Copyright © 1997, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

POST ROUTING INTERCONNECT PERFORMANCE OPTIMIZATION

by

Tianxiong Xue

Memorandum No. UCB/ERL M97/60

15 August 1997

POST ROUTING INTERCONNECT PERFORMANCE OPTIMIZATION

Copyright © 1997

by

Tianxiong Xue

Memorandum No. UCB/ERL M97/60

15 August 1997

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Abstract

Post Routing Interconnect Performance Optimization

by

Tianxiong Xue

Doctor of Philosophy in Engineering-Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Ernest S. Kuh, Chair

Due to the recent advances in silicon technologies, interconnect delay and crosstalk noise have become important concerns in high performance circuit design. Both of these are routing-dependent, i.e., they are determined by the routes of interconnects on the chip. Therefore, it is appropriate to address these concerns at the post routing level when the interconnect performance can be measured accurately and the routing resource available for optimization is known. In this dissertation, we present post routing performance optimization methods which improve the interconnect delay, delay skew and crosstalk risk of the chip after a feasible routing solution is obtained.

In the first part of the thesis, the post routing interconnect optimization problem is investigated. The goal is to improve the chip performance by optimizing the performance of critical nets under routing resource constraints. Since our optimization process does not invalidate the current routing solution of the chip, the time-consuming iterative layout process is avoided. For optimization of a distributed RC line topology, we develop a multi-link insertion and wiresizing approach which improves the maximum delay and delay skew of the net simultaneously without any restriction on its routing topology. For optimization of a lossy transmission line topology, we design a sensitivity-based wiresizing algorithm which computes the maximum delay and its sensitivities with respect to wire widths analytically using high order moments. Experiments show that both approaches can achieve significant improvement in interconnect performance, and the delay estimation methods used are accurate for guiding interconnect optimization compared with SPICE.

In the second part of the thesis, the post global routing crosstalk risk estimation and reduction problem is discussed. The aim is to minimize the crosstalk risk at global routing level

so that a risk-free final routing solution can be obtained. The entire optimization process is region-based and consists of three key components: crosstalk risk estimation, bound partitioning and risk reduction. The crosstalk risk of each region, which indicates whether a risk-free global routing solution of the region is possible, is estimated based on the region's crosstalk risk graph. During bound partitioning, the risk bound of each sensitive net is adjusted appropriately among its routing regions for accurate risk estimation. For regions with high risks, net ripping up and rerouting is applied so that the maximum crosstalk risk of the chip can be reduced.

Professor Ernest S. Kuh
Dissertation Committee Chair

To my parents,

Muyong Xue and Xianhua Meng

Contents

List of Figures

List of Tables

Acknowledgments

I am deeply indebted to my research advisor Prof. Ernest Kuh for his guidance and support during the course of my graduate study at Berkeley. While allowing me considerable freedom to pursue research on my own, he taught me the skills to successfully formulate and attack a research problem, carry it to completion and present it appropriately in both oral and written form. The things I acquired as his student will be invaluable to me throughout my life.

I am grateful to Prof. Robert Brayton for his interest in my research and his constant encouragement. I also thank Prof. Dorit Hochbaum for serving in my Thesis Committee, she was also part of my Qualifying Committee, together with Prof. Paul Gray.

I enjoyed fruitful collaboration in conducting my research in Kuh's group. Prof. Xianlong Hong helped me in learning layout system development. I collaborated with Dr. Takashi Fujii on timing-driven global routing and Charles Hough on on-chip power noise reduction. Prof. Qinjian Yu's method inspired my work in Chapter Three and Prof. Dongsheng Wang assisted me in generating test tools and data for my approach in Chapter Four and Five. Prof. Chung-Kuan Cheng has also been a constant support of my research through the years.

During my time at Berkeley, I shared good friendship and collaboration with Premal Buch. The pleasant atmosphere we created in 550-A4 makes my life in Cory Hall much more enjoyable, and I will miss those after mid-night project efforts as well as those "in-depth" discussions on both academic and non-academic issues. It was also a great pleasure for me to interact with other past and current members of Kuh's group: Dr. Narasimha Bhat, Dr. Henrik Esbensen, Julie Hu, Dr. John Lillis, Dr. Junfa Mao, Dr. Minshine Shih and Janet Wang. I also enjoyed great working experience with Jagesh Sanghavi, Timothy Kam and Desmond Kirkpatrick in Berkeley's CAD group.

This work was accomplished while I was funded by SRC, I would also like to thank the administrative support from Tahani Sticpewich, Corey Schaffer and Carol Sitea.

A special acknowledgment goes to Jennifer Liang for all her love, understanding and support through my years at Berkeley.

Finally, I am most grateful to my parents, Muyong Xue and Xianhua Meng, who have nurtured me through the years and always supported me to realize my dreams. Although I couldn't be by their side as often as I wished, their love and encouragement at the other end of the line has always been a constant source of inspiration for me. To them, I dedicate this humble achievement as a token of gratitude.

Chapter 1

Introduction

1.1 Computer-Aided VLSI Design

The last two decades have witnessed rapid development in Integrated Circuit (IC) technologies. Today, one single Pentium micro-processor has over 3 million transistors and operates at a frequency of over 100 MHz. If Moore's Law which predicts that the power and complexity of a chip doubles every 18 months continues to hold in the future, IC circuits having over 20 million transistors and operating at 1 GHz will become reality by the year 2000. It is simply impossible to handle circuits of such complexity in detail even for the most experienced designers, and thus Computer-Aided Design (CAD) for Very Large Scale Integrated (VLSI) circuit which aims at managing complex designs in a timely manner has become an indispensable part of semiconductor industry.

For the tractability of the process, computer-aided VLSI design is typically divided into well-defined stages (Fig. 1.1):

- Behavioral Synthesis: As the first step in the design process, it quantitatively specifies the "behavior" of the system in the form of a high level language, and defines the functionalities of components needed to implement the design.

- Logic Synthesis: At this level, the logic structures representing the functionalities of circuit components are derived in terms of boolean expressions and optimized with such metrics as: chip area, timing, power, etc.

- Physical Design: During the layout process, the circuit is represented as a netlist and a set of geometric patterns which perform the intended functions of the corresponding components.

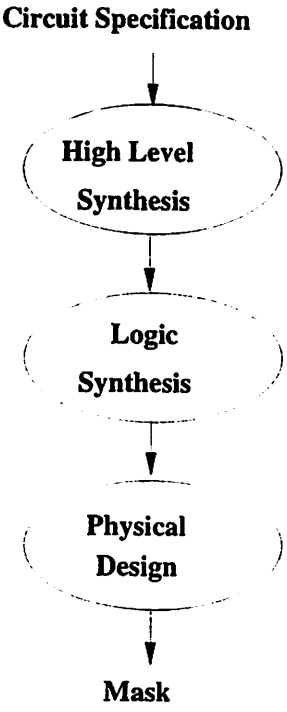

Figure 1.1: Overview of Computer-Aided VLSI Design

These geometric objects are placed and connected appropriately in order to generate a final mask for circuit fabrication.

The last stage in the design process, Physical Design, contains our area of interest and can be further divided into the following steps:

- **Partitioning** divides the circuit into a set of sub-components, each containing a number of functional blocks.
- **Floorplanning and Placement** determines the shape, size and location of each block on the chip.
- **Global Routing** generates the routes of interconnect wires connecting the blocks among the routing regions on the chip.
- **Detailed Routing** assigns the interconnects to routing tracks within each region.
- **Compaction** adjusts the spacing between interconnect wires such that certain metrics in design can be improved.

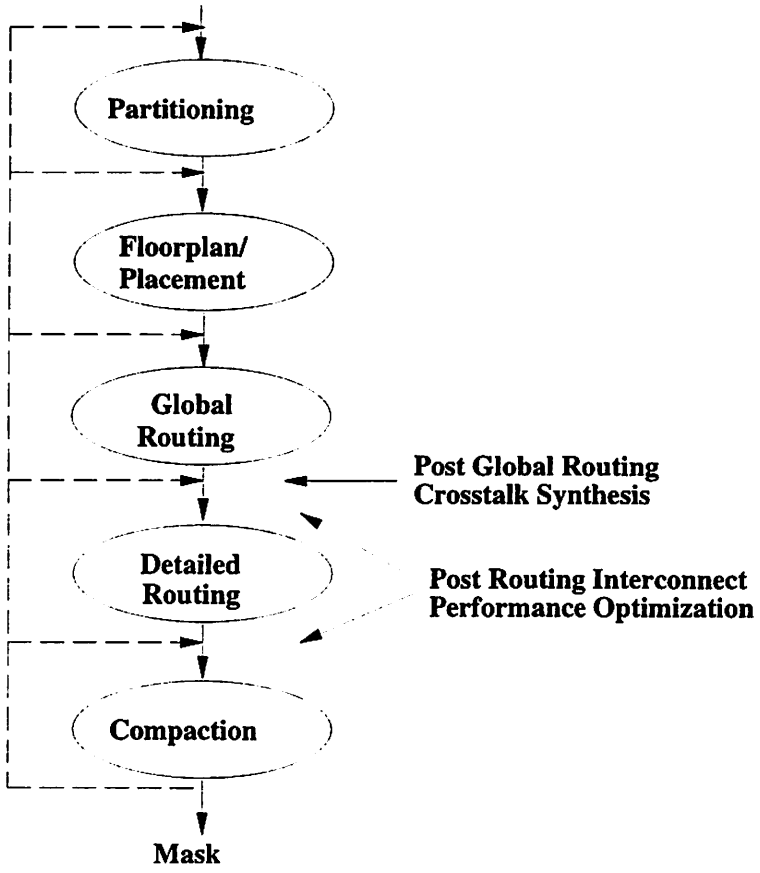

Figure 1.2: Flows of Physical Design

The work in this thesis focuses on performance optimization at the post routing level.

Both Physical Design and VLSI Design are iterative in nature, either within each step or between steps in the process. Most stages in the design flow share similar optimization objectives such as minimizing chip area, interconnect delay, power consumption, etc. Since the early 80's, great progress has been made in automatic synthesis at all stages for efficient design representation and optimization. Nevertheless, the growing complexities and aggressive performance requirements for circuit designs under advanced silicon technologies have raised many new problems yet to be solved and much work remains to be done in computer-aided VLSI design.

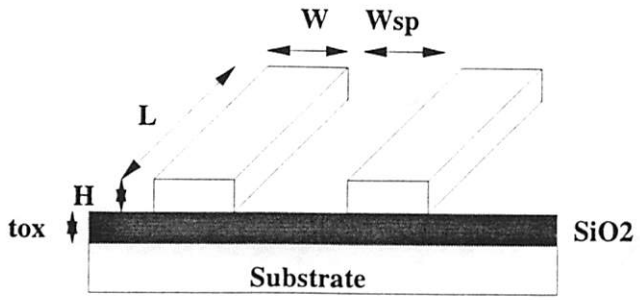

Figure 1.3: Modeling of Interconnects on the Chip

1.2 Interconnect Performance Optimization

1.2.1 Motivation

One of the most important challenges facing the physical design community today is the increasing dominance of interconnect performance in VLSI circuit design. Interconnect wires introduce capacitive, resistive and inductive parasitics into the circuit and can affect its operations. As silicon technology moves rapidly into the deep sub-micron territory, i.e., $0.35\mu m$ and below, the dimensions of devices and interconnect wires are scaling down constantly. As a result, the delay due to active devices in the circuit has decreased, while the parasitic effects of interconnects have increased significantly and begun to dominate the circuit performance. This situation is aggravated by the fact that the advanced technologies make the production of larger chips economically feasible, resulting in increases in the lengths of interconnect wires and their parasitic effects.

Fig. 1.3 shows the modeling of a pair of interconnect wires on a chip, whose parasitics, wire resistance R_{int}, wire capacitance C_{int} and coupling capacitance C_{coup}, can be expressed simply as:

$$R_{int} = \frac{\rho}{H}\frac{L}{W} \quad C_{int} = \frac{\epsilon_{ox}}{t_{ox}}LW \quad C_{coup} = \frac{\epsilon_{ox}}{W_{sp}}LH \tag{1.1}$$

where L, W, H are the length, width and height of the interconnect wire respectively, W_{sp} is the separation between interconnects, t_{ox} is the thickness of the oxide separating interconnects and substrate. ρ and ϵ_{ox} are the resistivity of the material and the permittivity of the oxide respectively.

When the technology is scaled down by a factor of S and the chip size is increased by a factor of S_C, the scaling effect on interconnect geometries, parasitics and RC delays can be summarized in Table 1.1 [Bakoglu 90]. Under Ideal Scaling, all interconnect dimensions are scaled down by the same factor S. As a result, the interconnect delay along the wire measured by the RC

Table 1.1: Scaling Effect of Interconnects

Parameter	Ideal Scaling	Quasi-Ideal Scaling
Thickness (H)	$1/S$	$1/\sqrt{S}$
Width (W)	$1/S$	$1/S$
Separation (W_{sp})	$1/S$	$1/S$
Oxide thickness (t_{ox})	$1/S$	$1/\sqrt{S}$
Length (L)	S_C	S_C
Resistance (R_{int})	$S^2 S_C$	$S^{3/2} S_C$
Coupling Cap. (C_{coup})	S_C	$\sqrt{S} S_C$
Capacitance (C_{int})	S_C	S_C
RC delay (t_D)	$S^3 S_C$	$S^{5/2} S_C$

product grows in a fourth-order fashion for $S_C = S$, due to the significant increase in interconnect resistance. Thus, it may start to dominate the chip performance under deep sub-micron technologies. To reduce the interconnect delay, other non-uniform scaling methods such as Quasi-Ideal Scaling can be adopted, which scales the vertical dimension only by factor $\sqrt{S}$ and thus reduces the delay by factor $\sqrt{S}$. However, since the wires become "taller" and "closer" to each other under such non-uniform scaling, other unwanted side effects are introduced due to the increased coupling capacitances (by factor $\sqrt{S}$) between interconnects.

Since both the interconnect delay and coupling capacitance have increased significantly after scaling, they have become very important issues to be dealt with under deep sub-micron technologies. In current estimates [Bakoglu 90], interconnect delay contributes up to 70% of the clock cycle in dense and high performance circuits, and thus must be reduced in order to further improve the operating frequencies of circuits. Crosstalk noise due to coupling capacitance is another serious concern especially under non-uniform scaling. If un-optimized, it may cause additional signal delay, logic hazards and even malfunctioning of the circuit.

1.2.2 The State of Art

In recent years, *performance-driven physical design* has become an active topic of CAD research, which aims at improving the interconnect performance of circuits at various stages in the layout process. Unlike conventional methods in physical design, which only attempt to generate a feasible layout solution of the chip with minimum area, performance-driven physical design has multiple objectives, which in addition to wirability and area minimization, include other important

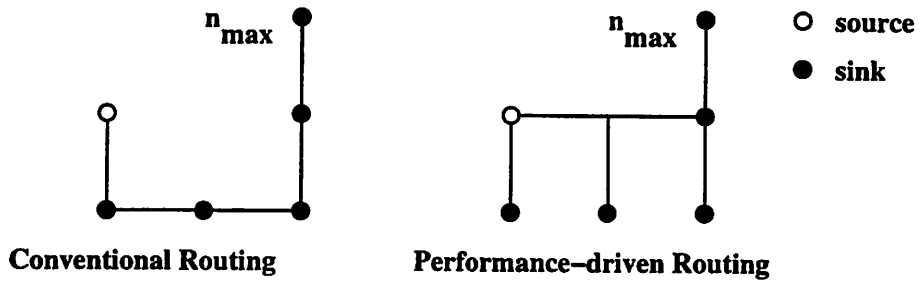

Figure 1.4: Conventional vs. Performance-driven Routing

metrics under deep sub-micron technologies, such as interconnect delay, clock skew, crosstalk noise, etc. The difficulty in generating a feasible layout solution having satisfactory chip performance is further aggravated by the fact that these multiple objectives are usually incompatible with each other. For example, improving interconnect delay may often require topology changes of wires that result in longer routes, more routing space consumption and larger chip area.

One key to performance-driven physical design is the performance-driven routing problem in which the objective is to generate routing topologies of *nets* under the current routing condition of the chip such that their interconnect performance in terms of delays and skews are satisfactory. In routing formulation, each net consists of a set of terminals carrying the identical signal from the source (output of a gate) to several sinks (input of gates). The difference between performance-driven and conventional routing can be further explained by the two different routing topologies of a 6-pin net shown in Fig. 1.4.

The topology of a conventional routing method minimizes the total wire length of the net. A performance-driven routing method, however, may produce a different topology of the net. Although its total wire length is larger than that by conventional method, its interconnect delay at the maximum delay node n_{max}, i.e., the time taken by the signal to reach certain voltage threshold at n_{max}, is smaller due to the much shorter path length from the source to n_{max}. The maximum *delay skew* of the net (maximum difference between node delays) also becomes smaller in the topology by performance-driven routing.

Current approaches to performance-driven routing focus on either the pre-routing level or the performance-driven placement and global routing phase.

- At the pre-routing stage, methods are developed to generate routing topologies of nets which minimize their interconnect delays or skews via topology construction or wiresizing.

- During placement and global routing, timing violation checking is applied to nets' topologies which estimates the current interconnect performance of nets and prohibits changes in nets' topologies that may lead to unsatisfactory net performance.

Both approaches can improve interconnect performance at certain stages in the layout process. However, due to the lack of routability considerations and the inherent incompatibilities between layout objectives, they can not guarantee satisfactory performance of a chip after its final layout solution is generated.

Despite its growing importance in high performance circuit design, crosstalk noise estimation and reduction has not been well addressed so far. Previous approaches focus mainly on the late stages in the layout process: post processing (compaction) and detailed routing.

- At the post processing stage, existing methods try to reduce the coupling noise between interconnect wires by adjusting the separating space between them.

- At the detailed routing stage, current approaches minimize the crosstalk noise at each individual net within a routing region by assigning them appropriately to corresponding tracks.

Although these localized approaches can reduce the crosstalk noise to a certain extent, their effectiveness often depends on the layout solutions from higher levels (e.g. global routing) since the routing flexibility in adjusting routes of nets is very limited at late stages in the layout process.

In summary, current approaches for interconnect performance optimization are often insufficient to generate final layout solutions with satisfactory performance in terms of routing area, interconnect delay, skew and crosstalk noise, etc. For satisfactory results, interconnect performance must be addressed comprehensively at not just one, but multiple stages in the layout process, and more effective methods for interconnect delay and crosstalk synthesis are very much in need.

1.2.3 Contribution - Post Routing Performance Optimization

Since both the interconnect delay and crosstalk noise are routing dependent, i.e., they are determined by the topologies of interconnect wires routed on the chip, it is appropriate to address these issues at the *post routing level* when a feasible routing solution of the chip has been obtained. From the chip routing solution, the current interconnect performance can be estimated accurately and those nets critical to chip performance can be identified and the routing resources available for their optimization can be computed. Therefore, we propose the *post routing performance optimization* process in performance-driven physical design, which consists of two main parts: post

routing interconnect performance optimization and post global routing crosstalk estimation and reduction (Fig. 1.2).

For interconnect performance optimization, methods are proposed for both distributed RC line and lossy transmission line topologies covering on-chip as well as off-chip interconnects in deep sub-micron IC, MCM and PCB circuit. The basic idea of these approaches is to achieve satisfactory chip performance by improving the maximum delay and skew of those critical nets using the routing space still available on the chip at the post routing stage. Unlike previous approaches at the pre-routing stage or during the placement/routing process, our methods consider the routability of those nets' topologies being optimized and keep the current routing solution feasible during the performance optimization process. With such routing resource-constrained optimization, the potential conflict among multiple objectives in performance-driven routing - routing feasibility and performance - may be resolved. The proposed approaches serve as complements rather than substitutes to other performance-driven routing methods, and they can be applied to any routing topologies which are either un-optimized or have been improved in other ways. If satisfactory chip performance can be achieved under current routing constraints, our post routing optimization approaches can speed up the physical design process by avoiding the time-consuming iterations in layout that is not guaranteed to converge.

To achieve a crosstalk risk-free layout solution of a chip, it is important to pursue crosstalk estimation and reduction not only at the detailed routing level as in previous approaches, but also at higher levels in the layout process such as the global routing level. At the global routing stage, an overall estimation of the current routing and crosstalk situation of the chip can be obtained and it is possible to adjust nets' routes globally on the chip for crosstalk reduction due to the high degree of routing flexibility available. By allowing early crosstalk estimation of the chip, the crosstalk synthesis in global routing can identify and eliminate crosstalk violations in a timely manner without progressing further into the detailed routing stage. In addition, it partitions the *risk tolerance bound* of each net - the maximum amount of noise it can tolerate without causing malfunctioning of the circuit - among its routing regions on the chip and thus permits constraint-driven crosstalk synthesis for each routing region at later stages in the optimization process. As its output, the post global routing crosstalk risk estimation and reduction approach generates a global routing solution of the chip in which every region is crosstalk risk-free, plus the partitions of bounds which reflect the current crosstalk situation of the chip. This information can greatly aid a crosstalk-driven detailed router in generating a final risk-free solution without time-consuming layout iterations and route adjustment.

1.3 Organization of the Thesis

This thesis consists of two parts. The first part describes the post routing interconnect performance optimization (Chapter 2 and 3); The second part is on crosstalk estimation and reduction at the global routing level (Chapter 4 and 5).

Chapter 2 presents approaches to post routing performance optimization of distributed RC line topologies via link insertion and wiresizing. First, the theoretical soundness of the link insertion approach for interconnect delay and skew reduction is established. Second, both single and multiple link insertion and wiresizing algorithms are developed which aim at achieving satisfactory chip performance with minimum new link routing area.

Chapter 3 discusses the post routing performance optimization of lossy transmission line topologies via high order moment computation. An exact moment matching model is first introduced which allows higher order moments (sensitivities) to be computed recursively from lower order moments for tree networks. Then, a sensitivity-based wiresizing approach for maximum delay minimization of an existing topology is designed.

Chapter 4 analyzes the region-based crosstalk estimation method at the global routing level. A crosstalk risk graph is first constructed for each routing region representing its crosstalk situation. The crosstalk risk of each region, which indicates whether a risk-free routing solution of the region is possible, is then quantitatively defined and estimated using a graph-based optimization approach.

Chapter 5 presents approaches to crosstalk risk reduction at the global routing level. For accurate risk estimation using the method in Chapter 4, the initial partitions of nets' risk tolerance bounds are adjusted appropriately among their routing regions via integer linear programming. If positive risk regions still exist after bounds partitioning, global routes adjustment is applied for crosstalk risk reduction.

The thesis concludes with Chapter 6, which summarizes the contributions and proposes future directions of this work.

Chapter 2

Post Routing Optimization of Distributed RC Lines

2.1 Introduction

The recent advent of sub-micron and deep sub-micron technologies has led to continuous scaling down in feature sizes and increase in chip area. As a result, interconnect delay has become a dominant factor in chip performance and must be addressed properly in performance-driven physical design for high density ICs and MCMs. Next generation of layout tools must have the ability to produce feasible solutions for sophisticated routing problems and guarantee their performance.

Many performance-driven routing algorithms have been proposed in recent years. Unlike conventional routing approaches which minimize the total wire length of the net, the objective of performance-driven routing is to construct routing topologies which minimize the interconnect delays of nets. Some methods adopt a geometric approach to delay minimization [Cong 92, Cong 93b, Hong 93], which considers both the total wire length of the net and the path lengths between source and sinks during topology construction; others [Boese 94, Hong 93] adopt Elmore delay[Elmore 48] or its upper bounds as the optimization goals. In addition, [Cong 93a, Sapatnekar 94, Hodes 94] employ interconnect wiresizing techniques to minimize the maximum delay of a tree topology by adjusting its wire width under Elmore delay model. In [McCoy 94], a non-tree routing approach is proposed, which greedily adds new edges into an existing tree topology based on a geometric routing graph as long as that leads to reduction in the maximum delay of the net.

Another category of problems closely related to performance-driven routing is clock routing, which aims at minimizing the maximum delay skew among sinks of a net. Numerous approaches have been proposed for zero-skew solutions. [Tsay 91] proposes the first approach for clock skew minimization using Elmore delay model. [Edahiro 91] etc. develop the DME method which builds a zero-skew clock tree with minimum total wire length in a bottom up and top-down fashion. [Zhu 93, Pullela 93] employs sensitivity-based wiresizing method to minimize the clock skew of a given routing topology. Recently, DME method has been extended to allow the construction of minimum wire length routing tree with skew within prescribed bounds[Cong 95, Huang 95].

Most of these existing performance-driven and clock routing algorithms can be characterized as pre-routing optimization methods, i.e., they construct the optimal routing topology for each net individually on a regular grid routing graph without considering its routability on the entire chip. During the chip routing process, all these initial optimal net topologies are to be routed on the chip simultaneously and they may compete for the limited routing resources available. As a result, some of them may not be realizable because of the routing congestions and blockages on the chip and are therefore subject to significant modifications in order to generate a feasible solution of the chip under routing resource constraints. Due to the changes in their topologies, the performance of these nets, which are optimized in initial construction, may no longer be satisfactory. For example, the initial topology of the 3-pin net in Fig. 2.1 generated by the pre-routing optimization method minimizes the net's maximum delay at sink 2. During the routing process, part of the initial topology is ripped up and re-routed in order to avoid congested routing regions on its original route. Due to the prolonged path length from source to the maximum delay sink 2 after routing, the maximum delay of the net increases and may no longer satisfy its specified requirement.

Beside the lack of routability considerations, most pre-routing methods set maximum delay or skew minimization as their objectives, which often comprises other important goals in routing such as routing area minimization. In most real design situations, minimizing the net routing area is the objective of optimization, while the maximum delay and skew of the net are only required to satisfy certain specified bounds. The difficulty with constraints specification at the pre-routing stage is that, although some initial information on interconnect performance requirements of the chip can be specified by the user, the actual critical nets and the constraints on their maximum delays or skews are routing-dependent, which can only be known exactly after a routing solution of the chip is obtained. Therefore, the constrained optimization problem can not be handled efficiently by those pre-routing methods.

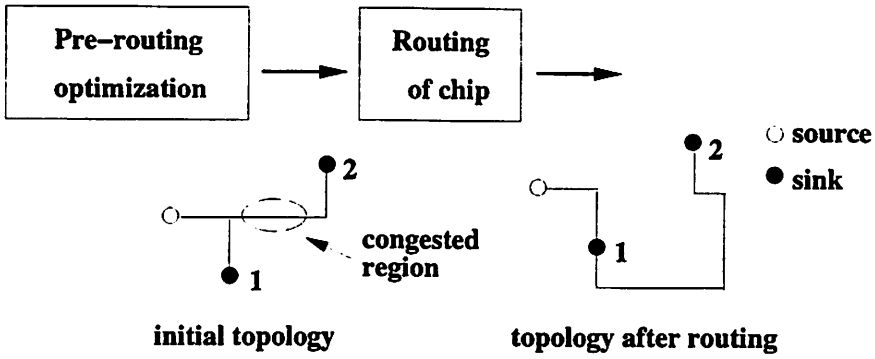

Figure 2.1: Problem with Pre-routing Optimization

Since pre-routing approaches cannot guarantee net performance after routing, timing violation checking is required during the routing process, which checks the nets' delays and skews and prohibits topology changes that may result in unsatisfactory net performance. However, timing violation checking also has some difficulties in generating a solution with satisfactory performance for the following reasons:

- Since the critical nets on the chip may keep on changing during the routing process, timing checking may have to be carried out on most nets in the circuit during the routing process, which is very time consuming. For fast checking, over-simplified delay models are often used in actual implementations which may cause inaccurate delay estimations and mis-guided topology modifications.

- Since timing violation checking prohibits certain nets to be rerouted during the routing process, it may seriously restrict the routing flexibility of the chip and result in infeasible or over-constrained routing solutions that are un-optimized in chip area.

- Finally, the primary objective in routing is to generate a feasible routing solution under limited routing resources. For that purpose, the topologies of certain nets may have to be ripped-up and re-routed even if that may result in timing violation of these nets.

Due to these difficulties, timing violation checking also cannot guarantee the chip performance after the routing is completed.

To overcome these limitations, we propose the post routing performance optimization approach [Xue 95a, Xue 95b] in this chapter, which is one extra step in the performance optimization process applied after a feasible routing solution of the chip is obtained (Fig. 2.2). From the chip

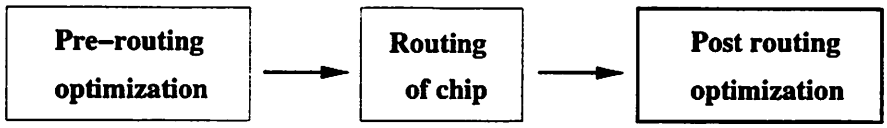

Figure 2.2: Post Routing Performance Optimization

routing solution, the critical path and critical nets of the circuit can be identified and the performance constraints for these nets in terms of their maximum delay and skew requirements be computed. In addition, the routing space available on the chip can be estimated based on the densities and capacities of its routing regions. Using these information, the proposed optimization process achieves satisfactory chip performance by improving the maximum delay and skew of those critical nets to satisfy their corresponding specifications under routing resource constraints. The current routing solution of the chip is kept feasible during the optimization process, which helps to speed up the design process by avoiding the iterations in layout flows often required by the existing routing tools which are time-consuming and not guaranteed to converge.

The basic approach for post routing optimization is link insertion and wiresizing, which establishes new interconnect wires having the shortest feasible lengths between the source and nodes in the existing topology of each critical net, and wiresizes each inserted link non-uniformly under routing resource constraints for performance optimization. The objective is to satisfy the performance requirements of each net with minimum link routing area consumption. Since the size of a chip is largely determined by the few congested regions in its routing solution, most routing regions on the chip are not fully occupied and have plenty of routing space left. These available routing resources can be utilized by the proposed approach for link insertion and wiresizing.

Compared with previous performance-driven and clock routing methods, the proposed approach has the following advantages:

1. It achieves reduction in both the maximum delay and skew of a net as shown by analysis in Section 2.3, while previous clock routing algorithms often sacrifice delay for skew minimization.

2. It is formulated as a constraint-driven optimization process instead of maximum delay or skew minimization and thus allows the trade-offs between routing area and net performance.

3. It is applicable to any arbitrary topology including tree and mesh structure, while most

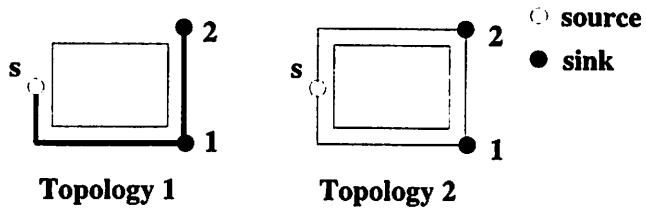

Figure 2.3: Link Insertion vs Wiresizing

previous methods restrict topologies to trees. Although tree structure is the most efficient for total wire length minimization in conventional routing, it does not necessarily correspond to good net performance in performance-driven routing and it limits the flexibility in routing topology construction.

Since link insertion changes both the net topology and its admittances, it has advantages over wiresizing-only approaches for performance optimization as demonstrated by the example in Fig. 2.3. Due to its existing topology, wiresizing on Topology 1 alone can never reduce the skew between sink 1 and 2 to zero. On the other hand, Topology 2 with a new link added between the source and sink 2 achieves larger reduction in maximum delay, zero skew between the sink 1, 2 and less routing area consumption compared with double wiresizing of Topology 1.

It is worth noting that the proposed link insertion and wiresizing approach is aimed at complementing, not replacing other methods for performance improvements such as topology construction, wiresizing, buffer insertion, etc. It can be applied to any routing topology - un-optimized or improved by other approaches - in order to achieve further improvement in its performance so that a satisfactory solution of the chip can be obtained under the current routing conditions without the expensive iterations in layout process.

The rest of this chapter is organized as follows. Section 2.2 discusses the delay model for distributed RC line topologies; Section 2.3 analyzes the theoretical soundness of our link insertion approach for performance improvement; Section 2.4 investigates the single link insertion and wiresizing method; Section 2.5 discusses the multi-link insertion and wiresizing method; Section 2.6 shows experimental results which demonstrate the effectiveness of our approach for post routing performance optimization; finally, Section 2.7 gives concluding remarks.

2.2 Interconnect Delay Modeling

2.2.1 Problem Formulation

In early technologies, the resistance of the interconnect wire was negligible compared to the driver resistance and the interconnect delay estimation modeled the total interconnect capacitance of the net as the capacitive load at the source. Due to the scaling down of geometric features under deep sub-micron technologies and the increase in chip area, the interconnect resistance increases drastically along with the interconnect capacitance, and it can no longer be ignored in interconnect delay estimation. To take interconnect resistance into account, we model each interconnect wire segment under deep sub-micron technologies, as a distributed RC line, which unlike lumped circuit, has the interconnect resistance and capacitance distributed uniformly along the wire. In the following analysis, the interconnect routing topology of critical net n on the chip is formulated as a distributed RC line network and denoted by N. N can be any arbitrary topology and is not restricted to tree structures.

For routing topology N, we adopt the convention that its ground node is not numbered, its source consists of a voltage source in series with a driver resistance R_d inserted between the ground and reference node n_{ref}, which separates the source from the rest of the topology (Fig. 2.4). The remaining nodes in N are numbered from 1 to $|N|$. The loading capacitance at every sink node in N is denoted by C_s. The two key elements in delay estimation of RC network N is its resistance matrix, $\mathbf{R} = [R_{ij}]$, and its capacitance vector $\mathbf{c} = [C_j]$. Each entry R_{ij} in $\mathbf{R}$ is equal to the potential in volts at node i if a $1A$ current were injected into node j while all nodes in N other than j were open circuited[Ruehli 86]. By its definition, R_{ij} is the mutual resistance between node i and j, which reflects how the voltage and current changes at one node may affect the other. In the special case where N is tree-structured, R_{ij} is simply the total resistance along the common path shared by node i and j. Each item C_j in the capacitance vector $\mathbf{c}$ is the ground capacitance at node $j \in N$, which includes both the sink capacitance at j and the capacitances of wires connecting to j under π lumped model.

2.2.2 Elmore Delay Model

We use Elmore delay for delay and skew estimation, which is the first order moment of the impulse response [Elmore 48]. The Elmore delay at node i, D_i, can then be expressed as,

$$D_i = \int_0^\infty t\dot{v}_i(t)dt = \int_0^\infty (v_e - v_i(t))dt \tag{2.1}$$

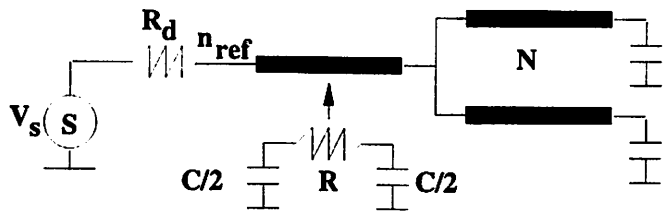

Figure 2.4: Formulation of Distributed RC Line Network N

where $v_i(t)$ and v_e are the current and final voltage at node i, respectively. For an RC network, the the voltage difference between v_e and $v_i(t)$, is given by:

$$v_e - v_i(t) = \sum_j R_{ij} C_j \dot{v}_j(t) \tag{2.2}$$

Substituting Eqn (2.2) into Eqn (2.1), the Elmore delay at node i is:

$$D_i = \sum_j R_{ij} C_j (v_j(\infty) - v_j(0)) = \sum_j R_{ij} C_j \tag{2.3}$$

According to Eqn (2.3), the Elmore delay at node i takes into account both the interconnect resistance along the path from the source to node i and the interconnect capacitance of the rest of the topology. It is the most accurate delay metric known that can be analytically computed as a function of interconnect resistance and capacitance, i.e., the geometric parameters of the wires. For exact Elmore delay computation, each distributed RC line can be represented by the equivalent Π lumped model (Fig. 2.4), in which R and C are the total resistance and capacitance of the wire segment, respectively. This modeling is not only valid for tree structures, but also for any arbitrary topologies which may contain meshes[Yu 95].

[Robinstein 83] shows that the response waveforms at nodes in a distributed RC line topology is monotonic, indicating that the impulse response at node i is non-negative. The 50% delay of the step response, τ_i, can then be expressed as:

$$\int_0^{\tau_i} \dot{v}_i(t) dt = 0.5 \tag{2.4}$$

i.e., the 50% delay τ_i is the median of the impulse response. Since $\int_0^\infty \dot{v}_i(t) dt = 1$, the non-negative impulse response $\dot{v}_i(t)$ can be treated as a distribution function and the Elmore delay is its mean according to Eqn (2.1). Therefore, Elmore delay D_i estimation approximates the actual 50% delay τ_i (the median) by the mean. When the impulse response is symmetric, its mean equals its median and Elmore delay estimation is accurate. However, the actual impulse response is always

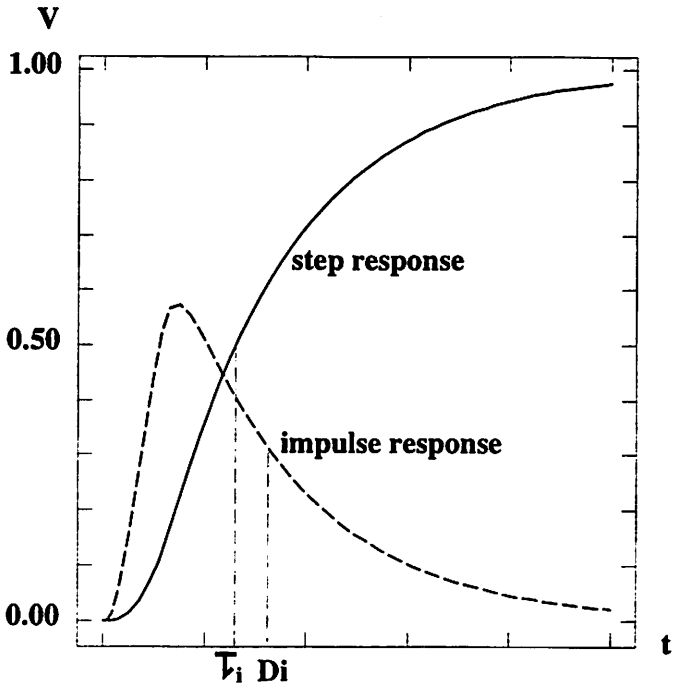

Figure 2.5: Step and Impulse Response at Node i

skewed asymmetrically towards left (Fig. 2.5), thus making its mean an upper bound on its median, i.e., Elmore delay is an upper bound on the 50% delay. [Gupta 95] indicates that D_i is also the upper bound on τ_i for any monotonic input signals besides the step input, and D_i asymptotically approaches τ_i if $\dot{v}_i(t)$ becomes "less skewed" under the conditions that: 1. node i is far away from the source. 2. the rising ramp of the input signal is slow.

Unlike simulation which requires perfect fitting between actual and approximated waveforms, the delay estimation used in guiding the construction of routing topologies is only required to have high degree of reliability, not precise accuracy. Here, high reliability means that the optimal solution constructed using Elmore delay estimation is also nearly optimal in terms of actual delay measured by SPICE simulation. [Boese 93] shows that although Elmore delay may not be the most accurate metric for delay estimation, it is very reliable for the construction distributed RC line topologies, which is also witnessed in our experiments. Therefore, Elmore delay is chosen for delay estimation in our proposed post routing performance optimization approach.

2.3 Analysis of Network with an Inserted Link

2.3.1 Definitions

For performance analysis, D_{max} is denoted as the maximum node delay of topology N, i.e., $D_{max} = max\{D_i | i \in N\}$. The delay skew between node i, j is defined as $DS_{ij} = D_i - D_j$ and the maximum delay skew in N is: $DS_{max} = max\{DS_{ij} | i, j \in N\}$. To study the impact of link insertion on delay and skew of the net, we establish a new routing topology $(N)_n$ by introducing an additional link e_n between n_{ref} and node n chosen in N. The wire resistance and capacitance of e_n are denoted by R_{e_n} and C_{e_n} respectively, the interconnect delay introduced by e_n due to R_{e_n} and C_{e_n} is denoted by D_{e_n}. Denote the resistance matrix of $(N)_n$ by $\mathbf{R}_n$, the delay at node i and the delay skew between node i, j in $(N)_n$ by $(D_i)_n$ and $(DS_{ij})_n$, respectively. Since the inserted link between n_{ref} and node in N does not affect the driver resistance R_d of the net (which is part of R_{ij} between every pair of node $i, j \in N$), R_d is excluded from R_{ij}s in the following discussions.

2.3.2 Changes in Node Delay

For node delay analysis, we first compare the entries in resistance matrices of N and $(N)_n$, i.e., $\mathbf{R}= [R_{ij}]$ and $\mathbf{R}_n= [(R_{ij})_n]$. The inserted link e_n at node n introduces extra admittance at n, and thus causes reductions in the mutual resistances between node n and the rest of nodes in N. Since e_n may provide an additional path to n_{ref} via node n from each node in N, it may affect the mutual resistance R_{ij} between node pairs $i, j \neq n$ as well. The change in R_{ij} due to the inserted link at n is determined by the following factors: 1. The wire resistance of the inserted link R_{e_n}; 2. The resistance at the chosen node R_{nn}; 3. Values of R_{ni} and R_{nj}, i.e., how strongly node i, j are related to n. Intuitively, an insertion link at node n has strong influence on its neighboring nodes, plus, a link having small resistance introduces large admittance into the topology, resulting in large reduction in resistances between nodes. The mutual resistances between nodes after link insertion can be quantitatively expressed by the following Lemma (its proof is given in Appendix A1):

Lemma 2.1 *(Mutual Resistances after Link Insertion) When link e_n is inserted between the reference node n_{ref} and node $n \in N$, the mutual resistance between node $i, j \in N$, $(R_{ij})_n$, can be expressed as:*

$$(R_{ij})_n = R_{ij} - \frac{R_{ni}}{R_{nn} + R_{e_n}} R_{nj} \tag{2.5}$$

Lemma 2.1 indicates that mutual resistances between node pairs can be reduced via link insertion. Since Elmore delay is determined by the product of $\mathbf{R}$ and $\mathbf{c}$ according to Eqn (2.3), link

insertion has the potential to achieve reduction in node delays. On the other hand, the inserted link e_n also introduces certain amount of wire delay D_{e_n} into the topology due to its own wire resistance R_{e_n} and capacitance C_{e_n}. The delay at node i after link insertion at node n can be expressed by the following theorem based on Lemma 2.1:

Theorem 2.1 *(Node Delay after Link Insertion) When a link e_n is inserted between n_{ref} and node $n \in N$, the delay at node $i \in N$ after link insertion, $(D_i)_n$, can be expressed as:*

$$(D_i)_n = D_i - \frac{R_{in}}{R_{nn} + R_e}(D_n - D_{e_n}) + R_d C_{e_n} \tag{2.6}$$

Proof:

In our proof, the delay changes due to interconnect and driver resistance are separated, and $(D_i)_n$ can be expressed according to Elmore delay definition as:

$$(D_i)_n = \sum_j (R_{ij})_n (C_j)_n + R_d C_{e_n}$$

which can be further divided into delay contributions from nodes in original network N and from the inserted link, respectively:

$$(D_i)_n = \sum_{j \in N} (R_{ij})_n (C_j)_n + \sum_{j \in e_n} (R_{ij})_n (C_j)_n + R_d C_{e_n}$$

Since D_{e_n} is the delay of e_n due to its own wire resistance and capacitance, it can be easily shown that the delay introduced by e_n after it is inserted at node $n \in N$ can be expressed as:

$$\sum_{j \in e_n} (R_{ij})_n (C_j)_n = \frac{R_{in}}{R_{nn} + R_{e_n}} D_{e_n} \tag{2.7}$$

Therefore,

$$\begin{aligned}(D_i)_n &= \sum_{j \in N} R_{ij} C_j - \sum_{j \in N} \frac{R_{in}}{R_{nn} + R_{e_n}} R_{nj} C_j + \frac{R_{in}}{R_{nn} + R_{e_n}} D_{e_n} + R_d C_{e_n} \\ &= D_i - \frac{R_{in}}{R_{nn} + R_e}(D_n - D_{e_n}) + R_d C_{e_n}\end{aligned}$$

□

It can be seen from Theorem 2.1 that whether the delay at node i can be reduced after link insertion depends on the mutual resistance between node i and n, R_{in} and the wire delay of e_n, D_{e_n}, which is typically much smaller than the delay at the node it connects, i.e., $D_{e_n} \ll D_n$, unless the length of the inserted link is much larger than the total wire length of the net. Therefore, link

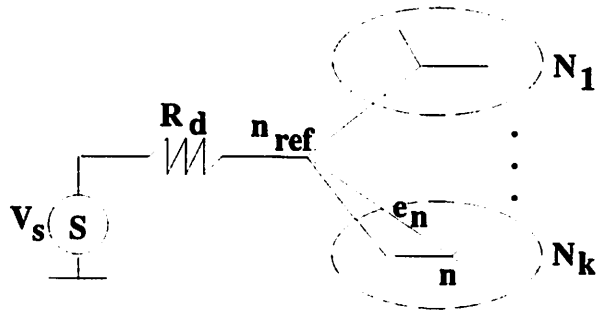

Figure 2.6: Decomposition of N

insertion at node n leads to delay reduction at node i if the delay reduction due to reduced mutual resistances between nodes offsets the delay increase due to its wire capacitance.

To analyze the impact of R_{in} on delay reduction, we decompose topology N into several sub-components $N_1, N_2, \ldots, N_k$ at its reference node n_{ref}. These sub-components branch out from n_{ref} and connect to each other via n_{ref} only, i.e., the removal of n_{ref} results in the decomposition of N into $k+1$ disjoint components. In graph theory terminology, n_{ref} is denoted as an *articulation point* of N (Fig. 2.6). Suppose node n is chosen within the sub-component N_k, the value of R_{in} is analyzed in the following cases:

1. If node i belongs to the same sub-component as n, i.e., $i, n \in N_k$, $R_{in} \neq 0$ and the delay at node i can possibly be reduced by link insertion at node n according to Theorem 2.1.

2. If node $i \notin N_k$, $R_{in} = 0$, indicating that the delay at node i can not be reduced by link e_n.

If n_{ref} is not an *articulation point* of N, i.e., there exists only one component at n_{ref} and $N = N_k$, the delay at every node in N can possibly be reduced, so does the maximum delay of the net $(D_{max})_n = max\{(D_i)_n | i \in N\}$. The delay reduction under the situation when n_{ref} is an articulation point of N and there exist multiple sub-components rooted at n_{ref} can be handled by inserting links to each sub-component containing the maximum delay node(s).

2.3.3 Changes in Delay Skew

The reductions in delays at different nodes in N after link insertion are influenced by their mutual resistances with n, and therefore can be different: The delays at nodes "closer" to n i.e., having larger R_{ni}s, are reduced more than those at nodes "further" away from n. Due to the

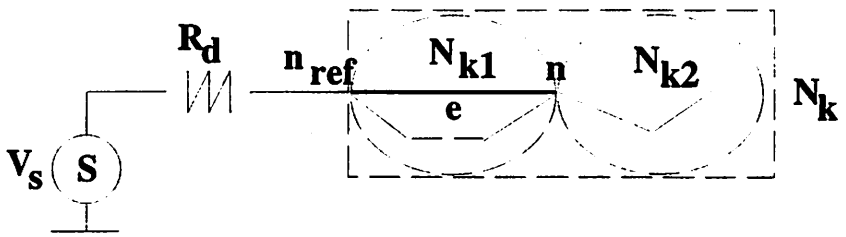

Figure 2.7: Decomposition of N_k

unbalanced reduction in node delays, the delay skew between nodes may also change, which can be expressed by the following corollary obtained directly from Theorem 2.1.

Corollary 2.1 *(Delay Skew after Link Insertion) When a link e_n is inserted between n_{ref} and node $n \in N$, the delay skew between node pair i, j, $(DS_{ij})_n$, can be expressed as:*

$$(DS_{ij})_n = DS_{ij} - \frac{R_{in} - R_{jn}}{R_{nn} + R_{e_n}}(D_n - D_{e_n}) \tag{2.8}$$

According to Corollary 2.1, the change in delay skew between node i, j is determined by the difference between their mutual resistances with node n: $RS(i, j, n) = R_{in} - R_{jn}$, assuming that $D_{e_n} \ll D_n$. To study the impact of link insertion on $RS(i, j, n)$, we further decompose component N_k into two sub-components, N_{k1} and N_{k2}. Here, n is an *articulation point* of N_k and N_{k2} is a branched out component from n, connecting to N_{k1} and rest of the topology only at n (Fig. 2.7). The value of $RS(i, j, n)$ is investigated under the following cases:

1. If both $i, j \in N_{k2}$, each path from i, j to the n_{ref} must pass through node n. Thus, $R_{in} = R_{jn} = R_{nn}$ and $RS(i, j, n) = 0$, which implies that the inserted link does not affect the delay skew between node i and j.

2. If $i \in N_{k2}$ and $j \in N_{k1}$, $R_{in} = R_{nn} > R_{jn}$ and $RS(i, j, n) > 0$, the delay skew between i, j decreases as the result of the link insertion.

3. If $i \in N_k$ and $j \notin N_k$, $R_{in} \neq 0$ but $R_{jn} = 0$, thus $RS(i, j, n) > 0$ and $(DS_{ij})_n$ decreases.

4. If both $i, j \notin N_k$, DS_{ij} will not be affected by the link insertion.

The delay and skew analysis in this section indicates that, unlike previous methods which often sacrifice delay for skew minimization, the proposed approach can achieve reduction in both node delay and delay skew simultaneously if a link is inserted appropriately between n_{ref} and node in N. This establishes the theoretical soundness of the link insertion and wiresizing algorithms for post routing performance optimization, which are discussed in the following sections.

2.4 Single Link Insertion and Non-uniform Wiresizing

In this section, we investigate the method of inserting an additional interconnect wire into the existing topology of a critical net obtained from a routing solution and adjusting its wire width appropriately so that the performance of the net can be best improved. This link insertion and wiresizing approach at the post routing level is formulated as a constrained optimization process, which instead of minimizing the maximum delay and skew of the critical net, minimizes the routing resource needed to achieve satisfactory net performance:

Improve the performance of a critical net to satisfy its specified requirements by introducing minimum amount of link routing area into its existing topology via link insertion and wiresizing under routing resource constraints.

The reasons for choosing link area as the optimization objective are as follows:

1. It minimizes the wire capacitance of the inserted link and thus is consistent with interconnect delay and power reduction.

2. It saves available routing space in the regions on the chip for the possible inserted links in other critical nets and further performance optimization at later stages in the layout process.

3. The performance of critical nets is only required to satisfy certain specified constraints in most cases, minimizing delay or skew may consume more routing resources than necessary and result in un-optimized chip area.

Denote ΔD_{max} and ΔDS_{max} as the reduction in maximum delay and delay skew, respectively. ΔD_{max} exists at the maximum delay node n_{max}, ΔDS_{max} exists between n_{max} and the minimum delay node n_{min} in N. According to Eqn (2.6) and (2.8), ΔD_{max} and ΔDS_{max} due to link insertion to n can be expressed respectively as:

$$(\Delta D_{max})_n = \frac{R_{n_{max}n}}{R_{nn}+R_{e_n}}(D_n - D_{e_n}) - R_d C_{e_n} \tag{2.9}$$

$$(\Delta DS_{max})_n = \frac{R_{n_{max}n} - R_{n_{min}n}}{R_{nn}+R_{e_n}}(D_n - D_{e_n}) \tag{2.10}$$

which indicate that the reduction in maximum delay and delay skew are determined by the following two factors:

- The choice of node n for link insertion.
- The R, C values of the inserted link e_n, i.e., the length and width of e_n which are determined by link insertion and wiresizing.

These two issues are discussed separately in the following sections.

2.4.1 Node Choice for Link Insertion

2.4.1.1 Upper Bound on Performance Improvement

Although the choice of node for link insertion and the geometric dimensions of the inserted link can both influence the reduction in maximum delay and skew of the net, the upper bound on the best performance improvement achievable is determined by the node choice as shown by the following corollary, which is obtained directly from Eqn (2.9) and (2.10) with $R_{e_n}, C_{e_n} \rightarrow 0$.

Corollary 2.2 *(Upper Bound on Net Performance Improvement) The upper bounds on maximum delay and skew reduction via link insertion and wiresizing are determined by the choice of node n for link establishment, i.e.,*

$$(\Delta D_{max})_n < \overline{(\Delta D_{max})_n} = \frac{R_{n_{max}n}}{R_{nn}} D_n \tag{2.11}$$

$$(\Delta DS_{max})_n < \overline{(\Delta DS_{max})_n} = \frac{R_{n_{max}n} - R_{n_{min}n}}{R_{nn}} D_n \tag{2.12}$$

Since $R_{n_{max}n} \leq R_{nn}$ (i.e., $R_{n_{max}n}/R_{nn} \leq 1$), according to Corollary 2.2, only those links which are inserted to nodes having large delays and are "close" to n_{max} (i.e., having large $R_{n_{max}n}$) can have significant impact on maximum delay and skew reduction. Therefore, only those nodes in the neighborhood of n_{max} need be considered as possible candidates for link insertion. In particular, the link to the maximum delay node n_{max} has the greatest potential for maximum delay and skew reduction as shown by the following theorem (its proof is given in Appendix A2):

Theorem 2.2 *A link to n_{max} has the largest upper bound on both maximum delay and skew reduction.*

Theorem 2.2 implies that the maximum delay node n_{max} is a good candidate node for link insertion. However, the best node choice is also influenced by the geometric dimensions of the feasible links to the nodes, which are determined by the current routing conditions on the chip. For example, if the link to n_{max}, $e_{n_{max}}$, is very long due to the detours under routing congestions, it may not have the best effect on performance improvement due to its large $R_{e_{n_{max}}}$ and $D_{e_{n_{max}}}$ compared to links to other nodes which have much smaller lengths. For constrained optimization, a link is preferred over others if it can achieve the best performance improvement under same routing area consumption. In the next two sections, the node choice problem is discussed under two mutually-exclusive situations characterized by node delays, net topology and potential link lengths.

2.4.1.2 Situation I

Denote l_n as the length of the feasible link e_n inserted between n_{ref} and node n, p_n as the length of the route from node n to n_{ref} through the existing topology N (when multiple routes exists between n and n_{ref} in a non-tree topology, p_n denotes the shortest one). Situation I is then defined under the following two assumptions:

A1. n_{max} is unique.

A2. $l_{n_{max}}/l_n \leq p_{n_{max}}/p_n, \forall n \in N$.

A1 states that there is one unique maximum delay node in N. A2 asserts that the ratio between lengths of inserted links to n_{max} and n is no larger than the ratio between the lengths of paths from these two nodes to the source through the existing topology. According to Eqn (2.6), the maximum delay reduction favors a short link to node having large delay and Theorem 2.2 indicates that a link to n_{max} has the best potential for performance improvement. Under these two assumptions in Situation I, n_{max} is the best node choice for link insertion as stated by the following theorem (whose proof is given in Appendix A3):

Theorem 2.3 *(Choice of Node for Link Insertion: Delay) With the same routing area consumption, the link to n_{max} achieves the largest reduction in D_{max} compared with link to any other node under A1 and A2 in Situation I.*

According to the analysis in Section 2.3, link insertion may lead to simultaneous reduction in both the delay and skew of a critical net. Since maximum delay skew DS_{max} is defined as the difference between the maximum and minimum delay of the net, larger reduction in D_{max} may also result in larger reduction in DS_{max} if D_{max} is reduced more than D_{min}. Therefore, conclusion similar to Theorem 2.3 can be also drawn about the reduction in DS_{max}, which is formally stated as follows (its proof is given in Appendix A4):

Theorem 2.4 *(Choice of Node for Link Insertion: Skew) With the same routing area consumption, the link to n_{max} achieves the largest reduction in DS_{max} compared with link to any other node under A1 and A2 in Situation I.*

Theorem 2.3 and 2.4 indicate that maximum delay and skew reduction are consistent in Situation I when a link is inserted between n_{ref} and n_{max}. For constraint-driven performance optimization, whose objective is to achieve satisfactory net performance with minimum link routing area consumption, the following corollary can be established:

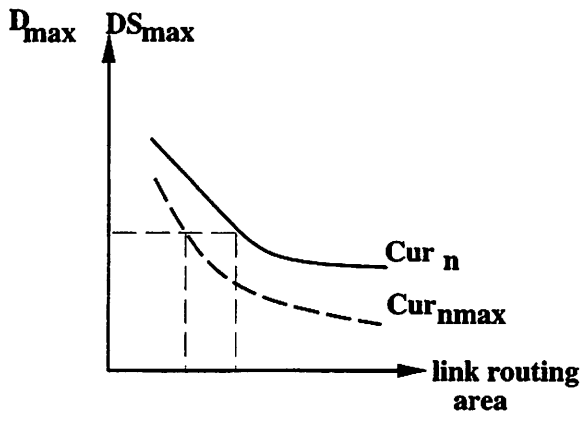

Figure 2.8: Link curves of D_{max} and DS_{max} vs Routing Area

Corollary 2.3 *(Node Choice: Situation I) For the same amount of reduction in maximum delay and delay skew, the link to n_{max} consumes minimum amount of routing area compared to links to other nodes under A1 and A2 in Situation I.*

Proof:

Assume that the route (length) of the inserted link e_n to node n is fixed, then the increase in its link routing area due to wiresizing leads to reduction in its resistance R_{e_n} and increase in its capacitance C_{e_n}. With appropriate wiresizing of the link, which will be discussed in later sections, the maximum delay and skew of the net decreases as the link routing area increases until the wire delay due to C_{e_n} dominates the delay reduction due to reduced mutual resistances. Therefore, two curves, denoted by Cur_n and $Cur_{n_{max}}$, can be plotted for links to node n and n_{max} respectively, representing the trade-offs between the link routing area and maximum delay (skew) of the net during link insertion and wiresizing (Fig. 2.8).

According to Theorem 2.3 and 2.4, the link to n_{max} achieves smaller D_{max} and DS_{max} compared with the link to node n under same link routing area consumption, i.e., $Cur_{n_{max}}$ is "lower" than Cur_n. Therefore, for any performance specification on D_{max} and DS_{max} which is achievable by links to both nodes, the link to n_{max} consumes less routing area than the link to node n, i.e., the link to n_{max} is the most link area-efficient compared to links to other nodes for the same amount of improvement in net performance.

□

Figure 2.8 also implies that a link to n_{max} can possibly achieve larger reduction in D_{max} and DS_{max} with less routing area compared with links to other nodes, which is witnessed during

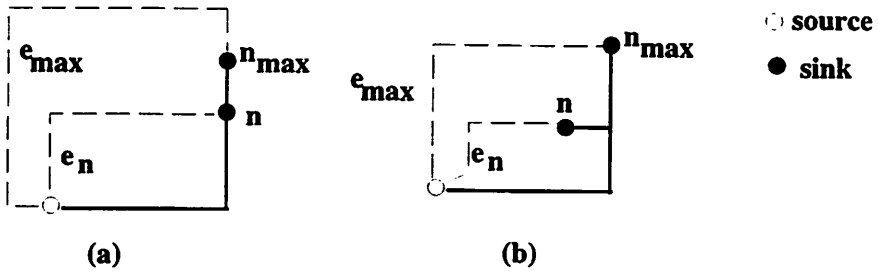

Figure 2.9: Conditions under which A1 in Situation II holds

our experiments.

2.4.1.3 Situation II

Situation II considers the node choice problem in the cases when at least one of the two assumptions in Situation I no longer holds, i.e.:

A1. There exists multiple maximum delay nodes in the net.

A2. $\exists n \in N$, s.t., $l_{n_{max}}/l_n \gg p_{n_{max}}/p_n$.

A2 happens if the path lengths from node n and n_{max} to n_{ref} through topology N, p_n and $p_{n_{max}}$ respectively, are comparable, but one of the following conditions about the link length is true:

1. $l_{n_{max}}$ is much longer than l_n due to the possible detours of e_{max} to avoid routing congestions under the current routing conditions of the chip (Fig. 2.9a).

2. Node n is actually much "closer" to the source than n_{max} in terms of Manhattan distance. As a result, $l_n \ll l_{n_{max}}$ (Fig. 2.9b).

According to Eqn (A.7) to (A.9), $l_{n_{max}}/l_n \gg R_{n_{max}n_{max}}/R_{nn}$ and $(l_{n_{max}}/l_n)^2 \gg D_{max}/D_n$ hold under A2 in Situation II. In this case, node n could be a better choice for link insertion compared to n_{max} for performance improvement according to Eqn (2.6) and (2.8), due to its relatively short link length to n_{ref}.

If A1 in Situation II holds, N_{max} is denoted as the set of nodes having the maximum delay in the net, i.e., $N_{max} = \{p | D_p = D_{max}, p \in N\}$. Since the lengths of links to different nodes in N_{max} may be different, the effectiveness of these links for performance improvement also varies.

Furthermore, if A1 and A2 in Situation II are both true, the best node choice for link insertion may not even belong to N_{max}. Therefore, there is no longer a definitive node choice for link insertion in Situation II that can be determined simply from the configurations of the net topology N. Instead, multiple nodes in N which have the potential to achieve good performance improvement should be considered and N_{cand} is denoted as the set of possible candidate nodes for link insertion.

Define $\underline{(\Delta D_{max})_n}$ ($\underline{(\Delta DS_{max})_n}$) and $\overline{(\Delta D_{max})_n}$ ($\overline{(\Delta DS_{max})_n}$) as the lower and upper bounds on maximum delay (skew) reduction by link to n, respectively; ($\underline{(\Delta D_{max})_n}$, $\overline{(\Delta D_{max})_n}$) and ($\underline{(\Delta DS_{max})_n}$, $\overline{(\Delta DS_{max})_n}$) are the *performance intervals* of node n, indicating the potential improvements in performance that could possibly be achieved by a link to node n. The actual improvements in performance after insertion and wiresizing of a link to n always fall within its performance intervals, i.e.,

$\underline{(\Delta D_{max})_n} \leq (\Delta D_{max})_n \leq \overline{(\Delta D_{max})_n}$ and $\underline{(\Delta DS_{max})_n} \leq (\Delta DS_{max})_n \leq \overline{(\Delta DS_{max})_n}$

According to Eqn (2.11) and Eqn (2.12) in Corollary 2.2, the upper bounds on performance improvement by link e_n are given by:

$$\overline{(\Delta D_{max})_n} = \frac{R_{n_{max}n}}{R_{nn}} D_n, \quad \overline{(\Delta DS_{max})_n} = \frac{R_{n_{max}n} - R_{n_{min}n}}{R_{nn}} D_n \tag{2.13}$$

The lower bounds on performance improvements are computed by assuming that e_n is un-optimized with minimum wire width w_0 (denoted by e_{n0}), i.e.,

$$\underline{(\Delta D_{max})_n} = \frac{R_{n_{max}n}}{R_{nn} + R_{e_{n0}}} (D_n - D_{e_{n0}}), \ \underline{(\Delta DS_{max})_n} = \frac{R_{n_{max}n} - R_{n_{min}n}}{R_{nn} + R_{e_{n0}}} (D_n - D_{e_{n0}}) \tag{2.14}$$

Node p should not be considered for link insertion if there exists node q s.t. $\overline{(\Delta D_{max})_p} < \underline{(\Delta D_{max})_q}$ or $\overline{(\Delta DS_{max})_p} < \underline{(\Delta DS_{max})_q}$, i.e., their performance intervals do not intersect and the best performance improvement possible by link to p is not as good as the least achievable by link to q. Therefore, N_{cand} can be established by comparing the performance intervals of nodes. First, the performance intervals of each node is computed according to Eqn (2.13) to (2.14) and node n having the largest lower bound on performance improvement is identified. Then, each node whose interval intersects with that of n (which indicates that it has the potential to achieve better performance improvement than n), is added into N_{cand}. Since a link to a maximum delay node has the largest upper bound on reduction in both D_{max} and DS_{max} according to Corollary 2.2, intervals of nodes in N_{max} definitely intersect with the one having the largest lower bound, and $N_{max} \subseteq N_{cand}$. The candidate node set can then be constructed as follows:

Candidate Node Set Construction Algorithm {

1. Compute delays of nodes in topology N, identify N_{max}, set $N_{cand} = N_{max}$.
2. Estimate the performance interval of each node in the net.
3. Identify node n having the largest lower bound on performance improvement, set $N_{cand} = N_{cand} \cup \{n\}$.
4. Add every node p whose performance interval intersects with that of node n into N_{cand}.

}

The N_{cand} so constructed contains all candidate nodes for link insertion that can possibly achieve the best improvement in net performance. An alternative way to construct N_{cand} at Step 4 is to compare the performance specifications of the net P_{sp}s with the performance intervals and add each node whose intervals contain P_{sp} to N_{cand}. This approach is valid if P_{sp}s are achievable by a single inserted link.

2.4.2 Link Insertion and Wiresizing

Although the choice of node for link insertion dictates the upper bounds on possible performance improvements by the link, the actual route and wire width of the inserted link determines its routing area and the actual reduction in maximum delay and skew of the net. Therefore, appropriate link insertion and wiresizing is critical for constrained optimization, which aims at achieving satisfactory net performance with minimum routing area consumption.

2.4.2.1 Problem Formulation

Suppose link e_n, inserted between n_{ref} and node $n \in N$, consists of k wire segments, each having its distinct length and width in a certain routing region on the route of e_n. Define $\mathbf{l}_{e_n} = (l_1, \ldots, l_k)$ and $\mathbf{w}_{e_n} = (w_1, \ldots, w_k)$ as the length and width vector of e_n, $(\mathbf{w}_{e_n})_{lb}$ and $(\mathbf{w}_{e_n})_{ub}$ as the vectors of lower and upper bounds on $\mathbf{w}_{e_n}$ respectively, i.e., $(\mathbf{w}_{e_n})_{lb} \leq \mathbf{w}_{e_n} \leq (\mathbf{w}_{e_n})_{ub}$. Here, $(\mathbf{w}_{e_n})_{lb}$ is usually set to the minimum wire width w_0 allowed for any interconnect wire on the chip, $(\mathbf{w}_{e_n})_{ub}$ can be estimated based on the routing soruce available in the regions along the route of e_n. The R, C values of each segment $i \in e_n$, R_i, C_i, are functions of its length l_i and width w_i, and can be expressed as:

$$R_i = r_o \frac{l_i}{w_i}, \quad C_i = c_0 l_i w_i \tag{2.15}$$

where r_0, c_0 are the unit resistance and capacitance of the wire under minimum width w_0, respectively. Notice that Eqn (2.15) is only the most commonly used expression for R_i and C_i, more sophisticated expressions for R_i and C_i can also be used in place of Eqn (2.15) without affecting the validity of the entire optimization approach. According to Eqn (2.15), the total R, C values of link e_n, R_{e_n} and C_{e_n}, can be computed by:

$$R_e = r_0 \sum_{i=1}^{k} \frac{l_i}{w_i}, \quad C_e = c_0 \sum_{i=1}^{k} l_i w_i \tag{2.16}$$

For constrained performance optimization, we measure the performance of the net by K, which is the relative difference between the maximum delay (skew) and its corresponding constraints, i.e.,

$$K_P = (P - P_{sp})/P_{sp} \tag{2.17}$$

where P is the current maximum delay or skew of the net, P_{sp} is its specified constraint. Define $K_{max} = max\{K_{delay}, K_{skew}\}$. According to the definition in Eqn (2.17), $K_{max} \leq 0$ implies that the performance requirements for the maximum delay and skew of the net are both satisfied, i.e., the performance of the net is satisfactory.

2.4.2.2 Link Establishment

Eqn (2.6) and Eqn (2.8) in Sec. 2.3 indicate that both the maximum delay and skew reduction favor the inserted link e_n to have small R_{e_n} and C_{e_n} which are both proportional to the length of l_{e_n} according to Eqn (2.16). Plus, inserting a short link is consistent with routing area minimization. Thus, link e_n should be established between node n_{ref} and n with the shortest feasible length in order to achieve the best result during constrained optimization.

The shortest possible length of e_n, $(l_{e_n})_{min}$, is equal to the Manhattan distance between the two nodes. However, the actual feasible length of e_n, l_{e_n}, may be longer than $(l_{e_n})_{min}$ due to the possible detours of e_n to avoid congested routing areas on the chip. In addition, the routing space available in the routing regions along the route of e_n, which determines the upper bound on wire width $(\mathrm{w}_{e_n}))_{ub}$, should also be considered when e_n is routed since w_{e_n} also affects the performance of the net. Therefore, a shortest path algorithm should be adopted for the establishment of link e_n between n_{ref} and n, which includes the routing congestions of the chip in its cost function for routing.

2.4.2.3 Non-Uniform Link Wiresizing

Once the route of link e_n is determined, its length vector $\mathbf{l}_{e_n}$ is fixed and its width vector $\mathbf{w}_{e_n}$ becomes the only adjustable variable during performance optimization, i.e, the maximum delay and delay skew of the net both become functions of $\mathbf{w}_{e_n}$. According to Eqn (2.6) and Eqn (2.8), the maximum delay and skew of the net after link insertion at node n can be expressed as:

$$\begin{aligned}(D_{max}(\mathbf{w}_{e_n}))_n &= D_{max} - \frac{R_{n_{max}n}}{R_{nn} + R_{e_n}(\mathbf{w}_{e_n})}(D_{max} - D_{e_{max}}(\mathbf{w}_{e_n})) \\ &\quad + R_d C_{e_n}(\mathbf{w}_{e_n}) \qquad (2.18)\\ (DS_{max}(\mathbf{w}_{e_n}))_n &= DS_{max} - \frac{R_{n_{max}n} - R_{n_{min}n}}{R_{nn} + R_{e_n}(\mathbf{w}_{e_n})}(D_{max} - D_{e_{max}}(\mathbf{w}_{e_n})) \qquad (2.19)\end{aligned}$$

Unlike in the case of $\mathbf{l}_{e_n}$, $C_{e_n}(\mathbf{w}_{e_n})$ is proportional to $\mathbf{w}_{e_n}$ while $R_{e_n}(\mathbf{w}_{e_n})$ is inversely proportional to $\mathbf{w}_{e_n}$ according to Eqn (2.16). Thus, both $(D_{max}(\mathbf{w}_{e_n}))_n$ and $(DS_{max}(\mathbf{w}_{e_n}))_n$ are non-monotonic, non-convex and non-posynomial with respect to $\mathbf{w}_{e_n}$ and the wiresizing of the inserted link should be formulated as a constrained optimization process which can be stated as follows:

Determine the optimal wire width vector of the inserted link s.t. the performance of the net satisfies its specified requirements and the total routing area of the link is minimized.

The routing area of the inserted link e_n is defined as the product between its length and width vectors, which is a function of $\mathbf{w}_{e_n}$ during the wiresizing:

$$Area_{e_n}(\mathbf{w}_{e_n}) = \mathbf{l}_{e_n}\mathbf{w}_{e_n} \qquad (2.20)$$

Denote $D_{max_{sp}}$ and $DS_{max_{sp}}$ as the specified constraints for the maximum delay and skew of the net respectively, the constrained optimization problem can then be formulated as:

Minimize $\quad Area_{e_n}(\mathbf{w}_{e_n})$

Subject to:

$$\begin{aligned}(D_{max}(\mathbf{w}_{e_n}))_n &\leq D_{max_{sp}}\\ (DS_{max}(\mathbf{w}_{e_n}))_n &\leq DS_{max_{sp}}\\ (\mathbf{w}_{e_n})_{lb} \leq \mathbf{w}_{e_n} &\leq (\mathbf{w}_{e_n})_{ub}\end{aligned}$$

In actual implementation, only one of $(D_{max}(\mathbf{w}_{e_n}))_n$ and $(DS_{max}(\mathbf{w}_{e_n}))_n$ needs to be considered as active constraint during the optimization since the maximum delay and skew reduction are consistent according to previous analysis.

2.4.2.4 Wiresizing Analysis

The formulation of the inserted link as a sequence of wire segments routed through various routing regions on the chip allows non-uniform wiresizing of the link, i.e., each wire segment of the link may have its distinct wire width. Non-uniform wiresizing provides extra flexibility in optimization compared to uniform wiresizing which enforces same wire width for the entire link (the later is actually a special case of the former), and its advantages can be summarized as follows:

- The delay introduced by the inserted link e_n is influenced not only by the values of its wire resistance and capacitance, R_{e_n} and C_{e_n}, but also by the way they are distributed along the link. Under fixed link route and routing area, the distribution of R_{e_n} and C_{e_n} are determined by the wire width vector $\mathbf{w}_{e_n}$ which can be adjusted via wiresizing of the link. Intuitively, large capacitive loads should be placed close to the source so that they will only be charged through small amount of wire resistance and contribute little to the delay of the net. Therefore, non-uniform wiresizing may result in smaller delay introduced by the inserted link and better improvement in net performance compared with uniform sizing.

- The upper bound on the wire width of the link, $(\mathbf{w}_{e_n})_{ub}$, is determined by the routing resources available along the route of the link in the current routing solution. Due to the difference in routing congestions in different regions on the chip, the allowance for wiresizing of each segment of the link may vary significantly. Unlike uniform sizing, which require an identical wire width for the entire link that can be to be no more than the minimum width upper bound along the route, non-uniform wiresizing is better suited to handle the various upper bounds situation by allowing $\mathbf{w}_{e_n}$ to follow $(\mathbf{w}_{e_n})_{ub}$ closely in the routing regions of e_n. Therefore, it may yield a wiresizing solution of the link having smaller R_{e_n} and achieving better improvement in net performance than uniform sizing.

From the analysis above, the following conclusion can be drawn about the wiresizing of the inserted link:

Proposition 2.1 *For the optimization of an inserted link, non-uniform wiresizing can achieve better improvement in net performance compared to uniform sizing with the same routing area consumption.*

Analogous to the proof of Corollary 2.3, two curves, Cur_{non} and Cur_{uni}, can be plotted representing the trade-offs between improvement in net performance and the link routing area under

the uniform and non-uniform wiresizing of a link to node n, respectively. Since Cur_{non} is "lower" than Cur_{uni} according to Proposition 2.1, the following corollary can be established similar to Corollary 2.3.

Corollary 2.4 *For the same amount of reduction in maximum delay and skew of the net, the non-uniform wiresizing of an inserted link consumes less routing area than the uniform wiresizing.*

2.4.2.5 Sequential Quadratic Programming (SQP)

Due to the non-convex nature of $(D_{max}(\mathbf{w}_{e_n}))_n$ and $(DS_{max}(\mathbf{w}_{e_n}))_n$ as functions of $\mathbf{w}_{e_n}$ according to Eqn (2.18) and (2.19), the link wiresizing process formulated in Sec. 2.4.2.3 is a non-linear programming problem, which is solved in our approach using the Sequential Quadratic Programming (SQP) method also known as "Constrained Quasi-Newton Method". SQP is an iterative procedure which attempts to solve the Kuhn-Tucker (KT) necessary condition for optimality of the problem using the quasi-Newton method for non-linear equations. At every iteration, a Quadratic Programming (QP) sub-problem is formulated based on an approximation made of the Hessian of the Lagrangian function, which solution is then used to establish a new search direction for the optimization. The basic idea of SQP can be described as follows:

Suppose that the constrained optimization problem is formulated in general as:

$$\begin{aligned} &\textit{Minimize} && f(x) \\ &\textit{Subject to:} && g_i(x) \leq 0, i = 1, \ldots, m \end{aligned}$$

Denote $L(x, \lambda) = f(x) + \sum_i \lambda_i g_i(x)$ as its Lagrangian function with λ_is as its Lagrangian multipliers, its KT equation can then be expressed as:

$$h(x, \lambda) = \partial L/\partial x = f_x + g_x^T \lambda \;=\; 0 \tag{2.21}$$

where $f_x = \partial f/\partial x, g_x = [\partial g_i(x)/\partial x], \lambda = [\lambda_i]$.
Applying quasi-Newton method to solve the non-linear equations, we obtain:

$$\begin{aligned} f_x + B\Delta x + g_x^T \lambda' &= 0 \\ g(x) + g_x \Delta x &\leq 0 \end{aligned} \tag{2.22}$$

where $\lambda' = \lambda + \Delta\lambda$ and B approximates the Jacobian of h (Hessian of L) by $f_{xx} + g_{xx}^T \lambda$.

Since Eqn (2.22) is the KT equation of the following QP problem with the quadratic approximation of L as its objective:

$$\textit{minimize} \qquad f(x) + f_x^T \Delta x + 1/2 \Delta x^T B \Delta x \tag{2.23}$$

$$s.t. \qquad g_i(x) + g_x \Delta x \leq 0$$

A QP sub-problem of Eqn (2.23) can be solved at each iteration during the optimization process for constrained optimization mimicing Newton's method. The solution of the QP problem is then used to form a new iterate

$$x_{k+1} = x_k + \alpha_k \Delta x$$

where α_k is determined by a line search procedure so that a sufficient decrease in a merit function is obtained [Brayton 81]. B approximating the Hessian matrix is updated at each iteration by any quasi-Newton method of update. Analysis [Brayton 81] shows that SQP method can guarantee super linear convergence by accumulating second order information regarding KT equations and is efficient for solving non-linear optimization problems.

2.4.3 Single Link Insertion and Wiresizing

Based on the previous analysis on node choice for link insertion and approach for link construction, a single link insertion and wiresizing algorithm is designed. It establishes a link between n_{ref} and the node chosen for best performance improvement with the shortest feasible length and wiresizes the link using a constrained optimization approach. The objective is to achieve satisfactory net performance with minimum link routing area. The constrained optimization of the inserted link terminates when the performance requirements of the net are satisfied, i.e., $K_{max} = max(K_{delay}, K_{skew}) \leq 0$, or no further improvement in net performance can be obtained.
Single Link Insertion and Wiresizing Algorithm{

1. Initialization:

   1.1 Input critical net topology N and its specified performance constraints P_{sp}s on maximum delay and skew.

   1.2 Calculate the node delays, identify maximum delay node(s) and initial performance gap K_{max} before optimization.

   1.3 Estimate the lengths and wire width upper bounds of the shortest feasible links from n_{ref} to nodes in N under current routing conditions of the chip (Sec. 2.4.2.2).

2. If assumptions A1 and A2 in Situation I are satisfied:

   2.1 Establish link $e_{n_{max}}$ to maximum delay node n_{max}.

2.2 Wiresize $e_{n_{max}}$ for performance improvement (Sec. 2.4.2.3).

3. Else (Situation II):

3.1 Construct candidate node set N_{cand} for link insertion. (Sec. 2.4.1.3).

3.2 Estimate the routing area of link to each node n in N_{cand} under constrained optimization.

3.3 Choose the link that can achieve satisfactory performance with minimum routing area.

}

2.5 Multi-Link Insertion and Wiresizing

The problem of node choice for link insertion discussed in Sec. 2.4.1 is based on the original net topology and it implicitly assumes that the maximum delay node(s) do not change during the optimization. Since insertion and wiresizing of link e_n to the current node choice n results in different amount of reductions in delays and skews at different nodes (largest at the maximum delay nodes according to our node choice method), the maximum delay node(s) as well as the ratio between nodes delays may change. When this happens, continued wiresizing of link e_n may no longer be able to improve the net performance with the best link area efficiency and some other node n' in the topology may emerge as the new best node choice for link insertion and wiresizing. Due to this possible change in best node choice during the constrained optimization process, and the fact that optimizing multiple links can achieve better performance improvement than optimizing a single link only, multi-link insertion and wiresizing algorithms are designed, whose objective is similar to that of the single link optimization:

Insert and wiresize multiple links to nodes in critical net topology N s.t. the performance requirements of the net are satisfied with minimum total link routing area consumption under routing resource constraints.

2.5.1 Sequential Link Insertion and Wiresizing

We first discuss a sequential link insertion and wiresizing algorithm, which always inserts and wiresizes a link to the current best node choice for performance improvement during the optimization process. Initially, a link e_n is established to node n which is chosen based on the original net topology, node delays and estimations on feasible link lengths before optimization. Then, link e_n is wiresized to satisfy performance constraints reduced gradually from the original

D_{max} (DS_{max}) to $D_{max_{sp}}$ ($DS_{max_{sp}}$) under the wiresizing formulation described in Sec. 2.4.2.3. This process continues until the best node choice for link insertion and wiresizing changes from n to n' based on the estimation of current node delays. When this happens, the optimization switches from sizing e_n to the optimization of link $e_{n'}$ between n_{ref} and n' instead. This sequential link optimization process continues until the net performance requirements are satisfied (i.e., $K_{max} \leq 0$) at any step in the algorithm.

Sequential Link Insertion and Wiresizing Algorithm {

1. Initialization:
   Same as Step 1 in the single link algorithm.
2. Choose the best node n for link insertion.
3. While performance requirements of the net are not satisfied ($K_{max} > 0$):
   - 3.1 Establish link e_n between n_{ref} and node n.
   - 3.2 Wiresize link e_n to satisfy a gradually reduced performance constraints (Sec. 2.4.2.3) until a different best node choice emerges.
   - 3.3 Identify the new best node choice n'.
   - 3.4 If $e_{n'}$ between n_{ref} and n' already exists:
     Set $n = n'$, goto Step 3.2 to wiresize $e_{n'}$ instead.
   - 3.5 Otherwise,
     - 3.5.1 Estimate the routing area of link $e_{n'}$ having minimum wire width, $Area_{e'_n}(w_0)$.
     - 3.5.2 If continued sizing of e_n can not achieve $K_{max} \leq 0$ or it consumes more additional area than $Area_{e'_n}(w_0)$:
       Set $n = n'$, goto Step 3.1 to establish $e_{n'}$.
     - 3.5.3 Otherwise,
       goto Step 3.2 and continue sizing e_n.

}

There are several points worth mentioning about this sequential link optimization algorithm:

1. Since same node may repeatedly become the best node choice during the optimization process, link $e_{n'}$ between n_{ref} and n' may already exist when n' is identified at Step 3.3. In that case, the algorithm simply switches to wiresize $e_{n'}$ instead of e_n (Step 3.4).

2. The establishment of a new link consumes certain amount of link area $Area_{e_{n'}}(w_0)$, even under minimum wire width w_0. Therefore, continued wiresizing of the current link e_n is preferred if it can lead to satisfactory net performance with less additional routing area assumption than $Area_{e'_n}(w_0)$ (Step 3.5.3). Otherwise, link $e_{n'}$ to n' is established and wiresized in place of e_n (Step 3.5.2).

2.5.2 Optimal Multi-Link Insertion and Wiresizing

The sequential link insertion and wiresizing algorithm described above is greedy in nature, since it always inserts and wiresizes a link that is best for performance improvement at the current step during the optimization process. Thus, the final solution it generates may not be the global optimal one due to the following reasons:

- The net performance improvement achievable by links to different nodes are not independent of each other. Therefore, a best node choice by the sequential algorithm at the current optimization step may not the best for a global optimal solution which includes a set of inserted links. In other words, it may preclude further performance improvement and savings in subsequent steps of link insertion and result in local optimality.

- A definitive node choice for link insertion may not always exist due to possible ties among multiple node choices that could lead to equal performance improvement and link area consumption at certain step in the optimization process. In that case, any arbitrary tie-breaking strategy may cause the sub-optimality of the final solution. For optimal results, each link to one of these tied nodes should be tried and compared.

In order to generate a multi-link solution which achieves satisfactory performance with minimum total link routing area, we design an optimal multi-link insertion and wiresizing algorithm. It adopts a recursive Branch-and-Bound approach, which at each step during the optimization, considers a set of candidate nodes N_{cand} (constructed similar to the algorithm in Sec. 2.4.1.3) for possible link insertion and wiresizing. During the Depth-First-Search process, if the optimization of the current link e_n to a node n in N_{cand} results in larger total link routing area than the best solution obtained so far that achieves satisfactory net performance, link e_n is pruned and the search backtracks to the previous step to try another node in N_{cand} for link insertion. This Branch-and-Bound method is applicable to link insertion and wiresizing in searching for an optimal solution since the total link routing area increases monotonically with the insertion and wiresizing of links

into the existing topology. The entire algorithm is outlined below in which $size\_sol(e_n)$ is denoted as the current wiresizing solution of link e_n.

Optimal Link Insertion and Wiresizing Algorithm {

1. Initialization:
   Same as Step 1 in the single link algorithm.

2. Denote the initial routing topology as N_0, call *Add-Link(N_0)*.

}

Add-Link(N) {

1. Identify the candidate node set N_{cand} for link insertion (Sec. 2.4.1.3).

2. For each node n in N_{cand}:

   2.1 Establish link e_n between n_{ref} and node n.

   2.2 Wiresize link e_n to n to satisfy a gradually reduced performance constraints (Sec. 2.4.2.3) until maximum delay node(s) changes or performance requirements for the ent are satisfied.

   2.3 If current total routing area of topology N, $Area(N) \cup size\_sol(e_n)$, does not exceed the best solution, $Area(N_{opt})$, obtained so far:

      2.3.1 If performance requirements are satisfied:
      $N_{opt} = N \cup \{sizing\_sol(e_n)\}$.

      2.3.2 Else
      Call Add-Link($N \cup \{sizing\_sol(e_n)\}$).

   2.4 Prune e_n, restore topology N and node delays to the values before e_n is introduced.

}

Notice that a node n may repeatedly appear in N_{cand}s along a path in the Branch-and-Bound searching tree from root to the leaf. Its first appearance leads to the insertion of link e_n to n, while its later appearances cause the continuous wiresizing of e_n. The size of the candidate node set $|N_{cand}|$ at each iteration is small since only links to a few nodes in the topology may have significant impact on the performance improvement of the net according to the analysis in Sec. 2.4.1. Therefore, the optimal multi-link insertion and wiresizing algorithm is feasible in real

Table 2.1: Interconnect parameters

Technology	$R_d(\Omega)$	$r_0(\Omega/\mu m)$	$c_0(fF/\mu m)$	$C_s(fF)$
IC	270	0.112	0.039	1.0
MCM	25	0.008	0.06	1000

applications as shown by our experimental results. In cases where very fast solution is needed, either the single link (Sec. 2.4) or the sequential link insertion and wiresizing algorithm (Sec. 2.5.1) can be used instead.

2.6 Experimental Results

The link insertion and wiresizing algorithms for post routing performance optimization have been implemented and tested on a DEC 5000/125 workstation. Since interconnects are typically modeled as distributed RC lines under deep-submicron IC and certain types of MCM technologies, our methods are tested under both 0.5 micron IC (courtesy of Micro-electronics Center of North Carolina) and MCM technologies (courtesy of Prof. Wayne Dai of UC Santa Cruz from data provided by AT&T) respectively (Table 2.1). Here R_d is the driver resistance of the net, r_0, c_0 are unit resistance and capacitance of the wire with minimum width w_0, C_s is the loading capacitance at each sink.

2.6.1 An Example

We illustrate our link insertion and wiresizing methods using a 8-pin net under MCM technology, whose original topology before optimization is shown in Fig. 2.10. The response waveforms at the maximum delay node n_{max} and minimum delay node n_{min} before optimization obtained by SPICE3f5 simulation are shown in Fig. 2.15 (a). It can be observed that the rising portion of the un-optimized waveform at n_{max} is not "sharp" and there exists a large gap between waveforms at n_{max} and n_{min}, indicating a significant maximum delay and delay skew. For satisfactory chip performance, the performance specification of the net requires that its D_{max} and DS_{max} be reduced by 50% and 70%, respectively.

We first show the optimization results by the single link insertion and wiresizing algorithm. It chooses node n for link establishment based on the original routing topology and wiresizes it either uniformly, i.e., the entire link must have the same width, or non-uniformly, i.e., each wire

segment of the link is allowed to have its distinct wire width in its routing region. The optimization solutions are shown in Fig. 2.11. Notice that the length of the inserted link e_n is longer than the Manhattan distance between the source and n due to possible detours under routing congestions in the current routing solution of the chip. The wire width of the inserted link is measured in terms of the minimum wire width w_0 and its upper bound w_{ub} is set to $4x w_0$ in our testing. In both sizing solutions, the net performance is improved significantly, and the non-uniform wiresizing achieves better performance improvement (47% vs. 45% in D_{max}, 69% vs. 68% in DS_{max}) and less link area ($20mmxw_0$ vs. $24mmxw_0$) than uniform wiresizing. However, the best results by single link methods fail to satisfy the performance requirements of the net.

To achieve satisfactory net performance, we apply the multi-link insertion and wiresizing algorithm (both sequential and optimal methods yield the same solution for this example). First, link e_n is established to the chosen node n and wiresized gradually until maximum delay node changes and a new node n' emerges as the new best node choice for link insertion (Fig. 2.12(a)). Since continued wiresizing of link e_n alone can not achieve satisfactory net performance, we stop optimizing e_n and establish the second link $e_{n'}$ between n_{ref} and n' instead. Link $e_{n'}$ is then wiresized until the performance requirements of the net are satisfied. The final solution shown in Fig. 2.12 (b) consumes a total link routing area of $16mmxw_0$. Therefore, the multi-link insertion and wiresizing algorithm can achieve better performance with less link area consumption compared to single link optimization.

Fig. 2.13 shows the area-performance trade-offs between the maximum delay (skew) of the net and the total link area during the optimization by single link uniform sizing, single link non-uniform sizing and the multi-link approaches, respectively. Here, both D'_{max} and DS'_{max} are measured as the percentages of their original values before optimization. Point p_1 in the two figures marks the insertion of link e_n into the topology. Initially, the curves by single link non-uniform sizing and the multi-link approach overlap, since both methods are optimizing the same link. At point p_2, another link $e_{n'}$ is introduced by the multi-link approach, which speeds up the reduction in both the maximum delay and skew of the net. It can be observed from Fig. 2.13 that the multi-link method achieves the best improvement in performance with the least amount of total link area among the three approaches.

Fig. 2.14 and 2.15 compare the response waveforms at the maximum and minimum delay nodes of the net before and after optimization respectively using SPICE simulation. It can been seen that the waveform at n_{max} rises much faster and the "gap" between the waveforms at n_{max} and n_{min} is "narrowed" significantly after optimization, indicating significant reduction in both the

maximum delay and delay skew of the net.

An important concern in adopting Elmore delay as the performance measure during the optimization process is its "fidelity" in guiding the optimal routing topology construction. To verify this, we compare the percentages of reductions in D_{max} and DS_{max} of the net measured by Elmore delay and SPICE simulation (measured at 50% voltage threshold), respectively. The difference between the corresponding ratios is only $1-2\%$, indicating that Elmore delay is highly reliable for guiding the optimization of distributed RC line networks. Similar conclusions about the fidelity of Elmore delay for performance-driven routing have also been made in [Boese 93].

2.6.2 Benchmark Testing

For post routing performance optimization, the optimal multi-link insertion and wiresizing method has been applied to topologies of critical nets in actual circuits after their global routing solutions are obtained. Four test circuits from the CBL/NCSU building-block benchmarks, $ami33, xerox, hp$ and $spert$ are used, the first three are IC circuits and the last one is a MCM circuit. The placement/global routing solution of these chips are generated by a performance-driven placement[Esbensen 96] and global router[Wang 96] respectively. Once their global routing solutions become available, the critical nets in these chips can be identified and their performance requirements in terms of their maximum delay and skew specifications are known. In addition, the routes, lengths and upper bounds on wire widths of the shortest feasible links to nodes in those topologies from their sources can be estimated based on the current routing solutions of the chips. With these information, the optimal link insertion and wiresizing algorithm described in Sec. 2.5.2 can be applied.

Table 2.2 - 2.5 show the testing results on critical nets from circuit $ami33, xerox, hp$ and $spert$, respectively. The average number of pins of these critical nets varies from 9 to 44. The total wire length of each topology is measured in terms of the number of units it crosses on the global routing graphs of these chips and the area of each inserted link is measured in unit x w_0. The performance requirements for maximum delay and skew are so specified that the performance of these critical nets can be significantly improved by inserting no more than two links into their original topologies. Recall that K_{max} is the relative difference between the current net performance and its corresponding specification. It can be observed that K_{max} is pretty large before optimization (whose average is in the range from 1.0 to 1.9), indicating that the original net performance is far from being satisfactory. After multi-link insertion and wiresizing, satisfactory performance

($K_{max} \leq 0$) is achieved for all critical nets with no more than two links inserted. On average, D_{max} and DS_{max} are reduced by 29% and 54% respectively for IC circuits; the reduction is even larger for MCM circuit at 50% and 60% respectively. The total link routing area introduced into each topology counts for 30% to 90% of its original net routing area, i.e., less than double sizing the original net in the worst case, which is moderate compared to other wiresizing approaches. Notice that the performance improvement listed in these tables are not the best achievable, since the constrained performance optimization terminates for link routing area efficiency as soon as the performance requirements of the nets are satisfied. For example, if the optimization of net $RXA8$ in circuit *spert* continues, D_{max} and DS_{max} can be further reduced to 51% and 59% respectively, with a higher link area utilization at 1486. On the other hand, the performance improvement in net $RXA8$ by two inserted links is bounded, which can not reach 60% and 70% reduction in maximum delay and skew respectively if the performance requirements of the net are so specified.

2.7 Conclusions

This chapter discusses the post routing performance optimization of distributed RC line topologies, which unlike previous methods for performance-driven and clock routing at the pre-routing level, achieves satisfactory chip performance by improving the performance of those critical nets via constrained optimization after a feasible routing solution of the chip is obtained. The basic approach for post routing optimization is link insertion and wiresizing, which can reduce the maximum delay and skew of a net simultaneously to satisfy their specified performance constraints by introducing new interconnect wires into its topology. This is accomplished without invalidating the current routing solution of the chip, therefore the proposed approaches can speed up the chip design cycle by avoiding the time-consuming iterations in the layout process. In addition, our method no longer restricts routing topologies to tree structures and thus allows more flexibilities in routing. Both single and multi-link insertion and wiresizing algorithms are analyzed and designed, which aim at achieving satisfactory net performance with the best link area efficiency. Experimental results on critical nets in actual routed circuits demonstrate that our approaches can achieve significant reduction in both the maximum delay and delay skew of all critical nets tested with moderate link routing area consumption.

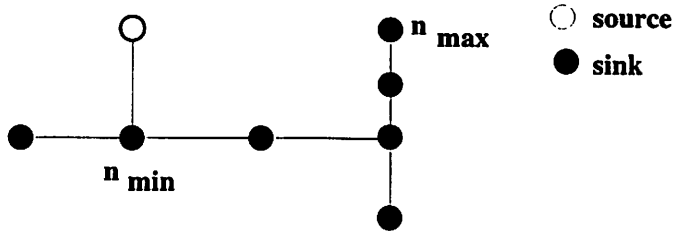

Figure 2.10: Original topology before optimization

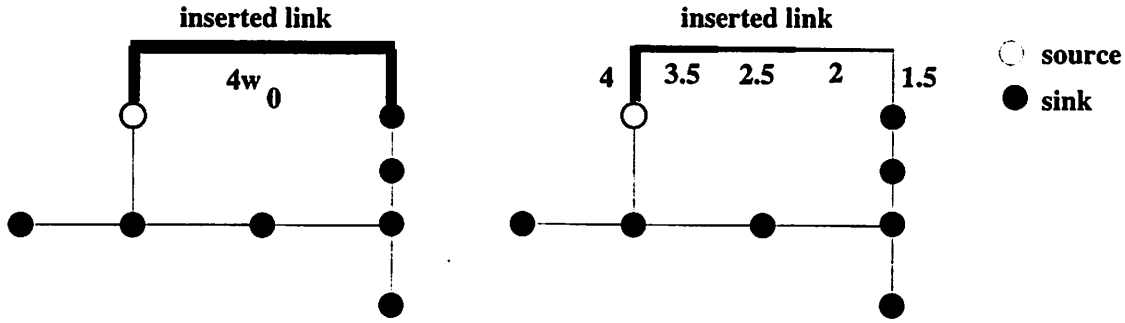

Figure 2.11: Single link: (a) Uniform Sizing (b) Non-Uniform sizing

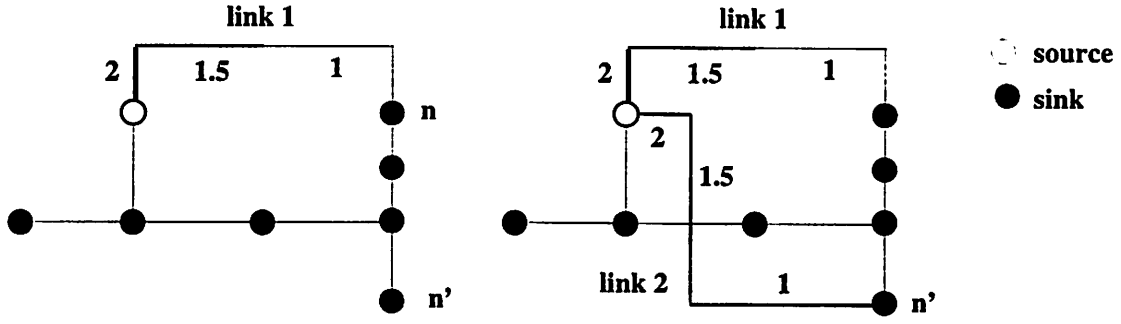

Figure 2.12: (a) Multi-link: Link One (b) Multi-link: Link Two

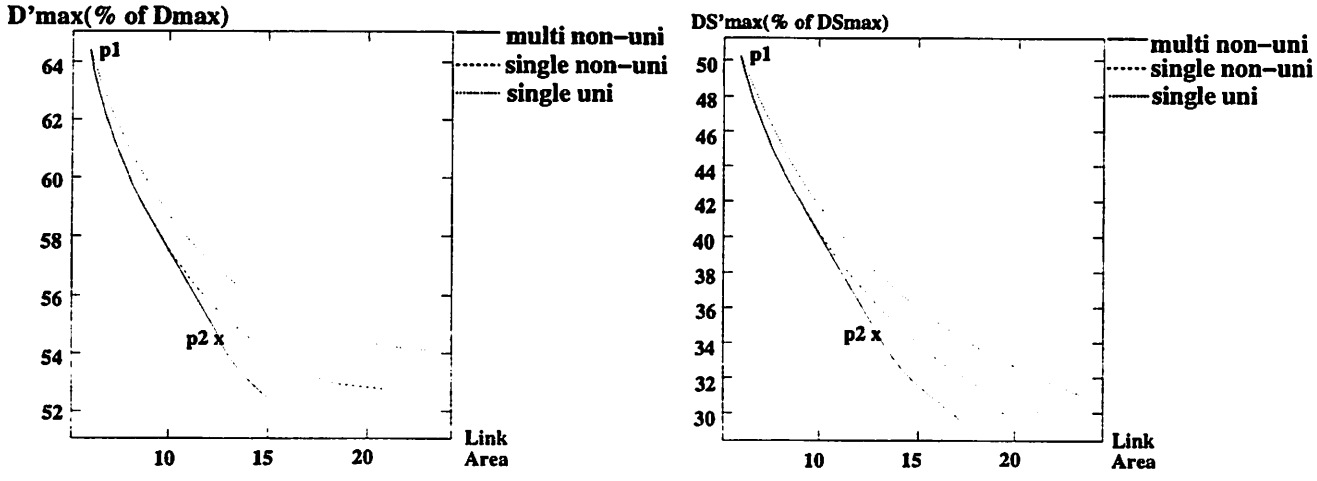

Figure 2.13: (a) Maximum delay vs. Link area (b) Maximum skew vs. Link area

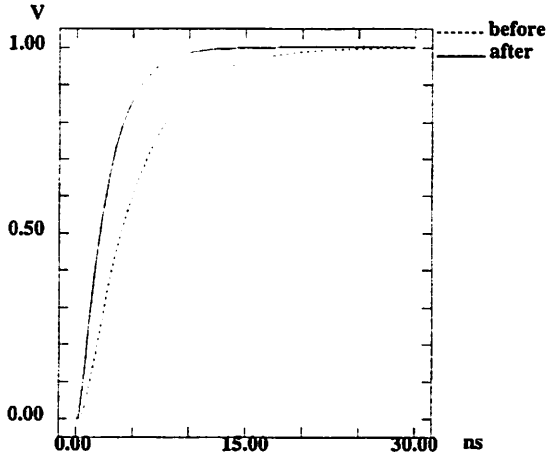

Figure 2.14: Maximum Delay Reduction

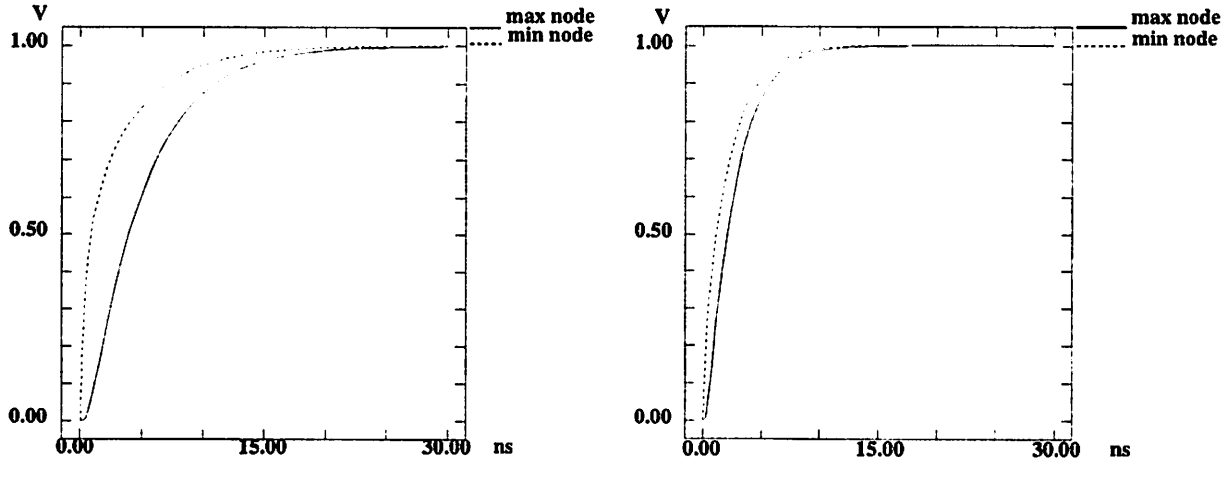

Figure 2.15: Maximum Skew Reduction (a) Before (b) After Optimization

Table 2.2: Test Circuit *ami33* (IC)

Crit. Net	No. Pins	Total Length	Spec. (-%)		Before	After Optimization			
			D_{max}	DS_{max}	K_{max}	D_{max}(-%)	DS_{max}(-%)	K_{max}	Link Area
P2F	47	1760	32	50	1.0	34	51	-0.01	370
P1F	44	1431	28	52	1.1	30	53	-0.02	710
P1G	50	1541	24	54	1.2	26	55	-0.01	534
POW	35	1540	38	66	1.9	38	66	0.00	340
avg	44	1578	30	55	1.3	32	56	-0.01	488

Table 2.3: Test Circuit *xerox* (IC)

Crit. Net	No. Pins	Total Length	Spec. (-%)		Before	After Optimization			
			D_{max}	DS_{max}	K_{max}	D_{max}(-%)	DS_{max}(-%)	K_{max}	Link Area
PH1	72	528	40	60	1.5	45	60	-0.01	240
VD1	20	495	40	60	1.5	42	60	-0.01	331
PH2	28	448	6	30	0.4	7	32	-0.02	284
NPH2	20	377	15	46	0.7	15	48	-0.02	344
avg	35	462	25	50	1.0	27	50	-0.02	300

Table 2.4: Test Circuit *hp* (IC)

Crit. Net	No. Pins	Total Length	Spec. (-%)		Before	After Optimization			
			D_{max}	DS_{max}	K_{max}	D_{max}(-%)	DS_{max}(-%)	K_{max}	Link Area
vd1	14	219	20	50	1.0	24	51	-0.02	219
mk1	8	189	35	60	1.5	37	60	-0.02	147
busa2	7	196	15	50	1.0	15	52	-0.01	200
busa1	7	161	30	60	1.5	30	60	0.00	194
avg	9	192	25	55	1.3	27	56	-0.01	190

Table 2.5: Test Circuit *spert* (MCM)

Crit. Net	No. Pins	Total Length	Spec. (-%)		Before	After Optimization			
			D_{max}	DS_{max}	K_{max}	D_{max}(-%)	DS_{max}(-%)	K_{max}	Link Area
RYA11	9	510	56	70	2.3	59	70	-0.01	512
RXA8	9	593	46	54	1.5	48	55	-0.02	607
RXA6	9	642	50	60	2.6	50	60	0.00	620
RXA20	9	517	55	64	1.9	57	65	-0.02	387
RXAW	9	533	45	52	1.1	45	52	0.00	471
avg	9	559	50	60	1.9	52	60	-0.01	519

Bibliography

[Brayton 81] R. Brayton, G. Hachtel, A. Sangiovanni-Vincentelli, "A Survey of optimization Techniques for Intergrated-Circuit Design", *Proc. of IEEE*, Vol. 69, pp. 1334-1362, 1981.

[Boese 93] K. Boese, A. Kahng, B. Mccoy and G. Robins, "Fidelity and Near-Optimality of Elmore-Based Routing Constructions", *Proc. IEEE Intl. Conf. Computer Design*, pp. 81-84, 1993.

[Boese 94] K. Boese, A. Kahng, B. Mccoy and G. Robins, "Rectilinear Steiner Trees with Minimum Elmore Delay", *Proc. 31st Design Automation Conf.*, pp. 381-386, 1994.

[Cong 92] J. Cong, A Kahng, G. Robins, M. Sarrafzadeh, C. Wong, "Provably Good Performance-Driven Global Routing", *IEEE Trans. Computer-Aided Design*, pp. 739-752, Nov. 1992.

[Cong 93a] J. Cong and K. Leung, "Optimal Wiresizing Under the Distributed Elmore Delay Model", *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design*, pp.634-639, 1993.

[Cong 93b] J. Cong and K. Leung and D. Zhou, "Performance-driven Interconnect Design Based on Distributed RC Delay Model", *Proc. 30th Design Automation Conf.*, pp. 606-611, 1993.

[Cong 95] J. Cong, C. Koh, "Minimum-Cost Bounded-Skew Clock Routing", *Proc. Int'l Symposium on Circuits and Systems*, 1995.

[Edahiro 91] M. Edahiro, "Minimum Skew and Minimum Path Length Routing in VLSI Layout Design", *NEC Research and Development*, pp. 569-575, 1991.

[Elmore 48] W.C.Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers", *J. Applied Physics*, 19.1, pp.55-63, 1948.

[Esbensen 96] H. Esbensen, E. Kuh, "An MCM/IC Timing-Driven Placement Algorithm Featuring Explicit Design Space Exploration", *Proc. IEEE Multi-Chip Module Conf.*, pp. 170-175, 1996.

[Gupta 95] R. Gupta, B. Krauter, B. Tutuianu, J. Willis, L. Pileggi, "The Elmore Delay as a Bound for RC Trees with Genralized Input Signals", *Proc. 32nd Design Automation Conf.*, pp.364-369, 1995.

[Hodes 94] T. Hodes, B. McCoy, G. Robins, "Dynamically-Wiresized Elmore-Based Routing Constructions", *IEEE Int'l Symp. Circuits and Systems*, Vol. I, pp. 463-466, 1994.

[Hong 93] X. Hong, T. Xue, E. Kuh, C. Cheng, J. Huang, "Performance-Driven Steiner Tree Algorithms For Global Routing", *Proc. 30th Design Automation Conf.*, pp.177-181, 1993.

[Huang 95] D. Huang, A Kahng, A. Tsao, "On the Bounded-Skew Clock and Steiner Routing problems", *Proc. 32nd Design Automation Conf.*, pp. 508-513, 1995.

[McCoy 94] B. McCoy and G. Robins, "Non-Tree Routing",*Proc. European Design & Testing*, pp. 430-434, 1994.

[Pullela 93] S. Pullela, N. Menezes, L. Pillage, "Reliable Non-Zero Skew Clock Trees Using Wire Width Optimization", *Proc. 30th Design Automation Conf.*, pp. 165-170, 1993.

[Robinstein 83] J. Robinstein, P. Penfield, M. Horowitz, "Signal Delay in RC Tree Networks", *IEEE Trans. Computer-Aided Design*, pp. 202-211, Feb. 1983.

[Ruehli 86] A. E. Ruehli, "Circuit Analysis, Simulation and Design", *Advances in CAD For VLSI*, Vol.3, 11.2, 1986.

[Sapatnekar 94] S. Sapatnekar, "RC Interconnect Optimization under the Elmore Delay Model", *Proc. 31st Design Automation Conf.*, pp. 387-391, 1994.

[Tsay 91] R. S. Tsay, "Exact Zero Skew", *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design*, pp. 336-339, 1991.

[Yu 95] Q. Yu, E. Kuh, " Exact Moment Matching Model of Transmission Lines and Application to Interconnect Delay Estimation", *IEEE Trans. on VLSI Systems, Vol.3, No. 2*, pp. 311-322, 1995.

[Xue 95a] T. Xue and E. Kuh, "Post Routing Performance Optimization via Tapered Link Insertion and Wiresizing", *Proc. European Design Automation Conf.* , pp. 74-79, 1995.

[Xue 95b] T. Xue and E. Kuh, "Post Routing Performance Optimization via Multi-Link Insertion and Non-Uniform Wiresizing", *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design*, pp. 575-580, 1995.

[Wang 96] D. Wang, E. Kuh, "Performance-Driven Interconnect Global Routing", *Proc. 6th Great Lakes Symp. on VLSI*, pp. 132-136, 1996.

[Zhu 93] Q. Zhu, W.M. Dai, J. Xi, "Optimal Sizing of High-Speed Clock Networks Based on Distributed RC and Lossy Transmission Line Models", *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design*, pp. 628-633, 1993.

Chapter 3

Post Routing Optimization of Lossy Transmission Lines

3.1 Introduction

The post routing performance optimization methods discussed in Chapter 2 are designed for interconnects modeled as distributed RC line topologies. Such modeling is appropriate for on-chip wires under deep sub-micron IC and some types of MCM technologies, where the resistance of the interconnect wire dominates its inductance and the transmission line effect can be ignored. The modeling of interconnects in other MCM and PCB circuits, however, is quite different. Due to their relatively large cross-sections, long interconnect lengths and large distances above ground planes, these off-chip wires usually have low resistances, large inductances and significant time-of-flight for signals traveling across the interconnects. Therefore, it is appropriate to model these wires as lossy transmission line (distributed RLC line) topologies, which beside wire resistances and capacitances, also take into account the inductances distributed along the interconnect wires.

Due to the strong inductive effect of lossy transmission line topologies, the estimation of their interconnect delays during performance optimization is difficult. Unlike in the case of distributed RC lines, Elmore delay[Elmore 48] is no longer accurate for guiding the optimization of transmission line topologies, since it does not incorporate the inductance of the wire in its computation. Several estimation methods other than Elmore delay have been used for delay estimation in recent years. [Wang 94, Zhu 93] adopt the S-parameter macro delay model for maximum delay and skew minimization, but the sensitivity computation in these methods uses finite

difference approximation which requires expensive analysis. [Menezes 94, Menezes 95] adopt high order moments for delay and skew minimization. The moment and sensitivity model they adopt during computation, however, requires decomposition of each wire into a sequence of segments, which is computationally inefficient. In addition, the methods in [Menezes 94, Menezes 95] can only be applied to distributed RC line topologies and they assume perfect shapes of output waveforms for delay and sensitivity computation which are unrealistic in real design.

As discussed in Chapter 2, the wiresizing of an existing topology is an effective approach to interconnect optimization and is especially well-suited for post routing performance optimization since the available resources in the routing regions can be incorporated easily as sizing constraints during the optimization. Methods in [Cong 93, Sapatnekar 94] can be applied to distributed RC line topologies only, because they adopt Elmore delay for performance estimation. For delay and skew optimization of lossy transmission line topology, [Menezes 94, Menezes 95, Zhu 93] adopt the Levenberg-Marquardt method which uses sensitivity information for least-square minimization, i.e., it minimizes the differences between delays at different sinks or between current and targeted delay values. These matching-based methods are better suited for generating zero-skew solutions rather than maximum delay minimization, since the later requires target delay specification for each sink of the topology. In addition, these approaches are inflexible for interconnect performance optimization since they strictly enforce the final delay and skew values allowed.

In this chapter, we present an analytical delay sensitivity computation method and a sensitivity-based wiresizing algorithm for lossy transmission line topologies [Xue 96a, Yu 96]. As a post routing performance optimization process, it minimizes the maximum delay of a lossy transmission line topology identified as critical for chip performance, using the routing resource still available on the chip. For interconnect delay estimation, it computes the maximum delay and its sensitivities with respect to the widths of wires in the topology via high order moments. For moment computation, an exact moment matching model[Yu 95, Yu 96] is adopted to represent each transmission line in the topology, whose parameters can be computed efficiently from the parasitics of the interconnect wires. Compared to other interconnect modeling methods[Bracken 92, Zhu 93] which are often computationally expensive, the model we adopted achieves analytical sensitivity computation and calculates higher order moments (sensitivities) recursively from lower order moments for tree networks. The obtained delay sensitivities are then used to guide the wiresizing of the topology for maximum delay minimization. This sensitivity-based wiresizing algorithm can achieve optimal interconnect performance under routing resource constraints. Experiments show that the delay approximation using high order moments is very accurate compared with SPICE sim-

ulation and the proposed algorithm can reduce maximum delay by average of over 60% with small penalties in routing area. Besides delay minimization, our approach also reduces the overshootings of response waveforms and generates final wiresizing solutions of topologies which are robust under parameter variations in the manufacturing process.

The rest of the chapter is organized as follows. Section 3.2 discusses the moment models of lumped circuit elements and lossy transmission lines; Section 3.3 analyzes the delay and sensitivity computation via high-order moments; Section 3.4 describes the sensitivity-based delay minimization algorithm; Section 3.5 shows experimental results which demonstrate the effectiveness of our approach; Finally, Section 3.6 gives concluding remarks.

3.2 Moment models

The methods for delay and sensitivity computation via higher order moments are based on the moment models of circuit elements, which include both the lumped elements and lossy transmission lines. Denote N as a lossy transmission line topology, consisting of floating resistors, inductors, lumped capacitors and lossy transmission lines. Denote $V(s)$ and $I(s)$ as the Laplace transform of the node voltage and current vectors of this linear circuit in the frequency domain, both can be expanded into Taylor series at $s = 0$ as,

$$V(s) = V^0 - V^1 s + V^2 s^2 + \ldots + (-1)^p V^p s^p + \ldots \tag{3.1}$$

and

$$I(s) = I^0 - I^1 s + I^2 s^2 + \ldots + (-1)^p I^p s^p + \ldots \tag{3.2}$$

Here, V^p and I^p are called the p-th order voltage and current moment vector, respectively. A circuit N^p induced from N by setting its voltage and current vectors to V^p and I^p is called a p-th order moment model of N.

A circuit model describing the relationship between the voltage and current moment of an element is called its moment model. A model relating a p-th order voltage moment with the j-th order current moments ($j \leq p$) or relating a p-th order current moment with the j-th order voltage moments ($j \leq p$) of a circuit element is called a p-th order moment model. The moment model for the lumped circuit elements and lossy transmission lines are discussed separately in the next two sections.

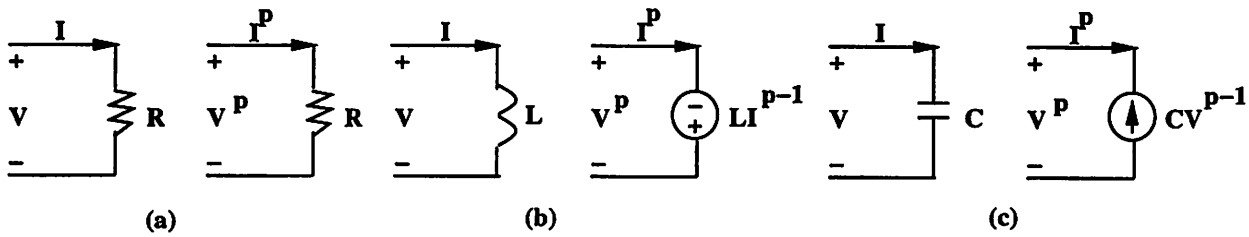

Figure 3.1: Moment Models of Lumped Circuit Elements

3.2.1 Moment model of lumped circuits

For 2-terminal lumped circuit elements: resistance R, inductance L and capacitance C, their corresponding moment models can be computed as follows:

1. For resistance R, $V^p = RI^p$, i.e., the p-th order moment model of a resistance R is itself, as shown in Fig. 3.1 (a).

2. For inductance L, $V^p = -LI^{p-1}$, i.e., its p-th order moment model is a voltage source with its direction opposite to that of the inductance current and its value determined by L and I^{p-1}, as shown in Fig. 3.1 (b).

3. For capacitance C, $I^p = -CV^{p-1}$, its p-th order moment model a current source with its direction opposite to that of the capacitance voltage and its value determined by C and V^{p-1}, as shown in Fig. 3.1 (c).

For the moment models of both capacitance and inductance, the moment computation can be implemented recursively from low orders up to high orders, where the p-th order moment model is an independent voltage (for inductance) and current source (for inductance), respectively. For a given lumped circuit, its p-th order moment model can be formed by replacing each element in the circuit with such model. Once all the p-th order moments of the node voltages and currents are found through circuit analysis, the $(p+1)$-th order moment model of the circuit can be formulated. In this way, the moment computation can be implemented recursively from order 0 to any order desired.

3.2.2 Moment model of lossy transmission line

Denote TL as a uniform lossy transmission line with r, l, c being its resistance, inductance and capacitance per unit length respectively. Define d as its wire length, thus, $R = rd$, $L = ld$ and

$C = cd$ become its total wire resistance, inductance and capacitance respectively under fixed wire width. Denote $V(x,s)$ and $I(x,s)$ as its line voltage and current moments at coordinate x along the line, where $x = 0$ and $x = d$ correspond to the two ends of the line, respectively. The Telegrapher's equations of the line can then be expressed as follows:

$$\frac{dV(x,s)}{dx} = -rI(x,s) - slI(x,s) \tag{3.3}$$

$$\frac{dI(x,s)}{dx} = -scV(x,s) \tag{3.4}$$

These two equations are in fact the KVL and KCL equations for an infinitesimal section of the line at coordinate x. Similar to Eqn (3.1) and (3.2), $V(x,s)$ and $I(x,s)$ can be expanded into Taylor series as:

$$V(x,s) = V^0(x) - V^1(x)s + V^2(x)s^2 + \ldots + (-1)^p V^p(x)s^p + \ldots \tag{3.5}$$

and

$$I(x,s) = I^0(x) - I^1(x)s + I^2(x)s^2 + \ldots + (-1)^p I^p(x)s^p + \ldots \tag{3.6}$$

Substituting Eqn. (3.5) and (3.6) into Eqn. (3.3) and (3.4), the coefficients of s^p ($p = 0, 1, 2, \ldots$) on both sides of the equations satisfies,

$$\frac{dV^p(x)}{dx} = -rI^p(x) + lI^{p-1}(x) \tag{3.7}$$

$$\frac{dI^p(x)}{dx} = cV^{p-1}(x) \tag{3.8}$$

To obtain a moment model for the transmission line TL, we integrate both sides of Eqn (3.7) and (3.8) along the wire [Yu 95, Yu 96]. First, a relation between the p-th order current moments at x and d can be established via integration of Eqn.(3.8) from d to x.

$$I^p(x) - I^p(d) = c\int_d^x V^{p-1}(y)dy \tag{3.9}$$

In the special case when $x = 0$, Eqn (3.9) becomes

$$I^p(0) = I^p(d) - c\int_0^d V^{p-1}(y)dy \tag{3.10}$$

Define U^p as the mean of the p-th order voltage moment $V^p(x)$ along the line (which is also called the p-th order mean for simplicity), i.e.,

$$U^p = \frac{1}{d}\int_0^d V^p(y)dy \tag{3.11}$$

Thus, Eqn (3.10) can be simply written as:

$$I^p(0) = I^p(d) - CU^{p-1} \tag{3.12}$$

Here, CU^{p-1} represents the total p-th order current moment through the capacitance of the line.

Next, we derive an equation relating the p-th order voltage moments at 0 and x by integrating both sides of Eqn (3.7) from 0 to x and using Eqn (3.9):

$$\begin{aligned} V^p(x) - V^p(0) &= -rxI^p(d) + lxI^{p-1}(d) - rc\int_0^x \int_d^y V^{p-1}(z)dzdy \\ &\quad + lc\int_0^x \int_d^y V^{p-2}(z)dzdy \end{aligned} \tag{3.13}$$

where V^{-1} is defined as 0 when $p = 1$. In the special case when $x = d$, Eqn (3.13) becomes

$$\begin{aligned} V^p(d) - V^p(0) &= -RI^p(d) + LI^{p-1}(d) - rc\int_0^d \int_d^x V^{p-1}(y)dydx \\ &\quad + lc\int_0^d \int_d^x V^{p-2}(y)dydx \end{aligned} \tag{3.14}$$

It can be shown via integration manipulations that

$$\int_0^d \int_d^x V^j(y)dydx = -\int_0^d xV^j(x)dx \tag{3.15}$$

Thus, the double integrals in Eqn (3.14) can be transformed into single ones, and Eqn (3.14) becomes

$$\begin{aligned} V^p(d) - V^p(0) &= -RI^p(d) + LI^{p-1}(d) \\ &\quad + rc\int_0^d xV^{p-1}(x)dx - lc\int_0^d xV^{p-2}(x)dx \end{aligned} \tag{3.16}$$

Define W^p as the mean value of weighted p-th order voltage moments along the line with the weight equals to the relative distance x/d (which is called the p-th order x-mean for simplicity), i.e.,

$$W^p = \frac{1}{d^2}\int_0^d xV^p(x)dx \tag{3.17}$$

Eqn (3.14) can be rewritten as:

$$V^p(d) - V^p(0) = -RI^p(d) + LI^{p-1}(d) + RCW^{p-1} - LCW^{p-2} \tag{3.18}$$

In Eqn (3.18), the first part, $-RI^p(d) + LI^{p-1}(d)$, represents the contribution to voltage moment from the load current $I(d)$. Since $I(d)$ can be regarded as a current component flowing through the whole line, its effect is the same as if it passed through a lumped RL branch. The second

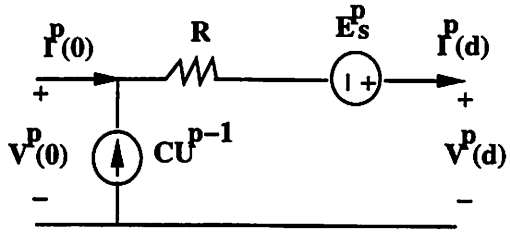

Figure 3.2: Moment Model of Lossy Transmission Line

part, $RCW^{p-1} - LCW^{p-2}$, represents the voltage drop caused by capacitance currents. For this part, the x-mean W characterizes the contribution to voltage moment from the capacitive currents. The weight x/d is introduced in W because the current flowing through the capacitance at position x only causes a voltage drop in the region $[0, x]$.

For equivalent circuit derivation, Eq.(3.18) can be rewritten as:

$$V^p(d) = V^p(0) - RI^p(d) + E_s^p \tag{3.19}$$

where E_s^p is defined as

$$E_s^p = LI^{p-1}(d) + RCW^{p-1} - LCW^{p-2} \tag{3.20}$$

To summarize, Eqn (3.12) and (3.19) characterize the the 2-port equations for the lossy transmission line TL. The equivalent p-th order moment model for TL is shown in Fig. 3.2. In this lumped model, $V^p(0)$ and $V^p(d)$ are regarded as node voltages, $I^p(0)$ and $I^p(d)$ are regarded as node currents at the two ends of the line, respectively. CU^{p-1} and E_s^p are independent current and voltage sources respectively, since they are only dependent on voltage and current moments of lower order. Clearly the crux of this transmission line modeling lies in the computation of U^p and W^p, which is discussed in the next subsection.

3.2.3 Computation of U^p and W^p

The voltage moment expression in Eqn (3.13) can be rewritten as:

$$V^p(x) = V^p(0) + (-rI^p(d) + lI^{p-1}(d))x + rcA^{p-1}(x) - lcA^{p-2}(x) \tag{3.21}$$

where

$$A^j(x) = -\int_0^x \int_d^y V^j(z)dzdy \tag{3.22}$$

c_1	c_2	c_3	c_4	c_5	c_6
$\frac{1}{3}$	$\frac{5}{24}$	$\frac{2}{15}$	$\frac{61}{720}$	$\frac{17}{315}$	$\frac{277}{8064}$
c_7	c_8	c_9	c_{10}	c_{11}	c_{12}
$\frac{62}{2835}$	$\frac{50521}{3628800}$	$\frac{1382}{155925}$	$\frac{540553}{95800320}$	$\frac{21844}{6081075}$	$\frac{598082943}{261534873600}$

Table 3.1: Coefficient Array C

According to the definition of U^p, W^p in Eqn (3.11) and (3.17), we have:

$$U^p = V^p(0) + \frac{1}{2}(-RI^p(d) + LI^{p-1}(d)) + RCX^{p-1} - LCX^{p-2} \tag{3.23}$$

and

$$W^p = \frac{1}{2}V^p(0) + \frac{1}{3}(-RI^p(d) + LI^{p-1}(d)) + RCZ^{p-1} - LCZ^{p-2} \tag{3.24}$$

where

$$X^j = \frac{1}{d^3}\int_0^d A^j(x)dx \tag{3.25}$$

$$Z^j = \frac{1}{d^4}\int_0^d xA^j(x)dx \tag{3.26}$$

Denote I^0, V^0 as the 0-th order current and voltage moments of the topology, respectively. Since

$$V^0(x) = V^0(0) - rI^0(d)x \tag{3.27}$$

we have

$$U^0 = V^0(0) - \frac{1}{2}RI^0(d) \tag{3.28}$$

$$W^0 = \frac{1}{2}V^0(0) - \frac{1}{3}RI^0(d) \tag{3.29}$$

The high order U^ps and W^ps can be calculated from current moments and lower order X and Zs.

According to Eqn (3.25) - (3.22), X^j and Z^j can be computed recursively starting from $j = 0$, and a coefficient array C used for characterizing X^j and Z^j is shown in Table 3.1 [Yu 95, Yu 96]. It can be observed that X^p and Z^p are polynomials of variables $V^k(0)$ and $I^k(d)$ where k ranges from 0 to p. For 0-th order moments,

$$X^0 = c_1V^0(0) + c_2(-RI^0(d)) \tag{3.30}$$

$$Z^0 = c_2V^0(0) + c_3(-RI^0(d)) \tag{3.31}$$

For higher order moments computation, an operator called $shift()$ is defined which has the following properties:

- For a term $A = c_i P$ where P is independent of c_i, $shift(A) = c_{i+2}P$.
- For a term $A = B + C$, $shift(A) = shift(B) + shift(C)$

And the following recursive formulas can be obtained from Eqn (3.25) - (3.22):

$$\begin{aligned} X^{p+1} &= c_1 V^{p+1}(0) + c_2(-RI^{p+1}(d) + LI^p(d)) + RCshift(X^p) \\ &\quad -LCshift(X^{p-1}) \end{aligned} \tag{3.32}$$

$$\begin{aligned} Z^{p+1} &= c_2 V^{p+1}(0) + c_3(-RI^{p+1}(d) + LI^p(d)) + RCshift(Z^p) \\ &\quad -LCshift(Z^{p-1}) \end{aligned} \tag{3.33}$$

For example,

$$\begin{aligned} X^1 &= c_1 V^1(0) + c_2(-RI^1(d) + LI^0(d)) + RCshift(X^0) \\ &= c_1 V^1(0) + c_2(-RI^1(d) + LI^0(d)) + RC(c_3 V^0(0) + c_4(-RI^0(d))) \end{aligned} \tag{3.34}$$

$$\begin{aligned} Z^1 &= c_2 V^1(0) + c_3(-RI^1(d) + LI^0(d)) + RCshift(Z^0) \\ &= c_2 V^1(0) + c_3(-RI^1(d) + LI^0(d)) + RC(c_4 V^0(0) + c_5(-RI^0(d))) \end{aligned} \tag{3.35}$$

Initially, $I^0(d)$ and $V^0(0)$ can be found via circuit analysis by replacing each transmission line with a resistance. After the p-th order moments of the original network are obtained, $V^p(0)$ and $I^p(d)$ are known, X^p, Z^p, U^p and W^p can then be computed and the $(p+1)$-th order moment model of the circuit can be formed.

The moment models discussed here is used to compute the interconnect delays and sensitivities as described in the following section.

3.3 Delay and Sensitivity Computation via Moments

3.3.1 Delay Computation Based on High Order Moments

For lossy transmission line topology N of critical net n, each wire segment in N is formulated as an uniform RLC transmission line. The transfer function at node $i \in N$ in frequency domain, $H_i(s)$, is defined as:

$$H_i(s) = \frac{V_i(s)}{V_{in}(s)} = m_0 - m_1 s + m_2 s \ldots \tag{3.36}$$

where $V_{in}(s)$ and $V_i(s)$ are the Laplace transforms of input signal $v_{in}(t)$ and output response at node i, $v_i(t)$, respectively. m_ns are the moments of $H_i(s)$ under Taylor expansions. Since $L\{\delta(t)\} = 1$, $h_i(t) = L^{-1}\{H_i(s)\}$ is the impulse response at node i. Under q-pole approximation, $H_i(s)$ can be expressed as:

$$H_i(s) = \sum_{j=1}^{q} \frac{k_j}{s - p_j} \tag{3.37}$$

where p_js and k_js are the poles and residues of $H_i(s)$, respectively.

Due to the wire inductance and capacitance of the transmission lines, signal takes certain flight time, τ_i, to propagate along the path from source s to node $i \in N$, and the actual transfer function at i, $G_i(s)$, becomes:

$$G_i(s) = H(s)e^{-s\tau_i} = m'_0 - m'_1 s + m'_2 s^2 \ldots \tag{3.38}$$

where m'_ns are the moments of $G_i(s)$, which are in fact the n-th order voltage moments $V^n(s)$ discussed in the previous section assuming impulse signals at the source. Since,

$$e^{-s\tau_i} = 1 - s\tau_i + (s\tau_i)^2/2 - \ldots \tag{3.39}$$

The relations between moments m_ns of $H_i(s)$ and m'_ns of $G_i(s)$ can be established according to Eqn (3.36) to (3.39):

$$m_0 = m'_0, \quad m_1 = m'_1 - \tau_i m'_0, \quad m_2 = m'_2 + \tau_i^2/2m'_0 - \tau_i m'_1, \ldots \tag{3.40}$$

Since $L\{u(t)\} = 1/s$, the step response at node i in the frequency domain, $V_i(s)$, can be computed by:

$$V_i(s) = G_i(s)/s = \frac{1}{s}\sum_{j=1}^{q} \frac{k_j}{s - p_j} e^{-s\tau_i} \tag{3.41}$$

And the step response in the time domain, $v_i(t)$, becomes:

$$v_i(t) = -\sum_{j=1}^{q} \frac{k_j}{p_j}(1 - e^{p_j(t-\tau_i)}) \tag{3.42}$$

Interconnect delay t_{d_i} at node i, defined as the time taken for the response waveform at i to reach certain specified voltage v_{Th}, can then be expressed as:

$$-\sum_{j=1}^{q} \frac{k_j}{p_j}(1 - e^{p_j(t_{d_i}-\tau_i)}) = v_{Th} \tag{3.43}$$

where k_j, p_js can be obtained from moments, $m_0, m_1, \ldots$, according to Eqn (3.36) and (3.37):

$$m_n = (-1)^{n+1} \sum_{j=1}^{q} \frac{k_j}{p_j} (\frac{1}{p_j})^n, \quad 0 \leq n \leq 2q-1 \tag{3.44}$$

In the special case of 2-pole approximation, i.e., $q = 2$, analytical expression of k_js and p_js can be obtained using up to 2-nd order moments [Kahng 93]:

$$p_{1,2} = \frac{2}{-m_1 \pm \sqrt{4m_2 - 3m_1^2}} \tag{3.45}$$

$$k_1 = -k_2 = -\frac{1}{\sqrt{4m_2 - 3m_1^2}} \tag{3.46}$$

3.3.2 Delay Sensitivity Computation

3.3.2.1 Delay Sensitivity w.r.t. Wire Width

Under q-pole approximation, the sensitivity of delay at node i with respect to width of wire segment l in topology N can be expressed using high order moments as:

$$\frac{\partial t_{d_i}}{\partial w_l} = \sum_{n=0}^{2q-1} \frac{\partial t_{d_i}}{\partial m_n} \frac{\partial m_n}{\partial w_l} \tag{3.47}$$

and $\partial t_{d_i}/\partial m_n$ and $\partial m_n/\partial w_l$ can be computed respectively by:

$$\frac{\partial t_{d_i}}{\partial m_n} = \sum_{j=1}^{q} (\frac{\partial t_{d_i}}{\partial k_j} \frac{\partial k_j}{\partial m_n} + \frac{\partial t_{d_i}}{\partial p_j} \frac{\partial p_j}{\partial m_n}) \tag{3.48}$$

$$\frac{\partial m_n}{\partial w_l} = \frac{\partial m_n}{\partial R_l} \frac{\partial R_l}{\partial w_l} + \frac{\partial m_n}{\partial L_l} \frac{\partial L_l}{\partial w_l} + \frac{\partial m_n}{\partial C_l} \frac{\partial C_l}{\partial w_l} \tag{3.49}$$

where R_l, L_l and C_l are the total resistance, inductance and capacitance of wire l, respectively.

Therefore, the computation of delay sensitivity with respect to wire width w_l, $\partial t_{d_i}/\partial w_l$, using high order moments can be divided into two parts: the computation of delay sensitivities with respect to moments, $\partial t_{d_i}/\partial m_n$, and moment sensitivities with respect to wire width, $\partial m_n/\partial w_l$ for each moment m_n, $0 \leq n \leq 2q-1$.

3.3.2.2 Computation of $\partial t_{d_i}/\partial m_n$

The computation of $\partial t_{d_i}/\partial m_n$ involves the calculation of t_{d_i}'s sensitivities with respect to k_j, p_js, and k_j, p_js' sensitivities with respect to moments. These sensitivities can be computed by differentiating Eqn (3.43) and (3.44), respectively.

By differentiating Eqn (3.43) with respect to k_j, p_j at t_{d_i}, we can obtain $\partial t_{d_i}/\partial k_j$ and $\partial t_{d_i}/\partial p_j$ from:

$$-\frac{1}{p_j}+\frac{1}{p_j}e^{p_j(t_{d_i}-\tau_i)}+(\sum_{u=1}^{q}k_u e^{p_u(t_{d_i}-\tau_i)})\frac{\partial t_{d_i}}{\partial k_j} = 0 \quad (3.50)$$

$$\frac{k_j}{p_j^2}-\frac{k_j}{p_j^2}e^{p_j(t_{d_i}-\tau_i)}+\frac{k_j(t_{d_i}-\tau_i)}{p_j}e^{p_j(t_{d_i}-\tau_i)}+(\sum_{u=1}^{q}k_u e^{p_u(t_{d_i}-\tau_i)})\frac{\partial t_{d_i}}{\partial p_j} = 0 \quad (3.51)$$

To find $\partial k_j/\partial m_n$s and $\partial p_j/\partial m_n$s for a certain m_n, we differentiate both sides of Eqn (3.44) with m_n and obtain $2q$ linear equations:

$$\frac{\partial m_u}{\partial m_n}=\sum_{j=1}^{q}((-\frac{1}{p_j})^{u+1}\frac{\partial k_j}{\partial m_n}+\frac{(-1)^u(u+1)k_j}{p_j^{u+2}}\frac{\partial p_j}{\partial m_n}),\ 0\le u\le 2q-1 \quad (3.52)$$

where $\partial m_u/\partial m_n = 0$, if $u \neq n$; and $\partial m_u/\partial m_n = 1$, otherwise. $2q$ variables, $\partial k_j/\partial m_n$s and $\partial p_j/\partial m_n$s, can be obtained by solving the $2q$ equations in Eqn (3.52).

Again, in the special case of 2-pole approximation, $\partial k_j/\partial m_n$s and $\partial p_j/\partial m_n$s can be computed analytically by differentiating Eqn (3.46) directly without solving the linear matrix in Eqn (3.52):

$$\partial k_1/\partial m_1 = -\partial k_2/\partial m_1 = \frac{6m_1}{\sqrt{(4m_2-3m_1^2)^3}} \quad (3.53)$$

$$\partial k_1/\partial m_2 = -\partial k_2/\partial m_2 = -\frac{4}{\sqrt{(4m_2-3m_1^2)^3}} \quad (3.54)$$

$$\partial p_{1,2}/\partial m_1 = -\frac{2}{(-m_1\pm\sqrt{4m_2-3m_1^2})^2}(-1\mp\frac{3m_1}{\sqrt{4m_2-3m_1^2}}) \quad (3.55)$$

$$\partial p_{1,2}/\partial m_2 = -\frac{2}{(-m_1\pm\sqrt{4m_2-3m_1^2})^2}(\pm\frac{2}{\sqrt{4m_2-3m_1^2}}) \quad (3.56)$$

3.3.2.3 Computation of $\partial m_j/\partial w_l$

Since $\partial R_l/\partial w_l, \partial L_l/\partial w_l$ and $\partial C_l/\partial w_l$ can be easily obtained by differentiating the expressions of R_l, L_l and C_l which are functions of w_l, the calculation of $\partial m_n/\partial R_l, \partial m_n/\partial L_l$ and $\partial m_n/\partial C_l$ becomes the major part in computing $\partial m_n/\partial w_l$ using Eqn (3.49).

According to the relations between m_ns and m'_ns specified in Eqn (3.40), moments m_ns and their sensitivities with respect to R_l, L_l and C_l can be obtained by computing m'_ns and their sensitivities first, i.e.,

$$\partial m_0/\partial w_l = \partial m'_0/\partial w_l,\ \ \partial m_1/\partial w_l = \partial m'_1/\partial w_l - \tau_i\partial m'_0/\partial w_l \ldots \quad (3.57)$$

Since m_n's are in fact the n-th order voltage moments V^ns discussed in the previous section under impulse signal, the method for high order moment and sensitivity computation is presented in the following section based on the exact moment matching model [Yu 95, Yu 96] for lossy transmission line introduced in Section 3.2.

3.3.3 Recursive Moment and Sensitivity Computation for Tree Network

In the following discussion of recursive moment computation for tree network N, the transmission line connecting to node $k \in N$ from root direction is numbered k, R_k, L_k, C_k are denoted as the wire resistance, inductance and capacitance of transmission line k, respectively. C_{sk} is denoted as the loading capacitance at node k, $son(k)$ is the set of son nodes for a non-leaf node k, which connects to k away from the root in the tree topology. δ_x^y is denoted as the abbreviation for differentiation, $\partial y/\partial x$, where x is a variable representing either R, L, or C value of any transmission line in N.

Using the exact moment matching model for lossy transmission line discussed in Section 3.2, moments and sensitivities of N can be computed efficiently from lower orders to higher orders in a recursive manner. Notice that for tree topology N, node 0 and d in the model refer to the near and far end of the transmission line from the root, respectively.

Moments and Sensitivities Computation Algorithm {

1. Start from 0-th order moments of the topology:
   The equivalent 0-th order moment network of N is formed simply by replacing each loading capacitor with an open circuit and each transmission line with a resistor. It can be easily shown that for each transmission line:

$$V^0 = 1, I^0 = 0, U^0 = 1, W^0 = 1/2 \tag{3.58}$$

   Here, U^p and W^p are the means of voltage moments defined by Eqn (3.11) and (3.17), respectively.

2. Compute $(p+1)$-th order moments recursively from p-th order moments:
   For the computation of $(p+1)$-th order moments, we construct the p-th order moment model of N, N^p, which has exactly the same moments as N up to p-th order at each node by replacing each transmission line in N with its p-th order moment model as shown in Fig. 3.2. The $(p+1)$-th order moments and sensitivities of N can then be computed in a bottom-up and top-down fashion:

2.1 Compute current moments I^{p+1}s and their sensitivities from leaves to root (bottom up):

2.1.1 For each transmission line $k \in N$, its current moment and sensitivity at the near end can be computed by Eqn (3.12) and its differentiating form:

$$\begin{aligned} I^{p+1}(0) &= I^{p+1}(d) - C_k U^p \\ \delta_x^{I^{p+1}(0)} &= \delta_x^{I^{p+1}(d)} - \delta_x^{C_k} U^p - C_k \delta_x^{U^p} \end{aligned} \tag{3.59}$$

2.1.2 If the far end d of line k is a leaf node of N, we have:

$$I_{leaf}^{p+1}(d) = -C_{sk} V_{leaf}^p(d) \tag{3.60}$$

2.1.3 Otherwise, the current moment and sensitivity at the far end can be computed from its son nodes as:

$$I_k^{p+1}(d) = -C_{sk} V_k^p(d) + \sum_{j \in son(k)} I_k^{p+1}(0) \tag{3.61}$$

$$\delta_x^{I^{p+1}(d)} = -C_{sk} \delta_x^{V_k^p(d)} + \sum_{j \in son(k)} \delta_x^{I^{p+1}(0)} \tag{3.62}$$

2.2 After I^{p+1}s and their sensitivities are calculated, the voltage moments V^{p+1}s and sensitivities are computed from root back to leaves (top-down):
For each transmission line $k \in N$, the voltage moment and sensitivity at its far end is obtained from Eqn (3.19) and (3.20),

$$\begin{aligned} V^{p+1}(d) &= V^{p+1}(0) - R_k I^{p+1}(d) + E_s^{p+1} \\ E_s^{p+1} &= L_k I^p(d) + R_k C_k W^p - L_k C_k W^{p-1} \end{aligned}$$

and their differential forms are,

$$\delta_x^{V^{p+1}(d)} = \delta_x^{V^{p+1}(0)} - \delta_x^{R_k} I^{p+1}(d) - R_k \delta_x^{I^{p+1}(d)} + \delta_x^{E_s^{p+1}} \tag{3.63}$$

$$\begin{aligned} \delta_x^{E_s^{p+1}} &= \delta_x^{L_k} I^p(d) + L_k \delta_x^{I^p(d)} + \delta_x^{R_k} C_k W^p + R_k \delta_x^{C_k} W^p + R_k C_k \delta_x^{W^p} \\ &\quad - \delta_x^{L_k} C_k W^{p-1} - L_k \delta_x^{C_k} W^{p-1} - L_k C_k \delta_x^{W^{p-1}} \end{aligned} \tag{3.64}$$

Notice that for each node $k \in N$, $V_k^{p+1}(d) = V_j^{p+1}(0), \forall j \in son(k)$.

2.3 Compute U^{p+1}, W^{p+1} and their sensitivities for each transmission line k:
Once the $(p+1)$-th order current and voltage moments are obtained, the U^{p+1}, W^{p+1}

of each transmission line k can be computed using Eqn (3.23) and (3.24):

$$\begin{aligned} U^{p+1} &= V^{p+1}(0) + 1/2(-R_k I^{p+1}(d) + L_k I^p(d)) + R_k C_k X^p \\ &\quad -L_k C_k X^{p-1} \\ W^{p+1} &= 1/2V^{p+1}(0) + 1/3(-R_k I^{p+1}(d) + L_k I^p(d)) + R_k C_k Z^p \\ &\quad -L_k C_k Z^{p-1} \end{aligned}$$

where X^p, Z^p are calculated recursively using Eqn (3.31) to (3.33).
Similar to the sensitivity computation in previous steps, $\delta_x^{U^{p+1}}$, $\delta_x^{W^{p+1}}$ and $\delta_x^{X^{p+1}}$, $\delta_x^{Z^{p+1}}$ can be obtained by differentiating these equations.

}

Notice that for each transmission line $k \in N$, $\delta_x^{R_k}, \delta_x^{L_k}, \delta_x^{C_k} = 1$, only when x represents the respective R_k, L_k, C_k variable of line k; in all other cases, $\delta_x^{R_k}, \delta_x^{L_k}, \delta_x^{C_k} = 0$. The complexity of Step 2 of the algorithm is linear to the number of nodes, i.e., the number of wire segments $|N|$, in the tree network N. The complexity of computing moments of N up to p-th order is therefore, $O(|N|p)$.

3.4 Delay Optimization Based on Sensitivity Analysis

Once the sensitivities of node delays with respect to the widths of wire segments in an existing topology of a critical net are computed using the methods discussed in Section 3.3, they can be utilized in a sensitivity-based wiresizing algorithm which minimizes the maximum delay of the net. This optimization process improves the net performance by adjusting the widths of wires in the existing topology appropriately using the routing space still available on the chip at th post routing stage.

3.4.1 Problem Formulation

Denote $\mathbf{w} = (w_1, \ldots, w_n)$ and $\mathbf{l} = (l_1, \ldots, l_n)$ as the respective width and length vector of a lossy transmission line topology N having $n = |N|$ wire segments. The R, L, C values of each transmission line $k \in N$ can be expressed simply by:

$$R_k = r_o \frac{l_k}{w_k}, \quad L_k = l_0 \frac{l_k}{w_k}, \quad C_k = c_0 l_k w_k \tag{3.65}$$

where r_0, l_0 and c_0 are the unit resistance, inductance and capacitance of a wire having minimum width w_0. Eqn (3.65) is the simplest way to represent the interconnect parasitics of a lossy transmission line, more accurate expression for R, L, C values under various interconnect modelings can be used in place of Eqn (3.65) without invalidating the sensitivity-based optimization discussed in this section.

During the wiresizing of routing topology N at the post routing stage, the length vector of N, $\mathbf{l}$, is fixed and $\mathbf{w}$ is the only adjustable vector. Since topology N is obtained from a feasible routing solution of the chip, the upper bound on the wire widths of N, $\mathbf{w}_{ub}$, can be specified based on the available space in the routing regions on the route of N. The lower bound on $\mathbf{w}$, $\mathbf{w}_{lb}$, is defined as the minimum wire width w_0 allowed for the topology. Together, $\mathbf{w}_{ub}$ and $\mathbf{w}_{lb}$ define the boundaries of a n-dimensional feasible space S for $\mathbf{w}$ during the wiresizing. The objective of sensitivity-based delay minimization is to find a width vector $\mathbf{w_{opt}}$ in the feasible space S via directed search such that the maximum delay of N as a function of $\mathbf{w}$ is minimized at $\mathbf{w_{opt}}$. Since the maximum delay of a lossy transmission line topology, $t_{d_{max}}$, can not be analytically expressed as a function of $\mathbf{w}$, it is calculated using Eqn (3.43) at each $\mathbf{w}$ in the feasible space. This constrained wiresizing problem can be formulated as:

$$\begin{aligned} &\textit{Minimize} && t_{d_{max}}(\mathbf{w}) \\ &\textit{Subject to:} && \\ & && \mathbf{w}_{lb} \leq \mathbf{w} \leq \mathbf{w}_{ub} \end{aligned}$$

For lossy transmission line topology, the delay at each node i consists of both the flying time on the path from the source to i and the rising delay of the response waveform at i. According to Eqn (3.65), the wiresizing of N can improve the maximum rising delay of the topology, while the signal propagation time from source to each node, determined by $\sum_i \sqrt{L_i C_i}$ of each segment i on the path, is independent of $\mathbf{w}$.

3.4.2 Sensitivity-based Wiresizing Algorithm

The sensitivity-based wiresizing algorithm is an iterative optimization process using the directed search method. Starting at a feasible point $\mathbf{w} \in S$, it analytically computes the maximum delay sensitivities with respect to the widths of wires in the topology via high order moments and adjusts the width of the most sensitive wire appropriately within S for maximum delay minimization. The entire algorithm is outlined as follows:

Sensitivity-based Wiresizing Algorithm {

1. (Initialization) Choose the initial feasible point w $\in S$, set the wire width increment Δw and sensitivity threshold ϵ.

2. (Direct Search) Repeat

   2.1 (Moment and delay computation) Compute the high order moments and the maximum delay $t_{d_{max}}$ of N at w.

   2.2 (Delay sensitivity computation) Compute the sensitivity vector of the maximum delay at w, $\partial t_{d_{max}}/\partial \mathrm{w}$.

   2.3 (Wiresizing) Choose a sizable wire i at w having maximum $|\partial t_{d_{max}}/\partial w_i|$:

      2.3.1 If $\partial t_{d_{max}}/\partial w_i > 0$, decrease w_i by Δw.

      2.3.2 otherwise, increase w_i by Δw.

   2.4 Update Δw accordingly.

3. Terminate when max $|\partial t_{d_{max}}/\partial \mathrm{w}| < \epsilon$.

}

Wire i is defined as sizable at w if it can be adjusted by Δw within S, i.e., $w_i - \Delta w \geq w_{i_{lb}}$ when $\partial t_{d_{max}}/\partial w_i > 0$, and $w_i + \Delta w \leq w_{i_{ub}}$ when $\partial t_{d_{max}}/\partial w_i < 0$. Δw may not be fixed in the wiresizing process, which can be reduced gradually at Step 2.3 as w approaches w_{opt} to allow fine adjustments of wire width within the feasible space. The algorithm usually starts at w having the minimum width w_0 for all wires and always chooses the steepest descent direction at each iteration in order to speed up the convergence of the algorithm and to minimize the maximum delay with the least amount routing resource consumption.

3.4.3 Algorithm Analysis

It is observed during our experiments of the sensitivity-based algorithm that the monotonicity property for the wiresizing of tree networks, i.e., wires should be non-increasingly sized on any path from the source to a sink for maximum delay minimization, still holds for lossy transmission line topologies, as also witnessed by results obtained with other methods [Menezes 95, Wang 94, Zhu 93]. This observation can be explained as follows. According to Eqn (3.62) to (3.64), the moment (voltage) sensitivities at the maximum delay node with respect to the resistance and inductance of a wire include currents from its entire sub-tree, while its sensitivities with respect to the wire capacitance are computed based on the path from source only. Therefore, the maximum

delay sensitivities with respect to the wires near the source are dominated by the resistive and inductive sensitivities and are much larger than its sensitivities with respect to the wires near the leaves. In another words, the widths of those wires close to the source have much higher impact on node delays than those near the leaves. This implies that, for delay optimization, the widths of wires near the source should be sized up compared to wires near the leaves and the wire widths of the topology should be non-increasing on any path from the source to sinks. In addition, a monotonically-sized path can lead to faster rising slope of the waveform and smaller delay at the sink than a non-monotonically sized one according to the properties of waveform transmission [Bakoglu 90].

The monotonicity property for wiresizing implies that there exists an optimal ratio r_{opt} among widths of wires in N for any structurally unsymmetric tree network N, which uniquely defines a width vector $\mathbf{w_{opt}}$ at which the maximum delay of N is minimized in the feasible space S.

Property 3.1 *For a structurally unsymmetric tree network N, there exists an unique point $\mathbf{w_{opt}} \in S$ such that maximum delay of N is minimized at $\mathbf{w_{opt}}$.*

In the general cases where tree topologies are structurally symmetric, we define $\mathbf{W_{opt}}$ as the set of points at which maximum delay of N is minimized. Due to the existence of the optimal width ratio r_{opt} of N, the delay at any non-optimal point $\mathbf{w} \in S$ can always be reduced by adjusting the width of at least one wire in N so that the current wire width ratio at $\mathbf{w}$, r, becomes closer to r_{opt}.

Property 3.2 *For any point $\mathbf{w} \notin \mathbf{W_{opt}}$, the maximum delay can be improved in at least one direction, i.e., there exists at least one wire i satisfying $|\partial t_{d_{max}}/\partial w_i| > 0$, such that $t_{d_{max}}(\mathbf{w}') < t_{d_{max}}(\mathbf{w})$, where $\mathbf{w}'$ is obtained by adjusting w_i with Δw.*

The two properties above indicate that, although the maximum delay of N may not be a convex function of $\mathbf{w}$, the sensitivity-based wiresizing algorithm will not be "trapped" at any local optimal point in the directed search process. In fact, it can always move to a new point in the feasible space S having a smaller maximum delay until a global minimum point is reached. For example, Fig. 3.3 shows the maximum delay of a transmission line topology (Testing Topology One in Section 5) plotted as a function of the widths of three wires in it (while the widths of the other wires remain unchanged). It can be seen that although the maximum delay function is not convex, $t_{d_{max}}(\mathbf{w})$ at any point $\mathbf{w} \in S$ can be improved in at least one direction on the function surface until it reaches a global optimal point in S.

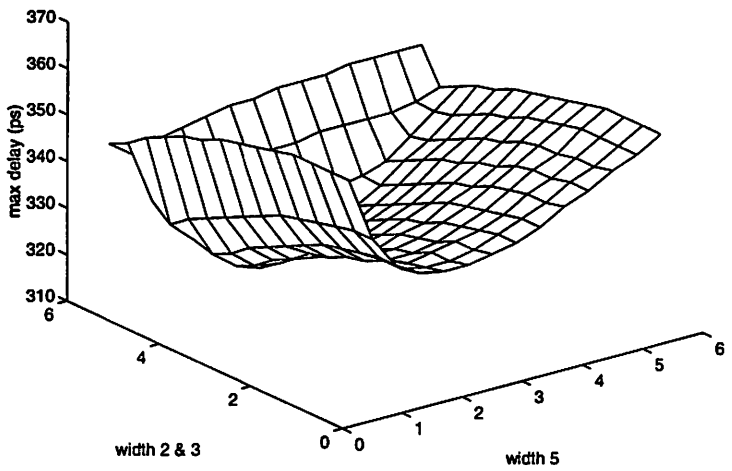

Figure 3.3: Example: Maximum Delay vs. Wire Widths

3.4.4 Other Advantages of the Algorithm

Besides maximum delay minimization, the sensitivity-based wiresizing algorithm described in Section 3.4.2 also reduces the overshootings of response waveforms at sinks of the topology and generates a final wiresizing solution robust under parameter variations.

3.4.4.1 Overshooting Reduction

Under lossy transmission line formulation, the damping condition of the topology is largely determined by the ratio between the driver resistance R_d at the source and the characteristic impedance Z of wires connecting to it [Bakoglu 90] (if there are multiple wires connecting to the source, the actual characteristic impedance Z can be computed as $Z = Z_1||Z_2\ldots$). If $R_d < Z$, N is under-damped and strong transmission line effect may cause overshootings of the response waveforms at sinks in N, resulting in waveform oscillations and possible malfunctioning of the circuit. According to the analysis in Sec. 3.4.3, the solution by the sensitivity-based wiresizing algorithm follows the monotonicity property, which implies that the wires connecting to the source should have the largest widths in N. Since characteristic impedance is proportional to wire capacitance and inversely proportional to wire inductance [Bakoglu 90], it is inversely proportional to wire width according to Eqn (3.65) and can be reduced remarkably after wiresizing. As the result, the ratio

between R_d and Z_k becomes much closer to 1 and the overshootings of the response waveforms can be largely eliminated due to improved dumping conditions.

3.4.4.2 Robustness under Parameter Variations

Another advantage of the sensitivity-based wiresizing is its ability to generate final solutions robust under parameter variations. During the manufacturing process, the widths of interconnect wires in the routing topologies may vary from their computed values in layout design. This fluctuation in wire widths may affect the optimality of the solutions obtained in routing in terms of their interconnect performances. Therefore, for the best performance of the circuit, the sensitivities of maximum delay with respect to the widths of wires in the topology should be minimized so that interconnect performance is affected little by the changes in wire widths. The sensitivity-based wiresizing algorithm in Sec. 3.4.2 generates the optimal solution $\mathbf{w_{opt}}$ satisfying $\max|\partial t_{d_{max}}/\partial \mathrm{w}_{opt}| \rightarrow 0$. This implies that the maximum delay will vary little when w fluctuates around $\mathbf{w_{opt}}$ during manufacturing, i.e., the wiresizing solution $\mathbf{w_{opt}}$ is robust under parameter variations.

3.5 Experimental Results

The sensitivity-based wiresizing algorithm using high order moments has been implemented and tested on a DEC 5000/125 workstation. Various groups of interconnect parameter values are tested on each of the three testing topologies shown below to verify the effectiveness of the algorithm. In our experiments, 2-pole approximation is used for delay estimation and optimization. Although larger number of poles can be used in our optimization method, experiments demonstrate that 2-pole approximation is accurate enough for guiding the delay minimization as illustrated in Sec. 3.5.1. For all testing topologies, the response waveforms at the maximum delay node by 2-pole approximation and simulation using SPICE3f5 are compared and the maximum delays are measured at threshold voltage of 0.5. The upper bound on the width of each wire during wiresizing is set to $6\mathrm{x}w_0$, where w_0 is the minimum wire width allowed for an interconnect wire.

The first testing topology (Fig. 3.4) is a tree network consisting of seven lossy transmission lines. Each wire segment is $0.01m$ in length and the loading capacitance C_s at each node is $1pF$. For the testing results shown in Fig. 3.5 to 3.7, the values of interconnect parameters of Topology

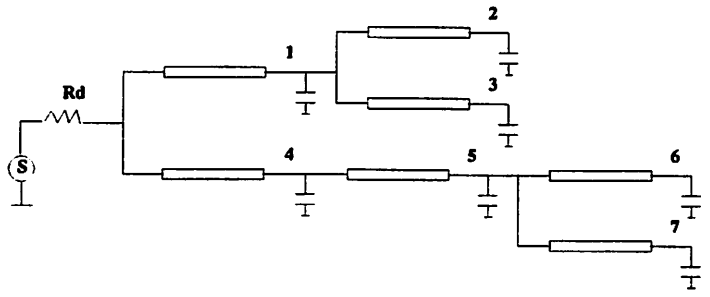

Figure 3.4: Testing Topology One

One are set as follows (Results under other groups of parameter values are listed in Table 3.2):

$$R_d = 15\Omega,\ \ R = 800\Omega/m,\ \ L = 3.8x10^5 pH/m,\ \ C = 30pF/m$$

3.5.1 Accuracy of Delay Approximation via Moments

Unlike moment matching method in simulation, whose goal is to achieve perfect fitting between the actual and approximated waveforms, interconnect performance optimization via moment computation only requires accurate delay approximation, i.e., the waveform generated via high order moments should well follow the rising slope of the actual response waveform. Therefore, fewer poles are actually needed in our sensitivity-based wiresizing approach compared to matching-based simulation methods.

Fig. 3.5 and 3.6 show the response waveforms at the maximum delay node of Topology One before and after optimization, generated by 2-pole approximation and SPICE simulation respectively. It can be seen that the waveforms by 2-pole approximation match well with the rising slopes of the actual response waveforms by SPICE. The difference between the delays measured at $0.5V$ is small, and the actual delay is cut by nearly the same amount as the estimated one after optimization. This demonstrates that 2-pole approximation is very accurate for delay estimation of lossy transmission line topologies. It can also be observed that the overshooting of the response waveform is eliminated after the optimization, verifying our analysis in Sec. 3.4.4.1. Furthermore, the 2-pole approximation generates the waveform three orders of magnitude faster than SPICE simulation due to the efficiency of our moments computation method.

3.5.2 Accuracy of Sensitivity-based Delay Analysis

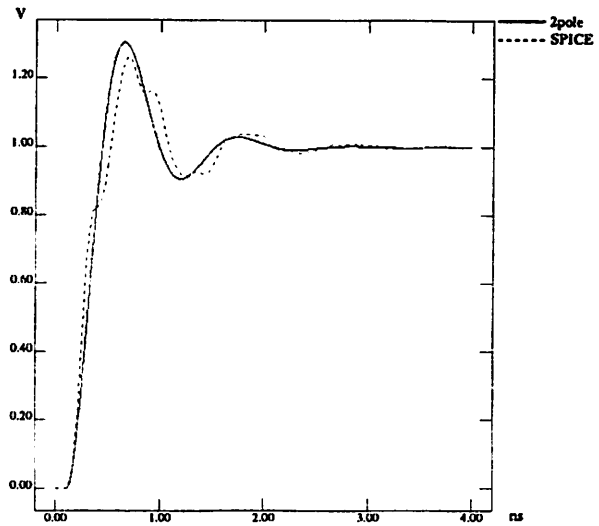

Figure 3.5: Waveforms at Maximum Delay Node Before Optimization

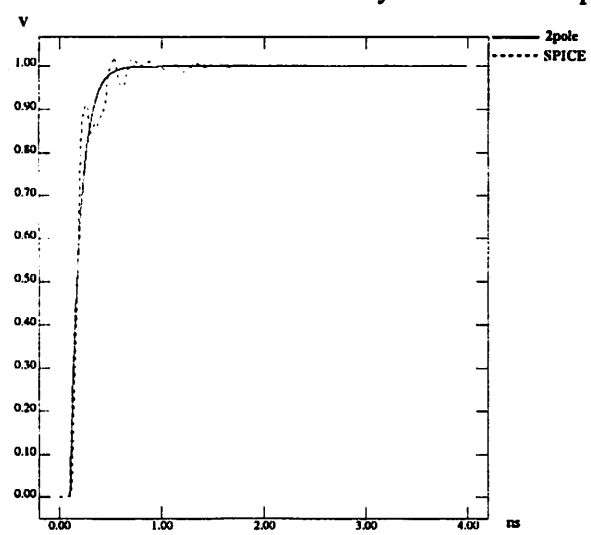

Figure 3.6: Waveforms at Maximum Delay Node After Optimization

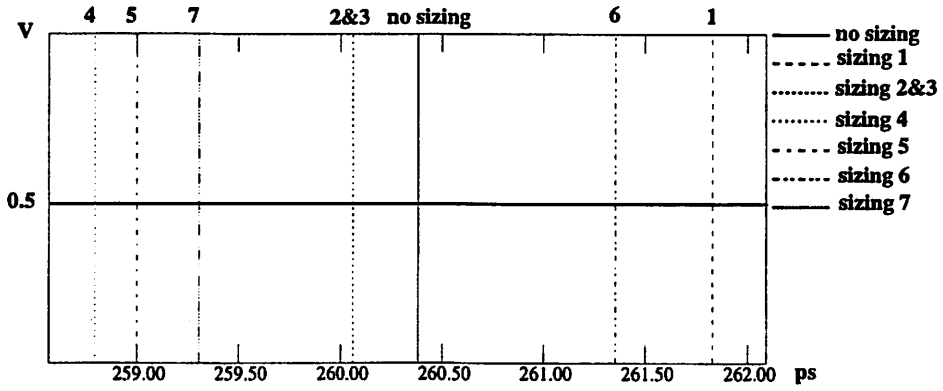

Figure 3.7: Accuracy of Sensitivity-based Delay Analysis

To investigate the accuracy of sensitivity-based wiresizing via high order moments, we pick a feasible width vector $\mathbf{w} = (1, 1, 1, 3, 2.4, 1.2, 1.2)$ of Topology One and compute the sensitivity vector of the maximum delay node 7 at w: $\partial t_{d_{max}}/\partial \mathbf{w} = (15, -3, -3, -16, -14, 10, -12)$. $\partial t_{d_{max}}/\partial \mathbf{w}$ indicates that, if only one wire width is allowed to be increased by Δw in the feasible space, the wires in N can be ordered as (4, 5, 7, 2&3, 6, 1) according to the changes they cause in $t_{d_{max}}$ from maximum decrease to maximum increase. This order suggests that the maximum delay at w, $t_{d_{max}}(\mathbf{w})$, can be reduced most by sizing up wire 4, while it is most penalized if width of wire 1 is increased.

Fig. 3.7 shows the actual changes in maximum delay by sizing up each wire in N one at a time, measured by the actual response waveforms at the maximum delay node generated by SPICE. It can be observed that they follow the same order as predicted by $\partial t_{d_{max}}/\partial \mathbf{w}$ and are proportional to the sensitivities estimated, i.e., the maximum delay will decrease if wire 4, 5, 7, 2&3 are sized up, while it will increase if the widths of wire 6, 1 are increased. This demonstrates that the sensitivity-based delay analysis is very accurate. It can also be seen from $\partial t_{d_{max}}/\partial \mathbf{w}$ that the widths of wires closer to the source have larger impact on maximum delay than those further aways from it along the path from the source to the maximum delay node 7, which is consistent with our analysis on the monotonicity of wiresizing for delay minimization in Sec. 3.4.3.

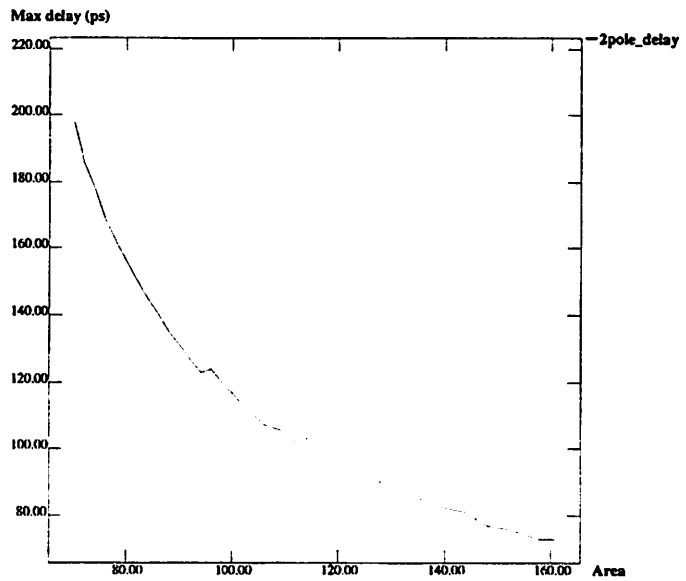

Figure 3.8: Maximum Delay vs. Routing Area

Table 3.2: Maximum Delay Minimization for Testing Topology One

Parameters				Before Optimization			After Optimization				
	/m				Maximum Delay			Maximum Delay			
R_d	R	L	C	Area	2-pole	SPICE	Area	2-pole	Cut	SPICE	Cut
Ω	Ω	nH	pF	mxw_0	ps	ps	mxw_0	ps	-%	ps	-%
15	800	380	30	0.07	197	157	0.16	76	61	71	55
10	1600	380	60	0.07	217	151	0.17	72	66	60	60
25	3200	1520	15	0.07	432	365	0.24	141	67	131	64

3.5.3 Delay Minimization

3.5.3.1 Testing Topology One

For maximum delay minimization, Fig. 3.8 shows the trade offs between the maximum delay and total routing area of Topology One during wiresizing, starting with minimum width w_0 for all wire segments in the topology. It can be observed that the maximum delay of the topology can be cut significantly by the sensitivity-based wiresizing approach with small increase in routing area.

For further testing of the wiresizing algorithm for delay minimization, various groups of interconnect parameter values are tested on Topology One and the results of maximum delay

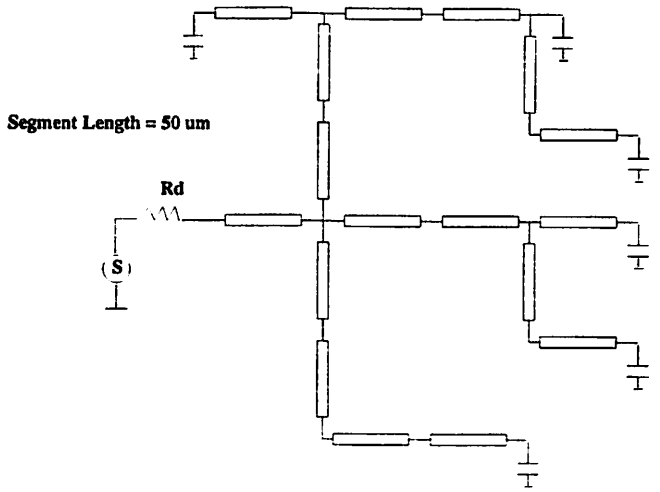

Figure 3.9: Testing Topology Two

Table 3.3: Maximum Delay Minimization for Testing Topology Two

Parameters /μm			Before Optimization			After Optimization				
				Maximum Delay			Maximum Delay			
R Ω	L pH	C fF	Area $\mu m x w_0$	2-pole ps	SPICE ps	Area $\mu m x w_0$	2-pole ps	Cut -%	SPICE ps	Cut -%
0.15	2.46	0.176	850	343	294	2870	123	64	103	65
0.15	2.46	1.76	850	359	300	2610	156	57	126	58
0.15	0.246	1.76	850	380	307	2673	179	53	141	54

minimization are listed in Table 3.2. On average, the maximum delay is reduced by 65% and 60% after wiresizing, measured by 2-pole approximation and SPICE simulation respectively. The difference between the delay cutting ratios is less than 5%. The routing area becomes 1.71 times larger after wiresizing, indicating a good area-delay trade off.

3.5.3.2 Testing Topology Two

The second testing topology is a tree network studied in [Kahng 93, Zhou 94]. The driver resistance is set as 10Ω, and the loading capacitance at each sink is $2pF$.

It can be seen from Table 3.3 that, under various groups of interconnect parameter values, the maximum delay of Topology Two is cut by average of 58% and 59%, measured by 2-pole approximation and SPICE simulation respectively. The difference between the two cutting rates is

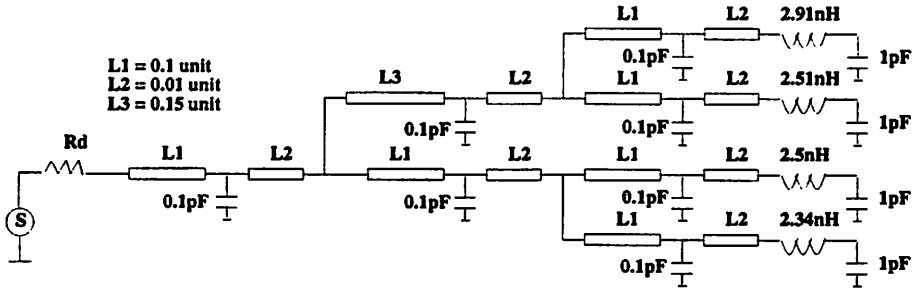

Figure 3.10: Testing Topology Three

Table 3.4: Maximum Delay Minimization for Testing Topology Three

Parameters				Before Optimization			After Optimization				
	/unit			Area	Max Rising Delay		Area	Max Rising Delay			
R_d	R	L	C	*unit*	2-pole	SPICE	*unit*	2-pole	Cut	SPICE	Cut
Ω	Ω	*nH*	*pF*	xw_0	ns	ns	xw_0	ns	-%	ns	-%
15	150	200	50	0.82	1.05	0.78	1.44	0.33	69	0.20	74
50	300	400	10	0.82	0.85	0.73	1.35	0.35	59	0.26	64
50	75	200	10	0.82	0.56	0.54	1.07	0.23	59	0.20	63

only 1%. The routing area increases an average of 2.19 times after wiresizing.

3.5.3.3 Testing Topology Three

The third testing topology is one of the benchmarks of 1993 IEEE Multi-Chip Module Conference (MCMC-93), provided by Performance Signal Integrity, Inc.

Table 3.4 shows that, with an average of only 57% increase in the routing area of Topology Three, the optimization via wiresizing cuts its maximum delay measured by 2-pole approximation and SPICE by average of 62% and 67% respectively under various groups of parameters. The difference between the two reduction rates is within 5%.

3.5.4 Summary

Each optimal solution in Table 3.2 to 3.4 is generated in less than 10 seconds of CPU time on a DEC 5000 workstation, due to the efficient moment and sensitivity computation at every step during the wiresizing process.

From the experimental results of these testing topologies, it can be observed that:

1. The sensitivity-based wiresizing algorithm can reduce maximum delay significantly by average of over 60%. The delay sensitivities computed using high order moments are very reliable in guiding the wiresizing process. The increase in routing area after wiresizing is moderate, ranging from 0.57 to 2.19 times of the original net area under minimum wire width. This indicates that our approach can achieve good area-delay trade offs.

2. The accuracy of the delay approximation using high order moments is very good, the difference in delay cutting rates measured by 2-pole approximation and SPICE simulation respectively is only 1 to 5%. This demonstrates that 2-pole approximation provides good estimation for guiding the performance optimization of lossy transmission line topologies.

3.6 Conclusions

This chapter discusses the post routing performance optimization of lossy transmission line topologies, which are typical for modeling off-chip interconnects under MCM and PCB technologies. A sensitivity-based approach is presented which improves the performance of an existing routing topology by adjusting the widths of its wires under routing resource constraints. The maximum delay and its sensitivities with respect to the width of each wire in the topology are computed via high order moments based on an exact moment matching model for each lossy transmission line. Compared with previous approaches, it achieves analytical moment (sensitivity) computation and calculates higher order moments (sensitivities) recursively from lower order moments for tree networks. Experiments show that the delay estimation using high order moments is very accurate compared with SPICE simulation and the sensitivity-based wiresizing approach can reduce maximum delays of the testing topologies significantly with small penalties in routing area. Besides delay optimization, the final solutions generated by the wiresizing algorithm also eliminate the overshootings of the response waveforms and are robust under parameter variations during the manufacturing process.

Bibliography

[Bakoglu 90] H. Bakoglu, "Circuits, Interconnections, and Packaging for VLSI", Addison-Wesley, 1990.

[Bracken 92] J. Bracken, V. Raghavan and R. Rohrer, "Interconnect Simulation with Asymptotic Waveform Evaluation(AWE)," *IEEE Trans. on Computer and Systems-I*, vol. 39, No. 11, pp. 869-878, Nov. 1992.

[Cong 93] J. Cong and K. Leung and Dian Zhou, "Performance-driven Interconnect Design Based on Distributed RC Delay Model," *Proc. 30th Design Automation Conf.* pp. 606-611, 1993.

[Elmore 48] W.C.Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers", *J. Applied Physics*, 19.1, pp.55-63, 1948.

[Kahng 93] A.B. Kahng, S. Muddu, "Optimal Equivalent Circuits for Interconnect Delay Calculations Using Moments," *Proc. ACM/IEEE European Design Automation Conf.*, Sept. 1994, pp. 164-169.

[Menezes 94] N. Menezes, S. Pullela, F. Dartu, L. Pillage, " RC Interconnect Synthesis - A Moment Fitting Approach", *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design*, pp. 418-425, 1994.

[Menezes 95] N. Menezes, S. Pullela, L. Pileggi, "Simultaneous Gate and Interconnect Sizing for Circuit-Level Delay Optimization," *Proc. 32nd Design Automation Conf.*, pp. 690-695, 1995.

[Sapatnekar 94] S. Sapatnekar, "RC Interconnect Optimization under the Elmore Delay Model," *Proc. of 31st Design Automation Conf.*, pp. 387-391, 1994.

[Wang 94] J. Wang, W. Dai, "Optimal Design of Self-Damped Lossy Transmission Lines for Multichip Modules," *Proc. Int'l Conf. on Computer and Design* , pp. 594-598, 1994.

[Xue 96a] T. Xue, E. Kuh, Q. Yu, "A Sensitivity-Based Wiresizing Approach to Interconnect Optimization of Lossy Transmission Line Topologies", *Proc. IEEE Multi-Chip Module Conf.*, pp. 117-121, 1996.

[Yu 95] Q. Yu, E. Kuh, "Moment Models of General Transmission Lines with Application to MCM Interconnect Analysis," *Proc. IEEE Multi-Chip Module Conf.*, pp. 158-163, 1995.

[Yu 96] Q. Yu, E. Kuh, T. Xue, "Moment Models of General Transmission Line with Application to Interconnect Analysis and Optimization", *IEEE Trans. on VLSI Systems*, Dec. 1996, pp. 477-494.

[Zhou 94] D. Zhou, S. Su, F. Tsui, D. Gao, "A Simplified Synthesis of Transmission Lines with A Tree Structure," *Int'l Journal of Analog Integrated Circuits and Signal Processing*, Jan. 1994, pp. 19-30.

[Zhu 93] Q. Zhu, W. Dai, J. Xi, "Optimal Sizing of High-Speed Clock Networks Based on Distributed RC and Lossy Transmission Line Models," *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design*, pp. 628-633, 1993.

Chapter 4

Post Global Routing Crosstalk Risk Estimation

4.1 Introduction

4.1.1 Motivation

Due to the scaling down of device geometries in deep-submicron technologies, interconnect wires are placed in increasingly closer proximity and higher density (Sec. 1.2.1). As a result, the coupling capacitance between adjacent nets has increased significantly and the crosstalk noise it causes has become an important concern in high performance circuit design. If un-optimized, crosstalk noise may cause signal delay, logic hazards, and even malfunctioning of a circuit.

The crosstalk noise is routing-dependent, since the coupling capacitances between nets are determined by the routes of interconnect wires on the chip. Therefore, similar to interconnect performance optimization discussed in Chapter 2 and 3, it is most appropriate to address crosstalk risk estimation and reduction at the post routing stage in layout after a feasible routing solution of the chip has been obtained.

According to the circuit layout flow, crosstalk synthesis can possibly be pursued at two levels in the routing process: the detailed routing level and the global routing level. Inspite of its increasing importance in high performance circuit design in recent years, crosstalk minimization is still a largely un-addressed problem and most of the previous approaches to crosstalk synthesis in routing are localized optimization methods at the post processing or detailed routing level [Chen 92, Chaudhary 93, Gao 93, Gao 94, Kirkpatrick 94]. They adopt net-based approaches which

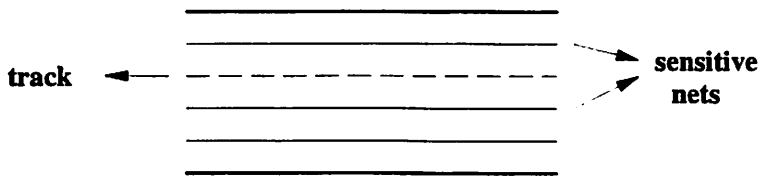

Figure 4.1: An Example

estimate the crosstalk noise at each net in a channel or switchbox individually and reduce the coupling between adjacent wires within each region via spacing (adjusting the separation space between wires) [Chen 92, Chaudhary 93], track permutation [Gao 94] or assignment (assigning nets appropriately to tracks) [Kirkpatrick 94]. Although these methods can achieve some reductions in the crosstalk noise of a circuit, they alone are often insufficient to achieve a crosstalk risk-free final layout solution of the chip since the optimization at the detailed routing level has very limited routing flexibility, i.e., it can only adjust the routes of nets locally within one routing region, not globally among all regions on a chip. Consequently, their effectiveness for crosstalk reduction depends heavily on the global routing solution of the circuit, and they often fail to achieve satisfactory results especially for those regions having high densities of sensitive nets and limited routing resources. For example, it is impossible to avoid crosstalk noises among three nets that are sensitive to each other and are routed in a region having only four routing tracks at the detailed routing level (Fig. 4.1).

Due to the limitations of these localized approaches, it is important to address crosstalk optimization not only in detailed routing, but also in global routing as well. Unlike "net-based" risk estimation and reduction "within" each routing region at the detailed routing level, the approach at the global routing level should be "region-based", which estimates the crosstalk risk for each routing region on the chip as a whole and reduces the risks by adjusting nets' routes globally "among" routing regions on the chip. Instead of generating a specific risk-free final solution for each region, its objective is to produce a global routing solution of the chip in which there exists a risk-free final solution of each region. The differences between net-based and region-based methods are summarized in Table 4.1.

The crosstalk synthesis in global routing can significantly improve the chances of generating a final risk-free routing solution of a chip for the following reasons:

1. It allows global estimation of the crosstalk violations on the chip and avoids the time-consuming iterations in routing by identifying and eliminating the crosstalk violations at an early stage. In many cases, once the nets routed through a region are known, it is possible

Table 4.1: Net-based vs. Region-based Crosstalk Synthesis

Approach	Net-Based	Region-Based
Based on	Detailed routing solution	Global routing solution
Estimation	Coupling noise on each net	Crosstalk risk of each region
Adjustment	Among tracks within one region	Among regions on a chip
Objective	A specific risk-free final solution	The existence of a risk-free final solution

to identify the feasibility of a risk-free final solution of the region before moving further to the detailed routing stage. For example, from the density and capacity of the region shown in Fig. 4.1 (obtained from a global routing solution), it is quite clear that a risk-free routing solution of the region is not possible regardless of the choice of the detailed routing methods and the current global routing solution must be adjusted for crosstalk avoidance.

2. Synthesis at the global routing level, which allows the routes of nets to be adjusted globally among all routing regions on the chip, provides much more routing flexibility for crosstalk risk reduction than those at the detailed routing level.

3. Whether a net is subject to crosstalk violation depends not only on the coupling noises from its adjacent nets, but also on its *risk tolerance bound* - the maximum amount of crosstalk noise it can tolerate without affecting the functionality of circuit. Therefore, the crosstalk synthesis should be formulated as a constrained optimization instead of noise minimization problem as in most previous methods. Typically, the risk tolerance bound is specified for each sensitive net which may suffer crosstalk noises within all of its routing regions on the chip. Thus, for constrained crosstalk synthesis of each region using risk tolerance bounds as constraints, the bound of each sensitive net must be partitioned appropriately among its routing regions on the chip (Fig. 4.2). Again, this risk tolerance bound partitioning problem can only be addressed at the global routing level based on an overall estimation of the current routing and crosstalk situation of the chip.

4.1.2 Algorithm Overview

In this and next chapter, we present a post global routing crosstalk optimization approach [Xue 96b, Xue 96c], which to our knowledge, is the first to estimate and reduce crosstalk risk at the global routing level. Given as its input a feasible global routing solution of the chip, sensitivities and

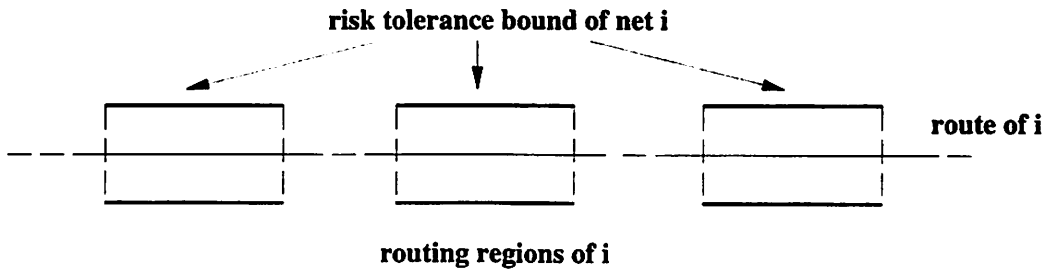

Figure 4.2: Risk Tolerance Bound Partitioning

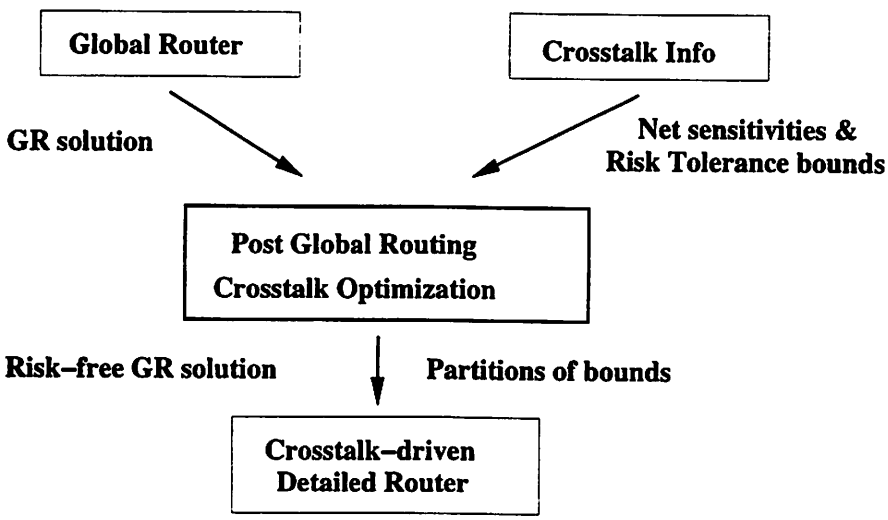

Figure 4.3: Post Global Routing Crosstalk Estimation and Reduction

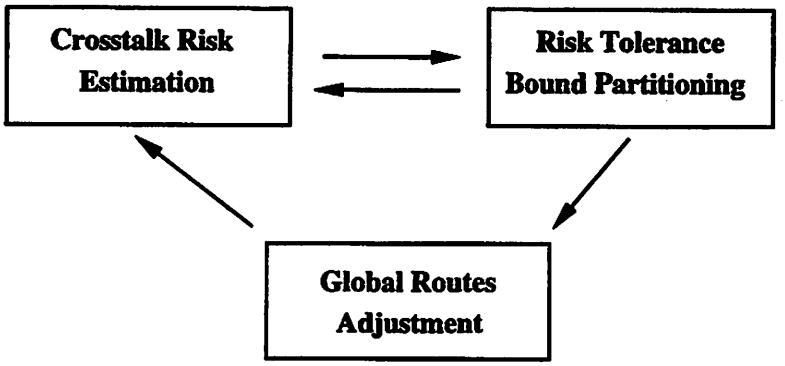

Figure 4.4: Crosstalk Risk Estimation and Reduction Process

risk tolerance bounds of nets, our approach produces a risk-free global routing solution in which all regions on the chip are free of crosstalk violations. Compared to the original global routing solution before optimization, this risk-free solution is a much better starting point for a crosstalk-driven detailed router to generate a risk-free final routing solution of the chip, i.e., much of the routability problems at the detailed routing level due to crosstalk violations can be eliminated at an earlier stage. In addition, it generates partitions of risk tolerance bounds of nets among their routing regions which reflect the current crosstalk situation of the chip. These partitioned bounds in each routing region provide necessary constraints (which are otherwise not available) for the constrained crosstalk optimization of each routing region at later stages in the layout process (Fig. 4.3).

The entire post global routing optimization approach iterates among three key components (Fig. 4.4): crosstalk risk estimation, risk tolerance bound partitioning and global routes adjustment. The region-based crosstalk risk estimation first constructs a crosstalk risk graph for each routing region representing its crosstalk situation based on the initial partitions of nets' risk tolerance bounds. The crosstalk risk of the region, which indicates whether a risk-free routing solution is possible, is then quantitatively defined and estimated using a graph-based optimization approach. For accurate risk estimation, the impact of bound changes on regions' risks is analyzed and the current partitions of nets' risk tolerance bounds are adjusted via integer linear programming to minimize the positive risks on the chip. If positive risk regions still exist after bound partitioning, global routes adjustment is applied. First, nets whose removal leads to the elimination of positive risks of the chip are identified, then they are ripped-up and re-routed with minimum cost alternative routes which consider both the routing congestions and crosstalk risks of routing regions on the chip. The entire iterative optimization process continues until a risk-free global routing solution is

obtained.

The rest of this chapter focuses on the methods for crosstalk risk estimation (the bound partitioning and risk reduction problem are discussed in Chapter 5) and is organized as follows: Section 4.2 presents the graph-based crosstalk representation; Section 4.3 introduces the quantitative definition of crosstalk risk of each routing region; Section 4.4 analyzes the region-based crosstalk risk estimation methods; Section 4.5 shows some experimental results of risk estimation via graph construction.

4.2 Crosstalk Risk Representation

4.2.1 Definitions

For graph-based crosstalk risk representation at the global routing level, $G = (V, E)$ is defined as a regular-grid *global routing graph* imposed on a chip (Fig. 4.5), in which each edge $e \in E$ represents a horizontal or vertical *routing region* on a routing layer. Denote N as the set of nets routed on the chip. Under global routing formulation, the pins of each net $n \in N$ are mapped onto node set V of G, i.e., $n \in V$, and the route of n consists of a series of routing regions on the chip, i.e., $route(n) \subseteq E$. The set of nets routed in e is denoted as $N(e)$. According to this formulation, each net $n \in N(e)$ is allowed to to be routed in only one direction, either horizontally or vertically within each region $e \in route(n)$ and it occupies an entire track in the region. Once the size, wire pitch of the chip and the graph pitch of G are known, the number of available routing tracks in region e can be specified, which is denoted as the capacity of e, $C(e)$. The crosstalk synthesis discussed in this thesis focuses on the intra-layer crosstalk noise between parallel coupled wires on the same layer, and it considers every routing region of a chip simultaneously during optimization for both two-layer and multi-layer design styles.

Although coupling capacitance exists between every pair of adjacent nets, crosstalk noise between some adjacent net pairs may not affect the proper functioning of the circuit due to logical and temporal isolations [Kirkpatrick 94]. For example, net i may be immune from the noise spike caused by the signal switch on its adjacent net j if these two nets are not active at the same time. This implies that not every pair of nets is subject to crosstalk concern during crosstalk synthesis and the *crosstalk sensitivity*, S_{ij}, can be specified for each net pair (i, j). For digital circuits, $S_{ij} \in \{0, 1\}$ and $S_{ij} = 1$ implies that nets i, j are subject to crosstalk concern during optimization, otherwise, they are regarded as "crosstalk-safe" when routed in adjacent tracks. S_{ij} can also be a real number,

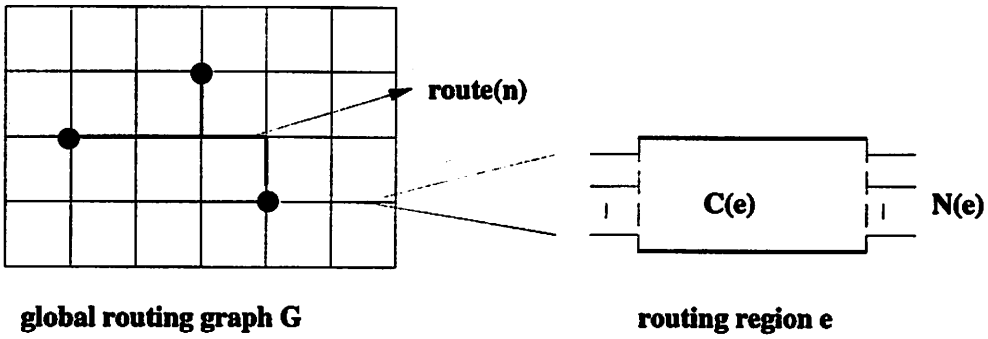

Figure 4.5: Global Routing Formulation

representing the amount of interactions between nets i, j. Once S_{ij}s are known, *sensitive net* set $N_s \subseteq N$ is defined as the set of nets that are sensitive to at least one other net on the chip, i.e., $N_s = \{i | \exists j \in N, s.t.\ S_{ij} \neq 0\}$ and $N_s(e) \subseteq N(e)$ is defined as the set of sensitive nets routed in region e, i.e., $N_s(e) = \{i \in N(e) | \exists j \in N(e), s.t.\ S_{ij} \neq 0\}$.

In our discussion, it is assumed that crosstalk noise exists only between net pairs routed in adjacent tracks in a region, noise between nets one or more tracks apart can be ignored. Denote $noise(i, j)$ as the crosstalk noise between adjacent net pair i, j. Since it is determined by the coupling capacitance between i, j, which is directly proportional to their coupling wire length, $noise(i, j)$ can be measured by:

$$noise(i, j) = S_{ij} len(i, j) \tag{4.1}$$

Here, $len(i, j)$ is the potential coupling wire length between nets i and j, if they are placed in adjacent tracks.

Define $Bound(i)$ as the *risk tolerance bound* of sensitive net $i \in N_s$, i.e., the maximum amount of crosstalk noise i can tolerate without affecting the functionality of the circuit. Thus, net i is "safe" from crosstalk violations if and only if the summation of noises from all of its adjacent nets along its route is less than $Bound(i)$, i.e.,

$$\sum_{e \in route(i)} \sum_{j \in Adj(i,e)} noise(i, j, e) = \sum_{e \in route(i)} \sum_{j \in Adj(i,e)} S_{ij} len(i, j, e) < Bound(i) \tag{4.2}$$

where $Adj(i, e)$ is the set of nets adjacent to net i in region e, $len(i, j, e)$ and $noise(i, j, e)$ are the potential coupling length and crosstalk noise between nets i, j respectively if they are adjacent in region e. Both the sensitivity information and risk tolerance bounds of nets can be extracted using temporal and functional analysis or specified by user. They are given as input to our crosstalk optimization process together with a feasible global routing solution of the chip (Fig. 4.3).

Since crosstalk noise at net i comes from all regions on its route according to Eqn (4.2), $Bound(i)$ must be partitioned accordingly among $route(i)$ for region-based crosstalk estimation and constrained optimization of each routing region on the chip. Denote $Bound(i, e)$ as the partitioned risk tolerance bound of net i in routing region $e \in route(e)$. The partition of $Bound(i)$ can then be expressed as:

$$Bound(i) = \sum_{e \in route(i)} Bound(i, e) \tag{4.3}$$

The risk tolerance bound partitioning methods for accurate risk estimation is discussed in Chapter 5.

4.2.2 Crosstalk Violations and Risk-free Routing Solution

Although the crosstalk noise at each net i in region e, determined by its couplings with its adjacent nets, can be calculated exactly only from a detailed routing solution of e, we can identify whether there a routing solution of e may exist in which each sensitive net is free of crosstalk violations once a global routing solution of e is obtained and $N_s(e), C(e)$ are known.

4.2.2.1 Crosstalk Violations

Since each net routed in region e occupies an entire track in the region under global routing formulation, no two nets share the same track in e. Therefore, each net $i \in N_s(e)$ can be adjacent to no more than two nets in its above and below tracks in e and the crosstalk violation may occur at net i in region e only in the following two cases:

- Case 1. The noise from one of i's adjacent nets in e violates its risk tolerance bound, i.e., $\exists j \in Adj(i, e)$ s.t. $noise(i, j, e) \geq Bound(i, e)$
- Case 2. The summation of noises from both of i's adjacent nets in e violates its bound, i.e., $\exists j, k \in Adj(i, e)$ s.t. $noise(i, j, e) + noise(i, k, e) \geq Bound(i, e)$

These two cases are referred to as crosstalk violation Case 1 and 2 in later discussions.

4.2.2.2 Risk-Free Global Routing Solution

In our following discussion, a *routing solution of region* e at the global routing level is defined as a routing order of nets in $N(e)$ in adjacent tracks (including empty ones) in e ranked from one side of the region to the other. If there exists a routing order of region e according to which each

net is free of crosstalk violation, it is denoted as a *risk-free routing solution* of e and e is defined as *risk-free* (nets that cause crosstalk violations at i under Case 1 and 2 can not be placed adjacent to i in a risk-free routing solution of region e). If every region on the chip is risk-free, the current global routing solution of the chip is defined as risk-free. Although multiple risk-free solutions of a region may exist at the global routing level, the goal of our region-based crosstalk risk estimation method is to identify the existence of one such solution for each region on the chip, not to find a specific one. Notice that the risk-free solution defined here at the global routing level is for the use of risk estimation and reduction purpose only, it does not necessarily correspond to the final routing solution of the region which is to be generated at later stages in the layout process.

4.2.3 Graph-Based Crosstalk Representation

To identify the existence of a risk-free global routing solution of each routing region, we introduce two types of crosstalk risk graphs to represent its crosstalk situation considering violation Case 1 and 2 defined in Sec. 4.2.2.1.

4.2.3.1 Crosstalk Risk Graph

From the global routing solution of a region e (Fig. 4.6(a)), its capacity $C(e)$ and sensitive nets set $N_s(e)$ are known. For the crosstalk representation of region e, a weighted *crosstalk risk graph*, $CRG(e) = (N_s(e), E_s(e))$ is first introduced (Fig. 4.6(b)). Each node i in $CRG(e)$ corresponds to a sensitive net routed in e, its weight represents the partitioned risk tolerance bound of net i in e, $B(i, e)$. The weight of each edge between nodes i, j is the potential crosstalk noise between nets i, j in region e, $noise(e)$, if they are routed in adjacent tracks, and edge $(i, j) \in E_s(e)$ if and only if $noise(i, j, e)$ is less than the partitioned risk tolerance bound of both nets in region e, i.e.,

$$noise(i, j, e) < Bound(i, e) \quad and \quad noise(i, j, e) < Bound(j, e) \tag{4.4}$$

According to Eqn (4.4), each edge (i, j) in $CRG(e)$ implies that nets i, j can be routed in adjacent tracks without causing crosstalk violations at nets i, j. Therefore, the crosstalk representation by $CRG(e)$ excludes crosstalk violations under Case 1. However, the net compatibility represented by $CRG(e)$ is only pair-wise, i.e., the fact that nets j, k are compatible with net i separately does not guarantee they can be placed adjacent to i at the same time, since the summation of noises from j, k may cause crosstalk violation at i under Case 2. For example, although both nets c, d can be placed adjacent to a separately according to Fig. 4.6(b), the total noise from c, d

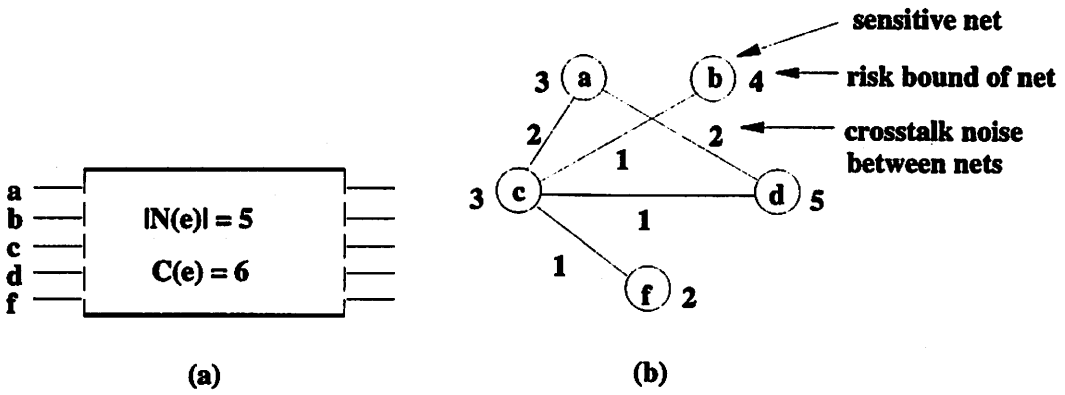

Figure 4.6: (a) Routing Solution of Region e (b) $CRG(e)$

violates the risk tolerance bound of net a in e and thus they are not allowed to be adjacent to a simultaneously in a risk-free routing solution of e.

4.2.3.2 Constrained Simple Path Sub-graph

For further crosstalk representation, we define a *constrained simple path sub-graph* of $CRG(e)$, $CRG_{csp}(e) = (N_s(e), E_p(e))$ where $E_p(e) \subseteq E_s(e)$ (Fig. 4.7). $CRG_{csp}(e)$ contains simple path segments only (isolated nodes are regarded as special simple path segments), i.e., $degree(i) \leq 2$ holds for every node i in $CRG_{csp}(e)$. In addition, the weight of each node $i \in CRG_{csp}(e)$ is larger than the summation of weights of its adjacent edges (which are no more than 2). In other words, the summation of the noises from nets j, k is less than the partitioned risk tolerance bound of net i in e, if nodes j, k are adjacent to node i in $CRG_{csp}(e)$, i.e.,

$$noise(i, j, e) + noise(i, k, e) < Bound(i, e), \quad (i, j), (i, k) \in E_p(e), \forall i \in N_s(e) \tag{4.5}$$

According to Eqn (4.5), crosstalk violation under Case 2 is also excluded for each net with respect to its adjacent ones according to the crosstalk representation by $CRG_{csp}(e)$. One important difference between $CRG(e)$ and $CRG_{csp}(e)$ is that, $CRG(e)$ is uniquely determined once the crosstalk information and the routing solution of e are known, while numerous $CRG_{csp}(e)$s of $CRG(e)$ exist as its constrained simple path sub-graph. For example, the minimum $CRG_{csp}(e)$ of $CRG(e)$ is the one consisting of isolated nodes only with no edges. On the contrary, our focus in crosstalk estimation is to construct a $CRG_{csp}(e)$ from $CRG(e)$ having the maximum number of edges, as discussed later in Sec. 4.4.

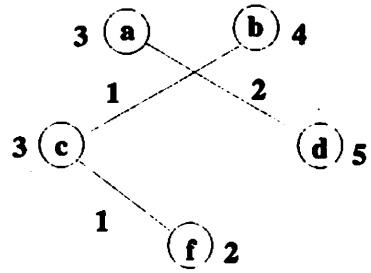

Figure 4.7: Constrained Simple Path Sub-graph $CRG_{csp}(e)$

4.3 Region-based Crosstalk Risk Definition

Based on the crosstalk risk graphs introduced above, our region-based crosstalk synthesis method quantitatively defines and estimates the crosstalk risk of each region as a whole at the global routing level.

4.3.1 Risk-free Routing Solution vs. Crosstalk Risk Graph

As previously analyzed, the objective of region-based estimation is to determine whether there exists a routing solution of each routing region on the chip in which every net is free of crosstalk violations under Case 1 and 2. According to the definition of $CRG_{csp}(e)$, every net i in region e is free of crosstalk violations if it is routed in adjacent tracks with nets j, k under the condition that nodes j, i, k are adjacent in that order on a simple path segment in a $CRG_{csp}(e)$. Therefore, nets $n_1, \ldots, n_p$ are crosstalk risk-free in region e if they are routed in e in the same order as their corresponding nodes appear on a simple path segment $p = (n_1, \ldots, n_p)$ in a $CRG_{csp}(e)$, i.e., each simple path segment $p \in CRG_{csp}(e)$ corresponds to a risk-free routing order of a subset of nets in $N_s(e)$. For example, path segment $p = (b, c, f)$ in Fig. 4.7 corresponds to a risk-free routing order of nets b, c, f in the region.

In graph theory, a *Hamiltonian path* in graph G is defined as a special simple path segment that visits every node in G exactly once. According to this definition, a $CRG_{csp}(e)$ is a Hamiltonian path itself if it contains just one simple path segment (recall that isolated nodes are regarded as special segments). From the definition of a routing solution in global routing in Sec. 4.2.2.2, a Hamiltonian path in $CRG_{csp}(e)$ corresponds to a risk-free routing solution of all the sensitive nets $N_s(e)$ routed in e. Thus, region e is identified as risk-free if a Hamiltonian path exists in one of the $CRG_{csp}(e)$s of $CRG(e)$.

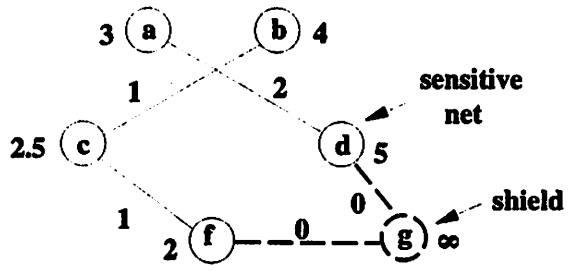

Figure 4.8: Construction of Hamiltonian path using shield

Proposition 4.1 *A routing region e is risk-free in global routing if one of its $CRG_{csp}(e)$s has a Hamiltonian path.*

4.3.2 Shields

By its definition, a Hamiltonian path in $CRG_{csp}(e)$ is a simple path segment having $|N_s(e)| - 1$ edges. Due to the configuration of $CRG(e)$ of region e, it is not always possible to find a $CRG_{csp}(e) \subseteq CRG(e)$ which contains a Hamiltonian path. For example, no $CRG_{csp}(e)$ may contain a Hamiltonian path if $CRG(e)$ itself has less than $|N_s(e)| - 1$ edges. When multiple simple path segments exist in a $CRG_{csp}(e)$ of region e, the nets corresponding to the end nodes of these path segments can not be routed adjacent to each other in region e free of crosstalk violations under Case 1 or 2.

To generate a risk-free routing solution of the region, we introduce the concept of *shields*, which are the non-sensitive nets or empty tracks in the region, each having zero crosstalk with other nets and infinite risk tolerance bound. These shields can be used to separate those nets corresponding to the end nodes of the simple path segments in $CRG_{csp}(e)$ so that they are no longer subject to crosstalk violations. From graph point of view, each shield s can "connect" two disjoint simple path segments p_1 and p_2 in a $CRG_{csp}(e)$ into a longer path segment, $p_1 \cup \{s\} \cup p_2$, which corresponds to a risk-free routing order of nets on both p_1 and p_2. Therefore, if there are enough shields in region e to connect all the simple path segments in $CRG_{csp}(e)$ together as one, a risk-free routing solution of region e can be found and e is risk-free in global routing.

Fig. 4.8 shows an example of shield application. Two disjoint path segments (a, d) and (b, c, f) are connected together via shield node g to form a Hamiltonian path, which corresponds to a risk-free routing solution of the region in the order of $(a, d, \mathrm{g}, f, c, b)$.

4.3.3 Analytical Crosstalk Risk Definition

4.3.3.1 Shield Estimation

Denote $P(e)$ as the number of simple path segments in a $CRG_{csp}(e)$ of $CRG(e)$, $S_{avail}(e)$ as the number of shields available in region e and $S_{need}(e)$ as the number of shields needed in e to generate a risk-free routing solution of the region. According to the shield definition, $S_{avail}(e)$ equals the total number of empty tracks and non-sensitive nets in e and can be expressed as:

$$S_{avail}(e) = C(e) - |N_s(e)| \tag{4.6}$$

To estimate $S_{need}(e)$ in region e, the following lemma is introduced first:

Lemma 4.1

$$P(e) = |N_s(e)| - |E_p(e)| \tag{4.7}$$

where $N_s(e), E_p(e)$ are node and edge set of a $CRG_{csp}(e)$, respectively.

Proof:

Consider an initial graph G consisting of $|V|$ isolated nodes only, which corresponds to $|V|$ simple path segments. Since each edge links two path segments (including isolated nodes) together as one, it can reduce the number of simple path segments in G by 1. Therefore, for a $CRG_{csp}(e)$ having $|E_p(e)|$ edges, the number of its simple path segments is reduced by $|E_p(e)|$ from its initial value $|N_s(e)|$, i.e.,

$$P(e) = |N_s(e)| - |E_p(e)|$$

□

According to Lemma 4.1, $S_{need}(e)$ of region e, which is determined by the configuration of $CRG_{csp}(e)$, can be calculated by the following theorem:

Theorem 4.1

$$S_{need}(e) = |N_s(e)| - |E_p(e)| - 1 \tag{4.8}$$

Proof:

By its definition, $S_{need}(e)$ is the number of shields needed in region e to connect all simple path segments in $CRG_{csp}(e)$ into a Hamiltonian path. Since each shield can "adhere" two disjoint

simple path segments together as one and thus reduce the number of path segments in $CRG_{csp}(e)$ by one, $S_{need}(e)$ can be expressed as:

$$S_{need}(e) = P(e) - 1 \tag{4.9}$$

According to Eqn (4.7), we have:

$$S_{need}(e) = |N_s(e)| - |E_p(e)| - 1$$

□

4.3.3.2 Risk Definition

According to Theorem 4.1, the number of shields needed for a risk-free routing solution of region e is related to the number of simple path segments in one of its $CRG_{csp}(e)$s. The possible $CRG_{csp}(e)$s of a $CRG(e)$ are not unique, plus, there may exist multiple $CRG_{csp}(e)$s having the same number of edges $|E_p(e)|$. For risk estimation, we are interested in the existence of a risk-free routing solution of region e, which is determined by the number of shields currently available, $S_{avail}(e)$, and the minimum number of shields needed in e, $S_{need-min}(e)$. $S_{need-min}(e)$ can be estimated based on a special $CRG_{csp}(e)$ of $CRG(e)$, $CRG_{csp-max}(e)$, having the maximum number of edges, $|E_{p-max}(e)|$. Therefore, the risk of region e, $Risk(e)$ is quantitatively defined as the difference between $S_{need-min}(e)$ and $S_{avail}(e)$, which can be uniquely specified for region e once its $CRG(e)$ is known.

$$Risk(e) = S_{need-min}(e) - S_{avail}(e) = 2|N_s(e)| - |E_{p-max}(e)| - C(e) - 1 \tag{4.10}$$

The example in Fig. 4.7 corresponds to a $CRG_{csp-max}(e)$ of $CRG(e)$ of region e shown in Fig. 4.6, having $C(e) = 6$, $|N_s(e)| = 5$ and $|E_{p-max}(e)| = 3$. Thus, $S_{avail}(e) = S_{need-min}(e) = 1$ and $Risk(e) = 0$, i.e., there exist a risk-free routing solution of the region using a shield as shown in Fig. 4.8.

$Risk(e)$ indicates whether region e is risk-free in global routing. If $Risk(e) \leq 0$, there are more than enough shields in region e to generate a risk-free routing solution of e and e is defined as risk-free. Otherwise, $Risk(e)$ is the minimum number of extra shields needed in e for a risk-free routing solution, which should be minimized during the risk reduction phase discussed in Chapter 5. According to this analysis, the following proposition can be established.

Proposition 4.2 *The current global routing solution of the chip is crosstalk risk-free if and only if* $Risk(e) \leq 0$ *holds for every routing region e on the chip.*

4.4 Crosstalk Risk Estimation

4.4.1 Problem Analysis

According to Eqn (4.10), the key to the crosstalk risk estimation of region e is to identify $|E_{p-max}(e)|$, the maximum number of edges possible in a constrained simple path sub-graph of $CRG(e)$, $CRG_{csp-max}(e)$. Similar to the non-uniqueness of Hamiltonian path in a graph, there may exist multiple $CRG_{csp-max}(e)$s of $CRG(e)$, all having $|E_{p-max}(e)|$ edges. For crosstalk risk estimation, one of these $CRG_{csp-max}(e)$s is constructed. The construction of a $CRG_{csp-max}(e)$ from $CRG(e)$ can be formulated as a generalized approach for finding a Hamiltonian path in a graph and the following theorem holds:

Theorem 4.2 *The construction of a $CRG_{csp-max}(e)$ from $CRG(e)$, i.e., the crosstalk risk estimation problem, is NP-complete.*

Proof:

We establish the proof by reducing the Hamiltonian path problem for an arbitrary graph to the $CRG_{csp-max}(e)$ construction problem in polynomial time.

A Hamiltonian path in graph G is the largest possible maximum simple path sub-graph G_{sp-max} of G, since it has only one simple path connecting all nodes in G. In other words, if a Hamiltonian path exists in G, it is also a G_{sp-max} of G and can be found via G_{sp-max} construction method. Therefore, the problem of finding a Hamiltonian path in G can be reduced in polynomial time to the problem of constructing a $CRG_{csp-max}(e)$ from $CRG(e)$ by setting $CRG(e)$'s nodes' weight to infinity and edges' weight to 1, which effectively eliminates the noise constraints in $CRG_{csp-max}(e)$.

Since no transitive relations hold for the sensitivities and coupling wire length between net pairs, i.e., S_{ij} is independent of S_{ik} and S_{jk}, $len(i,j,e)$ is independent of $len(i,k,e)$ and $len(j,k,e)$, the potential crosstalk noise $noise(i,j,e)$ between nets i,j in region e is also independent of $noise(i,k,e)$ and $noise(j,k,e)$. This implies that the possible edges and their weights among nodes in $CRG(e)$ are not related to each other, i.e., $CRG(e)$ is a totally arbitrary graph.

Since finding a Hamiltonian path in an arbitrary graph is known to be NP-complete, the construction of a $CRG_{csp-max}(e)$ from $CRG(e)$, i.e., the risk estimation of region e is also NP-complete.

□

4.4.2 Crosstalk Risk Estimation Algorithm

Due to the non-uniqueness of $CRG_{csp-max}(e)$s of $CRG(e)$ and the various possible ways of applying shields to connect the simple path segments in them, there may exist many possible risk-free routing solutions of region e. For crosstalk risk estimation, we are interested in the existence of a risk-free routing solution of region e rather than finding a specific one, which is the task of later stages in the layout process.

Due to the NP-complete nature of the crosstalk estimation problem, we develop a two-step iterative algorithm to construct a $CRG_{csp-max}(e)$ from $CRG(e)$. First, we construct an initial $CRG_{csp-max}(e)$ by sequentially removing minimum number of edges from $CRG(e)$. Second, the graph is iteratively improved to include more edges so that local optimal solution can possibly be avoided. Since we are not confined to keep any specific configuration of $CRG_{csp-max}(e)$ for risk estimation, edges can be inserted or removed from $CRG_{csp}(e)$ freely during the graph construction process.

4.4.2.1 Initial $CRG_{csp-max}(e)$ Construction

Define the degree of edge (i,j) in $CRG(e)$, $degree(i,j)$, as the summation of its node degrees in $CRG(e)$, i.e., $degree(i,j) = degree(i) + degree(j)$. If $degree(i,j) > 4$, the degree of at least one of i,j is larger than 2, which is not allowed in $CRG_{csp-max}(e)$.

For the construction of the initial $CRG_{csp-max}(e)$, edges are removed sequentially from $CRG(e)$ until the degree of each node is no more than 2 and the noise constraints for $CRG_{csp-max}(e)$ are also satisfied. According to the definition of edge degree, those edges having the largest degrees should be removed from $CRG(e)$ first in order to minimize the number of edges which need to be deleted under the node degree constraints for $CRG_{csp-max}(e)$. To this end, the following two heuristics are adopted:

H1. Remove edges with the largest degrees first.

H2. Among edges having the same degree, remove those having the largest weight (noise) first.

H2 is applied here in order to speed up the satisfaction of noise constraints for $CRG_{csp-max}(e)$.

Denote the set of edges removed during the initial $CRG_{csp-max}(e)$ construction as $E_{rem}(e)$, the algorithm can then be outlined as follows:

Initial $CRG_{csp-max}(e)$ Construction Algorithm {

1. Set initial graph $G(e) = CRG(e)$.
2. While there exists node i in current $G(e)$ with $degree(i) > 2$:
   2.1 Compute degrees of edges in $G(e)$.
   2.2 Remove edges from $G(e)$ according to H1 and H2.
3. While there still exist crosstalk violations at nodes:
   Remove edges from $G(e)$ according to H2.
4. Output $G(e)$ as the initial $CRG_{csp-max}(e)$.

4.4.2.2 Iterative $CRG_{csp-max}(e)$ Improvement

In Sec. 4.4.2.1, the initial $CRG_{csp-max}(e)$ is constructed from $CRG(e)$ in a greedy fashion. To avoid a local optimal solution, we design a two-phase improvement process which incrementally adds new edges into the initial $CRG_{csp-max}(e)$. These two phases iterate until no further increase in the number of edges in current $CRG_{csp-max}(e)$ can be obtained.

Phase I:

Since edges are removed sequentially from $CRG(e)$ during the initial construction step, some removed edges in the process can possibly be "safely" added back to the current $CRG_{csp-max}(e)$. Therefore we check if any previously removed edges in $E_{rem}(e)$ can now be re-inserted into $CRG_{csp-max}(e)$ without violating its node degree and noise constraints. The complexity of this phase is the size of $E_{rem}(e) \subseteq E_s(e)$, i.e., $O(|E_s(e)|)$.

Phase II:

To further improve the $CRG_{csp-max}(e)$ obtained after Phase I, we apply the so-called k-Opt heuristics in Phase II, which is similar to the one used by [Johnson 90] to solve the Traveling Salesman Problem. k-Opt checks whether more than k previously removed edges can be added back to the current $CRG_{csp-max}(e)$ when k edges randomly picked from it are removed. A successful application of k-Opt results in at least one more edges in $CRG_{csp-max}(e)$. If k-Opt heuristics is applied with k ranging from 1 to $|E_s(e)| - 1$, i.e., an exhaustive search on all possible edge configurations, a globally optimal $CRG_{csp-max}(e)$ can be found. However, this is not feasible in practice due to the $O(|E_s(e)|^{k+1})$ complexity of k-Opt. In typical global routing situations, each

routing region contains 10-30 tracks and routed nets, thus, the number of nodes in the corresponding $CRG(e)$ and $CRG_{csp-max}(e)$ is also within the same range. For graphs of this size, 1-Opt and 2-Opt are sufficient to yield excellent results for iterative $CRG_{csp-max}(e)$ improvement as shown by the experimental results in Sec. 4.5.1.

4.5 Experimental Results

The risk estimation method discussed in previous sections has been implemented on a DEC 5000 workstation. To verify it effectiveness, we tested separately the graph construction approach on graph examples and the region-based estimation method under various crosstalk specifications on a circuit from the CBL/NCSU building-block benchmarks.

4.5.1 Examples of Graph Construction

We first apply the two-phase $CRG_{csp-max}(e)$ construction method to three graph examples adopted as $CRG(e)$s in our experiment. These graphs are often used in the study of the Hamiltonian path problem in graph theory. Since the focus in this part of experiment is on the construction of a maximum simple path sub-graph of these examples using the proposed two-step method, we ignore the crosstalk noise constraints for $CRG_{csp-max}(e)$ by setting the weight of its edges and nodes to 1 and ∞, respectively. Risk estimation results under various crosstalk specifications are investigated in Sec. 4.5.2.

4.5.1.1 Test Example One

Fig. 4.9 shows the $CRG(e)$ of a fully-routed routing region having 20 sensitive nets and routing tracks. After the initial $CRG_{csp-max}(e)$ construction step in Sec. 4.4.2.1, one of the Hamiltonian paths of the graph is found which has 19 edges (Fig. 4.10). According to the risk definition in Eqn (4.10), $Risk(e) = 2 * 20 - 19 - 20 - 1 = 0$, i.e., region e is crosstalk risk-free.

4.5.1.2 Test Example Two

The $CRG(e)$ shown in Fig. 4.11 is for a routing region having 15 sensitive nets and routing tracks. Fig. 4.12 shows its initial $CRG_{csp-max}(e)$ constructed as in Sec. 4.4.2.1, which contains two disjoint simple path segments and 13 edges. The initial risk estimation is: $Risk(e) = 2 * 15 - 13 - 15 - 1 = 1$, indicating one more shield is needed in region e for a risk-free

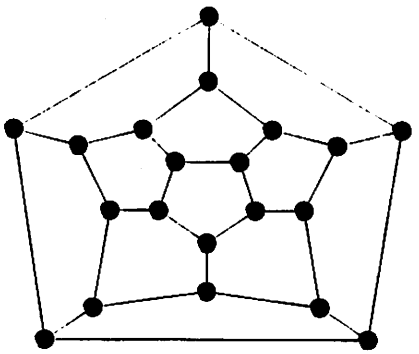

Figure 4.9: $CRG(e)$ of Test Example One

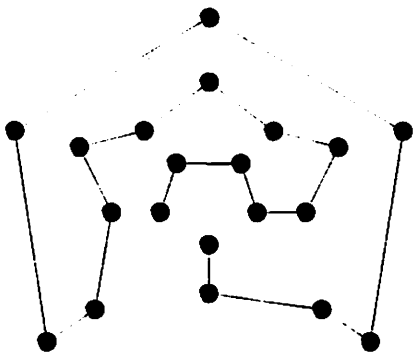

Figure 4.10: A $CRG_{csp-max}(e)$ of Test Example One

routing solution. For more accurate estimation on the region's risk, the iterative $CRG_{sp-max}(e)$ improvement method in Sec. 4.4.2.2 is used. When 1-opt is applied to the initial $CRG_{csp-max}(e)$, two edges, b and c, which are removed during initial $CRG_{csp-max}(e)$ construction, can be added back into $CRG_{csp-max}(e)$ after edge a is deleted from it. As a result, the number of edges in $CRG_{csp-max}(e)$ is increased by 1 and a Hamiltonian path of $CRG(e)$ is found (Fig. 4.13). The final risk estimation based on the improved graph becomes: $Risk(e) = 2 * 15 - 14 - 15 - 1 = 0$, indicating that region e is risk-free in global routing.

4.5.1.3 Test Example Three

As analyzed in previous sections, it is not always possible to find a Hamiltonian path in a $CRG(e)$ even when the two-step $CRG_{csp-max}(e)$ construction method is applied. Fig. 4.14 shows the $CRG(e)$ of a routing region e having 16 sensitive nets and tracks. Its final $CRG_{csp-max}(e)$ after improvement (Fig. 4.15) still contains 3 disjoint simple path segments. This indicates that at least 2 more shields are needed in region e for a risk-free routing solution of e, i.e., region e should

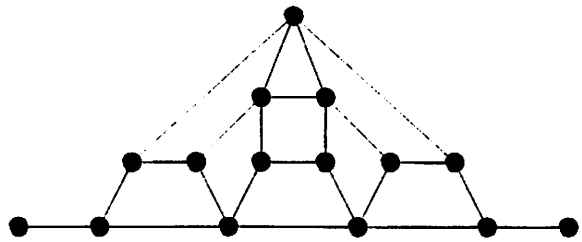

Figure 4.11: $CRG(e)$ of Test Example Two

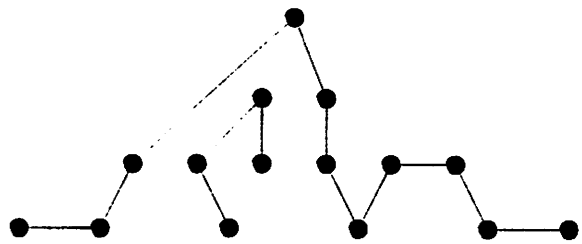

Figure 4.12: An Initial $CRG_{csp-max}(e)$ of Test Example Two

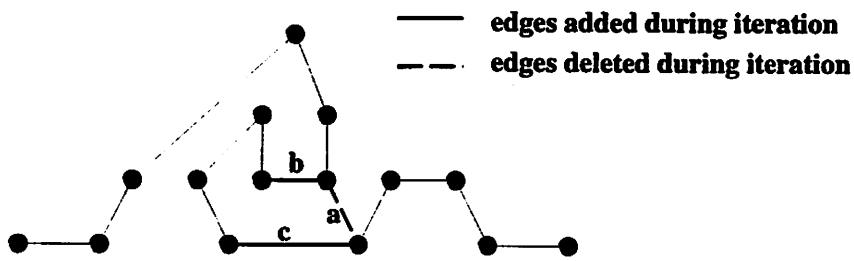

Figure 4.13: A Final $CRG_{csp-max}(e)$ of Test Example Two After Iterative Improvement

have at least 18 tracks to be risk-free in global routing.

4.5.2 A Testing Circuit

For crosstalk estimation of an entire circuit, we test the region-based risk estimation method on a global routing solution of circuit $ami33$ which is one of the testing circuits constructed from the CBL/NCSU building-block benchmarks and used in our experiments. The specification of $ami33$ is listed in Table 4.5.2, where G_{size} refers to the size of global routing graph of the chip specified by its number of rows and columns. The number of routing regions in global routing equals $rowxcol$.

The feasible placement/global routing solution of the chip is generated by a performance-driven placement [Esbensen 96] and a global router [Wang 96], respectively. In our experiments,

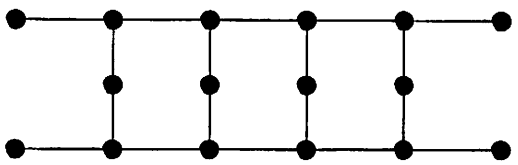

Figure 4.14: $CRG(e)$ of Test Example Three

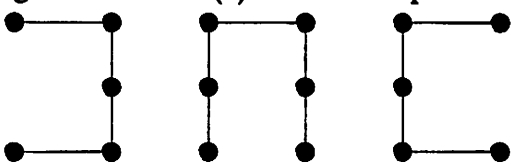

Figure 4.15: A Final $CRG_{csp-max}(e)$ of Test Example Three

Table 4.2: Specifications of Circuit $ami33$

Circuit	# macro cells	# nets	# pins	G_{size} (row x col)
ami33	33	123	442	28 x 23

two types of crosstalk specifications are used:

- *Net sensitivity ratio:* the percentage of net pairs in the circuit that are subject to crosstalk risk concern (i.e., $S_{ij} = 1$) during the optimization.
- *Risk tolerance bound of net:* the percentage of total wire length of each net that is allowed to be coupled with its sensitive nets during optimization.

Since there is no standard benchmark having these crosstalk information, our approach estimates the average crosstalk risk of all routing regions on the chip under various possible values of both net sensitivity ratio, ranging from 60% to 100%, and the risk tolerance bound of each net, ranging from 5% to 100%.

Fig. 4.16 illustrates how the average risk of regions on $ami33$ changes under different net sensitivity ratios and risk tolerance bounds. Here, the results are measured with the risk tolerance bound of each net partitioned uniformly among its routing regions (results under more accurate bound partitionings are shown in Chapter 5). It can be observed that the crosstalk risk decreases as the risk tolerance bound increases and net sensitivity ratio decreases. This is due to the fact that nets having larger bounds are less vulnerable to crosstalk violations and fewer shields, whose number equals positive risk, are needed in each region on the chip when fewer net pairs are subject

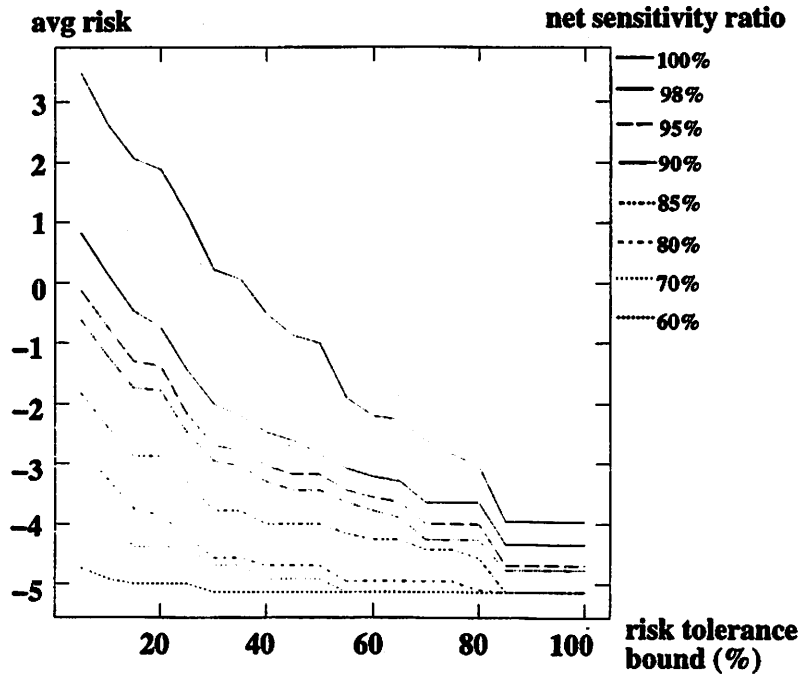

Figure 4.16: Region-based Crosstalk Estimation under Various Parameters

to crosstalk concern. The risk estimation of the entire circuit takes approximately 1 second of CPU time.

The testing results on *ami33* illustrate the relation between regions' risks vs. the sensitivities and risk tolerance bounds of nets in a circuit. Region-based risk estimation is the core of both the risk tolerance bound partitioning and risk reduction methods that are to be discussed in Chapter 5, more testing results involving the risk estimation method discussed in this Chapter will be presented in Sec. 5.5.

Chapter 5

Crosstalk Risk Reduction at the Global Routing Level

5.1 Introduction

Once the crosstalk risk of each routing region on the chip is estimated using the graph-based approach in Chapter 4, those regions having positive risks can be identified. According to the risk definition in Eqn (4.10), the total positive risk of these regions, $\sum(Risk(e) > 0)$, equals the total number of extra shields needed to generate a risk-free global routing solution of the chip. The objective of crosstalk reduction at the global routing level can then be stated as:

Eliminate those positive risk regions so that every routing region on the chip has a risk-free global routing solution.

5.1.1 Approaches to Crosstalk Risk Reduction

As analyzed in Chapter 4, there exists a one-to-one correspondence between the crosstalk risk graph of region e, $CRG(e)$, and its risk estimation, $Risk(e)$. In other words, once the $CRG(e)$ of region e is constructed, the number of edges $|E_{p-max}(e)|$ in its maximum constrained simple path sub-graph $CRG_{sp-max}(e)$ can be computed and $Risk(e)$ is uniquely determined by Eqn (4.10):

$$Risk(e) = 2|N_s(e)| - |E_{p-max}(e)| - C(e) - 1$$

Since the capacity of region e, $C(e)$, is fixed during global routing, the two adjustable variables in $Risk(e)$ are:

1. The number of edges in $CRG_{sp-max}(e)$, $|E_{p-max}(e)|$, which can be estimated once $CRG(e)$ is known.

2. The number of sensitive nets routed in region e, $|N_s(e)|$, determined by the global routing solution of the chip.

Notice that the configuration of $CRG(e)$ is determined by the partitioned risk tolerance bounds $B(i,e)$s of sensitive nets in $N_s(e)$, once the global routing solution and the sensitivities between net pairs are known. Therefore, there are two ways to reduce $Risk(e)$ of positive risk region e:

1. Adjust the partitioned risk tolerance bounds $B(i,e)$s so that more edges can be added into $CRG(e)$ and $CRG_{sp-max}(e)$, i.e., larger $|E_{p-max}(e)|$ value. During this bound partitioning process, the current routing solution of the chip remains unchanged. This approach is discussed in detail in Sec. 5.3.

2. Change the current routing solution by reducing the number of sensitive nets $N_s(e)$ routed in e via net ripping-up and re-routing (Although reducing $|N_s(e)|$ may also affect $|E_{p-max}(e)|$, it does not cause increase in $Risk(e)$ as analyzed in Sec. 5.4). This approach is discussed in detail in Sec. 5.4.

5.1.2 Examples

We use examples to further illustrate how the risk tolerance bound partitioning and net's ripping-up and re-routing affect the configuration of CRG_{sp-max}s and the risk estimation of routing regions.

Fig. 5.1 shows the CRG_{sp-max}s of two routing regions, Region 1 and 2. They have 5 and 4 routed sensitive nets respectively and each has 5 routing tracks, i.e., $C(1) = C(2) = 5$, $N_s(1) = 5, N_s(2) = 4$. Sensitive net f is routed through both regions, which has a total risk tolerance bound of 5 units, $Bound(f) = 5$. The CRG_{sp-max}s in Fig. 5.1 are configured under Partition One of $Bound(f)$ among Region 1 and 2: $Bound_1(f) = Bound_1(f,1) + Bound_1(f,2) = 2+3$. According to the risk definition in Eqn (4.10), $Risk_1(1) = 2*5-3-5-1 = 1 > 0$ and $Risk_1(2) = 2*4-3-5-1 = -1 < 0$, i.e., Region 2 is risk-free under Partition One of $Bound(f)$, while Region 1 is not, which needs one extra shield in it to have a risk-free routing solution.

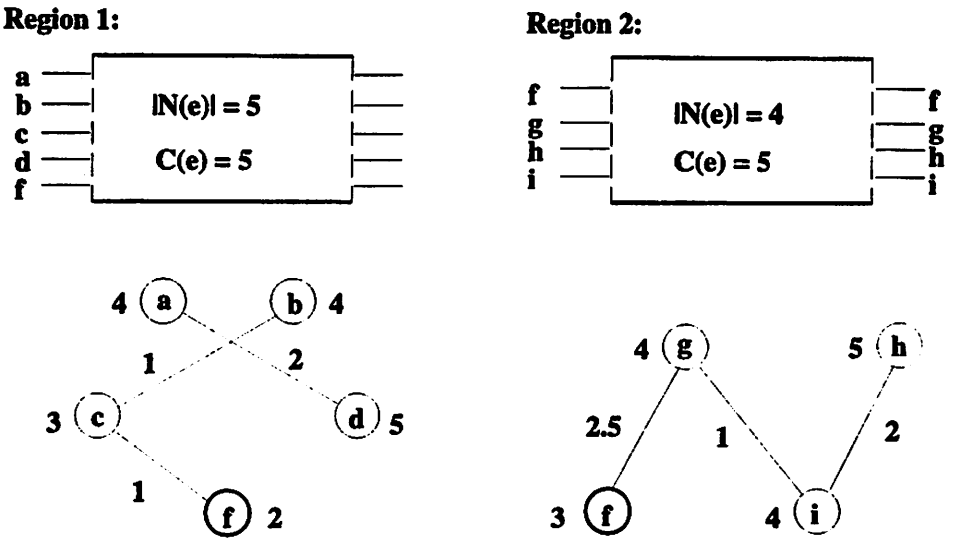

Figure 5.1: $CRG_{sp-max}(e)$ of Region 1 and 2 Under Partition One of $Bound(f)$

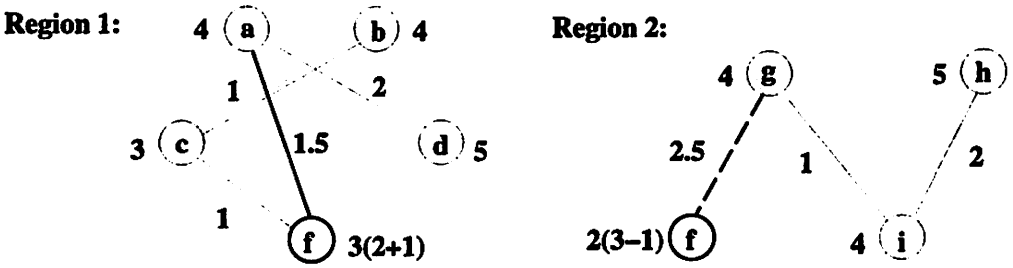

Figure 5.2: $CRG_{sp-max}(e)$ of Region 1 and 2 Under Partition Two of $Bound(f)$

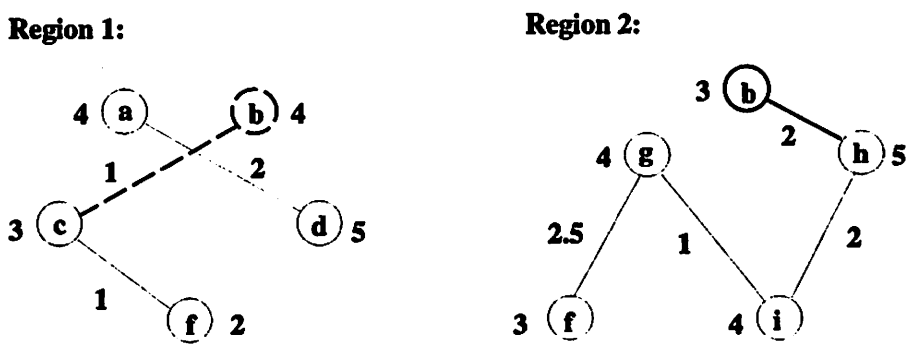

Figure 5.3: $CRG_{sp-max}(e)$ of Region 1 and 2 After Route Adjustment of net b

5.1.2.1 Risk Tolerance Bound Partitioning

To achieve more accurate crosstalk estimation considering the crosstalk situations of both regions, we generate Partition Two of $Bound(f)$ by increasing $Bound(f,1)$ from 2 to 3 and reducing $Bound(f,2)$ from 3 to 2, i.e., $Bound_2(f) = Bound_2(f,1) + Bound_2(f,2) = 3+2$. The partitioned bounds of other nets remain unchanged. Due to the increase in $Bound(f,1)$, edge $(a,f) \in CRG(1)$ with weight 1.5, which violates $Bound_1(f,1) = 2\,(noise(a,f)+noise(c,f) = 2.5$, causing violation Case 2) and is excluded from $CRG_{sp-max}(1)$ under Partition One, can now be included into it. As a result, $Risk_2(1) = 2*5-4-5-1 = 0$ is reduced from 1 to 0, indicating a risk-free estimation for Region 1 can be found. On the other hand, edge (f,g) with weight 2.5 is removed from $CRG_{sp-max}(2)$ since it violates $Bound_2(e,2) = 2$ (violation Case 1) under reduced $Bound(f,2)$. Nevertheless, $Risk_2(2) = 2*4-2-5-1 = 0$ is still non-positive, since there is one empty track in Region 2 that can be used as a shield to connect the two path segments (f) and (g,i,h) in $CRG_{sp-max}(2)$ together as one. Therefore, both regions are risk-free under Partition Two of $Bound(f)$. Notice that the increases in $Bound(i,e)$s in some regions must be compensated by the decreases in $Bound(i,e)$s in other regions on $route(i)$, since the risk tolerance bound of each net is a constant.

The example above indicates that a smaller and more accurate estimation of regions' risks can be obtained if the partitions of the risk tolerance bound of nets are adjusted appropriately among their routing regions on the chip. Our goal for risk tolerance bound partitioning can then be stated as:

Partition the risk tolerance bound of each net among its routing regions to reflect their crosstalk situations so that the total positive risk of regions is minimized and an accurate estimation of regions' risks can be obtained.

5.1.2.2 Global Routes Adjustment

Besides appropriate risk tolerance bound partitioning, another way to eliminate the positive risk of Region 1 in Fig. 5.1 is global routes adjustment. One possibly way of doing it is to rip-up net b from Region 1 and re-route it through Region 2. As a result, node b and its connecting edge (b,c) are removed from $CRG_{sp-max}(1)$, while node b and edge (b,h) are added into $CRG_{sp-max}(2)$, where $Bound(b,2) = 3$ and $noise(b,h,2) = 2$. Since ripping-up net b reduces $|N_s(e)|$ by one and frees one more track in Region 1, the updated $Risk(1)$ becomes: $Risk(1) = 2*4-2-5-1 = 0$, i.e., Region 1 becomes risk-free. On the other hand, although there is one more sensitive net routed

in Region 2 which consumes an empty tracks there, $Risk(2) = 2 * 5 - 4 - 5 - 1 = 0$ is still non-positive since net b can be adjacent to e without causing crosstalk violations. Therefore, both Region 1 and 2 are risk-free after globally adjusting the route of net b.

This example demonstrates that we can eliminate those positive risk regions by ripping-up certain nets from them and re-routing the nets appropriately through other low risk regions on the chip. Thus, the objective of global routes adjustment can be stated as:

Rip-up a set of nets from positive risk regions and re-route the nets with the best alternative routes such that every routing region on the chip becomes risk-free in global routing.

For the rest of this chapter, Sec. 5.2 analyzes the impact of bound adjustment on region's risk; Sec. 5.3 presents the risk tolerance bound partitioning algorithm; Sec. 5.4 discusses global routes adjustment via net ripping-up and re-routing; finally, Sec. 5.5 shows experimental results.

5.2 Adjustment in Risk Tolerance Bound

For risk tolerance bound partitioning, we first analyze the impact of risk bound changes on the configuration of crosstalk risk graphs and the risk estimations of routing regions.

5.2.1 Risk Tolerance Bound vs. Crosstalk Risk Graph

According to the definition of $CRG(e)$ and $CRG_{sp-max}(e)$, an edge (i, j) fails to appear in $CRG_{sp-max}(e)$ only in the following two cases:

1. It is excluded from $CRG(e)$ if the potential noise between net pair i, j causes crosstalk violation under Case 1 at either net i or j in region e, i.e.,
   $noise(i, j) \geq Bound(i, e)$ or $noise(i, j) \geq Bound(j, e)$.

2. It is excluded from $CRG_{sp-max}(e) \subseteq CRG(e)$ if one of the followings happens:

   2.1 It causes node degree violation at node i or j, i.e., $degree(i) > 2$ or $degree(j) > 2$.

   2.2 It causes crosstalk violation under Case 2 at either net i or j, i.e.,
   $\exists (i, k)$ or $(j, k) \in CRG_{sp-max}(e)$ s.t. $noise(i, j) + noise(i, k) \geq Bound(i, e)$ or
   $noise(i, j) + noise(j, k) \geq Bound(j, e)$.

Therefore, an edge (i, j) cannot be included in $CRG_{sp-max}(e)$ under noise constraints if $noise(i, j)$ causes crosstalk violation under Case 1 or 2 to happen:

1. At both nets i and j, and edge (i,j) is denoted as "locked".

2. At only one of nets i and j, and edge (i,j) is denoted as "half-locked".

Edge (i,j) is characterized as "free" if it causes crosstalk violations at neither net i nor j.

Given a global routing solution of region e and the sensitivities among net pairs, the configuration of $CRG(e)$ and $CRG_{sp-max}(e)$ are determined by the partitioned risk tolerance bounds of sensitive nets routed in e. By increasing $Bound(i,e)$ or $Bound(j,e)$ appropriately, we can eliminate the crosstalk violations at net i or j and switch the status of edge (i,j) from "locked" to "half-locked" or from "half-locked" to "free". If edge (i,j) is switched to "free", it may become a new edge in $CRG_{sp-max}(e)$ (when the degree constraints at nodes i and j are also satisfied), and $Risk(e)$ is reduced by 1 according to Eqn (4.10). On the other hand, reducing $Bound(i,e)$ or $Bound(j,e)$ may cause crosstalk violations to happen at net i or j and switch "free" edge (i,j) to "half-locked" or "locked" status. As a result, there may be one less edge in $CRG_{sp-max}(e)$ and an increase in $Risk(e)$. The change in $Risk(e)$ due to adjustment in $Bound(i,e)$ can be characterized by the following theorem:

Theorem 5.1 *The change in $Risk(e)$ of region e caused by adjusting the risk tolerance bound of net $i \in N_s(e)$, $Bound(i,e)$, equals one of $\{-2,-1,0,1,2\}$.*

Proof:

The increase (decrease) in $Bound(i,e)$ can lead to more (less) "half-locked" or "free" ("locked") edges connecting to node i. Since $CRG_{sp-max}(e)$ consists of simple path segments only, $degree(i) \leq 2$ holds for every node i in $CRG_{sp-max}(e)$. Therefore, the increase (decrease) in $Bound(i,e)$ can at most add (remove) the two edges connecting to node i into $CRG_{sp-max}(e)$, i.e., the change in edge number of $CRG_{sp-max}(e)$, $|E_{p-max}(e)|$, equals one of $\{-2,-1,0,1,2\}$. According to the definition of $Risk(e)$ in Eqn (4.10),

$$Risk(e) = 2|N_s(e)| - |E_{p-max}(e)| - C(e) - 1$$

the change in $Risk(e)$ due to adjustment in $Bound(i,e)$ also equals one of $\{-2,-1,0,1,2\}$.
□

5.2.2 Characterization of Adjustment in Risk Tolerance Bound

Theorem 5.1 implies that the change in $Risk(e)$ is not continuous with respect to the adjustment in $Bound(i,e)$, since $Bound(i,e)$ may affect the status of connecting edges at node i

in $CRG_{sp-max}(e)$ and thus $|E_{p-max}(e)|$ only when it satisfies or violates the noise constraints for net i. In addition, the impact of adjustment in $Bound(i,e)$ on $Risk(e)$ is bounded by 2. Therefore, it is possible to characterize the adjustment in $Bound(i,e)$ into discrete amounts according to the changes it may cause in $Risk(e)$.

For risk reduction, the amount of increase in $Bound(i,e)$, $Inc(e)$, can be characterized as:

- $Inc_0(i,e)$: the amount of increase in $Bound(i,e)$ that does not affect $Risk(e)$ but may switch some edges from "locked" to "half-locked".
- $Inc_{1,2}(i,e)$: the minimum amount of increase in $Bound(i,e)$ that creates new "free" edges and reduces $Risk(e)$ by 1 and 2, respectively.

For further characterization of $Inc_0(i,e)$, $Inc_{0k}(i,e) \leq Inc_0(i,e)$ is defined as the minimum amount of increase in $Bound(i,e)$ that can release k "locked" edges connecting to i.

Clearly, $Inc_0(i,e) < Inc_1(i,e) < Inc_2(i,e)$. Notice that it is not always possible to specify $Inc_1(i,e)$ and $Inc_2(i,e)$ since "free" edges connecting to i may not always become new edges in $CRG_{sp-max}(e)$ and contribute to the reduction in $Risk(e)$ due to the constraints on nodes' degree.

Similar to $Inc(e)$, the amount of decrease in $Bound(i,e)$, $Cut(e)$, can be characterized as:

- $Cut_0(i,e)$: the maximum amount of decrease in $Bound(i,e)$ that does not affect $Risk(e)$ but may switch some edges from 'half-locked" to 'locked".
- $Cut_{1,2}(i,e)$: the maximum amount of decrease in $Bound(i,e)$ that reduces "free" edges and increases $Risk(e)$ by 1 and 2, respectively.

$Cut_{0k}(i,e) \leq Cut_0(i,e)$ is further specified as the maximum amount of decrease in $Bound(i,e)$ that switches k edges from "half-locked" to "locked" status.

Again, $Cut_0(i,e) < Cut_1(i,e) < Cut_2(i,e)$ and $Cut_{1,2}(i,e)$ can not be specified if the removal of certain "free" edges connecting to i does not affect $|E_{p-max}(e)|$ and $Risk(e)$. Since the increase in $Risk(e)$ caused by reduction in $Bound(i,e)$ is bounded by 2, $Cut_k(i,e) = Bound(i,e)$, where $k \in \{0,1,2\}$ is the maximum impact on $Risk(e)$ that can be specified.

5.3 Risk Tolerance Bound Partitioning

5.3.1 Problem Statement

The adjustments in the risk tolerance bound of each net are characterized using the current crosstalk risk graph and risk estimation of each routing region on the chip, both are obtained based on the current global routing solution and the partitions of nets' risk tolerance bounds. Starting with an initial partitions of bounds, the risk tolerance bound partitioning process iteratively estimates the impact of bound changes of each net on its routing regions' risks, and adjusts the current partitions of bounds appropriately so that the total positive risk of the chip is minimized and the risk estimation becomes accurate.

The key step in the risk tolerance bound partitioning process is the adjustment of bound of each net among its routing regions for positive risk minimization. Since $Bound(i)$ is partitioned among all the routing regions of net i, the adjustment in its partitioned bound in one region will affect its bounds in other regions (and thus their risks) along $route(i)$. Therefore, the risk minimization of different routing regions are not independent of each other and must be considered simultaneously during the partitioning process. Due to the discrete nature of $Risk(e)$ as functions of adjustments in $Bound(i,e)$s, the risk tolerance bound partitioning process is formulated as an integer linear programming (ILP) problem.

Within each routing region e, the adjustments in risk tolerance bounds of nets routed in e are not independent of each other, i.e., although $Risk(e)$ can be reduced by 1 due to the increase in $Bound(i,e)$ by $Inc_1(i,e)$ and $Bound(j,e)$ by $Inc_1(j,e)$ separately, it may not be reduced by 2 if both $Bound(i,e)$ and $Bound(j,e)$ are increased at the same time. In our ILP formulation, the impact of the bounds' adjustments on $Risk(e)$ is linearized, i.e., we assume the changes in bounds of different nets in e are independent of each other. Despite its inaccuracy, this assumption significantly simplifies the optimization formulation and can still guide the bound adjustments toward the right direction for risk reduction. The accurate estimation on regions' risks can always be obtained for the next round of adjustments after the current partitioned bounds are updated according to the ILP solutions.

Since adjustments in $Bound(i,e)$s may not have immediate impact on $Risk(e)$ if $Inc_{1,2}(i,e)$s or $Cut_{1,2}(i,e)$s can not be specified, a two-phase ILP formulation is designed with the following objectives:

- Phase I: Switch maximum number of edges from "locked" to "half-locked" status so that they

may later become "free" edges for risk reduction in Phase II.

- Phase II: Minimize the total positive risk of the chip by switching maximum number of edges to "free" in $CRG_{sp-max}(e)$ for risk reduction.

These two phases are discussed separately in the following sections.

5.3.2 ILP Formulation for the Release of "Locked" Edges

The risk tolerance bound adjustment for switching edges from "locked" to "half-locked" status is served as the pre-processing step for risk reduction. Since the release of "locked" edges does not affect the current risks of routing regions, the adjustments in $Bound(i,e)$s should be bounded by $Cut_0(i,e)$s and $Inc_0(i,e)$s respectively, i.e., the risk of no region should increase for the release of "locked" edges in other regions.

In our ILP formulation, $u_k(i,e)$ and $v_k(i,e)$ are defined as binary variables indicating whether k edges will switch from "locked" to "half-locked" or from "half-locked" to "locked" status due to respective adjustment in $Bound(i,e)$. In other words, $Bound(i,e)$ will increase by $Inc_{0k}(i,e)$ if $u_k(i,e)=1$ and it will decrease by $Cut_{0k}(i,e)$ if $v_k(i,e)=1$. This ILP phase aims at maximizing the total number of "half-locked" edges in CRGs of those positive risk regions so that their risks can be reduced most during the Phase II of the optimization, and it can be formulated as follows:

Maximize $\sum_e E(e),\ \forall e, Risk(e)>0$

Subject to:

$$\sum_{i\in N_s(e)}\sum_k k u_k(i,e) - \sum_{j\in N_s(e)}\sum_k k v_k(j,e) = E(e) \qquad \forall e, Risk(e)>0$$

$$\sum_{e\in route(i)}\sum_k Inc_{0k}(i,e)u_k(i,e) \le \sum_{e\in route(i)}\sum_k Cut_{0k}(i,e)v_k(i,e) \qquad \forall i\in N_s$$

$$0 \le \sum_k u_k(i,e) + \sum_k v_k(i,e) \le 1,\ \ u_k(i,e), v_k(i,e)\in\{0,1\} \qquad \forall e\in route(i), \forall i\in N_s$$

The first constraint defines $E(e)$ as the change in the number of "half-locked" edges in $CRG(e)$ of region e after bound adjustment. Notice that $E(e)$ is the linearized approximation of the actual changes, since the bound adjustments of different nets in $N_s(e)$ are not independent of each other. The second constraint specifies the "supply" and "demand" relation for the partitioning of $Bound(i)$ of each net $i\in N_s$, i.e., the increases in $Bound(i,e)$s in some regions must be balanced by the decreases in $Bound(i,e)$s in other regions on $route(i)$. Both types of adjustments are bounded

by $Inc_{0k}(i,e)$ and $Cut_{0k}(i,e)$, respectively. The third constraint indicates that $Bound(i,e)$ can only be increased or decreased by a certain amount once at a time for every net i in a region e.

Although only portions of Inc_0s and Cut_0s are used for bound adjustment in Phase I, the new bound partitions after ILP may result in changes in regions' risks. This is due to the fact that Incs and Cuts are estimated for each net separately, while bounds at different nets in a region may be adjusted at the same time. As discussed later, Phase I can be integrated with Phase II of ILP for risk minimization in actual implementations.

5.3.3 ILP Formulation for Positive Risk Minimization

As previously analyzed, $Risk(e)$ can possibly be reduced when edges are switched from "half-locked" to "free" and become eligible in $CRG_{csp-max}(e)$. In Phase II of ILP formulation for positive risk minimization, $x_{1,2}(i,e)$ and $y_{0-2}(e)$ are defined as binary variables indicating whether $Risk(e)$ is reduced by 1 and 2 or increased by 0, 1 and 2 due to the respective adjustments in $Bound(i,e)$. In other words, $Bound(i,e)$ is increased by $Inc_{1,2}(i,e)$ if $x_{1,2}(i,e) = 1$ and it is decreased by $Cut_{0-2}(i,e)$ if $y_{0-2}(e) = 1$. The objective of this ILP phase is to minimize the total positive risk of all regions on the chip and it can be formulated as:

Minimize $\sum_e R(e) \quad \forall e, Risk(e) > 0$

Subject to:

$$Risk(e) + \sum_{i \in N_s(e)} (y_1(i,e) + 2y_2(i,e) - x_1(i,e) - 2x_2(i,e)) = R(e) \qquad \forall e, Risk(e) > 0$$

$$Risk(e) + \sum_{i \in N_s(e)} (y_1(i,e) + 2y_2(i,e) - x_1(i,e) - 2x_2(i,e)) \leq 0 \qquad \forall e, Risk(e) \leq 0$$

$$\sum_{e \in route(i)} (Inc_1(i,e)x_1(i,e) + Inc_2(i,e)x_2(i,e))$$

$$\leq \sum_{e \in route(i)} (Cut_0(i,e)y_0(i,e) + Cut_1(i,e)y_1(i,e) + Cut_2(i,e)y_2(i,e)) \qquad \forall i \in N_s$$

$$0 \leq x_1(i,e) + x_2(i,e) + y_0(i,e) + y_1(i,e) + y_2(i,e) \leq 1 \qquad \forall e \in route(i),$$

$$x_1(i,e), x_2(i,e), y_0(i,e), y_1(i,e), y_2(i,e) \in \{0,1\} \qquad \forall i \in N_s$$

Similar to the ILP formulation in Phase I, the first constraint defines $R(e)$ as the updated risk of positive risk region e after bound adjustment. The second constraint indicates that the adjustments of bounds of sensitive nets among their routing regions for positive risk reduction should not negatively affect the risks of those non-positive risk regions on the chip. The third constraint enforces that the "demands" for bound increases in some regions can be no more than the

"supplies" from bound decreases in other regions along the route of each sensitive net. The fourth constraint specifies that $Bound(i, e)$ can be updated only once at a time for each net i and region e. Like $E(e)$, $R(e)$ is a linearized approximation of the actual risk of region e under the updated bounds partitions due to the dependencies among the nets' bounds adjustments. Nevertheless, minimizing $R(e)$ points to the right direction of bound adjustment for positive risk minimization.

5.3.4 ILP Implementation Techniques

Due to their discrete nature, ILP problems are usually difficult and time-consuming to solve. To speed up the risk tolerance bound partitioning process, we adopt several techniques in its actual implementation.

5.3.4.1 Integration of the Two Phase ILP

It can be observed that the two phase ILP optimization for release of "locked" edges and risk reduction discussed separately in Sec. 5.3.2 and 5.3.3 are formulated in similar fashion. Therefore, it is possible to integrate them into one ILP in actual implementation in order to simplify the bound partitioning process. One way to achieve this is to treat all "locked" edges as "half-locked" during optimization and thus skip ILP Phase I of bound partitioning. In other words, it is assumed that the adjustment in $Bound(i, e)$ can always change the status of edge (i, j) to "free" and affect the risk of region e regardless of the noise constraints at net j. Although this may over estimate the improvement in $Risk(e)$ if $Bound(i, e)$ and $Bound(j, e)$ are not adjusted during the same round of ILP, it nonetheless points to the right direction for adjusting $Bound(i, e)$ and edge (i, j) can actually become "free" after no more than two consecutive rounds of adjustments in both $Bound(i, e)$ and $Bound(j, e)$ (it is regarded as "free" but may actually be "half-locked" after one round of adjustment). Since the accurate risk of each routing region is estimated based on updated risk tolerance bounds after each round of ILP adjustment, the inaccuracy during the ILP formulation does not affect the quality of the final risk tolerance bound partitions.

5.3.4.2 Control of Problem Size

The efficiency of ILP solving process is determined to a large extent by the size of the ILP problem in terms of the number of variables and constraints in it. In our ILP formulations, binary variables are adopted and they are further constrained under the restriction that the bound of each net can only be adjusted once at a time in each region. Still, effective implementation techniques

need to be adopted in order to solve ILP problems efficiently for testing circuits having large number of positive risk regions and sensitive nets.

One way to reduce the size of the ILP formulation for risk reduction is to decompose it into a series of ILP problems of smaller sizes. More specifically, we can substitute the optimization objective of:

$$\sum_{e} R(e) \;\; \forall e, \;\; Risk(e) > 0 \tag{5.1}$$

by a series of objectives in the form of:

$$\sum_{e} R(e) \;\; \forall e, \;\; Risk(e) = k, \;\; k = 1, \ldots, max\_risk \tag{5.2}$$

where k is a positive number ranging from 1 to the maximum positive risk of all regions. Thus, instead of minimizing the total positive risk on the chip, we iteratively minimize the risks of those regions having a specific positive risk. Since the complexity of solving an ILP problem grows non-linearly with respect to its size, solving a series of ILP problems small in size with Eqn (5.2) as its objective is much more efficient than solving one large ILP problem optimizing Eqn (5.1). In addition, the quality of the solutions by the two approaches are comparable since both Eqn (5.2) and (5.2) minimizes the positive risks on the chip.

For further reductions in the size of ILP problems, extra constraints can be imposed on the number of nets whose bounds are allowed to adjusted at the same time within each routing region, or on the number of regions to be considered in the constraints during the bound partitioning process. This also allows the exploration of the possible trade-offs between the quality of the bound partitions and the running time of the ILPs determined by their problem sizes. In other words, when only rough partitions of nets' bounds are required, fast partitioning solutions can be obtained by setting strict limits on the number of regions and nets to be considered at every step during the optimization.

5.3.5 Risk Tolerance Bound Partitioning Algorithm

Based on the ILP formulation for bound adjustments, the risk tolerance bound partitioning algorithm is designed as an iterative optimization process. Initially, the risk tolerance bound of each sensitive net is partitioned uniformly among its routing regions on the chip. Then the crosstalk risk graph and risk estimation of each routing region can be computed. At each iteration during the bound partitioning process, the possible changes in bounds are first characterized for each sensitive

net among its routing regions, then the current bound partitions are adjusted via ILP formulation to minimize the total positive risks on the chip. After each round of bounds adjustment, the crosstalk risk graphs and regions' risks are updated. This process continues until the positive risks of regions can not be reduced further and risk estimation of the chip becomes accurate.

Risk tolerance bound partitioning algorithm {

1. Input a feasible global routing solution of the chip, plus sensitivities and risk tolerance bounds of nets.
2. Initial bound partitioning:
   Partition the risk tolerance bound of each net uniformly among its routing regions.
3. Construct the crosstalk risk graph and estimate the crosstalk risk of each region.
4. While reduction in positive risk is possible:
   - 4.1 Calculate $Incs$ and $Cuts$ of bound of each net among its routing regions based on their current CRGs and $Risk$s.
   - 4.2 Solve ILP problem for positive risk minimization, adjust partitions of bounds.
   - 4.3 Update crosstalk risk graphs and risks of routing regions.

}

The regions' positive crosstalk risks may be over-estimated initially, since the initial uniform bound partitions may not reflect the actual crosstalk situation on the chip. After risk tolerance bound partitioning, the total positive risk of the chip is minimized, indicating fewer regions and nets are subject to global routes adjustment for crosstalk risk reduction under accurate risk estimations. This speeds up the generation of a risk-free global routing solution of the chip as shown by the experimental results in Sect. 5.5.

5.4 Global Routes Adjustment

If positive risk regions still exist under accurate risk estimation after risk tolerance bound partitioning, global routes adjustment is applied. According to the analysis in Sec. 5.1.2.2, it reduces the positive risks of regions via ripping-up and re-routing certain set of nets so that a risk-free global routing solution of the chip can be obtained. These two phases in global routes adjustment are discussed separately in the following sections.

5.4.1 Net Ripping-up

Since adjusting the routes of nets globally may affect the quality of the current global routing solution of the chip in terms of the routing densities, total wire length, number of vias and interconnect delays, the number of nets whose routes have to be adjusted for risk reduction should be minimized. Thus, the objective of net ripping-up can be stated as:

Identify a minimum set of sensitive nets need to be ripped-up from those positive risk regions so that they can become risk-free.

For each positive risk region e, we define $N_r(e) \subseteq N_s(e)$ as the minimum set of sensitive nets need to be ripped-up from it in order to generate a risk-free routing solution of e, i.e., the removal of nets in $N_r(e)$ from e reduces $Risk(e)$ to non-positive. Intuitively, ripping-up a net from region e frees one extra track in it which can then be used to separate two sensitive nets subject to crosstalk considerations in the region, so it can never result in increase in $Risk(e)$. However, ripping-up different nets in e may have different impact on $Risk(e)$, in particular, the removal of a nets which has to be shielded from others in the region can reduce the number of shield needed in e, i.e., the risk of the region. The relation between net ripping-up and risk reduction can be stated by the following theorem:

Theorem 5.2 *The reduction in $Risk(e)$ of region e caused by ripping-up net i from e, $Risk_{dec}(i, e)$, equals one of $\{0, 1, 2\}$, more precisely,*

$$Risk_{dec}(i, e) = 2 - degree(i) \tag{5.3}$$

where $degree(i)$ is the degree of node i in $CRG_{sp-max}(e)$.

Proof:

Ripping-up net i from region e reduces the number of sensitive nets in region e $|N_s(e)|$ by 1. From graph point of view, it deletes node i and its connecting edges from $CRG_{sp-max}(e)$. As a result, the number of edges in $CRG_{sp-max}(e)$, $|E_{p-max}(e)|$, is reduced by $degree(i)$. According to the definition of $Risk(e)$ in Eqn (4.10), the reduction in $Risk(e)$ can be expressed as:

$$Risk_{dec}(i, e) = 2 * \Delta|N_s(e)| - \Delta|E_{p-max}(e)| = 2 - degree(i)$$

Since $CRG_{sp-max}(e)$ is a simple path sub-graph, $degree(i) \in \{0, 1, 2\}$. Therefore, $Risk_{dec}(i, e)$ equals one of $\{0, 1, 2\}$.

□

According to Theorem 5.2, ripping-up a net whose corresponding node has degree 0 or 1 in $CRG_{sp-max}(e)$ can reduce $Risk(e)$ by 2 or 1 respectively, since 2 or 1 shields needed to separate net i from the sensitive nets in its above and/or below tracks are no longer needed. On the other hand, ripping-up a net having node degree 2 in $CRG_{sp-max}(e)$ does not affect $Risk(e)$, since no shield is needed for net i in a risk-free routing solution of e. Thus, the minimum set of sensitive nets to be ripped-up from region e for risk reduction can be constructed as follows:
$N_r(e)$ *Construction Algorithm*{

1. Set $G(e) = CRG_{sp-max}(e), N_r(e) = \emptyset, R(e) = Risk(e)$.

2. While $R(e) > 0$:

   2.1 While $R(e) > 0$ and $\exists$ node $i \in G(e)$ having degree 0:
   Remove node i from $G(e)$, $N_r(e) = N_r(e) \cup \{i\}$, $R(e) = R(e) - 2$.

   2.2 If $R(e) > 0$:

      2.2.1 Choose a node $i \in G(e)$ having degree 1, break ties by selecting the one which connects to another node $j \in G(e)$ also having degree 1.

      2.2.2 Remove node i and its connecting edge from $G(e)$, $N_r(e) = N_r(e) \cup \{i\}$, $R(e) = R(e) - 1$.

}

Nodes having degree 0 in $CRG_{sp-max}(e)$ are chosen first at Step 2.1 since their removal can reduce $Risk(e)$ most. Each node i having degree 1 connects to another node j having degree 1 or 2 in $CRG_{sp-max}(e)$. At Step 2.2.2, priority is given to node i connecting to node j with degree 1, since j can become a new 0 degree node after node i and edge (i, j) are removed. This iterative net selecting process continues until $\sum_{i \in N_r(e)} Risk_{dec}(i, e) \geq Risk(e)$, i.e., ripping-up sensitive nets in $N_r(e)$ from e can lead to a risk-free routing solution of e.

5.4.2 Net Re-routing

Once nets to be ripped-up from a positive risk region are identified, they are re-routed through other regions on the chip. Analogous to net ripping-up, the re-routing of net i in a region e on its new route may result in increase in $Risk(e)$, i.e.,

$$Risk_{inc}(i, e) = 2 - degree(i) \tag{5.4}$$

where $degree(i)$ is the degree of node i in $CRG_{sp-max}(e)$. Therefore, the new alternative routes of those ripped-up nets should consider the crosstalk risks of regions on the chip, in addition to other concerns in global routing such as regions' densities, wire lengths, number of vias, timing constraints, etc. To this end, we adopt a modified version of the global router developed in [Wang 96], which is extended from the original one to take into account the regions' crosstalk risks. To minimize the increase in positive risk on the chip, it determines the new routes of those ripped-up nets as follows:

1. Choose those regions having the lowest risks when other routing constraints are satisfied.
2. Among regions having the same risks, choose those in which they may cause the least increases in risks.

The motivation behind re-routing those ripped-up nets according to these two rules is to minimize the number of new positive risk regions that may be created in the process so that few iterations in global routes adjustment are required to generate a risk-free global routing solution of the chip.

5.4.3 Global Routes Adjustment

The global routes adjustment is the final stage in the crosstalk synthesis process after accurate estimation of regions' risks have been obtained via risk tolerance bound partitioning. It is formulated as an iterative optimization process, which updates the regions' risks and partitions of risk tolerance bounds after each round of net ripping-up and re-routing. Its objective is to eliminate the positive risk regions on the chip so that a risk-free global routing solution of the chip can be obtained:

Global Routes Adjustment Algorithm {

1. Input a feasible global routing solution, sensitivities and risk tolerance bounds of nets.
2. Estimate the crosstalk risks of regions accurately via graph-based estimation and risk tolerance bound partitioning.
3. While there exists region e on the chip with $Risk(e) > 0$:
   - 3.1 Identify the minimum set of nets $N_r(e)$ need to be ripped up from region e to reduce its risk to non-positive.
   - 3.2 Reroute those ripped-up nets in $N_r(e)$ with minimum cost alternative routes considering regions' crosstalk risks.

Table 5.1: Benchmark specifications

Circuit	# macro cells	# nets	# pins	G_{size} (row x col)
ami33	33	123	442	28 x 23
hp	11	83	309	289 x 228
xerox	10	203	696	24 x 24
ami49	49	408	953	184 x 139

3.3 Update risk estimation and bounds' partitions after global routes adjustment.

}

5.5 Experimental Results

The risk tolerance bound partitioning and global routes adjustment algorithms have been implemented and tested on a DEC 5000/125 workstation. Four circuits constructed from the CBL/NCSU building-block benchmarks, ami33, hp, xerox and ami49 are used in our experiments. The specifications of these circuits are listed in Table 5.1, where G_{size} refers to the size of the global routing graph of the chip.

The feasible global routing solution of these chips are generated by a performance-driven placement [Esbensen 96] and a global router [Wang 96], respectively. In our experiments, circuit $ami33$ and $xerox$ are each tested under two different placement/global routing solutions, denoted as $*.1$ and $*.2$ respectively. The ILPs for bound partitioning are solved by *lp-solve* optimization tool. As the testing on risk estimation method in Sec. 4.5.2, the crosstalk information of each circuit is specified by net sensitivity ratio, which is the percentage of net pairs in the circuit that are subject to crosstalk risk concern, and the risk tolerance bound of each net, which is the percentage of the total net length allowed for coupling with other sensitive nets. As an integral part of the the bound partitioning and global routes adjustment algorithms, the risk estimation methods discussed in Chapter 4 are further tested and verified in our experiments for crosstalk risk reduction.

Fig. 5.4 compares the total number of extra shields needed for a risk-free global routing solution, i.e., total positive risk on the chip, for circuit $ami33.1$ under two different partitions of risk tolerance bounds: uniform and adjusted by our bound partitioning algorithm. Here, the experimental results are obtained under 100% sensitivity ratio, which is the worst situation possible assuming every pair of nets is subject to crosstalk noise concern during optimization. It can be seen

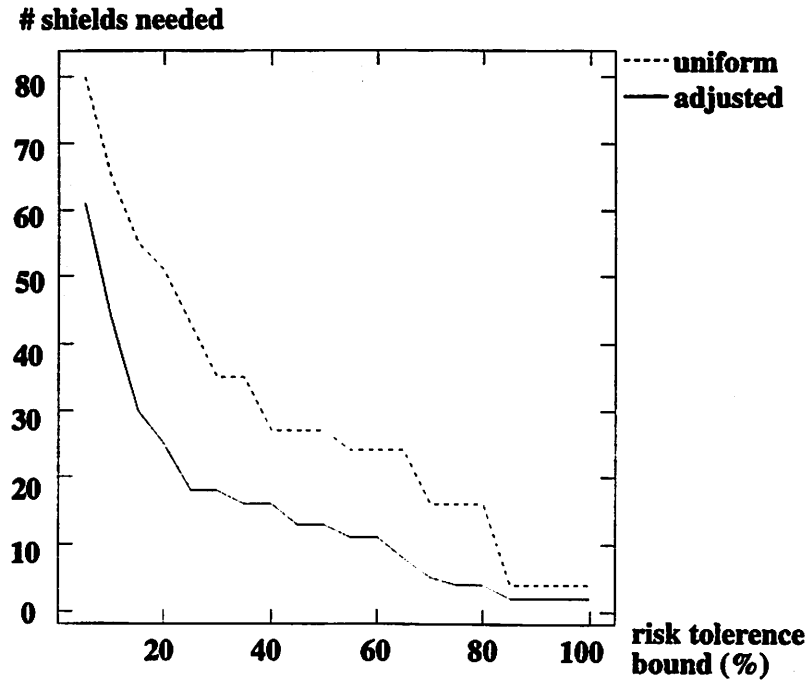

Figure 5.4: Uniform vs. Adjusted Risk Tolerance Bound Partitioning

that the risk estimation becomes more accurate under adjusted bounds partitions, and the number of extra shields needed on the chip is reduced drastically by over 50% for the entire range of bound specifications from 5% up to 100% of the total net length.

For crosstalk risk reduction, our main focus is on those regions having positive risks on the chip. Table 5.2 and 5.3 show estimations of positive risk regions under uniform and adjusted partitions of risk tolerance bounds before and after global routes adjustment, respectively. Here, results are measured under the most conservative net sensitivity ratio at 100% and the risk tolerance bound at 50% of net wire length.

When applied before global routes adjustment (Table 5.2), adjusted bound partitions reduce the numbers of positive risk regions, extra shields needed on the chip and nets need to be ripped up for risk-free routing solution of the chip by average of 40%, 59% and 55% respectively. This implies that much fewer nets need to be ripped-up and re-routed during global routes adjustment according to the accurate risk estimation of the chip. In case of circuit *ami*33.2, global routes adjustment is avoided since adjusted partitions of nets' bounds eliminate all positive risk regions on the chip.

After only one round of ripping-up and re-routing of those identified nets for positive risk

Table 5.2: Estimation of Positive Risk Regions Before Global Routes Adjustment

Testing Circuit	# positive risk regions			total # shields needed			# nets to be ripped-up		
	uniform	adjusted	-%	uniform	adjusted	-%	uniform	adjusted	-%
ami33.1	7	4	43	27	13	52	15	8	47
ami33.2	13	0	100	17	0	100	13	0	100
hp	39	39	0	105	59	44	72	48	33
xerox.1	12	5	58	44	10	77	24	5	79
xerox.2	53	43	19	175	88	50	103	60	42
ami49	214	166	22	375	270	28	232	166	28

Table 5.3: Estimation of Positive Risk Regions After Global Routes Adjustment

Testing Circuit	# positive risk regions		total # shields needed		# nets to be ripped-up	
	uniform	adjusted	uniform	adjusted	uniform	adjusted
ami33.1	0	-	0	-	0	-
hp	0	-	0	-	0	-
xerox.1	11	0	25	0	14	0
xerox.2	15	0	38	0	23	0
ami49	24	0	48	0	24	0

reduction of the circuits, all three measures on positive risk regions are reduced to 0 under adjusted bound partitions as shown in Table 5.3 (for circuit $ami33.1$ and hp, partitions of nets' bounds do not need to be adjusted). This indicates that, due to the significantly reduced number of nets need to be adjusted for risk reduction, one round of global routes adjustment is sufficient to generate a risk-free global routing solution for each circuit tested. Plus, it demonstrates that our global routes adjustment method is very efficient for risk reduction since it takes into account the risks of routing regions on the chip and creates no new positive risk regions during the net ripping-up and re-routing process. Our experiments also show that there are little changes in routing densities and wire lengths of nets in the global routing solutions since only the routes of a small percentage of nets on the chip are adjusted.

5.6 Conclusions

Previous approaches to crosstalk synthesis are mainly localized optimization methods at the detailed routing level. Due to the limited routing flexibilities, they alone often fail to achieve satisfactory results for risk minimization. Furthermore, the problem of partitioning the risk tolerance

bounds of nets among their routing regions, which is critical for constrained crosstalk optimization, has not been adequately addressed.

In order to achieve risk-free final solutions of chips, we must address the crosstalk synthesis at the global routing as well as the detailed routing level. This chapter and Chapter 4 proposes a post global routing crosstalk optimization approach, which to our knowledge, is the first to estimate and reduce crosstalk risk in global routing. Unlike previous net-based approaches, our method is region-based, which quantitatively defines and estimates the risk of each routing region on the chip as a whole using a graph-based optimization approach. For accurate risk estimation and constrained optimization of each region, the risk tolerance bound of each sensitive net is partitioned appropriately among its routing regions via integer linear programming. Finally, the routes of certain set of sensitive nets are adjusted globally among all regions on the chip via net ripping-up and re-routing in order to eliminate those regions having positive risks. At the end of the optimization process, a risk-free global routing solution is obtained together with partitions of nets' risk tolerance bounds which reflect the crosstalk situation of the chip. These can greatly facilitate the generation of a risk-free final solution of the chip by a crosstalk-driven detailed router at later stages in the layout process. The experimental results on CBL/NCSU benchmarks are very promising, which indicate that our methods are very efficient in generating risk-free global routing solutions of chips.

Bibliography

[Chaudhary 93] K. Chaudhary, A. Onozawa and E. Kuh, "A Spacing Algorithm for Performance Enhancement and Cross-Talk Reduction", *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design*, pp. 697-702, 1993.

[Chen 92] H. Chen and C. Wong, "Wiring and Crosstalk Avoidance in Multi-Chip Module Design", *Proc. Custom Integrated Circuits Conf.*, pp. 28.6.1-28.6.4, 1992.

[Esbensen 96] H. Esbensen, E. Kuh, "An MCM/IC Timing-Driven Placement Algorithm Featuring Explicit Design Space Exploration", *Proc. IEEE Multi-Chip Module Conf.*, pp. 170-175, 1996.

[Gao 93] T. Gao, C. Liu, "Minimum Crosstalk Channel Routing", *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design*, pp. 692-696, 1993.

[Gao 94] T. Gao, C. Liu, "Minimum Crosstalk Switchbox Routing", *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design*, pp. 610-615, 1994.

[Johnson 90] D. Johnson, "Local Optimization and the Traveling Salesman Problem", *Proc. 17th Int'l Colloquium on Automata, Languages and Programming*, pp. 446-460, 1990.

[Kirkpatrick 94] D. Kirkpatrick, A. Sangiovanni-Vincentelli, "Techniques for Crosstalk Avoidance in the Physical Design of High-Performance Digital Systems", *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design*, pp. 616-619, 1994.

[Wang 96] D. Wang, E. Kuh, "Performance-Driven Interconnect Global Routing", *Proc. 6th Great Lakes Symp. on VLSI*, pp. 132-136, 1996.

[Xue 96b] T. Xue, E. Kuh and D. Wang, "Post Global Routing Crosstalk Risk Estimation and Reduction", *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design*, pp. 302-309, 1996.

[Xue 96c] T. Xue, E. Kuh and D. Wang, "A Post Global Routing Optimization Approach to Crosstalk Estimation and Reduction", *Proc. TECHCON 96*, Sept. 1996.

Chapter 6

Conclusions

6.1 Summary of Thesis

The interconnect performance issues have become increasingly important and challenging as they dominate high performance circuit design under deep sub-micron technologies at $0.25\mu m$ and beyond. Different from previous approaches, this thesis proposed a novel direction for interconnect performance optimization at the post routing level, which reduces the interconnect delay, skew and crosstalk under constrained optimization after a feasible routing solution of the chip has been obtained. As complements to optimization methods at other stages in the layout process, our methods aim at achieving satisfactory performance of the chip while maintaining the wirability of the layout solution. If successful, they can significantly speed up the circuit design process by avoiding the time-consuming iterations in layout which are not guaranteed to converge.

The post routing interconnect delay optimization, which improves the performance of an existing critical net topology under routing resource constraints, is discussed in Chapter 2 and 3 for distributed RC line and lossy transmission line topologies, respectively. These two models cover all possible interconnect types under deep sub-micron IC, MCM and PCB technologies. For distributed RC line topologies, a link insertion and wiresizing approach is designed, which improves the interconnect delay and skew of a critical net by inserting and wiresizing new interconnect wires into its topology appropriately. Since the inserted wires introduce extra admittance into the topology and cause un-balanced decreases in the mutual resistances between nodes, both the maximum delay and delay skew of the net can possibly be reduced. In addition, the proposed method can be applied to any arbitrary routing topologies either un-optimized or which have been improved by other means, and it no longer restricts topologies to tree structures. For lossy transmission line topologies,

a sensitivity-based wiresizing method is proposed, which minimizes the maximum delay of the existing topology by adjusting its wire widths under routing resource constraints. The delay and its sensitivities with respect to the widths of wires in the topology are computed using high order moments based on an exact moment matching model for each lossy transmission line. Compared to other approaches, our method achieves analytical sensitivity computation and calculates higher order moments (sensitivities) recursively from lower order moments for tree networks. Experiments shows significant improvement in interconnect performance for both types of interconnect modelings using the proposed algorithms, which demonstrate the promising potentials of our approaches in performance-driven physical design.

Chapter 4 and 5 discuses the post global routing crosstalk risk estimation and reduction method, which to our knowledge, is the first to address the crosstalk synthesis at the global instead of detailed routing level. In contrast to net-based approaches previously reported, our region-based approach quantitatively defines and estimates the crosstalk risk for each routing region on the chip and adjusts routes of nets globally for risk reduction. By estimating the crosstalk risks at the global routing level, it can identify and eliminate crosstalk violations at an early stage before moving further into the detailed routing and thus avoid many iterations in the layout process. In addition, it partitions the risk tolerance bound of each sensitive net appropriately among its routing regions for accurate risk estimation and crosstalk constrained optimization of each region on the chip. The objective of our approach is to generate a risk-free global routing solution of the chip in which every routing region is crosstalk risk-free. Compared to the un-optimized global routing solution, it is a much better starting point for the crosstalk driven detailed router to produce a final crosstalk risk-free routing solution of the chip. Experiments shows that our methods are very efficient at eliminating positive risk regions and generating a risk-free global routing solution of the chip due to accurate risk estimations and global routes adjustment with crosstalk considerations.

6.2 Future Directions

Due to its increasing dominance in circuit performance, interconnect performance issues should be considered not only in routing, but also at other stages in the layout and VLSI design process. In particular, it has to be addressed in performance-driven floorplanning/placement and logic synthesis in order to achieve satisfactory chip performance under deep sub-micron technologies. To take the parasitic effect of interconnects into consideration, high level synthesis must possess the following features:

- It should perform efficient routability analysis of the chip by generating rough routing topologies of interconnects to assess the quality of the current design solution obtained.
- It should conduct fast performance analysis on interconnect delay, skew and crosstalk and adjust the current solution appropriately for performance improvement.

The evaluation of the quality of the current solution in high level synthesis is a very difficult task not only because of the inaccuracy in routability and performance analysis, but also due to the inherent conflicts among multiple design objectives. For example, it is difficult to choose between two placement solutions of a chip, one with minimum chip area but 20% larger critical path delay than specified, while the other has satisfactory chip performance but 20% bigger area than the first one. To solve this dilemma, an estimation on the potential performance improvement that could possibly be achieved at later stages in the design process is absolutely necessary.

The post routing performance optimization methods presented in this thesis is well suited to meet that need. Besides the ability of generating interconnect topologies having optimal performance at the routing stage of the layout process, they also provide new ways for fast estimation of the potential improvement in current chip performance. For instance, the proposed approach in Chapter 2 can estimate the lower and upper bounds (i.e., the performance interval) on interconnect delay and skew reduction very fast according to the current critical net topology without pursuing the actual link insertion and wiresizing. If it predicts that satisfactory performance of the first placement solution given above can be achieved since the maximum delay of its critical nets can be reduced by at least 20% via link insertion, then that placement may be preferred. On the other hand, if the performance of the first one can be improved by no more than 20% according to prediction, the second placement should be considered. Similarly, fast estimation on the crosstalk situation of the chip can be obtained using our region-based crosstalk synthesis methods once the routing regions are defined and the rough nets' topologies are available. It can make the early identification and elimination of crosstalk violations possible in high level synthesis.

In summary, interconnect performance optimization under deep sub-micron technologies is by no means a solved problem. It affects routing as well as high level synthesis and must be addressed comprehensively at various stages in VLSI and physical design process for satisfactory design solutions. It is hoped that the work presented in this thesis may provide new ways for effective interconnect performance optimization and its potential can be further explored in a wide range of applications so that it may contribute successfully to the task of high performance circuit design.

Appendix A

Proofs in Chapter 2

A.1 Proof of Lemma 2.1

Lemma 2.1 *(Mutual Resistances after Link Insertion) When link e_n is inserted between the reference node n_{ref} and node $n \in N$, the mutual resistance between node $i, j \in N$, $(R_{ij})_n$, can be expressed as:*

$$(R_{ij})_n = R_{ij} - \frac{R_{ni}}{R_{nn} + R_{e_n}} R_{nj}$$

Proof:

Without any loss of generality, each matrix of N orders nodes in N from 1 to n (i.e., n is ordered last) in the following analysis.

Denote $\mathbf{G} = [G_{ij}]$ as the admittance matrix of N, which represents the admittances between nodes in N and can be computed according to the resistances of wire segments in N. $\mathbf{G}$ is symmetric and strongly dominant diagonally, satisfying the following properties:
$G_{ii} > 0$, $G_{ij} \leq 0, j \neq i$, and $G_{ii} \geq -\sum_{j \neq i} G_{ij}$, $\forall i \in N$.

By its definition, the resistance matrix $\mathbf{R}$ is the inverse of $\mathbf{G}$, i.e., $\mathbf{R} = \mathbf{G}^{-1}$. For resistance analysis, $\mathbf{R}$ is further decomposed as $\mathbf{R} = \mathbf{AB}$, where both $\mathbf{A}, \mathbf{B}$ are nxn matrices with:

$$\mathbf{BG} = \begin{bmatrix} 1 & 0 & \dots & 0 & G_{1n}^{(n-1)} \\ 0 & 1 & \dots & 0 & G_{2n}^{(n-1)} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & G_{nn}^{(n-1)} \end{bmatrix}$$

and

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \dots & 0 & -G_{1n}^{(n-1)}/G_{nn}^{(n-1)} \\ 0 & 1 & \dots & 0 & -G_{2n}^{(n-1)}/G_{nn}^{(n-1)} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 1/G_{nn}^{(n-1)} \end{bmatrix}$$

Here, $G_{in}^{(n-1)}$ is the value of G_{in} after $n-1$ rounds of pivoting on nodes from 1 to $n-1$.

According to this decomposition, matrix **B** and **G** can be expressed in the following form respectively:

$$\mathbf{B} = \begin{bmatrix} \mathbf{X}^{-1} & 0 \\ \mathbf{z} & 1 \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \mathbf{X} & \mathbf{y}^T \\ \mathbf{y} & 1 \end{bmatrix}$$

where **X** is a $(n-1)$x$(n-1)$ symmetric matrix and **y**, **z** are 1x$(n-1)$ and $(n-1)$x1 vectors, respectively. Based on this formulation, it can be easily shown via matrix manipulation that:

$$B_{nn} = G_{nn}^{(n-1)} = 1, \quad B_{nj} = -G_{jn}^{(n-1)}, j \neq n \tag{A.1}$$

Since $\mathbf{R} = \mathbf{AB}$, R_{ij}s can be expressed as:

$$R_{ij} = B_{ij} - \frac{G_{in}^{(n-1)}}{G_{nn}^{(n-1)}} B_{nj}, \quad i \neq n \tag{A.2}$$

$$R_{nj} = -\frac{G_{jn}^{(n-1)}}{G_{nn}^{(n-1)}}, \quad i = n, j \neq n \tag{A.3}$$

$$R_{nn} = \frac{1}{G_{nn}^{(n-1)}}, \quad i = n, j = n \tag{A.4}$$

When a distributed RC link e_n is inserted between n_{ref} and n, the only element that will change in the new admittance matrix $\mathbf{G}_n$ is: $(G_{nn})_n = G_{nn} + 1/R_{e_n}$, $\mathbf{B}_n$ remains the same as **B**.

Since $(G_{nn})_n > G_{nn} > 0$, we have:

$(G_{nn}^{(n-1)})_n > G_{nn}^{(n-1)} > 0$ (**R** is non-singular), and

$(G_{in}^{(n-1)})_n = G_{in}^{(n-1)} \leq 0, i \neq n.$

The mutual resistance $(R_{ij})_n$ after link insertion can then be studied in following two cases:

Case 1. If $i \neq n$, according to Eqn (A.1) and (A.2),

$$\begin{aligned} (R_{ij})_n &= (B_{ij})_n - \frac{(G_{in}^{(n-1)})_n}{(G_{nn}^{(n-1)})_n} (B_{nj})_n \\ &= B_{ij} - \frac{G_{in}^{(n-1)}}{G_{nn}^{(n-1)} + 1/R_{e_n}} B_{nj} \end{aligned}$$

$$
\begin{aligned}
&= R_{ij} + \frac{G_{in}^{(n-1)}}{G_{nn}^{(n-1)}} B_{nj} - \frac{G_{in}^{(n-1)}}{G_{nn}^{(n-1)} + 1/R_{e_n}} B_{nj} \\
&= R_{ij} + \frac{G_{in}^{(n-1)}/G_{nn}^{(n-1)}}{1/G_{nn}^{(n-1)} + R_{e_n}} R_{nj} \\
&= R_{ij} - \frac{R_{in}}{R_{nn} + R_{e_n}} R_{nj}
\end{aligned}
$$

Case 2. If $i = n$, according to Eqn (A.1), (A.3) and (A.4),

$$
\begin{aligned}
(R_{nj})_n &= \frac{(B_{nj})_n}{(G_{nn}^{(n-1)})_n} \\
&= \frac{G_{nn}^{(n-1)}}{G_{nn}^{(n-1)} + 1/R_{e_n}} R_{nj} \\
&= \frac{R_{e_n}}{R_{nn} + R_{e_n}} R_{nj} \qquad \text{(A.5)}
\end{aligned}
$$

Eqn (A.5) is in fact a special case of Eqn (2.5) when $i = n$. Therefore, Eqn (2.5) holds for all $i, j \in N$.

□

A.2 Proof of Theorem 2.2

For the proof of Theorem 2.2, the following lemma is established first:

Lemma A.1

$$
\frac{R_{n_{max}n_{max}} - R_{n_{max}n_{min}}}{R_{n_{max}n_{max}}} \geq \frac{R_{n_{max}n} - R_{n_{min}n}}{R_{nn}} \qquad \text{(A.6)}
$$

Proof:

According to the definition and property of resistance matrix:

$R_{n_{min}n} \geq min(R_{n_{max}n}, R_{n_{max}n_{min}})$, thus,

$R_{n_{min}n} R_{n_{max}n_{max}} \geq min(R_{n_{max}n} R_{n_{max}n_{max}}, R_{n_{max}n_{min}} R_{nn})$.

Since $R_{n_{max}n_{max}} R_{nn} \geq max(R_{n_{max}n} R_{n_{max}n_{max}}, R_{n_{max}n_{min}} R_{nn})$, we have,

$R_{n_{max}n_{max}} R_{nn} + R_{n_{min}n} R_{n_{max}n_{max}} \geq R_{n_{max}n} R_{n_{max}n_{max}} + R_{n_{max}q} R_{nn}$, i.e.,

$$
\frac{R_{n_{max}n_{max}} - R_{n_{max}n_{min}}}{R_{n_{max}n_{max}}} \geq \frac{R_{n_{max}n} - R_{n_{min}n}}{R_{nn}}
$$

□

According to Lemma A.1, Theorem 2.2 can be proven as follows:

Theorem 2.2 *A link to n_{max} has the largest upper bound on both maximum delay and skew reduction.*

Proof:

Since $R_{n_{max}n}/R_{nn} \leq 1$,

$$\overline{(\Delta D_{max})_{n_{max}}} = D_{max} > R_{n_{max}n}/R_{nn}D_n = \overline{(\Delta D_{max})_n}, \forall n \neq n_{max}$$

From Lemma A.1,

$$\begin{aligned}\overline{(\Delta DS_{max})_{n_{max}}} &= \frac{R_{n_{max}n_{max}} - R_{n_{max}n_{min}}}{R_{n_{max}n_{max}}} D_{max} \\ &> \frac{R_{n_{max}n} - R_{n_{min}n}}{R_{nn}} D_n = \overline{(\Delta DS_{max})_n}, \forall n \neq n_{max}\end{aligned}$$

i.e., link to n_{max} may achieve the largest reduction in both maximum delay and skew of the net.

□

A.3 Proof of Theorem 2.3

Theorem 2.3 *(Choice of Node for Link Insertion: Delay) With the same routing area consumption, the link to n_{max} achieves the largest reduction in D_{max} compared with link to any other node under A1 and A2 in Situation I.*

Proof:

For distributed RC line topology N, the following approximations are made with respect to the route length p_n from node n to n_{ref}:

1. The resistance between node n and the source, R_{nn}, is proportional to p_n, i.e., $R_{nn} \propto p_n$.
2. The Elmore delay at node n due to the contribution from wire resistance and capacitance is proportional to the square of p_n, i.e., $D_n \propto (p_n)^2$.

Under these two approximations, A2 in Situation I can be translated into constraints on ratios between node resistances (delays) and ratios between lengths of inserted links:

$$R_{n_{max}n_{max}}/R_{nn} \geq l_{n_{max}}/l_n \quad and \quad D_{max}/D_n \geq (l_{n_{max}}/l_n)^2, \forall n \in N \tag{A.7}$$

Since the inserted links to node n_{max} and n have the same routing area, we have:

$$C_{e_{max}} \approx C_{e_n}, \; R_{e_{max}} \approx (l_{n_{max}}/l_n)^2 R_{e_n}, \; and \; D_{e_{max}} \approx (l_{n_{max}}/l_n)^2 D_{e_n} \tag{A.8}$$

under the following assumptions on the routing area $Area_e$, wire capacitance C_e and resistance R_e of the inserted link:

$$Area_e \approx l_e w_e,\ C_e \propto l_e w_e \ and\ R_e \propto l_e/w_e \tag{A.9}$$

Assuming that the maximum delay node n_{max} is unchanged during optimization (the situation when n_{max} switches is discussed in Sec. 2.5), the maximum delay of the net after link insertion to node n_{max} and n can be expressed respectively as follows according to Eqn (2.6):

$$\begin{aligned}(D_{max})_{n_{max}} &= D_{max} - \frac{R_{n_{max}n_{max}}}{R_{n_{max}n_{max}} + R_{e_{max}}}(D_{max} - D_{e_{max}}) + R_d C_{e_{max}} \\ (D_{max})_n &= D_{max} - \frac{R_{n_{max}n}}{R_{nn} + R_{e_n}}(D_n - D_{e_n}) + R_d C_{e_n}\end{aligned}$$

$(D_{max})_{n_{max}}$ and $(D_{max})_n$ are compared in the following two cases:

Case 1: If $l_{n_{max}}/l_n \geq 1$, D_{max} can be approximated by $(l_{n_{max}}/l_n)^2 D_n$, and $D_{e_{max}}$ by $(l_{n_{max}}/l_n)^2 D_{e_n}$ respectively. According to Eqn (A.7) to (A.9),

$$\frac{R_{n_{max}n_{max}}}{R_{n_{max}n_{max}} + R_{e_{max}}}(D_{max} - D_{e_{max}}) \geq \frac{(l_{n_{max}}/l_n)^2 R_{n_{max}n_{max}}}{R_{n_{max}n_{max}} + (l_{n_{max}}/l_n)^2 R_{e_n}}(D_n - D_{e_n})$$

Since $l_{n_{max}}/l_n \geq 1$, it can be easily shown that:

$$\frac{(l_{n_{max}}/l_n)^2 R_{n_{max}n_{max}}}{R_{n_{max}n_{max}} + (l_{n_{max}}/l_n)^2 R_{e_n}} \geq \frac{R_{n_{max}n}}{R_{nn} + R_{e_n}}$$

When link insertion leads to delay reduction of the net, the delay introduced by the inserted link is less than the delay at the node it connects to, i.e., $D_{e_n} < D_n$, and thus

$$\frac{R_{n_{max}n_{max}}}{R_{n_{max}n_{max}} + R_{e_{max}}}(D_{max} - D_{e_{max}}) \geq \frac{R_{n_{max}n}}{R_{nn} + R_{e_n}}(D_n - D_{e_n})$$

Case 2: If $l_{n_{max}}/l_n < 1$, D_{max} can be approximated by D_n and $D_{e_{max}}$ by D_{e_n}, i.e.,

$$\begin{aligned}\frac{R_{n_{max}n_{max}}}{R_{n_{max}n_{max}} + R_{e_{max}}}(D_{max} - D_{e_{max}}) &\geq \frac{R_{n_{max}n_{max}}}{R_{n_{max}n_{max}} + (l_{n_{max}}/l_n)^2 R_{e_n}}(D_n - D_{e_n}) \\ &\geq \frac{R_{n_{max}n}}{R_{nn} + R_{e_n}}(D_n - D_{e_n})\end{aligned}$$

Since $R_d C_{e_{max}} = R_d C_{e_n}$ according to Eqn (A.9), $(D_{max})_{n_{max}} < (D_{max})_n$ holds in both cases, i.e., the link to n_{max} leads to the largest reduction in maximum delay compared with link to any other node under the same routing area consumption.

□

A.4 Proof of Theorem 2.4

Theorem 2.4 *(Choice of Node for Link Insertion: Skew) With the same routing area consumption, the link to n_{max} achieves the largest reduction in DS_{max} compared with link to any other node under A1 and A2 in Situation I.*

Proof:

Since DS_{max} exists between maximum delay node n_{max} and minimum delay node n_{min} of the net, the maximum delay skews after link insertion to node n_{max} and n respectively can be expressed as:

$$\begin{aligned}
(DS_{max})_{n_{max}} &= (D_{max})_{n_{max}} - (D_{min})_{n_{max}} \\
&= DS_{max} - \frac{R_{n_{max}n_{max}} - R_{n_{max}n_{min}}}{R_{n_{max}n_{max}} + R_{e_{max}}}(D_{max} - D_{e_{max}}) \\
&= DS_{max} - \frac{R_{n_{max}n_{max}} - R_{n_{max}n_{max}}}{R_{n_{max}n_{max}}} \frac{D_{max} - D_{e_{max}}}{1 + R_{e_{max}}/R_{n_{max}n_{max}}} \\
(DS_{max})_n &= (D_{max})_n - (D_{min})_n \\
&= DS_{max} - \frac{R_{n_{max}n} - R_{n_{min}n}}{R_{nn} + R_{e_n}}(D_n - D_{e_n}) \\
&= DS_{max} - \frac{R_{n_{max}n} - R_{n_{min}n}}{R_{nn}} \frac{D_n - D_{e_n}}{1 + R_{e_n}/R_{nn}}
\end{aligned}$$

According to Lemma A.1,

$$\frac{R_{n_{max}n_{max}} - R_{n_{max}n_{min}}}{R_{n_{max}n_{max}}} \geq \frac{R_{n_{max}n} - R_{n_{min}n}}{R_{nn}}$$

Similar to the proof of Theorem 2.3, we compare

$$\frac{D_{max} - D_{e_{max}}}{1 + R_{e_{max}}/R_{n_{max}n_{max}}} \quad with \quad \frac{D_n - D_{e_n}}{1 + R_{e_n}/R_{nn}}$$

under the following two cases:

Case 1: If $l_{n_{max}} \geq l_n$, we approximate D_{max} by $(l_{n_{max}}/l_n)^2 D_n$ and $D_{e_{max}}$ by $(l_{n_{max}}/l_n)^2 D_{e_n}$, thus,

$$\begin{aligned}
\frac{D_{n_{max}} - D_{e_{max}}}{1 + R_{e_{max}}/R_{n_{max}n_{max}}} &\geq \frac{(l_{n_{max}}/l_n)^2 (D_n - D_{e_n})}{1 + (l_{n_{max}}/l_n)^2 R_{e_n}/R_{n_{max}n_{max}}} \\
&\geq \frac{D_n - D_{e_n}}{1 + R_{e_n}/R_{nn}}
\end{aligned}$$

Case 2: If $l_{n_{max}} < l_n$, we approximate $D_{n_{max}}$ by D_n and $D_{e_{max}}$ by D_{e_n}, again:

$$\begin{aligned}
\frac{D_{n_{max}} - D_{e_{max}}}{1 + R_{e_{max}}/R_{n_{max}n_{max}}} &\geq \frac{D_n - D_{e_n}}{1 + (l_{n_{max}}/l_n)^2 R_{e_n}/R_{n_{max}n_{max}}} \\
&\geq \frac{D_n}{1 + R_{e_n}/R_{nn}}
\end{aligned}$$

Thus:

$$\frac{R_{n_{max}n_{max}} - R_{n_{max}n_{min}}}{R_{n_{max}n_{max}}} \frac{D_{n_{max}} - D_{e_{max}}}{1 + R_{e_{max}}/R_{n_{max}n_{max}}} \geq \frac{R_{n_{max}n} - R_{n_{min}n}}{R_{nn}} \frac{D_n - D_{e_n}}{1 + R_{e_n}/R_{nn}}$$

holds in both cases, i.e., $(DS_{max})_{n_{max}} \leq (DS_{max})_n$, the link to n_{max} achieves the largest maximum delay skew reduction under the same routing area consumption.

□