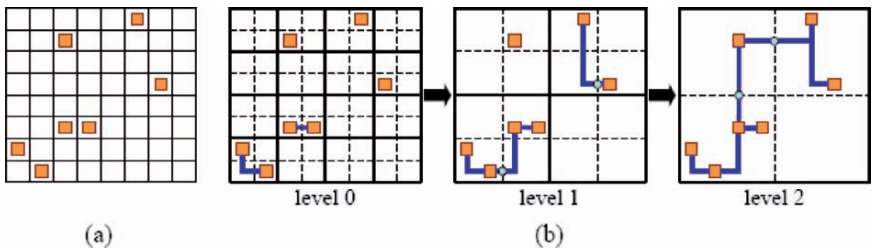

Figure 1-7. An Example of Global Routing Using the Bottom-Up Hierarchical Approach: (a) A Global Routing Instance with a 7-Pin Net and (b) Level-by-Level Bottom-Up Hierarchical Global Routing. The Solid Rectangles Represent Super Cells, and the Dots Denote Merging Points.

2.2.3 Hybrid Hierarchical Approach

The deficiency in the top-down and bottom-up hierarchical approaches is that the routing decision made at one hierarchical level may be suboptimal for the subsequent levels. In order to alleviate this problem, Lin et al. [75] proposed the first hybrid hierarchical approach that combines the bounded maze-searching algorithm with both top-down and bottom-up hierarchical methods into a unified routing framework.

Their algorithm consists of three phases: (1) neighboring propagation, (2) preference partition, and (3) bounded routing. Phase 1 performs bounded maze searching by propagating W circles of waves out of each terminal, where W is a user-defined parameter. If the connection is not found, phase 2 recursively maps the terminal and blockages onto the adjacent upper level (Figure 1-8(a)) and calls the bounded maze-search algorithm until a path is

found. Then, the connected path is mapped back to the lower level to form the preferred region (Figure 1-8(b)). Phase 3 performs the routing by taking the preference information into consideration (Figure 1-8(c)). By means of a parameter-controlled technique, their hybrid routing demonstrates a fast speed comparable to a hierarchical router and produces routing solutions with quality similar to a maze router.

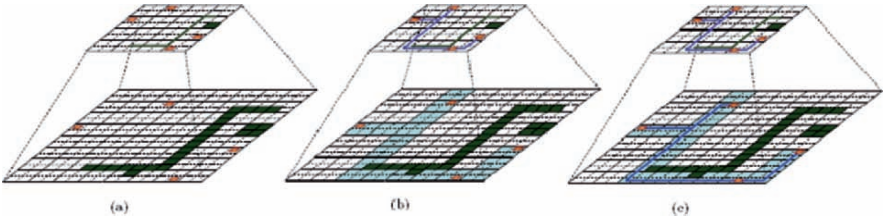

Figure 1-8. An Example of Global Routing Using the Hybrid Hierarchical Approach: (a) Mapping Pins and Blockages Up One Level, (b) Making Connection on the Upper Level and Mapping Down the Preferred Region, and (c) Performing the Routing Within the Preferred Region.

Later on, Hayashi and Tsukiyama [46] proposed another hybrid hierarchical global routing algorithm. The flow of their algorithm consists of two loops for the hierarchical levels, with a top-down hierarchical inner loop embedded in a bottom-up hierarchical outer loop. Specifically, the global routing mainly proceeds in a bottom-up manner, but an additional top-down refinement procedure is applied when an initial routing at each hierarchical level is obtained.

Compared with pure top-down or bottom-up hierarchical routing, the hybrid hierarchical approach has more information to generate better routing solutions.

2.3 Multilevel Routing Framework

The two-level, hierarchical routing framework, however, is still limited in handling the dramatically growing complexity in current and future IC designs which may need to handle hundreds of millions of gates in a single chip. As pointed out in Cong et al. [31], for a 0.07 μm process technology, a 2.5×2.5 cm^2 chip may contain over 360,000 horizontal and vertical routing tracks. To handle such high design complexity, the two-level, hierarchical approach becomes insufficient. Therefore, it is desired to employ more levels of routing for larger IC designs. Another reason is that opposite to hierarchical approaches, which lack of local information, multilevel approaches collect information at low levels to facilitate routing processes at top levels.

The multilevel method was originally used as a means of accelerating numerical algorithms for partial differential equations. In the past decade, it has

been also applied to other areas, such as image processing, combinatorial optimization, control theory, statistical mechanics, quantum electrodynamics, and linear algebra. Multilevel framework has attracted much attention in the literature recently. It employs a two-stage technique: coarsening followed by uncoarsening. The coarsening stage iteratively groups a set of circuit components (e.g., circuit nodes, cells, modules, routing tiles) based on a predefined cost metric until the number of components being considered is smaller than a given threshold. Then, the uncoarsening stage iteratively ungroups a set of previously clustered circuit components and refines the solution by using a combinatorial optimization technique (e.g., simulated annealing, local refinement). The multilevel framework has been successfully applied to VLSI physical design. For example, the famous multilevel partitioners: *ML* [10], *hMETIS* [63], and *HPM* [33], the multilevel floorplanner: *MB*-trees* [72], and the multilevel placer: *mPL* [17], all show the promise of the multilevel framework for large-scale circuit partitioning and placement.

2.3.1 Previous Multilevel Routing Framework

A framework similar to multilevel routing was presented in Cong et al. [31, 36] and Lin and Chang [74]. Lin et al. [75], and Hayashi and Tsukiyama [46] presented hybrid hierarchical *global* routers for multilayer VLSI's, in which both bottom-up (coarsening) and top-down (uncoarsening) techniques were used for global routing. Marek-Sadowska [78] proposed a global router based on outermost loop approach. The approach is similar to the coarsening stage of multilevel routing.

Recently, Cong et al. proposed a pioneering multilevel global-routing approach for large-scale, full-chip, routability-driven routing [31]. The routing area is partitioned into routing tiles. Their algorithm goes through multilevel planning to find a tile-to-tile path for each net among these tiles. For large designs, the number of tiles may be too many for these algorithms to handle. Their multilevel approach first accurately estimates the routing resource using a line-sweeping algorithm on the finest tile level. A recursive coarsening process is then employed to merge the tiles and build coarser-level representations. At each coarsening stage, the resource of each tile is estimated from the previous finer-level tiles in order to form a current tile. This coarsening process is known in multilevel literature as the "downward pass." Once the coarsening process has reduced the number of tiles to below a certain threshold, the initial routing is computed using a multicommodity flow-based algorithm. The initial routing result is refined in the reverse direction of coarsening, known as the "upward pass," by a modified maze-searching algorithm. When the final tile-to-tile paths are found at the finest level of tiles for all the nets, a gridless detailed routing algorithm [30] is

applied to find the exact path for each net. Thus, most of stages in this multilevel routing framework we used for global routing stage. This multilevel routing framework is shown in Figure 1-9.

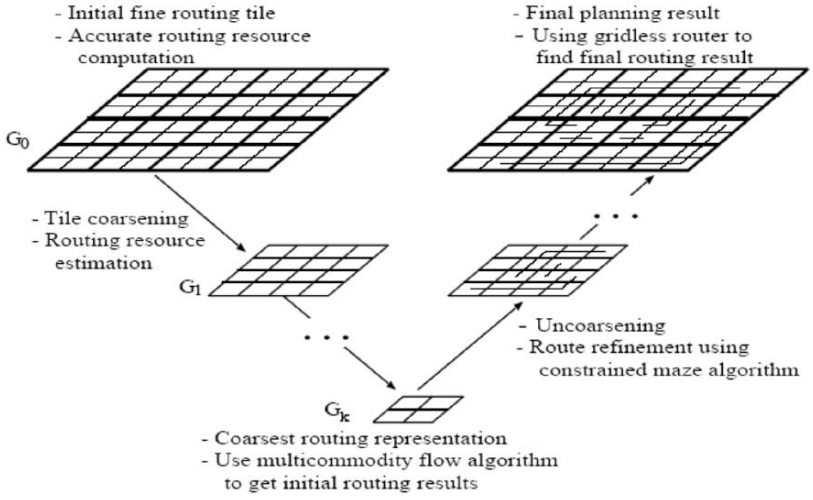

Figure 1-9. Multilevel Routing Framework Flow of Cong et al. [31].

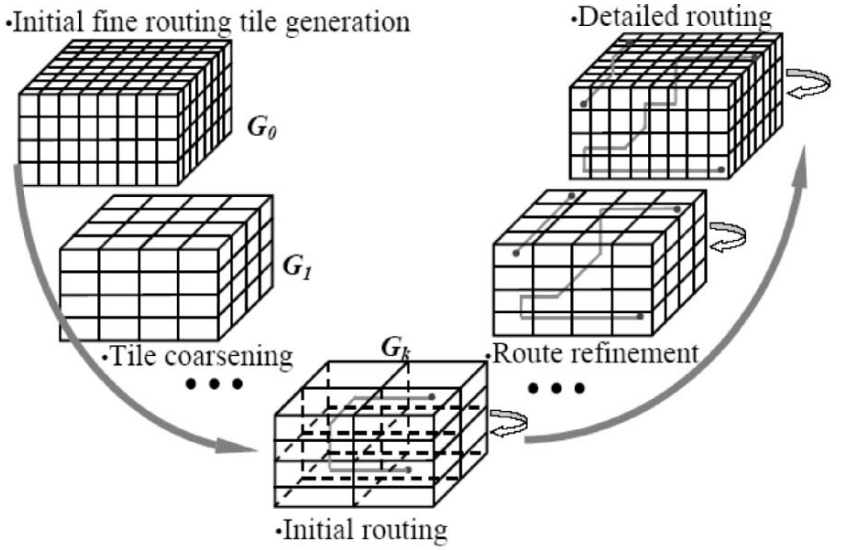

Figure 1-10. Multilevel Routing Framework Flow of Cong et al. [36].

Cong et al. later proposed an enhanced multilevel routing system, named *MARS* [36], which incorporates resource reservation, a graph-based Steiner tree heuristic and a history-based multi-iteration scheme to improve the quality of the multilevel routing algorithm [36]. The final results of the multilevel algorithm are tile-to-tile paths for all the nets. The results are then used to guide a detailed routing algorithm to final the exact connection for each net. Their experimental results showed better routing quality or running times than the traditional two-stage approach of global routing followed by detailed routing and also the hierarchical approaches. This multilevel routing framework is shown in Figure 1-10.

Lin and Chang also proposed a multilevel approach for full-chip routing, which considers both routability and performance [19, 74]. This framework integrates global routing, detailed routing, and resource estimation together at each level, leading to a more accurate routing resource estimation during coarsening and thus facilitating the solution refinement during uncoarsening. Different from the work presented in Cong et al. [31], their framework has the following features:

(1) Integrate global routing, detailed routing, and resource estimation together into each level of the framework. Specifically, at each level of the coarsening stage, global routing is performed to obtain a good initial solution for all nets in the tiles being considered and then detailed routing to obtain the exact routing patterns for these nets. Since the exact routing patterns are known, resource estimation is more accurate. With these good properties, the refinement task conducted at the following uncoarsening stage becomes much easier. In contrast, the work in Cong et al. [31] performs resource estimation only during the coarsening stage and global routing only during the uncoarsening stage. After the multilevel processing is finished, the final global routing result is then fed into a detailed router to obtain the final routing solution. It is obvious that this approach can have better interaction among global routing, detailed routing, and resource estimation since they are considered together. For example, global and detailed routers usually use rip-up and reroute to refine a routing solution based on the results of resource estimation. If these three tasks are performed separately, the rerouting process conducted at the global routing stage may be in vain since it does not know if the rerouting is useful for the detailed router. Also, the detailed router may fail to find a path because of the low flexibility induced from the separated global routing. Therefore, making the above three tasks interact with each other can significantly improve routing quality.

(2) A two-stage refinement method composed of Z-pattern routing followed by maze routing is used in this multilevel framework, which makes rerouting much more effective.
(3) Unlike the work that considers routability alone [31], they also apply a recalling modification method to perform timing-driven routing.

The experimental results show the best routability among the previous works. Detail of this multilevel routing framework is shown in Figure 1-11.

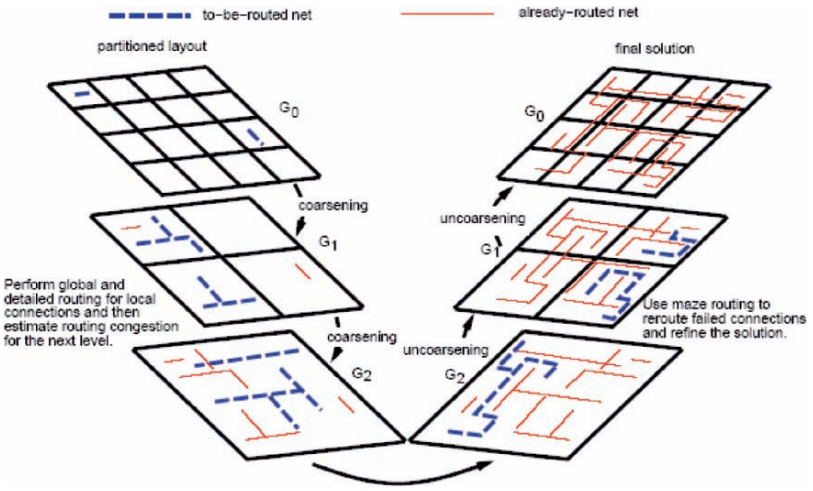

Figure 1-11. Multilevel Routing Framework Flow of Chang and Lin [19].

2.3.2 Our Multilevel Routing Framework

Different from the aforementioned works, our framework performs congestion-driven *global* routing at the coarsening stage, followed by an intermediate stage of routing *layer/track assignment* for optimization, and then *detailed* routing at the uncoarsening stage. By performing detailed routing after layer/track assignment, we can preserve more flexibility for allocating nets for optimization.

Figure 1-12 shows our multilevel framework, and Table 1-1 summarizes the differences of the framework among Cong et al. [31], Lin and Chang [74], and ours (we just compare with Cong et al. [31] because Cong et al. [36] is just the enhanced version of Cong et al. [31]). Our multilevel framework starts with coarsening of the finest tiles of level 0. At each level, pattern routing is used for routability-driven global routing. After the coarsening stage, we perform a layer/track assignment for optimization. At the uncoarsening stage, we perform detailed routing. Further, the unroutable nets

are handled by point-to-path maze routing [19, 36, 74] and rip-up and reroute to refine the routing solution level by level.

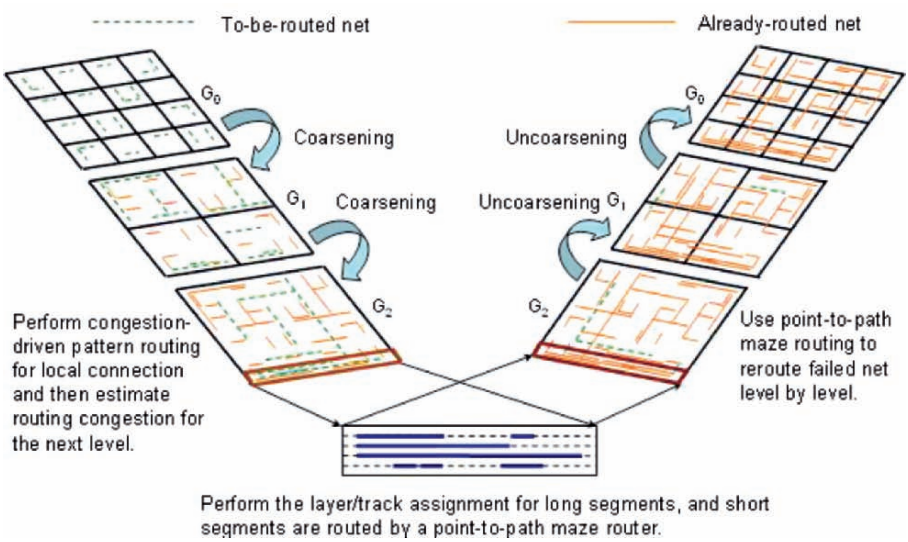

Figure 1-12. Our Multilevel Routing Framework Flow.

Table 1-1. Framework Comparison Between Cong et al. [31], Lin and Chang [74], and Ho et al. [52].

	Coarsening stage	Initial routing	Uncoarsening stage
Cong *et al.* in ICCAD 01	Resource estimation	Multicommodity flow	Global maze refinement
Lin and Chang in ICCAD 02	Global routing Detailed routing Resource estimation	No initial routing	Global and detailed maze refinement
Our Framework in ICCAD 03	Global routing Resource estimation	Track/layer assignment	Global and detailed maze refinement

3. ORGANIZATION OF THE BOOK

This book presents a novel multilevel routing framework, *m*SIGMA, which is based on the *m*ultilevel optimization paradigm and has been developed for signal-integrity and manufacturability optimization.

The traditional multilevel framework employs a two-stage technique: coarsening followed by uncoarsening. The coarsening stage iteratively groups a set of circuit components (e.g., circuit nodes, cells, modules, routing tiles) based on a predefined cost metric until the number of components being considered is smaller than a threshold. Then, the uncoarsening stage iteratively ungroups a set of previously clustered circuit components and refines the solution by using a combinatorial optimization technique.

Different with the previous multilevel routing framework, we introduce an intermediate track assignment phase between coarsening and uncoarsening stages, for improving run-time and doing optimization on SIGnal-integrity and MAnufacturability issues.

Contributions of our work are briefly introduced in the following.

3.1 Multilevel Routing Framework

The continuous increase of problem size in IC routing has become a great challenge to existing routing algorithms. The conventional method for handling large problem size is to "divide-and-conquer," which breaks the routing problem into two successive steps, global routing and detailed routing.

Conventionally, many routing algorithms adopt a flat framework to find paths for all nets. Those algorithms can be classified into sequential and concurrent approaches. But the major problem of the flat frameworks lies in their weak scalability for handling larger designs.

As technology advances, technology nodes are getting smaller and circuit sizes are getting larger. To cope with the increasing complexity, researchers proposed to use hierarchical approaches to handle the problem. This top-down approach has a global view of the routing problem, but a main drawback is that hierarchical approaches lack local information; therefore, it could induce local congestion.

The multilevel framework has attracted much attention in the literature recently. It employs a two-stage technique: coarsening followed by uncoarsening. The coarsening stage iteratively groups a set of circuit components (e.g., circuit nodes, cells, modules, routing tiles) based on a predefined cost metric until the number of components being considered is smaller than a threshold. Then, the uncoarsening stage iteratively ungroups a set of previously clustered

circuit components and refines the solution by using a combinatorial optimization technique (e.g., simulated annealing, local refinement). The multilevel framework has been successfully applied to VLSI physical design. In this book, we will focus on the multilevel routing framework.

3.2 Multilevel Full-Chip Routing Considering Crosstalk and Performance

In this book, we first present a novel framework for fast multilevel routing considering crosstalk and performance optimization. To handle the crosstalk minimization problem, we incorporate an intermediate stage of layer/track assignment into the multilevel routing framework. For performance-driven routing, we use a minimum-radius minimum-cost spanning-tree (MRMCST) heuristic for global routing. Compared with the state-of-the-art multilevel routing with the routability mode, the experimental results show that our router achieved a 6.7X run-time speedup, reduced the respective maximum and average crosstalk (coupling length) by about 30% and 24%, reduced the respective maximum and average delay by about 15% and 5%. And compared with the timing-driven mode, the experimental results show that our router still achieved a 5.9X run-time speedup, reduced the respective maximum and average crosstalk by about 35% and 23%, reduced the respective maximum and average delay by about 7% and 10% in comparable routability, and resulted in fewer failed nets.

3.3 Multilevel Full-Chip Routing Considering Antenna Effect Avoidance

As technology advances into nanometer territory, the antenna effect problem has caused significant impact on routing tools. The antenna effect is a phenomenon of plasma-induced gate oxide degradation caused by charge accumulation on conductors. It directly influences reliability, manufacturability and yield of VLSI circuits, especially in deep-submicron technology using high-density plasma. Furthermore, the continuous increase of the problem size of IC routing is also a great challenge to existing routing algorithms. In this book, we present another framework for multilevel full-chip routing with antenna avoidance using built-in jumper insertion approach. Compared with the state-of-the-art multilevel routing, the experimental results show that our approach reduced 100% antenna-violated gates and results in fewer wire length, vias, and delay increase.

3.4 Multilevel Full-Chip Routing for the X-Based Architecture

As technology advances into nanometer territory, the interconnect delay has become the first-order factor on chip performance. To handle this effect, the X-architecture has been proposed for high-performance integrated circuits. The X-architecture presents a new way of orienting a chip's microscopic interconnect wires with the pervasive use of diagonal routes. It can reduce the wire length and vias, and thus improve performance and routability. Furthermore, the continuous increase of the problem size of IC routing is also a great challenge to existing routing algorithms. In this book, we present the first multilevel framework for full-chip routing using the X-architecture. To take full advantage of the X-architecture, we explore the optimal routing for three-terminal nets on the X-architecture and develop a general X-Steiner tree (XST) algorithm based on the Delaunay triangulation approach for the X-architecture. The multilevel routing framework adopts a two-stage technique of coarsening followed by uncoarsening, with a trapezoid-shaped track assignment embedded between the two stages to assign long, straight diagonal segments for wire length reduction. Compared with the state-of-the-art multilevel routing for the Manhattan architecture, experimental results show that our approach reduced wire length by 18.7% and average delay by 8.8% with similar routing completion rate and vias.

This page intentionally blank

Chapter 2

ROUTING CHALLENGES FOR NANOMETER TECHNOLOGY

As IC process geometries scale down to the nanometer territory, the industry faces severe challenges of SIGnal-integrity and MAnufacturing limitations. To meet the signal timing requirements, it is dispensable to address the signal-integrity issues in routing stage. To guarantee yield and reliability, routing for manufacturability has played a pivotal role in resolution and thus yield enhancement for the imperfect manufacturing process. In this chapter, we introduce major challenges arising from nanometer process technology and key existing techniques for handling the challenges in routing problems for nanometer technology.

1. ROUTING REQUIREMENT FOR THE NANOMETER ERA

Routing large, complex designs is definitely a requirement of today's design starts, but most of the routing performance, capacity, and flexibility would not produce DRC-clean GDSII if the design cannot converge on timing and be free from Signal-integrity and manufacturability issues at the end of the design cycle.

To create the best wires for final routing in general, all nets in the design must be optimized using SIGMA approach, which simultaneously combines SIGnal integrity and MAnufacturability (SIGMA) optimization during the final routing stage. Furthermore, the router must be capable to handle the ever-increasing design complexity of gigascale integration. As process advances to nanometer technology, foundries may fabricate billions of transistors in a

single chip within this decade. As shown in Figure 2-1 from Intel's presentation at ISSCC 2003 [2], the number of transistors per die will still grow exponentially in the near future. Therefore, we need CAD tools of very large-scale designs to handle the increasing complexity. That is why the frameworks of CAD tools have evolved from a flat, a hierarchical, to a multilevel framework.

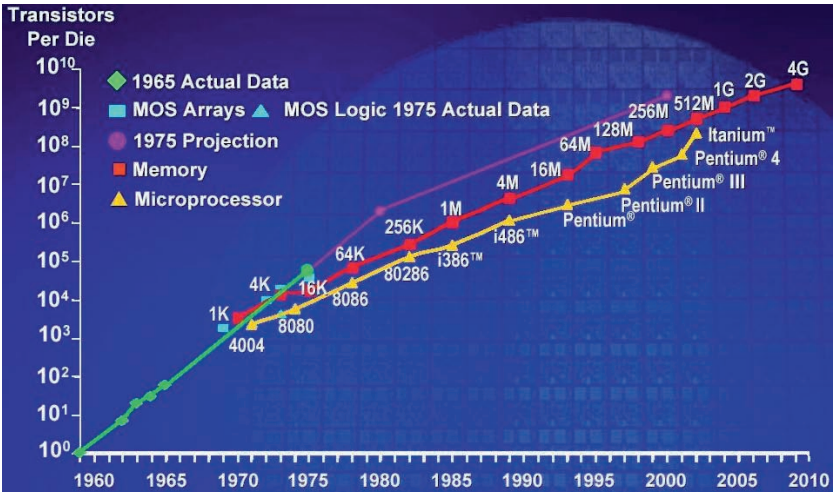

Figure 2-1. Chip Complexity Chart from Intel [2].

1.1 Signal-Integrity Problems

In most conventional IC design flows, signal-integrity analysis is performed as a post-layout activity. Unfortunately, this is a bad time to start the analysis for signal integrity effects. Attempting to analyze and correct the signal-integrity issues at post-layout often results in costly and time-consuming design iterations, failed schedules, reduced product performance, larger die sizes, and poorer manufacturing yield.

The timing in deep-submicron designs (designs using process technologies at 0.18 μm or lower) is inherently dominated by interconnect-dependent RC delay, not cell delay as used to be the case. Special tools are needed to ensure that the integrity of signals in the wires, and the integrity of the wires and other circuit components themselves are maintained. In general, signal integrity refers to the physical effects that cause signal deteriorations and physical deformations, both of which pose a threat to design failure. In the following, we will discuss signal-integrity issues including crosstalk noise and special manufacturing rules such as process antenna rules.

1.1.1 Crosstalk Problems

With the scaling of the horizontal dimensions of wires, the aspect ratio of the horizontal to vertical dimensions is reduced, resulting in increased ratios of coupling capacitance to substrate capacitances. As shown in Figure 2-2, C_{xcoup} and $C_{crossover}$ represent the coupling capacitances and C_{fringe} and C_{area} represent the substrate capacitances. When the signals in the neighboring wires switch, the coupling capacitors cause transfer of charge between them. Depending on the relative rate of switching (rise and fall times of the signals) and the amount of mutual capacitance, there can be significant crosstalk noise.

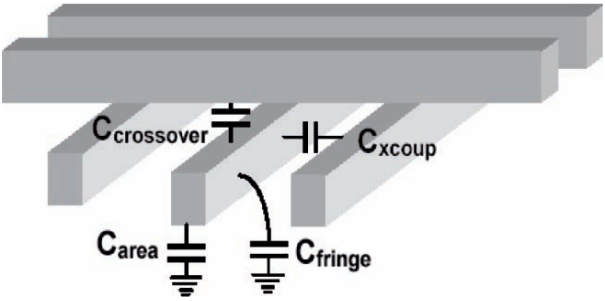

Figure 2-2. Types of Interconnect Wire Capacitance.

Crosstalk noise between neighboring signal wires can cause two major problems that affect the operational integrity of IC designs:

(1) Crosstalk delay changes the signal propagation on some of the nets, reducing achievable clock speed as illustrated in Figure 2-3.
(2) Crosstalk glitch causes voltage spikes on some nets, resulting in false logic states being captured in the registers as shown in Figure 2-4.

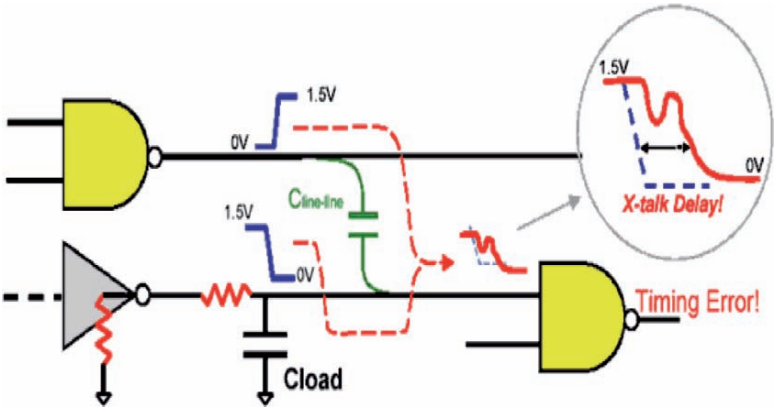

Figure 2-3. Signal Delay Caused by Crosstalk Noise.

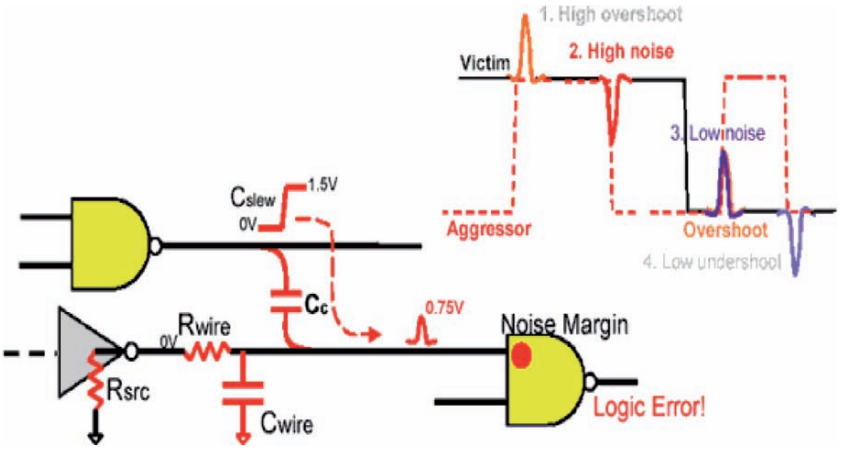

Figure 2-4. Logic Error Caused by Crosstalk Noise.

Attempting to fix the effects of crosstalk after layout is both costly and risky from the point of view in chip design and time to market. These types of effects must be analyzed for and corrected automatically throughout the routing flow, starting early in the flow. To efficiently achieve concurrent analysis and correction of crosstalk noise, it is necessary for a routing engine to own some prevention capabilities.

1.1.2 Process Antenna Effects

The processes used to manufacture deep-submicron integrated circuits themselves give rise to special rules that must be adhered to. Today, many conventional systems still rely on time-consuming post-layout fixes to address compliance to these complex rules. Physical design tools must be able to process these rules and automatically generate layouts that conform to these rules, such that post-layout fixing is avoided.

Process antenna effect is a manufacturing concern that needs to be addressed during physical design. Antenna effect problems occur in the chip manufacturing. During the metalization steps (laying down metal wires on top of the devices), some wires connected to the polysilicon gates of transistors in the design could be floating because the upper metal layers have not been deposited yet. The floating wire acts as an antenna that collects charges during certain processing step such as plasma etching (see Figure 2-5). If the energy built-up in the floating wire reaches a high enough threshold, the logic gate(s) that it is connected to could suffer permanent damage due to transistor gate oxide breakdown. This renders the die useless.

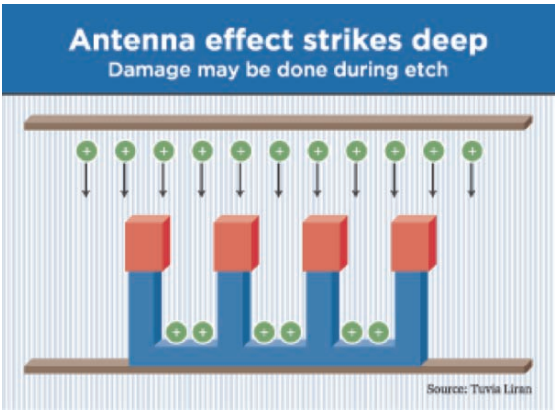

Figure 2-5. Antenna Effect Caused by Etching Process.

Antenna rules are commonly specified in terms of a ratio between the wire and the gate size (A_w/A_g) for each metal and cut (via) layer. This ratio shows how much wire could be connected to the input of the logic gate before an antenna problem could occur. In other words, this implies how much floating charge the transistor gates can handle. The semiconductor manufacturer generally gives the gate size, while the antenna checker calculates the wire size. The wire size is calculated based upon the wire (charge) accumulation method specified by the manufacturer.

1.2 Manufacturability Problems

For the most part, design teams have focused exclusively on timing closure. To ensure that the result would be manufacturable. Above 0.13 μm, manufacturing procedures such as chemical-mechanical polishing (CMP) were performed after generating the fully routed, and otherwise correct, GDSII. Design teams could then ignore the effects of physical manufacturing processes. Most design teams run into manufacturability issues for the first time they face 0.13 μm. Processes using copper wiring, optical proximity correction (OPC) and phase-shift masking (PSM) lead to exceedingly complex and arcane design rules (see Figure 2-6). Antenna rules, to take as an example, require careful handling to avoid via proliferation and minimize wire lengths. On the other hand, if one via fails, double-via insertion provides a redundant via which can serve as a fault-tolerant substitute for the failing one. However, if the amount of inserted redundant vias is not well controlled, it may adversely worsen the yield and reliability of the design because the pattern distortion of the vias might become serious. Therefore, via-density check is also suggested to be considered during double-via insertion.

Furthermore, foundries continue to change the interconnect architecture after the introduction of a new process in order to optimize interconnect delay.

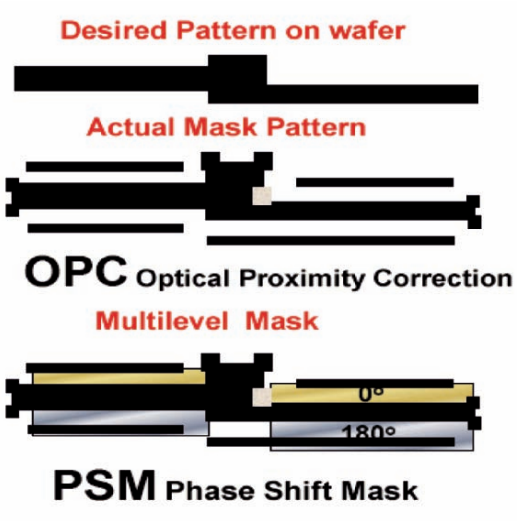

Figure 2-6. Optical Proximity Correction (OPC) and Phase Shift Masking (PSM) Problems.

1.2.1 Optical Proximity Correction

OPC is probably the most popular Resolution Enhancement Technique (RET). It can deal with some types of image-shape distortions such as line shortening, corner rounding, and line-width variations by adding or subtracting some features as serifs or line segments as shown in Figures 2-7(b)–(d). However, OPC may generate a great number of polygons with a very high mask cost, and it might fail if there is no enough space among wires after the physical design stage. Therefore, a few existing works targeted on the assurance of OPC success and the reduction of OPC pattern features.

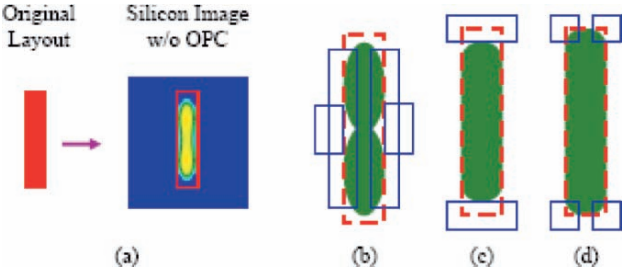

Figure 2-7. (a) Optical Proximity Effects. Three Major OPC Techniques: (b) Line Biasing, (c) Hammerhead, and (d) Serif.

Huang and Wong [59] presented a pioneering work on OPC-friendly maze routing based on the Lagrangian relaxation formulation. The router is grid-based and considers only 2-pin connections. Furthermore, it uses the flat framework and thus handles only hundreds of connections.

Wu et al. [101] formulated two OPC-aware maze routing problems and solved them by modifying the well-known Lee algorithm. In this work, simple OPC costs of routing grid cells are calculated by the estimated light intensity generated by each routed net.

Chen and Chang [25] presented the first multilevel, full-chip gridless detailed router considering the OPC effects. The router integrates global routing, detailed routing, and congestion estimation together at each level of the multilevel routing. It can handle nonuniform wire widths and consider routability and OPC simultaneously. For each line, its OPC cost is defined as the total number of pattern features and applied a modified Dijkstra's shortest path algorithm to optimize the routing and OPC costs simultaneously. Experimental results show that this work not only efficiently obtains significantly better routing solutions with 100% routing completion, but also archives an effective reduction of OPC pattern features.

Mitra et al. [82] developed an RET-aware detailed router to handle full-chip capacity to enhance the overall printability. After the initial routing result, a fast lithography simulation is applied to introduce the lithography hotspot map (LHM). According to the LHM, the router generates routing blockages for the stage of rip-up and reroute. After wire spreading, rip-up, and reroute, the number of edge placement error (EPE) hotspots is reduced by 40%.

Some OPC-related research results also focus on standard-cell characterization and fast simulation. They are highlighted below. Cao et al. [16] suggested using dummy poly insertion to shield intercell optical interference. This technique effectively reduces timing variation induced by the deviation of gate dimensions. Since the dummy polys are carefully inserted, it will not create any new transistor on the silicon. Thus, this method incurs no overhead on layout versus schematic (LVS) verification. In this work, they first simulated the images of the modified standard cells. Then they characterized the timing and leakage properties of each cell. Experimental results show that the average timing variation and power leakage variation of these shielded cells are reduced by 11.4% and 8.0%, respectively. However, the dummy polys may potentially induce parasitic capacitance and reduce the performance of the transistors.

Pawlowski et al. [85] proposed a cellwise approach to tackle the complicated and time-consuming OPC task. When adjusting the feature geometry of a cell, this method stresses the influence of boundaries of abutted cells in a row. The experimental results show that, compared with a commercial tool, this method has up to 100X speedup and 35X reduction in mask data size.

1.2.2 Phase Shift Masking

In addition to OPC, PSM is another popular technique to improve the image quality. It can be categorized into two types: attenuated PSM (AttPSM) and alternative PSM (AltPSM). AttPSM allows small amount (6–10%) of light transmission and forces the phase of the penetrated light changed by 180 degrees in the opaque region of a mask. AltPSM aggressively improves the aerial image which is mainly used in the poly layer and critical metal layers, such as the first metal layer. There are two types of AltPSM: *bright-field* and *dark-field AltPSM*. The design shapes of the bright-field AltPSM are opaque regions of a mask, and it is usually used for the poly layer. The design shapes of the dark-field AltPSM are bright regions of a mask, and it is usually used for metal layers. Although AltPSM can further reduce the *critical dimension* (*CD*), it results in a very strong proximity effect. If some feature patterns are closer than a certain threshold, therefore, they must be assigned to opposite phases. Consequently, it may generate phase-conflict problems. As an example shown in Figure 2-8, for the horizontal and the vertical segments, the lower right side of the T-junction must be assigned to respective 0- and 180-degree phases for the horizontal and the vertical segments, causing a phase conflict at the corner. Therefore, how to assign phases or modify layouts with no phase conflicts is a main challenge of PSM.

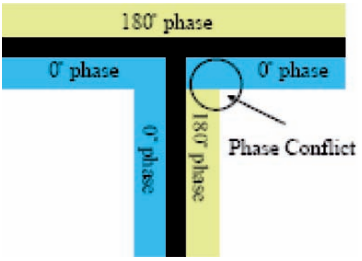

Figure 2-8. For the Horizontal and the Vertical Segments, the Lower Right Side of the T-Junction must be Assigned to Respective 0- and 180-Degree Phases for the Horizontal and the Vertical Segments, Causing a Phase Conflict at the Corner.

The problem on AltPSM has been studied to some degree. Berman et al. [13] presented a layout modification and phase-assignment algorithm for dark-field altPSM. The algorithm first constructs a conflict graph according to the dependency of the phase assignment. Then it deletes a set of edges with the minimum total weight to make the graph bipartite, where the weight is an estimated cost measuring the difficulty for layout modification.

Chiang et al. [27] presented a new computationally efficient algorithm to solve the phase-conflict problem for the bright-field PSM. In this work, the phase-conflict problem is formulated as a minimum weighted conflict-cycle removal problem. Although this problem can be solved optimally in polynomial time, this approach incurs some area overhead.

Cao et al. [15] divided the phase-conflict problems into intracell compliance and intercell composability problems from the viewpoint of celllibrary design. In this work, both the problems are formulated as Boolean satisfiability (SAT) problems solved by a SAT solver. The method considers only the problems for cells. However, the intercell connections occupying the first metal layer, which may also induce phase conflict, are not considered.

McCullen [80] proposed a phase-correct routing algorithm for the dark-field PSM. During the routing stage, the author solved some phase conflicts by widening the distance between two line ends, or moving the wiring jog to a separate wiring layer which runs orthogonally to the primary wiring direction of the nonjogged portions of the wire.

1.2.3 Double Via Insertion

If a via fails, double-via insertion provides a redundant via which can serve as a fault-tolerant substitute for the failing one. Figure 2-9 shows an example via doubling from the work [24], where different colors represent wires of different layers, and a yellow square represents a redundant via. In Figure 2-9, every via is accompanied by a redundant via to make the connection between different layers more reliable.

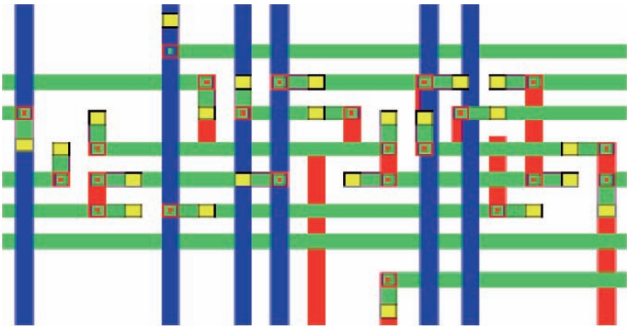

Figure 2-9. An Example of Double-Via Insertion, where Different Colors Represent Wires of Different Layers, and a Yellow Square Represents a Redundant Via.

Xu et al. [102] proposed the first work to consider double-via insertion during maze routing. By assigning double-via costs to the routing graph, they formulated the problem as a multi-objective maze routing problem and

applied Lagrangian relaxation to solve it. However, they only consider the redundant via at the detailed routing stage, their cost modeling for the number of vias is not accurate enough, and the high-time complexity of Lagrangian relaxation limits the feasible problem size to be within hundreds of nets.

Yao et al. [103] developed a grid-based router which features via minimization global routing followed by double-via aware detailed routing. The work claimed that the post-layout double-via insertion problem can be solved by a maximum bipartite matching formulation, which was recently shown to be incorrect for some cases by Lee and Wang in Lee et al. [71].

Lee and Wang [70] formulated the double-via insertion problem as a maximum independent-set (MIS) problem. Since the MIS problem for a general graph is NP-complete, they resorted to heuristics for the problem.

Luo et al. [76] considered the single vias of a design one by one to perform redundant-via insertion, and therefore their approach can get a locally optimal solution. Besides, since the approach might change the routing results of the timing noncritical nets, it may also induce timing violations even if designers prohibit the timing-critical nets from redundant-via insertion.

Chen et al. [24] presented a new full-chip gridless routing system considering double-via insertion for yield enhancement. To fully consider double vias, the router applies a novel two-pass, bottom-up routability-driven routing framework whose flow is shown in Figure 2-10. In this work, for each net, its cost function is defined as a weighted summation of the number of vias in the net and redundant-via related penalty function for the net. Moreover, a new post-layout double-via insertion algorithm was also proposed to achieve a higher insertion rate. Based on a bipartite graph matching formulation, they developed an optimal double-via insertion algorithm for the cases with up to three routing layers and the stack-via structure, and then extend the algorithm to handle the general cases. Experiment results show that the method significantly improves the via count, the number of dead vias, double-via insertion rates, and running times.

Lee et al. [70] recently addressed the problem of simultaneously performing redundant-via insertion and line-end extension in the post-routing stage under via-density consideration. Via-density rules constrain the maximum and minimum numbers of vias in each layout area of a predefined size. A two-stage approach was proposed to solve this problem. In the first stage, the approach ignores the maximum via density rule and tries to insert redundant vias and line-end extended vias as many as possible. This is done by formulating the problem as a maximum weighted independent-set (MWIS) problem and solved by a heuristic modified from Lee and Shen [69] equipped with two speed-up techniques. In the second stage, excess redundant vias are removed from the violating regions such that after the removal, the

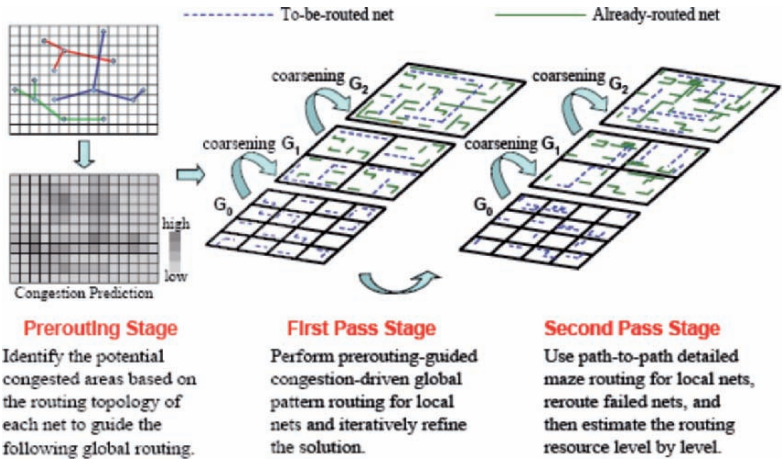

Figure 2-10. The Two-Pass, Bottom-Up Routing Framework.

maximum via-density rule is met while the total number of redundant vias removed is minimized. This density-aware redundant-via removal problem is formulated as a set of smaller zero-one integer linear programming (0–1 ILP) problems, each of which can be solved independently without sacrificing the optimality. This is the first work which concurrently considers redundant-via insertion, line-end extension, and via density.

1.2.4 X-Architecture

The processes that are used to manufacture deep-submicron integrated circuits themselves give rise to special rules that must be adhered to. Today, many conventional systems still rely on time-consuming post-layout fixes to address compliance to these complex rules. Physical design tools must be able to process these rules and automatically generate layouts that conform to these rules, such that post-layout fixing is avoided.

While diagonal wiring has been discussed for years, and short diagonal jogs have even been used for years, pervasive diagonal wiring has not been used on an IC before 2002. The fundamental reasons for this are not manufacturing limitations, as might be suspected, but EDA limitations, and the changes required to take full advantage of the X-architecture are significant and numerous. In particular, routing must be not only octilinear, but also in non-preferred direction. In addition, significant changes are required at least in floorplanning, placement, routing, extraction, power routing, clock routing, and wire length estimation.

Beyond 90 nm, routers will have to optimize the wiring specifically to facilitate manufacturing processes. Nanometer designs will challenge any

router that is not designed specifically to account for these advanced process considerations.

The most prevailing consortium that advocates routing at 45-degree increments is the X-initiative [6]. While lithographic considerations can impinge on the use of arbitrary angles for wiring, now the use of 45-degree wires is fully supported by nearly all current manufacturing technologies. A more realistic example on the left side of Figure 2-11 shows a portion of a commercial chip, routed with a commercial Manhattan router. On the right side of Figure 2-11 is the routing solution using X-architecture. In 2004, Toshiba and Cadence have launched the industry's first commercial system-on-chip (SoC) devices built on the innovative X-architecture design [6]. Toshiba says the TC90400XBG digital-media application processor is approximately 11% faster than comparable Manhattan-layout embedded chips in its product line. Thus, the X-architecture, the first production-worthy approach to the pervasive use of diagonal interconnect, shows promise to reduce the total interconnect while simultaneously improving the chip performance, power, and cost.

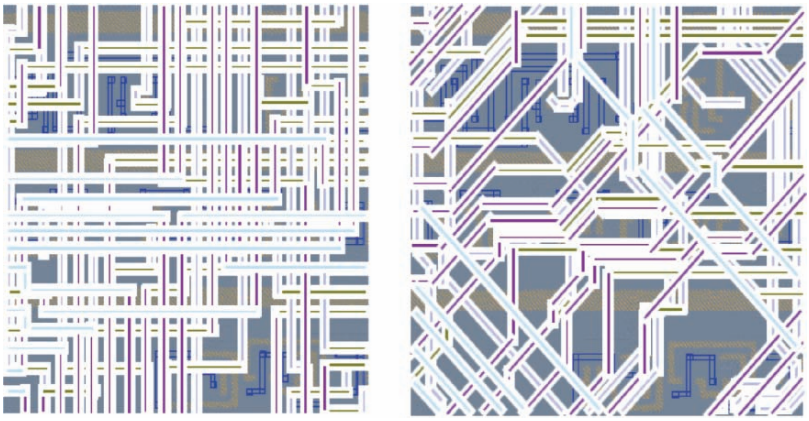

Figure 2-11. Contrasting Manhattan (left) and X Routing (right).

Chapter 3

MULTILEVEL FULL-CHIP ROUTING CONSIDERING CROSSTALK AND PERFORMANCE

In this chapter, we present a novel framework for fast multilevel routing considering crosstalk and performance optimization. To handle the crosstalk minimization problem, we incorporate an intermediate stage of layer/track assignment into the multilevel routing framework. For performance-driven routing, we use a minimum-radius minimum-cost spanning-tree (MRMCST) heuristic for global routing.

1. INTRODUCTION

With decreasing feature sizes, higher clock rates, and increasing interconnect densities, crosstalk has become a major concern of comparable importance to area and timing in IC design. Crosstalk profoundly affects the circuit performance in very deep submicron (VDSM) technology; it is introduced by a coupling between two neighboring wires. For example, two adjacent wires form the coupling capacitor. A voltage or a current change on one wire can thus interfere with the signal on the other wire. Crosstalk is an unwanted variation which makes the behavior of a manufactured circuit deviate from the expected response. The deleterious influences of crosstalk can be classified into two categories. One is malfunctioning, which makes the logic values of circuit nodes differ from what we desire; the other is timing change, which is caused by switching behavior. Therefore, in addition to routability and timing performance, an effective nanometer router must have the ability and the authority to enact wire spacing, layer switching, net reordering, and parallel wire reduction to correct crosstalk issues in routing (see Figure 3-1).

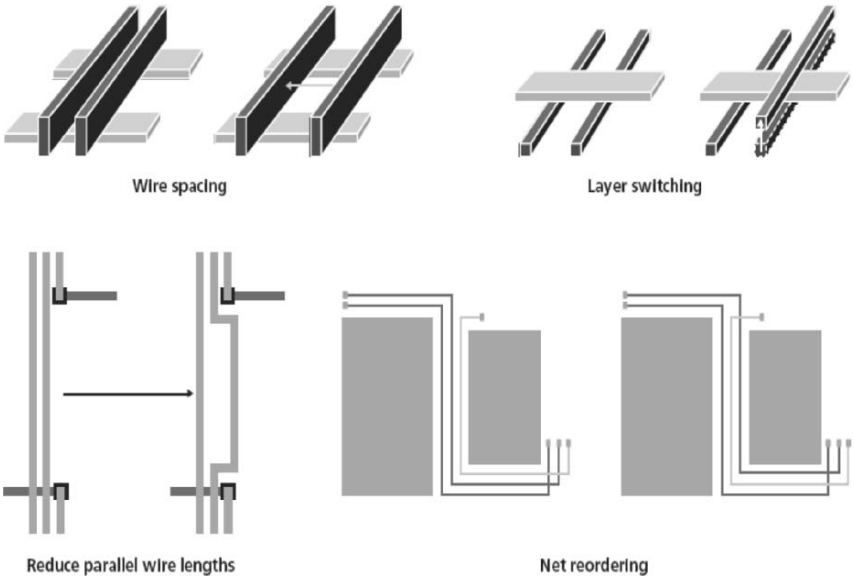

Figure 3-1. Correcting Crosstalk Issues During Detailed Routing.

A framework similar to multilevel routing was presented in Hayashi and Tsukiyama [46], Lin et al. [75], and Marek-Sadowska [78]. Lin et al. [75] and Hayashi and Tsukiyama [46] presented hybrid hierarchical global routers for multilayer VLSI's, in which both bottom-up (coarsening) and top-down (uncoarsening) techniques were used for global routing. Marek-Sadowska [78] proposed a global router based on outer-most loop approach. The approach is similar to the coarsening stage of multilevel routing. Recently, Cong et al. [31] proposed a pioneering multilevel global-routing approach for large-scale, full-chip, and routability-driven routing. They later proposed an enhanced multilevel routing system, named MARS [36], which incorporates resource reservation, a graph-based Steiner tree heuristic and a history-based multi-iteration scheme to improve the quality of the multilevel-routing algorithm in Cong et al. [31]. The final results of both multilevel algorithms are tile-to-tile paths for all the nets. The results are then fed into a detailed router to find the exact connection for each net. Lin and Chang [19, 74] also proposed a multi-level approach for full-chip routing, which considers both routability and perform-ance. This framework integrates global routing, detailed routing, and resource estimation together at each level, leading to more accurate routing resource estimation during coarsening and thus facilitating the solution refinement during uncoarsening. Their experimental results show the best routability among many previous works.

Different from the aforementioned works, our proposed multilevel routing framework has the following distinguished features:

(1) A new framework that performs congestion-driven global routing at the coarsening stage, followed by an intermediate stage of routing layer/track assignment for crosstalk optimization, then a detailed routing at the uncoarsening stage. By performing detailed routing after the layer/track assignment, we can preserve more flexibility for allocating nets for crosstalk optimization.
(2) The MRMCST heuristic [89] is adopted to construct routing trees for performance optimization.
(3) A point-to-path maze-searching algorithm is proposed for better wire length and routability optimization.
(4) An efficient and effective layer/track assignment scheme is incorporated for crosstalk and run-time optimization.

Figure 3-2 shows our multilevel framework. Given a netlist, we first run the MRMCST algorithm to construct the topology for each net, and then decompose each net into 2-pin connections, with each connection corresponding to an edge of the MRMCST. Our multilevel framework starts with coarsening of the finest tiles of level 0. At each level, pattern routing is used for routability-driven global routing. After the coarsening stage, we perform a crosstalk-driven layer/track assignment for crosstalk optimization. At the uncoarsening stage, we perform detailed routing. Further, the unroutable nets are handled by point-to-path maze routing [19, 36, 74] and rip-up and re-route to refine the routing solution level by level.

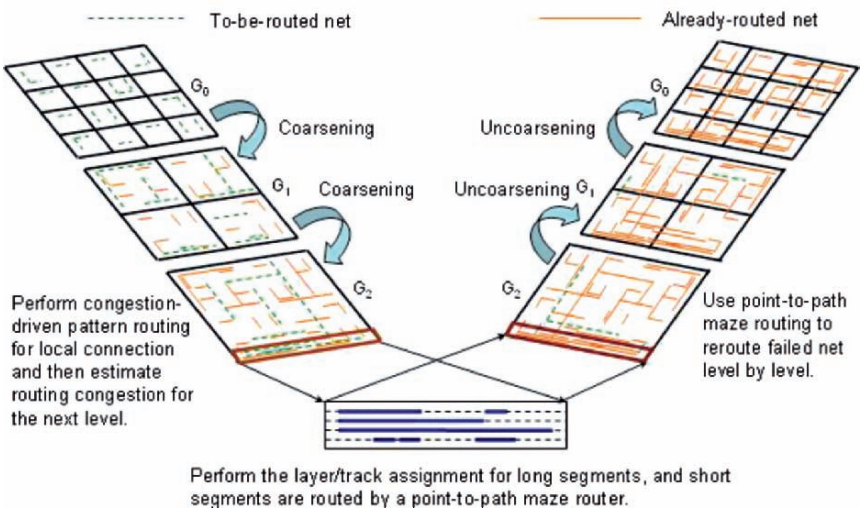

Figure 3-2. Multilevel Routing Framework for Crosstalk and Performance Optimization.

The rest of this chapter is organized as follows. The second section presents the delay model for our performance-driven routing. The third section presents our novel framework for run-time and crosstalk optimization. Experimental results are shown in the fourth section, and a summary given in the fifth section.

2. ELMORE DELAY MODEL

Before we step forward to performance-driven routing, the delay model used, Elmore delay model [40], should first be introduced. As shown in Figure 3-3, a π-model is used to model a wire segment. A via is also modeled as a π-model, with R and C values twice of the wire segment ones. All sinks and the source of a net are treated as drivers, modeled by a capacitance connected to the upstream and a resistance connected to the downstream. Calculation of Elmore delay is obtained by adding the products of these sections of resistances and the downstream capacitances together.

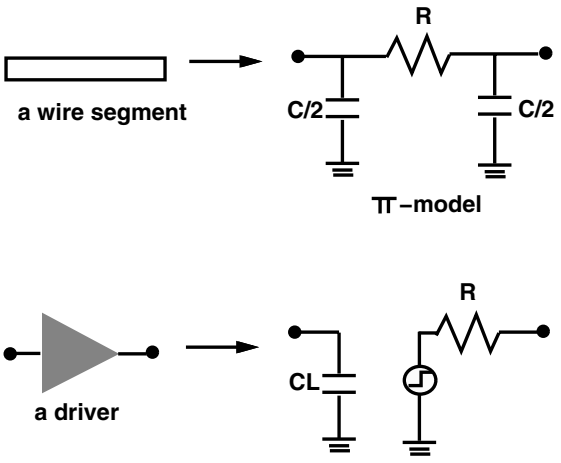

Figure 3-3. Models for a Wire Segment and a Driver.

Let T be an RC tree (a tree with resistances and capacitances) with points $P = \{p_1, p_2, \ldots, p_n\}$; c_i be the capacitance of p_i; r_i the resistance of the edge between p_i and its immediate predecessor. The subtree capacitance at node i is given as:

$$C_i = c_i + \sum_{j \in S_i} C_j,$$

where S_i is the set of all the immediate successors of p_i. Let $\delta_{i,j}$ be the path between p_i and p_j, excluding p_i and including p_j. Then the delay between two nodes i and j is:

$$t_{ij} = \sum_{j \in \delta_{i,j}} r_j C_j,$$

Now we show an example to calculate Elmore delay. Figure 3-4 shows the tree of a net. Applying the π-model to this tree results in an RC tree, which delay is:

$$t_{03} = r_0(c_1 + c_2 + c_3 + c_4 + c_1^s + c_2^s + c_3^s) + r_2(\frac{c_2}{2} + c_3 + c_4 + c_2^s + c_3^s) + r_4(\frac{c_4}{2} + c_3^s)$$

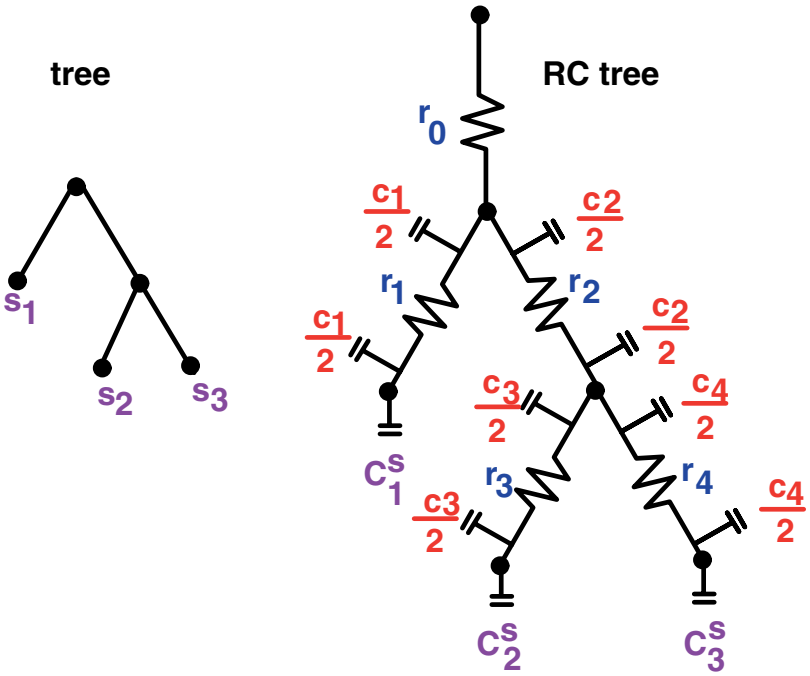

Figure 3-4. A Tree and Its Corresponding RC Tree.

3. MULTILEVEL ROUTING FRAMEWORK

Our multilevel routing algorithm is inspired by the work in Chang and Lin [19, 74]. Nevertheless, different from the framework of Chang and Lin [19, 74] that integrates global routing, detailed routing, and resource estimation together at each level, we perform global routing in the coarsening stage,

followed by layer/track assignment in an intermediate stage, and then detailed routing in the uncoarsening stage. At the coarsening stage, a fast congestion-driven pattern routing [64] is used for global routing level by level. After the coarsening stage, we perform layer/track assignment for crosstalk optimization. At this intermediate stage, long and straight segments tend to be assigned to specified layers/tracks, leading to more efficient detailed routing in the uncoarsening stage since often only short segments need to be handled during detailed routing. At the uncoarsening stage, the unroutable nets are routed by point-to-path maze routing and by rip-up and reroute to refine the routing solution level by level.

3.1 Performance-Driven Routing Tree Construction

In VDSM IC designs, interconnection delay dominates the performance of a circuit. Therefore, improving the wire delay also improves the overall chip performance. Many techniques have been developed to facilitate high-performance IC designs. For example, the algorithms for constructing performance-driven routing trees have received much attention [32]. Given a point set as shown in Figure 3-5(a), the minimum spanning tree (MST) topology leads to the minimum total wire length, and thus congestion is often easier to be controlled than other topologies. However, its topology may result in longer critical paths and degrade the circuit performance (see Figure 3-5(b)). In contrast, a shortest path tree (SPT) may result in the best performance, but its total wire length (and congestion) may be significantly larger than the length obtained by the MST algorithm (see Figure 3-5(c)). Cong et al. used the idea of incrementally modifying an MST to construct a performance-driven routing tree (a.k.a. shallow-light tree) for a smooth trade-off between the tree radius (maximum signal delay) and the tree cost (total interconnection length) (see Figure 3-5(d)). On the one hand, minimizing wire length minimizes a driver's output resistance and the total wire capacitance. On the other hand, minimizing the path length from the source to a sink also minimizes loading capacitance. Thus, both wire length and path length minimization are comparably important for RC delay minimization.

Different from the work presented in Cong et al. [32], our algorithm tries to find a timing-driven routing tree. We make use of the MRMCST, i.e., a minimum-cost spanning tree with a minimum radius. Since finding the MRMCST is NP-hard [89], we resort to a heuristic to obtain efficient solutions.

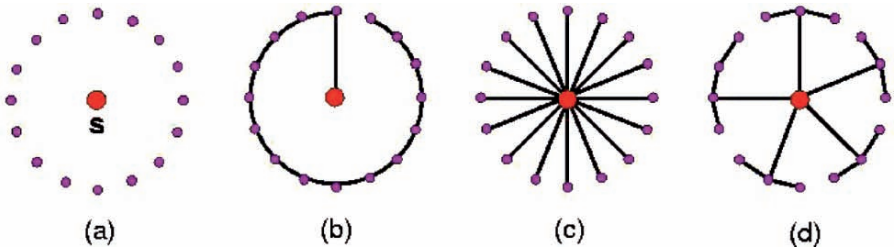

Figure 3-5. (a) Point Set, (b) Minimum Spanning Tree, (c) Shortest Path Tree, and (d) Shallow-Light Tree.

Given a vertex v in a graph $G = (V, E)$, its *eccentricity*, denoted by *ecc* (v), is the distance from v to the farthest vertex in G, which is also referred to as the *radius* of G with respect to v. The *diameter* of a graph is the longest path between any two vertices in the graph. The pseudo-center of a graph $G = (V, E)$, denoted by *pc* (G), is a point on an edge or a vertex of V such that the distances from *pc* to the farthest vertices of V are the same. It is known that the pseudo-center must belong to the diameter of the graph, and *ecc* (*pc* (G)) is the radius of G [47]. Note that given an edge-weighted graph $G = (V, E)$, its MST in general is not unique. The *essential edges* are those edges that must be included in every MST of G, and the *optional edges* are those that may be included in an MST of G.

We shall modify the edge-coloring process of $G = (V, E)$ introduced by Tarjan [96] to color the essential edges blue, the optional edges green, and the non-MST edges red.

Initially there are $n = |V|$ disjoint components, each containing a vertex of V. As edges are colored green or blue, disjoint components containing the end vertices of newly colored (green or blue) edges are merged together to form a new component. When the number of components becomes one, the algorithm will terminate and the remaining uncolored edges are colored red.

The set of blue (or essential) edges must belong to every MST and the set of green (or optional) edges may belong to an MST. The former is referred to as the intersection graph of all the MSTs, denoted as MSTIG, and the single component that remains in the above edge-coloring algorithm is referred to as the union graph of all the MSTs, denoted as MSTUG [89]. The edge-coloring algorithm is summarized in Figure 3-6. It can be shown that the MSTUG and MSTIG can be constructed in $O(m \lg n)$ time, where m is the number of edges and n is the number of vertices.

Algorithm : MSTUG and MSTIG(G)
Input : A connected graph $G = (V, E)$; in which each edge $e \in E$ has a cost, $cost(e)$;
Output : Partition E into three sets, GREEN (set of optional edges), BLUE (set of essential edges), and RED (set of discarded edges).

begin
1 Partition the edges of G into equivalence classes $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{max}$ s.t. two distinct edges are in the same class iff they have the same edge cost.
2 **while** (there exists more than one component) **do**
3 **For** each $e \in \varepsilon_i$
if both ends of e are in the same current component
then color it RED and delete it from ε_i
4 **If** $|\varepsilon_i| = 0$ **then** goto Step 6
else if $|\varepsilon_i| = 1$ **then** color it BLUE and goto Step 6.
else color them GREEN.
5 **For** each GREEN edge $e \in \varepsilon_i$
if e is a bridge in the current component
then recolor it BLUE.
6 **If** there is only one component (i.e. connected)
then color all uncolored edges RED and stop.
end

Figure 3-6. Algorithm for Constructing an MSTUG and an MSTIG.

Note that the MSTIG consists solely of blue edges, and it may contain a forest of more than one tree, and these blue trees are interconnected by green or optional edges to form the MSTUG. The MRMCST is then obtained by selecting the optional edges in an optimal manner to connect the blue trees.

Since the problem of finding the MRMCST is NP-hard [89], heuristics are proposed to obtain suboptimal solutions. The strategy by which the optional edges are selected determines the *quality* of the suboptimal MRMCST. A greedy method, called *locally optimal connection strategy* (LOCS), was introduced in Seo and Lee [89]. We have implemented it with some modifications and incorporated it into our multilevel framework.

Let the blue tree containing the source s be denoted T_s. If there exist more than one optional edge incident to a vertex in T_s, we break the tie by choosing the edge $e = (p, q)$, where $p \in T_s$, and $q \in T$, to minimize $f(e, T)$ defined below:

$$f(e,T) = dist(s,p) + \cos t(e) + dist(q, pc(T)) + ecc(pc(T)),$$

where $pc(T)$ denotes the pseudo-center of T, $dist(s, p)$ is the distance from s to p, and $cost(e)$ is the length of edge e. The blue tree T is then merged with T_s to form a new *super* blue tree T_s.

Figure 3-7 shows an example of the LOCS for two blue trees B_1 and B_2, where s is the source. In this example, the LOCS will select edge $e = (d, i)$ to connect B_1 and B_2, and the path of $ecc(s)$ will update to s-c-pc_1-d-i-pc_2-j-k.

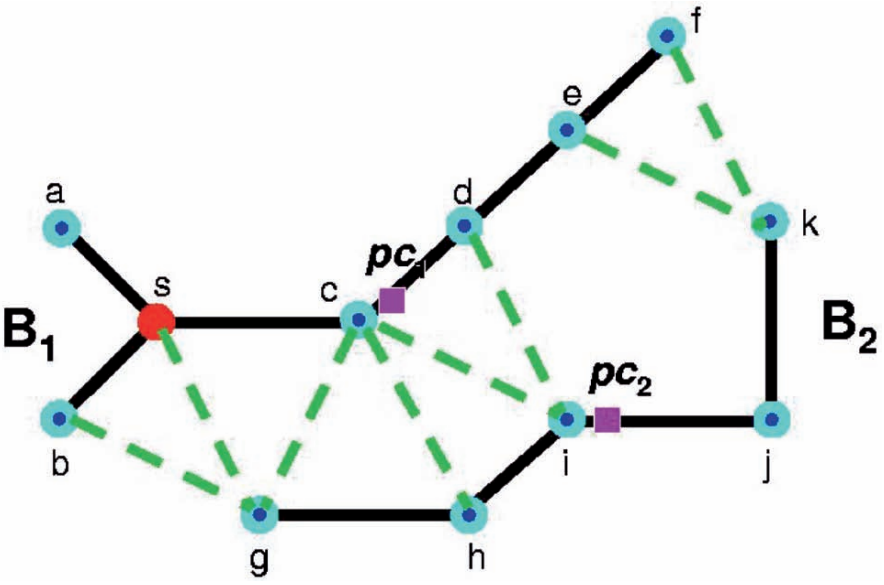

Figure 3-7. Example of the Locally Optimal Connection Strategy (LOCS).

We then apply LOCS successively starting with the blue-tree containing the source until all the blue-trees are connected. Then, we obtain a sub-optimal MRMCST.

The sub-MRMCST algorithm that employs LOCS is summarized in Figure 3-8. Figure 3-9(c) shows the suboptimal MRMCST obtained from the graph shown in Figure 3-9(b), where MSTUG and MSTIG were constructed.

```
Algorithm : sub-MRMCST
   Input  : MSTUG, MSTIG and source s
  Output  : A sub-optimal MRMCST
begin
1     T_s = The blue tree containing s;
2     NumOfTrees = number of blue trees;
3     Find the pseudo-center pc for each blue tree;
4     For all vertices not in T_s, find the distance to
      its corresponding pc;
5     For each blue tree T_i, find ecc(pc(T_i)) and
      optional edges incident to it;
6     For all vertices in T_s, find their distances from s;
7     Mark all vertices in T_s "visited";
8     For each optional edge e incident to T_s, mark it
      "inserted" and call
      InsHeap(e, FirstCost, SecondCost);
9     while (NumOfTrees > 1) do
10        MinE = (v_1, v_2) = PopHeap();
11        while (both v and w are marked "visited")
          do
12            MinE = (v_1, v_2) = PopHeap();
13        NumOfTrees - -;
14        T_s = T_s + MinE = (v_1, v_2);
15        Mark v_2 "visited"; find dist(s, v_2);
16        For each "unvisited" vertex w in the blue
          tree containing v_2 (say T_i)
17            Traverse T_i from v_2 to update
              dist(s, w) and mark w "visited";
18        For each "uninserted" optional edge
          e = (v_3, v_4), v_3 ∈ T_i and v_4 ∈ T_j, j ≠ i,
19            FirstCost = cost(v_3, v_4);
20            SecondCost = dist(s, v_3) + cost(v_3, v_4)
                           + dist(v_4, pc(T_j)) +
                           ecc(pc(T_j));
21            InsHeap(e, FirstCost, SecondCost);
22            Mark e "inserted";
23    MRMCST = T_s;
end
```

Figure 3-8. A Heuristic for Constructing a Suboptimal MRMCST.

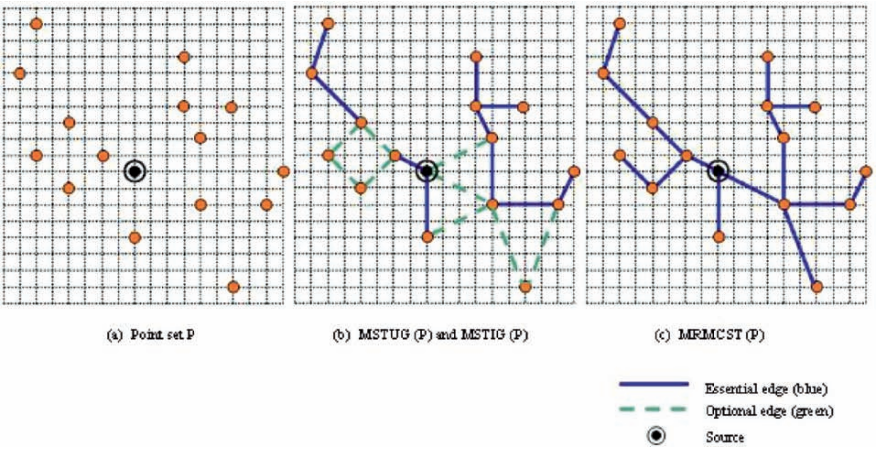

Figure 3-9. Example of MRMCST Construction: (a) Given Vertex Set, (b) MSTUG Containing All Edges and MSTIG Containing All Solid Edges, and (c) Resulting MRMCST.

Theorem 3.1: *The sub-MRMCST heuristic runs in* $O(n + m_{opt} \lg m_{opt})$ *time, where* n *is the number of vertices and* m_{opt} *is the number of optional edges.*

Proof Sketch: The merging (connecting) cost is $O(E_i)$ when the blue tree $T_i = (V_i, E_i)$ is connected to the super blue tree. Hence, the total connecting cost will be $O(n)$. Since every optional edge is inserted into the priority queue exactly once and each insertion/deletion for the priority queue needs $O(\lg m_{opt})$ time, the total time complexity for MRMCST construction is $O(n + m_{opt} \lg m_{opt})$, where n is the number of vertices and m_{opt} is the number of optional edges.

After a suboptimal MRMCST is constructed, timing analysis based on the Elmore delay model is performed from the tree source to all sinks. If a target node violates the timing constraint, we modify the tree topology by deleting this local connection and then tracing back from the target node to the tree source to find a new parent for the connection that can meet the timing constraint (although this process might increase the total wire length and thus the total wire capacitance, the decrease of the path delay due to lower source-to-sink loading capacitance is even more significant). After all nets meet the timing constraint, we start to route them in the coarsening stage.

3.2 Crosstalk-Driven Layer/Track Assignment

As fabrication technology shrinks into the VDSM era, on-chip minimum feature sizes continue to decrease, and devices and interconnection wires are placed in closer proximity in order to reduce interconnection delay and

routing area. The increasing of aspect ratio of wires and the decreasing of interconnect spacing have made the coupling capacitance larger than self-capacitance. In fact, the ratio of coupling capacitance is reported to be even as high as 70% ~ 80% of the total wiring capacitance, even in 0.25 μm technology.

Crosstalk is mostly caused by coupling capacitance between interconnection wires. In general, the crosstalk between two wires is proportional to their coupling capacitance, which is determined by the relative positions of these wires. The coupling capacitance between a pair of parallel wires is proportional to their coupling length, and is inversely proportional to their separating distance. The coupling capacitance between a part of orthogonal wires is negligible in comparison with the coupling capacitance between a pair of parallel wires in current technology. Consequently, it is reasonable to assume that there is crosstalk only between adjacent parallel wires.

Recently, there has been much research on the coupling problem in both global and detailed routing. Zhou and Wong [104] minimized crosstalk at the global routing stage. Chaudhary et al. [20] proposed wire spacing after detailed routing to reduce crosstalk. This technique can be applied as a post-processing and used for improving an existing layout, but it is not suitable for routing.

However, both global routing and detailed routing are not the best stage to address crosstalk. It might be too early to handle crosstalk during global routing since the relative positions and ordering of nets are not determined at this stage; therefore, the best that one can possibly do is to use rough statistical estimators that discourage nets from entering regions where unwanted proximities seem likely. Conversely, it is too late for detailed routing to handle crosstalk since area routers that embed one net at a time may encounter unsolvable rip-up/reroute problems when trying to embed a late-routing net that must traverse a region already dense with conflicting aggressor or victim nets.

To address these problems, Kay and Rutenbar [65] suggested an integer linear programming (ILP)-based layer/track assignment method to do crosstalk optimization. However, the ILP-based approach is very time-consuming and thus not suitable for large and complex design. Batterywala et al. [11] proposed a fast track assignment heuristic considering routability, but crosstalk was not addressed in their work.

Inspired by the work in Batterywala et al. [11], we propose a fast layer/track assignment heuristic for crosstalk optimization. After the coarsening stage, we may obtain several long horizontal and vertical segments. To simplify the layer/track assignment problem, we only assign segments which span more than one complete global cell in a row or a column, and handle short segments during detailed routing. The layer/track assigner works on a

full row or column of the global cell array at a time, where a row (column) is called a *panel*.

We first build a horizontal constraint graph $HCG(V, E)$ for all segments in the panel. Each vertex $v \in V$ corresponds to a segment in the panel. Two vertices v_i and v_j are connected by an edge $e \in E$ iff these segments belong to two different nets and their spans overlap. The edge cost of $e = (v_i, v_j) \in E$ represents the coupling length if v_i and v_j are assigned to adjacent tracks. We define the crosstalk-driven layer (CLA) assignment problem as follows:

Given a set of layers L and a set of segments S, find an assignment of segments to the layers that minimizes the sum of the coupling costs (lengths) of all nets in all layers.

Here the cost for CLA comes from the overlapping lengths of nets since nets are not yet assigned to tracks during the layer assignment and all information we have is the spans of nets. The CLA problem can be formulated as the max-cut, k-coloring (MC) problem [93]. However, the MC problem is NP-complete [93]. Thus, we resort to a simple yet efficient heuristic by constructing a MST from the given HCG. Since a tree can be k colored in linear time if we have k layers, we shall first partition the vertices incident on edges with larger costs (coupling lengths) and allocate the corresponding segments to different layers.

Let R be the set of tracks inside a panel. Each track $t \in R$ can be represented by its set of constituent contiguous intervals. Denoting these intervals by x_i, we have $t \equiv \bigcup x_i$. Each x_i is either:

(1) A blocked interval, where no segment from S can be assigned
(2) An occupied interval, where a segment from S has been assigned
(3) A free interval, where no segment from the set S has yet been assigned

A segment $seg \in S$ is said to be assignable to $t \in T$, $t \equiv \bigcup x_i$, iff $x_i \bigcap seg \neq \phi$, which implies that either x_i is a free interval or is an interval occupied by a segment of the same net. Thus, the crosstalk-driven track assignment (CTA) problem can be defined as follows:

Given a set of tracks T, a set of segments S, and a cost function $F: S \times T \rightarrow N$, which represents the cost of assigning a segment to a track, find an assignment of segments to the tracks that minimizes the sum of the coupling costs (lengths) among adjacent nets of the assignment.

After layer assignment, most of the edges with larger costs in an *HCG* are eliminated, and the *HCG* is decomposed into k subgraphs $subHCG_1$, $subHCG_2$, ..., $subHCG_k$ if we have k layers. Figure 3-10 shows an example

of the track assignment problem for a *subHCG*, where $S = \{a, b, c, d, e, f\}$, $T = \{1, 2, 3, 4\}$, and obstacles on tracks are shaded in grey (e.g., the two obstacles on tracks 3 and 4). We use a bipartite assignment graph to indicate the assignability of segments to tracks. For example, as shown in Figure 3-10(b), edges between vertex *a* and vertices 1, 2, and 3 are introduced since segment *a* can be assigned to track 1, 2, or 3, but not track 4. For easier implementation, we merge the *subHCG* and the bipartite assignment graph into a combination graph, as shown in Figure 3-10(c).

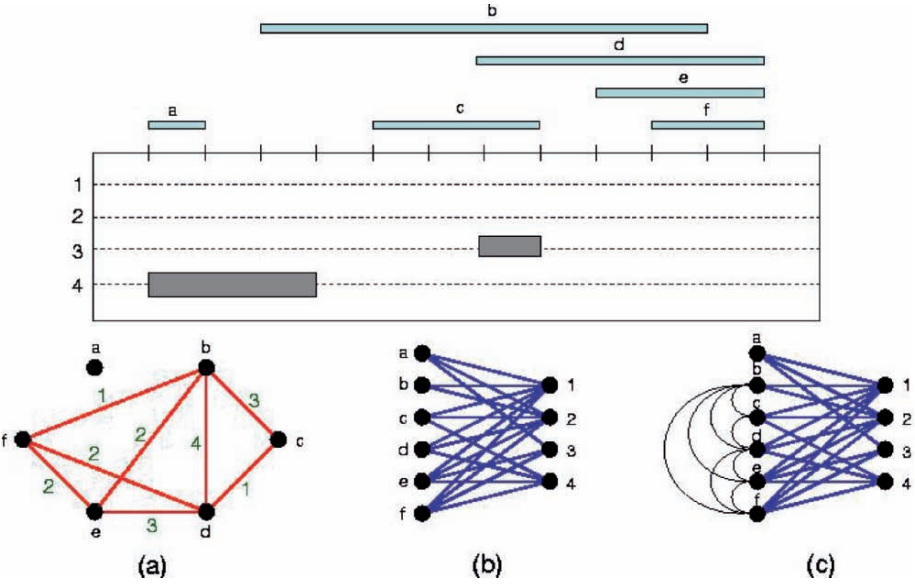

Figure 3-10. Constraint Graph Modeling for Track Assignment: (a) SubHCG for a Given Instance, (b) Corresponding Bipartite Assignment Graph, and (c) Combination Graph.

Since each vertex $v \in V$ corresponds to a segment and each edge $e \in E$ corresponds to the coupling cost in $HCG(V, E)$, the CTA problem can be formulated as the Hamiltonian path problem which has been proven to be NP-complete [37]. We resort to a heuristic for the CTA problem. Our track assignment algorithm starts by finding the maximal sets of conflicting segments. This is equivalent to finding the largest clique V_c in the subgraph $subHCG_i$. Since the HCG is an interval graph [42] (a graph induced from interval interactions), finding the largest clique can be done in polynomial time. The algorithm first assigns one maximal subset of conflicting segments at a time by starting from the largest clique. Then we choose the longest segment in the clique as the source *s* and assign it to the uppermost available track. Then, we choose the min-cost edge (s, i) (and thus the minimal coupling) and assign the segment associated with *i* to the first available

track. If all tracks are occupied, we refer to the net associated with i as a failed net which will be reconsidered at the uncoarsening stage. We repeat the procedure by finding the min-cost edge (i, j) for further processing, where j is an unvisited vertex.

Figure 3-11(a) shows the final track assignment for the instance of Figure 3-10. The maximum clique in the *subHCG* is $\{b, d, e, f\}$, and the longest segment in the clique is b. We thus assign segment b to the uppermost available track, which is track 1. See Figure 3-11(b) for the updated combination graph after assigning b to track 1. Then, our heuristic makes b the source for constructing the Hamiltonian path for the clique. The min-cost edge $e = (b, f)$ incident on b is chosen, and f is assigned to the first available track. See Figure 3-11(c) for the updated combination graph after assigning f to track 2. The process is repeated until all vertices in the clique are visited. We then have the track assignment solution shown in Figure 3-11(a).

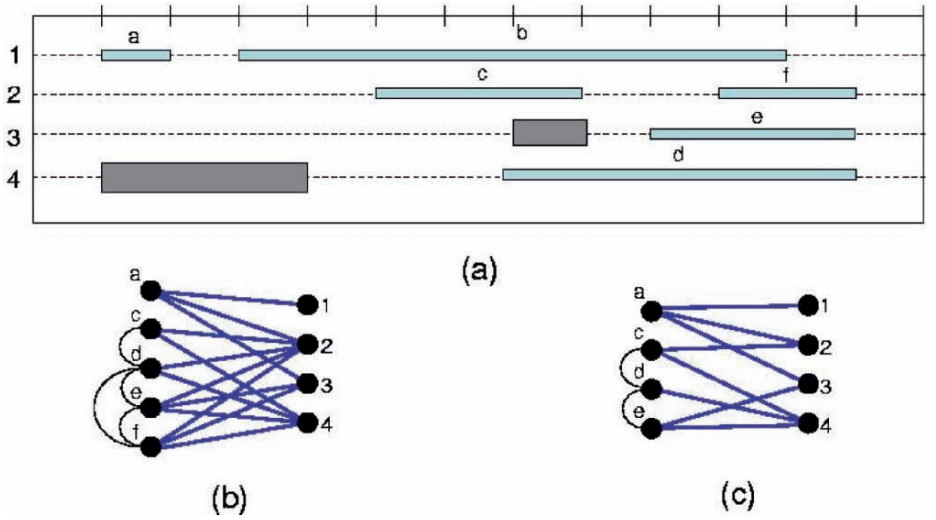

Figure 3-11. Process of Track Assignment: (a) Final Track Assignment for the Instance of Figure 3-10, (b) Resulting Combination Graph After Assigning b to Track 1, and (c) Resulting Combination Graph After Assigning f to Track 2.

After the track assignment, the actual track position of a segment is known. Thus, we can perform point-to-segment maze routing to complete the routing.

4. EXPERIMENTAL RESULTS

We have implemented our crosstalk-driven multilevel system in the C++ language on a 1 GHz SUN Blade 2000 workstation with 1GB memory. We compared our results with Chang and Lin [19, 74] based on the six

benchmark circuits provided by the authors. See Table 3-1 for the benchmark circuits. (Note that the benchmark circuits used in Chang and Lin [19], Cong et al. [31], and Lin and Chang [74] also contain Mcc1, Mcc2, Struct, Prim1 and Prim2. However, as pointed out in Chang and Lin [19, 74], those circuits do not have the information of net sources, and thus we cannot calculate the delay for nets for those benchmarks. Therefore, we shall focus our comparative studies on the six benchmark circuits listed in Table 3-1. The design rules for wire/via widths and wire/via separation for detailed routing are the same as those used in Chang and Lin [19], Cong et al. [31], and Lin and Chang [74].

Table 3-1 describes the set of benchmark circuits. In the table, "Size" gives the layout dimensions, "#Layers" denotes the number of routing layers used, "#Nets" represents the number of 2-pin connections after net decomposition. Since the results reported in Chang and Lin [19, 74] are better than those in Cong et al. [31] and Cong et al. [36], we shall compare our multilevel router with that in Chang and Lin [19, 74].

Table 3-1. The Benchmark Circuits Used in the Experiments of Chapter 3.

Circuits	Size (μm)	#Layers	#Nets	#Pins
S5378	4330x2370	3	3124	4734
S9234	4020x2230	3	2774	4185
S13207	6590x3640	3	6995	10562
S15850	7040x3880	3	8321	12566
S38417	11430x6180	3	21035	32210
S38584	12940x6710	3	28177	42589

To perform experiments on timing-driven routing, we used the same resistance and capacitance parameters as those used in Chang and Lin [19, 74]. First, we constructed a SPT for a net by connecting all sinks directly to their net source to obtain the timing constraints. We then assigned the timing bound of each sink as the multiplication of the constant k and the shortest path delay of the net. A via is modeled as the π-model circuit, with its resistance and capacitance being twice of those of a wire segment, and the Elmore delay model is used for our delay computation. All the parameters were the same as those used in Chang and Lin [19, 74], and both routers were run on the same machine. Experimental results on run-time, routing completion rate, delay, and crosstalk with comparable routability (for routability optimization) are listed in Table 3-2. (Note that we set the timing constraint ratio k used in Chang and Lin [19, 74] to 5.5 to obtain comparable routability with ours for fair comparisons). And the results of timing-driven routing with comparable routability are listed in Table 3-3. (For this experiment, k is set to 2 for Chang

and Lin [19, 74]). In the table, "D_{max}" represents the critical path delay, "D_{avg}" represents the average net delay, "C_{max}" represents the maximum coupling length of a net, and "C_{avg}" represents the average coupling length. Compared with the routability mode of Chang and Lin [19, 74], the experimental results show that our router achieved a 6.7X run-time speedup, reduced the respective maximum and average crosstalk (coupling length) by about 30% and 24%, reduced the respective maximum and average delay by about 15% and 5%. And compared with the timing-driven mode (k = 2 for Chang and Lin [19, 74]), the experimental results show that our router still achieved a 5.9X run-time speedup, reduced the respective maximum and average crosstalk by about 35% and 23%, reduced the respective maximum and average delay by about 7% and 10% in comparable routability, and resulted in fewer failed nets.

Table 3-2. Results on Delay, Crosstalk, Run-Time, and Routing Completion Rate with Comparable Routability.

Circuits	Results of [59]						Our Results					
	D_{max} (ps)	D_{avg} (ps)	C_{max} (μm)	C_{avg} (μm)	Time (s)	Cmp. Rates	D_{max} (ps)	D_{avg} (ps)	C_{max} (μm)	C_{avg} (μm)	Time (s)	Cmp. Rates
S5378	37308	1403	507	25.4	35	99.7%	27577	1258	342	20.2	10.6	99.8%
S9234	25512	1072	579	21.7	26.2	99.7%	23591	1009	426	17.8	8.1	99.9%
S13207	55337	1262	1526	29.2	106.7	99.8%	52034	1243	1211	22.7	22.6	99.8%
S15850	76297	1302	2913	28.3	538.8	99.3%	68317	1253	2274	22.9	62.6	99.7%
S38417	121419	1170	5704	25.6	899.9	99.5%	105575	1146	4732	20.9	71.3	99.8%
S38584	150936	1208	23196	26.8	1953.7	99.6%	131877	1151	18810	22.6	255.6	99.8%
Avg.	1.15	1.05	1.30	1.24	6.7	1	1	1	1	1	1	+0.2%

Table 3-3. Results on Delay, Crosstalk, Run-Time, and Routing Completion Rate with Comparable Routability in Timing-Mode Comparison.

Circuits	Results of [59]						Our Results					
	D_{max} (ps)	D_{avg} (ps)	C_{max} (μm)	C_{avg} (μm)	Time (s)	Cmp. Rates	D_{max} (ps)	D_{avg} (ps)	C_{max} (μm)	C_{avg} (μm)	Time (s)	Cmp. Rates
S5378	13651	798	507	24.7	38.6	94.6 %	12854	751	331	20.3	11.2	95.2 %
S9234	11426	659	579	21.0	27.8	94.3 %	10019	599	426	17.6	8.6	94.2 %
S13207	20149	749	1413	27.7	113.5	93.1 %	18769	693	1109	20.1	24.1	94.4 %
S15850	28049	859	3211	25.9	558.8	93.1 %	25221	743	2510	21.9	73.7	93.6 %
S38417	40500	702	5722	24.6	944.5	93.4 %	38957	670	4385	20.4	90.9	93.7 %
S38584	129267	739	24268	26.7	2187.1	93.7 %	129267	655	17613	22.1	357.0	94.0 %
Avg.	1.07	1.10	1.35	1.23	5.9	1	1	1	1	1	1	+0.4%

The results reveal the effectiveness of the intermediate stage of layer and track assignments and our suboptimal MRMCST for performance-driven routing tree construction. Since many segments are routed in the layer/track assignment stage (which is very efficient), the search space during the uncoarsening stage is significantly reduced. Consequently, the running time and solution quality can be improved simultaneously. Also, compared with Chang and Lin [19, 74] that were based on the classical performance-driven routing tree construction, the experimental results on timing have shown that our suboptimal MRMCST leads to significantly better maximum and average delays.

It should be noted that the coupling capacitance is not included in delay computation for fair comparison with Chang and Lin [19, 74]. If coupling capacitance is considered, our router shall be able to obtain even better timing reduction due to the significant crosstalk reduction.

To demonstrate the effectiveness of the heuristics used in CLA and CTA, we also conducted the following two experiments: first, we performed CLA only for crosstalk minimization, and then the track assignment greedily without considering the cost of the coupling length; second, we simply assigned longer segments to lower layers and then performed CTA for crosstalk minimization. The results are compared to that reported above by minimizing crosstalk using both CLA and CTA. As shown in Table 3-4, performing CLA and CTA together can reduce the respective coupling costs by 4.6% (4.4%) and 10.2% (10.0%), compared with the results obtained by performing CLA and CTA alone.

Table 3-4. Results of CLA and CTA Comparisons.

Circuits	Results with only CLA		Results with only CTA		Our Results	
	C_{max} (μm)	C_{avg} (μm)	C_{max} (μm)	C_{avg} (μm)	C_{max} (μm)	C_{avg} (μm)
S5378	355	20.2	416	22.4	342	20.2
S9234	480	18.1	501	20.1	426	17.8
S13207	1312	23.6	1373	24.9	1211	22.7
S15850	2398	24.8	2368	24.8	2274	22.9
S38417	4780	23.8	4732	24.0	4732	20.9
S38584	18136	22.9	19618	23.3	18810	22.6
Comp.	+4.6%	+4.4%	+10.2%	+10.0%	–	–

Figure 3-12 shows the routing solution for benchmark "s5378" obtained from the crosstalk- and performance-driven multilevel router.

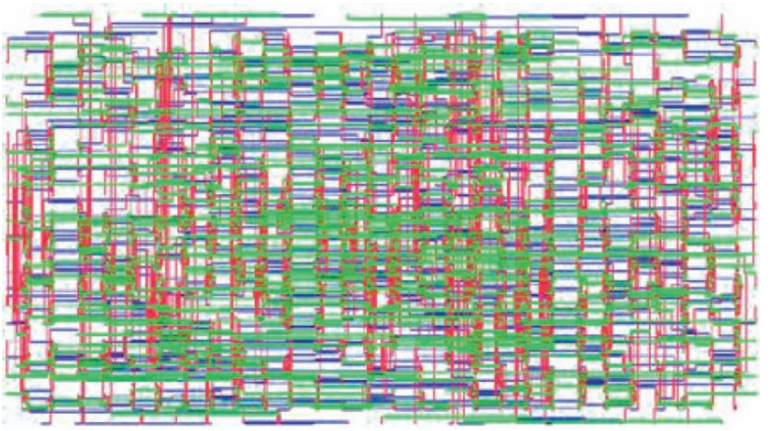

Figure 3-12. Routing Solution for Benchmark "s5378."

5. SUMMARY

We have implemented a crosstalk-driven multilevel system in the C++ language on a 1 GHz SUN Blade 2000 workstation with 1GB memory. We compared our results with Chang and Lin [19, 74] based on the six benchmark circuits provided by the authors. See Table 3-1 for the benchmark circuits. Note that the benchmark circuits used in Chang and Lin [19], Cong et al. [31], and Lin and Chang [74] also contain Mcc1, Mcc2, Struct, Prim1 and Prim2. However, as pointed out in Chang and Lin [19, 74], those circuits do not have the information of net sources, and thus we cannot calculate the delay for nets for those benchmarks. Therefore, we shall focus our comparative studies on the six benchmark circuits listed in Table 3-1. The design rules for wire/via widths and wire/via separation for detailed routing are the same as those used in Chang and Lin [19], Cong et al. [31], and Lin and Chang [74].

This page intentionally blank

Chapter 4

MULTILEVEL FULL-CHIP ROUTING CONSIDERING ANTENNA EFFECT AVOIDANCE

As technology advances into nanometer territory, the antenna effect problem has caused significant impact on routing tools. The antenna effect is a phenomenon of plasma-induced gate oxide degradation caused by charge accumulation on conductors. It directly influences reliability, manufacturability and yield of VLSI circuits, especially in deep-submicron technology using high-density plasma. Furthermore, the continuous increase of the problem size in IC routing is also a great challenge to existing routing algorithms. In this chapter, we present a framework for multilevel full-chip routing with antenna avoidance using built-in jumper insertion approach.

1. INTRODUCTION

With the continuous and rapid increase in complexity of VLSI designs and fabrication technologies, manufacturing yield and product reliability are now among the most important design issues, such as quick turn-around time, small die size, high speed, low power, and so on [48]. The fine feature-size of modern IC technologies is typically achieved by using plasma-based processes. As the technology enters the deep-submicron era, more stringent process requirements cause some advanced high-density plasma reactors adopted in the production lines to achieve fine-line patterns [67]. However, these plasma-based processes have a tendency to charge conducting components of a fabricated structure. The existing experimental evidence indicates that charging may affect the quality of the thin oxide. This is called the *antenna effect* (also called "plasma-induced gate oxide damage"). During metallization, chips are usually processed "from

the bulk up," each time adding an additional layer of interconnect. Before the chip is fully assembled, individual nets are disconnected and consist of pieces of floating metal. Long-floating interconnects act as temporary capacitors to store charges gained from the energy provided during fabrication steps such as CMP. A random discharge of a floating node due to subsequent process steps could permanently damage transistors, rendering the IC useless [61, 88, 90, 100]. For example, the exposed polysilicon and metal structures connected to a thin oxide transistor collect charge from the processing environment (e.g., reactive ion etch) and may damage the transistor when the discharging current flows through the thin oxide.

Although the mechanism of the gate oxide damage is not very well understood, the experimental relationships between the amount of damage and the antennas have been studied [90, 100]. Since the plasma damage is caused by the electrical charging of devices during plasma processes, the damage increases with an increase in the area of the exposed conductor (antenna) during the plasma process. In order to reduce or prevent damage to the gate oxide from the plasma process, and thus to ensure reliability of a chip, a circuit layout rule that considers the antenna effect (antenna rule) is employed. The conventional antenna rule restricts the maximum antenna size or antenna ratio allowed for circuit layout. Recent studies show that the damage, considering all plasma-based manufacturing operations, increases in proportion to both the area and the perimeter of antennas [88, 91]. A more accurate model considering the cumulative oxide damage is discussed in Chen and Koren [26]. These models provide a good guideline for routers or physical EDA tools to help reduce damage from the antenna effect and get higher yield and reliability.

Maly et al. [97] proposed a method for detecting an antenna violation. They calculated both the area and the perimeter of antennas using a general-purpose design rule checking (DRC) program. However, the method does not indicate any measures to feed the antenna information back to layout generation. On the other hand, Wang et al. [98] proposed a channel router which considers the antenna effect. They introduced a layer restriction to a conventional channel router, which limits the maximum length of the wires with antenna problems. Chen and Koren [26] also proposed a network bipartitioning approach for layer restriction. But both of them consider antenna effect minimization only in 3-layer channel routing. Shirota et al. [91] proposed a router which combines a traditional router with a modification of wires for reducing the antenna effect damage, using a rip-up and reroute method. But this method fixes the antenna after routing; it is not a built-in approach. The diode insertion method is also proposed to fix the antenna problem [21, 58]. It is the simplest way to deal with antenna problems by forcing a discharge path. But in today's high-density VLSI layouts, there is simply not enough room for "under-the-wire" diode insertion for all wires. Furthermore, it will cause congestion, add capacitance to the

net, reduce room for ECO, and generate leakage power. Thus, designers often prefer a jumper-based solution to a diode-based solution for more advanced process technology.

Routing complexity is also an important problem for modern routers. To cope with the increasing complexity, researchers have proposed multilevel approaches to handle the problem [31, 36, 52, 74]. The multilevel framework has attracted much attention in the literature recently [35]. It employs a two-stage technique: coarsening followed by uncoarsening. The coarsening stage iteratively groups a set of circuit components (e.g., circuit nodes, cells, modules, routing tiles) based on a predefined cost metric, until the number of components being considered falls below a certain threshold. Then, the uncoarsening stage iteratively ungroups a set of previously clustered circuit components and refines the solution by using a combinatorial optimization technique (e.g., simulated annealing, local refinement). The multilevel framework has been successfully applied to partitioning, floorplanning, placement, and routing in VLSI physical design.

In this chapter, we propose a multilevel router for reducing the antenna effect damage by built-in jumper insertion. The three main features of the proposed method are:

(1) A bottom-up approach is used for jumper prediction.
(2) A state-of-the-art multilevel routing framework [52] is adopted for run-time speedup and antenna fixing.
(3) Nets that failed to route or violate antenna rule are routed at the uncoarsening stage for better routing completion.

Experimental results show that our approach reduced 100% antenna-violated gates and results in fewer wire length, vias, and delay increase.

The rest of this chapter is organized as follows. Section 4.2 describes the antenna effect damage. Section 4.3 presents our multilevel framework for reducing antenna effect damage. Experimental results are shown in Section 4.4, and a summary given in Section 4.5.

2. ANTENNA EFFECT DAMAGE

The mechanism of antenna damage is not fully understood, but there is experimental evidence indicating when charging occurs and how it may affect the quality of gate oxide [90, 100]. Charging occurs when conductor layers not covered by a shielding layer of oxide are directly exposed to plasma. The amount of such charging is proportional to this plasma-exposed area. If the charged conductor layers are connected only to the gate oxide,

Fowler–Nordheim (F–N) tunneling current will discharge through the thin oxide and cause damage to it.

Process antenna rules adhere to the design requirement that the total charge accumulated on metal connected to a polysilicon gate during any stage of metalization cannot exceed a certain threshold, beyond which the excessive charge accumulation may permanently damage the gate. Let *gate-strength* (g, L) be the maximum length of a wire of minimum width on layer L that can be directly connected to the gate g without causing an antenna violation. The larger the values of gate-strength, the easier it is to fix the antenna violation. In 0.18 μm technology and above, gate-strength of 1,000 μm and above is very common, and fixing by post-processing suffices. In 0.13 μm and below technology, however, the average and worst-case gate-strength's are substantially reduced (about 20 ~ 100 μm [73, 83]). This is due in part to the use of cells with small gate areas (for example, extensive use of low-power cells) and a tightening of the antenna ratio. When the worst-case gate-strength is merely a handful of cell rows, antenna fixing becomes very challenging.

On the other hand, if the amount of charge collected by connected conductor layer patterns could be released through a low-impedance path, such as a previously formed diffusion layer pattern (e.g., source/drain), it will not introduce the gate oxide damage.

A more accurate analysis of the cause of the charging collected during the deep submicron VLSI manufacturing operations shows that the perimeter length of conductor layer patterns must also be included in the calculation [91]. There are three types of plasma-based manufacturing processes:

(1) Conductor layer pattern etching process: The amount of accumulated charge is proportional to the perimeter length of conductor layer patterns. Etching process is used to divide conductor layer plates into innumerable routing patterns. In the late stage of the process, the perimeters of the routings are directly exposed to plasma.
(2) Ashing process: The amount of accumulated charge is proportional to the area of the conductor layer patterns. Ashing process is used to remove remaining photo resist layers after etching process of a conductor layer. In the late stage of the process, the area of a conductor layer pattern is directly exposed to plasma.
(3) Contact etching process: The amount of accumulated charge is proportional to the total area of the contacts. Contact etching process is used to dig holes between two conductor layers. In the late stage of the process, the area of all the contacts on the lower conductor layer pattern is directly exposed to plasma.

As a result, considering the above three plasma-based processes, the risk of gate oxide damage is proportional to the area and perimeter length of antenna routings and inversely proportional to the area and perimeter length of the gate oxide.

There are three kinds of solutions to reduce the antenna effect [21]:

(1) Jumper insertion: Break only signal wires with antenna violation and route to the highest level by jumper insertion. This reduces the charge amount for violated nets during manufacturing.
(2) Embedded protection diode: Add protection diodes on every input port for every standard cell. Since these diodes are embedded and fixed, they consume unnecessary area when there is no violation at the connecting wire.
(3) Diode inserting after placement and routing: Fix those wires with antenna violations that have enough room for "under-the-wire" diode insertion. During wafer manufacturing, all the inserted diodes are floating (or grounded). One diode can be used to protect all input ports that are connected to the same output ports. But this approach works only if there is enough room for diode insertion (see Figure 4-1).

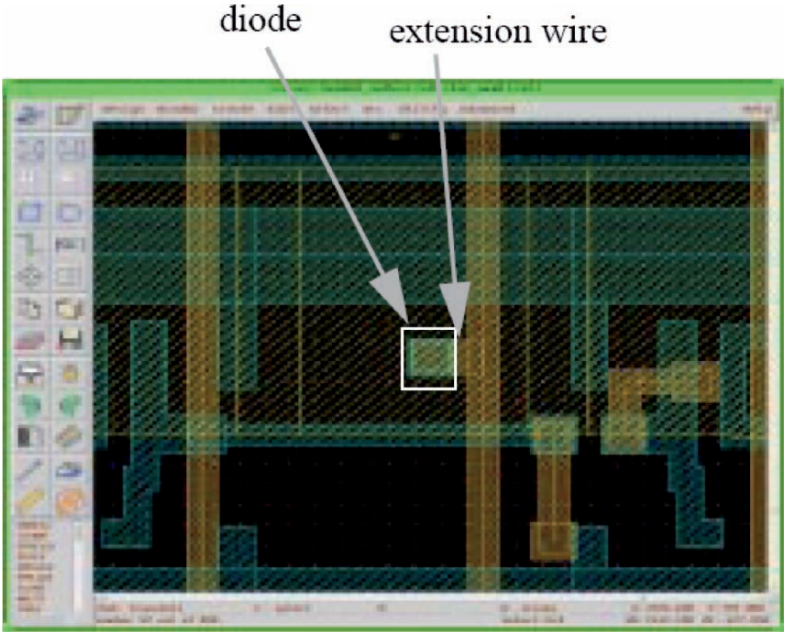

Figure 4-1. Diode Insertion with Wire Extension.

Jumper insertion is the most popular way to solve the antenna problem. We show its usage in the following example. Suppose we have a two-terminal net in which "a" is the source node and "b" is the terminal node (see Figures 4-2(a) (b)). In this case, the approximate gate-strength of "b" is the sum of the length of segments 4, 5, and 6, which may violate the minimum allowable gate-strength. If we add a jumper at the long segment 5 (see Figures 4-3(a) (b)), the approximate gate-strength of "b" is just the sum of the length of segments 8, 9, and 10, which will not violate the minimum allowable gate-strength. Thus, if we add jumpers appropriately, the antenna problem can be easily solved.

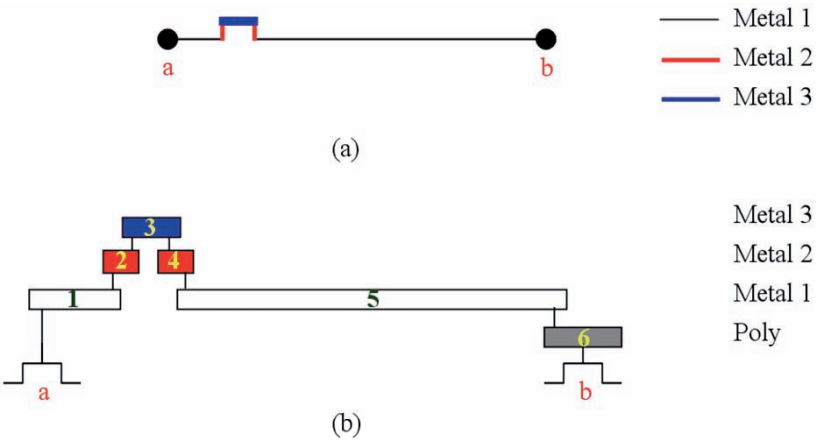

Figure 4-2. (a) A 2-Pin Net and (b) Its Cross Section.

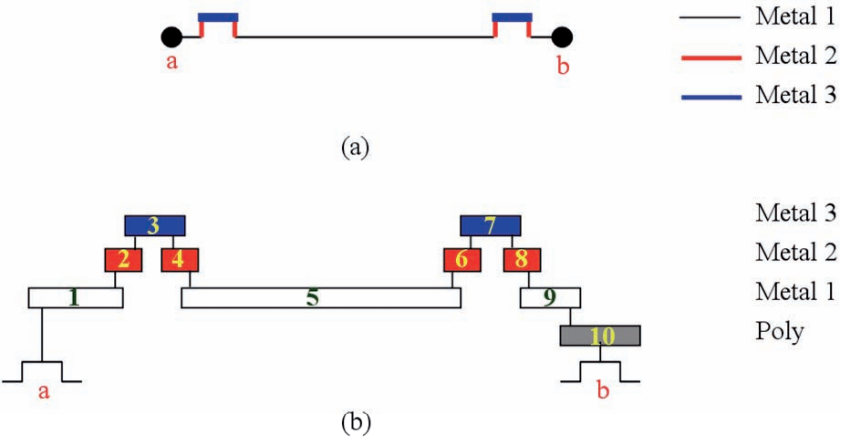

Figure 4-3. (a) A 2-Pin Net with Jumper Insertion and (b) Its Cross Section.

3. MULTILEVEL ROUTING FRAMEWORK

Our multilevel routing algorithm is inspired by the work of Ho et al. [52]. As illustrated in Figure 4-4, G_0 corresponds to the routing graph of the level 0 of the multilevel coarsening stage. Before the coarsening process, we first perform the optimal jumper prediction for every net. After that, we can indicate which 2-pin nets are needed to insert jumpers by using minimum number of jumpers. Then, our congestion-driven global router first finds routing paths for the *local nets* (or *local 2-pin connections*) (those nets [connections] that entirely sit inside a tile) at each level. After the global routing is performed, we merge four adjacent tiles of G_0 into a larger tile and at the same time perform resource estimation for use at the next level (i.e., level 1 here). Coarsening continues until the number of tiles at a level, said the *k*th level, is below a given threshold. After coarsening, in order to break the cumulative length from the gates, we first break in two those segments of 2-pin nets that need jumpers, if the length of those segments exceeds the minimum allowable gate-strength. If they have not exceeded the minimum allowable gate-strength, then we try to assign the remaining segments to the highest layer. Segments assigned to the highest layer have the same utility as jumper. If the highest layer is too congested, we assign segments to the lower layer and fix them by adding jumpers near the gate input using a distance-aware maze router after the track assignment phase. After the layer assignment, a track assignment for fast routing completion and antenna

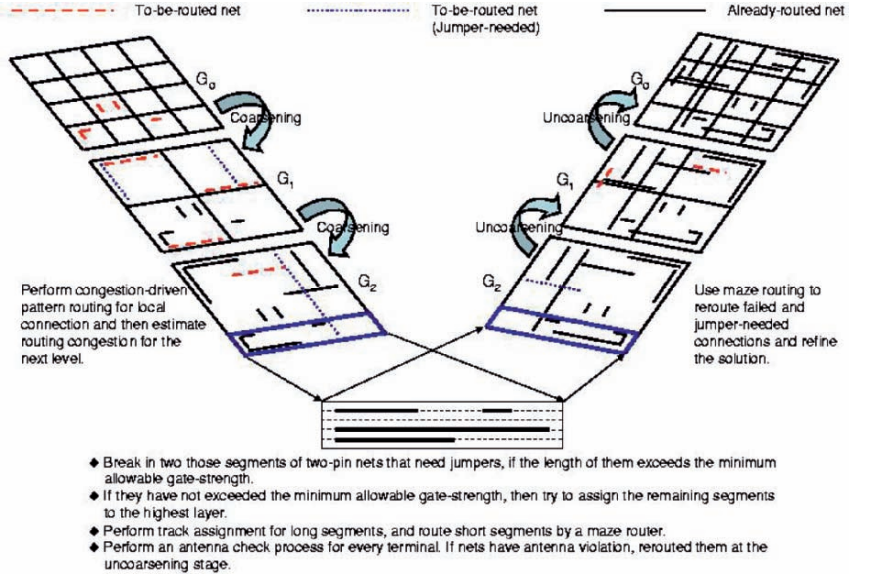

Figure 4-4. Multilevel Routing Framework Flow for Antenna Avoidance.

avoidance is performed to assign straight segments to underlying routing resources. After that, an antenna violation checking process is performed on every terminal. If nets have antenna violations, we identify them as failed nets that will be rerouted at the uncoarsening stage. During uncoarsening, the unroutable and antenna-violated nets are considered. Maze routing and rip-up and reroute are performed to refine the routing solution. Then we proceed to the next level (level k-1) of uncoarsening by dividing each tile to four finer tiles. The process continues up to level 0 when the final routing solution is obtained.

3.1 Bottom-Up Optimal Jumper Prediction

Given a netlist, we first use the minimum-radius minimum-cost spanning tree algorithm (MRMCST) discussed in Section 3.3.1 to construct a performance-driven routing tree for each multi-pin net. The MRMCST can minimize the critical path and preserve minimum total wire length at the same time. After that, we decompose each net into 2-pin connections, with each connection corresponding to an edge of the MRMCST. Then we decompose each net into 2-pin connections, with each connection corresponding to an edge of the minimum spanning tree. Each net to be connected is composed of a set of terminals, one of which is a driver and the others receivers. At the beginning of the interconnect fabrication process, the receiver-type terminals are in poly, and driver-type terminals are in diffusion. Any incomplete interconnect segments connected to one or more receivers forms the antenna. The risk of gate oxide damage is proportional to the amount of charges collected by the antenna and inversely proportional to the area of the gate oxide.

To avoid the antenna violation, we require that the total effective conductor connecting to a gate is less than or equal to a threshold, $A_{\max}$. The threshold can be the wire length limit, wire area limit, or any model of the strength of antenna effect caused by conductors. Typically, a net is modeled as a routing tree, where a node in the tree denotes a circuit terminal (a gate or a diffusion) and an edge denotes the interconnection between two circuit terminals. Since the interconnection connecting to a diffusion terminal will not cause any antenna violation, as explained in Section 4.2, we shall focus on those connecting to gate terminals. To minimize the antenna-violated terminals, we can break signal wires with antenna violation and route them to the highest levels by jumper insertion. This reduces the charge amount for violated nets during manufacturing. But each jumper needs at least two vias and will cause delay. In the 0.13 μm process technology, the short gate-strength results in a dramatic increase in the number of jumpers that need to

be added to the wire. Thus, it is very important to minimize antenna area and jumpers at the same time.

Let $T = (V, E)$ be a routing tree. The set V of nodes represents all circuit terminals, the set E of edges denotes the wires connecting the circuit terminals, and an upper bound on the allowable antenna area is A_{max}. We intend to add the minimum number of jumpers such that the accumulating interconnect length associated with each node is smaller than A_{max}. Let $L(v)$ denote the sum of interconnect lengths between the node v and all its neighbors. We formulate the problem of jumper insertion on a routing tree for antenna avoidance (JITA) as follows:

Given a routing tree $T = (V, E)$ with s as root and an upper bound A_{max}, find the minimum set J of jumper positions, $j \neq v$ for any $j \in J$ and $v \in V$, such that $L(v) \leq A_{max}$, $\forall v \in V$.

Note that the routing tree, which represents a net in any layout design stage in this formulation, can be a Steiner tree or a spanning tree (for example, a net to be globally routed, or a net after detailed routing in the post-layout stage). Therefore, the JITA problem is applicable to the antenna estimation in the global/detailed routing stage. The antenna violation will be fixed in the post-layout stage.

In this chapter, we present a bottom-up approach to solve the JITA problem. Given a net and a source, we first transform the tree topology by using the source as a root (see Figure 4-5). Then we compute the position of the jumper by accumulating interconnect length from each terminal in bottom-up fashion.

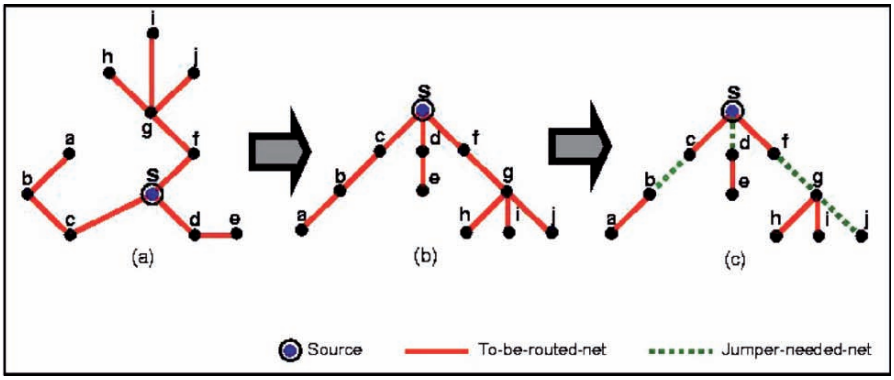

Figure 4-5. (a) A Net Topology, (b) The Transformed Tree Topology, and (c) The Transformed Tree Topology with Jumper-Needed Nets Marked.

To compute the jumper position, we have two possible scenarios. As shown in Figure 4-6, Line 4 determines whether the cumulative length $C(v)$

```
Algorithm : JumperPredict(T)
    Input : MST T and source s;
   Output : Jumper positions of T
begin
1   Pick unvisited v s.t. all descendants of v have been
    visited and let w as the precedent of v.
2   for all node v, set C(v) = 0 and visit(v) = 0
3   while w ≠ s do
4       if (C(v) < Amax)
5           if (C(v) + leng(v, w) > Amax)
6               JumperAdded(v, Up);
7               SecondJumperCheck(leng(v, w) − (Amax
                −C(v)))
8           else
9               AccumulateSegment();
10      else
11          InsHeap(v, w, leng(v, w));
12          UpJumper=0;
13          while HeapSize ≠ 0
14              AccLen=AccLen+PopHeap()→length;
15              if (AccLen>Amax)
16                  if (PopHeap()→node ≠ w)
17                      JumperAdded(v, Dn);
18                  else
19                      JumperAdded(v, Up);
20                      UpJumper= 1;
21              HeapSize−−;
22          if (UpJumper== 1)
23              SecondJumperCheck(leng(v, w) − 1);
24          else
25              AccumulateSegment();
26      visit(v)=1;
end
```

Figure 4-6. Algorithm for Jumper Prediction.

of descendants of the terminal v (e.g., $\sum_{i=1}^{m} d(e_i)$) in Figure 4-7(a), where $d(e_i)$ denotes the length between the node v and the root of the ith subtree and m denotes the number of subtrees which is less than the allowable antenna area ($A_{\max}$). For this case, there are two possible scenarios. First, if the sum of $C(v)$ and the length between v and its precedent w (i.e., $u(e)$) does not exceed the $A_{\max}$, we accumulate the total length and the number of gates to w for further computation. Second, if the sum of $C(v)$ and the length between v and its precedent w exceeds the $A_{\max}$, we add a jumper at the position near v (see Figure 4-7(b)). After that, if the remaining length connecting to w also exceeds the $A_{\max}$, we add a jumper at the position near w (Subroutine SecondJumperCheck). But if w is the source node, we do not need to add a jumper on that edge since the antenna damage can be discharged through w. Line 10 determines whether the cumulative length of descendants ($C(v)$) of the terminal v is greater than the allowable antenna area ($A_{\max}$). For this case, we first rank the length of edges adjacent to v in increasing order by using Heap. Then we accumulate the lengths of edges adjacent to v in increasing order. If the edge length plus the previous cumulative length starts to exceed the $A_{\max}$, we add a jumper at the edge near v (see Figure 4-7(c)). Line 22 considers whether we add a jumper at the edge (v, w), if so, we will do a SecondJumperCheck for that edge. The algorithm is summarized in Figure 4-6.

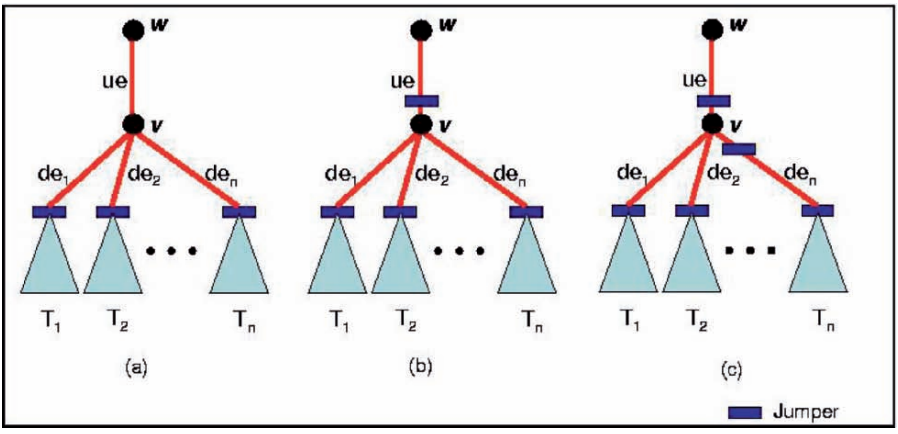

Figure 4-7. (a) The Initial Case Before the Jumper-Prediction Process, (b) The Case of $\sum_{i=1}^{m} d(e_i) < A_{\max}$, but $\sum_{i=1}^{m} d(e_i) + u(e) > A_{\max}$, Add a Jumper on $u(e)$ Then Do SecondJumperCheck, and (c) The Case of $\sum_{i=1}^{m} d(e_i) > A_{\max}$, Accumulate the Length of Edges Adjacent to v in Increasing Order. If the Cumulative Length Exceeds the A_{max}, Add a Jumper at the Edge Near v

Algorithm JumperPrediction is greedy in nature. To prove that Algorithm JumperPrediction finds the optimal jumper set (of the minimum size), therefore, we can show that the JITA problem exhibits *optimal substructure* and has the *greedy-choice property*.

Theorem 4.1: The JITA problem exhibits optimal substructure.

Proof Sketch: We prove this property by contradiction. Given a tree $T = (V, E)$, suppose that the jumper set J is the optimal solution of the tree. Every jumper in J decomposes the given tree into two subtrees. Let some jumper $j \in J$ decompose T into subtrees T_1 and T_2. Let the jumper set $J_1 \subseteq J$ ($J_2 \subseteq J$) be the set of jumpers in T_1 (T_2). Thus, $J = J_1 \bigcup J_2 \bigcup \{j\}$. If J_1 does not form an optimal solution on T_1, let J_1' be the optimal solution on T_1 and thus $|J_1'| < |J_1|$. Let $J' = J_1' \bigcup J_2 \bigcup \{j\}$. We have $|J'| = |J_1'|+|J_2|+1 < |J_1|+|J_2|+1 = |J|$. This contradicts the assumption of the optimality of set J. Therefore, the JITA problem has optimal substructure.

It is obvious that the JITA problem has the greedy-choice property, and Algorithm JumperPrediction finds the best solution in each step. We show that our approaches compute the optimal jumper sets for all the cases by using the greedy-choice property.

Given a net with n nodes, the best case time complexity of our jumper-prediction algorithm is $O(n)$ when the cumulative length of descendants for all nodes is less than the allowable antenna area (Lines 4–9). And the worst-case time complexity happens when the net topology is a star graph (all the nodes are connected to the source directly) and the cumulative length of descendants for a source is greater than the allowable antenna area (Lines 10–26). Since the complexity is determined by sorting, the worst-case time complexity is $O(n \lg n)$.

By this algorithm, we can predict which edges are in the best position to insert jumpers and then route these edges by distance-aware detailed routing at the coarsening stage.

3.2 Multilevel Routing with Antenna Avoidance

After the jumper positions are predicted, our multilevel framework starts by coarsening the finest tiles of level 0. At each level, tiles are processed one by one, and only local nets (connections) are routed. The global routing is based on the approach used in the Pattern Router [64]. It first routes local nets (connections) on the tiles of level 0. Let the multilevel routing graph of level i be $G_i = (V_i, E_i)$. Let $R_e = \{e \in E_i \mid e \text{ is the edge chosen for routing}\}$. We apply the cost function α: $E_i \rightarrow R_e$ to guide the routing:

$$\alpha(R_e) = \sum_{e \in R_e} c_e,$$

where c^e is the congestion of edge e and is defined by

$$c_e = 1/2^{(p_e - d_e)},$$

here p_e and d_e are the capacity and density associated with e, respectively.

Pattern routing uses an L-shaped or a Z-shaped route to make the connection, and generates the shortest path length between two points. Since the wire length is minimum, we do not include wire length in the cost function at this stage. We measure the routing congestion based on the commonly used channel density. After the detailed routing finishes routing a net, the channel density associated with an edge of a multilevel graph is updated accordingly.

Our global router first tries L-shaped pattern routing. If that routing fails, we try Z-shaped pattern routing. This can be considered as a simple version of rip-up and reroute. If both pattern routes fail, we give up routing the connection, so an overflow occurs. We refer to a *failed net* (*failed connection*) as one that causes an overflow. The failed nets (connections) will be reconsidered (refined) at the uncoarsening stage. There are at least two advantages to using this approach. First, routing resource estimation is more accurate than the use of global routing alone, since we can precisely evaluate the routing region. Second, we can obtain a good initial solution for the following refinement very effectively since pattern routing enjoys very low-time complexity and uses fewer routing resources due to its simple L-shaped and Z-shaped routing patterns.

After the global routing is completed, in order to break the cumulative length from the gates, we first break in two those segments of 2-pin nets that need jumpers, if the length of those segments exceeds the minimum allowable gate-strength. If they have not exceeded the minimum allowable gate-strength, then we try to assign the remaining segments to the highest layer. If the highest layer is too congested, we assign segments to the lower layer and fix them by adding jumpers near the gate input using distance-aware maze routing after the track assignment phase. Then, an intermediate step of track assignment between coarsening and uncoarsening stages is used for fast routing completion and antenna avoidance.

The track assigner works on a full row or column of the global cell array at a time. To simplify the track assignment problem, we only assign segments which span more than one complete global cell in a row or a column, and handle short segments during detailed routing.

Let T be the set of tracks inside a panel. Let S be the set of segments which need to be track assigned in this panel. Each track $t \in T$ can be

represented by its set of constituent contiguous intervals. Denoting these intervals by x_i, we have $t \equiv \bigcup x_i$, each of this x_i is either:

(1) A blocked interval, where no segment from S can be assigned
(2) An occupied interval, where a segment from S has been assigned
(3) A free interval, where no segment from the set S has yet been assigned

A segment $seg \in S$ is said to be assignable to $t \in T$, $t \equiv \bigcup x_i$, iff $x_i \bigcap seg \neq \phi$, which implies that either x_i is a free interval or is an interval occupied by a segment of the same net. Thus, the antenna-aware track assignment problem can be defined as follows:

Given a set of tracks T, a set of segments S, and a cost function $F: S \times T \rightarrow N$, which represents the cost of assigning a segment to a track, find an assignment of segments to the tracks that minimizes the sum of the costs (accumulated lengths).

In our implementation, we have considered the basic cost metrics such as the planar anchoring cost and the track and via obstruction cost defined in Batterywala et al. [11]. To better utilize the tracks in the panel, we will try to assign segments to the tracks in a bottom-up fashion. After these segments have been assigned, other segments are assigned by the well-known left-edge algorithm [45] for efficient track assignability. Furthermore, these segments for track assignment are long and may violate the antenna rules. If the segments need jumpers after the jumper-prediction phase, we just add jumpers at the two end sides, floating the segments so that they would not cause damage to gates. Thus, track assignment is a suitable stage to address antenna avoidance.

After the track assignment phase, the actual track position of a segment is known. Thus, we perform an antenna check for every terminal. Since the accumulated gate-strength is kept in every terminal, the antenna-check process can be performed in a short time. An accurate damage function which considers all plasma-based manufacturing operations is adopted for the antenna check. From all such operations (etching of conductor layer pattern, ashing, etching of via pattern, etc.), it is known that the transistor-gate damage increases in proportion to the antenna area and moreover the antenna perimeter length. The adoption of this damage function makes the calculation of damage accurate. If nets have an antenna violation, we regard them as the failed nets to be routed at the uncoarsening stage. The uncoarsening stage starts to refine each local failed net (connection), left from the

coarsening stage. The global router is now replaced by a maze router with the following cost function β: $E_i \rightarrow R$:

$$\beta(R_e) = \sum_{e \in R_e} (a * l_e + b * c_e + c * o_e),$$

where a, b, and c are user-defined parameters, l_e is the length of the net (connection), and $o_e \in \{0, 1\}$. If an overflow happens, o_e is set to 1; otherwise it is set to 0.

There is a trade-off among minimizing wire length, congestion, overflow, and jumper insertion. At the uncoarsening stage, we intend to resolve the overflow in a tile. Therefore, we let c be much larger than a or b. Also, we perform the detailed maze routing after the global maze routing. Iterative refinement of a failed net is stopped when a route is found or after several tries (say, three) have been made. Uncoarsening continues until the first level G_0 is reached and the final solution is found. This two-stage approach: global and local refinement of detailed routing outlines our overall refinement scheme.

4. EXPERIMENTAL RESULTS

We have implemented our multilevel system with antenna avoidance in C++ language on a 1 GHz SUN Blade 2000 workstation with 1GB memory. See Table 4-1 for the benchmark circuits. The design rules for wire/via widths and wire/via separation for detailed routing are the same as those used in Ho et al. [52].

Table 4-1 describes the set of benchmark circuits. In the table "Size" gives the layout dimensions "#Layers" denotes the number of routing layers

Table 4-1. The Benchmark Circuits Used in the Experiments of Chapter 4.

Circuits	Size (μm)	#Layer	#Nets	#Diffusions	#Gates
S5378	4330x2370	3	3124	1694	3040
S9234	4020x2230	3	2774	1486	2699
S13207	6590x3640	3	6995	3781	6781
S15850	7040x3880	3	8321	4472	8094
S38417	11430x6180	3	21035	11309	20901
S38584	12940x6710	3	28177	14753	27836

used "#Nets" represents the number of 2-pin connections after net decomposition "#Drivers" represents the number of driver-type terminals (diffusions), and "#Receivers" represents the number of receiver-type terminals.

Experimental results on wire length, vias, run-time, violated gates, and delay are listed in Table 4-2, where "D_{avg}" represents the average net delay. To perform experiments on timing-driven routing, we used the same resistance and capacitance parameters as those used in Ho et al. [52] for comparison. A via is modeled as the π-model circuit, with its resistance and capacitance being twice those of a wire segment. As mentioned in Leung [73], we set $A_{\max}$ to 100 μm in this experiment. Compared with Ho et al. [52], the experimental results show that our new multilevel router can reduce antenna-violated gates by about 100% and resulted in fewer increases in wire length, vias, run-time, and delay. It integrates: (1) a fast and optimal bottom-up jumper insertion approach and (2) an antenna fixer for fixing and/or rerouting nets that violate the antenna rule at the uncoarsening stage. Also, by using the routability-based multilevel routing framework, we can achieve 100% routing completion for all circuits, with very small overheads in wire length, delay, and run-time. The results show the effectiveness of our multilevel router in coping with the antenna effects.

Table 4-2. Results of Wire Length, the Number of Violated Gates, Run-Time, the Number of Jumpers, and Completion Rate Comparison.

Circuits	Results without antenna avoidance [41]					Our Results				
	Wirelength (μm)	#Vias	#Violated Gate	Time (s)	D_{avg} (ps)	Wirelength (μm)	#Vias	#Violated Gate	Time (s)	D_{avg} (ps)
S5378	8.4e7	7451	129	10.6	1258	8.4e7	7533	0	12.5	1271
S9234	6.0e7	6239	75	8.1	1009	6.1e7	6315	0	10.9	1015
S13207	2.3e8	16003	304	22.6	1243	2.4e8	16242	0	29.9	1281
S15850	2.9e8	19126	354	62.6	1253	3.0e8	19534	0	75.8	1279
S38417	8.0e8	49816	683	71.3	1146	8.2e8	50521	0	86.9	1171
S38584	1.1e9	65798	974	255.6	1151	1.2e9	67068	0	307.0	1194

5. SUMMARY

As technology advances into nanometer territory, the antenna effect problem has caused significant impact on routing tools. The antenna effect is a phenomenon of plasma-induced gate oxide degradation caused by charge accumulation on conductors. It directly influences reliability, manufacturability and yield of VLSI circuits, especially in deep-submicron technology using high-density plasma. Furthermore, the continuous increase of the

problem size of IC routing is also a great challenge to existing routing algorithms. In this chapter, we proposed a novel framework for multilevel full-chip routing with antenna avoidance using built-in jumper insertion approach. Compared with the state-of-the-art multilevel routing, the experimental results show that our approach reduced 100% antenna-violated gates and results in fewer wire length, vias, and delay increase.

This page intentionally blank

Chapter 5

MULTILEVEL FULL-CHIP ROUTING FOR THE X-BASED ARCHITECTURE

As technology advances into the nanometer territory, the interconnect delay has become a first-order factor on chip performance. To handle this effect, the X-architecture has been proposed for high-performance integrated circuits. The X-architecture presents a new way of orienting a chip's microscopic interconnect wires with the pervasive use of diagonal routes. It can reduce the wire length and via count, and thus improve performance and routability. Furthermore, the continuous increase of the problem size in IC routing is also a great challenge to existing routing algorithms. In this chapter, we present the first multilevel framework for full-chip routing using the X-architecture. To take full advantage of the X-architecture, we explore the optimal routing for three-terminal nets on the X-architecture and develop a general XST algorithm based on the Delaunay triangulation approach for the X-architecture [55]. The multilevel routing framework adopts a two-stage technique of coarsening followed by uncoarsening, with a trapezoid-shaped track assignment embedded between the two stages to assign long, straight diagonal segments for wire length reduction.

1. INTRODUCTION

As integrated circuit geometries keep shrinking, interconnect delay has become the dominant factor in determining circuit performance. To minimize interconnect delay, two key IC technologies have been introduced: (1) copper and low-*k* dielectrics have replaced aluminum (as of the 180 nm and 130 nm nodes), reducing both resistance and capacitance and (2) ICs have been adapted to a new interconnect architecture, called the *X-architecture*, to shorten interconnect length and thus circuit delay.

The traditional Manhattan architecture has its obvious advantages of easier design (placement, routing, etc.), but it adds significant and needless wire length over the Euclidean optimum. As reported in Stan et al. [94], the average Manhattan wire length is significantly longer than the average Euclidean distance. As shown in website [6], and in Paluszewski [84] and Tarjan [96], the X-architecture's pervasive uses of diagonal routing can reduce wire length and via count. In addition, the wire length and via count reduction make the routing problem easier to solve, resulting in faster timing closure. These benefits contribute toward an increased probability of first-silicon success.

The most prevailing consortium that advocates routing at 45-degree increments is the X-initiative [6]. While lithographic considerations can impinge on the use of arbitrary angles for wiring, the use of 45-degree wires is fully supported by nearly all current manufacturing technologies. Recently, Toshiba and Cadence have launched the industry's first commercial system-on-chip (SoC) devices built on the innovative X-architecture design (see Figure 5-1 [6]). Toshiba says that the TC90400XBG digital-media application processor is approximately 11% faster than comparable Manhattan-layout embedded chips in its product line. Thus, the X-architecture, the first production-worthy approach to the pervasive use of diagonal interconnect, shows promise to reduce the total interconnect while simultaneously improving the chip performance, power, and cost.

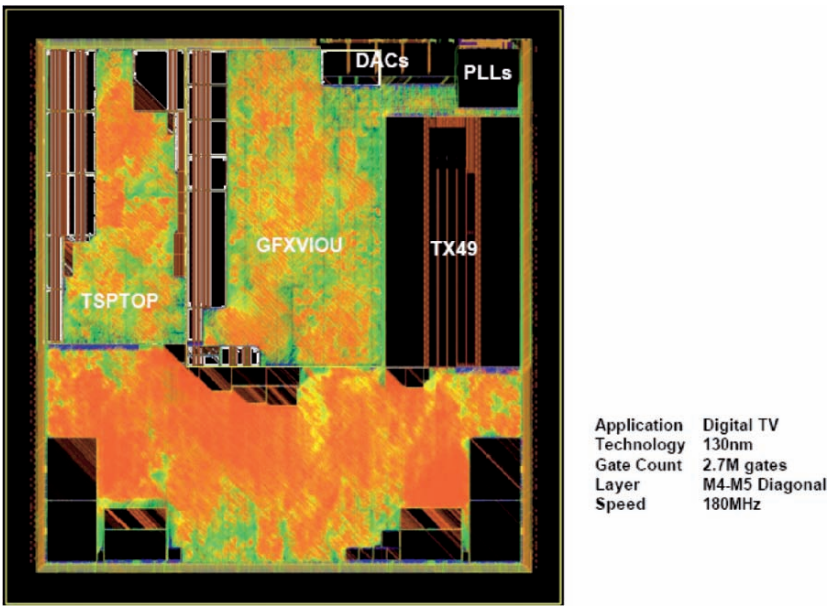

Figure 5-1. Layout of the TC90400XBG Digital-Media Application Processor [6].

Routing complexity is also an important problem for modern routers. To cope with the increasing complexity, researchers have proposed multilevel approaches to handle the problem [19, 31, 36, 52, 74]. The multilevel framework has attracted much attention in the literature recently [35]. It employs a two-stage technique: coarsening followed by uncoarsening. The coarsening stage iteratively groups a set of circuit components (e.g., circuit nodes, cells, modules, and routing tiles) based on a predefined cost metric, until the number of components being considered falls below a certain threshold. Then, the uncoarsening stage iteratively ungroups a set of previously clustered circuit components and refines the solution by using a combinatorial optimization technique (e.g., simulated annealing, local refinement). The multilevel framework has been successfully applied to partitioning, floorplanning, placement and routing in VLSI physical design.

Figure 5-2 shows our multilevel framework for the X-architecture.

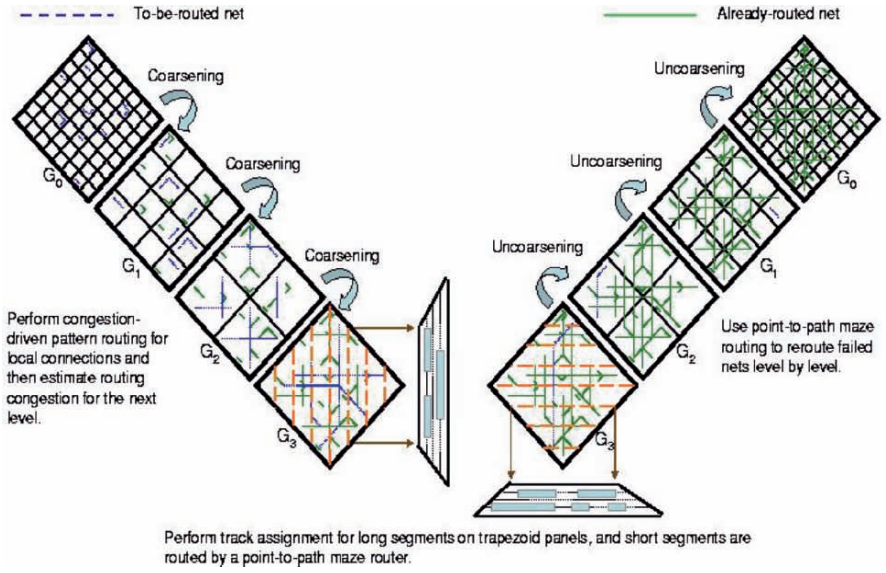

Figure 5-2. Multilevel Routing Framework Flow for the X-Based Architecture.

To take full advantage of the X-architecture, we explore the optimal routing for three-terminal nets on the X-architecture and develop a general XST algorithm based on the Delaunay triangulation approach for the X-architecture. Given a netlist, we first run the XST algorithm to construct the topology for each net. We then decompose each net into 2-pin connections, with each connection corresponding to an edge of the XST. Our multilevel framework starts from coarsening the finest tiles of the lowest level. At each level, pattern routing for the X-architecture is used for routability-driven

global routing. After the coarsening stage, we perform a trapezoid-shaped track assignment for diagonal segments. Most long, straight diagonal segments get track-assigned, and thus we can get lots of run-time improvement—the track assignment process not only takes less computation time than detailed maze routing but also only short segments (segments in lower levels) are delegated to the detailed router. In the uncoarsening stage, the unroutable nets are retried by point-to-path maze routing, rip-up and reroute to refine the routing solution level by level.

The rest of this chapter is organized as follows. Section 5.2 presents the routing model for the multilevel routing framework. Section 5.3 presents our novel X-architecture Steiner tree construction. Section 5.4 presents the routability-driven pattern routing. Section 5.5 presents the trapezoid-shaped track assignment. Experimental results are shown in Section 5.6, and a summary given in Section 5.7.

2. MULTILEVEL X-ROUTING FRAMEWORK

Routing in modern ICs is a very complex process, so we cannot easily obtain solutions directly. Our routing algorithm is based on a graph-search technique guided by the congestion information associated with routing regions and topologies, which assigns higher costs to nets passing through congested areas to balance the net distribution among routing regions. In this chapter, we consider the four-layer routing cases for experiments on the X-architecture. In these cases, the first two layers are routed in the preferred direction H and V. Layers 3 and 4 are routed at eight compass directions (which is called *liquid routing*) to reduce the number of vias [6, 97].

Before we can apply the graph-search technique to multilevel routing, we first need to model the routing resource as a graph whose topology can represent the chip structure. Figure 5-3 illustrates the graph modeling. For the modeling, we first partition a chip into an array of rectangular subregions, each of which may accommodate tens of routing tracks in each dimension. These subregions are usually called global cells (*GCs*). A node in the graph represents a *GC* in the chip, and an edge denotes the boundary between two adjacent *GCs*. Then we add some diagonal edges to connect each two diagonal adjacent nodes to obtain the multilevel routing graph G_0 for the X-architecture. Each edge is assigned a capacity according to the physical area or the number of tracks of a tile. A global router finds *GC*-to-*GC* paths for all nets on G_0 to guide the detailed router. The goal of global routing is to route as many nets as possible while meeting the capacity constraint of each edge and any other constraint, if specified.

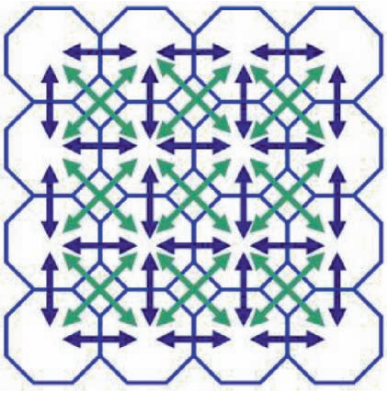

Figure 5-3. Routing Graph for the X-Architecture.

As illustrated in Figure 5-3, G_0 corresponds to the routing graph of the level 0 of the multilevel coarsening stage. At each level k, our global router just finds routing paths for the *local nets* (or *local 2-pin connections*) (those nets [connections] that entirely sit inside GC_{k+1}). After the global routing is performed, we merge four GC_k of G_k into a larger GC_{k+1} and at the same time perform resource estimation for use at level $k + 1$. Coarsening continues until the number of *GCs* at a level, say the kth level, is below a threshold. After finishing coarsening, a trapezoid-shaped track assignment is performed to assign the longer, straight diagonal segments to underlying routing resources. The uncoarsening stage task is to refine the routing solution of the unassigned segments that belong to level k where both pins are located in GC_{k+1}. During uncoarsening, the unroutable nets are directed to perform by point-to-path maze routing or rip-up and re-route, to refine the routing solution. Then we proceed to the next level (level k-1) of uncoarsening by expanding each GC_k to four finer GC_{k-1}. The process continues until we go back to level 0 when the final routing solution is obtained.

Our multilevel routing algorithm is inspired by the work of Ho et al. [52]. In the coarsening stage, a fast congestion-driven pattern routing is used for global routing, level by level. After the coarsening stage, we perform a trapezoid-shaped track assignment for diagonal segments. Most longer, straight diagonal segments get track-assigned, and thus can get lots of run-time improvement—the track assignment process not only takes less computation time than detailed maze routing but also only short segments (segments in level 0 and level 1) are delegated to the detailed router. In the uncoarsening stage, the unroutable nets are retried by point-to-path maze routing, rip-up and reroute to refine the routing solution level by level.

3. X-ARCHITECTURE STEINER TREE CONSTRUCTION

The Steiner minimal tree problem has been proven to be NP-hard in Garey et al. [41] and Lee and Shen [69]. In recent years, people have paid more attention to the algorithms for λ-geometry Steiner minimal tree problem. Coulston [38] presented an exact algorithm for constructing exact octagonal Steiner minimal trees (OSMT). Kahng et al. [62] proposed a highly scalable algorithm for both rectilinear and octilinear Steiner trees. The most recent progress is Zhu's [105] octilinear Steiner tree construction based on spanning graphs. In this chapter, we propose an XST algorithm based on the Delaunay triangulation approach. By the optimal routing of each three-terminal net, we can extend the idea to construct our XST tree in $O(n \lg n)$ time.

3.1 Three-Terminal Net Routing Based on X-Architecture

Without loss of generality, given a two-terminal net $\delta = (1, 2)$, its optimal routing solution is one of the path around the parallelogram formed by δ (see Figure 5-4). Also, given any three-terminal net $\gamma = (1, 2, 3)$, if terminal 3 is located in the merged region of the two bounding boxes of the other two terminals (see Figure 5-5), the optimal routing solution is the OMST of γ inside this region.

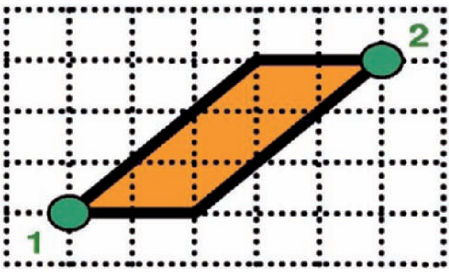

Figure 5-4. Optimal Routing of a Two-Terminal Net.

Lemma 5.1: *The optimal routing solution of a three-terminal net, of which one terminal is located in the merged region of the other two terminals, is the OMST of it.*

But if terminal 3 is not in the merged region, we can still find the optimal solution by connecting it to the nearest point of the previously formed

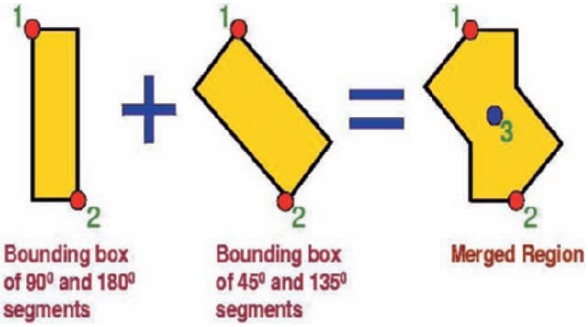

Figure 5-5. Merged Region of Two Bounding-Boxes.

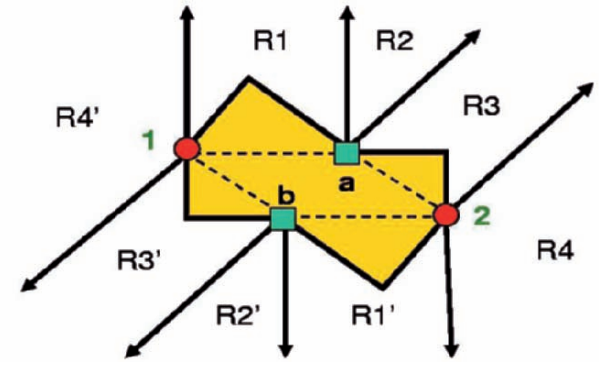

Figure 5-6. Octal Regions of a Two-Terminal Net.

two-terminal net. For example, given a two-terminal net $\delta = (1, 2)$, the plane can be divided into eight octal regions as shown in Figure 5-6.

If terminal 3 is located in R2 (R2′), we connect it to the Steiner point *a* (*b*), which is the apex of the parallelogram formed by terminals 1 and 2, to obtain the optimal routing solution (see Figure 5-6). If terminal 3 is located in R4 (R4′), we can take terminal 2 (1) as the internal terminal inside the merged region of terminal 1 (2) and 3, and the optimal routing solution can be obtained by Theorem 5.1. But if terminal 3 is located in R1 (R1′, R3, or R3′), we divide the region into R1a and R1b by the vertical line *D* passing through the point *V* (see Figure 5-7(a)). Without loss of generality, if terminal 3 is located in R1a, we can take terminal 1 as the third terminal to connect it to the Steiner point *S* of the two-terminal net formed by terminals 2 and 3 to obtain the optimal routing solution (see Figure 5-7(b)).

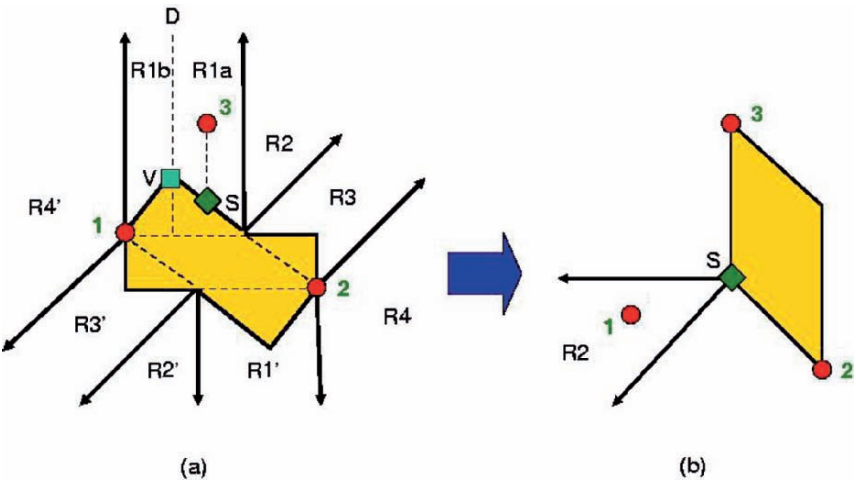

Figure 5-7. (a) If Terminal 3 is Located in Region R1a, the Optimal Steiner Point will be S and (b) Terminal 1 is in Region R2 Formed by Terminals 2 and 3.

If the connected edge is perpendicular to the two-terminal net, we will refine the solution to a shorter wire length. As shown in Figure 5-8, the refinement process will change the T-shaped portion (with length equal to $3\sqrt{2}$) of the net shown in the center part of Figure 5-8(a) to the L-shaped one (with length equal to 4) shown in Figure 5-8(b) to reduce the total wire length.

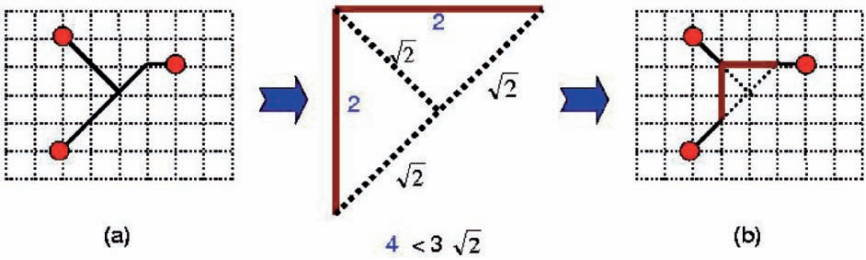

Figure 5-8. A Line is Perpendicular to Another One, and Refinement will Result in the Optimal Solution.

The algorithm for three-terminal net routing on the X-architecture, called *X3TR*, is summarized in Figure 5-9.

```
Algorithm : X3TR (3-Terminal Net Routing on X-Architecture)
    Input : A three-terminal net Γ = (α, β, γ);
   Output : The optimal routing tree T_o
begin
1  if (ThirdInsideMergedRegion(Γ)==True)
2      T_o = OMST(Γ);
3  else
4      OutsidePT = FindOutsidePT(Γ);
5      if (OctalRegion(OutsidePT)==R4 || R4')
6          T_o = OMST(Γ);
7      else if (OctalRegion(OutsidePT)==R2 || R2')
8          SteinerPT = ApexOfParallelogram(OutsidePT);
9          T_o = OMST(Γ, SteinerPT);
10     else if (OctalRegion(OutsidePT)==R1 || R1' || R3 || R3')
11         SteinerPT = VerticalOfParallelogram(OutsidePT);
12         T_o = OMST(Γ, SteinerPT);
end
```

Figure 5-9. Algorithm for Three-Terminal Net Routing Based on X-Architecture.

Since the numbers of terminals and wires being considered are both constant, we have the following theorem:

Theorem 5.1: *The X3TR algorithm finds the optimal routing of the minimum wire length for a three-terminal net on the X-architecture in constant time.*

3.2 X-Steiner Tree Algorithm by Delaunay Triangulation

Since the optimal routing solution for each three-terminal net can be found easily, we use the Delaunay triangulation approach [12] to divide all terminals into groups of three-terminal nets (see Figure 5-10(a)). After that, we compute the optimal wire length of all three-terminal nets, and sort them by their wire length (see Figure 5-10(b)). Further, we iteratively pick up a group of three-terminal nets with the minimal wire length, then route and merge them to the XST until it is constructed.

The time complexity for building the Delaunay triangulation is $O(n \lg n)$, where n is the number of terminals [12]. And computing the optimal wire length of all three-terminal nets and sorting also take $O(n \lg n)$ time. Thus, the total time complexity for the XST construction is $O(n \lg n)$ time.

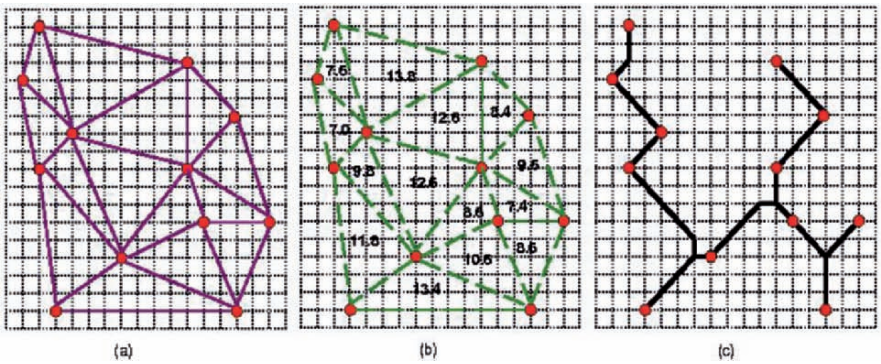

Figure 5-10. (a) Delaunay Triangulation of Terminals, (b) Optimal Wire Length of Each Triangle, and (c) XST.

Theorem 5.2: *The XST construction runs in* O(n *lg* n) *time, where* n *is the number of terminals.*

The algorithm for building an XST is stated in Figure 5-11, and given Figure 5-10(a) as input, the result of XST building is shown in Figure 5-10(c).

Algorithm : X-Architecture Steiner Tree
Input : Delaunay Triangulation DT of terminal set N
Output : X-Architecture Steiner Tree of DT

begin
1 For each terminal v in N
2 Set each v as a new subtree;
3 For each triangle T in DT
4 Compute the optimal wirelength of T;
5 Sort the wirelength of each T in DT in increasing order;
6 **while** (the number of subtree $>$ 1) **do**
7 Route the triangle with minimal wirelength;
8 Merge the three subtrees to a new subtree;
9 Refine the routing result if needed;
end

Figure 5-11. X-Architecture Steiner Tree Algorithm.

4. ROUTABILITY-DRIVEN PATTERN ROUTING

Given a netlist, we first run the XST algorithm to construct the topology for each net, and then decompose each net into 2-pin connections, with each connection corresponding to an edge of the XST. Our multilevel framework starts from coarsening the finest tiles of level 0. At each level, tiles are

processed one by one, and only local nets (connections) are routed. The global routing, which is based on the approach used in the pattern router [64] for the X-architecture, first routes local nets (connections) on the tiles of level 0. Figure 5-12 illustrates the pattern routing for the X-architecture. In a multilevel routing graph $G_i = (V_i, E_i)$, we define $R_e = \{e \in E_i \mid e$ is the edge chosen to be routed$\}$. Then the cost of R_e is defined as:

$$\cos t(R_e) = \sum_{e \in E} c_e,$$

where c_e is the congestion of edge e and is defined by

$$c_e = 1/2^{(p_e - d_e)},$$

here p_e and d_e are the capacity and density associated with e, respectively.

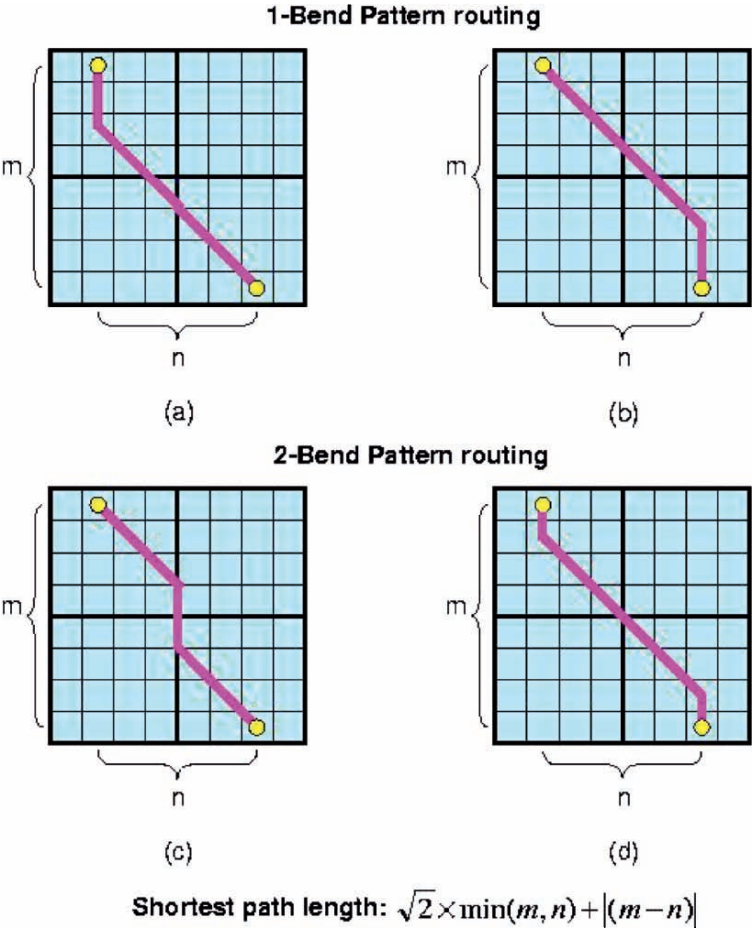

Figure 5-12. Routing Pattern for the X-Architecture.

Pattern routing uses a 1-bend or a 2-bend route to make the connection, and generates the shortest path length between two points. Since the wire length is minimum, we do not include wire length in the cost function at this stage. We measure the routing congestion based on the commonly used channel density. After the detailed routing finishes routing a net, the channel density associated with an edge of a multilevel graph is updated accordingly. Our global router first tries 1-bend pattern routing. If that routing fails, we try 2-bend pattern routing. This can be considered as a simple version of rip-up and reroute. If both pattern routes fail, we give up routing the connection, so an overflow occurs. We refer to a *failed net* (*failed connection*) as one that causes an overflow. The failed nets (connections) will be reconsidered (refined) at the uncoarsening stage. There are at least two advantages to using this approach. First, routing resource estimation is more accurate than the use of global routing alone, since we can precisely evaluate the routing region. Second, we can obtain a good initial solution for the following refinement very effectively since pattern routing enjoys very low-time complexity and uses fewer routing resources due to its simple 1-bend and 2-bend routing patterns.

5. TRAPEZOID-SHAPED TRACK ASSIGNMENT

Due to the lithography issues of nanometer technology, the high via count is more likely to reduce yield. Reducing the number of vias is one of the key challenges for today's routers. In Figure 5-13, we show the difference of a 2-pin connection between the Manhattan and the X-architecture. In this example, the total wire length of the X-architecture is less than that of the Manhattan architecture $(1+2\sqrt{2} \approx 3.828 < 5)$. But the via count of the X-architecture is

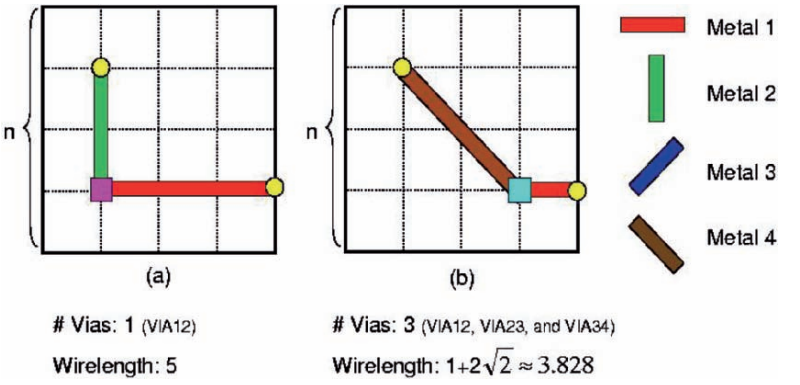

Figure 5-13. Difference Between Manhattan and X-Architectures.

larger than that of the Manhattan architecture (3 > 1). Therefore, if the wire length of the 2-pin net is short, the delay caused by via increase may offset the gains in the reduction of wire length [97].

To overcome the drawback of via increase and fully utilize the benefit of wire length reduction of the X-architecture, we assign only the long diagonal segments to tracks for better delay reduction.

In the gridded environment, each grid is λ apart from its immediate neighbors, where λ is the minimum spacing requirement dictated by the physical design rules. For the Manhattan architecture, this constitutes a perfect environment because there is at least λ distance between every grid point. But for the X-architecture, this commonly used grid-based model has a drawback: as shown in Figure 5-14(a), if a grid point has a 45(135)-degree wire passing through, topological design rules of the minimum spacing requirement dictates that the adjacent grid points cannot be used for routing ($\sqrt{2}\lambda/2 < \lambda$).

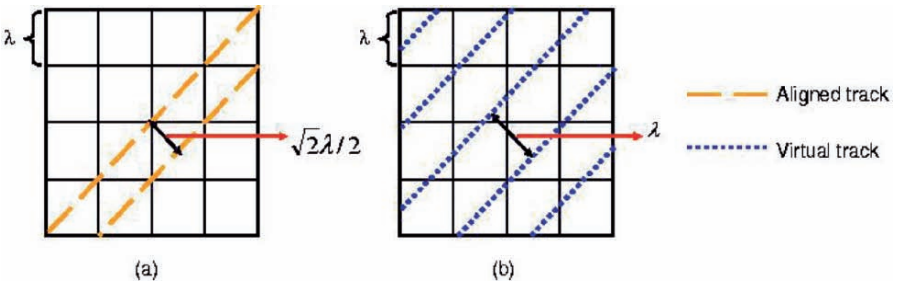

Figure 5-14. Virtual Track to Meet the Minimum Spacing Rule (λ).

To overcome this drawback, we shift the aligned tracks to the *virtual tracks* for meeting the design rules (see Figure 5-14(b)). Although the virtual tracks are not aligned on the grids, we can use short wrong-way jogs, which are used on the non-preferred direction routing layer and thus include no vias, to connect the end points to the nearest grid. The connection strategy is shown in Figure 5-15.

In this chapter, we propose a fast track assignment heuristic for long diagonal segments. After the coarsening stage, we get several long diagonal segments. To simplify the track assignment problem, we track-assign only segments which span more than one complete diamond-shaped global cell and delegate short segments to the detailed router. The track assigner works on a trapezoid-shaped row or column of the diamond-shaped global cell array one at a time (see Figure 5-16). Each trapezoid-shaped row (column) is called a *trapezoid panel*.

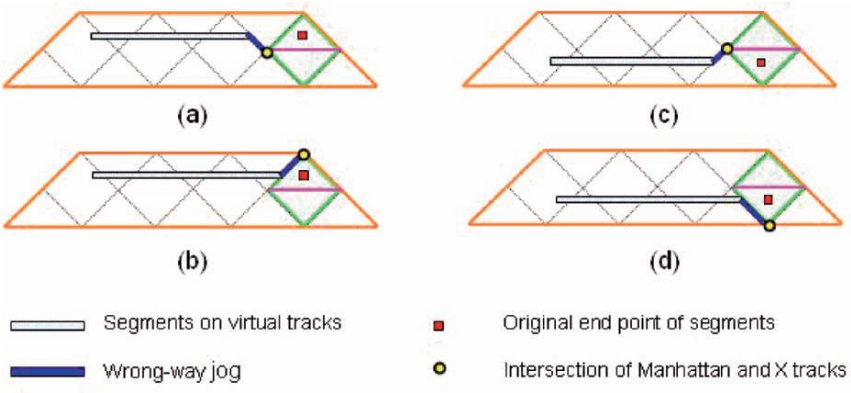

Figure 5-15. Example of Wrong-Way Jogs Connection.

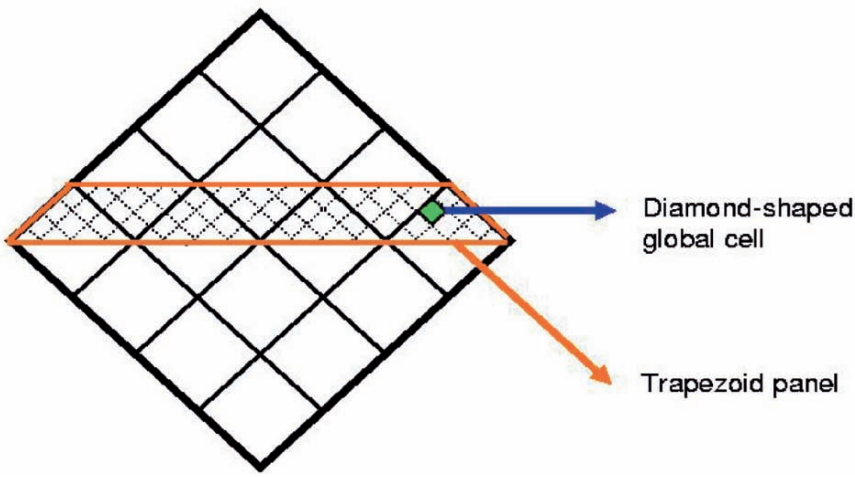

Figure 5-16. Trapezoid Panel.

Let T be the set of tracks inside a trapezoid panel. Let S be the set of segments which need to be track assigned in this trapezoid panel. Each track $t \in T$ can be represented by its set of constituent contiguous intervals. Denoting these intervals by x_i, we have $t \equiv \bigcup x_i$, each of this x_i is either:

(1) A blocked interval, where no segment from S can be assigned
(2) An occupied interval, where a segment from S has been assigned
(3) A free interval, where no segment from the set S has yet been assigned

A segment $seg \in S$ is called a left (right) segment, if the left- (right-) end terminal is in the left (right) zone. If a segment is said to be assignable to

$t \in T$, $t \equiv \bigcup x_i$, iff $x_i \bigcap seg \neq \phi$, which implies that either x_i is a free interval or is an interval occupied by a segment of the same net. Thus, a trapezoid-shaped track assignment problem can be defined as follows:

Given a set of tracks T, a set of segments S, and a cost function $F: S \times T \rightarrow N$, which represents the cost of assigning a segment to a track, find an assignment of segments to the tracks that minimizes the sum of the costs.

In our implementation, we have considered the basic cost metrics such as the planar anchoring cost and the track and via obstruction cost defined in Batterywala et al. [11]. To better utilize the tracks in the panel, we will try to assign the left and right segments to the tracks in the bottom-up fashion. After these segments have been assigned, other segments are assigned by the well-known left-edge algorithm [45] for efficient track assignability. To take the advantage of the X-architecture, we only assign those long and diagonal segments for track assignment. The longer L-shaped segment is routed in diagonal direction, the greater the wire length is reduced. Thus, track assignment is a suitable stage to reduce the wire length of the X-architecture.

An example is shown in Figure 5-17, and the solution is shown in Figure 5-18. After the track assignment phase, we use the short wrong-way jogs, which include no vias, to connect the two-end terminals to their nearest grid point. After that, we can perform point-to-path maze routing to complete both end points, which span at most two global cells.

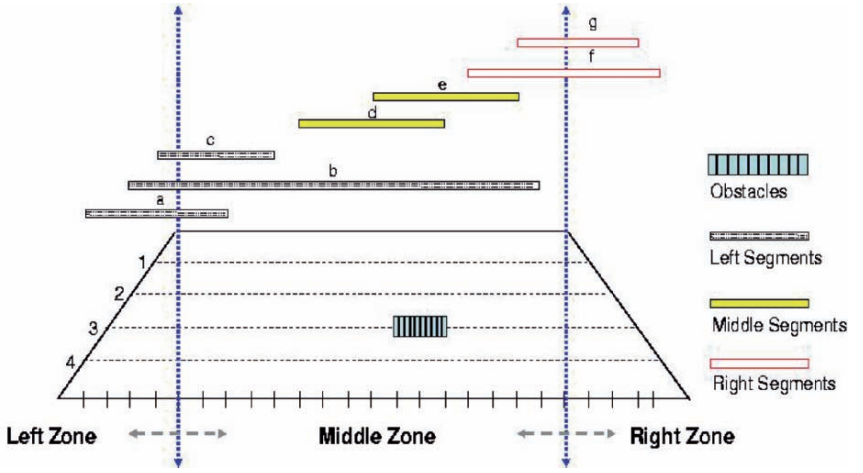

Figure 5-17. Example of a Trapezoid-Shaped Track Assignment Problem.

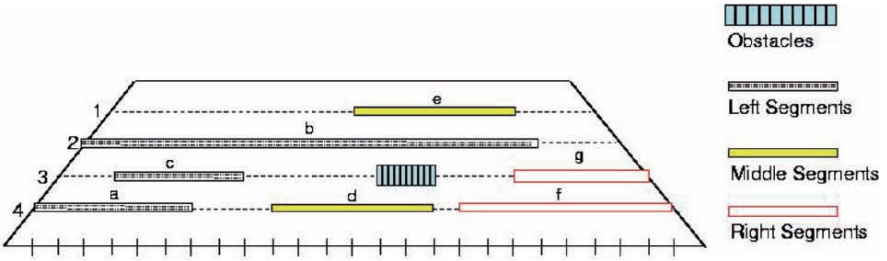

Figure 5-18. Solution to the Trapezoid-Shaped Track Assignment Problem Given in Figure 5-17.

6. EXPERIMENTAL RESULTS

We implemented our multilevel X-routing system in the C++ language on a 1 GHz SUN Blade 2000 workstation with 1GB memory. We compared our results with Ho et al. [52] based on the six benchmark circuits provided by the authors (see Table 5-1 for the benchmark circuits). In Table 5-1, "Circuits" denotes the names of the circuits, "Size" gives the layout dimensions, "#Layers" denotes the number of routing layers used, and "#Nets" represents the number of 2-pin connections after net decomposition.

Table 5-1. Benchmark Circuits Used in the Experiments of Chapter 5.

Circuits	Size (μm)	#Layers	#Nets	#Pins
S5378	4330x2370	4	3124	4734
S9234	4020x2230	4	2774	4185
S13207	6590x3640	4	6995	10562
S15850	7040x3880	4	8321	12566
S38417	11430x6180	4	21035	32210
S38584	12940x6710	4	28177	42589

Experimental results on wire length, the number of vias, the routing completion rate, and average net delay are listed in Table 5-2, where "D_{avg}" represents the average net delay. To perform experiments on timing-driven routing, we used the same resistance and capacitance parameters as those used in Ho et al. [52] for comparison. A via is modeled as the π-model circuit, with its resistance and capacitance being twice those of a wire segment. Compared with Garey et al. [41], the experimental results show that our multilevel X router reduced the wire length and average delay by about 18.7% and 8.8% with similar routability and number of vias in shorter

running time. The improvement of via count is not as we expected, because the work of Ho et al. [52] uses four layers for track assignment which reduces lots of vias, and our multilevel X router just uses top-two layers instead.

Table 5-2. Results of Wire Length, Via Counts, Completion Rate, Delay, and Run-Time Comparison.

Circuits	Results of [41]					Our Results				
	Wirelength (μm)	#Vias	Cmp. Rates	D_{avg} (ps)	Time (s)	Wirelength (μm)	#Vias	Cmp. Rates	D_{avg} (ps)	Time (s)
S5378	8.4e7	7451	99.8%	1258	10.6	7.2e7	7432	99.8%	1119	10.5
S9234	6.0e7	6239	99.9%	1009	8.1	5.5e7	6323	99.8%	956	7.9
S13207	2.3e8	16003	99.8%	1243	22.6	1.8e8	15897	99.8%	1074	20.9
S15850	2.9e8	19126	99.7%	1253	62.6	2.2e8	18999	99.4%	1178	54.1
S38417	8.0e8	49816	99.8%	1146	71.3	6.1e8	49131	99.0%	1034	59.8
S38584	1.1e9	65798	99.8%	1151	255.6	8.8e8	65018	99.1%	1068	198.1
Avg.	1.19	1.00	1.00	1.09	1.13	1	1	1	1	1

It should be noted that the 18.7% improvement in wire length is significantly better than the 11% improvement obtained by Toshiba's tool for routing its TC90400XBG digital-media application processor, as mentioned earlier. The difference also implicitly reveals the effectiveness of our multilevel routing.

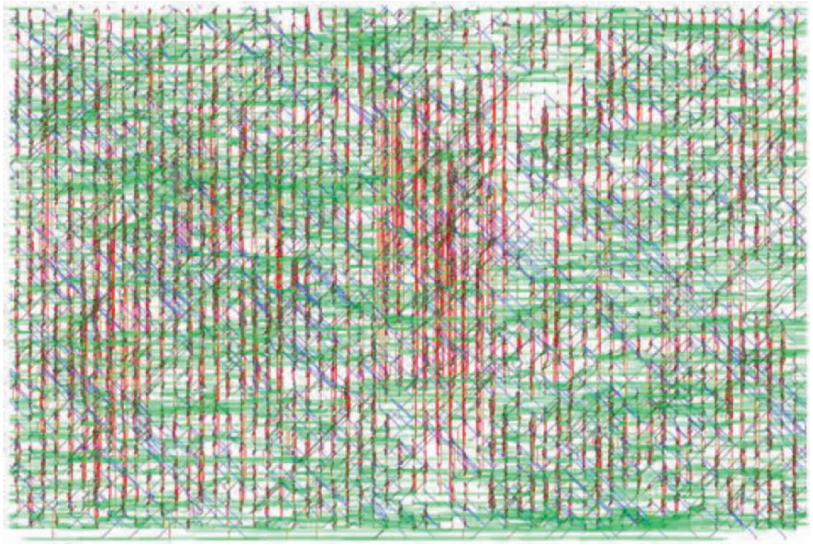

Figure 5-19. Routing Solution for Benchmark "s38417."

The experimental results also reveal the effectiveness of the intermediate stage of track assignments, because most longer, straight diagonal segments that get track-assigned can take full advantage of the X-architecture. Furthermore, the XST algorithm can also reduce the wire length and average net delay. Thus, the results show that our new multilevel X-routing framework is capable of handling the X-architecture for interconnect optimization. Figure 5-19 shows the routing solution for benchmark "s38417" obtained from the X router.

7. SUMMARY

As technology advances into the nanometer territory, the interconnect delay has become a first-order effect on chip performance. To handle this effect, the X-architecture has been proposed for high-performance integrated circuits. The X-architecture presents a new way of orienting a chip's microscopic interconnect wires with the pervasive use of diagonal routes. It can reduce the wire length and via count, and thus improve performance and routability. Furthermore, the continuous increase of the problem size in IC routing is also a great challenge to existing routing algorithms. In this chapter, we presented the first multilevel framework for full-chip routing using the X-architecture. To take full advantage of the X-architecture, we explored the optimal routing for three-terminal nets on the X-architecture and develop a general XST algorithm based on the Delaunay triangulation approach for the X-architecture. The multilevel routing framework adopts a two-stage technique of coarsening followed by uncoarsening, with a trapezoid-shaped track assignment embedded between the two stages to assign long, straight diagonal segments for wire length reduction. Compared with the state-of-the-art multilevel routing for the Manhattan architecture, experimental results show that our approach reduced wire length by 18.7% and average delay by 8.8% with similar routing completion rates and via counts.

Chapter 6

CONCLUDING REMARKS AND FUTURE WORK

In this chapter, we give concluding remarks and discuss some future research directions. We have presented a novel multilevel full-chip router, namely *m*SIGMA for SIGnal-integrity and MAnufacturability optimization in this book. To handle both the signal-integrity and manufacturability problem, we incorporate an intermediate stage of layer/track assignment into the multilevel routing framework. This intermediate stage not only speeds up the runtime, but also provides a suitable stage for doing optimization. Furthermore, we have adopted this novel multilevel routing framework to consider crosstalk, performance, and antenna effect problems, and the experimental results have shown that our approach is efficient and effective. In order to take advantage of the X-architecture, we also adapt our multilevel routing framework on the X-based architecture. The experimental results show that our approach is capable of handling diagonal segments well.

In the following section, we give some concluding remarks on the study and future research directions.

1. MULTILEVEL ROUTING FRAMEWORK

We have introduced the concept of the multilevel routing framework in Chapter 1. Traditionally, many routing algorithms adopt a flat framework of finding paths for all nets. Those algorithms can be classified into sequential and concurrent approaches. But the major problem of the flat frameworks lies in their weak scalability for handling larger designs.

As technology advances, technology nodes are getting smaller and circuit sizes are getting larger. To cope with the increasing complexity, researchers proposed hierarchical approaches to handle the problem. This top-down approach

has a global view of the routing problem, but a main drawback is that hierarchical approaches lack local information; therefore, it could induce local congestion.

The multilevel framework has attracted much attention in the literature recently. It employs a two-stage technique: coarsening followed by uncoarsening. The coarsening stage iteratively groups a set of circuit components (e.g., circuit nodes, cells, modules, routing tiles) based on a predefined cost metric until the number of components being considered is smaller than a given threshold. Then, the uncoarsening stage iteratively ungroups a set of previously clustered circuit components and refines the solution by using a combinatorial optimization technique (e.g., simulated annealing, local refinement). The multilevel framework has been successfully applied in VLSI physical design.

2. ROUTING CHALLENGES FOR NANOMETER TECHNOLOGY

In Chapter 2, we have presented the routing challenges for nanometer technology. As IC process geometries were scaled down to the nanometer territory, the industry faces severe challenges of signal-integrity and manufacturing limitations. To meet the signal timing problems, it is dispensable to address the signal-integrity issues in routing stage. To guarantee yield and reliability, routing for manufacturability have played a pivotal role in resolution and thus yield enhancement for the imperfect manufacturing process. In this chapter, we also introduced major challenges arising from nanometer process technology and key existing techniques for handling the challenges in routing problems for nanometer technology.

3. MULTILEVEL FULL-CHIP ROUTING CONSIDERING CROSSTALK AND PERFORMANCE

In Chapter 3, we have presented a novel framework for fast multilevel routing considering crosstalk and performance optimization. To handle the crosstalk minimization problem, we incorporate an intermediate stage of layer/track assignment into the multilevel routing framework. For performance-driven routing, we use a minimum-radius minimum-cost spanning-tree (MRMCST) heuristic for global routing. Compared with the state-of-the-art multilevel routing with the routability mode, the experimental results show

that our router achieved a 6.7X run-time speedup, reduced the respective maximum and average crosstalk (coupling length) by about 30% and 24%, reduced the respective maximum and average delay by about 15% and 5%. And compared with the timing-driven mode, the experimental results show that our router still achieved a 5.9X run-time speedup, reduced the respective maximum and average crosstalk by about 35% and 23%, reduced the respective maximum and average delay by about 7% and 10% in comparable routability, and resulted in fewer failed nets.

4. MULTILEVEL FULL-CHIP ROUTING CONSIDERING ANTENNA EFFECT AVOIDANCE

In Chapter 4, we have presented a novel framework for multilevel full-chip routing considering antenna effect avoidance using a built-in jumper insertion approach. The antenna effect is a phenomenon of plasma-induced gate oxide degradation caused by charge accumulation on conductors. It directly influences reliability, manufacturability and yield of VLSI circuits, especially in deep-submicron technology using high-density plasma. Furthermore, the continuous increase of the problem size in IC routing is also a great challenge to existing routing algorithms. Compared with the state-of-the-art multilevel routing, the experimental results show that our approach reduced 100% antenna-violated gates and resulted in fewer wire length, vias, and delay increase.

5. MULTILEVEL FULL-CHIP ROUTING FOR THE X-BASED ARCHITECTURE

In Chapter 5, we have presented the first multilevel framework for full-chip routing using the X-architecture. The X-architecture presents a new way of orienting a chip's microscopic interconnect wires with the pervasive use of diagonal routes. It can reduce the wire length and vias, and thus improve performance and routability. Furthermore, the continuous increase of the problem size in IC routing is also a great challenge to existing routing algorithms. To take full advantage of the X-architecture, we explore the optimal routing for three-terminal nets on the X-architecture and develop a general XST algorithm based on the Delaunay triangulation approach for the X-architecture. The multilevel routing framework adopts a two-stage technique of coarsening followed by uncoarsening, with a trapezoid-shaped track assignment embedded between the two stages to assign long, straight diagonal

segments for wire length reduction. Compared with the state-of-the-art multilevel routing for the Manhattan architecture, experimental results show that our approach reduced wire length by 18.7% and average delay by 8.8% with similar routing completion rate and vias.

6. FUTURE RESEARCH DIRECTIONS

In this book, we have presented a novel framework for fast multilevel routing considering signal-integrity and manufacturability optimization. The experimental results have shown that our algorithm is very efficient and effective. Some future research directions are shown below.

RET-Aware Routing: Although RET-aware routing has been studied to some degree, there is still much room for research in this topic. It is desired to develop a technique that can accurately predict RET behaviors and facilitate the manufacturing process, applying OPC, PSM, or more than one RET at the same time. Further, since the simulation for RET behaviors is usually very time-consuming, efficiency should always be considered.

RDR-Aware Routing: The concept of restricted design rules (RDRs) appears to enhance manufacturability by restricting the layouts that designers produce. Using only regular features in a layout will improve lithographic printability and make RETs easier to implement. However, because over-restricted rules may decrease the performance and increase the chip area, it is difficult to strictly follow these RDRs in practice. Therefore, an appropriate RDR-aware router should follow RDRs in lithographically critical regions but allow some exceptions of RDRs in noncritical regions to optimize the performance and the chip area. For this reason, an analyzer should also be provided to identify the critical regions.

CMP-Aware Routing: Aggressive technology scaling has led to much higher resistance and larger coupling capacitance on interconnect. According to ITRS roadmap [60], copper dishing/erosion after CMP and scattering effect may increase resistance significantly [28]. Also, coupling capacitance between wires becomes dominant over ground and fringing capacitance at 0.18 μm technology (over 60% of the total capacitance), and increases rapidly with higher wire aspect ratio of the advanced technologies. Therefore, interconnect delay will suffer from the increased resistance and coupling capacitance more seriously in the future.

Meanwhile, manufacturability and yield are becoming ever-serious concerns at 90 nm node and below, and are shown to be heavily affected by design patterns. Especially, topography (thickness) variation after CMP is

shown to be systematically determined by wire-density distribution. Even after CMP, intrachip topography variation can still be on the order of 20–40%. Such topography variation leads to not only significant performance degradation due to increased wire resistance and capacitances, but also acute manufacturing issues like etching and printability.

The main reason for the above problems is wire density distribution. Higher wire density usually leads to copper thickness reduction due to erosion after CMP, making resistance worse. Also, the reduced copper thickness after CMP can worsen the scattering effect, further increasing resistance. Meanwhile, a region with higher wire density tends to have less spacing between wires, thus significantly increasing coupling capacitance between them. For a lower wire density region, dummy fill is performed before CMP to make density distribution more uniform, expecting less topography variation after CMP. However, dummy fill is usually applied in the post-design stage, and has limitations due to strict rules (such as fill size, pattern, spacing to other features) which intend to minimize the disturbance on the existing design. Thus, nonuniform distribution of density still exists even after the dummy fill, creating topography variation which wastes the minuscule depth of focus.

Timing-Aware Double-Via Insertion: According to our empirical study and industry experience, it has been observed that the addition of redundant vias could change the timing behavior of a design positively and negatively. In other words, some path delays may increase while some others may decrease. Although Luo et al. [76] has considered the timing issue, they cannot guarantee that the resulting design can still satisfy timing constraints after applying their approach. Therefore, how to tackle the timing issue more accurately during double-via insertion is still worthy for further study.

This page intentionally blank

References

[1] http://www.cadence.com/

[2] http://www.intel.com/

[3] http://www.magma-da.com

[4] http://www.mentor.com

[5] http://www.synopsys.com

[6] http://www.xinitiative.org/

[7] L. C. Abel, “On the ordering of connections for automatic wire routing,” *IEEE Transactions on Computers*, vol. C-2, no. 11, pp. 1227–1233, Nov. 1972.

[8] S. B. Akers, “A modification of lee’s path connection algorithm,” *IEEE Transactions on Electronic Computers*, vol. EC-16, no. 2, pp. 97–98, Feb. 1967.

[9] C. Albrecht, “Global routing by new approximation algorithms for multi- commodity flow, ” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 5, pp. 622–632, May 2001.

[10] C. J. Alpert, J.-H. Huang, and A. B. Kahng, “Multilevel circuit partitioning,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 8, pp. 655–667, Aug. 1998.

[11] S. H. Batterywala, N. Shenoy, W. Nicholls, and H. Zhou, “Track assignment: A desirable intermediate step between global routing and detailed routing,” in *Proc. IEEE International Conference on Computer-Aided Design*, pp. 59–66, Nov. 2002.

[12] M. Berg, M. Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 2nd Edition, Springer-Verlag 2000.

[13] P. Berman, A. B. Kahng, D. Vidhani, H. Wang, and A. Zelikovsky, "Optimal phase conflict removal for layout of dark field alternating phase shifting masks," *IEEE Trans. Computer-Aided Design*, vol. 19, no. 2, pp. 175–187, Feb. 2000.

[14] M. Burstein and R. Pelavin, "Hierarchical wire routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 2, no. 4, pp. 223–234, Oct. 1983.

[15] K. Cao, P. Dhawan, and J. Hu, "Library cell layout with Alt-PSM compliance and composability," in *Proc. ACM/IEEE Asia Pacific Design Automation Conference*, pp. 216–219, Jan. 2005.

[16] K. Cao, S. Dobre, and J. Hu, "Standard cell characterization considering lithography induced variations," in *Proc. ACM/IEEE Design Automation Conference*, pp. 801–804, Jun. 2006.

[17] T. F. Chan, J. Cong, T. Kong, J. R. Shinnerl, "Multilevel optimization for large-scale circuit placement," in *Proc. IEEE International Conference on Computer-Aided Design*, pp. 171–176, Nov. 2000.

[18] Y.-W. Chang, K. Zhu and D. F. Wong, "Timing-driven routing for symmetrical-array-based FPGAs," *ACM Trans. on Design Automation of Electronic Systems*, vol. 5, no. 3, pp. 433–450, Jul. 2000.

[19] Y. W. Chang and S. P. Lin, "MR: A new framework for multilevel full-chip routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 5, pp. 793–800, May 2004.

[20] K. Chaudhary, A. Onozawa, and E. S. Kuh, "A spacing algorithm for performance and crosstalk reduction," in *Proc. IEEE International Conference on Computer-Aided Design*, pp. 697–702, Nov. 1993.

[21] P. H. Chen, S. Malkani, C.-M. Peng, and J. Lin, "Fixing antenna problem by dynamic diode dropping and jumper insertion," in *Proc. IEEE International Symposium on Quality Electronic Design*, pp. 275–282, Mar. 2000.

[22] H. Chen, C. K. Cheng, A. B. Khang, I. I. Mandoiu, Q. Wang, and B. Yao, "The Y- Architecture for on-chip interconnect: Analysis and methodology," in *Proc. IEEE International Conference on Computer-Aided Design*, pp. 13–19, Nov. 2003.

[23] H. Chen, B. Yao, F. Zhou, and C. K. Cheng, "The Y-Architecture: Yet another on-chip interconnect solution," in *Proc. ACM/IEEE Asia South Pacific Design Automation Conference*, pp. 840–846, Jan. 2003.

[24] H.-Y. Chen, M.-F. Chiang, Y.-W. Chang, L. Chen, and B. Han, "Novel full-chip gridless routing considering double-via insertion," in *Proc. ACM/IEEE Design Automation Conference*, pp. 755–760, Jun. 2006.

[25] T.-C. Chen and Y.-W. Chang, "Multilevel gridless routing considering optical proximity correction." in *Proc. IEEE/ACM Asia South Pacific Design Automation Conference*, pp. 1160–1163, Jan. 2005.

[26] Z. Chen and I. Koren, "Layer reassignment for antenna effect minimization in 3-layer channel routing," in *Proc. IEEE Workshop on Defect and Fault-Tolerance*, pp. 77–85, Nov. 1996.

[27] C. Chiang, A. B. Kahng, S. Shinha, and X. Xu, "Fast and efficient phase conflict detection and correction in standard-cell layouts," in *Proc. IEEE International Conference on Computer-Aided Design*, pp. 149–156, Nov. 2005.

[28] M. Cho, D. Z. Pan, H. Xiang, and R. Puri, "Wire density driven global routing for CMP variation and timing," in *Proc. IEEE International Conference on Computer-Aided Design*, pp. 487–492, Nov. 2006.

[29] B. Choi, C. Chiang, J. Kawa, and M. Sarrafzadeh, "Routing resources consumption on M-arch and X-arch," in *Proc. IEEE International Symposium on Circuits and Systems*, pp. V-73–V-76, May 2004.

[30] J. Cong, J. Fang and K. Khoo, "DUNE: A multi-layer gridless routing system with wire planning," in *Proc. ACM International Symposium on Physical Design*, pp. 12–18, Apr. 2000.

[31] J. Cong, J. Fang and Y. Zhang, "Multilevel approach to full-chip gridless routing," in *Proc. IEEE International Conference on Computer-Aided Design*, pp. 396–403, Nov. 2001.

[32] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong, "Provably good performance-driven global routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, no. 6, pp. 739–752, Jun. 1992.

[33] J. Cong, S. Lim, and C. Wu, "Performance driven multilevel and multiway partitioning with retiming," in *Proc. ACM/IEEE Design Automation Conference*, pp. 274–279, Jun. 2000.

[34] J. Cong and P. H. Madden, "Performance driven global routing for standard cell design," in *Proc. ACM International Symposium on Physical Design*, pp. 73–80, Apr. 1997.

[35] J. Cong and J. Shinnerl, *Multilevel optimization in VLSI CAD*, Kluwer Academic Publishers, 2003.

[36] J. Cong, M. Xie, and Y. Zhang, "An enhanced multilevel routing system," in *Proc. IEEE International Conference on Computer-Aided Design*, pp. 51–58, Nov. 2002.

[37] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, MIT Press, 2001.

[38] C. S. Coulston, “Constructing exact octagonal steiner minimal trees,” in *Proc. ACM Great Lake System on VLSI*, pp. 1–6, Apr. 2003.

[39] T. Deguchi, T. Koide and S. Wakabayashi, “Timing-driven hierarchical global routing with wire-sizing and buffer-insertion for VLSI with multi-routing-layer,” in *Proc. ACM/IEEE Asia South Pacific Design Automation Conference*, pp. 99–104, Jan. 2000.

[40] W. C. Elmore, “The transient response of damped linear networks with particular regard to wide band amplifiers,” *J. Appl. Phys.*, vol. 19, pp. 55–63, Jan. 1948.

[41] M. R. Garey, R. L. Graham, and D. S. Johnson, “The complexity of computing steiner minimal trees,” *SIAM Journal on Applied Mathematics*, pp. 835–859, Jun. 1977.

[42] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, 1980.

[43] L. Guibas and J. Stolfi, “Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams,” *ACM Transactions on Graphics*, vol. 4, no. 2, pp. 74–123, Apr. 1985.

[44] F. O. Hadlock, “A shortest path algorithm for grid graphs,” *Networks*, pp. 323–334, 1977.

[45] A. Hashimoto and J. Stevens, “Wire routing by optimizing channel assignment within large apertures,” in *Proc. ACM/IEEE Design Automation Conference*, pp.155–169, Jun. 1971.

[46] M. Hayashi and S. Tsukiyama, “A hybrid hierarchical global router for multi-layer VLSI’s,” *IEICE Trans. Fundamentals*, vol. E78-A, no. 3, pp. 337–344, Mar, 1995.

[47] J. M. Ho, C. H. Chang, D. T. Lee, and C. K. Wong, “Minimum diameter spanning trees and related problems,” *SIAM J. Computing*, vol. 20, no. 5, pp. 987–997, Oct. 1991.

[48] H. T. Heineken, J. Khare, W. Maly, P. K. Nag, C. Ouyang, and W. A. Pleskacz, “CAD at the design-manufacturing interface,” in *Proc. ACM/IEEE Design Automation Conference*, pp. 321–326, Jun. 1997.

[49] J. Heisterman and T. lengauer, “The efficient solutions of integer programs for hierarchical global routing,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 6, pp. 748–753, Jun. 1991.

[50] D. Hightower, “A solution to line routing problems on the continuous plane,” in *Proc. ACM/IEEE Design Automation Workshop*, pp. 1–24, Jun. 1969.

[51] T.-Y. Ho, Y.-W. Chang, and S.-J. Chen, “Crosstalk-driven multilevel full-chip routing,” in *Proc. the 14th VLSI Design/CAD Symposium*, pp. 25–28, Aug. 2003.

[52] T.-Y. Ho, Y.-W. Chang, S.-J. Chen, and D. T. Lee, "A fast crosstalk- and performance-driven multilevel routing system," in *Proc. IEEE International Conference on Computer-Aided Design*, pp. 382–387, Nov. 2003.

[53] T.-Y. Ho, Y.-W. Chang, and S.-J. Chen, "Multilevel routing with antenna avoidance," in *Proc. ACM International Symposium on Physical Design*, pp. 34–40, Apr. 2004.

[54] T.-Y. Ho, Y.-W. Chang, and S.-J. Chen, "Multilevel routing with jumper insertion for antenna avoidance," in *Proc. IEEE International SOC Conference*, pp. 63–66, Sep. 2004.

[55] T.-Y. Ho, C.-F. Chang, Y.-W. Chang, and S.-J. Chen, "Multilevel full-chip routing for the X-based architecture," in *Proc. ACM/IEEE Design Automation Conference*, pp. 597–602, Jun. 2005.

[56] T.-Y. Ho, Y.-W. Chang, S.-J. Chen, and D. T. Lee, "Crosstalk- and performance-driven multilevel full-chip routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 6, pp. 869-878, Jun. 2005.

[57] T. C. Hu and M.-T. Shing. A decomposition algorithm for circuit routing. In *VLSI Circuit Layout: Theory and Design*, pp. 144–152. IEEE Press, New York, NY, 1985.

[58] L.-D. Huang, X. Tang, H. Xiang, D. F. Wong, and I.-M. Liu, "A polynomial time optimal diode insertion/routing algorithm for fixing antenna problem," in *Proc. ACM/IEEE Design Automation and Test in Europe*, pp. 470–475, Feb. 2002.

[59] L.-D. Huang and M. D. F. Wong, "Optical proximity correction (OPC)-friendly maze routing," in *Proc. ACM/IEEE Design Automation Conference*, pp. 186–191, Jun. 2004.

[60] *The International Technology Roadmap for Semiconductors*, 2001.

[61] A. B. Kahng, "Research directions for coevolution of rules and routers," in *Proc. ACM International Symposium on Physical Design*, pp. 122–125, Apr. 2003.

[62] A. B. Kahng, I. Mandoiu, and A. Zelikovsky, "High scalable algorithms for rectilinear and octilinear steiner trees," in *Proc. ACM/IEEE Asia South Pacific Design Automation Conference*, pp. 827–833, Jan. 2003.

[63] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: Application in VLSI domain," *IEEE Trans. on VLSI Systems*, vol. 7, no. 1, pp. 69-79, Mar. 1999.

[64] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Pattern routing: use and theory for increasing predictability and avoiding coupling," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 7, pp. 777–790, Nov. 2002.

[65] R. Kay and R. A. Rutenbar, "Wire packing: A strong formulation of crosstalk-aware chip-level track/layer assignment with an efficient integer programming solution," in *Proc. ACM International Symposium on Physical Design*, pp. 61–68, Apr. 2000.

[66] C. K. Koh and P. H. Madden, "Manhattan or Non-Manhattan? A study of alternative VLSI routing architectures," in *Proc. ACM Great Lake System on VLSI*, pp. 47–52, Apr. 2000.

[67] S. Krishnan, S. Rangan, S. Hattangady, G. Xing, K. Brennan, M. Rodder, and S. Ashok, "Assessment of charge-induced damage to ultra-thin gate MOSFETs," in *Proc. IEEE International Electron Devices Meeting*, pp. 445–448, Dec. 1997.

[68] C. Y. Lee, "An algorithm for path connection and its application," *IRE Transactions on Electronic Computers*, vol. EC-10, no. 3, pp. 346–365, Sep. 1961.

[69] D. T. Lee and C. F. Shen, "The steiner minimal tree problem in the lambda-geometry plane," in *Proc. International Symposium on Algorithms and Computation*, pp. 247–255, Dec. 1996.

[70] K.-Y. Lee and T.-C. Wang, "Post-routing redundant via insertion for yield/reliability improvement," in *Proc. ACM/IEEE Asia Pacific Design Automation Conference*, pp. 303–308, Jan. 2006.

[71] K.-Y. Lee, T.-C.Wang, and K.-Y. Chao, "Post-routing redundant via insertion and line end extension with via density consideration," in *Proc. IEEE International Conference on Computer Aided Design*, pp. 633–640, Nov. 2006.

[72] S.-C. Lee, Y.-W. Chang, J.-M. Hsu, and H. Yang, "Multilevel large-scale module floorplanning/placement using B*-trees," in *Proc. ACM/IEEE Design Automation Conference*, pp. 812–817, Jun. 2003.

[73] H. K.-S. Leung, "Advanced routing in changing technology landscape," in *Proc. ACM International Symposium on Physical Design*, pp. 118–121, Apr. 2003.

[74] S.-P. Lin and Y.-W. Chang, "A novel framework for multilevel routing considering routability and performance," in *Proc. IEEE International Conference on Computer-Aided Design*, pp. 44–50, Nov. 2002.

[75] Y.-L. Lin, Y.-C. Hsu, and F.-S. Tsai, "Hybrid routing," IEEE *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 2, pp. 151–157, Feb. 1990.

[76] F. Luo, Y. Jia, and W.-M. Dai, "Yield-preferred via insertion based on novel geotopological technology," in *Proc. ACM/IEEE Asia Pacific Design Automation Conference*, pp. 730–735, Jan. 2006.

[77] W. Maly, C. Ouyang, S. Ghosh, and S. Maturi, "Detection of an antenna effect in VLSI designs," in *Proc. International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 86–94, Nov. 1996.

[78] M. Marek-Sadowska, "Global router for gate array," in *Proc. IEEE International Conference on Computer Design*, pp. 332–337, Oct. 1984.

[79] M. Marek-Sadowska, "Router planner for custom chip design," in *Proc. IEEE International Conference on Computer-Aided Design*, pp. 246–249, Nov. 1986.

[80] K. McCullen, "Phase correct routing for alternating phase shift masks," in *Proc. ACM/IEEE Design Automation Conference*, pp. 317–320, Jun. 2004.

[81] K. Mikami and K. Tabuchi, "A computer program for optimal routing of printed circuit connectors," in *Proc. of IFIP*, pp. 1475–1478, Nov. 1968.

[82] J. Mitra, P. Yu, and D. Z. Pan, "RADAR: RET-aware detailed routing using fast lithography simulations," in *Proc. ACM/IEEE Design Automation Conference*, pp. 369–372, Jun. 2005.

[83] R. H. J. M. Otten, R. Camposano, and P. R. Groeneveld, "Design automation for deep submicron: present and future," in *Proc. ACM/IEEE Design Automation and Test in Europe*, pp. 650–657, Feb. 2002.

[84] M..Paluszewski, P. Winter, and M. Zachariasen, "A new paradigm for general architecture routing," in *Proc. ACM Great Lake Symposium on VLSI*, pp. 202–207, Apr. 2004.

[85] D. M. Pawlowski, L. Deng, and M. D. F. Wong, "Fast and accurate OPC for standard cell layouts," in *Proc. ACM/IEEE Asia Pacific Design Automation Conference*, pp. 7–12, Jan. 2007.

[86] P. Raghavan and C. D. Thompson, "Randomized rounding: A technique for provably good algorithms and algorithmic proofs," *Combinatorica*, vol. 7, no. 4, pp. 365–374, Dec. 1987.

[87] S. M. Sait and H. Youssef, *VLSI Physical Design Automation: Theory and Practice*. World Scientific Publishers, Singapore, 1999.

[88] L. K. Scheffer, "Physical CAD changes to incorporate design for lithography and manufacturability," in *Proc. ACM/IEEE Asia South Pacific Design Automation Conference*, pp. 768–773, Jan. 2004.

[89] D. Y. Seo and D. T. Lee, "On the complexity of bicriteria spanning tree problems for a set of points in the plane," *PhD Dissertation,* Northwestern University, 1999.

[90] H. Shin, C.-C. King, and C. Hu, "Thin oxide damage by plasma etching and ashing process," in *Proc. IEEE International Reliability Physics Symposium*, pp. 37–41, Apr. 1992.

[91] H. Shirota, T. Sadakane, M. Terai, and K. Okazaki, "A new router for reducing "Antenna effect" in ASIC design," in *Proc. IEEE Custom Integrated Circuit Conference*, pp. 27.5.1–27.5.4, Sep. 1998.

[92] J. Soukup, "Fast maze router," in *Proc. ACM/IEEE Design Automation Conference*, pp. 100–102, Jun. 1978.

[93] M. Sriram, S. Kang, J. D. Cho, and S. Raje, and M. Sarrafzadeh, "Crosstalk-minimum layer assignment," in *Proc. IEEE Custom Integrated Circuit Conference*, pp. 29.7.1–29.7.4, May. 1993.

[94] M. R. Stan, F. Hamzaoglu, and D. Garrett, "Non-manhattan maze routing," in *Proc. Brazilian Symposium on Integrated Circuit Design*, pp. 260–265, Feb. 2004.

[95] D. Sylvester, C. Hu, O. S. Nakagawa, and S. Y. Oh, "Interconnect scaling: Signal integrity and performance in future high-speed CMOS designs," in *Proc. IEEE VLSI Symposium on Technology*, pp. 42−43, Jun. 1998.

[96] R. E. Tarjan, *Data structures and network algorithms*, Conference Board of the Mathematical Sciences, CBMS 44, Society for Industrial and Applied mathematics, 1983.

[97] S. Teig, "The X Architecture: not your father's diagonal wiring," in *Proc. IEEE System Level Interconnect Prediction*, pp. 33–37, Apr. 2002.

[98] K. P. Wang, M. Marek-Sadowska, and W. Maly, "Layout design for yield and reliability," in *Proc. ACM Physical Design Workshop*, pp. 190–197, Apr. 1996.

[99] D. Wang and E. Kuh, "A new timing-driven multilayer MCM/IC routing algorithm," in *Proc. MCM Conference*, pp. 89–94, Feb. 1997.

[100] H. Watanabe, J. Komori, K. Higashitani, M. Sekine, and H. Koyama , "A wafer level monitoring method for plasma-charging damage using antenna PMOSFET test structure," *IEEE Trans. on Semiconductor Manufacturing*, vol. 10, no. 2, pp. 228–232, May 1997.

[101] Y.-R. Wu, M.-C. Tsai, and T.-C. Wang, "Maze routing with OPC consideration," in *Proc. ACM/IEEE Asia Pacific Design Automation Conference*, pp. 198–203, Jan. 2005.

[102] G. Xu, L.-D. Huang, D. Z. Pan, and M. D. F. Wong, "Redundant-via enhanced maze routing for yield improvement," in *Proc. ACM/IEEE Asia Pacific Design Automation Conference*, pp. 1148–1151, Jan. 2005.

[103] H. Yao, Y. Cai, X. Hong, and Q. Zhou, "Improved multilevel routing with redundant via placement for yield and reliability," In *Proc. IEEE Great Lake Symposium on VLSI*, pp. 143–1146, Apr. 2005.

[104] H. Zhou and D. F. Wong, "Global routing with crosstalk constraints," in *Proc. ACM/IEEE Design Automation Conference*, pp. 374–377, Jun. 1998.

[105] Q. Zhu, H. Zhou, T. Jing, X. Hong, and Y. Yang, "Efficient octilinear steiner tree construction based on spanning graphs," in *Proc. ACM/IEEE Asia South Pacific Design Automation Conference*, pp. 687–690, Jan. 2004.