University of California

Los Angeles

Modeling and Optimization of VLSI Interconnects

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Lei He

1999

The dissertation of Lei He is approved.

Miodrag Potkonjak

Stephen E. Jacobsen

Milos D. Ercegovac

Jason Cong, Committee Chair

University of California, Los Angeles

1999

To My Wife Shiping Xu.

Table of Contents

List of Figures

List of Tables

Acknowledgments

My deep gratitude and appreciation first go to my advisor Prof. Jason Cong for his continuous encouragement, guidance, and support. While giving me the considerable freedom to conduct research on my own, he helped me to develop my skills to identify and solve a research problem, and to present its formulation and solution precisely and clearly. I believe that what I learned from him will guide me well beyond my graduation.

I am also grateful to Prof. Milos D. Ercegovac, Prof. Stephen E. Jacobsen, and Prof. Miodrag Potkonjak, for serving as members of my dissertation committee, and for their many helpful comments. I especially appreciate the help and advice that Prof. Potkonjak has given me on my career planning.

I thank Prof. Andrew B. Kahng for his help in my research. The main results presented in Chapter 3 were from a joint work with him. In addition, Dr. David Noice, Mr. Nagesh Shirali and Dr. Steve H.-C. Yen from Cadence Design Systems, Inc. also provided their inputs and supports.

I enjoyed the opportunity to work with Dr. Norman Chang, Dr. Shen Lin and Dr. O. Sam Nakagawa, during my research internship with Hewlett-Packard Laboratories. Discussions with them stimulated results presented in Chapter 7 (and in a joint paper with them). I especially appreciate Dr. Chang's help and advice for my career planning.

I treasure my experience with UCLA, and would like to thank these wonderful people in the VLSI CAD group and other groups in the Computer Science Department. To name a few, they are Eugene Ding, Patrick H. Madden, Kei-Yong Khoo, John C. Peck Jr., Albert Chung-Wen Tsao, Dennis Jenhsin Huang, Yutao He, Chang Wu, Chin-Chih Chang, Jie Fang, Sung Lim, Songjie Xu. Zhigang

Pan helped to collect partial results for comparisons presented in Chapter 6. Ms. Verra Morgan gave me a lot of advice on both academic and personal lives. I owe her a special appreciation.

My deep gratitude and appreciation also go to many people outside UCLA, especially Prof. Lawrence Pileggi, Prof. Sachin Sapatnekar, and Prof. Martin D. F. Wong. They gave me invaluable help and advice on my career planning. Prof. Dian Zhou provided parameters for the MCNC technology used in Chapter 2, and Mr. Heming Chan at Intel Design Technology Department provided multiple source routing examples used in the same chapter.

I am most grateful to my parents, who have always been a source of motivation for me. I dedicate this dissertation to my wife Shiping Xu. This achievement is so humble when compared with her constant support and inspiration, which are more precious during her own journey towards the JD degree from the Stanford Law School.

This work was supported in part by Defense Advanced Research Project Agency (DARPA) under Contracts DAAL01-96-K-3600 and J-FBI-93-112, by the National Science Foundation (NSF) Young Investigator Award MIP9357582, by grants from Intel Corporation and Cadence Design Systems, Inc. under the California MICRO Program, by a GTE Fellowship, and by a Prize for Engineering and Technology from Dimitris N. Chorafas Foundation.

Vita

1968	Born, People's Republic of China.
1986–1990	B.S. program in Electrical Engineering, Fudan University, Shanghai, China. Top Student Award, 1987 and 1988. Best Graduating Student Award, 1990.
1990–1992	Research Associate, VLSI CAD Laboratory, Fudan University. Prototyped a CAD framework environment.
1991	Teaching assistant for an upper level undergraduate course: Analog IC Design, Electrical Engineering Dept., Fudan University.
1992-1994	MS program in Electrical Engineering, Fudan University. Motorola Fellowship, 1993. Best Paper Award from China's Computer Association CAD/CAM Conference, 1993.
1992-1994	Graduate Student Researcher, VLSI CAD Laboratory, Fudan University. Developed a fast-timing simulation tool, and a digital-analog mixed-mode SPICE.
1994	Teaching assistant for an upper level undergraduate course: Fundamentals of Statistics, Electrical Engineering Dept., Fudan University.
1994 to present	PhD Program in Computer Science, UCLA. Prize for Engineering and Technology, the Dimitris N. Chorafas Foundation, 1997. GTE Fellowship, 1998.

1994 to present	Graduate Research Assistant, UCLA VLSI CAD Laboratory. Working on interconnect-driven design, involving interconnect capacitance and inductance extraction, device and interconnect modeling, and interconnect-centric layout optimization.
1996	Summer intern, Cadence Design Systems, Inc. Verified and automated a 2 1/2-dimensional capacitance extraction methodology shipped with the Cadence Silicon Ensemble 5.0 product.
1998	Research intern, Hewlett-Packard Laboratories. Proposed an efficient inductance extraction methodology; applied it to interconnect analysis, planning, and optimization for the state-of-the-art microprocessor designs.

Publications

L. He, K. H. Zhang, and P. S. Tang, "An efficient feedback processing method for relaxation based fast timing simulation", in *Proc. IEEE Int'l Symposium on VLSI Technology, Systems, and Applications*, May 1993.

L. He, S. Chen, K. H. Zhang, and P. S. Tang, "Implementation of digital-analog mixed-mode simulation inside SPICE", in *Proc. Int'l Conf. on Computer-Aided Design and Computer Graphics*, Vol. 2, pages 577-81, Aug 1993.

L. He, K. H. Zhang, and P. S. Tang, "A switch-level fast-timing simulator," In

Proc. Int'l Conf. on Computer-Aided Design and Computer Graphics, Vol. 2, pages 565-70, Aug 1993.

Y. Q. Zhang, L. He, J. R. Tong, and P. S. Tang, "An integrated CAD software development environment," *Journal of China's Computer-Aided Design and Computer Graphics*, Vol. 5, No. 3, 1993.

L. He, J. R. Tong, and P. S. Tang, "Development and maintenance of CAD software," *Journal of China's Computer-Aided Design and Computer Graphics*, Vol. 6, No. 1, 1994.

L. He, K. H. Zhang, and P. S. Tang, "Fast timing simulation considering feedback processing," *Journal of China's Electronics*, April 1994.

L. He, K. H. Zhang, and P. S. Tang, "FTSIM: a switch level fast timing simulator," *Acta Electronica Sinica*, Vol.23, (no.2):17-21, Feb. 1995.

J. Cong and L. He, "Optimal wiresizing for interconnects with multiple sources," in *Proc. ACM/IEEE Int'l Conf. on Computer-Aided Design*, November 1995.

J. Cong and L. He, "Simultaneous transistor and interconnect sizing based on the general dominance property," in *Proc. ACM SIGDA Int'l workshop on Physical Design*, April 1996.

J. Cong and L. He, "Optimal wiresizing for interconnects with multiple sources," *ACM Trans. on Design Automation of Electronic Systems*, pages 478-511, Oct.

1996.

J. Cong, L. He, C.-K. Koh, and P. H. Madden, "Performance optimization of VLSI interconnect layout," *Integration, the VLSI Journal (Invited)*, Vol. 21, No. 1&2, pages 1-94, November 1996.

J. Cong and L. He, "An efficient approach to simultaneous transistor and interconnect Sizing," in *Proc. ACM/IEEE Int'l Conf. on Computer-Aided Design*, November, 1996.

J. Cong, L. He, A. B. Kahng, D. Noice, N. Shirali, and S. H.-C. Yen, "Analysis and justification of a simple, practical 2 1/2-D capacitance extraction methodology," in *Proc. ACM/IEEE Design Automation Conference*, pages 627-632, June 1997.

J. Cong, L. He, K.-Y. Khoo, C.-K. Koh, and Z. Pan, "Interconnect design for deep submicron ICs," *Proc. ACM/IEEE Int'l Conf. on Computer-Aided Design (Embedded Tutorial)*, pages 478–485, November 1997.

J. Cong, L. He, C.-K. Koh, and Z. Pan, "Global interconnect sizing and spacing with consideration of coupling capacitance," *Proc. ACM/IEEE Int'l Conf. on Computer-Aided Design*, pages 628–633, November 1997.

J. Cong and L. He, "An efficient technique for device and interconnect optimization in deep submicron designs," in *Proc. ACM Int'l Symposium on Physical Design*, April 1998.

J. Cong and L. He, "Theory and algorithm of local-refinement based optimization with application to device and interconnect sizing," *IEEE Trans. on Computer-Aided Design*, pages 406-420, April 1999.

L. He, N. Chang, S. Lin, and O. S. Nakagawa, "An efficient inductance modeling for on-chip interconnects," in *Proc. IEEE Custom Integrated Circuits Conference*, pages 457-460, May 1999.

Abstract of the Dissertation

Modeling and Optimization of VLSI Interconnects

by

Lei He

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 1999

Professor Jason Cong, Chair

As very large scale integrated circuits move into the era of deep-submicron technology and gigahertz clock frequency, the system performance has increasingly become dominated by the interconnect delay. Contributions of this dissertation include the following three related aspects for the design of high-performance interconnects: interconnect modeling, interconnect optimization, and optimization theory.

Concerning interconnect modeling, we study the two extraction problems to compute the capacitance and inductance values respectively from complex 3-dimensional interconnect structures. For each problem, we first propose and validate "foundations" that can be used to reduce the problem complexity, then present a simple yet accurate methodology directly based on the foundations. The resulting extraction methodologies are applicable to both IC and MCM/PCB designs, and have been used in the industry for real designs.

Concerning interconnect optimization, we formulate and solve the following three problems:

- The multi-source wire sizing (*MSWS*) problem. It determines the optimal

widths for all wire segments to minimize the delay along a given routing tree with multiple sources.

- The simultaneous transistor and interconnect sizing (*STIS*) problem using the accurate device model. It assigns optimal wire widths to interconnects and optimal sizes to transistors to minimize the delay for multiple critical paths, where a path may contain multiple routing trees and related devices.

- The global interconnect sizing and spacing (*GISS*) problem. Given the topology for multiple routing trees, the GISS problem finds the wire sizing and spacing solution *simultaneously* for all trees, with consideration of coupling capacitance between them.

We reveal interesting properties and present efficient algorithms for all those problems. Experiments show that our formulations and algorithms have achieved significant reduction of the interconnect delay.

As to the optimization theory, we formulate three classes of optimization problems: the simple, monotonically constrained, and bounded CH-problems. We reveal that the dominance property holds for those CH-problems under different types of local-refinement operations. This property immediately leads to an efficient polynomial-time algorithm, which provides a unified solution to a number of interconnect optimization problems, including the MSWS, STIS, and GISS problems.

CHAPTER 1

Introduction

1.1 Computer-Aided Design for VLSI Circuits and Systems

Following the prediction of the Moore's Law [57], the complexity of integrated circuits was doubled for every 18 months in the past three decades. As projected by the 1997 National Technology Roadmap for Semiconductors(NTRS'97) [78] (see Table 1.1), this exponential growth will continue for at least another decade, and a single-chip with 21 million transistors and more than 1 gigahertz clock frequency is expected by the end of this year (1999). Given such a high complexity and clock frequency, the only feasible way to finish the design within a reasonable amount of time is the computed-aided design.

As shown in Figure 1.1, the computer-aided design of VLSI circuits and systems is typically divided into the following stages:

- *High level synthesis*: based on the circuit specifications, to generate the system behavior by RTL-level components, and to define the function for each component.

- *Logic synthesis*: to generate functions of RTL-level components by boolean expressions, and to implement boolean expressions by logic gates given in the library.

Tech. (μm)	0.25	0.18	0.15	0.13	0.10	0.07
Year	1997	1999	2001	2003	2006	2009
Transistor #	11M	21M	40M	76M	200M	520M
Across chip clock (MHz)	750	1200	1400	1600	2000	2500
Area (mm^2)	300	340	385	430	520	620
Wiring levels	6	6-7	7	7	7-8	8-9
Min. wire width (μm)	0.25	0.18	0.15	0.13	0.10	0.07
Min. wire spacing (μm)	0.34	0.24	0.21	0.17	0.14	0.10
Wire aspect ratio	1.8	1.8	2.0	2.1	2.4	2.7

Table 1.1: Summary of NTRS'97

- *Physical design*: to implement logic gates by geometric patterns, and to place and connect these geometric patterns to obtain the layout for fabrication.

Different objectives such as area, delay and power minimizations are often shared among these stages. Iterations between stages or within a stage are sometimes needed. For example, a failed physical design procedure may lead to a logic re-synthesis, or another physical design procedure based on the current logic synthesis result. Simulations are invoked after each stage to verify the function and timing correctness, and may lead to more iterations.

The last stage, physical design, contains the topics studied in this dissertation. It is also called *layout* design. For a layout, the geometric implementation for a logic gate is a *cell*. A set of related cells is a *block*, which often implement a specific logic function. Connection points on the block or cell are *pins*. A *net* is a set of pins to be electrically connected. Then, an integrated circuit (*IC*) can be represented by a *netlist*, which is a set of nets. Five iterative steps are involved

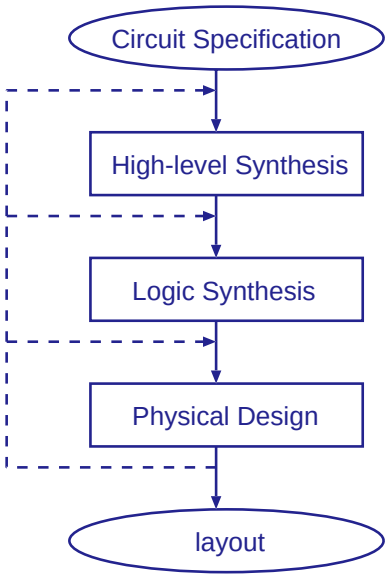


Figure 1.1: Overall flow for the computer-aided design of VLSI circuits and systems.

in the physical design process, namely, *partitioning*, *floorplanning*, *placement*, *global routing* and *detailed routing* (see Figure 1.2).

Partitioning is the first step for the physical design procedure. For the purpose of "divide-and-conquer", partitioning divides the netlist into a set of sub-netlist. Each sub-netlist contains a number of blocks, and is small enough to be handled by the design tools, or the silicon implementation capacities such as the maximum die area for ICs. The objective for partitioning often is to minimize the total number of nets that belong to more than one sub-netlist.

Floorplanning and placement determine shapes, orientations and locations for blocks and cells on a layout. Floorplanning handles *flexible* blocks whose shapes and orientations are not yet fully determined. Placement deals with *fixed* blocks that have specific layout and orientations. The objective for floorplanning and placement is to avoid overlap of blocks and cells, and to minimize the total wire

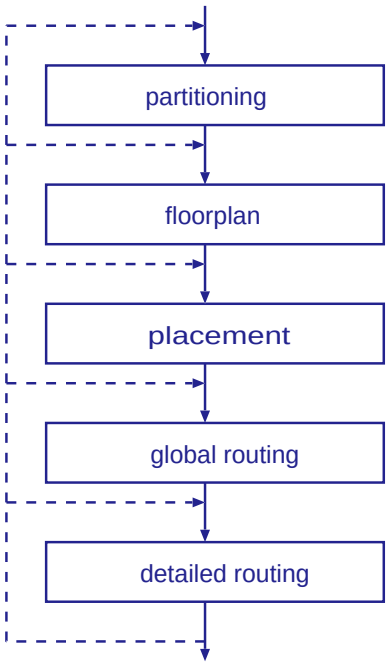


Figure 1.2: Conventional flow for physical design.

length and the chip area.

Global and detailed routing complete the electrical connections (via metal wires) between blocks or cells as indicated by the netlist. The entire routing area is divided into disjoint, smaller routing regions. Global routing only assigns each net to a set of routing regions to minimize the total routing area and the congestion in routing regions. The detailed routing implements nets within each and every routing region by metal wires with exact geometric specifications including layer assignments. In this dissertation, metal wires are also called as *interconnects* or *wires*.

1.2 Contributions of this Dissertation

As very large scale integrated circuits move into the era of deep-submicron technology and gigahertz clock frequency, the system performance has increasingly become dominated by the interconnect delay [33, 22]. Therefore, interconnect modeling and optimization is drawing considerable attention. Contributions of this dissertation include the following three related aspects for the design of high-performance interconnects: interconnect modeling, interconnect optimization, and optimization theory.

We study the following interconnect modeling problems:

- *Interconnect capacitance extraction* problem. It computes the capacitance values from complex 3-dimensional interconnect structures. We propose and validate five "foundations" that can be used to simplify capacitance extraction, then present a simple yet accurate 2 1/2-dimensional extraction methodology directly based on the foundations. This methodology is able to generate capacitance on fly during interconnect design and optimization, and can be used for both IC and MCM/PCB designs.

- *Interconnect inductance extraction* problem. It computes the inductance values from three-dimensional on-chip interconnect structures. Our approach is based on the partial inductance model. It is accurate and efficient, and again can be used during the interconnect design and optimization procedure. It has been used to generate distributed RLC models for on-chip interconnects with consideration of process variations, and has been extended to MCM/PCB designs. In addition, several design freedoms, including buffer insertion and shielding insertion, are studied to reduce the inductive effect.

We also investigate the following interconnect optimization problems:

- *Multi-source wire sizing (MSWS)* problem. For this problem, we are given a routing tree with multiple sources. The goal is to determine the optimal widths for all wire segments such that the delay is minimized. We further study the wire sizing problem without an *a priori* fixed wire segmenting, which allows us to use much finer wire segmenting for more delay reduction or less wire area, but still to be able to find the optimal solution in a timely manner.

- *Simultaneous transistor and interconnect sizing (STIS)* problem. We assume that the circuit netlist is given and the routing topology is fixed, and apply device models more accurate compared to many used in interconnect optimization works. We then assign optimal wire widths to all wire segments and optimal sizes to all transistors for minimizing the delay and/or power for multiple critical paths. Here, a path contains several nets/routing-trees. Compared to the single-net sizing problems such as the MSWS problem, this problem has a much higher complexity but is able to achieve more delay and power reduction.

- *Global interconnect sizing and spacing (GISS)* problem. Given the topology for multiple nets, the GISS problem finds the wire sizing and spacing solution optimal to *all* nets, with consideration of coupling capacitance between neighboring wires. The formulation uses coupling capacitance generated on fly during wire sizing and spacing procedure, and can be treated as part of the STIS problem. Therefore, we are able to consider simultaneous device sizing, and wire sizing and spacing for multiple paths, using accurate models for device delay and interconnect coupling capacitance.

In addition, this dissertation makes a nice contribution to the theory of the local-refinement (LR) based optimization. We formulate three classes of optimization problems: the simple, monotonically constrained, and bounded CH-programs. We reveal the dominance property (Theorem 8) under the local refinement (LR) operation for the simple CH-program, as well as the general dominance property (Theorem 9) under the pseudo-LR operation for the monotonically constrained CH-program and under the extended-LR operation for the bounded CH-program. These properties enable a very efficient polynomial-time algorithm, using different types of LR operations to compute tight lower and upper bounds of the exact solution to any CH-program. The formulation and algorithm for CH-programs unify solutions to several important CAD problems, including the MSWS, STIS, and GISS problems. Other algorithmic contributions will be summarized in Section 1.3, with details in Chapters 2-7.

Formulations and solutions to these problems belong to the general framework of the *interconnect design and optimization* problem. Given the netlist to connect devices, the problem is to determine the interconnect topologies, wire sizing and spacing solutions, buffer locations and sizes, driver/receiver sizes, and shielding insertion. The objective can be optimizing delay, power, signal integrity, and skew (in case of clock nets).

The solution to the interconnect design and optimization problem can be used in an innovative physical design flow to better handle the interconnect delay. As shown in Figure 1.3, the innovative physical design flow is different from the conventional physical design in the following aspects:

During floorplanning, *interconnect planning* is carried out simultaneously with planning for logic blocks. Based on certain estimation models (see [32] for an example on interconnect delay estimation considering the impact of wire sizing),

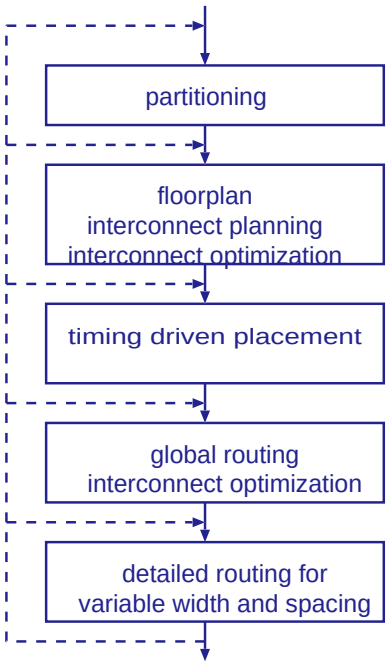


Figure 1.3: An innovative physical design flow.

the impact of wire sizing and spacing, interconnect coupling noise, shielding insertion, and buffer insertion must be considered by the interconnect planning. On the other hand, detailed but quick interconnect optimization may be carried out for critical nets. At the same time, placement should explicitly consider timing constraints, rather than simply minimize the total wire length. In addition, interconnect optimization techniques, especially buffer insertion, should be considered to meet time constraints.

Global routing extensively applies interconnect optimization to synthesize interconnect structures, including topologies, wire ordering and shielding solutions, wire widths and spacings, layer assignments, and buffer locations and sizes. Then, detailed routing geometrically implements the synthesized interconnect structures with multiple widths and spacings. For the sake of completeness, a post-layout

interconnect optimization can be invoked to further refine wire widths and spacings, and device sizes, with consideration of the exact configuration of interconnect structures.

1.3 Overview of the Dissertation

The remainder of the dissertation includes the following parts:

In Chapter 2, we study the multiple-source wiresizing (MSWS) problem. Our contributions include:

- We formulate the wiresizing problem to minimize delay for nets with multiple sources under the distributed RC model. Decomposing an MSIT into a source subtree (SST) and a set of loading subtrees (LSTs), we show a number of interesting properties of the optimal wiresizing solutions under this decomposition, including the LST separability, the LST monotone property, the SST local monotone property, and the dominance property. These properties lead to effective algorithms to compute the optimal wire width assignment for any given MSIT. We have tested our algorithm on multi-source nets extracted from the multi-layer layout of a high-performance Intel processor. HSPICE simulation shows that our methods reduce the average delay by up to 23.5% and the maximum delay by up to 37.8% for the submicron CMOS technology.

- We study the optimal wiresizing problem using a *variable* segment-division rather than an *a priori* fixed segment-division used in all previous works. We show the bundled refinement property that leads to a very efficient wire sizing algorithm based on bundled refinement operations and the segment-division refinement operations. The algorithm yields a speedup of over 100x

time and does not lose any accuracy, when compared to the method based on an a *priori* fixed segment-division. The algorithm has been extensively used in the practice.

- We also investigate the *fidelity* of the Elmore delay model for wiresizing optimization using a ranking technique. We have found that the optimal wiresizing solution selected according to the Elmore delay model is about 0.06% worse than the optimal wiresizing solution selected according to the SPICE-computed delay, when the delays of both solutions are measured by SPICE simulation. This experiment convincingly justifies our formulation based on the Elmore delay model for the current submicron CMOS technology.

These results were first presented in [14, 16]. To the best of our knowledge, it is the first work which presents an in-depth study of both the optimal wiresizing problem for MSITs and the optimal wiresizing problem under a *variable* segment-division.

In Chapter 3, we address interconnect capacitance extraction during interconnect optimization. Our contributions include:

- We show how basic drivers in process technology (planarization and minimum metal density requirements) actually simplify the extraction problem; we do this by proposing and validating five "foundations" through detailed experiments with a 3-D field solver on representative $0.50\mu m$, $0.35\mu m$ and $0.18\mu m$ process parameters.

- We present a simple yet accurate 2 1/2-D extraction methodology directly based on the foundations. This methodology has been productized and is being shipped with the Cadence Silicon Ensemble 5.0 product for the timing

verification purpose. The methodology is also able to generate capacitance on fly during interconnect design and optimization, and can be used for both IC and MCM/PCB designs.

These results were first presented in [21].

In Chapter 4, we present the theory and algorithm for local-refinement based optimization. Our contributions in this chapter include:

- We formulate three classes of optimization problems: the simple, monotonically constrained, and bounded CH-programs. The simple CH-program contains the single-source and multi-source wire sizing problems, and is a subset of the monotonically constrained CH-program. In turn, the monotonically constrained CH-program is a subset of the bounded CH-program.

- We generalize the concept of the LR operation, and introduce the concepts of the pseudo-LR operation and the extended-LR operation. We then reveal the dominance property (Theorem 8) under the LR operation for the simple CH-program, as well as the general dominance property (Theorem 9) under the pseudo-LR operation for the monotonically constrained CH-program and under the extended-LR operation for the bounded CH-program.

- Based on the dominance property and the general dominance property we propose a very efficient polynomial-time algorithm, using different types of LR operations to compute tight lower and upper bounds of the exact solution to any CH-program.

These results were first presented in [20, 19, 25]. Note that both the MSWS problem in Chapter 2 and the STIS and GISS problems to be presented in Chapters 5 and 6 belong to CH-programs. Therefore, the formulation and algorithm of CH-programs unify the solutions to all three problems.

In Chapter 5, we study the simultaneous transistor and interconnect sizing (*STIS*) problem. Our contributions include:

- We assume that the circuit netlist is given and routing topology is fixed, and then formulate the STIS problem to assign optimal wire widths to all wire segments and optimal sizes to all transistors, for minimizing the delay and/or power for multiple critical paths. Compared to the single-net sizing problems such as the MSWS problem, this problem has much higher complexity but is able to achieve more delay and power reduction.

- We apply the LR-based bound-computation algorithm presented in Chapter 4 to the STIS problem under both the simple device model and the accurate STL-bounded device model. The STL-bounded model is based on tables pre-computed from SPICE simulations for the device delay, so that it is much more accurate than many models used in previous device and interconnect optimization algorithms. Experiments show that the bound-computation algorithm can efficiently handle both simple and STL-bounded models, and obtain solutions close to the global optimum in both cases. According to SPICE simulations, the solution obtained by the STIS algorithm under the simple model achieves up to 14.4% delay reduction when compared to the solution given by manual optimization (reported in [10]). Furthermore, the solution obtained by the STIS algorithm under the accurate STL-model achieves up to 15.1% additional delay reduction when compared to the solution obtained by the STIS algorithm under the simple model.

These results were first presented in [20, 19, 25], and were among the first in-depth studies on the simultaneous interconnect and device sizing problems.

In Chapter 6, we study the interconnect sizing and spacing problem for the single-net and multiple nets, respectively. Our contributions include:

- We formulate the global interconnect sizing and spacing (*GISS*) problem based on the concept of *asymmetric wire sizing*. Given the topology for multiple nets, the problem finds the wire sizing and spacing solution optimal for *all* nets, and considers coupling capacitance extracted during wire sizing and spacing procedure.

- We pose the GISS problem as a CH-program, which directly leads to an effective and efficient solution based on bound computation using different types of local-refinement operations. Note that this GISS formulation can be treated as part of the STIS formulation. Therefore, we have a unified formulation and solution to the problem of simultaneous device sizing, and wire sizing and spacing for multiple paths, under accurate models for device delay and interconnect coupling capacitance.

- We also solve the single-net interconnect sizing and spacing (*SISS*) problem. It is a simpler version of the GISS problem assuming that neighboring wires are fixed for the specific net. Experiments show that GISS algorithm may achieve up to 39% delay reduction, when compared with SISS algorithm applied iteratively to multiple nets.

These results were first presented in [23, 19, 25], and were among the first in-depth studies of the global interconnect sizing and spacing problem using accurate capacitance model.

In Chapter 7, we study the inductance extraction problem for on-chip interconnects. Our contributions include:

- We propose and *theoretically* validate two foundations which allow us to reduce the problem size of inductance extraction without loss of accuracy, and present a table-based inductance extraction methodology directly based on the two foundations.

- We use this efficient inductance extraction methodology to generate distributed RLC models for on-chip interconnects with consideration of process variations. We have applied this RLC model to interconnect modeling and optimization for designs of the state-of-the-art microprocessors in Hewlett-Packard Company.

These result were first present in [44]. To the best of our knowledge, it is the first work that presents an efficient and accurate table-based inductance extraction method for on-chip interconnects. Note that the table-based inductance extraction has been recently extended to off-chip interconnects in MCM/PCB designs.

In Chapter 8, we discuss the future works and conclude the dissertation. Finally, several proofs for the MSWS problem are presented in the Appendix (Chapter 9).

CHAPTER 2

Wiresizing Optimization for Nets with Multiple Sources

All wiresizing works [30, 29, 73, 82, 7, 6, 56, 83] and simultaneous device and wire sizing works [26, 55, 49] assume that there is a unique source in each interconnect tree (called *single source interconnect tree (SSIT)*) and minimize the delay between the source and a set of critical sinks. However, there exist many important interconnect structures with multiple potential sources, each driving the interconnect at a different time, such as those in global signal buses. We call such interconnect structures as *multi-source interconnect trees (MSITs)*. Even those single-source wiresizing algorithms based on the mathematical programming [72, 56, 55, 82, 6] or the sensitivity analysis [73, 83] might be adapted to minimize the delay between the multiple source-sink pairs by modifying their objective functions, none of existing sizing works explicitly considers MSITs. It is of both theoretical and practical interest to understand the properties of the optimal wiresizing solutions for MSITs and develop efficient algorithms directly for MSITs.

In this chapter, we study the multiple-source wiresizing (MSWS) problem. Our contributions include:

- We formulate the wiresizing problem to minimize delay for nets with multiple sources under the distributed RC model. Decomposing an MSIT into a

source subtree (SST) and a set of loading subtrees (LSTs), we show a number of interesting properties of the optimal wiresizing solutions, including the LST separability, the LST monotone property, the SST local monotone property, and the dominance property. These properties lead to effective algorithms to compute the optimal wire width assignment for any given MSIT. We have tested our algorithm on multi-source nets extracted from the multi-layer layout of a high-performance Intel processor. SPICE simulation shows that our methods reduce the average delay by up to 23.5% and the maximum delay by up to 37.8% for the submicron CMOS technology.

- We also study the optimal wiresizing problem using a *variable* segment-division rather than an *a priori* fixed segment-division used in all previous works. We show the bundled refinement property that leads to a very efficient wire sizing algorithm based on bundled refinement operations and the segment-division refinement operations. The algorithm yields a speedup of over 100x time and does not lose any accuracy, when compared to the method based on an *a priori* fixed segment-division.

- Finally, we investigate the fidelity of the Elmore delay model for wiresizing optimization using the ranking technique similar to [4]. We have found that the optimal wiresizing solution selected according to the Elmore delay model is about 0.06% worse than the optimal wiresizing solution selected according to the SPICE-computed delay, when the delays of both solutions are measured by SPICE simulation. This experiment convincingly justifies our formulation based on the Elmore delay model for the current submicron CMOS technology.

Those results are first presented in [14, 16]. To the best of our knowledge, it is the first work which presents an in-depth study of both the optimal wiresizing

problem for MSITs and the optimal wiresizing problem under a *variable* segment-division.

The remainder of this chapter is organized as follows: In Section 2.1, we present the formulation of the *MSIT* wiresizing problem. In Section 2.2 and 2.3, we study the properties of the optimal wiresizing solutions for *MSIT* designs, under the a *priori* fixed and the variable segment-divisions, respectively. These properties lead to efficient algorithms given in Section 2.4. Section 2.5 shows experimental results, including the fidelity study of the Elmore delay model. Section 2.6 concludes the chapter with discussions of future works. The proofs of the Theorems 3, 5 and 6 are given in the Appendix at the ending of this proposal. Proofs of other theorems, together with more experimental results, can be found in a technical report [15].

2.1 Problem Formulation

2.1.1 Multi-Source Wiresizing (MSWS) Problem

We call the wiresizing problem for MSITs as the *multi-source wiresizing (MSWS) problem*. For an MSIT, each pin in the MSIT can be a source (driver), or a sink (receiver), or both at different times. We assume, however, no two sources in the MSIT are active at the same time. Let a *node* be either a pin or a Steiner point in the MSIT and $src(MSIT)$ the set of pins which can be sources of the MSIT, we say that $sink^i(MSIT)$ is the set of sinks in the MSIT when pin N_i is the source of the MSIT. Besides, let a *segment* connect two *node*s and $\{S_1, S_2, \cdots, S_m\}$ be the set of segments in the MSIT. In order to capture the distributed resistive property of interconnects and achieve better wiresizing solutions, a segment is divided into a sequence of *uni-segments*. The term of "uni-segment" is coined

based on this assumption that the wire width is *uniform* within a uni-segment. The *segment-division* determines the set of all uni-segments, $\{E_1, E_2, \cdots, E_n\}$, in the MSIT. Our wiresizing problem is formulated to find a wire width for each uni-segment from a set of given choices $\{W_1, W_2, \cdots, W_r\}$ $(W_1 < W_2 < \cdots < W_r)$. Different from our formulation, a segment in [29] is not further divided and is simply treated as a uni-segment[1], and a segment in [26] is divided into a sequence of wires of unit length and such a wire of unit length is treated as a uni-segment. Thus, both segment-divisions in [29, 26] are given *a priori* and fixed during the wiresizing procedure. In our formulation, the segment-division is in fact a variable during the wiresizing procedure and is defined by the wiresizing procedure, which will be discussed later in Section 2.3. For simplicity, we assume that an *a priori* fixed segment-division is given in this section.

The modeling technique similar to those used in [29] is applied. Each uni-segment is treated as a π-type RC circuit containing resistance r_E and capacitance c_E, respectively. Let the unit-width unit-length wire have wire resistance r_0, wire area capacitance c_a and wire fringing capacitance c_f, then $r_E = r_0 \cdot \frac{l_E}{w_E}$ and $c_E = c_a \cdot w_E \cdot l_E + c_f \cdot l_E$ for uni-segment E with width w_E and length l_E. The driver at source N_i is modeled by an output capacitance C_d^i and a fixed-value resistor R_d^i connected to an idle voltage source, and the receiver at sink N_j by a loading capacitor c_s^j. Thus, a given interconnect including its drivers and receivers is modeled by a distributed RC tree. The Elmore delay [37] t^{ij} in the RC tree from source N_i to sink N_j is a function of the segment-division $\mathcal{E}$ and the wiresizing solution $\mathcal{W}$. It can be written as the Eqn. (2.1) according to the Elmore delay

[1] we note that artificial degree-2 Steiner points can be introduced within a segment in [29] to achieve certain segment-division.

formulation for RC trees [65].

$$t^{ij}(MSIT, \mathcal{E}, \mathcal{W}) = \sum_{E \in P(N_i, N_j)} r_E \cdot (\frac{c_E}{2} + C_E) \tag{2.1}$$

where the summation is taken over all uni-segments on the unique path $P(N_i, N_j)$ from source N_i to sink N_j, and C_E is the total downstream capacitance of uni-segment E with respect to source N_i. In order to handle multiple source-sink pairs, we further introduce the following weighted delay formulation Eqn. (2.2).

$$t(MSIT, \mathcal{E}, \mathcal{W}) = \sum_{N_i \in src(MSIT)} \sum_{N_j \in sink^i(MSIT)} \lambda^{ij} \cdot t^{ij}(MSIT, \mathcal{E}, \mathcal{W}) \tag{2.2}$$

where λ^{ij} is the penalty weight to indicate the priority of the Elmore delay t^{ij} between source N_i and sink N_j.

With these definitions, we give the general formulation of the MSWS problem as follows:

Formulation 1 *Given an MSIT, a segment-division $\mathcal{E}$ and a set of possible wire width choices, the multi-source wiresizing (MSWS) problem for delay minimization is to determine a wiresizing solution $\mathcal{W}$ which gives a wire width w_E for every uni-segment E under $\mathcal{E}$, such that the weighted delay $t(MSIT, \mathcal{E}, \mathcal{W})$ is minimized.*

When there is only one source in an interconnect tree, the MSWS problem degenerates into the *single-source wiresizing (SSWS)* problem. Note that we assume a given segment-division in Formulation 1. A more general wiresizing problem, the multi-source wiresizing problem *without* an *a priori* given segment-division *(the MSWS/E problem)* will be presented in Section 2.3.

2.1.2 Weighted Delay Formulation

For simplicity, we assume that all interconnects belong to the same layer and the assumption will be removed later in Section 2.2.3. It is not difficult to verify that the Elmore delay t^{ij} between source N^i and sink N^j can be formulated as follows:

$$\begin{aligned} & t^{ij}(MSIT, \mathcal{E}, \mathcal{W}) \\ = \; & \mathcal{K}_0^{ij} + \mathcal{K}_1^i \cdot \sum_{E \in MSIT} l_E \cdot w_E + \\ & \mathcal{K}_2 \cdot \sum_{E,E' \in MSIT} f^{ij}(E,E') \cdot \frac{l_E \cdot l_{E'} \cdot w_{E'}}{w_E} + \\ & \mathcal{K}_3 \cdot \sum_{E,E' \in MSIT} f^{ij}(E,E') \cdot \frac{l_E \cdot l_{E'}}{w_E} + \\ & \mathcal{K}_4 \cdot \sum_{E} g^{ij}(E) \cdot \frac{l_E}{w_E} + \mathcal{K}_5 \cdot \sum_{E \in MSIT} h^{ij}(E) \cdot \frac{l_E^2}{w_E} \end{aligned} \tag{2.3}$$

where w_E and l_E are respectively the (wire) width and length of the uni-segment E. $\mathcal{K}_0^{ij}, \mathcal{K}_1^i, \mathcal{K}_2, \cdots, \mathcal{K}_5$ are constants independent of the wiresizing solution, as given in the following:

$$\begin{aligned} \mathcal{K}_0^{ij} &= R_d^i \cdot C_d^i + R_d^i \cdot \sum_{u \in sink^i(MSIT)} c_s^u + R_d^i \cdot \sum_{E \in MSIT} c_f + \sum_{E \in P(N_i, N_j)} \frac{r_0 \cdot c_a}{2} \\ \mathcal{K}_1^i &= R_d^i \cdot c_a \\ \mathcal{K}_2 &= r_0 \cdot c_a \\ \mathcal{K}_3 &= r_0 \cdot c_f \\ \mathcal{K}_4 &= r_0 \\ \mathcal{K}_5 &= \frac{r_0 \cdot c_f}{2} \end{aligned}$$

Recall that R_d^i and C_d^i are the driving resistance and output capacitance for the driver at source N_i, and c_s^u the sink capacitance at sink N_u. These parameters can take account the different sizes of drivers/receivers at different sources/sinks of an MSIT. Besides, $f^{ij}(E,E'), g^{ij}(E)$ and $H^{ij}(E)$ defined below, again, are constants

independent of the wiresizing solution.

$$f^{ij}(E,E') = \begin{cases} 1 & if\ E \in P(N_i,N_j)\ and\ E' \in Des^i(E) \\ 0 & otherwise \end{cases} \quad (2.4)$$

$$g^{ij}(E) = \begin{cases} \sum_{u \in sink^i(E)} c_s^u & if\ E \in P(N_i,N_j) \\ 0 & otherwise \end{cases} \quad (2.5)$$

$$h^{ij}(E) = \begin{cases} 1 & if\ E \in P(N_i,N_j) \\ 0 & otherwise \end{cases} \quad (2.6)$$

where $Des^i(E)$ is the set of downstream uni-segments of E with respect to source N_i, and $sink^i(E)$ the set of downstream sinks of E with respect to source N_i.

Assume that λ^{ij}'s are normalized, i.e.,

$$\sum_{N_i \in src(MSIT)} \sum_{N_j \in sink^i(MSIT)} \lambda^{ij} = 1$$

the objective function Eqn. (2.2) becomes:

$$\begin{aligned} & t(MSIT,\mathcal{E},\mathcal{W}) \\ = \ & \mathcal{K}_0 + \mathcal{K}_1 \cdot \sum_{E \in MSIT} l_E \cdot w_E + \\ & \mathcal{K}_2 \cdot \sum_{E,E' \in MSIT} F(E,E') \cdot \frac{l_E \cdot l_{E'} \cdot w_{E'}}{w_E} + \\ & \mathcal{K}_3 \cdot \sum_{E,E' \in MSIT} F(E,E') \cdot \frac{l_E \cdot l_{E'}}{w_E} + \\ & \mathcal{K}_4 \cdot \sum_{E \in MSIT} G(E) \cdot \frac{l_E}{w_E} + \mathcal{K}_5 \cdot \sum_{E \in MSIT} H(E) \cdot \frac{l_E^2}{w_E} \end{aligned} \quad (2.7)$$

where

$$\mathcal{K}_0 = \sum_{N_i \in src(MSIT)} \sum_{N_j \in sink^i(MSIT)} \lambda^{ij} \cdot K_0^{ij} \quad (2.8)$$

$$\mathcal{K}_1 = \sum_{N_i \in src(MSIT)} \sum_{N_j \in sink^i(MSIT)} \lambda^{ij} \cdot K_1^{i} \quad (2.9)$$

$$F(E,E') = \sum_{N_i \in src(MSIT)} \sum_{N_j \in sink^i(MSIT)} \lambda^{ij} \cdot f^{ij}(E,E') \quad (2.10)$$

$$G(E) = \sum_{N_i \in src(MSIT)} \sum_{N_j \in sink^i(MSIT)} \lambda^{ij} \cdot g^{ij}(E) \quad (2.11)$$

$$H(E) = \sum_{N_i \in src(MSIT)} \sum_{N_j \in sink^i(MSIT)} \lambda^{ij} \cdot h^{ij}(E) \quad (2.12)$$

Our MSWS problem is aimed to find the optimal w_E's to minimize the weighted delay formulation Eqn. (2.7). Although this weighted delay formulation for multiple sources and multiple sinks is very similar to that for the single source and multiple sinks in [29], the coefficient functions F, G and H have very different properties, which lead to much higher complexity and very different properties for the MSWS problem when compared to the *SSWS* problem. These properties will be discussed in Section 2.2.

2.2 Properties of Optimal MSWS Solutions

The single-source wiresizing problem (SSWS) under the an *a priori* fixed segment-division was studied in [29], and the polynomial-time optimal algorithm was developed based on the separability, the monotone property and the dominance property. The presence of multiple sources, however, greatly complicates the wiresizing problem. For example, with multiple sources, even a monotone wiresizing solution is not well defined. Nevertheless, our research have revealed a number of interesting properties of the optimal MSWS solutions under the decomposition of MSITs. Some of them generalize the results on the SSWS problem, and others are unique for the MSWS problem. These properties to be presented in this section and Section 2.3 will enable us to apply the algorithms developed in [29] to the MSWS problem to certain extent and to develop even more efficient algorithms in Section 2.4.

2.2.1 Decomposition of an MSIT

When there is only one source in the routing tree, each segment has a unique signal direction and the ancestor-descendant can be defined with respect to the direction. The MSWS problem is most complicated by the fact that, in general, there is no fixed signal direction for a segment. In order to reduce the complexity with the MSWS problem, we decompose an MSIT into the *source subtree (SST)* and a set of *loading subtrees (LSTs)* (see Figure 2.1). The SST [2] is the subtree spanned by all source nodes in the MSIT. After we remove the SST from the MSIT, the remaining segments form a set of subtrees, each of them is called an LST. When every pin of an MSIT can be a source at different times, the entire MSIT becomes the SST and there is no LST.

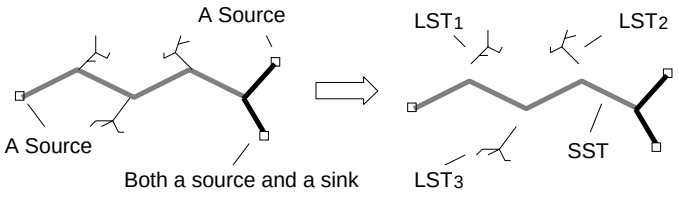


Figure 2.1: An $MSIT$ can be decomposed into the source subtree SST, and a set of loading subtrees (three LSTs here) branching off from the SST. The dark segments belong to the SST.

Parallel to the ancestor-descendent relation in an *SSIT*, the left-right relation is introduced in an MSIT. We choose an arbitrary source as the leftmost node $Lsrc$. The direction of the signal flowing out from $Lsrc$ is defined as the right direction along each segment S. Under such definition, the signal in any LST always flows rightward, but the signal may flow either leftward or rightward in a

[2]Note that SST defined here is different from that defined in [29], where SST is used to denote a single stem tree.

segment in the SST. The properties of optimal MSWS solutions will be studied in the context of the MSIT decomposition.

2.2.2 Properties of Optimal MSWS Solutions

A. LST Separability

Theorem 1 *Given the wire width assignment of the SST, the optimal width assignment for each LST branching off from the SST can be carried out independently. Furthermore, given the wire width assignment of both the SST and a path P originated from the root of an LST, the optimal wire width assignment for each subtree branching off from P can be carried out independently.*

The first part of Theorem 1 is the separability between LSTs. Thus, for the MSIT in Figure 2.1, the optimal wire widths for LST_1,LST_2 and LST_3 can be computed independently if the wire widths for the SST are given. While, the second part of Theorem 1 is the separability within an LST, which is the counterpart of the separability in the SSWS problem since an LST can be viewed as an SSIT with its driver located at the branching node from the SST. Because the separability may not hold within the SST, the $MSWS$ problem has much higher complexity than the $SSWS$ problem in general.

B. LST Monotone Property

Theorem 2 *For an MSIT, there exists an optimal wiresizing solution $\mathcal{W}^*$ where the wire widths decrease monotonically rightward within each LST in the MSIT.*

Again, with respect to the analogy between an LST and an $SSIT$, and replacing the left-right relation in the LST with the ancestor-descendent relation

in an $SSIT$, the LST monotone property just like the monotone property for the SSWS problem. Because the optimal wiresizing algorithm OWSA developed in [29] for the SSWS problem is based on the separability and the monotone property, according to Theorems 1 and 2, it can be applied independently to each LST when the wire width assignments for the SST is given. Since OWSA is a polynomial-time algorithm, the optimal wire widths for the entire MSIT will be computed in the polynomial-time with respect to the given wire widths for the SST.

Furthermore, it is worthwhile to emphasize that the monotone property for the MSWS problem just holds within an LST. The root uni-segment in an LST may be wider than the uni-segment from which the LST branches off. An optimal MSWS solution based on the parameter for the second metal layer (M2) given in Table 2.4 is shown in Figure 2.2. The total wire length is $600\mu m$. In the optimal solution, the wire width assignment is *monotone* within the LST, however, the root uni-segment of the LST is wider than uni-segments in the SST. This example also shows that the monotone property like that in the *SSWS* problem does not hold for any particular source in an MSIT.

C. SST Local Monotone Property

Although the signal direction is changeable in the segments of the SST when different sources are active, surprisingly, our study shows that optimal $MSWS$ solutions still satisfy a local monotone property (Theorem 3) given after Lemma 1.

Lemma 1 *Given an MSIT and a segment S in the MSIT, for any uni-segments E_1 and E_2 ($E_1 \neq E_2$) within segment S, $F(E_1, E_2)$ defined in Eqn. (2.10) is an invariant (denoted $F_l(S)$) if E_1 is left to E_2, and $F(E_1, E_2)$ is another invariant*

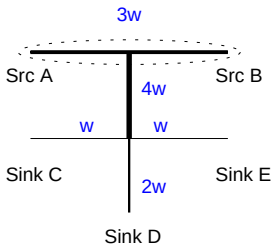


Figure 2.2: The optimal wire width assignments for a two-source net with W being the minimum wire width. The SST is surrounded by the dashed curve. Segments outside the curve belong to an LST. The wire width assignment is *monotone* within the LST. However, the root uni-segment of the LST is wider than uni-segments in the SST.

(denoted $F_r(S)$) if E_1 is right to E_2.

Theorem 3 *There exists an optimal wiresizing solution for an MSIT, such that the wire widths within each segments is monotone: (1) if $F_l(S) > F_r(S)$, the wire widths within S decrease monotonically rightward. (2) if $F_l(S) = F_r(S)$, the wire width within S does not change. (3) if $F_l(S) < F_r(S)$, the wire widths within S increase monotonically rightward.*

Of course, the local monotone property holds for segments in LSTs, where the $F_l(S)$ is always greater than $F_r(S)$ (in fact, $F_r(S) = 0$) and the wire widths always decrease rightward, just as given by the LST *monotone* property in an even stronger sense.

D. Dominance Property

Definition 1 *Given two wiresizing solutions $\mathcal{W}$ and $\mathcal{W}'$, we define $\mathcal{W}$ dominates $\mathcal{W}'$ if $w_E \geq w'_E$ for every uni-segment E.*

Definition 2 *Given a wiresizing solution $\mathcal{W}$ for the routing tree, and any particular uni-segment E in the tree, a local refinement on E is defined to be the operation to minimize the objective function Eqn. (2.7) by changing only the wire width of E while keeping wire width assignment of $\mathcal{W}$ on other uni-segments unchanged.*

Theorem 4 *Suppose that $\mathcal{W}^*$ is an optimal wiresizing solution for an MSIT. If a wiresizing solution $\mathcal{W}$ dominates $\mathcal{W}^*$, then the wiresizing solution obtained by any local refinement of $\mathcal{W}$ still dominates $\mathcal{W}^*$. Similarly, if $\mathcal{W}$ is dominated by $\mathcal{W}^*$, then the wiresizing solution obtained by any local refinement of $\mathcal{W}$ is still dominated by $\mathcal{W}^*$.*

Although the dominance property was proven based on the ancestor-descendant relation in [29] for the SSWS problem, we proved that it not only holds for the MSWS problem, but also independent of the ancestor-descendent relation in the SSWS problem, or the left-right relation in the MSWS problem. Theorem 4 enables efficient computations of lower and upper bounds of the optimal wiresizing solution for the MSWS problem by the greedy wiresizing algorithm GWSA [29] originally developed for the SSWS problem. It applies the local refinement operation iteratively to every uni-segment to compute the lower or the upper bound of the optimal wiresizing solution. A much more powerful refinement operation, called the bundled refinement operation, which may compute the lower or the upper bound for a number of uni-segments in a single operation, will be introduced in Section 2.3.2.

2.2.3 Extensions to Multi-layer Layout

Up to now, all properties are discussed under the assumption that all wires lay in the same routing layer. In the real layout designs, interconnects are often routed using more than one layer. Similar to the extension made for the SSWS problem in [29], the MSWS formulation can be extended to the multi-layer cases. In the multi-layer formulation, the LST separability and the dominance property still hold. The LST monotone property holds within each layer, i.e., there always exist an optimal wiresizing solution such that the wire widths decrease monotonically rightward within each layer for each LST. Furthermore, even in the same layer, if the allowable minimum and maximum wire widths are different from segment to segment due to obstacles in the routing area or reliability considerations, the LST monotone property holds only within segments in the same layer such that these segments have uniform allowable minimum and maximum wire widths. Moreover, it is reasonable to assume that each segment always stays in the same layer and its allowable minimum and maximum wire widths remains unchanged within the segment. In this case, the local monotone property always holds. Note that all discussions and the bundled refinement property to be presented in Section 2.3, same as the dominance property, hold for any layer assignment and any allowable minimum or maximum wire width.

2.3 Properties of Optimal MSWS/E Solutions

Up to now, both the *MSWS* problem defined here and the *SSWS* problem studied in [29, 73, 56, 55, 49, 82, 83] are only studied in the context of an *a priori* fixed segment-division. Intuitively, a finer segment-division may lead to better wiresizing solution. However, it is difficult to choose a proper segment-division.

For the best accuracy, a very fine, often uniform segment-division needs to be chosen, which results in the high memory usage and computation time due to the large number of uni-segments. We now investigate methods to obtain the optimal wiresizing results using a non-uniform and coarser segment-division. A novel contribution of our work is to introduce an *MSWS* formulation based on a *variable* segment-division. The segment-division might be finer in some regions but coarser in others. Moreover, we begin with a coarser segment-division then proceed to a finer one. Theorem 5 to be presented in Section 2.3.2 justifies this strategy and leads to much more efficient algorithms with the same accuracy when compared with previous works. All properties in this section hold for both the *MSWS* problem and the *SSWS* problem, but we shall concentrate on the *MSWS* problem since the *SSWS* problem can be treated as a special case.

2.3.1 Segment-Division and Bundled-Segment

We assume that $minLength$ is a constant determined by the user or the technology such that the wire widths are allowed to change every $minLength$ long, in other words, $minLength$ is the minimum length that a uni-segment can be. Given an MSIT, let $\mathcal{E}_0$ be the segment-division where each uni-segment is a segment in the MSIT, and $\mathcal{E}_F$ the uniform segment-division where each uni-segment is $minLength$ long.[3] Given two segment-divisions $\mathcal{E}$ and $\mathcal{E}'$, if each uni-segment in $\mathcal{E}$ corresponds to a single or multiple uni-segments in $\mathcal{E}'$, we say that $\mathcal{E}'$ is a refinement of $\mathcal{E}$. An segment-division $\mathcal{E}$ is *valid* only if $\mathcal{E}$ is a refinement of $\mathcal{E}_0$ and the length of every uni-segment is a multiple of $minLength$. Clearly, among all valid segment-divisions, $\mathcal{E}_0$ is coarsest and $\mathcal{E}_F$ is finest.

With these definitions, the *variable segment-division multi-source wiresizing*

[3]For the simplicity of presentation, we assume that the length of any segment in an MSIT is a multiple of $minLength$.

(MSWS/E) problem, can be formulated as follows:

Formulation 2 *Given an MSIT, the minimum uni-segment length minLength, and a set of possible wire width choices, the MSWS/E problem for delay minimization is to determine both an segment-division $\mathcal{E}$ and a wiresizing solution $\mathcal{W}$, such that the weighted delay $t(MSIT, \mathcal{E}, \mathcal{W})$ is minimized.*

Definition 1 will be extended to consider the variable segment-division cases.

Definition 3 *Given two wiresizing solutions $\mathcal{W}$ and $\mathcal{W}'$, we define $\mathcal{W}'$ dominates $\mathcal{W}$ if $w'_E \geq w_E$ for every uni-segment E under the finest segment-division $\mathcal{E}_F$.*

The concept of *bundled-segment* will be defined in order to achieve a segment-division as coarse as possible without the loss of wiresizing accuracy.

Definition 4 *Given an MSIT, a segment S and the finest segment-division $\mathcal{E}_F$, let $E_1, \cdots, E_p$ be a maximal sequence of successive uni-segments in S under $\mathcal{E}_F$ such that all uni-segments in this sequence have the same wire width in the optimal wiresizing solution under $\mathcal{E}_F$, We say that these uni-segments in the sequence form a bundled-segment.*

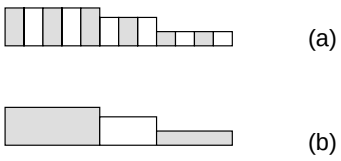


Figure 2.3: (a) The optimal wiresizing solution for segment S with twelve uni-segments under the finest segment-division $\mathcal{E}_F$. (b) Segment S contains only three bundled-segments which define a coarser segment-division with fewer computation costs to achieve the wiresizing solution same as that obtained by $\mathcal{E}_F$.

Figure 2.3 illustrates the concept of the bundled-segment by showing the optimal wiresizing solution for segment S in an MSIT. It has twelve uni-segments under the finest segment-division $\mathcal{E}_F$ (Figure 2.3.a), but just three bundled-segments (Figure 2.3.b). Clearly, the segment-division defined by the bundled-segments can achieve the wiresizing solution same as that obtained by the finest segment-division $\mathcal{E}_F$. For a long segment or a small $minLength$ used in order to achieve a better wiresizing solution, the number of uni-segments under the finest segment-division tends to be quite large while the number of bundled-segments in the segment is always bounded by a really small constant, as given by the following corollary of the local monotone property (Theorem 3).

Corollary 1 *Each segment in an MSIT has at most r bundled-segments where r is the number of possible wire width choices.*

Obviously, using the segment-division defined by the bundled-segments can achieve the required wiresizing solution for the lowest costs. An operation which leads to the computation of the optimal width for a bundled-segment directly, instead of treating it as a sequence of uni-segments under the finest segment-division $\mathcal{E}_F$, will be presented in the next subsection.

2.3.2 Bundled Refinement Property

Let $\mathcal{W}$ be a wiresizing solution which dominates the optimal solution $\mathcal{W}^*$, and E be a uni-segment under the current segment-division $\mathcal{E}$ and in segment S. Without loss of generality, we assume $F_l(S) \geq F_r(S)$ and treat E as two uni-segments E_l and $\overline{E_l}$ during the bundled refinement operation. E_l is the leftmost part of E, with length $minLength$ (recall $minLength$ is the length for a uni-segment in the finest segment-division $\mathcal{E}_F$); $\overline{E_l}$ is the remaining part of E. Let

$\tilde{w}_{E_l}$ be the locally optimized width for E_l based on the objective function Eqn. (2.7) while keeping the width assignment of $\mathcal{W}$ on $\overline{E_l}$ and any uni-segment E' other than E. Then, $\tilde{w}_{E_l}$ is regarded as a refined upper bound of the *entire* uni-segment E (not only E_l). This operation is called a *bundled refinement operation for the upper bound (BRU).*

The rational for the BRU operation is as follows: if $F_l(S) \geq F_r(S)$, in the optimal solution $\mathcal{W}^*$, E_l is always wider than all uni-segments under $\mathcal{E}_F$ in $\overline{E_l}$ (according to the local monotone property). The refinement of an upper bound of $w^*_{E_l}$ is still an upper bound of it (according to the dominance property), thus also gives an (possibly refined) upper bound of the optimal wire width assignments for any uni-segment under $\mathcal{E}_F$ in $\overline{E_l}$. Note that E will not be divided into E_l and $\overline{E_l}$ when performing the BRU operation on uni-segments other than E.

Similarly, the *bundled refinement operation for the lower bound (BRL)* can be defined for a wiresizing solution $\mathcal{W}$ dominated by $\mathcal{W}^*$. Again, assuming $F_l(S) \geq F_r(S)$, we treat E as two uni-segments E_r and $\overline{E_r}$. E_r is the rightmost part of E, with length $minLength$; $\overline{E_r}$ is the remaining part of E. Let $\tilde{w}_{E_r}$ be the locally optimized width for E_r based on the objective function Eqn. (2.7) while keeping the assignment of $\mathcal{W}$ on $\overline{E_r}$ and any uni-segment E' other than E. Then, $\tilde{w}_{E_r}$ is regarded as a refined lower bound of the *entire* uni-segment E.

Concerning the bundled refinement operation, the bundled refinement property similar to the dominance property for the local refinement operation will be given as Theorem 5, which leads to the bundled wiresizing algorithm to be presented in Section 2.4.1.

Theorem 5 *Let $\mathcal{W}^*$ be an optimal wiresizing solution under $\mathcal{E}_F$. If a wiresizing solution $\mathcal{W}$ dominates $\mathcal{W}^*$, then the wiresizing solution obtained by any BRU operation on $\mathcal{W}$ under any segment-division $\mathcal{E}$ still dominates $\mathcal{W}^*$. Similarly,*

if $\mathcal{W}$ is dominated by $\mathcal{W}^*$, then the wiresizing solution obtained by any BRL operation on $\mathcal{W}$ under any segment-division $\mathcal{E}$ is still dominated by $\mathcal{W}^*$.

2.4 Optimal MSWS Algorithm

2.4.1 Bundled Wiresizing Algorithm

Based on the dominance property (Theorem 4), the greedy wiresizing algorithm GWSA [29] originally developed for the SSWS problem is applicable to the MSWS problem. Working on an *a priori* defined segment-division, GWSA can use local refinement operations to compute the lower or the upper bound of the optimal wiresizing solution starting with the minimum or the maximum wire width assignment, respectively. Based on the bundled refinement property, a new algorithm, *bundled wiresizing algorithm (BWSA)* (Table 2.3) is proposed to compute the lower and upper bounds of the optimal wiresizing solution for an MSIT. BWSA also starts with the minimum and maximum wire width assignments, but uses bundled refinement operations instead of local refinement operations, and a gradually refined segment-division rather than a fixed one. BWSA achieves the same optimal lower and upper bounds for much less computation costs when compared with GWSA.

A. Overview

Starting with the coarsest segment-division $\mathcal{E}_0$, we perform BRU and BRL iteratively through an *MSIT*. We assign the minimum width to all uni-segments (in this case, each uni-segment is a segment), then traverse *MSIT* and perform BRL operation on each uni-segment. This process is repeated until no improvement is achieved on any uni-segment in the last round of traversal. Because the minimum

wire width assignment is dominated by the optimal wiresizing solution, according to the bundled refinement property, the result wiresizing solution is still dominated by the optimal wiresizing solution and is a lower bound of it. Similarly, we assign the maximum width to all uni-segments and perform BRU operations, obtain an upper bound of the optimal wiresizing solution. This is the first *pass* of *BWSA*.

After each *pass*, we check the lower and upper bounds. If there is a gap between the lower and upper bounds for an uni-segment (which is called a *non-convergent* uni-segment) and it is still longer than the minimum uni-segment length $minLength$, we divide it into two uni-segments of the almost equal length (they may differ by $minLength$ in order to maintain a *valid* segment-division), and let each uni-segment inherit the lower and upper bounds from their parent. After the refinement of all non-convergent uni-segments, another *pass* to tighten the lower and upper bounds is carried out by performing bundled refinement operations under the refined segment-division. Note that the bundled refinement is only needed for uni-segments who are just refined, because only these uni-segments are not convergent.

This BWSA algorithm iterates through a number of passes until we either have the identical lower and upper bounds for all uni-segments under current segment-division (in this case we get an optimal wiresizing solution), or each non-convergent uni-segment is $minLength$ long. The pseudo-codes of the BWSA algorithm are given in Tables 2.1-2.3.

B. Optimality

In order to discuss the optimality of the lower and upper bounds obtained by the BWSA algorithm, we define the following $\mathcal{E}_F$-tight lower and upper bounds.

```
Function gBWSA_L/U(minLength, E, W_lower/W_upper)

    W ← W_lower/W_upper;
    do
        progress ← false;
        for each uni-segment E of E do
            w ← BRL(minLength, E, W_lower, E)
                or BRU(minLength, E, W_upper, E);
            if w ≠ W(E) then
                progress ← true;
                W(E) ← w;
            end if
        end for;
    while progress = true;
    return W;
end Function;
```

Table 2.1: gBWSA_L/U: Given the minimum uni-segment length $minLength$, an segment-division $\mathcal{E}$, a lower/upper bound $\mathcal{W}_{lower}/\mathcal{W}_{upper}$ of the optimal wiresizing solution, and a set of possible wire widths $\{W_1, W_2, \cdots, W_r\}$, compute a tight lower/upper bound using BRL or BRU.

Function SBSR($minLength, \mathcal{E}, \mathcal{W}_{lower}, \mathcal{W}_{upper}$)

$\mathcal{E}' \leftarrow \phi$;
for each uni-segment E of $\mathcal{E}$ **do**
if $\mathcal{W}_{lower}(E) \neq \mathcal{W}_{upper}(E)$, and E is longer than $minLength$
then divide E into two uni-segments, E' and E'',
with (nearly) equal lengths;
$\mathcal{W}_{lower}(E') = \mathcal{W}_{lower}(E'') = \mathcal{W}_{lower}(E)$;
$\mathcal{W}_{upper}(E') = \mathcal{W}_{upper}(E'') = \mathcal{W}_{upper}(E)$;
$\mathcal{E}' \leftarrow \mathcal{E}' + \{E', E''\}$;
else $\mathcal{E}' \leftarrow \mathcal{E}' + \{E\}$;
end if;
end for;
return $\mathcal{E}'$;
end Function;

Table 2.2: Selective Binary Segment-division Refinement (SBSR): Given the minimum uni-segment length $minLength$, an segment-division $\mathcal{E}$, a lower bound and an upper bound of the optimal wiresizing solution, return a refined segment-division.

Function BWSA($\mathcal{E}_0, minLength$)

$\mathcal{E}' \leftarrow \mathcal{E}_0$; $\quad \mathcal{W}_{lower} \leftarrow W_1$; $\quad \mathcal{W}_{upper} \leftarrow W_r$;

do

$\mathcal{E} \leftarrow \mathcal{E}'$;

$\mathcal{W}_{lower} \leftarrow$ gBWSA_L($minLength, \mathcal{E}, \mathcal{W}_{lower}$);

$\mathcal{W}_{upper} \leftarrow$ gBWSA_U($minLength, \mathcal{E}, \mathcal{W}_{upper}$);

$\mathcal{E}' \leftarrow SBSR(minLength, \mathcal{E}, \mathcal{W}_{lower}, \mathcal{W}_{upper})$;

while $\mathcal{E} \neq \mathcal{E}'$

return $\mathcal{W}_{lower}$ and $\mathcal{W}_{upper}$;

end Function;

Table 2.3: Bundled Wiresizing Algorithm (BWSA) : Given the coarsest segment-division $\mathcal{E}_0$, the minimum uni-segment length $minLength$ and a set of possible wire widths $\{W_1, W_2, \cdots, W_r\}$, return the $\mathcal{E}_F$-tight lower and upper bounds of the optimal wiresizing solution.

Definition 5 *If a wiresizing solution $\mathcal{W}$ dominates the optimal solution $\mathcal{W}^*$ and can not be further refined by any local refinement operation under the finest segment-division $\mathcal{E}_F$, $\mathcal{W}$ is an $\mathcal{E}_F$-tight upper bound. Similarly, $\mathcal{W}$ is an $\mathcal{E}_F$-tight lower bound if $\mathcal{W}$ is dominated by $\mathcal{W}^*$ and can not be further refined by any local refinement operation under $\mathcal{E}_F$.*

It is easy to find that the lower and upper bounds given by the GWSA algorithm are $\mathcal{E}_F$-tight. Besides, it is worthwhile to mention that there may be more than one $\mathcal{E}_F$-tight upper (or lower) bounds for an $\mathcal{W}^*$. An experimental example of non-unique $\mathcal{E}_F$-tight bounds will be given in Section 2.5.2.

With this definition, we proved the following important result concerning the optimality of the BWSA algorithm.

Theorem 6 *The lower and upper bounds provided by BWSA are $\mathcal{E}_F$-tight.*

Basically, Theorem 6 suggests that the quality of the wiresizing solutions obtained by the BWSA algorithm starting from the *coarsest* segment-division is as good as those obtained by the GWSA algorithm using the *finest* segment-division $\mathcal{E}_F$.

C. Complexity

Recall that our MSWS/E problem aims to find the optimal wiresizing solution for every wire which is $minLength$ long. In order to achieve the required accuracy, the finest segment-division $\mathcal{E}_F$ where each uni-segment is $minLength$ long must be used by GWSA, while BWSA can determine a proper, usually coarser, segment-division during the wiresizing procedure. If we use $minLength$ as the wire length unit, the total wire length n is a natural metric to measure the problem size. We proved the following Theorem 7.

Theorem 7 *Given an MSIT and r wire width choices, if the total wire length is n when regarding $minLength$ as the length unit, both GWSA and BWSA have the worst-case complexity of $O(n^3 \cdot r)$ for the MSWS/E problem.*

It is worthwhile to emphasize that the final uni-segment produced by BWSA is often much longer than $minLength$ and BWSA runs much faster than GWSA in the practice, which is supported by extensive experiments in Section 2.5.2 and [15]. In fact, because BWSA runs much faster than GWSA and obtains the lower and upper bounds same tight as those obtained by GWSA, we always use BWSA instead of GWSA. Furthermore, we like to mention that due to the fact that BWSA computes both lower and upper bounds of the optimal wiresizing solution based on the bundled refinement property, we can tell easily when the optimal wire widths are achieved for those uni-segment that their lower and upper bounds meet, so that we do not have to further refine the segment-division for them. Similar segment-division refinement scheme may not be used optimally in other wiresizing methods [73, 56, 55, 82, 83, 49] until there is an easy way to determine that the current wiresizing solution is the optimal wiresizing solution or partial of it belongs to the optimal wiresizing solution.

2.4.2 Optimal Wiresizing Algorithm Using Bundled Refinement

Given an MSIT, BWSA can be used to compute the $\mathcal{E}_F$-tight lower/upper bounds of the optimal wiresizing solution. If the lower and upper bounds meet, which is very likely in practice, we get the optimal wiresizing solution immediately. Otherwise, the optimal solution shall be found between the lower and upper bounds. Because of the LST separability and the LST monotone property, OWSA, originally developed for the SSWS problem in [29], can be used independently for every LST with respect to the given wire width assignments for the SST. How-

ever, since the separability in SST does not hold in general, the optimal wire width assignments for non-convergent uni-segments in the SST will be found by enumeration between the $\mathcal{E}_F$-tight lower and upper bounds and subject to the local monotone property. Thus, the optimal wiresizing algorithm using bundled refinement (*OWBR* algorithm) has been developed, which works as the following:

1. Compute the $\mathcal{E}_F$-tight lower and upper bounds by BWSA;
2. Enumerate the wire width assignments for the SST between the $\mathcal{E}_F$-tight lower and upper bounds and subject to the local monotone property;
3. Apply OWSA independently to each LST during the enumeration of wire width assignments for the SST and subject to the $\mathcal{E}_F$-tight lower and upper bounds.

Our experiments show that BWSA gives the convergent bounds on all uni-segments in an MSIT for almost all cases. For those cases which have non-convergent uni-segments, the percentage of non-convergent uni-segments is very small. Moreover, the gap between the lower and upper bounds on each non-convergent uni-segment is also very small (usually being one in our experiments). Therefore, OWBR runs very fast in practice. Note that the OWBR algorithm can be extended to the multi-layer case same as the extension of the OWSA algorithm in [29]. Experimental results with multi-layer MSIT designs will be presented in Section 2.5.

2.5 Experimental Results

We have implemented the OWBR algorithm in ANSI C for the Sun SPARC station environment and tested our algorithm on multi-source nets extracted

from the multi-layer layout of an Intel high-performance microprocessor. In this section, we shall present both the comparison of different wiresizing solutions, the comparison between the BWSA algorithm and the GWSA algorithm and the fidelity study of the Elmore delay model versus the SPICE-computed delay to justify our formulation based on the Elmore delay model.

The parameters used in our experiments are summarized in Table 2.4. These parameters are based on the $0.5\mu m$ CMOS technology in North Carolina Microelectronic Center (MCNC) [54]. Since only parameters about the first and the second metal layers (M1 and M2) are available, we only use layers M1 and M2 in our experiments. The wire width choices in each layer are $\{W, 2W, 3W, 4W, 5W\}$ with W being the minimum allowable wire width in the layer. Note that our algorithms are still valid if wire widths are not multiples of the minimum width. The minimum uni-segment length $minLength$ is set to be 10 μm. We assume that the driver is an inverter, its p-type transistor is $105.9\mu m$ wide and n-type $53.5\mu m$ wide. Its effective resistance is 156 Ω based on SPICE simulation. We model the driver as a resistor of this value during the wiresizing procedure. In addition, the loading capacitance in every loading is set to be 3.720 fF. Note that both our formulation and implementation can handle cases where different sources have different driver resistances and different sinks have different loading capacitances.

Metal Layer:	M1	M2
Wire Resistance ($\Omega/\Box$):	0.068	0.044
Wire Capacitance (area) ($aF/\mu m^2$):	130.6	41.3
Fringing Capacitance (2 sides) ($aF/\mu m$):	161.9	150

Table 2.4: Parameters based on MCNC $0.5\mu m$ submicron CMOS technology.

2.5.1 Comparison between Different Wiresizing Solutions

We will report SPICE-computed delays instead of calculated Elmore delay values in the comparison between different wiresizing solutions. For the SPICE simulation in this chapter, the driver is modeled by parameters given in [54] for SPICE Level-3 MOSFET model, and every wire of $minLength$ long (10 μm) by an RC circuit. The use of SPICE simulation results not only shows the quality of our MSWS solutions, but also verifies the validity of our interconnect modeling and the correctness of our MSWS problem formulation.

The test suite used for our algorithms comprises real multi-source nets provided by Intel [43]. These nets were extracted from the top-level floor-plan of a high-performance microprocessor. Most pins of these nets can serve as both sources and sinks at different times, and almost all pairs between sources and sinks (excluding feed-through pins) are timing critical. We use 1-Steiner tree algorithm [46] to route these nets. Table 2.5 summarizes the routing trees for these nets.

	total pin number	total segment number	total wire length (μm)
net1	3	4	3600
net2	4	6	6600
net3	9	13	10070
net4	4	5	10570
net5	11	10	16980
net6	19	19	31980

Table 2.5: Routing trees for multi-source nets extracted from the layout for a high-performance Intel microprocessor

We applied our OWBR algorithm to these MSITs. First, we assume that all wires in M2; then, we assume that all wires parallel to X-axis in M1, the rest in M2. Let min_width be the wiresizing solution with minimum wire width W everywhere, and opt_msws the multi-source wiresizing solution given by our OWBR algorithm. Also, let *wire length* denote the total wire length of a routing tree, and *normalized area* denote the area ratio of wiresizing solution versus the min_width wiresizing solution, which is equivalent to the average wire width if the minimum wire width W is scaled to 1. Both *average delay* and *maximum delay* are only in terms of critical source-sink pairs. In these experiments, we assign $\lambda^{ij} = 1$ for a critical source-sink pair and $\lambda^{ij} = 0$ otherwise. Thus, the objective in Eqn. (2.7) is equivalent to the *average delay* among critical source-sink pairs. Comparisons between different wiresizing solutions are shown in Tables 2.6 and 2.7. In terms of the average delay, which is the objective of our MSWS formulation, the opt_msws solutions consistently outperform the min_width solutions. The delay reduction is up to 23.5% and 12.6% for the single-layer case and the multi-layer case, respectively. It is interesting to observe that although the *average delay* is our objective, experimental results show that this formulation reduces the maximal delay substantially (only in one example, opt_msws loses 0.69% in terms of the maximum delay, but still wins in terms of the average delay). The maximum delay reduction is up to 36.3% and 37.8% for the single-layer case and the multi-layer case, respectively. Besides, the delay reduction for nets with larger span is observed to be more significant. It often happens in our experiments that the optimal wiresizing solution for nets with fewer pins and shorter total wire lengths is simply the minimum wire width solution. In general, the optimal wiresizing is more effective for global nets with more pins and longer total wire lengths.

	Normalized Area	Average Delay (ns)		Maximum Delay (ns)	
	opt_msws	min_width	opt_msws	min_width	opt_msws
net1	1.000	0.1198	0.1198(0.0%)	0.1224	0.1224 (0.0%)
net2	1.044	0.2004	0.1994(-0.50%)	0.2572	0.2567 (-0.2%)
net3	1.475	0.3504	0.3241(-7.5%)	0.5230	0.4025 (-23.0%)
net4	2.000	0.5007	0.3846(-23.2%)	0.5853	0.4873 (-16.7%)
net5	1.775	0.6375	0.5711(-10.4%)	0.9496	0.7635 (-19.6%)
net6	2.706	1.8968	1.4512(-23.5%)	3.4505	2.1979 (-36.3%)

Table 2.6: Multi-source wiresizing results on several nets in an Intel microprocessor layout. All wires are assumed to be on layer M2.

	Normalized Area	Average Delay (ns)		Maximum Delay (ns)	
	opt_msws	min_width	opt_msws	min_width	opt_msws
net1	1.000	0.1198	0.1198(0.0%)	0.1224	0.1224 (0.0%)
net2	1.044	0.3250	0.3245(-0.15%)	0.3777	0.3803 (+0.69%)
net3	2.000	0.5336	0.4983(-6.61%)	0.8583	0.7853 (-8.51%)
net4	2.000	0.5514	0.5282(-4.54%)	0.8009	0.7000 (-12.6%)
net5	1.775	0.6445	0.5850(-9.23%)	0.9447	0.7623 (-19.3%)
net6	3.224	2.2752	1.9885(-12.6%)	4.3826	2.7238 (-37.8%)

Table 2.7: Multi-source wiresizing results on several nets in an Intel microprocessor layout. We assume that all wires parallel to X-axis are on layer M1, the rest on layer M2.

2.5.2 Speed-up Using Variable Segment-Division

We applied both BWSA and GWSA algorithms to the test suite of Intel nets. Because the time for BWSA to compute the $\mathcal{E}_F$-tight lower and upper bounds for most nets in the test suite is too small to measure, we compared the total running time. In Table 2.8, the *BWSA-based* algorithm is just OWBR, i.e., BWSA to compute $\mathcal{E}_F$-tight lower and upper bounds, followed by enumerating for the SST and OWSA for LSTs. The *GWSA-based* algorithm is just to replace BWSA by GWSA in the OWBR scheme. The BWSA-based algorithm is observed to run more than 100x faster than the GWSA-based algorithm.

It is worthwhile to mention that BWSA gives identical $\mathcal{E}_F$-tight lower and upper bounds for net3 while GWSA *does not*, which is also an example of existence of multiple $\mathcal{E}_F$-tight bounds for the optimal solution as mentioned in Section 2.4.1. Also note that, in case of OWBR, the total running time is *not* dominated by the time to compute $\mathcal{E}_F$-tight lower and upper bounds, one reason is that the current implementation builds the data structure for the finest segment-division even if the bundled refinement does not need it at all. Thus, the total running time still can be further reduced in future implementation without building the data structure for the finest segment-division[4].

2.5.3 Fidelity of the Elmore Delay Model

The concept of *fidelity* for the Elmore delay model was introduced by Boese *et al* in [4] for the routing tree topology optimization to measure if an optimal or near-optimal solution selected according to the Elmore delay model is nearly optimal according to the actual delay (e.g., computed using SPICE). We shall investigate the fidelity of the Elmore delay model for the optimal wiresizing problem, i.e., to

[4]It has been done in [18, 20].

find out how good the solution given by our optimal wiresizing formulation based on the Elmore delay model is in terms of the real delay.

We measure the *fidelity* again on the test suite of Intel nets and assume all wires in the M2 layer. Since the number of total wiresizing solutions is prohibitively large to enumerate, we randomly generate 1,000 wiresizing solutions for every MSIT. In a solution, a random wire width is assigned for every wire $minLength$ long ($10\mu m$). We obtain both the weighted average Elmore delay and the weighted average 50% delay computed by SPICE for each solution and then ranked the 1,000 solutions for each MSIT, using the technique similar to [4]: first rank solutions according to their weighted Elmore delays, then rank them according to their weighted SPICE-computed delay. The absolute difference between the two rankings of a wiresizing solution is its *ranking difference* and we average ranking difference over 1,000 solutions for every MSIT. In order to know how large the SPICE-computed delay difference may be with respect to the average ranking difference, *delay difference* is computed in the following way: Let the average ranking difference is d. For a wiresizing solution whose SPICE-computed delay ranking is i, we compute the relative difference between the $(i+d)$-th and i-th SPICE-computed delays, as well as that between the $(i-d)$-th and i-th SPICE-computed delays. Between the two values, the one with the larger absolute value is defined as the delay difference for the average ranking difference d.

The average ranking differences and the associated average delay differences are given in Table 2.9. Let's take net1 as an example to show how good the optimal solution selected according to the Elmore delay model might be. The average rank difference for 1,000 wiresizing solutions is 23.61. Thus, the optimal solution selected according to the Elmore delay, on average, might be $\lceil 23.61 \rceil$ away from

	net1	net2	net3	net4	net5	net6
GWSA-based algorithm (s)	0.07	8.18	172.37	15.67	38.10	227.92
BWSA-based algorithm (s)	0.07	0.15	0.37	0.37	0.97	3.37
Speedup factor	1	54.5	465.8	42.3	39.3	67.63

Table 2.8: Running time comparison between GWSA-based and BWSA-based algorithms

net	Overall (1,000)		Best-100 (Elmore Delay)		Best-10 (Elmore Delay)	
	Ranking Difference	Delay Difference	Ranking Difference	Delay Difference	Ranking Difference	Delay Difference
net1	23.61	0.1048%	11.36	0.0150%	2.800	0.0017%
net2	121.7	0.7223%	69.81	0.1007%	23.10	0.0173%
net3	53.50	0.3264%	33.31	0.1300%	15.40	0.0508%
net4	170.7	0.8517%	54.54	0.0410%	1.400	0.0490%
net5	38.52	0.1462%	14.52	0.0580%	0.900	0.0012%
net6	54.27	0.1812%	25.45	0.0286%	2.000	0.0026%

Table 2.9: Average differences in ranking and SPICE-computed delay for Intel nets based on $0.5\mu m$ CMOS technology

the top one in the ranking according to SPICE-computed delays. Since the ranking difference of [23.61] accounts for only 0.1448% SPICE-computed delay difference, on average, the optimal solution selected according to Elmore delay is only 0.1448% worse than the optimal one selected according to the SPICE-computed delays, when delays of both solutions are measured by SPICE simulation. [5]

Over the 1,000 random solutions for each net, the average ranking differences are between 23.61 and 170.7, and average delay differences between 0.1648% and 0.8517%. In addition, we measure the average delay difference for the best-100 and best-10 wiresizing solutions according to the Elmore delay model for each random solution set, respectively. It is interesting to find that the better the wiresizing solutions according to the Elmore delay model, the less the average delay difference they have. Taking net1 as an example, the average delay difference is 0.1048% for the 1,000 solutions, but only 0.0150% for the best-100, and even less, 0.0017% for the best-10. It implies that, in general, in the area near the optimum in the solution space, the Elmore delay model has a even higher fidelity.

Based on data of the best-10 in every random solution set, the optimal wiresizing solution selected according to the Elmore delay model is less than 0.06% worse than the optimal solution selected according to the SPICE-computed delay.[6] Thus, we believe that the Elmore delay model has really high fidelity for wiresizing optimization, i.e., the optimal solution selected according to the Elmore delay model is also the optimal solution or nearly the optimal solution selected

[5] Note that the Elmore delay value of the optimal solution selected according to the Elmore delay model is often quite different from the SPICE-computed delay of the same solution, with 24% error for the optimal wiresizing solution for net1.

[6] We also enumerate the wiresizing solutions for net1 and net2 by assuming that each segment in the routing tree has a uniform wire width. Even higher fidelity is observed when compared with this set of random wiresizing experiments. For these two nets, the Elmore delay model gives the best-5 solutions same as those given by the SPICE-computed delay.

according to the SPICE-computed delay. Note that the inductance is not taken into consideration in our SPICE simulation, since the inductive effect is negligible under the current CMOS technology. The higher-order delay model used in [56, 55] does not consider the inductance, either.

2.6 Conclusions and Future Work

The results in chapter have shown convincingly that proper sizing of the wire segments in multi-source nets can lead to significant reduction in the interconnect delay. We have also developed an efficient wiresizing algorithm named BWSA algorithm. It achieves the wiresizing solution same as the GWSA algorithm (originally developed for the single-source wire sizing problem[29], and extended to the multi-source wire sizing problem in this work), but runs 100x time faster and uses much less memory space. Compared to the minimum wire width solution, the optimal wiresizing solution obtained by our algorithm reduces the average delay by up to 23.5% and the maximum delay by up to 37.8%, respectively. It takes several seconds to obtain the optimal wiresizing solution for the largest example in our test suite extracted from a high-performance Intel microprocessor. In practice, the BWSA algorithm has been used for single-source wiresizing [28, 29], simultaneous driver and wire sizing [26], simultaneous transistor and interconnect sizing [20] and simultaneous buffer and wire sizing [27].

Simultaneous driver and wire sizing for multi-source nets has been solved by being posed as a CH-program [20], which will be presented in Chapter 4. Other recent works on multi-source net optimization include the following: in [39], an optimal shape function for a bi-directional wire is derived. A bi-directional wire is like a wire segment in our SST. Similar to Theorem 3, the wire shape is shown to be monotonic. In [48], an augmented RC-diameter (ARD) is proposed as a

performance measure for MSITs, and a buffer insertion algorithm is developed to minimize the ARD.

In addition to weighted delay minimization, another wiresizing optimization objective is to minimize the maximum delay in interconnects. If we assume the single-source net and the fixed-value resistor model for the driver, approaches in [72, 55, 6] are able to achieve the optimal continuous wiresizing solution, and the approach in [49] is able to achieve the optimal discrete wiresizing solution. It is worthwhile to mention that the approach in [6] is based on a Lagrangian relaxation procedure to iteratively apply the weighted delay minimizations (same as that in [28, 29] and similar to our formulation). It adjusts the weight assignments after each iteration until the optimal weight assignments are achieved to minimize the maximum delay by using the weighted delay minimization. In order to minimize the maximum delay for multi-source nets, the optimal continuous solution might be achieved by extensions of approaches in [72, 55, 6]. However, the optimal algorithm to obtain the discrete solution is still open. We have shown experimentally that our weighted delay formulation could reduce the maximum delay very well. It is worthwhile to find out whether an optimal algorithm exists.

The topologies of MSITs may affect the delay reductions that can be achieved by the optimal wiresizing even though our OWBR algorithm is able to achieve the optimal wiresizing solution for any MSIT topology. For single-source nets, simultaneous tree construction and wiresizing has been explored very recently in [50, 60]. Also, a min-cost min-diameter A-tree algorithm has been proposed [31] for multi-source tree construction. However, it is still open how to combine the routing tree construction and wiresizing to achieve the largest delay reduction for multi-source nets. It will be an interesting direction for the future work.

In this chapter, the *simple* model is used for the interconnect capacitance: it

is assumed that for a wire with width w and length l, its capacitance is given by $w \cdot c_a + c_f$ with c_a and c_f being constants. The simple capacitance model might not be true for the DSM designs where the coupling capacitance becomes more significant. We will present an accurate interconnect capacitance model in Chapter 3, and then extend our wire sizing algorithm to consider the coupling capacitance in Chapter 6.

CHAPTER 3

A Simple and Practical 2 1/2-D Capacitance Extraction Methodology

This chapter addresses interconnect capacitance extraction during interconnect optimization. Our contributions include:

- We show how basic drivers in process technology (planarization and minimum metal density requirements) actually simplify the extraction problem; we do this by proposing and validating five "foundations" through detailed experiments with a 3-D field solver on representative $0.50\mu m$, $0.35\mu m$ and $0.18\mu m$ process parameters.

- We present a simple yet accurate 2 1/2-D extraction methodology directly based on the foundations. This methodology has been productized and is being shipped with the Cadence Silicon Ensemble 5.0 product for the timing verification purpose. The methodology is also able to extract capacitance on fly during interconnect design and optimization. Moreover, it can be used for MCM/PCB designs.

We will use this capacitance extraction methodology, in other words, the 2 1/2-D capacitance model, to study the interconnect sizing and spacing problem in Chapter 6.

The remainder of this chapter is organized as follows. Section 3.1 presents the introduction to the interconnect capacitance extraction problem, and an overview of this chapter. Section 3.2 uses 3-D numerical extractions based on representative 0.5 μm, 0.35 μm and 0.18 μm processes to justify these foundations. The 2 1/2-D RC extraction methodology, which is based on these foundations and is used in Cadence Silicon Ensemble 5.0 product, is then described in Section 3.3; example structures are used to show the accuracy of the methodology. We conclude in Section 3.4 that the 2 1/2-D extraction method proposed in chapter is sufficient for delay calculation and estimation for performance-driver layout in deep submicron designs.

3.1 Introduction

In deep-submicron VLSI, complex 3-dimensional interconnect structures pose a difficult challenge for parasitic capacitance extraction. Many extraction approaches exist, including 1-D, 2-D and 2 1/2-D analytic models [70, 2, 9, 69, 71, 11, 35, 1, 81] as well as 2-D and 3-D field solvers [66, 79, 76, 59, 42, 58]. These techniques span a wide range of cost-accuracy regimes.

- 1-D analysis uses (per-unit length) total capacitance, equivalent to (per-unit area) area capacitance and (per-unit length) edge capacitance when some wire width is assumed.
- 2-D analysis uses (per-unit area) area capacitance and (per-unit length) lateral+fringing capacitance, where geometries on neighboring layers are at best probabilistically modeled.
- 2 1/2-D analysis (sometimes called "2-D, 3-Body" analysis) uses (per-unit area) area capacitance and (per-unit length) lateral and fringing capaci-

tances, where geometries on one or more neighboring layers are explicitly modeled but then lumped additively into "above" and "below" corrections to the coupling calculation.

- 3-D analysis uses numerical techniques (finite-element as in [66] and Ansoft's products, or finite-difference as in [79, 76] and Avant!'s Raphael) to solve Laplace's equation for potential distribution, then applies Gauss's theorem to yield charge distribution, finally applies $Q = [C]V$ to determine self- and mutual capacitances of all conductors. Boundary-element numerical techniques [59] can exploit the flatness and rectilinearity of IC geometries. In addition, multipole algorithms [58] has been recently reported.

However, an effective design flow requires a *principled* matching of the parasitic extraction methodology to the actual requirements for accuracy and runtime.

Our discussion will focus on the difficult task of post-routing capacitance extraction during performance-driven layout design. Such an extraction must be accurate: correlation with "final" verification engines is needed for design convergence. Such an extraction must also be fast: it may be performed dozens of times on full-chip layout, and thousands of times on critical signal nets, during the iterative loop (placement, routing, extraction, delay calculation, timing/noise analysis) of layout design. Simple 1- and 2-D extraction may not suffice in deep-submicron design. First, wire aspect ratios (thickness divided by width) are becoming larger for more advanced processes, so that lateral and fringe couplings become much more significant (where C_f is the fringe coupling in this chapter, and C_x is the lateral coupling). Second, increased packing densities, lower supply voltages, and use of dynamic logic for performance optimization all lead to much tighter signal integrity margins, so previously second-order details such as

crossover and crossunder couplings must be modeled. At the same time, full 3-D numerical extraction is difficult to support during layout due to its time complexity. For these reasons, the 2 1/2-D approach has been well-studied in recent years [11, 3, 1].

Our first contribution lies in showing how basic drivers in process technology (planarization and minimum metal density requirements) are actually making the parasitic capacitance extraction problem easier, enabling a simple yet accurate 2 1/2-D methodology. Our second contribution is a simple yet accurate 2 1/2-D extraction methodology [84] that has recently been developed and validated in cooperation with major ASIC suppliers; this methodology has been productized and is being shipped with the first release of the Cadence Silicon Ensemble (SE) 5.0 product.

Using a numerical 3-D field solver and leading-edge technology parameters, we first establish the following "foundations".

1. Ground, and neighboring wires in the same layer have significant shielding effects. Thus, both must be considered for accurate modeling.

2. Coupling between wires on layer $i-1$ and wires on layers $i+1$ is negligible when the metal density on layer i exceeds a certain threshold.

3. During capacitance extraction for wires on layer i, layers $i \pm 2$ can be treated as ground planes with negligible error. There is no need to look beyond layers $i \pm 2$.

4. Coupling analysis to wires in the same layer need only consider nearest neighbors independently, with the widths of same-layer neighbor wires having negligible effect on the coupling.

5. The interaction of layer $i-1$ in conjunction with layer $i+1$ on layer i is negligible; therefore, corrections for orthogonal crossovers and crossunders can be performed independently.

These foundations imply that for a given victim wire on layer i, wires on layers $i-2$ and $i+2$ can be treated as ground planes (with farther layers being ignored), while nearest neighbor wires on layer i, as well as all crossover and crossunder wires on layers $i-1$ and $i+1$, must be considered.

The above foundations justify a simplified 2 1/2-D extraction methodology, and can be used to guide the development of analytical formulas for parasitic capacitance. The Cadence Silicon Ensemble 5.0 product contains just such a simplified methodology, which has been developed and validated in cooperation with Motorola and other ASIC suppliers. The methodology entails one-time use of a 3-D field solver to determine the capacitance matrices for multiple predetermined "patterns" (i.e., multi-layer interconnect structures) that vary in widths and spacings of the victim on layer i, and widths and spacings of wires on crossover/crossunder layers $i-1$ and $i+1$. Linearity assumptions allow simple derivation of coefficients that capture per-unit length area, lateral and fringe components of the parasitic capacitance, along with corrections for crossovers and crossunders. In the actual post-routing extraction, a victim net's same-layer neighbors, crossovers, and crossunders are extracted from the geometric layout database, and total capacitance of the victim net is determined by simple pattern-matching and linear interpolation. Experiences of Cadence industrial partners show that the methodology produces satisfactory results.

3.2 Foundations

3.2.1 Preliminaries

A multilayer VLSI process has metal interconnects, or wires, on layers $1, 2, \ldots, k$ (i.e., M1, M2, ..., Mk). Currently, there are $k = 6$ layers in leading-edge processes, but $k = 8$ or more will be seen by the turn of the century. We call the multilayer geometric structure of wires a *pattern*. We assume that wires in adjacent layers are orthogonal, which is often true in layout designs. We use geometric parameters of maximum-density local interconnects in $0.50\mu m$, $0.35\mu m$ and emerging $0.18\mu m$ processes (see Table 22 in NTRS'94 [77]), and normalize all dimensions with respect to the minimum wire width in a given interconnect structure. The following *normalized* dimensions are used: For $0.50\mu m$ processes, wire width = 1.0, wire thickness = 1.0, and dielectric height between adjacent layers = 1.5(see Figure 3.1); for $0.35\mu m$ processes, wire width = 1.0, wire thickness = 1.5, and dielectric height between adjacent layers = 1.5; for $0.18\mu m$ processes, wire width = 1.0, wire thickness = 2.5, and dielectric height between adjacent layers = 3.0.[1]

[1]Let $AR = thickness/width$ denote the aspect ratio for a wire. We see that higher AR values are achieved in more advanced processes, and are deployed for better wiring density. According to Table 22 in NTRS'94 [77], the expected value of AR is 1.5 for $0.35\mu m$ processes. For such processes, the expected ratio between the wire thickness and the dielectric thicknesses is $0.6\mu m : 1.0\mu m = 1 : 1.67$. Compared with the $AR = 1.5$ case in our experiment using the normalized wire thickness 1.5 and the normalized dielectric thickness 1.5, the Roadmap's $0.35\mu m$ process has a *relatively* thicker dielectric, perhaps to provide better isolation. Therefore, our foundations concerning the coupling between different layers can be safely extended to the Roadmap's $0.35\mu m$ process. Furthermore, AR according to the Roadmap is generally interpreted with respect to the minimum wire width. In real designs, smaller AR values often apply due to non-minimum wire width on layers above M1 and possibly even including M1; this again implies relatively less coupling between sidewalls and allows our foundations to be safely extended.

Note that in more recent NTRS'97 [78], the AR becomes smaller (see Table 1.1): for example, $AR = 2.5$ for $0.18\mu m$ process in NTRS'94, but it becomes 1.8 for the same generation of process in NTRS'97. Nevertheless, the trend holds for both that the more advanced the process, the higher the value of AR. In experiments of this chapter, we cover the range of AR from 1.0 to 2.5, which implies that the conclusion in this chapter can be safely extended for processes from $0.25\mu m$ process to $0.10\mu m$ process (with AR from 1.8 to 2.4) in NTRS'97.

Let s be the edge-to-edge spacing between a wire (the *victim*) and its same-layer neighboring wires (*neighbors*). We typically study the "extreme" cases of $s = 1.0$ and $s = \infty$, with the latter meaning that the neighbors are too far away to have significant coupling to the victim.

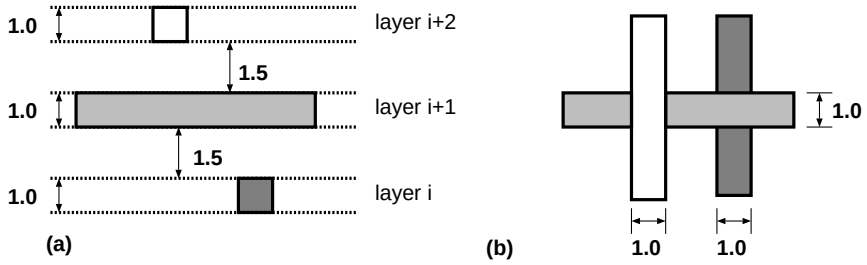


Figure 3.1: For the maximum-density local interconnect structure in $0.50\mu m$ process, we assume (a) the cross-section view – the thickness is 1.0 for all wires, and the height is 1.5 for all dielectrics (inter-layer spacings); (b) the top view – all wire have width 1.0.

We use an industrial-strength multipole-accelerated 3-D field solver, Fastcap[2] [58], to obtain coupling capacitances between multiple conductors in the form of a capacitance matrix. Since we run Fastcap on normalized patterns in free space, we obtain *normalized capacitances* as output. For example, if the minimum wire width in a pattern is $0.36\mu m$, a normalized capacitance of $100pF$ implies actual capacitance of

$$100(pF) \cdot 0.36(\mu m/m) \cdot \epsilon_r = 0.1404fF$$

where we assume that the relative permittivity of SiO_2 is $\epsilon_r = 3.9$. We will report only *normalized* values for all dimensions and capacitances (in units of pF), as only the ratios between different capacitance values are significant to our study.

[2]Fastcap is a public-domain program available by anonymous ftp from rle-vlsi.mit.edu. It is an element of several commercial products, e.g., from Quantic and Ansoft.

3.2.2 Coupling between wires on layer i and wires on layer $i-2$

Two experiments study coupling between wires on layers i and $i-2$, as well as effects of ground planes and same-layer neighboring wires. The pattern for the first experiment has one wire (*victim*) on layer i and one wire on layer $i-2$, but no wires on layer $i-1$ (see Figure 3.2). Let s_{center} be the horizontal distance between the *centers* of the two wires. We shift the wire on layer $i-2$ and observe the change of the ratio between $C_{i,i}/C_{i,i-2}$, where $C_{i,i}$ is the total capacitance for the victim, and $C_{i,i-2}$ the coupling between the two wires. We also study the two cases where layer $i-3$ is ground, and where there is no ground at all. Last, we consider two possible spacings ($s = 1.0, \infty$) for the two same-layer neighbors of the victim. All wires have length 20 and the ground is a 40×40 plane. Table 3.1 shows the following for $0.18\mu m$ process:

- The ground has a strong shielding effect on $C_{i,i-2}$. In the case of no neighbors and full overlap ($s_{center} = 0$), $C_{i,i-2}/C_{i,i} = 28.4\%$ when there is no ground versus 16.3% when there is a bottom ground plane.

- Neighboring wires also have a significant shielding effect on $C_{i,i-2}$. With two neighbors at $s = 1.0$, $s_{center} = 0$ and a bottom ground present, $C_{i,i-2}/C_{i,i} = 1.8\%$ versus 16.3% when there are no neighbors.

- The parallel-plate capacitance from overlap of the victim on layer i and the wire on layer $i-2$ is not the dominant component in the coupling. From full overlap ($s_{center} = 0$) to non-overlap ($s_{center} = 1$), relative changes in the coupling capacitance are less than 2%.

Similar trends can be observed for $0.50\mu m$ and $0.35\mu m$ processes.

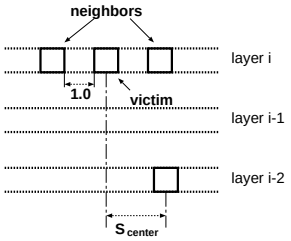


Figure 3.2: Cross-section of a pattern in the first experiment of Section 2.2.

In practice, there is always at least one ground (e.g., the substrate), and the likelihood of neighboring wires is high (see Footnote 3). Furthermore, it is unlikely that there are no wires on layer $i-1$. We study the effect of wires on layer $i-1$ using a pattern with one wire on layer i (the victim), one parallel and fully-overlapped wire on layer $i-2$ (similar to $s_{center}=0$ in the first experiment), and a number of wires (*crossunders*) on layer $i-1$ (see Figure 3.3). We vary the number of crossunders and observe the change in both $C_{i,i}$ and the ratio $C_{i,i}/C_{i,i-2}$.

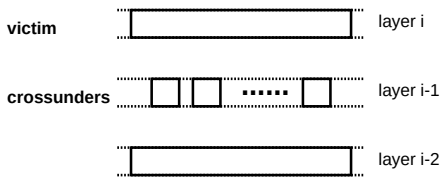


Figure 3.3: Cross-section for a pattern in the second experiment of Section 2.2.

Again, four possible combinations are studied: (i) layer $i-3$ is a bottom plane, or there is no ground; and (ii) the victim has two same-layer neighbors at spacing $s=1.0$ or $s=\infty$. All wires have length 20 and the ground is a 40×40 plane. Crossunders on layer $i-1$ are evenly distributed over the ground plane. The

$0.50\mu m, C_{i,i}/C_{i,i-2}$				
S_{center}	$s = \infty$		$s = 1.0$	
	with ground	without ground	with ground	without ground
0.0	459.6/103.5	424.3/170.9	880/48	874/71
1.0	458.1/101.6	422.5/168.2	897/47	873/68
4.0	499.0/71.0	403.6/139.0	877/26	867/48
10.0	441.4/25.0	374.9/88.0	876/7	865/27
$0.35\mu m, C_{i,i}/C_{i,i-2}$				
S_{center}	$s = \infty$		$s = 1.0$	
	with ground	without ground	with ground	without ground
0.0	483.8/106.4	449.2/174.0	1063/44.22	1063/65.36
1.0	483.0/104.1	446.9/171.4	1063/43.14	1061/64.08
4.0	475.0/78.9	432.0/148.4	1058/26.29	1056/49.11
10.0	466.0/33.2	403.7/98.3	1057/8.98	1053/28.40
$0.18\mu m, C_{i,i}/C_{i,i-2}$				
s_{center}	$s = \infty$		$s = 1.0$	
	with ground	w/o ground	with ground	w/o ground
0.0	486.6/79.49	458.4/130.1	1428/24.77	1424/37.96
1.0	486.5/78.78	451.9/127.4	1428/24.41	1424/37.63
4.0	484.6/71.70	454.8/123.1	1428/21.03	1424/36.91
10.0	479.4/46.96	446.5/100.4	1427/12.30	1424/24.40

Table 3.1: $C_{i,i}/C_{i,i-2}$, where $C_{i,i}$ is total capacitance of the layer-i victim, and $C_{i,i-2}$ is its coupling to the wire on layer $i-2$.

$0.50\mu m, C_{i,i}/C_{i,i-2}$				
	$s = \infty$		$s = 1.0$	
	with ground	without ground	with ground	without ground
2x	484.8/95.98	457.7/165.7	881.9/50.03	880.0/75.03
4x	537.6/47.57	530.9/76.65	895.2/26.69	894.5/37.61
8x	622.2/6.066	621.9/10.5	924.4/4.59	924.9/6.51
12x	632.2/3.31	632.4/5.17	928.7/2.95	927.0/3.33
$0.35\mu m, C_{i,i}/C_{i,i-2}$				
	$s = \infty$		$s = 1.0$	
	with ground	without ground	with ground	without ground
2x	534.9/66.37	511.3/120.5	1072/30.9	1070/47.9
4x	579.9/43.9	575.0/68.4	1083/22.21	1083/30.94
8x	654.2/8.85	654.2/16.6	1106/4.392	1105/7.09
12x	685.4/1.22	681.1/5.42	1117/0.209	1117/1.17
$0.18\mu m, C_{i,i}/C_{i,i-2}$				
	$s = \infty$		$s = 1.0$	
	with ground	w/o ground	with ground	w/o ground
2x	534.5/48.45	521.5/82.3	1433/16.64	1427/25.25
4x	581.3/21.99	578.5/31.6	1437/9.185	1450/11.54
8x	622.2/3.47	622.5/6.86	1440/3.45	1457/2.67
12x	635.9/2.47	636.7/4.21	1443/2.43	1458/1.95

Table 3.2: $C_{i,i}/C_{i,i-2}$ values for the second experiment of Section 2.2.

capacitance values are given in Table 3.2, where 2x - 12x indicates from 2 to 12 crossunders on layer $i-1$. Note that 12x corresponds to 30% of the area on layer $(i-1)$ being occupied.[3] We observe that, in addition to the strong shielding effect on $C_{i,i-2}$ due to the ground or same-layer neighbors, more crossunders on layer $i-1$ imply less significant $C_{i,i-2}$. For example, in $0.18\mu m$ process, when there are twelve crossunders and a bottom ground, $C_{i,i-2}/C_{i,i} = 0.4\%$ with no neighbors on layer i, and 0.2% with two neighbors on layer i. Given the minimum area occupancy of metal layers in deep-submicron processes, the coupling between a wire on layer i and a wire on layer $i-2$ is not significant. We conclude the following from these two experiments:

Foundation 1 *Ground, and neighboring wires on the same layer, have significant shielding effects. Thus, both must be considered for accurate modeling.*

Foundation 2 *Coupling between wires in layer $i+1$ and wires on layers $i-1$ is negligible when the metal density on layer i exceeds a certain threshold.*

These two foundations are further verified below. Foundation 2 implies that capacitance extraction can be simplified by treating layer $i-2$, and symmetrically layer $i+2$, as ground planes. We now verify this.

3.2.3 Coupling between wires on layers $i \pm 2$ and i

Three experiments show that layers $i \pm 2$ can be treated as ground planes for wires on layer i. In the first experiment, there is one *victim* wire of length 20

[3] In deep-submicron processes ($\leq 0.35\mu m$) the minimum area occupancy of a metal layer is typically set by the foundry to 30% for uniformity of etch rate or CMP planarization. Foundries may specify certain shapes of dummy metal which are small enough so as to not hold much charge during manufacturing (e.g., 2.5 μm by 2.5 μm). The key observation is that maximum possible occupancy is 50%, even with the line-to-line spacing = 1.0. Thus, conductor structure on adjacent orthogonal wiring layers is fairly predictable.

on layer i and one *crossunder* of length 20 on layer $i-1$. The *real* pattern (see Figure 3.4) in practice has a number of wires on layer $i-2$, with layer $i-3$ acting as a ground plane (or the substrate is a bottom ground plane). We assume that wires on layer $i-2$ are 40 units long. We also solve a *model* pattern by treating layer $i-2$ as a 40×40 ground plane without looking beyond this layer (i.e., there is no ground plane on layer $i-3$). Our experiment varies the number (density) of wires on layer $i-2$ and compares $C_{i,i}$ and $C_{i,i-1}$ for different patterns. $C_{i,i}$ is again the total capacitance for the victim, and $C_{i,i-1}$ is the coupling between the victim and the crossunder. In Table 3.3, 2x - 16x indicates from 2 to 16 wires on layer $i-2$ for the real pattern; *GND* indicates the model pattern. To take $0.18\mu m$ as an example, compared with the model pattern, $C_{i,i}$ in cases 8x - 16x differs by less than 0.7% when the victim has no neighbors ($s = \infty$), and by less than 0.4% when it has two neighbors at spacing $s = 1.0$; the corresponding values are 6.2% and 7.5% for $C_{i,i-1}$.

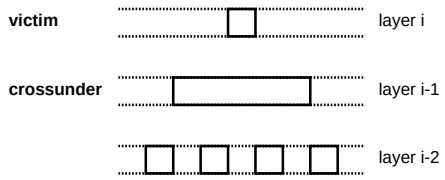


Figure 3.4: Cross-section of the real pattern in the first experiment of Section 2.3: the victim on layer i, one crossunder on layer $i-1$ and a number of wires on layer $i-2$. Layer $i-3$ is a ground plane, but is not shown in the figure.

Due to the minimum area occupancy requirement, a more likely scenario has a number of crossunders on layer $i-1$ instead of a single crossunder. Our second experiment in this section assumes that there is one *victim* on layer i, two same-layer neighbors of the victim at spacing $s = 1.0$ or $s = \infty$, and twelve crossunders $(u1, \cdots, u12)$ on layer $i-1$ (see Figure 3.5). All wires on layers i and

	$0.50\mu m, C_{i,i}/C_{i,i-1}$		$0.35\mu m, C_{i,i}/C_{i,i-1}$		$0.18\mu m, C_{i,i}/C_{i,i-1}$	
layer $i-2$	$s=\infty$	$s=1.0$	$s=\infty$	$s=1.0$	$s=\infty$	$s=1.0$
2x	482.2/77.6	883.4/35.4	495.8/102.1	1065/40.6	513.7/112.9	1431/31.93
4x	483.5/75.7	882.0/35.1	507.1/96.98	1066/39.33	519.5/112.0	1432/33.04
8x	491.7/74.4	885.4/34.2	551.7/96.68	1077/46.04	527.4/100.5	1430/27.64
12x	496.6/72.1	886.3/33.3	527.2/87.68	1072/38.42	529.5/99.17	1430/26.64
16x	498.1/71.0	886.3/33.0	529.7/86.15	1073/37.99	530.7/98.00	1433/27.83
GND	481.5/74.8	881.4/35.6	507.7/95.49	1068/38.99	531.0/95.46	1428/30.55

Table 3.3: $C_{i,i}/C_{i,i-1}$, where $C_{i,i}$ is the total capacitance of the victim, and $C_{i,i-1}$ the coupling between the victim and the crossunder.

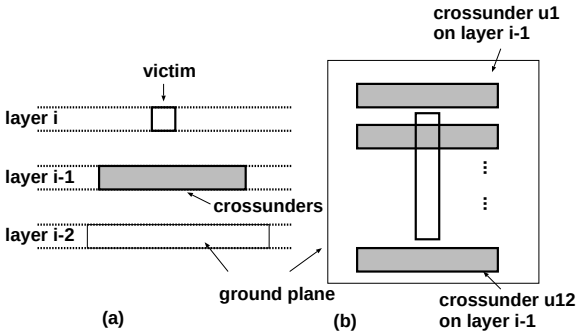


Figure 3.5: Layer $i-2$ is modeled as a ground plane: (a) the cross-section view; (b) the top view.

$i-1$ have length 20. The *real* pattern has twelve wires uniformly distributed in a 40×40 plane on layer $i-2$ and a 40×40 ground on layer $i-3$; the wires on layer $i-2$ have length 40. There are two model patterns: the *model1* pattern treats layer $i-2$ as a 40×40 ground, and the *model2* pattern treats layer $i-2$ as free space. We compare the total capacitance $C_{i,i}$ for the victim wire and the coupling $C_{i,u1}, \cdots, C_{i,u12}$ between the victim wire and crossunders $u1, \cdots, u12$. Tables 3.4-3.6 show that

(i) Both model1 and model2 can produce similar values for total capacitance $C_{i,i}$ and couplings $C_{i,l1}$ and $C_{i,r1}$ between the victim and its same-layer neighbors, when compared with the real pattern.

(ii) Model1 is better than model2 when used to solve the coupling between the victim and crossunders. For $0.18\mu m$ process, compared with the real pattern, the largest deviations for $C_{i,u1}$, $\cdots$, $C_{i,u12}$ are 10.7% when $s = \infty$, and 14.0% when $s = 1.0$ for model1; corresponding values for model2 are 45% and 72%. Furthermore, for couplings $C_{i,u3}$, $\cdots, C_{i,u10}$ between the victim and crossunders not at the boundary (see Figure 3.5), the deviation is at most 4.2% for model1. Since most crossunders in real designs are not at the boundary, the error introduced by model1 is negligible.

(iii) Even when same-layer neighbors have their strongest shielding at minimum spacing $s = 1.0$, the coupling mainly due to crossunders accounts for $1-(621.2+626.0)/1447 = 13.8\%$ of the total capacitance $C_{i,i}$ for $0.18\mu m$ process. More significant percentages are observed for representative $0.50\mu m$ and $0.35\mu m$ geometric parameters.. Therefore, coupling between the victim and crossunders must be considered in the capacitance extraction. We conclude that when there is no layer $i+1$ and beyond, layer $i-2$ can be viewed as a ground for computing total capacitances for wires on layer i, or coupling capacitances between wires on layer i and wires on layer $i-1$.

$0.50\mu m$	s	$C_{i,i}$	$C_{i,l1}$	$C_{i,r1}$	$C_{i,u1}$	$C_{i,u2}$	$C_{i,u3}$	$C_{i,u4}$	$C_{i,u5}$	$C_{i,u6}$	$C_{i,u7}$	$C_{i,u8}$	$C_{i,u9}$	$C_{i,u10}$	$C_{i,u11}$	$C_{i,u12}$
real	∞	618.6	-	-	7.987	11.82	33.42	65.14	71.9	73.16	73.13	71.8	63.67	29.24	9.774	5.236
model1		619.2	-	-	8.152	12.08	33.87	66 .1	73.28	74.56	74.52	73.07	64.52	29.79	9.912	5.497
model2		618.6	-	-	9.692	13.53	35.6	69.25	76.26	77.76	77.61	76.08	67.63	32.23	12.63	9.185
real	1.0	928.2	316.1	315.9	3.751	5.736	17.5	32.81	34.25	34.46	34.39	34.33	32.37	15.28	4.839	2.532
model1		927.9	316.6	313.6	3.935	5.959	17.72	33.06	34.58	34.9	34.91	34.69	32.69	15.57	4.994	2.767
model2		927.1	315.1	316.2	4.596	6.662	18.47	34.28	35.82	36.04	36.01	35.83	33.77	16.69	6.209	4.408

Table 3.4: Total capacitance $C_{i,i}$ of the victim on layer i and couplings between the victim and crossunders on layer $i-1$. $0.50\mu m$ technology is assumed in the experiment.

$0.35\mu m$	s	$C_{i,i}$	$C_{i,l1}$	$C_{i,r1}$	$C_{i,u1}$	$C_{i,u2}$	$C_{i,u3}$	$C_{i,u4}$	$C_{i,u5}$	$C_{i,u6}$	$C_{i,u7}$	$C_{i,u8}$	$C_{i,u9}$	$C_{i,u10}$	$C_{i,u11}$	$C_{i,u12}$
real	∞	669.8	-	-	9.63	13.78	37.21	70.38	77.71	78.78	78.75	77.53	68.61	32.56	11.64	6.519
model1		669.7	-	-	9.817	13.93	37.72	71.39	79.04	80.28	80.14	78.63	70.37	33.17	11.94	7.431
model2		669.7	-	-	11.7	16.36	39.96	73.94	81.26	82.36	82.17	80.68	71.64	34.91	14.83	10.59
real	1.0	1110	406.4	404.7	4.459	6.483	18.32	32.69	34.01	34.2	33.92	34.19	32.36	15.9	5.51	3.059
model1		1113	408.7	405.1	4.115	6.109	18.4	33.9	35.44	35.57	35.4	35.27	33.37	16.37	5.443	3.168
model2		1110	404.5	407.4	5.264	7.584	19.37	34.47	35.85	35.86	35.6	35.39	33.62	16.83	6.461	4.422

Table 3.5: Total capacitance $C_{i,i}$ of the victim on layer i and couplings between the victim and crossunders on layer $i-1$. $0.35\mu m$ technology is assumed in the experiment.

$0.18\mu m$	s	$C_{i,i}$	$C_{i,l1}$	$C_{i,r1}$	$C_{i,u1}$	$C_{i,u2}$	$C_{i,u3}$	$C_{i,u4}$	$C_{i,u5}$	$C_{i,u6}$	$C_{i,u7}$	$C_{i,u8}$	$C_{i,u9}$	$C_{i,u10}$	$C_{i,u11}$	$C_{i,u12}$
real	∞	629.5	-	-	16.4	19.89	36.89	50.95	59.62	61.09	60.59	61.47	51.39	32.92	16.78	12.28
model1		629.7	-	-	16.97	20.21	37.47	53.14	60.98	62.8	62.17	62.01	52.24	33.65	17.24	13.69
model2		630.8	-	-	18.56	21.24	37.45	56.19	63.3	65.61	65.58	62.85	54.74	35.73	20.21	17.94
real	1.0	1447	621.2	626	5.632	7.623	13.11	15.34	16.22	16.65	17.06	18.66	16.29	11.27	5.383	4.095
model1		1447	621	626.3	5.567	7.478	13.38	15.8	17.16	17.49	17.99	19.67	17.04	11.69	5.68	4.668
model2		1447	615.2	616.9	6.11	7.347	13.38	19.33	20.67	20.97	21.11	21.17	20.07	14.22	8.294	7.041

Table 3.6: Total capacitance $C_{i,i}$ of the victim on layer i and couplings between the victim and crossunders on layer $i-1$. $0.18\mu m$ technology is assumed in the experiment.

The third experiment of this section considers the impact of wires on layers $i+1$ and beyond. To have a geometric structure that still can be solved by Fastcap[4], we assume one victim on layer i with two same-layer *neighbors* at spacing $s = 1.0$ or ∞, six crossunders on layer $i-1$ and six crossovers on layer $i+1$. All wires on layer i have length 10, and all crossunders/crossovers have length 20. Crossunders/crossovers are uniformly distributed in 20×20 planes such that the area occupancy for layers $i \pm 1$ is 30%. In the *real* pattern, there are six wires distributed in a 20×20 plane on layer $i+2$ or $i-2$. These wires have length 20; layers $i \pm 2$ also have area occupancy of 30%. In the *model* pattern, layers $i \pm 2$ are 20×20 ground planes.

Tables 3.7-3.9 report total capacitance $C_{i,i}$ for the victim, same-layer couplings $C_{i,l1}$ and $C_{i,r1}$ between the victim and its neighbors, crossunder couplings $C_{i,u1}, \cdots, C_{i,u6}$ between the victim and crossunders $u1, \cdots, u6$, and crossover couplings $C_{i,o1}, \cdots, C_{i,o6}$ between the victim and crossovers $o1, \cdots, o6$. The difference between the model pattern and the real pattern is less than 0.5% for $C_{i,i}$, $C_{i,l1}$ and $C_{i,r1}$, and about 5% for crossunder or crossover coupling if the crossunder or crossover is not at the boundary in our pattern. We conclude:

Foundation 3 *During capacitance extraction for wires on layer i, layers $i \pm 2$ can be treated as ground planes with negligible error. There is no need to look beyond layers $i \pm 2$.*

[4]Shorter wires mean that the coupling due to the front and end sidewalls, as well as wire corners, is more significant. Our experiments use wires that are as long as possible, subject to Fastcap being run on a workstation with 400MB RAM.

	s	$C_{i,i}$	$C_{i,l1}$	$C_{i,r1}$	$C_{i,u1}$	$C_{i,u2}$	$C_{i,u3}$	$C_{i,u4}$	$C_{i,u5}$	$C_{i,u6}$	$C_{i,o1}$	$C_{i,o2}$	$C_{i,o3}$	$C_{i,o4}$	$C_{i,o5}$	$C_{i,o6}$
real	∞	399.0	-	-	3.74	30.4	58.8	58.7	30.3	3.70	3.69	30.8	58.2	58.8	30.8	3.65
model		398.9	-	-	3.63	31.3	59.4	59.4	31.25	3.50	3.68	31.5	59.9	59.9	31.5	3.69
real	1.0	512.8	142.8	142.2	2.32	18.2	32.7	32.6	18.2	2.24	2.29	18.4	32.1	32.2	18.4	2.24
model		512.4	142.2	142.0	2.25	18.7	32.7	32.7	18.6	2.20	2.27	18.8	33.3	33.3	18.8	2.27

Table 3.7: Total capacitance $C_{i,i}$ of the victim on layer i, couplings $C_{i,l1}$ and $C_{i,r1}$ between the victim and its same-layer neighbors, and couplings between the victim and crossunders on layer $i-1$ or crossovers on layer $i+1$. $0.50\mu m$ technology is used in this experiment.

	s	$C_{i,i}$	$C_{i,l1}$	$C_{i,r1}$	$C_{i,u1}$	$C_{i,u2}$	$C_{i,u3}$	$C_{i,u4}$	$C_{i,u5}$	$C_{i,u6}$	$C_{i,o1}$	$C_{i,o2}$	$C_{i,o3}$	$C_{i,o4}$	$C_{i,o5}$	$C_{i,o6}$
real	∞	365.0	-	-	5.16	28.7	52.0	52.0	28.5	5.15	5.27	28.7	52.1	51.7	28.6	5.19
model		365.2	-	-	5.32	29.0	52.3	52.2	28.9	5.25	5.38	29.3	52.5	52.6	29.3	5.32
real	1.0	496.3	151.5	151.2	2.95	16.0	26.6	26.6	15.8	2.91	3.05	15.9	26.7	26.6	15.9	2.96
model		496.7	151.8	151.6	3.02	16.1	26.6	26.6	16.0	2.94	3.07	16.2	26.9	27.0	16.3	3.01

Table 3.8: Total capacitance $C_{i,i}$ of the victim on layer i, couplings $C_{i,l1}$ and $C_{i,r1}$ between the victim and its same-layer neighbors, and couplings between the victim and crossunders on layer $i-1$ or crossovers on layer $i+1$. $0.35\mu m$ technology is used in this experiment.

	s	$C_{i,i}$	$C_{i,l1}$	$C_{i,r1}$	$C_{i,u1}$	$C_{i,u2}$	$C_{i,u3}$	$C_{i,u4}$	$C_{i,u5}$	$C_{i,u6}$	$C_{i,o1}$	$C_{i,o2}$	$C_{i,o3}$	$C_{i,o4}$	$C_{i,o5}$	$C_{i,o6}$
real	∞	418.9	-	-	13.51	33.77	52.18	52.26	33.95	13.67	13.93	34.09	52.43	52.35	34.31	14.17
model		418.9	-	-	14.16	34.68	52.54	52.53	34.39	14.26	14.08	34.52	52.67	52.59	34.31	14.15
real	1.0	779.9	310.4	310.4	5.099	11.73	16.09	16.09	11.85	5.03	6.042	13.95	19.46	19.32	13.86	6.135
model		781.6	309.5	308.9	6.663	12.66	17.46	17.34	12.58	6.613	6.679	13.71	18.47	18.31	13.6	6.706

Table 3.9: Total capacitance $C_{i,i}$ of the victim on layer i, couplings $C_{i,l1}$ and $C_{i,r1}$ between the victim and its same-layer neighbors, and couplings between the victim and crossunders on layer $i-1$ or crossovers on layer $i+1$. $0.18\mu m$ technology is used in this experiment.

3.2.4 Coupling between wires on the same layer

To isolate the impact of crossunders and crossovers, we study the following two patterns. The *optimistic* pattern treats layers $i \pm 1$ as ground planes; this emphasizes the shielding effect due to crossunders and crossovers and in general leads to underestimation of couplings between wires on layer i. The *pessimistic* pattern treats layers $i \pm 2$ as ground planes without any wires on layers $i \pm 1$; this removes all shielding effects due to crossunders and crossovers and in general leads to overestimation of couplings between wires on layer i.

We first study the coupling between a victim wire and its non-immediate neighbors. We use a 40×40 ground plane and wires of length 20. Orthogonally with respect to the optimistic and the pessimistic patterns, we again have a *real* pattern and a *model* pattern. The real pattern has five wires on layer i, *l2*, *l1*, *victim*, *r1* and *r2* at spacing $s = 1.0$. The model pattern has only the immediate neighbors (*l1* and *r1*) of the victim. According to Table 3.10, differences in the total capacitance for the victim between the real pattern and the model pattern are less than 0.2%. The error associated with the model pattern for the coupling between the victim and its immediate neighbors is approximately 3%. Therefore, coupling analysis to wires in the same layer need only consider nearest neighbors.

We also study interactions between the victim's two neighbors as well as the effect of neighbor widths. We consider only the worst case interaction, given by the pessimistic pattern with wires (victim and two neighbors) on layer i and ground planes on layers $i \pm 2$. We use a 40×40 ground plane and wires of length 10. To observe the interaction between the victim's neighbors, we vary the spacings between the victim and its neighbors, and measure the change in total capacitance of the victim. Let C_{l-r} be the total capacitance of the victim when the left and right neighbors are distance l and r away ($l = \infty$ or $r = \infty$

$0.50\mu m$	real					model		
	$C_{i,i}$	$C_{i,l2}$	$C_{i,l1}$	$C_{i,r1}$	$C_{i,r2}$	$C_{i,i}$	$C_{i,l1}$	$C_{i,r1}$
optimistic	909.3	23.27	316.2	312.0	26.89	906.4	327.9	327.6
pessimistic	886.6	32.66	332.9	327.2	37.24	881.9	349.9	351.5
$0.35\mu m$	real					model		
	$C_{i,i}$	$C_{i,l2}$	$C_{i,l1}$	$C_{i,r1}$	$C_{i,r2}$	$C_{i,i}$	$C_{i,l1}$	$C_{i,r1}$
optimistic	1120.0	0.8489	377.8	380.3	0.8425	1115	386.9	386.5
pessimistic	891.9	27.33	328.0	324.1	31.7	888.5	341.1	341.5
$0.18\mu m$	real					model		
	$C_{i,i}$	$C_{i,l2}$	$C_{i,l1}$	$C_{i,r1}$	$C_{i,r2}$	$C_{i,i}$	$C_{i,l1}$	$C_{i,r1}$
optimistic	1451	32.04	602.2	602.1	31.67	1449	602.2	619.9
pessimistic	1436	54.9	616.6	616.5	54.86	1436	639.8	639.6

Table 3.10: Total capacitance $C_{i,i}$ of the victim wire on layer i and couplings $C_{i,l2}, C_{i,l1}, C_{i,r1}$ and $C_{i,r2}$ between the victim wire and its neighbors ($l2, l1, r1$ and $r2$).

$0.50\mu m$															
spacing	1-1	1-2	1-3	1-4	1-∞	2-2	2-3	2-4	2-∞	3-3	3-4	3-∞	4-4	4-∞	∞-∞
simulated	471.5	416.3	399.5	393.0	388.2	359.7	343.0	336.3	329.6	325.9	318.9	313.2	309.1	303.1	300.0
derived	-	415.6	398.7	390.3	385.8	-	342.8	334.4	329.8	-	317.5	313.0	-	304.6	-
$0.35\mu m$															
spacing	1-1	1-2	1-3	1-4	1-∞	2-2	2-3	2-4	2-∞	3-3	3-4	3-∞	4-4	4-∞	∞-∞
simulated	568.4	490.2	466.8	457.4	447.4	410.0	385.7	376.3	365.2	360.9	351.2	339.8	340.7	329.1	318.3
derived	-	489.2	464.65	454.6	443.4	-	385.5	375.4	364.2	-	350.8	339.6	-	329.5	-
$0.18\mu m$															
spacing	1-1	1-2	1-3	1-4	1-∞	2-2	2-3	2-4	2-∞	3-3	3-4	3-∞	4-4	4-∞	∞-∞
simulated	764.5	639.2	600.0	582.5	559.7	511.5	471.0	452.8	430.3	429.7	411.0	387.7	393.3	368.1	341.7
derived	-	638.0	597.1	578.9	553.1	-	470.6	452.4	426.6	-	411.5	385.7	-	367.5	-

Table 3.11: Simulated and derived total capacitances of the victim in cases of different neighbor spacing.

capacitance $C_{i,i}$				
neighbor widths	1	2	3	4
$0.50\mu m$	471.5	471.8	471.7	471.4
$0.35\mu m$	568.4	568.8	568.6	568.5
$0.18\mu m$	764.5	765.2	764.9	764.4

Table 3.12: Total capacitances $C_{i,i}$ for the victim in case of different neighbor widths.

indicates no left or right neighbor). Simulation and derived results are given in Table 3.11. The *derived* values are based on formula $C_{l-r} = (C_{l-l} + C_{r-r})/2$. Since differences between the simulated and derived values are often less than 1.0%, we see that couplings on opposite sides can be considered independently. To assess the effect of neighbor widths, we assume that the victim has a fixed width of 1.0, and that two neighbors (at spacing 1.0) have identical widths. We vary the widths of the neighbors and observe the change in total capacitance of the victim (see Table 3.12). Since the maximum variation is less than 0.2%, widths of neighbors can be ignored. We summarize the experiments of this section by:

Foundation 4 *Coupling analysis to wires in the same layer need only consider nearest neighbors independently, with the widths of same-layer neighbor wires having negligible effect on the coupling.*

3.2.5 Coupling between wires on layer i and wires on layers $i \pm 1$

To study the interaction between crossunder coupling and the crossover coupling, we first observe the impact of the crossover coupling on the crossunder coupling.

We assume that layer i has twelve wires $i1$, $\cdots$, $i12$, i.e., area occupancy of 30%, and that layer $i-1$ has one wire (*crossunder*) and same-layer neighbors at spacings $s = 1.0$.[5] We solve one pattern which treats layer $i+1$ as a ground plane (*full crossing*), which models the greatest possible effect due to crossovers, and a second pattern which treats layer $i+2$ as a ground plane without any wires on layers $i+1$ (*no crossing*), which models no effect due to crossovers. Again, we use a 40×40 ground plane and wires of length 20.

We compute the crossunder coupling between wires on layer i and the central wire on layer $i-1$ (see Table 3.13). The difference between the two extreme cases is less than 6% (excluding cases at the boundary). Recall that the total crossunder and crossover coupling accounts for about one third of total capacitance for a victim; hence, the crossunder coupling can be computed independently without considering crossovers while introducing an error of at most 2% for the victim's total capacitance. Due to the symmetry between crossunders and crossovers, we have:

Foundation 5 *The joint interaction of layers $i-1$ and $i+1$ on layer i is negligible; therefore, corrections for orthogonal crossovers and crossunders can be performed independently.*

3.3 2 1/2-D Methodology

The above foundations justify a simplified yet accurate 2 1/2-D extraction methodology. The Cadence Silicon Ensemble 5.0 product contains just such a methodology [84]. For a *victim* wire on layer i, it looks into a neighborhood structure

[5] We note that a good experiment setting here should have wires of certain density on layer i, which leads to alleviated coupling between crossovers and crossunders and is a more realistic case.

	crossover	$C_{i1,2}$	$C_{i2,2}$	$C_{i3,2}$	$C_{i4,2}$	$C_{i5,2}$	$C_{i6,2}$	$C_{i7,2}$	$C_{i8,2}$	$C_{i9,2}$	$C_{i10,2}$	$C_{i11,2}$	$C_{i12,2}$
$0.50\mu m$	full	1.628	10.71	24.02	25.29	25.41	25.03	25.22	25.00	25.06	22.27	7.734	0.8726
	no	1.709	10.78	24.08	25.36	25.47	25.08	25.29	25.06	25.16	22.31	7.800	0.9085
$0.35\mu m$	full	2.262	11.62	23.39	24.36	24.42	24.07	24.25	24.07	24.79	21.81	8.234	0.8522
	no	2.659	12.11	23.96	25.04	25.19	24.9	25.12	24.94	25.61	22.47	8.758	1.328
$0.18\mu m$	full	6.31	4.798	8.827	11.2	11.77	11.45	11.14	11.91	11.99	11.55	9.545	7.137
	no	5.946	4.902	8.576	11.5	12.35	12.25	12.09	12.91	12.91	12.27	9.935	7.334

Table 3.13: Crossunder couplings between wires $i1, \cdots, i12$ on layer i and the crossunder on layer $i-1$ with its same-layer neighbors at spacing 1.0 and ∞, for both full and no crossover.

which contains layers $i \pm 2$ as ground planes, all same-layer immediate neighbors, all crossunders on layer $i-1$, and all crossovers on layer $i+1$. Rather than running 3-D field solver each time, capacitance components are generated based on predetermined capacitance coefficients and then added up to obtain the lumped capacitance for the victim. Lateral, area and fringe capacitance coefficients are predetermined for different widths and spacings for the victim on layer i. Crossover and crossunder correction capacitances are predetermined for different wire widths and spacings in both crossing layers $i \pm 1$ and layer i. In this section, we first present methods to generate capacitance coefficients by using 3-D simulation on predetermined patterns, then discuss the Cadence 2 1/2-D method to compute the lumped capacitance from capacitance coefficients. The methodology is developed and validated in cooperation with Motorola and other ASIC suppliers. Examples are also given to show the accuracy of the methodology.

3.3.1 Capacitance Component Generation

We assume the following:

- the substrate is a ground plane for layer i only if $i = 1$ or $i = 2$;
- each of layer $i+2$ and layer $i-2$, if it exists, is a ground plane (resp. the *top* and *bottom* ground planes), and hence no couplings to layers beyond them need be considered; and
- if there is no layer $i+2$, then there is no top ground plane (the substrate or layer $i-2$ is the bottom ground plane).

For 3-D simulation, ground planes should be large enough to cover wires on layers i and $i \pm 1$. In addition, we assume all wires have length l.

3.3.1.1 Lateral, Area and Fringe Capacitances

In addition to the bottom ground and the possible top ground, we assume a pattern with three wires (victim and neighbors) on layer i (see Figure 3.6) and no wires on layer $i \pm 1$. It is used to generate lateral, area and fringe capacitances for width w and spacing s, where lateral capacitance is the coupling between sidewalls of the victim and sidewalls of its neighbors, area capacitance is the coupling between grounds and the top/bottom sides of the victim, and fringe capacitance is the coupling between grounds and sidewalls of the victim.

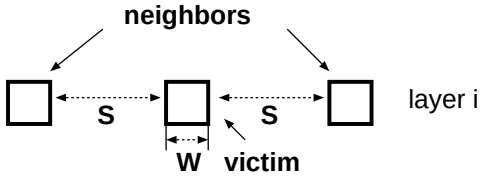


Figure 3.6: The geometric structure on layer i for lateral, area and fringe capacitance generation.

3-D simulation by FastCap is used to solve the pattern. Let the total capacitance for the victim be C_{self}, and coupling capacitance between it and its neighbors be C_{c1} and C_{c2}[6]. Then, the lateral capacitance coefficient C_l, defined as the lateral capacitance *per-length* and *per-side*, is given by

$$C_l = (C_{c1} + C_{c2})/(2 \cdot l) \tag{3.1}$$

The remaining part $C_{self} - (C_{c1} + C_{c2})$ is assumed to be the area and fringe capacitances. It needs to be separated into two parts and there are many ways to do it. One way is to run the above simulation with another width w' for the victim, and C'_{self}, C'_{c1} and C'_{c2} are obtained. The following equations are used to

[6] In theory, $C_{c1} = C_{c2}$. But numeric roundoff may cause slight difference.

separate the area and fringe components:

$$C_{self} - (C_{c1} + C_{c2}) = 2 \cdot l \cdot C_a + 2 \cdot (l + w) \cdot C_f \quad (3.2)$$

$$C'_{self} - (C'_{c1} + C'_{c2}) = 2 \cdot l \cdot C_a + 2 \cdot (l + w') \cdot C_f \quad (3.3)$$

where C_a is the area capacitance coefficient, defined as the area capacitance *per-length* and *per-side*, and C_f is the fringe capacitance coefficient, defined as the fringe capacitance *per-length* and *per-side*. Coefficients C_a and C_f can be computed by solving Equations (2) and (3) simultaneously.

3.3.1.2 Crossover and Crossunder Correction Capacitances

According to Foundation 5, crossover and crossunder correction capacitances (C_{over} and C_{under}) can be generated independently. We simulate two patterns in order to compute C_{over}. First, we run 3-D simulation by FastCap on the pattern in Figure 3.7(a). In addition to the bottom ground and the possible top ground, there are three wires on layer i and three wires on layer $i+1$. Wires on layer i have width w and spacing s. Crossovers on layer $i+1$ have width w_c and spacing s_c. In general, crossover correction capacitance is a function of w, s, w_c and s_c. We obtain the total capacitance C_{self} for the central wire (the victim) on layer i.

Then, we run FastCap on the pattern in Figure 3.7(b). Difference from the pattern in Figure 3.7(a), it has only two wires on layer $i+1$. We obtain the total capacitance C'_{self} for the victim. Finally, crossover correction capacitance C_{over} is given by

$$C_{over} = (C_{self} - C'_{self})/4 \quad (3.4)$$

The division is necessary since C_{over} is defined for each corner, and added to the

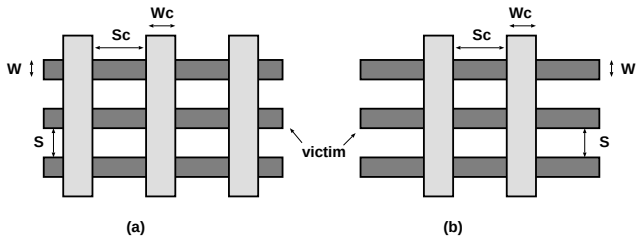


Figure 3.7: The geometric structure on layers i and $i+1$ for crossover correction capacitance generation.

lumped capacitance for the victim corner by corner as shown in Table 3.14. The crossunder correction capacitance C_{under} can be generated similarly.

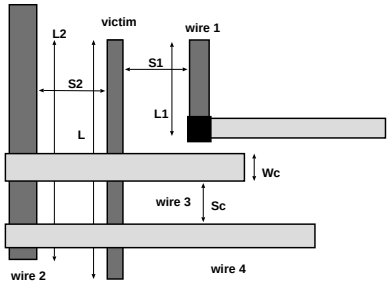


Figure 3.8: An example for 2 1/2-D capacitance analysis.

3.3.2 Algorithm for 2 1/2-D Analysis

A typical situation that can occur in gate-array and standard-cell digital IC designs is shown in Figure 3.8. The victim wire on layer i is being analyzed. *Wire1* and *wire2* are same-layer neighbors and *wire3* and *wire4* are crossovers. We first obtain geometric parameters from the layout database, including width w for the victim, *effective* lengths[7] and spacings, $< l_1, s_1 >$ and $< l_2, s_2 >$, for neighbors wire1 and wire2, and the width and spacing, $< w_c, s_c >$, for every crossover

[7]The effective length of a neighbor for a victim is length of the parallel part of both wires.

and crossunder (only the width and spacing for the crossover wire3 are shown in this Figure). This information is then used to calculate the capacitance for the victim as given in Table 3.14. In this table, the $look\_up\_range$ is the minimum spacing between the victim and its neighbor when the coupling between them is negligible. Furthermore, linear interpolation is used to compute values between different widths in the look-up table, and linear interpolation on $1/spacing$ is used to compute values between different spacings in the look-up table. Therefore, at least two widths and at least two spacing for each width should be pre-analyzed and stored in the look-up table. If only two spacings are used for a width, one should be the $look\_up\_range$. Moreover, linear extrapolation is used for widths and spacings that exceed values given in the look-up table. Closed-form formulas which are more sophisticated than our linear interpolation and extrapolation are were presented in [11, 1]. Those formulas can also be used here. However, in gate-array and standard-cell digital IC designs, choices of widths and spacings are usually limited. Therefore, a look-up table with a reasonable size can be sufficient with limited using of interpolation and extrapolation. It was shown by real designs that our linear interpolation and extrapolation produces satisfactory results.

3.3.3 Examples with 2 1/2-D analysis

Two nets were extracted from real designs and their lumped capacitances were computed based on the 2 1/2-D methodology. We also separated the two nets into a number of small sections and used 3-D simulator FastCap to solve all the relevant geometries for such sections. Capacitances from different sections were then added up. We compared results from two methods in Table 3.15. The errors are 0.54% for the smaller net and 3.33% for the larger one. Moreover, our method

Begin 2 1/2-D analysis

For any given victim wire segment of width w and length l

Let the lumped capacitance for the victim be $C_{self} = 0$

For each side of length l

Find out all the same-layer neighbors within *look_up_range*

Let their effective lengths be l_i and spacing s_i

For each neighbor

Lookup lateral, area and fringe capacitance coefficients

(C_l, C_a and C_f) for s_i and w

$C_{self} = C_{self} + (C_l + C_a + C_f) \cdot l_i$

For any crossover of width w_c

Get the next crossover and determine spacing s_c

Lookup the crossover correction capacitance C_{over} for w, s_i, w_c, s_c

$C_{self} = C_{self} + C_{over}$

For each side of a crossunder of width w_c

Determine spacing s_c to the neighboring crossunder in the side

Lookup the crossunder correction capacitance C_{under} for w, s_i, w_c, s_c

$C_{self} = C_{self} + C_{under}$

For each side of width w

Find out all the same-layer neighbors within *look_up_range*

Let their effective lengths be l_i and spacing s_i

For each neighbor

Lookup lateral and fringe capacitance coefficients

(C_l and C_f) for s_i and w

$C_{self} = C_{self} + (C_l + C_f) \cdot l_i$

End

Table 3.14: Algorithm for 2 1/2-D analysis.

	2 1/2-D analysis	3-D simulation	error
smaller net	6.53552	6.5713	-0.54%
larger net	3152.42	3261.17	-3.33%

Table 3.15: Comparison between 2 1/2-D analysis and 3-D simulation

has been validated in cooperation with Motorola and other ASIC suppliers by comparing extracted values with measured values.

A software tool has been developed to automate the coefficient generation. The input to this program consists of thicknesses[8], permissible widths and spacings for wires on every metal layer, and thickness and effective permittivity for the dielectric between adjacent metal layers. The tool automatically generates capacitance coefficients based on 3-D simulation and writes out a LEF file containing all capacitance coefficients.

The extraction methodology can be used for MCM and PCB designs as well. In Table 3.16, we report the capacitance coefficients computed for the MMS MCM technology that is available through the MCM Interconnect Designer's Access Service from Information Science Institute. There are one power plane and one ground plane to sandwich each pair of routing planes. The wires are $4\mu m$ thick, and are $12\mu m$ and $21.5\mu m$ away from the power plane and ground plane, respectively. In the following, we compute the capacitance value for a $38\mu m$-wide line with two $19\mu m$-wide lines at spacings of $31\mu m$ and $56\mu m$, respectively. We assume that all lines are 1cm long, and obtain a capacitance value of 2.265pF using capacitance coefficients in Table 3.16. The capacitance value given by FastCap is 2.300pF. The difference between two values is less than 2%.

[8]It is possible to have different wire thicknesses in a metal layer.

width(μm)	space(μm)	$c_a(fF/\mu m)$	$c_f(fF/\mu m)$	$c_x(fF/\mu m)$
19	31	0.42	0.51	0.22
	43	0.43	0.59	0.11
	56	0.43	0.65	0.06
38	31	0.82	0.48	0.26
	43	0.82	0.57	0.14
	56	0.83	0.63	0.07

Table 3.16: Capacitance coefficients for a MCM technology. c_a, c_f and c_x are unit-length area, fringe and lateral capacitances, respectively.

3.4 Conclusions

We have addressed post-routing capacitance extraction as it is typically required during performance-driven layout. The context of our methodology would be any standard deep-submicron ASIC design flow, e.g., with a back-end block implementation loop consisting of (1) (incremental) placement, (2) (incremental) global and detailed routing, (3) parasitic extraction, (4) delay calculation and timing/noise analysis (5) driver, buffer and routing topology/wirewidth optimization, and (6) netlist engineering change. It can be also used for MCM/PCB designs.

We have validated five "foundations" that allow simplification of the capacitance extraction problem for multilayer interconnects. We have also described a simple yet accurate 2 1/2-D extraction methodology, now implemented in a commercial cell-based layout tool. We showed that this methodology gives results that are very close (within a few percent) to measured silicon and 3-D numerical simulation. In Chapter 6, we will use this 2 1/2-D extraction methodology,

in other words, the 2 1/2-D capacitance model to study the interconnect sizing and spacing problem with consideration of coupling capacitance between multiple nets.

CHAPTER 4

Theory and Algorithm for Local-Refinement Based Optimization

The local refinement (LR) operation is an optimization technique for solving multi-variable optimization problems with complex objective functions. It was first introduced in [29] to solve the single-source wire sizing problem using iterative LR operations. In Chapter 2, the algorithm was extended to the multi-source wire sizing problem. Furthermore, the bundled refinement operation, which is also called bundled-LR operation, was proposed to replace the LR operation and obtain a speedup of up to 100x. We may refer the algorithms using the LR or bundled-LR operations as LR-based optimization algorithm. The effectiveness and efficiency of LR-based optimization lead us to study in this chapter the following problem: can we use the LR-based algorithm to solve other problems?

Our contributions in this chapter include:

- We formulate three classes of optimization problems: the simple, monotonically constrained, and bounded CH-programs. The simple CH-program contains the single-source and multi-source wire sizing problems, and is a subset of the monotonically constrained CH-program. In turn, the monotonically constrained CH-program is a subset of the bounded CH-program.
- We generalize the concept of the LR operation, and introduce the concepts

of the pseudo-LR operation and the extended-LR operation. We then reveal the dominance property (Theorem 8) under the LR operation for the simple CH-program, as well as the general dominance property (Theorem 9) under the pseudo-LR operation for the monotonically constrained CH-program and under the extended-LR operation for the bounded CH-program.

- Based on the dominance property and the general dominance property we propose a very efficient polynomial-time algorithm, using different types of LR operations to compute tight lower and upper bounds of the exact solution to any CH-program.

Note that the dominance property in [29] and Chapter 2 is applicable only to the single-source and multi-source wire sizing problem, respectively. The dominance property to be presented is applicable to a class of optimization problems, the simple CH-program that includes the single-source and multi-source wire sizing problems. Nevertheless, the general dominance property is applicable to even larger classes of optimization problems.

The remainder of the chapter is organized as follows: We formulate three classes of CH-programs in Section 4.1, and present their properties in Section 4.2. We propose a polynomial-time bound-computation algorithm in Section 4.3, and compare the algorithm for CH-programs with the coordinate-descent algorithm for the posynomial program in Section 4.4. We will apply the formulations and algorithm of CH-programs to solve the simultaneous device sizing, and wire sizing and spacing problems under accurate models for device delay and interconnect coupling capacitance in Chapters 5 and 6. Results in this chapter have been presented in conference papers [18, 20, 19], and a journal paper [25].

4.1 Formulations of CH-functions

We first define the CH-function (*Cong-He* function)[1] as a function of a positive vector $\mathbf{X} = \{x_i \mid x_i \geq 0, i = 1, \cdots, n\}$ with the following form:

$$f(\mathbf{X}) = \sum_{p \geq 0} \sum_{q \geq 0} \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} \left(\frac{a_{p,q,i,j}(\mathbf{X})}{x_i^p}\right) \cdot \left(b_{p,q,i,j}(\mathbf{X}) \cdot x_j^q\right) \tag{4.1}$$

where coefficients $a_{p,q,i,j}(\mathbf{X})$ and $b_{p,q,i,j}(\mathbf{X})$, as well as exponents p and q, are positive.

Depending on the coefficient $a_{p,q,i,j}(\mathbf{X})$ and $b_{p,q,i,j}(\mathbf{X})$, we define the following three types of CH-functions:

Definition 6 (simple CH-function) *Eqn. (4.1) is a simple CH-function if coefficients $a_{p,q,i,j}$ and $b_{p,q,i,j}$ are constants.*

The concept of simple CH-function was first introduced in [18, 20]. It was shown that many previous works on device and interconnect sizing problems, including the single-source and multi-source wire sizing problems [29, 16], continuous wire sizing problem [8], and simultaneous driver and wire sizing problem [26], use simple CH-functions as objective functions.

In some applications however, coefficients $a_{p,q,i,j}(\mathbf{X})$ and $b_{p,q,i,j}(\mathbf{X})$ may vary as functions depending on $\mathbf{X}$. For two vectors $\mathbf{X}$ and $\mathbf{X}'$, we say that $\mathbf{X}$ dominates $\mathbf{X}'$ (denoted by $\mathbf{X} \geq \mathbf{X}'$) if $x_i \geq x_i'$ for $i = 1, \cdots, n$. We then define the following *monotonically constrained CH-function*:

Definition 7 (monotonically constrained CH-function) *Eqn. (4.1) is a monotonically constrained CH-function, if it satisfies the following monotonic*

[1] CH-function was called CH-posynomial in [20, 19]. We renamed it to show that it is not, in general, a posynomial.

constraints: for any vector $\mathbf{X}' \geq \mathbf{X}$, *(i)* $\frac{a_{p,q,i,j}(\mathbf{X}')}{x_i'^p} \leq \frac{a_{p,q,i,j}(\mathbf{X})}{x_i^p}$ *and* $a_{p,q,i,j}(\mathbf{X}') \geq a_{p,q,i,j}(\mathbf{X})$*; (ii)* $b_{p,q,i,j}(\mathbf{X}') \cdot x_j'^q \geq b_{p,q,i,j}(\mathbf{X}) \cdot x_j^q$ *and* $b_{p,q,i,j}(\mathbf{X}') \leq b_{p,q,i,j}(\mathbf{X})$.

The monotonically constrained CH-function was defined differently (and called bounded-variation CH-posynomial[2]) in [19], where we say (i) $a_{p,q,i,j}$ is a function depending only on x_i. With respect to an increase of x_i, $\frac{a_{p,q,i,j}(x_i)}{x_i^p}$ monotonically decreases and $a_{p,q,i,j}(x_i)$ monotonically increases; (ii) $b_{p,q,i,j}$ is a function depending only on x_j. With respect to an increase of x_j, $b_{p,q,i,j}(x_j) \cdot x_j^q$ monotonically increases and $b_{p,q,i,j}(x_j)$ monotonically decreases. It is easy to see that Definition 7 subsumes the old definition and covers a wider class of functions, because now each coefficient may vary as a function of all variables in $\mathbf{X}$, instead of a single variable in [19].

We finally remove the monotonic constraints for the CH-function by formulating the following *bounded CH-function*:

Definition 8 (bounded CH-function) *Eqn. (4.1) is a bounded CH-function, if its coefficients are bounded: for any* p, q, i *and* j*, there exist positive constant* $a^L_{p,q,i,j}$, $a^U_{p,q,i,j}$, $b^L_{p,q,i,j}$ *and* $b^U_{p,q,i,j}$, *such that* $a^L_{p,q,i,j} \leq a_{p,q,i,j}(\mathbf{X}) \leq a^U_{p,q,i,j}$ *and* $b^L_{p,q,i,j} \leq b_{p,q,i,j}(\mathbf{X}) \leq b^U_{p,q,i,j}$.

Clearly, the simple CH-function is a subset of the monotonically constrained CH-function, which in turn is a subset of the bounded CH-function (see Figure 4.1). In addition, the simple CH-function is a subset of the posynomial. A posynomial [36] is a function of a positive vector $\mathbf{X}$ having the form $g(\mathbf{X}) = \sum_{i=1}^{m} u_i(\mathbf{X})$ with

$$u_i(\mathbf{X}) = c_i x_1^{a_{i1}} x_2^{a_{i2}} \cdots x_n^{a_{in}}, \qquad i = 1, 2, \cdots, m \tag{4.2}$$

[2]We saved the name "bounded" for the type of CH-function defined in Definition 8, which was called the general CH-posynomial in [19].

where the exponents a_{ij} are real numbers and the coefficients c_i are positive. For example,

$$f(x_1, x_2, x_3) = x_1^{1/2} + 1/x_2 + x_3 \tag{4.3}$$

is a simple CH-function as well as a posynomial. However,

$$f(x_1, x_2, x_3) = x_1^2 \cdot x_2 \cdot \frac{1}{x_3} \tag{4.4}$$

is a posynomial but *not* a simple CH-function. On the other hand, the monotonically constrained and bounded CH-functions may be no longer a posynomial. For example,

$$f(x_1, x_2) = \frac{1}{\ln x_1} \cdot x_1^2 + \frac{x_2}{x_1}, \quad x_1 > 3 \tag{4.5}$$

is neither a simple CH-function nor a posynomial. However, one can easily verify that it is a monotonically constrained CH-function by treating $\frac{1}{\ln x_1}$ as the coefficient function for x_1^2.

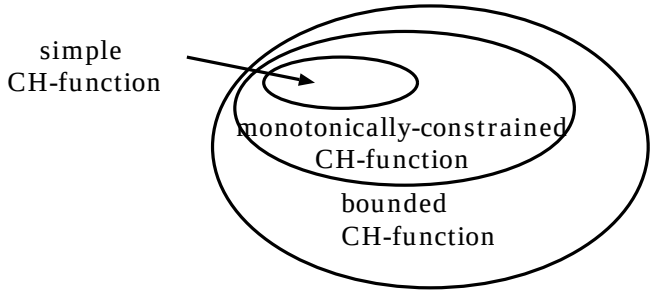


Figure 4.1: The simple CH-function is a subset of the monotonically constrained CH-function, which is in turn a subset of the bounded CH-function.

4.2 Properties for CH-programs

We define the *CH-program* as an optimization problem to minimize a CH-function subject to $\mathbf{L} \leq \mathbf{X} \leq \mathbf{U}$ (i.e., $l_i \leq x_i \leq u_i$ for $i = 1, \cdots, n$). It may be a sim-

ple, monotonically constrained or bounded CH-program depending on whether its objective function is a simple, monotonically constrained or bounded CH-function. We will introduce the dominance property for the simple and monotonically constrained CH-programs, as well as the general dominance property for the monotonically constrained and bounded CH-programs.

4.2.1 Dominance Property

We first define the following local refinement operation:

Definition 9 (local refinement operation) *Given a function $f(\mathbf{X})$ and a solution vector (or simply, a solution) $\mathbf{X}'$, the local refinement operation for any particular variable x_i is to minimize $f(\mathbf{X})$ by only varying x_i while keeping all values of other $x_j (j \neq i)$ in $\mathbf{X}'$ fixed.*

Such an operation is also called an *LR operation* in short. The resulting solution vector is called the *local refinement* of $\mathbf{X}'$ (with respect to x_i).

Furthermore, we define

$$g(\mathbf{X}) \quad = \quad \sum_{i=1}^{n} \sum_{j=1}^{n} A_i(x_i) \cdot B_j(x_j) \tag{4.6}$$

where $A_i(x_i)$ is a function depending only on x_i, and it decreases with respect to an increase of x_i; $B_j(x_j)$ is a function depending only on x_j, and it increases with respect to an increase of x_j. We may prove the following Lemma 2:

Lemma 2 *Let $\mathbf{X}^*$ an exact solution to minimize $g(\mathbf{X})$ (Eqn. (4.6)). For any solution $\mathbf{X}'$ of $f(\mathbf{X})$, if $\mathbf{X}'$ dominates $\mathbf{X}^*$, any local refinement of $\mathbf{X}'$ leads to a solution that still dominates $\mathbf{X}^*$. Similarly, if $\mathbf{X}'$ is dominated by $\mathbf{X}^*$, any local refinement of $\mathbf{X}'$ leads to a solution that is still dominated by $\mathbf{X}^*$.*

Proof:

With respect to any particular x_i and Eqn. (4.6), we define the following:

$$\Theta(\mathbf{X} - \{x_i\}) = \sum_{j=1, j\neq i}^{n} \sum_{k=1, k\neq i}^{n} A_j(x_j) \cdot B_k(x_k) \tag{4.7}$$

Then, the objective function (4.6) can be written as the following for x_i:

$$\begin{aligned} g(\mathbf{X}, i) &= \sum_{j=1, j\neq i}^{n} A_i(x_i) \cdot B_j(x_j) + \sum_{j=1, j\neq i}^{n} A_j(x_j) \cdot B_i(x_i) \\ &+ \Theta(\mathbf{X} - \{x_i\}) \end{aligned} \tag{4.8}$$

Let $\mathbf{X}^*$ be the exact solution to minimize Eqn. (4.6), and $\tilde{x}_i$ be the *local optimum* of x_i with respect to a particular solution $\mathbf{X}'$. According to the definition of the LR operation, we have:

$$\begin{aligned} &\sum_{j=1, j\neq i}^{n} A_i(\tilde{x}_i) \cdot B_j(x'_j) + \sum_{j=1, j\neq i}^{n} A_j(x'_j) \cdot B_i(\tilde{x}_i) + \Theta(\mathbf{X}' - \{x_i\}) \\ \leq &\sum_{j=1, j\neq i}^{n} A_i(x^*_i) \cdot B_j(x'_j) + \sum_{j=1, j\neq i}^{n} A_j(x'_j) \cdot B_i(x^*_i) + \Theta(\mathbf{X}' - \{x_i\}) \end{aligned} \tag{4.9}$$

where x^*_i is the value for x_i in the exact solution $\mathbf{X}^*$. Because x^*_i must be the *local optimum* for x_i with respect to $\mathbf{X}^*$, we have:

$$\begin{aligned} &\sum_{j=1, j\neq i}^{n} A_i(x^*_i) \cdot B_j(x^*_j) + \sum_{j=1, j\neq i}^{n} + A_j(x^*_j) \cdot B_i(x^*_i) + \Theta(\mathbf{X}' - \{x_i\}) \\ \leq &\sum_{j=1, j\neq i}^{n} A_i(\tilde{x}_i) \cdot B_j(x^*_j) + \sum_{j=1, j\neq i}^{n} + A_j(x^*_j) \cdot B_i(\tilde{x}_i) + \Theta(\mathbf{X}' - \{x_i\}) \end{aligned} \tag{4.10}$$

Summing up (4.9) and (4.10), we obtain:

$$\begin{aligned} &\sum_{j=1, j\neq i}^{n} (A_j(x'_j) - A_j(x^*_j)) \cdot (B_i(\tilde{x}_i) - B_i(x^*_i)) \\ + &\sum_{j=1, j\neq i}^{n} (B_j(x'_j) - B_j(x^*_j)) \cdot (A_i(\tilde{x}_i) - A_i(x^*_i)) \\ \leq &\ 0 \end{aligned} \tag{4.11}$$

When $\mathbf{X}'$ dominates $\mathbf{X}^*$ (i.e., $\mathbf{X}' \geq \mathbf{X}^*$), according to the definition of Eqn. (4.6), we have the following:

$$A_j(x'_j) - A_j(x^*_j) \leq 0 \tag{4.12}$$

$$B_j(x'_j) - B_j(x^*_j) \geq 0; \tag{4.13}$$

and

$$A_i(x'_i) - A_i(x^*_i) \leq 0, \tag{4.14}$$

$$B_i(x'_i) - B_i(x^*_i) \geq 0. \tag{4.15}$$

For the purpose of contraction, we assume that the dominance property does not hold so that $\tilde{x}_i < x^*_i$. Then, according to definition of Eqn. (4.6), we have

$$A_i(\tilde{x}_i) - A_i(x^*_i) > 0, \tag{4.16}$$

$$B_i(\tilde{x}_i) - B_i(x^*_i) < 0. \tag{4.17}$$

Without loss of generality, we assume that for at least one j, the strict inequality holds for either Eqn. (4.12) or (4.13)[3], then,

$$\begin{aligned} & \sum_{j=1, j\neq i}^{n} (A_j(x'_j) - A_j(x^*_j)) \cdot (B_i(\tilde{x}_i) - B_i(x^*_i)) \\ + & \sum_{j=1, j\neq i}^{n} (B_j(x'_j) - B_j(x^*_j)) \cdot (A_i(\tilde{x}_i) - A_i(x^*_i)) \\ > & \ 0 \end{aligned} \tag{4.18}$$

holds, which is a contraction to Eqn. (4.11). Therefore, the dominance property holds for $\mathbf{X}' \geq \mathbf{X}^*$. The dominance property can be proved symmetrically for $\mathbf{X}' \leq \mathbf{X}^*$. □

Because the simple CH-function is a subset of Eqn. (4.6), we have the following dominance property based on Lemma 2:[4]

[3]Otherwise, x'_i is the local optimal value for variable x_i, Theorem 1 still holds.

[4]Nevertheless, Lemma 2 also reveals that the dominance property holds for the monotonically constrained CH-program when coefficients are functions of single variables, like the

Theorem 8 (Dominance Property) *Let $f(\mathbf{X})$ be a simple CH-function, and $\mathbf{X}^*$ an exact solution to minimize $f(\mathbf{X})$. For any solution $\mathbf{X}'$ of $f(\mathbf{X})$, if $\mathbf{X}'$ dominates $\mathbf{X}^*$, any local refinement of $\mathbf{X}'$ leads to a solution that still dominates $\mathbf{X}^*$. Similarly, if $\mathbf{X}'$ is dominated by $\mathbf{X}^*$, any local refinement of $\mathbf{X}'$ leads to a solution that is still dominated by $\mathbf{X}^*$.*

The dominance property under the LR operation was first introduced for the single-source wire sizing problem [29], and was extended to the multi-source wire sizing problem [16]. In [20], it was revealed that the dominance property holds for all simple CH-programs. It was also shown that both wire sizing problems [29, 16], the simultaneous driver/buffer and wire sizing problem, and simultaneous transistor and interconnect sizing problem are all simple CH-programs if simple device and capacitance models are used. Therefore, the dominance property holds for these problems and enables an LR-based algorithm, which uses iterative LR operations to compute optimal sizes for both devices and wires.[5]

When coefficients for variable x_i, like the case of simple CH-program, are all constants, the LR operation of x_i is a single-variable posynomial program that can be solved very efficiently.[6] The LR operation for other CH-programs may be less efficient, however. First, it might be no longer a posynomial program. An

bounded-variation CH-program defined in [19]. The dominance property, however, may not hold for the new defined monotonically constrained CH-program when coefficients are functions of solution vector $\mathbf{X}$.

[5] The SDWS algorithm for simultaneous driver and wire sizing problem in [26] is different from and less efficient than the LR-based algorithm in [20] because mathematic package Maple is invoked iteratively to compute the driver size..

[6] According to [36], a posynomial program is the following minimization problem:

$$min \;\; g_0(\mathbf{X}) \qquad subject\ to\ g_k(\mathbf{X}) \leq 1$$
$$k = 1, 2, \cdots, p \ and \ \mathbf{X} > 0$$

where each g_k $(k = 0, 1, 2, \cdots, p)$ is a posynomial function. In the case of LR operation of x_i for a simple CH-program, the local optimum is also a global optimum no matter whether x_i has continuous or discrete value. More detailed discussion of posynomial programs can be found in Section 4.4.

example is the LR operation of x_1 to minimize Eqn. (4.5), where a logarithm function is involved. Second, when a coefficient varies depending on a table rather than a closed-form formula, we may have to enumerate all possible values for x_i in order to find out its local optimal value (an example is given in Section 6.2.1).

The usage of the LR operation is also limited by the fact that the dominance property under the LR operation generally does *not* hold for a monotonically constrained or bounded CH-program. To overcome these limitations, we introduce the pseudo-LR and extended-LR operations, then show a general dominance property.

4.2.2 General Dominance Property

The pseudo-LR and extended-LR operations (in short, the *PLR* and *ELR* operations) are defined as the following:

Definition 10 (pseudo-LR operation) *Given a CH-function $f(\mathbf{X})$ and a solution vector $\mathbf{X}'$, the pseudo-LR operation for variable x_i with respect to $\mathbf{X}'$ is an LR operation using constant coefficients $a_{p,q,i,j}(\mathbf{X}')$ and $b_{p,q,i,j}(\mathbf{X}')$ when solving the "local-optimal" x_i for any p, q, i and j.*

That is, we fix the coefficients under the *current* solution when performing an PLR operation. The PLR and LR operations are same for a simple CH-program, but may produce different results for a monotonically constrained CH-program.

Definition 11 (extended-LR operation) *Given a CH-function $f(\mathbf{X})$ and a solution $\mathbf{X}'$, the extended-LR operation for a particular variable x_i in $\mathbf{X}'$ is the LR operation using the following coefficients for any p, q, $j \neq i$, and $k \neq i$:*
(i) When $\mathbf{X}' \geq \mathbf{X}^*$, we replace $a_{p,q,ij}(\mathbf{X}')$ and $a_{p,q,k,j}(\mathbf{X}')$ by $a^U_{p,q,i,j}$ and $a^L_{p,q,k,j}$, and replace $b_{p,q,j,i}(\mathbf{X}')$ and $b_{p,q,k,j}(\mathbf{X}')$ by $b^L_{p,q,j,i}$ and $b^U_{p,q,j,k}$.

(ii) When $\mathbf{X}' \leq \mathbf{X}^*$*, we replace* $a_{p,q,ij}(\mathbf{X}')$ *and* $a_{p,q,k,j}(\mathbf{X}')$ *by* $a^L_{p,q,i,j}$ *and* $a^U_{p,q,k,j}$*, and replace* $b_{p,q,j,i}(\mathbf{X}')$ *and* $b_{p,q,k,j}(\mathbf{X}')$ *by* $b^U_{p,q,j,i}$ *and* $b^L_{p,q,j,k}$*.*

We call the solution given by the PLR or ELR operation as the *pseudo-* or *extended-local refinement* of $\mathbf{X}'$, respectively. Note that the lower and upper bounds are *not* unique for coefficient functions. The definition of the ELR operation is applicable to any valid lower and upper bounds.

According to these definitions, even though coefficients are functions of the variable vector $\mathbf{X}$ in the monotonically constrained or bounded CH-program, coefficients during each PLR or ELR operation are still treated as constants. Therefore, the PLR or ELR operation for a monotonically constrained or bounded CH-program again becomes a single-variable posynomial program that can be solved very efficiently, exactly as the LR operation for a simple CH-program. We will illustrate the PLR and ELR operations using the following CH-function:

$$\begin{aligned} f(x_1, x_2) &= \frac{a_1(x_1, x_2)}{x_1} \cdot (b_2(x_1, x_2) \cdot x_2^2) \\ &+ \frac{a_2(x_1, x_2)}{x_2} \cdot (b_1(x_1, x_2) \cdot x_1), \end{aligned} \tag{4.19}$$

The pseudo-local refinement of x_1 with respect to $\mathbf{X}' = \{x'_1, x'_2\}$ is

$$\tilde{x}_1^{PLR} = \sqrt{\frac{a_1(x'_1, x'_2) \cdot (b_2(x'_1, x'_2) \cdot x_2'^3)}{a_2(x'_1, x'_2) \cdot b_1(x'_1, x'_2)}}. \tag{4.20}$$

If we assume that $a_1(x_1, x_2) \in [a_1^L, a_1^U]$, $a_2(x_1, x_2) \in [a_2^L, a_2^U]$, $b_1(x_1, x_2) \in [b_1^L, b_1^U]$ and $b_2(x_1, x_2) \in [b_2^L, b_2^U]$. When $\{x'_1, x'_2\}$ is dominated by exact solution $\{x_1^*, x_2^*\}$, the extended-local refinement of x_1 concerning $\{x'_1, x'_2\}$ is

$$\tilde{x}_1^{ELR} = \sqrt{\frac{a_1^L \cdot (b_2^L \cdot x_2'^3)}{a_2^U \cdot b_1^U}}. \tag{4.21}$$

Even though we assume continuous variables in this example, our definition for the PLR and ELR operations (as well as the LR operation) applies to both

continuous and discrete variables. We may prove the following theorem concerning the PLR and ELR operations:

Theorem 9 (General Dominance Property) *Let* $\mathbf{X}^*$ *an exact solution to minimize a CH-function* $f(\mathbf{X})$.
(i) When $f(\mathbf{X})$ *is a monotonically constrained CH-function, for any solution* $\mathbf{X}'$ *of* $f(\mathbf{X})$, *if* $\mathbf{X}'$ *dominates* $\mathbf{X}^*$, *any pseudo-local refinement of* $\mathbf{X}'$ *leads to a solution that still dominates* $\mathbf{X}^*$; *if* $\mathbf{X}'$ *is dominated by* $\mathbf{X}^*$, *any pseudo-local refinement of* $\mathbf{X}'$ *leads to a solution that is still dominated by* $\mathbf{X}^*$.
(ii) When $f(\mathbf{X})$ *is a bounded CH-function, for any solution* $\mathbf{X}'$ *of* $f(\mathbf{X})$, *if* $\mathbf{X}'$ *dominates* $\mathbf{X}^*$, *any extended-local refinement of* $\mathbf{X}'$ *leads to a solution that still dominates* $\mathbf{X}^*$; *if* $\mathbf{X}'$ *is dominated by* $\mathbf{X}^*$, *any extended-local refinement of* $\mathbf{X}'$ *leads to a solution that is still dominated by* $\mathbf{X}^*$.

We will prove the general dominance property first under the PLR operation for the monotonically constrained CH-program, then under the ELR operation for the bounded CH-program.

A: Proof under PLR operation for Monotonically-Constrained CH-program:

With respect to any particular $x_i^p (p > 0)$, we define the following:

$$\Psi_p(\mathbf{X}, i) = \sum_{q \geq 0} \sum_{j=1, j \neq i}^{n} \left(\frac{a_{q,p,j,i}(\mathbf{X})}{x_j^q}\right) \cdot b_{q,p,j,i}, \tag{4.22}$$

$$\Phi_p(\mathbf{X}, i) = \sum_{q \geq 0} \sum_{j=1, j \neq i}^{n} a_{p,q,i,j}(\mathbf{X}) \cdot (b_{p,q,i,j}(\mathbf{X}) \cdot x_j^q), \tag{4.23}$$

We also define

$$\begin{aligned} \Theta(\mathbf{X}, i) &= \sum_{q \geq 0} \sum_{r \geq 0} \sum_{j=1, j \neq i}^{n} \sum_{k=1, k \neq j \neq i}^{n} \left(\frac{a_{qr,jk}(\mathbf{X})}{x_j^q}\right) \cdot (b_{qr,jk}(\mathbf{X}) \cdot x_k^r) \\ &+ \Psi_0(\mathbf{X}, i) + \Phi_0(\mathbf{X}, i). \end{aligned} \tag{4.24}$$

Then, the objective function (4.1) can be written as the following for x_i:

$$f(\mathbf{X},i) \;=\; \sum_{p>0}\{\Psi_p(\mathbf{X},i)\cdot x_i^p \;+\; \Phi_p(\mathbf{X},i)\cdot\frac{1}{x_i^p}\}+\Theta(\mathbf{X},i) \tag{4.25}$$

where $\Theta(\mathbf{X},i)$ is defined in Eqn. (4.24).

For any two solutions $\mathbf{X}$ and $\mathbf{X}'$ to a monotonically constrained CH-program, if $\mathbf{X}$ is dominated by $\mathbf{X}'$, one is easy to verify that:

$$\Psi_p(\mathbf{X},i) \;\geq\; \Psi_p(\mathbf{X}',i) \tag{4.26}$$

$$\Phi_p(\mathbf{X},i) \;\leq\; \Phi_p(\mathbf{X}',i) \tag{4.27}$$

Let $\mathbf{X}^*$ be the exact solution to minimize Eqn. (4.1), and $\tilde{x}_i$ be the *pseudo-local optimum* of x_i with respect to a solution $\mathbf{X}'$. According to the definition of the PLR operation, we have:

$$\begin{aligned}&\sum_{p>0}(\Psi_p(\mathbf{X}',i)\cdot\tilde{x}_i^p \;+\; \Phi_p(\mathbf{X}',i)\cdot\frac{1}{\tilde{x}_i^p}) \;+\; \Theta(\mathbf{X}',i)\\ \leq\; &\sum_{p>0}(\Psi_p(\mathbf{X}',i)\cdot x_i^{*p} \;+\; \Phi_p(\mathbf{X}',i)\cdot\frac{1}{x_i^{*p}}) \;+\; \Theta(\mathbf{X}',i)\end{aligned} \tag{4.28}$$

where x_i^* is the value for x_i in the exact solution $\mathbf{X}^*$. Because x_i^* must be the *local optimum* (as well as the pseudo-local optimum) for x_i with respect to $\mathbf{X}^*$, we have:

$$\begin{aligned}&\sum_{p>0}(\Psi_p(\mathbf{X}^*,i)\cdot x_i^{*p} \;+\; \Phi_p(\mathbf{X}^*,i)\cdot\frac{1}{x_i^{*p}}) \;+\; \Theta(\mathbf{X}^*,i)\\ \leq\; &\sum_{p>0}(\Psi_p(\mathbf{X}^*,i)\cdot\tilde{x}_i^p \;+\; \Phi_p(\mathbf{X}^*,i)\cdot\frac{1}{\tilde{x}_i^p}) \;+\; \Theta(\mathbf{X}^*,i)\end{aligned} \tag{4.29}$$

Summing up (4.28) and (4.29), we obtain:

$$\begin{aligned}&\sum_{p>0}\{\Psi_p(\mathbf{X}',i) \;-\; \Psi_p(\mathbf{X}^*,i)\}\cdot(\tilde{x}_i^p - x_i^{*p})\\ +\;&\sum_{p>0}\{\Phi_p(\mathbf{X}',i) \;-\; \Phi_p(\mathbf{X}^*,i)\}\cdot(\frac{1}{\tilde{x}_i^p}-\frac{1}{x_i^{*p}})\\ =\;&\sum_{p>0}\{\{\Psi_p(\mathbf{X}',i) \;-\; \Psi_p(\mathbf{X}^*,i) \;+\; \frac{\Phi_p(\mathbf{X}^*,i) \;-\; \Phi_p(\mathbf{X}',i)}{\tilde{x}_i^p\cdot x_i^{*p}}\}\cdot(\tilde{x}_i^p - x_i^{*p})\}\\ \leq\;&0\end{aligned} \tag{4.30}$$

If $\mathbf{X}^*$ is dominated by $\mathbf{X}'$, according to Eqn. (4.26) and (4.27), we have $\Psi_p(\mathbf{X}^*, i) \geq \Psi_p(\mathbf{X}', i)$ and $\Phi_p(\mathbf{X}^*, i) \leq \Phi_p(\mathbf{X}', i)$. We know that $\tilde{x}_i^p - x_i^{*p} \geq 0$, and therefore $\tilde{x}_i - x_i^* \geq 0$, i.e. $\mathbf{X}^*$ is still dominated by the pseudo-local refinement of $\mathbf{X}'$. Similarly, if $\mathbf{X}^*$ dominates $\mathbf{X}'$, then $\Psi_p(\mathbf{X}^*, i) \leq \Psi_p(\mathbf{X}', i)$, $\Phi_p(\mathbf{X}^*, i) \geq \Phi_p(\mathbf{X}', i)$, and $\tilde{x}_i - x_i^* \leq 0$, therefore, $\mathbf{X}^*$ still dominates the pseudo-local refinement of $\mathbf{X}'$.

B. Proof under ELR Operation for Bounded CH-program:

Let $\mathbf{X}^*$ be the exact solution to minimize Eqn. (4.1). According to Eqn. (4.22), (4.23) and (4.24), we have the following with respect to $\mathbf{X}^*$:

$$\Psi_p(\mathbf{X}^*, i) = \sum_{q \geq 0} \sum_{j=1, j \neq i}^{n} \left(\frac{a_{q,p,j,i}(\mathbf{X}^*)}{x_j^{*q}}\right) \cdot b_{q,p,j,i}(\mathbf{X}^*), \tag{4.31}$$

$$\Phi_p(\mathbf{X}^*, i) = \sum_{q \geq 0} \sum_{j=1, j \neq i}^{n} a_{p,q,i,j}(\mathbf{X}^*) \cdot (b_{p,q,i,j}(\mathbf{X}^*) \cdot x_j^{*q}), \tag{4.32}$$

$$\begin{aligned} \Theta(\mathbf{X}^*, i) &= \sum_{q \geq 0} \sum_{r \geq 0} \sum_{j=1, j \neq i}^{n} \sum_{k=1, k \neq j \neq i}^{n} \left(\frac{a_{q,r,j,k}(\mathbf{X}^*)}{x_j^{*q}}\right) \cdot (b_{q,r,j,k}(\mathbf{X}^*) \cdot x_k^{*r}) \\ &+ \sum_{q \geq 0} \sum_{j=1, j \neq i}^{n} a_{0,q,i,j}(\mathbf{X}^*) \cdot (b_{0,q,i,j}(\mathbf{X}^*) \cdot x_j^q) \\ &+ \sum_{p \geq 0} \sum_{j=1, j \neq i}^{n} \frac{a_{p,0,j,i}(\mathbf{X}^*)}{x_j^p} \cdot b_{p,0,j,i}(\mathbf{X}^*) \end{aligned} \tag{4.33}$$

We emphasize that all coefficients are those with respect to $\mathbf{X}^*$.

In addition, for a solution $\mathbf{X}'$ dominating $\mathbf{X}^*$, we define the following according to the definition of the ELR operation:

$$\Psi'_p(\mathbf{X}', i) = \sum_{q \geq 0} \sum_{j=1, j \neq i}^{n} \left(\frac{a^L_{q,p,j,i}}{x_j^{*q}}\right) \cdot b^L_{q,p,j,i} \tag{4.34}$$

$$\Phi'_p(\mathbf{X}', i) = \sum_{q \geq 0} \sum_{j=1, j \neq i}^{n} a^U_{p,q,i,j} \cdot (b^U_{p,q,i,j} \cdot x_j^q) \tag{4.35}$$

$$\Theta'(\mathbf{X}', i) = \sum_{q \geq 0} \sum_{r \geq 0} \sum_{j=1, j \neq i}^{n} \sum_{k=1, k \neq j \neq i}^{n} \left(\frac{a^L_{q,r,j,k}}{x_j^q}\right) \cdot (b^U_{q,r,j,k} \cdot x_k^r)$$

$$+ \sum_{q\geq 0}\sum_{j=1,j\neq i}^{n} a_{0,q,i,j}(\mathbf{X}')\cdot(b_{0,q,i,j}(\mathbf{X}')\cdot x_j^q)$$
$$+ \sum_{p\geq 0}\sum_{j=1,j\neq i}^{n} \frac{a_{p,0,j,i}(\mathbf{X}')}{x_j^p}\cdot b_{p,0,j,i}(\mathbf{X}') \tag{4.36}$$

One can easily verify that

$$\Psi_p(\mathbf{X}^*,i) \geq \Psi'_p(\mathbf{X}',i) \tag{4.37}$$
$$\Phi_p(\mathbf{X}^*,i) \leq \Phi'_p(\mathbf{X}',i) \tag{4.38}$$

Let $\tilde{x}_i$ be the extended local optimum for x_i with respect to $\mathbf{X}'$. We have:

$$\sum_{p>0}(\Psi'_p(\mathbf{X}',i)\cdot\tilde{x}_i^p + \Phi'_p(\mathbf{X}',i)\cdot\frac{1}{\tilde{x}_i^p}) + \Theta'(\mathbf{X}',i)$$
$$\leq \sum_{p>0}(\Psi'_p(\mathbf{X}',i)\cdot x_i^{*p} + \Phi'_p(\mathbf{X}',i)\cdot\frac{1}{x_i^{*p}}) + \Theta'(\mathbf{X}',i) \tag{4.39}$$

where x_i^* is the value for x_i in the exact solution $\mathbf{X}^*$.

Because x_i^* must be the *local optimum* of x_i with respect to $\mathbf{X}^*$, and Eqn. (4.31)-(4.33) are defined with respect to $\mathbf{X}^*$, we have:

$$\sum_{p>0}(\Psi_p(\mathbf{X}^*,i)\cdot x_i^{*p} + \Phi_p(\mathbf{X}^*,i)\cdot\frac{1}{x_i^{*p}}) + \Theta(\mathbf{X}^*,i)$$
$$\leq \sum_{p>0}(\Psi_p(\mathbf{X}^*,i)\cdot\tilde{x}_i^p + \Phi_p(\mathbf{X}^*,i)\cdot\frac{1}{\tilde{x}_i^p}) + \Theta(\mathbf{X}^*,i) \tag{4.40}$$

Summing up (4.39) and (4.40), we obtain:

$$\sum_{p>0}\{\Psi'_p(\mathbf{X}',i) - \Psi_p(\mathbf{X}^*,i)\}\cdot(\tilde{x}_i^p - x_i^{*p})$$
$$+ \sum_{p>0}\{\Phi'_p(\mathbf{X}',i) - \Phi_p(\mathbf{X}^*,i)\}\cdot(\frac{1}{\tilde{x}_i^p} - \frac{1}{x_i^{*p}})$$
$$= \sum_{p>0}\{\Psi'_p(\mathbf{X}',i) - \Psi_p(\mathbf{X}^*,i) + \frac{\Phi'_p(\mathbf{X}^*,i) - \Phi_p(\mathbf{X}',i)}{\tilde{x}_i^p\cdot x_i^{*p}}\}\cdot(\tilde{x}_i^p - x_i^{*p})$$
$$\leq 0 \tag{4.41}$$

Because $\mathbf{X}^*$ is dominated by $\mathbf{X}'$, using Eqn. (4.37) and (4.38), we know that $\tilde{x}_i^p - x_i^{*p} \geq 0$, and therefore $\tilde{x}_i - x_i^* \geq 0$, i.e. $\mathbf{X}^*$ is still dominated by the extended local refinement of $\mathbf{X}'$.

The case when a solution $\mathbf{X}'$ is dominated by $\mathbf{X}^*$ can be proved symmetrically. □

Note that the simple CH-program is a subset of the monotonically constrained CH-program, and the PLR operation is same as the LR operation in the case of simple CH-program, Theorem 2 also shows that the dominance property holds under the LR operation for the simple CH-program.

4.3 Local-Refinement based Algorithm

Again, let $\mathbf{X}^*$ be an exact solution to a CH-program. We say that a solution $\mathbf{X}$ is the lower bound of $\mathbf{X}^*$ if $\mathbf{X}$ is dominated by $\mathbf{X}^*$, and $\mathbf{X}$ is an upper bound of $\mathbf{X}^*$ if $\mathbf{X}$ dominates $\mathbf{X}^*$. Theorems 1 and 2 enable an algorithm based on different types of LR operations to compute a set of lower and upper bounds for $\mathbf{X}^*$.

Bound-Computation Algorithm
1. Initialize lower and upper bounds;
2. If lower and upper bounds do not meet
3. Perform ELR operation on every x_i of the lower bound iteratively;
4. Perform ELR operation on every x_i of the upper bound iteratively;
5. Goto 2 if there is any improvement in 3 and 4;
6. Return ELR-tight lower and upper bounds.

Table 4.1: Bound-computation algorithm using the ELR operation

Because the bounded CH-program is the most general case, we use the ELR

operation to illustrate the bound-computation algorithm (see Table 4.1). Starting with the initial lower and upper bounds ($\mathbf{L}$ and $\mathbf{U}$), the algorithm carries out *interleaved* passes of lower- and upper-bound computations. A *pass* of lower-bound computation will perform an ELR operation on every x_i of a lower bound $\mathbf{X}$ in an *arbitrary* order. Because $\mathbf{X}$ is dominated by $\mathbf{X}^*$, its extended-local refinement becomes closer to $\mathbf{X}^*$ but is still a lower bound. Similarly, a pass of upper bound computation will perform an ELR operation on every x_i of an upper bound $\mathbf{X}$. The iteration of passes is stopped when the lower and upper bounds meet for every x_i, or both bounds are ELR-tight. We say that a lower or upper bound is *ELR-tight* if it can not be improved by any ELR operation.[7] Although the ELR operation may use any valid lower and upper bounds for coefficients according to Definition 11, in general, the closer the lower and upper bounds for coefficients, the smaller the gap between the resulting ELR-tight lower and upper bounds. Because reducing the size of the solution space may narrow the range for coefficients, lower- and upper-bound computations are carried out alternately. The algorithm guarantees that within the resulting ELR-tight lower and upper bounds, there would exist an exact solution to the bounded CH-program.

For a simple or monotonically constrained CH-program, we may replace the ELR operation in Table 4.1 by the LR or PLR operation, respectively. Then, the algorithm computes the LR-tight or PLR-tight lower and upper bounds, where a lower or upper bound of an exact solution is *LR-tight* or *PLR-tight* if it can not be improved by any LR or PLR operation. In essence, the bound-computation algorithm generalizes the greedy wiresizing algorithm GWSA that

[7] Even though the lower and upper bounds are ELR-tight, there may still be a gap between them. We say that the computation for a variable x_i is *convergent* if its lower and upper bounds are identical. The ELR operation does not guarantee the convergence for all variables. We define the *convergence rate* as the percent of variables that has identical lower and upper bounds. Both *average gap* among all variables and convergence rate will be presented for our experiments later on in Sections 5.3 and 6.3.

has been used for computing LR-tight lower and upper bounds for the exact wire sizing solution under fixed c_a and c_f in [29, 16]. When the exact solution has the monotone property like those for the single-source and multi-source wire sizing problems (see Chapter 2), the bundled refinement operation, which is also called the bundled-LR (*BLR*) operation [14], can be used to speed up the LR, PLR or ELR operation. We also use the LR-based algorithm to refer to the bound-computation algorithm, where LR, in general, refers to the LR, PLR, ELR and BLR operations.

The LR-based algorithm has the same worst-case complexity when using different types of LR operations. Let r be the average number of the possible values for variables $x_i (i = \{1, \cdots, n\}) \in \mathbf{X}$ when all variables x_i have discrete values. Because each pass of the lower- and upper-bound computation at least changes the value of one variable to narrow the solution space by at least one unit, the worst-case number of passes is $\Theta(r \cdot n)$. In addition, each pass has at most $2n$ LR operations. Therefore, the bound-computation algorithm needs $\Theta(r \cdot n^2)$ LR operations. We observed in our experiments that the total number of LR operations is much smaller than $\Theta(r \cdot n^2)$ and is *empirically* linear with respect to the number of variables.

4.4 Comparison with Posynomial Program

In order to better appreciate the implications of Theorems 8 and 9, we compare the CH-programs with the posynomial program (defined in Footnote 6). When every variable is of *continuous* value, the posynomial program has the important property that the local optimum is unique, and therefore is also the global optimum. The posynomial program plays an important role in the device and wire sizing works. In [38], the transistor sizing problem was first formulated as

a posynomial program and solved by a sensitivity-based method. Later on, the posynomial program formulation was used for transistor sizing [75], wire sizing [74] and simultaneous gate and wire sizing [55], and was solved by being transformed into the convex program.[8] Note that optimality of these solutions depends on the assumption that the local optimum is unique. The assumption holds for the continuous sizing formulation and simple models for the interconnect capacitance and device delay, but may be not true for the discrete sizing formulation and more general models for the interconnect capacitance and device delay.

Our LR-based algorithm is similar to the coordinate descent approach [53] for the posynomial program. The approach iteratively optimizes the value for each variable (i.e., coordinate) while keeping the values for the rest of the variables fixed.[9] Because the local optimum is unique for the posynomial program regarding continuous variables, one may even start with an *arbitrary* solution (see [13]) rather than a lower or upper bound used in the LR-based algorithm. However, when the variables $x_1, x_2, \cdots, x_n$ are of discrete values for the simple CH-program, or when the coefficients are not constants as in the monotonically constrained or bounded CH-program (for both continuous or discrete variables), there may be more than one local optimum.[10] Then, the global optimum can not be achieved by the coordinate descent approach starting from an arbitrary solution. However, the LR-based algorithm, which respectively uses the LR, PLR or ELR operations

[8] Same as the method in [55], methods in [75, 74] minimize the maximum delay.

[9] An alternative method, called the steepest descent approach or the gradient method [53], minimizes the objective function along the direction of the steepest gradient, and may simultaneously change all coordinates. In general, it is $n-1$ times faster than the coordinate descent approach, where n is again the number of variables [53]. However, because of the special nature of the sizing problems, the LR-based optimization (the coordinate descent approach) turns out to be very efficient in experiments. In fact, it was recently shown that when using the simple device and capacitance models, the LR-based algorithm can be finished in a linear time for the continuous wire sizing problem [13].

[10] The simple CH-program using continuous variables belongs to the posynomial program, and therefore has a unique local optimum.

for a simple, monotonically constrained or bounded CH-program, can still be used to compute lower and upper bounds for the exact (i.e., *globally* optimal) solution. In the following, we will apply the ELR operation to the simultaneous transistor and interconnect sizing problem under the table-based device model, and apply the PLR and ELR operations in Chapter 6 to the global interconnect sizing and spacing (*GISS*) problem considering the coupling capacitance for multiple nets. Both problems are no longer the simple CH-program, and may have multiple local optimum solutions.

4.5 Conclusions and Discussions

In this chapter we formulated three classes of optimization problems: the simple, monotonically constrained, and bounded CH-programs. We revealed the dominance property (Theorem 8) under the local refinement (*LR*) operation for the simple CH-program, as well as the general dominance property (Theorem 9) under the pseudo-LR (*PLR*) operation for the monotonically constrained CH-program and under the extended-LR (*ELR*) operation for the bounded CH-program. These properties enable a very efficient polynomial-time algorithm, using the LR, PLR, or ELR operation for computing lower and upper bounds of the exact solution to any CH-program. In addition, we introduced in Chapter 2 the bundled-LR (*BLR*) operation, which may be used to speed up the LR, PLR and ELR operations. We also called the bound-computation algorithm as the LR-based algorithm, where LR, in general, refers to the LR, PLR, ELR or BLR operation.

The algorithm can be used to unify solutions to several problems, including the single-source and multi-source wire sizing problems [29, 16], continuous wire sizing problem [8], and simultaneous driver/buffer and wire sizing problem [26, 6, 27].

These problems assume the simple models for the device delay and interconnect capacitance, and are all simple CH-program where the LR operation can be used for bound computations. We will apply the formulations and algorithm of CH-programs, especially those for the bounded CH-program, to solve the much more general simultaneous device sizing, and wire sizing and spacing problems in Chapters 5 and 6. Models that are much more accurate than the simple models will be used for device delay and interconnect capacitance.

CHAPTER 5

Application of Local-Refinement based Optimization to Simultaneous Device and Interconnect Sizing

To minimize interconnect delay in DSM designs, many optimization techniques have been proposed to including interconnect topology optimization, buffer insertion, device sizing, and wire sizing studied in the previous chapter. We believe that the most effective approach to performance optimization in DSM designs is to consider both logic and interconnect designs throughout the entire design process (from RTL level to layout design). This motivates our study of the simultaneous device and interconnect sizing problem for DSM designs.

Several recent studies considered the simultaneous device and interconnect sizing problem. One class of algorithms minimizes the weighted delay. In [26], the simultaneous driver and wire sizing problem was formulated to minimize the weighted delay between the source and a set of sinks for a single net. Procedures of device sizing and wire sizing are alternately carried out, with device sizes computed by closed-form formulas (via Maple) and wire widths computed by algorithms from [29, 16]. In [18, 20], the simultaneous transistor and interconnect sizing problem was studied to minimize the weighted delay for multiple paths (a path contains multiple nets). The local refinement operation, previously used *only* for wire sizing solutions [29, 26, 16], is applied to optimize both devices and

interconnects. It leads to a unified and very efficient algorithm. Recently, the simultaneous buffer insertion and wire sizing problem was also addressed [12]. It is assumed that the number of buffers to insert is given for each wire segment, and that the wire widths between any two buffers are monotonic. Therefore, the problem can be solved as a convex quadratic program to find the lengths of wire segments for different wire widths.

The other class of simultaneous device and interconnect sizing algorithms considers the maximum delay. In [55], the simultaneous gate and wire sizing problem was formulated to minimize the area under the maximum-delay constraint for multiple paths. The problem is shown to be a posynomial program, and is transformed into a convex program solved by a sequential quadratic programming technique. In addition, the simultaneous buffer insertion and wire sizing problem was studied to minimize the maximum delay from the source to a set of sinks for a single net [49]. The potential locations for buffer insertion are *a priori* given. Based on a bottom-up dynamic programming approach, buffers are then inserted with optimal sizes, and optimal wire widths determined simultaneously. In general, the algorithms for minimizing the weighted delay are more efficient. By adjusting the weight assignments, a sequence of such minimizations can be used to minimize the maximum delay under the area constraint or to minimize the area under the delay constraint. In particular, a Lagrangian relaxation technique was proposed in [6] to optimally assign the weights for the sequence of weighted-delay minimizations. The simultaneous buffer and wire sizing problem was also solved [6].

However, most of these works assumed over-simplified models for device delay and interconnect capacitance. Those assumptions are no longer realistic for DSM designs. Accurate models will be used to solve the simultaneous device and wire

sizing problem in this chapter and Chapter 6.

Our contributions in this chapter include:

- We assume that the circuit netlist is given and routing topology is fixed, and then formulate the STIS problem to assign optimal wire widths to all wire segments and optimal sizes to all transistors, for minimizing the delay and/or power for multiple critical paths. Compared to the single-net sizing problems such as the MSWS problem, this problem has much higher complexity but is able to achieve more delay and power reduction.

- We apply the LR-based bound-computation algorithm presented in Chapter 4 to the STIS problem under both the simple device model and the accurate STL-bounded device model. The STL-bounded model is based on tables pre-computed from SPICE simulations for the device delay, so that it is much more accurate than many models used in previous device and interconnect optimization algorithms. Experiments show that the bound-computation algorithm can efficiently handle both simple and STL-bounded models, and obtain solutions close to the global optimum in both cases. According to SPICE simulations, the solution obtained by the STIS algorithm under the simple model achieves up to 14.4% delay reduction when compared to the solution given by manual optimization (reported in [10]). Furthermore, the solution obtained by the STIS algorithm under the accurate STL-model achieves up to 15.1% additional delay reduction when compared to the solution obtained by the STIS algorithm under the simple model.

The rest of the chapter is organized as follows: We describe the STIS problem formulation in Section 5.1, and apply the LR-based algorithm to solving the

STIS problem, for both simple and STL-bounded models in Section 5.2. We finally present experimental results in Section 5.3, and conclude the chapter in Section 5.4. Results of this chapter have been presented in conference papers [18, 20, 19], and the journal paper [25].

5.1 Formulation and Solution of STIS Problem

5.1.1 Device and Interconnect Models

5.1.1.1 Simple Model

Almost all simultaneous device and wire sizing works [26, 55, 49] assumed simple gates like inverters and buffers, and used gate sizing formulation where an optimal size is found for each gate. We will use transistor sizing formulation where an optimal size is found for each transistor, in order to consider complex gates and to achieve better results when compared with the gate sizing formulation.

In our formulation, we partition all transistors and interconnects into DCCs. A DCC is defined as a set of transistors and wires which are connected by DC-connected paths containing only transistor channels or wires, and the DC current can not cross the boundary of a DCC. In most cases, a DCC comprises a logic gate G and a routing tree connecting the output of G to the inputs of all logic gates driven by G. However, in cases like simultaneous driver and wire sizing for multi-source nets, a DCC contains last-stage drivers at all sources and the routing tree.

We model a transistor as an idle switch connected to an effective resistor, similar to the switch-level timing tool [61]. For example, a transistor of size d and output load c_l is assumed to have a delay $t_d = t_0 + r_d \cdot c_l$, where t_0 and r_d

are the *intrinsic delay* and *effective resistance* of the transistor, respectively. In addition, $r_d = r_0/d$, where r_0 is the *unit-size effective-resistance* for the transistor. The *simple model* used in most works assumes that t_0 and r_0 are *constants*.

In addition to effective resistance r_d, the gate, source and drain capacitances c_g, c_s and c_d are also needed to characterize a transistor of size x. we have:

$$r_d = r_{d0}/x \tag{5.1}$$

$$c_g = c_{g0} \cdot x + c_{g1} \tag{5.2}$$

$$c_s = c_{s0} \cdot x + c_{s1} \tag{5.3}$$

$$c_d = c_{d0} \cdot x + c_{d1} \tag{5.4}$$

Again, the simple model assumes that c_{g0}, c_{s0} and c_{d0}, as well as c_{g1}, c_{s1} and c_{d1} are constants determined by the technology.

Overall, the logic gate is characterized by a RC network for the transistor sizing formulation. Note that a gate can be characterized by a "macro" transistor connected to Vdd in the gate sizing formulation with one size for each gate like [26, 55, 49]. It is a simple case of the RC network for the transistor sizing formulation. Therefore, this switch-level transistor model is able to consider both transistor sizing and gate sizing formulations.

We model a routing tree as a distributed RC tree. Similar to wiresizing work [16], each wire segment is divided into a sequence of uni-segments. A uni-segment is treated as a π-type RC circuit and the wire width is assumed uniform within a uni-segment. The segmentation controls how aggressively we perform wiresizing optimization. For simplicity, we assume that all uni-segments have the same wire length and the unit-width uni-segment has wire resistance r_0, wire area capacitance c_a and wire fringing capacitance c_f. Then, the resistance r and the

capacitance c for a uni-segment with width x are

$$r = r_0/x \tag{5.5}$$

$$c = c_a \cdot x + c_f \tag{5.6}$$

Again, the simple model used in most works assume that r_0, c_a and c_f are constants.

5.1.1.2 More Accurate Model

Most existing device and interconnect sizing works assume the simple model for the device delay and interconnect capacitance, where r_0 is a constant for the device delay, and both c_a and c_f are constants for the interconnect capacitance. The simple model is no longer realistic for DSM designs. For example, we computed r_0 for an inverter in Table 5.1. We apply HSPICE simulations, and use device parameters for the $0.18\mu m$ technology in Table 5 of the 1994 National Technology Roadmap for Semiconductors (NTRS'94) [77]. When the inverter is driven by a rising input, we first measure two delay values t_1 and t_2 for a pair of output loads c_1 and c_2 under the same size and input switching time. Using the assumption that $t_1 = t_0 + r_d \cdot c_1$ and $t_2 = t_0 + r_d \cdot c_2$, we can obtain $r_d = (t_1 - t_2)/(c_1 - c_2)$, and $t_0 = t_1 - r_d \cdot c_1$. We then compute t_0 values for different combinations of size, input switching time (t_s) and output load (c_l). Because we assume that the intrinsic delay t_0 is a constant in this paper, we derive the "best" t_0 value by least-square-fitting over t_0 values for different combinations of size, t_s and c_l. Finally, we use the "best" t_0 value to compute $r_0 = (t_d - t_0)/c_l \cdot d$, where t_d is the inverter delay, and d the size for the n-transistor in the inverter. We compute r_0 for the n-transistor under different combinations of size, t_s and c_l. Similarly, when the inverter is driven by a falling input, r_0 for the p-transistor can be determined in the same way under different combinations of size, t_s and c_l. As one can see

from Table 5.1, r_0 is clearly *not* a constant. Its value may vary by a factor of 2.

size = 100x						
	n-transistor			p-transistor		
c_l / t_s	0.05ns	0.1ns	0.2ns	0.05ns	0.1ns	0.2ns
0.225pF	12200	13370	19180	17200	19920	24550
0.425pF	8135	9719	12500	17180	17190	18820
0.825pF	8124	8665	10250	17090	17150	17290
1.625pF	8114	8170	8707	16140	17140	17150
3.225pF	7578	8137	8251	14710	16940	17100
size = 400x						
	n-transistor			p-transistor		
c_l / t_s	0.05ns	0.1ns	0.2ns	0.05ns	0.1ns	0.2ns
0.501pF	12200	15550	19150	18200	19970	27030
0.901pF	11560	13360	17440	17340	19590	24560
1.701pF	8463	9688	12470	17070	17420	18790
3.301pF	7725	8812	10420	17030	16780	17440
4.901pF	7554	8480	10010	16090	17020	17060

Table 5.1: Unit-size effective-resistance for n- and p-transistor

Little progress has been made for optimization beyond the simple device model. The simultaneous buffer insertion and wire sizing algorithm [49] was extended to consider the impact of the input switching time for the device delay. The unit-size effective-resistance, in essence, is assumed to be $r_0 = r_0' + \delta \cdot t_s$, where r_0' is the unit-size effective-resistance under the step input, t_s the input switching time, and δ an empirical constant. The algorithm based on the bottom-up dynamic-programming, however, no longer has a polynomial-time complexity under the extended device model. The posynomial program formulation for the simultaneous gate and wire sizing problem [55] was also extended to accommodate a voltage-ramp gate model, which considers the impacts of the input switching

time and output loading under the C_{eff} model [63]. The resulting sizing problem, however, is no longer a posynomial program. It is unknown how far away the solution obtained by solving a posynomial program is from the exact solution under the voltage-ramp model. Furthermore, runtime of the two works is pretty high.

We will call the device table, like Table 5.1, *STL-bounded* model, where r_0 is determined by the size, input switching time (t_s) and output load (c_l), and its value is *bounded* (i.e., there exist lower and upper bounds for r_0) for any given ranges of size, t_s and c_l. We build tables for the STL-bounded device model via HSPICE simulations. These models are more accurate than the simple model, and have been widely used for verification purposes. However, there are virtually no existing algorithms that allow us to use these models for the device and interconnect sizing problems. In the following of this chapter, the device and interconnect sizing algorithm will be developed using the STL-bounded device model.

For DSM designs, the coupling capacitance between neighboring wires becomes the dominant capacitance component. Because the coupling capacitance depends strongly on spacing, the simple capacitance model given in Eqn. (5.6) is no longer valid, and it is needed to study the simultaneous interconnect sizing and spacing problem, rather than interconnect sizing only, for the maximum delay reduction. We will extend the STIS formulation and algorithm to consider the coupling capacitance and simultaneous interconnect sizing and spacing formation in Chapter 6.

5.1.2 Problem Formulation

Our delay formulation is similar to those in [61], and is based on the Elmore delay formulation in [65] (see [20] for details). The delay is computed first for each a *stage*. It is defined as a DC-connected path from a power supply (either the Vdd or the ground) to the gate node of a transistor, containing both transistors and wires. The delay of a stage $P(N_s, N_t)$ with N_s being the source and N_t being the sink can be written as Eqn. (5.7) under the Elmore delay model.

$$\begin{aligned} & t(P(N_s, N_t), \mathbf{X}) \\ = & \sum_{i,j} f(i,j) \cdot \frac{r_0(i)}{x_i} \cdot c_a(j) \cdot x_j + \sum_{i,j} f(i,j) \cdot \frac{r_0(i)}{x_i} \cdot c_f(j) \\ + & \sum_i g(i) \cdot \frac{r_0(i)}{x_i} + \sum_i r_0(i) \cdot h(i) + \sum_i h(i) \cdot \frac{r_0(i)}{x_i} \cdot c_f(i) \end{aligned} \tag{5.7}$$

where x_i is the width for a transistor M_i or a wire E_i, $r_0(i)$ is its unit-size effective-resistance, and $c_a(i)$ and $c_f(i)$ are its unit-area capacitance and unit-length fringe capacitance. Coefficients $f(i,j), g(i)$ and $h(i)$ are determined by the transistor netlist and routing topology as the following:

$$f(i,j) = \begin{cases} 1 & if\ M_i/E_i\ charges\ M_j/E_j \\ 0 & otherwise \end{cases}$$

$$g(i) = \begin{cases} \sum c^l & if\ M_i/E_i\ charges\ loading\ capacitance\ c^l \\ 0 & otherwise \end{cases}$$

$$h(i) = \begin{cases} 1/2 & if\ M_i/E_i\ \in\ P(N_s, N_t) \\ 0 & otherwise \end{cases}$$

In order to simultaneously minimize delays along multiple critical paths, we then minimize the weighted delay $t(\mathbf{X})$ of all stages in the set of critical paths denoted

as $\mathcal{P}$:

$$t(\mathbf{X}) = \sum_{P(N_s,N_t)\in\mathcal{P}} \lambda_{st} \cdot t(P(N_s,N_t),\mathbf{X}) \tag{5.8}$$

where the weight λ_{st} indicates the criticality of stage $P(N_s,N_t)$. After we eliminate those terms independent of $\mathbf{X}$, Eqn. (5.8) can be re-written as

$$\begin{aligned} & t(\mathbf{X}) \\ = & \sum_{i,j} F(i,j) \cdot \frac{r_0(i)}{x_i} \cdot c_a(j) \cdot x_j \\ + & \sum_{i,j} F(i,j) \cdot \frac{r_0(i)}{x_i} \cdot c_f(j) \\ + & \sum_i G(i) \cdot \frac{r_0(i)}{x_i} + \sum_i H(i) \cdot \frac{r_0(i)}{x_i} \cdot c_f(i) \end{aligned} \tag{5.9}$$

where $F(i,j), G(i)$ and $H(i)$ are weighted functions of $f(i,j), g(i)$ and $h(i)$, respectively.

We formulate the following simultaneous transistor and interconnect sizing (STIS) problem:

Formulation 3 *Given the lower and upper bounds (*$\mathbf{L}$ *and* $\mathbf{U}$*) for the width of each transistor and wire, the STIS problem is to determine a width for each transistor and wire (or equivalently, a sizing solution* $\mathbf{X}$*,* $\mathbf{L} \leq \mathbf{X} \leq \mathbf{U}$*) such that the weighted delay through multiple critical paths given by Eqn. (5.9) is minimized.*

Note that a sequence of weighted-delay minimization can be used to minimize the maximum delay by adjusting the weight assignment based on the Lagrangian-relaxation method as in [6]. Therefore, we focus on how to minimize weighted delay in this paper. In addition, we assume that the possible width is from a *discrete* width set determined by the technology. The discrete sizing problem

is more difficult than the continuous sizing problem, but is more convenient for placement and routing tools and fabrication.

5.2 STIS Algorithm

5.2.1 Bound Computation for STIS Problem

Under the simple models, r_0, c_a and c_f are constants for each wire/transistor, and Eqn. (5.9) is a simple CH-function. In this case, the STIS problem is a simple CH-program. Because the simple model is a simplified case of the STL-bounded model, and the latter is more suitable for DSM designs, we present the STIS algorithm under the STL-bounded device model. For simplicity of presentation, we assume here that c_a and c_f are constants for each wire segment, but will remove the assumption in Chapter 6.

In the STL-bounded model, r_0 is pre-computed and stored in tables (e.g., see Table 5.1) indexed by the size, input switching time (t_s), and output load (c_l). It could be very accurate depending on the table size.[1] Because the value for r_0 is bounded, it is easy to verify the following Theorem 10:

Theorem 10 *The STIS problem under the STL-bounded device model is a general CH-program.*

Note that the STL-bounded model might *not* be monotonic with respect to the sizing solution $\mathbf{X}$. Therefore, the STIS problem is unlikely a monotonically

[1] In our experiments, r_0 table for a type of gate (e.g., an inverter) considers the combinations of five different device sizes (from 1x to 800x of the minimum size), three different input switching times, and five different load capacitances. Therefore, the total table size is $5 \times 3 \times 5 \times m = 75m$, where m is the number of gate types. Satisfactory optimization results are obtained according to experiments in Section 5.3. For simplicity, we assume that c_l is the lumped capacitance in this paper. Extension to the effective capacitance model [63] is ongoing work and will be discussed briefly in Section 5.4.

constrained CH-program, and the LR and PLR operations are not applicable. It can be justified by the following observations: r_0 in our model is a monotonic function of t_s, whereas t_s is *not* monotonic with respect to $\mathbf{X}$, because the optimal wire sizing solution (see [29, 74, 16]) to minimize t_s often has neither minimum nor maximum wire width.

Therefore, the ELR operation is needed in the LR-based algorithm (Table 4.1) to compute lower and upper bounds for an exact solution to the STIS problem. We assume that $r_0(i) \in [r_0^L(i), r_0^U(i)]$ and $r_0(j) \in [r_0^L(j), r_0^U(j)]$. In an ELR operation on a transistor M_i for the lower-bound computation, we use $r_0^L(i)$ instead of $r_0(i)$, and $r_0^U(j)$ instead of $r_0(j)$ for M_j, where M_j is an upstream transistor in the same net for M_i. Symmetrically, in an ELR operation on M_i for the upper-bound computation, we use $r_0^U(i)$ instead of $r_0(i)$ for M_i, and $r_0^L(j)$ instead of $r_0(j)$ for an upstream transistor M_j.

We determine $r_0^L(i)$ as follows: Let $\mathbf{X}^L$ and $\mathbf{X}^U$ be lower and upper bounds of the exact solution $\mathbf{X}^*$. We assume that transistor M_i has size $x_i \in [x_i^L, x_i^U]$, input switching time $t_s(i) \in [t_s^L(i), t_s^U(i)]$, and capacitance load $c_l(i) \in [c_l^L(i), c_l^U(i)]$. We often observe in our experiments that $r_0(i)$ increases with respect to an increase of x_i or $t_s(i)$, but decreases with respect to an increase of $c_l(i)$. Therefore, $r_0^L(i)$ for M_i can be obtained by table lookup using x_i^L, $t_s^L(i)$ and c_l^U(i). Symmetrically, $r_0^U(i)$ is determined using x_i^U, $t_s^U(i)$ and $c_l^L(i)$. In addition, contributions of transistors or wires to $c_l^U(i)$ are computed using sizes in $\mathbf{X}^U$, and contributions to $c_l^L(i)$ computed using sizes in $\mathbf{X}^L$. After the ELR operation on M_i, for every stage $P(N_i, N_j)$ (N_i is the source, N_j is the sink) driven by M_i, we will update the lower and upper bounds for the switching time $t_s(j)$ at sink N_j, because $t_s(j)$ is the input switching time for the transistor M_j with gate connected to node N_j. The lower or upper bound of $t_s(j)$ is assumed to be the lower or upper bound of

the delay through $P(N_i, N_j)$, respectively. As $\mathbf{X}^L$ and $\mathbf{X}^U$ move closer during the ELR-based optimization procedure, the range of r_0 is also narrowed. In general, the closer the values for r_0^U and r_0^L, the smaller the gap between the lower and upper bounds given by the ELR operations.

Because the unit-size resistance $r_0(i)$ is a constant for each wire segment E_i, we can simply use the LR operation for E_i. Furthermore, in order to achieve better wire sizing solutions, we can divide a wire segment into a sequence of uni-segments, then find a wire width for each uni-segment [16]. We assume that each segment always stays in the same layer, has the fixed r_0, c_a and c_f, as well as same allowable wire widths.[2] With these assumptions, we have proved the following *local monotone property*:

Theorem 11 *(***local monotone property***) There exists an optimal STIS solution where the wire widths for uni-segments are monotonic within each wire segment.*

The proof is available from the technical report [17]. This theorem enables us to use the BLR operation [16] instead of the LR operation for each wire segment E_i. The BLR operation is shown to be 100x faster than the LR operation for the wiresizing problem [16].

5.2.2 Overall Algorithm for STIS Problem

Let $\mathbf{L}'$ and $\mathbf{U}'$ be the ELR-tight lower and upper bounds given by the above bound-computation procedure. If $\mathbf{L}'$ and $\mathbf{U}'$ are identical, we obtain the exact solution to the STIS problem under the STL-bounded model. Otherwise, we

[2] Different segments may have different r_0, c_a and c_f if they are in different layers, or have different spacings to neighboring wires.

traverse all wire segments and transistors by iterative PLR operations until there is no improvement in the last round of traversal. Note that the PLR operation is bounded by $\mathbf{L}'$ and $\mathbf{U}'$, and it uses r_0 obtained from the device table. Even though the PLR operation may lead to further improvement over $\mathbf{L}'$ and $\mathbf{U}'$, in general it does *not* lead to a lower or upper bound of the exact solution.[3]

Our experiments in Section 5.3.4 show that the ELR-tight lower and upper bounds ($\mathbf{L}'$ and $\mathbf{U}'$) are often close to each other in most cases. Therefore, we can simply treat $\mathbf{L}'$ as the final solution (for smaller area and often lower power-dissipation). We will also show in the experiments that the STIS problem to minimize a weighted-sum of delay and area is a CH-program. Therefore, a trade-off can be obtained between delay and area. A similar approach can be used to better minimize the capacitive power by minimizing the weighted-sum of delay and capacitive power.

5.3 Experimental Results

For all experiments in this paper, we computed the delays via HSPICE using the distribute RC model and the level-3 MOSFET model that is also used in HSPICE simulations for device-table generation. The use of HSPICE simulation results not only shows the quality of our sizing solutions, but also verifies the validity of our interconnect and device modeling, and the correctness of our problem formulations.

[3]In our experiments, we tried to use PLR operations starting from either the minimum or maximum sizing solution. The resulting solutions are often outside the range defined by $\mathbf{L}'$ and $\mathbf{U}'$, and are worse than $\mathbf{L}'$.

5.3.1 Area-Delay Trade-off for Transistor Sizing

Since the algorithm based the local refinement operation was used only for the optimal wiresizing problem [29, 26, 14] in the past, we first use the STIS algorithm to solve the transistor sizing problem to show that the local refinement operation is also applicable to transistor sizing problem. We study the transistor sizing problem for area-delay trade-off under the following objective function:

$$obj(\mathbf{X}, \gamma) = \gamma \cdot \frac{\sum_{x_i \in \mathbf{X}} x_i}{\sum_{x_i \in \mathbf{L}} x_i} + (1 - \gamma) \cdot \frac{t(\mathbf{X})}{t(\mathbf{X} = \mathbf{L})} \tag{5.10}$$

It is the scaled weighted sum of area and delays, and γ ($0 \leq \gamma \leq 1$) can be adjusted for area-delay trade-off. One can easily verify that Eqn. (5.10) is still a CH-posynomial. Again, we can apply the STIS algorithm efficiently.

We sized 8bit, 16bit and 32 bit ripple-adders, respectively, under the simple step model. We report total device areas and maximum delays by HSPICE simulation in Table 5.2. We use parameters of MCNC $0.5\mu m$ CMOS technology [54], same as those used in Chapter 3, and assume that each primary input to these adders comes from a 2x inverter and each primary output drives a 2x inverter. The STIS algorithm optimizes the 32bit adder containing 1,026 transistors in 54 seconds. A smooth delay-area trade-off is observed and the maximum delay is reduced by up to 30.8% with about 2x times area when compared with the minimum-size design. These adders are implemented in CMOS complex gates, and we simply assume that every transistor has the same timing criticality and the same weight penalty. We plan to use the Lagrangian relaxation method [6] to obtain the optimal weight penalty assignment and study the impact of weight penalty assignment. Besides, the area-delay trade-off formulation can be easily extended for power-delay trade-off.

	8bit adder, 258 transistors		16bit adder, 514 transistors		32bit adder, 1026 transistors	
γ	delay(ns)	area(μm^2)	delay(ns)	area(μm^2)	delay(ns)	area(μm^2)
0.00	2.558(-30.8%)	312.0	5.611(-25.3%)	623.2	11.241(-25.8%)	1245.6
0.01	2.918(-21.1%)	298.2	6.173(-17.8%)	479.2	12.324(-18.6%)	904.4
0.02	3.018(-18.4%)	241.2	6.549(-12.8%)	416.7	13.314(-12.1%)	840.9
0.10	3.597(-2.78%)	156.4	7.345(-2.3%)	310.4	14.842(-2.0%)	618.4
1.00	3.700	152.8	7.515	303.3	15.145	604.0

Table 5.2: Delay-area trade-off during transistor sizing on ripple-adders under a 0.5μm process.

5.3.2 Simultaneous Driver and Wire Sizing for Multi-source Nets

We then use the STIS algorithm to solve the simultaneous driver and wire sizing (SDWS) problem for multi-source nets.[4] These nets are extracted from an Intel high-performance microprocessor design and were used in [31, 14] for topology construction and wiresizing optimization. We assume that a chain of cascade drivers is used for each source and the first stage is a minimum-size (1x) driver. We compare our STIS method with the DS+WS method. The DS+WS method uses a constant stage ratio $\alpha = (C_L/C_0)^{1/N}$ where C_0 is the gate capacitance of the 1x driver, and C_L the total loading capacitance when the wires have the minimum width. The stage number N is chosen such that α is around e, the base of natural logarithms, for performance optimization. Then, the DS+WS method applies the OWBR algorithm [14] to obtain the optimal wiresizing solution with respect to the given driver sizing solution. The STIS method uses the same stage number N and assumes the first stage is also a 1x driver, but the sizes of both wires and transistors in other stages are determined by the STIS algorithm. Again, we use parameters of MCNC $0.5\mu m$ CMOS technology [54]. We assume the nets are driven by a clock of $20\text{M}Hz$ and report the HSPICE simulation results in Table 5.3. Even though the OWBR algorithm achieves the optimal wiresizing solutions under the given driver sizing solutions, the STIS formulation consistently outperforms DS+WS: the maximum delay is reduced by up to 17.7%, and more significantly, the total device area and total power dissipation are reduced by 57.4% and 61.7%, respectively. Although we compute the optimal width for every transistor and every $10\mu m$-long wire, the total runtime of the STIS algorithm for all nets is just 7.18 seconds. Furthermore, we point out that our STIS algorithm can be used to find even better sizing solutions by

[4] The SDWS formulation in [26] is applicable only to single-source nets.

	# of	length	max delay (ns)		device area (μm^2)		wire area (μm^2)		average power (mW)	
	drivers	(μm)	DS+WS	STIS	DS+WS	STIS	DS+WS	STIS	DS+WS	STIS
net1	15	3600	1.063	0.876(-17.7%)	563.0	188.6	4320	3240	31.9	2.17
net2	24	6600	1.064	0.928(-12.3%)	4232.4	734.8	19923	7953	31.6	7.02
net3	36	10070	1.326	1.225(-7.6%)	8773.2	3222.4	23967	19242	45.8	14.2
net4	24	10570	1.197	1.120(-6.4%)	6071.4	1331.7	40041	22725	47.9	11.4
net5	30	31980	1.868	1.808(-3.2%)	8406.8	6456.8	139410	139410	53.2	45.8
total	129	179800			28046.8	11934.3 (-57.4%)	227661	192570 (-15.4%)	210.4	80.59 (-61.7%)

Table 5.3: Driver and wire sizing for multi-source nets under a $0.5\mu m$ process. DS+WS – separate driver sizing and wire sizing, STIS – simultaneous driver and wire sizing.

trying different stage numbers, as we already observed during our experiments.

5.3.3 Comparison between Manual Optimization and STIS Algorithm

To illustrate the effectiveness of the STIS algorithm, we first compare the sizing solution obtained by our algorithm and the manual optimization applied to a spread spectrum IF transceiver chip in [10]. The design is under the 1.2 μm two-layer metal SCMOS technology. There are two clock nets, *dclk* and *clk*; each uses a chain of four cascade drivers in the clock signal source and chains of four cascade buffers in order to drive long interconnects and register files. The maximum delays of the two nets need to be minimized to reduce the clock skew. Therefore, source drivers and buffers are tuned manually via iterative procedures of layout, extraction and HSPICE simulation. We retain the manual sizing solutions for the first stage drivers at the source and for the drivers of the register files, then apply the STIS algorithm to optimize the sizes for every $10\mu m$-long wire and the rest of the drivers and buffers. We use two formulations under the simple device model, one is simultaneous transistor and wire sizing formulation (*stis/simple*) where optimal sizes are found for p- and n- transistors in each driver/buffer, and the other one is simultaneous gate and wire sizing formulation (*sgws/simple*) where an optimal size is found for each driver/buffer. We also assume that the allowable wire widths are $\{w, 2w, 3w, 4w, 5w\}$ with $w = 1.2\mu m$ being the minimum wire width in the $1.2\mu m$ technology, and the allowable transistor sizes are multiples of $0.6\mu m$ between $1.2\mu m$ and $500\mu m$. The constant value for r_0 in the simple model is determined under the typical input switching time, device size and output load. The fixed ratio between p- and n- transistors in the *sgws/simple* formulation is tuned to make sure that the inverter will have same pull-up and pull-down resistance values.

net	# of devices	wire length (μm)	max delay (ns)			average power (mW)		
			manual	sgws/simple	stis/simple	manual	sgws/simple	stis/simple
dclk	154	41518.2	4.6324	4.3447(-6.2%)	3.9635(-14.4%)	60.85	46.09(-24.3%)	46.29(-24.2%)
clk	367	59304.0	6.2016	6.1578(-0.7%)	5.9035(-4.8%)	499.7	286.8(-42.6%)	285.6(-42.8%)

Table 5.4: Comparison between manual optimization and STIS algorithms

Because the simple device model is applied, we use the LR operation to compute the LR-tight lower and upper bounds for devices. Experiments show that the identical LR-tight lower and upper bounds are achieved for almost all devices and wire segments, therefore we use the LR-tight lower bounds as the final sizing solution. We report HSPICE simulation results in Table 5.4. When compared with the manual optimization, *sgws/simple* and *stis/simple* formulations reduce the maximum delay by up to 6.2% and 14.4%, respectively. More significantly, both reduce the power consumption by 42.6% and 42.8%. Because we use the same simple model for two formulations in this experiment, the extra delay reduction (8.2%) of the *stis/simple* formulation comes from the flexibility of the transistor sizing formulation.

5.3.4 Comparison between Simple and STL-bounded Models

We then apply our STIS algorithm under different device models. We use the 0.18 μm technology given in the NTRS'94 [77] in order to study the impact of the DSM technologies. The wire sheet-resistance $R_{\square} = 0.0638\Omega$. We generate device and capacitance tables via HSPICE simulations and numerical extractions, respectively, and use c_a and c_f values where the wire is $1.10\mu m$ wide and neighboring wires are $1.65\mu m$ away[5]. We size two global nets, one is a $2cm$ line with five buffers optimally inserted for delay minimization. The other is the above *dclk* net. In addition to different device models (simple model versus STL-bounded model), we also use different sizing formulations (*sgws* versus *stis*). There are four combinations, including *sgws/simple* and *stis/simple* using the LR operation for devices, and *sgws/bounded* and *stis/bounded* using the ELR operation for devices. For simplicity, we assume that the fixed ratio between p- and n- transistors

[5]We lump the coupling capacitance into c_f, therefore it is in fact the effective-fringe capacitance.

for the gate sizing formulation is 1.0. For both nets, we find the optimal wire width for each 10 μm-long wire, and assume that allowable transistor sizes are multiples of $0.18\mu m$ between $0.18\mu m$ and $144\mu m$, and that allowable wire widths are multiples of $0.56\mu m$ between $0.56\mu m$ and 5.6 μm.

Table 5.5 summarizes experimental comparisons between different formulations. We computed convergence rate under different formulations. For the simple model, the computation for a transistor or wire is *convergent* if its LR-tight lower and upper bounds are identical. For the STL-bounded model, the computation for a transistor or wire is *convergent* if its ELR-tight lower and upper bounds are identical. The convergence is not significantly different. For example, computations for about 85% transistor are convergent in *dclk* net under all four formulations. We also computed the average width and the average gap between lower and upper bounds for all wire segments and transistors, respectively. The ELR operation does give larger gap than the LR operation. However, the difference is small. Overall, the average gap is only 1% of the average width, except that net *dclk* has a large gap, nearly 10% of the transistor size.

We simply use the ELR-tight lower bound as the final solution under the STL-bounded model, and the LR-tight lower bound as the final solution under the simple model, because lower and upper bounds given by bound computations are very close to each other. Table 5.5 also give the maximum delay via HSPICE simulation. The solutions under the STL-bounded model are consistently better than those under the simple device model. When compared with the *sgws/simple* formulation, the *sgws/bounded* formulation further reduce the maximum delay by up to 6.4%. When compared with the *stis/simple* formulation, the *stis/bounded* formulations further reduce the maximum delay by up to 15%. Note that both *sgws/simple* and *stis/simple* formulations already give very good sizing solutions

net	sgws/ simple	sgws/ bounded	stis/ simple	stis/ bounded	sgws/ simple	sgws/ bounded	stis/ simple	stis/ bounded
	convergence rate for transistors				convergence rate for wire			
dclk	85.8%	83.2%	87.7%	86.7%	99.4%	95.9%	97.1%	95.2%
line	60.0%	100%	70.0%	60.0%)	98.4%	70.9%	88.4%	72.9%
	average width / average gap (for transistors, μm)				average width / average gap (for wires, μm)			
dclk	5.39/0.07	13.0/1.91	17.2/1.53	21.6/2.36	2.50/0.003	2.78/0.025	2.69/0.017	2.82/0.030
line	108/0.108	112/0.0	126/0.97	125/1.98	4.98/0.004	4.99/0.106	5.05/0.032	5.11/0.091
	maximum delay (ns)				runtime (s)			
dclk	1.159(0%)	1.007(-6.4%)	1.132(0%)	0.961(-15%)	1.18	2.32	0.88	3.17
line	0.821(0%)	0.818(-0.4%)	0.751(0%)	0.694(-7.6%)	0.72	0.58	0.55	1.22

Table 5.5: Comparisons between different device and wire sizing formulations

as shown in the experiment of Section 5.3.3. Although ELR operations under the STL-bounded model are more complex, the runtime is still impressively small. It used just 3.17 seconds to optimize *dclk* net of 154 buffers and 41518.2μm wires, when the transistor sizing formulation is used and wire segments are 10μm long. Therefore, our STIS algorithm is extremely efficient.

5.4 Conclusions and Discussions

We have formulated the STIS problem, which assigns optimal wire widths to all wire segments and optimal sizes to all transistors, for minimizing the delay and/or power for multiple critical paths. Compared to the single-net sizing problems such as the MSWS problem in Chapter 2 and [29, 8, 26, 6], this formulation has extremely high complexity but is able to achieve more delay and power reduction.

We have applied the LR-based bound-computation algorithm presented in Chapter 4 to the STIS problem under both the simple device model and the accurate STL-bounded device model. The STL-bounded model is based on tables pre-computed from SPICE simulations for the device delay, so that it is much more accurate than many models used in previous device and interconnect optimization algorithms. We first showed that the STIS problem is a bounded CH-program, then developed the STIS algorithm based on bound-computation using the ELR operation. According to Theorem 9, our bound-computation guarantees that there exist exact solutions to the STIS problem between resulting lower and upper bounds. Experiments also showed that our algorithms obtained solutions close to the global optimum in the most cases. Based on SPICE simulations, the solution obtained by the STIS algorithm under the simple model achieves up to 14.4% delay reduction when compared to the solution given by manual optimization (reported in [10]), and the solution obtained by the STIS algorithm

under the accurate STL-model achieves up to 15.1% additional delay reduction when compared to the solution obtained by the STIS algorithm under the simple model. Moreover, the algorithms are *extremely* efficient. It took less than 10 seconds to optimize the largest example in this chapter.

For the simplicity of presentation,the simple model is used for the interconnect capacitance in this chapter. It is assumed that for a wire with width w and length l, its capacitance is given by $w \cdot c_a + c_f$ with c_a and c_f being constants. The simple capacitance model is not true for the DSM designs where the coupling capacitance can no longer be ignored. In Chapter 6, we will extend the STIS formulation to accommodate the concept of simultaneous interconnect sizing and spacing under the 2 1/2-D capacitance model presented in Chapter 6. Because the simultaneous interconnect sizing and spacing problem will be solved also by the LR-based bound computation algorithm, therefore, we have a unified formulation and solution to the simultaneous device sizing, wire sizing and spacing problem under accurate models for device delay and interconnect capacitance.

CHAPTER 6

Application of Local-Refinement based Optimization to Simultaneous Interconnect Sizing and Spacing

In DSM designs, rather than the conventional ground capacitance, the coupling capacitance between neighboring wires becomes the dominant capacitance component. Because the coupling capacitance depends strongly on spacing, it is needed to study the simultaneous interconnect sizing and spacing problem for further delay reduction when compared to wire sizing only.

In this chapter, we study the simultaneous interconnect sizing and spacing problem for the single-net and multiple nets, respectively. Our contributions include:

- We formulate the multi-net (i.e., global) interconnect sizing and spacing (*GISS*) problem based on the concept of *asymmetric wire sizing*. Given the topology for multiple nets, the problem finds the wire sizing and spacing solution optimal for *all* nets, and considers coupling capacitance extracted during wire sizing and spacing procedure.

- We pose the GISS problem as a CH-program, which directly leads to an effective and efficient solution based on bound computation using different types of LR operations. Note that this GISS formulation can be treated as

a part of the STIS formulation. Therefore, we have a unified formulation and solution to the problem of simultaneous device sizing, and wire sizing and spacing for multiple paths, under accurate models for device delay and interconnect coupling capacitance.

- We also solve the single-net interconnect sizing and spacing (*SISS*) problem. It is a simpler version of the GISS problem assuming that neighboring wires are fixed for the specific net. Experiments show that GISS algorithm may achieve up to 39% delay reduction, when compared with SISS algorithm applied iteratively to multiple nets.

Part of those results was first presented in [23, 19, 25]. An alternative approach to the GISS problem was also presented in [23], and will be covered in Mr. Zhigang Pan's thesis. These were among the first in-depth studies of the global interconnect sizing and spacing problem using accurate capacitance model.

The remainder of this chapter is organized as follows: We formulate SISS and GISS problems in Section 6.1, and present solutions to the two problems in Section 6.2. We then show experiment results in Section 6.3, and conclude this chapter in Section 6.4.

6.1 Problem Formulation

6.1.1 Introduction

In almost all existing works on wire sizing (e.g., [30, 29, 73, 82, 7, 6, 56, 83]) and simultaneous device and wire sizing (e.g., [26, 55, 49, 20, 12]), the capacitance for a wire of width w and length l is given by $c_a \cdot w \cdot l + c_f \cdot l$, where c_a and c_f are *unit-area capacitance* and *unit-length fringe capacitance* for the wire. Both

are assumed to be *constants*.

These assumptions about the interconnect capacitance are no longer realistic for DSM designs. We computed the capacitance for the *basic geometric structure* (see Figure 6.1), where the *victim* wire is centered between two neighboring wires on the same layer and both top and down grounds (two layers away from the victim). We assume that wires in the basic geometric structure have same widths, then apply a numerical capacitance extraction tool FastCap [58] to solve the structure, using interconnect geometric parameters for the $0.18\mu m$ technology in Table 22 of NTRS'94.[1] Figure 6.2(a) depicts the unit-length ground capacitance c_g between the victim and grounds, with each curve for c_g under different wire widths but a fixed edge-to-edge spacing (in short, *spacing*). If we assume $c_g = c_a \cdot w \cdot l + c_f \cdot l$, the curve slope should be c_a, and the curve intercept should be c_f. Because none of these curves is linear, and different curves have different intercepts, neither c_a nor c_f is a constant. The total capacitance of the victim is

$$c_{total} = c_g + c_x \cdot l = c_a \cdot w \cdot l + (c_f + c_x) \cdot l$$

where c_x is the *unit-length coupling capacitance* between the victim and the neighboring wires. One can define the *unit-length effective-fringe capacitance* $c_{ef} = c_f + c_x$, and compute $c_{total} = c_a \cdot w \cdot l + c_{ef} \cdot l$. We also obtained c_{ef} for different widths for the victim, under the assumption that the *center-to-edge spacing* (see Figure 6.1) from the center of the victim to the edges of its neighboring wires is fixed. As shown in Figure 6.2(b) for two different center-to-edge spacing, c_{ef} is a not a constant either. In fact, c_{ef} is very sensitive to changes of the spacing.

With consideration of coupling capacitance, we say a capacitance model is a *simple model* if both c_a and c_{ef} are constants for the interconnect capacitance.

[1] NTRS'94 gives capacitance values only for the minimum width and spacing. Our extracted capacitance values closely match those given in NTRS'94 (see [22]).

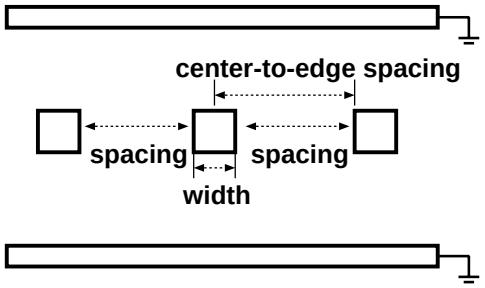


Figure 6.1: The basic geometric structure for capacitance extraction.

All aforementioned sizing works were only able to consider the simple capacitance model. Because c_x (and therefore c_{ef}) becomes the dominant capacitance component for DSM designs and depends strongly on the spacing, it is needed to study the simultaneous interconnect sizing and spacing problem, rather than wire sizing only, to take account variable c_x for the maximum delay reduction.

One very recent work [80] considered coupling capacitance for the simultaneous interconnect sizing and spacing problem for multiple nets. The problem is formulated using the concept of the dominant time constant, and is solved as a semidefinite program. There are two drawbacks however. First, the capacitance for a wire of width w and the unit-length is assumed to be simply $c_a \cdot w + c'_x/s$, where c'_x is the unit-length coupling capacitance coefficient, and s the spacing. Clearly, the capacitance model is not accurate enough. It is not clear whether the semidefinite programming is capable of handling the accurate capacitance model like the one we presented in Chapter 3. Second, the dominant time constant is an approximation to the maximum delay among multiple sinks in a net. Because there is no formula for the delay to a specific sink, it is difficult to efficiently minimize the delay of a critical sink in a net, or the delay of a critical path, where a path involves multiple nets.

In the following, we are going to solve the simultaneous interconnect sizing

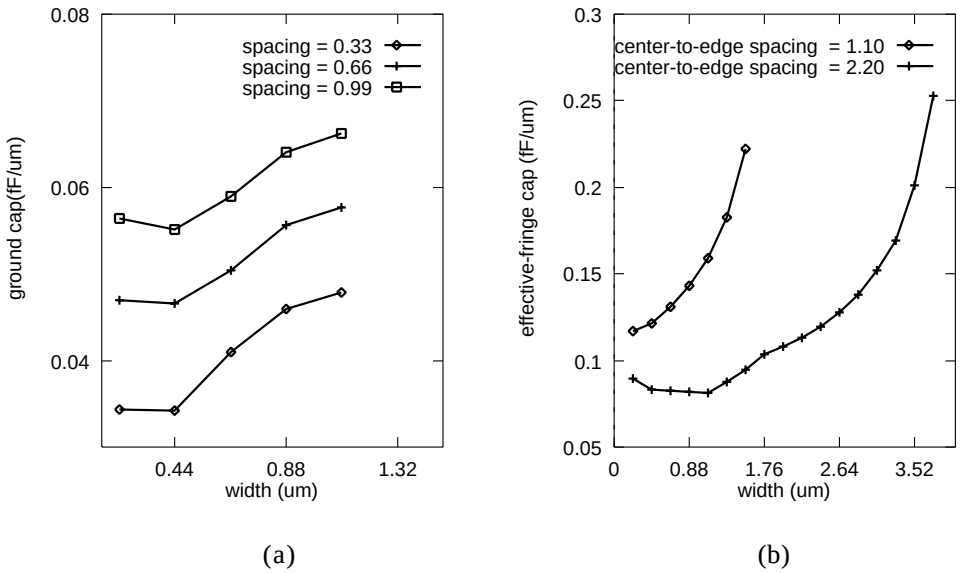


Figure 6.2: (a) Ground capacitance and (b) effective-fringe capacitance for the central wire (the victim) in the basic geometric structure shown in Figure 6.1. Each curve in (a) has the same spacing but different wire widths, and each curve in (b) has the same center-to-edge spacing but different wire widths. The capacitance values are given for the unit-length wire.

and spacing problem under the accurate table-based capacitance model presented in Chapter 3. We assume that the capacitance for a wire of width w and length l is given by

$$c_{total} = c_a \cdot w \cdot l + c_{ef} \cdot l. \tag{6.1}$$

where c_a and c_{ef} are *not* constants, but functions of widths and spacings. In addition, their values are bounded for any given ranges of widths and spacings. We will call this capacitance model as *WS-bounded* capacitance model. These models are more accurate than the simple models used in all most all existing works. We are going to present the formulations for simultaneous interconnect sizing and spacing problems using this model.

6.1.2 Single-net and Multi-net Interconnect Sizing and Spacing Problems

Similar to the STIS problem presented in Chapter 5, we still want to minimize the delay for multiple critical paths. Therefore, the weighted delay formulation Eqn. (5.9) in Section 5.1.2 is applicable here, except that we should use c_{ef} instead of c_f taking account of the coupling capacitance. With this change, the weighted delay for multiple paths are

$$\begin{aligned} & t(\mathbf{X}) \\ = & \sum_{i,j} F(i,j) \cdot \frac{r_0(i)}{x_i} \cdot c_a(j) \cdot x_j \\ + & \sum_{i,j} F(i,j) \cdot \frac{r_0(i)}{x_i} \cdot c_{ef}(j) \\ + & \sum_{i} G(i) \cdot \frac{r_0(i)}{x_i} + \sum_{i} H(i) \cdot \frac{r_0(i)}{x_i} \cdot c_{ef}(i) \end{aligned} \tag{6.2}$$

where x_i is the width for a transistor M_i or a wire E_i, $r_0(i)$ is its unit-size effective-resistance, and $c_a(i)$ and $c_{ef}(i)$ are its unit-area capacitance and unit-length

effective-fringe capacitance. During optimization, coefficients $F(i,j), G(i)$ and $H(i)$ are constants determined by the transistor netlist and routing topology, as well as weight assignment for each path. Note that for the STIS problem in Chapter 5, we assume that the unit-area capacitance c_a and unit-length effective-fringe capacitance c_{ef}, which is the sum of fringe capacitance and coupling capacitance, are constants for each wire segment. We now proceed to remove this assumption using the more general WS-bounded capacitance model. For simplicity of presentation, we assume that the device sizes are fixed, and study the simultaneous interconnect sizing and spacing problem with consideration of the coupling capacitance. However, our algorithm and implementation are able to use the STL-bounded device model and the WS-bounded capacitance model at the same time.

Our formulation for the simultaneous interconnect sizing and spacing problem was first presented in [23]. We assume that an initial layout is *a priori* given and defines the *initial central-line* for each wire segment. The *initial pitch-spacing*, i.e., the distance between the initial central-lines, remains *unchanged* during the sizing procedure. We consider two wire sizing formulations. One is the *symmetric* wire sizing formulation, where wires are always symmetric with respect to initial central-lines as illustrated in Figure 6.3(a). In contrast, in the *asymmetric* wire sizing formulation shown in Figure 6.3(b), wires of same widths are asymmetric with respect to initial central-lines, and have smaller capacitance and less delay. Because neighboring wires are, in general, asymmetrically away from interested nets, the asymmetric wire sizing formulation is capable of further reducing the interconnect delay.

Given the asymmetric formulation, in general, the wire sizing solution for wire segment E_i needs to be represented by a pair of widths $(x_i^{\uparrow}, x_i^{\downarrow})$, where $x_i^{\uparrow}$ is the

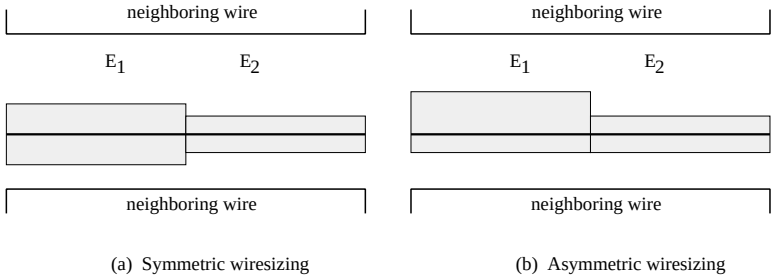


Figure 6.3: (a) symmetric wire sizing, and (b) asymmetric wire sizing. The asymmetric wire sizing has smaller capacitance and less delay.

width of the piece of wire above (or left to) the initial central-line when E_i is a horizontal (or vertical) segment, and $x_i^{\downarrow}$ the width of the piece of wire on the other side of the initial central-line. Similarly, we denote the spacing above (or left to) E_i as $s_i^{\uparrow}$, and spacing on the other side as $s_i^{\downarrow}$. In order to maintain the connectivity, we say that a wire width x_i is *valid* if $x_i^{\uparrow}$ and $x_i^{\downarrow}$ are at least $W_{min}/2$, where W_{min} is the minimum wire width set by the manufacture technology.

With consideration of both symmetric and asymmetric wire sizing formulations, we define the following *global interconnect sizing and spacing (GISS)* problem:

Formulation 4 *Given multiple nets with initial central-line for each wire segment E_i, the GISS problem is to determine a valid wire width $(x_i^{\uparrow}, x_i^{\downarrow})$ for each E_i in these nets such that the weighted delay for multiple nets, given by Eqn. (6.2), is minimized.*

We also define a more restricted *single-net interconnect sizing and spacing (SISS)* problem:

Formulation 5 *Given multiple nets with initial central-line for each wire segment E_i, the SISS problem is to assume that all neighboring wire segments for a*

specific net have fixed wire sizing solutions, and then to determine a valid wire width $(x_i^{\uparrow}, x_i^{\downarrow})$ *for each* E_i *in this net such that the weighted delay for this net, given by Eqn. (6.2), is minimized.*

Note that both formulations treat that c_a and c_{ef} are functions of wire widths and spacings. Furthermore, the SISS formulation can be treated as a special case of the GISS formulation where all nets except the specific net do not have contributions to the weighted delay given in Eqn. (6.2). In the following, we will solve first the SISS problem then the GISS problem, and solve first the symmetric wire sizing formulation and then the asymmetric wire sizing formulation.

6.2 Properties and Algorithms

6.2.1 Bound Computation for Symmetric SISS Problem

Our WS-bounded capacitance model is a table-based model simplified from the 2.5D capacitance model presented in Chapter 3. In this model, for a wire segment E with width w and spacing $s^{\uparrow}$ and $s^{\downarrow}$ to its two nearest neighboring wires, we denote c_a as $c_a(w, s^{\uparrow}, s^{\downarrow})$, and c_{ef} as $c_{ef}(w, s^{\uparrow}, s^{\downarrow})$. Furthermore, for two spacing values s_1 and s_2, we have shown in Chapter 3 that the following approximations

$$c_a(w, s_1, s_2) = \{c_a(w, s_1, s_1) + c_a(w, s_2, s_2)\}/2, \tag{6.3}$$

$$c_{ef}(w, s_1, s_2) = \{c_{ef}(w, s_1, s_1) + c_{ef}(w, s_2, s_2)\}/2 \tag{6.4}$$

are accurate enough. Therefore, we only need to use the numerical capacitance extraction to solve the *basic geometric structure* with equal spacings (see Figure 6.1). We consider different width and spacing combinations, and store c_a and c_{ef} in two-dimensional tables indexed by widths and spacings. Then, for the given wire segment with width w and spacings $s^{\uparrow}$ and $s^{\downarrow}$ to its two nearest neighboring

wires, $c_a(w, s^{\uparrow}, s^{\downarrow})$ and $c_{ef}(w, s^{\uparrow}, s^{\downarrow})$ are obtained from two-dimensional tables using Eqn. (6.3)-(6.4). Justifications and more details about the capacitance model can be found in Chapter 3.

For wire segment E_i with width $x_i = (x_i^{\uparrow}, x_i^{\downarrow})$ and spacing $(s_i^{\uparrow}, s_i^{\downarrow})$, let $s_i^{\uparrow}$ and $s_i^{\downarrow}$ be the spacings between E_i and its neighboring wire segments E_j and E_k, respectively. Because our SISS formulation assumes that the initial central-lines are fixed, $s_i^{\uparrow}$ can be determined by $x_i^{\uparrow}$ and $x_j^{\downarrow}$, and $s_i^{\downarrow}$ by $x_i^{\downarrow}$ and $x_k^{\uparrow}$. Therefore, $c_a(i)(x_i, s_i^{\uparrow}, s_i^{\downarrow})$ and $c_{ef}(i)(x_i, s_i^{\uparrow}, s_i^{\downarrow})$ can be rewritten as $c_a(i)(x_i^{\uparrow}, x_i^{\downarrow}, x_j^{\downarrow}, x_k^{\uparrow})$ and $c_{ef}(i)(x_i^{\uparrow}, x_i^{\downarrow}, x_j^{\downarrow}, x_k^{\uparrow})$. Because coefficients c_a and c_{ef} are bounded, according to the definitions of the SISS problem and the bounded CH-program, we have the following Theorem 12:

Theorem 12 *The symmetric SISS problem under the WS-bounded capacitance model is a bounded CH-program.*

Note that the SISS problem is easier than the STIS problem in the sense that coefficient c_a or c_{ef} for any wire segment in SISS are functions of just one variable for the symmetric formulation, and are functions of two variables for the asymmetric formulation, whereas coefficient r_0 in STIS may depend on all variables.

Based on this theorem, we may use the ELR operation to compute the lower and upper bounds for x_i^*, the optimal width for a wire segment E_i. We assume that $c_a \in [c_a^L, c_a^U]$ and $c_{ef} \in [c_{ef}^L, c_{ef}^U]$. In an ELR operation on a wire E_i during the lower-bound computation, we use $c_a^U(i)$ and $c_{ef}^L(i)$ instead of $c_a(i)$ and $c_{ef}(i)$ for E_i, and use $c_a^L(d)$ and $c_{ef}^L(d)$ instead of $c_a(d)$ and $c_{ef}(d)$ for E_d, where E_d is a downstream segment in the same net as E_i. Similarly, during the upper-bound computation for E_i, we use $c_a^L(i)$ and $c_{ef}^U(i)$ instead of $c_a(i)$ and $c_{ef}(i)$ for E_i, and use $c_a^U(d)$ and $c_{ef}^U(d)$ instead of $c_a(d)$ and $c_{ef}(d)$ for downstream segment E_d.

The bound-computation for the SISS problem can be simplified when the WS-bounded model is monotonically constrained. We first define the following *monotonically constrained capacitance table*:

Definition 12 *A capacitance table is monotonically constrained if the following is true with respect to the basic geometric structure in Figure 6.1: for any two combinations of widths and spacings* (w_1, s_1) *and* (w_2, s_2)*, if* $w_1 \leq w_2$ *and* $s_1 \geq s_2$*, then* $c_a(w_1, s_1, s_1) \geq c_a(w_2, s_2, s_2)$*,* $c_a(w_1, s_1, s_1) \cdot w_1 \leq c_a(w_2, s_2, s_2) \cdot w_2$*, and* $c_{ef}(w_1, s_1, s_1) \leq c_{ef}(w_2, s_2, s_2)$*,*

Intuitively, when the width increases and the spacing decreases, c_{ef} often increases because of the larger coupling capacitance (see c_{ef} for center-to-edge spacing $= 1.1\mu m$ in Figure 6.2(b)). Also, the unit-area capacitance c_a often decreases because of the stronger shielding effect due to that the neighboring wires are closer. Note that the unit-length area-capacitance (like $c_a(w_1, s_1, s_1) \cdot w_1$ and $c_a(w_2, s_2, s_2) \cdot w_2$) often still increases even though the unit-area capacitance c_a decreases.

We say that the WS-bounded model is monotonically constrained if its capacitance table is monotonically constrained. One may easily verify the following theorem using the definitions for the SISS problem and the monotonically constrained CH-program:

Theorem 13 *The symmetric SISS problem under the WS-bounded capacitance model is a a monotonically constrained CH-program if the capacitance model is monotonically constrained.*

In this case, the PLR operation can be used instead of the ELR operation. To tighten a lower- (upper-) bound x_i for a wire E_i, we assume that its neighboring wires have lower- (upper-) bound widths at spacings $s_i^{\uparrow}$ and $s_i^{\downarrow}$ away from E_i.

We obtain $c_a(x_i, s_i^{\uparrow}, s_i^{\downarrow})$ and $c_{ef}(x_i, s_i^{\uparrow}, s_i^{\downarrow})$ by table lookup, and perform an PLR operation on x_i. Compared with the ELR operation, the PLR operation is more efficient and may lead to smaller gaps between lower and upper bounds.

In order to exploit the optimality of the ELR operation and the efficiency of the PLR operation, our implementation of the ELR operation is a hybrid of both operations. When working on a wire E_i, we first check capacitance values with respect to all valid widths and spacings for E_i,[2] then use an PLR operation if Definition 12 is satisfied. Otherwise, we use an ELR operation.

By using the ELR or PLR operation, we obtain lower and upper bounds only for the optimal total-width x_i^*. If the resulting bound is x_i, we assign $x_i^{\uparrow} = x_i^{\downarrow} = x_i/2$ for the symmetric SISS problem. Therefore, starting with the minimum and maximum symmetric wire sizing solutions for all wire segments, and using iterative ELR or PLR operations, we can compute ELR-tight lower and upper bounds for the globally optimal solution to the symmetric SISS problem.

Let $c_{total}^0 = c_a(i) \cdot w + c_{ef}(i)$ be *unit-length capacitance* for a wire. The delay objective Eqn. (6.2) can be re-written as

$$\begin{aligned} & t(\mathbf{X}) \\ = & \sum_{i,j} F(i,j) \cdot \frac{r_0(i)}{x_i} \cdot c_{total}^0(j) \\ + & \sum_i G(i) \cdot \frac{r_0(i)}{x_i} + \sum_i H(i) \cdot \frac{r_0(i)}{x_i} \cdot c_{total}^0(i), \end{aligned} \tag{6.5}$$

where x_i is the width for a transistor M_i or a wire E_i, and $r_0(i)$ is its unit-size effective-resistance. When the WS-bounded model is monotonically constrained, it is easy to verify that Eqn. (6.2) is an instance of Eqn. (4.6). According

[2] A dynamic-programming scheme is used based on two-dimensional cache tables, which, similar to our capacitance tables, are indexed by widths and spacings. For given maximum width and minimum spacing, the cache tables return the minimum or maximum values for c_a and c_{ef}, or imply that the PLR operation can be used.

to Lemma 2, the LR operation can be used to compute the lower and upper bounds for the optimal solution to the SISS problem. The LR operation is less efficient than the PLR operation as shown in the following: we assume that for wire segment E_i with width $x_i = \{x_i^{\uparrow}, x_i^{\downarrow}\}$, there are ten choices for both $x_i^{\uparrow}$ and $x_i^{\downarrow}$. Because c_a and c_{ef} in the WS-bounded model depend on $x_i^{\uparrow}$ and $x_i^{\downarrow}$, we need to enumerate these width choices to find the *local optimal* value for x_i. It costs ten computations for the symmetric wire sizing formulation, and 10×10 computations for the asymmetric wire sizing formulation. On the other hand, the PLR operation needs only one computation to find out the *pseudo-local optimal* value for x_i. Even though the LR operation may need fewer numbers of passes of lower- and upper-bound computations in the bound-computation algorithm, overall, using LR operations is less efficient than using PLR operations in our experiments.

6.2.2 Bound Computation for Symmetric GISS Problem

For the interested wire segment E_i with width x_i and spacings $s_i^{\uparrow}$ and $s_i^{\downarrow}$ to its two nearest neighboring wires E_j and E_k, only x_i is a variable, but wire widths x_j and x_k are both constants for the SISS problem. However, x_i, x_j and x_k are all variables for the GISS problem. In this case, we represent the unit-length effective-fringe capacitance $c_{ef}(i)$ for E_i as

$$c_{ef}(i) \;=\; c_{ef}^{\uparrow}(i) + c_{ef}^{\downarrow}(i), \tag{6.6}$$

where $c_{ef}^{\uparrow}(i)$ is the unit-length effective-fringe capacitance between E_i and E_j, and $c_{ef}^{\downarrow}(i)$ the unit-length effective-fringe capacitance between E_i and E_k. Furthermore, we re-write

$$c_{ef}^{\uparrow}(i) \;=\; c_{ef}^{0\uparrow}(i) \cdot (x_i + x_j), \tag{6.7}$$

$$c_{ef}^{\downarrow}(i) \;=\; c_{ef}^{0\downarrow}(i)\cdot(x_i + x_k). \tag{6.8}$$

where with respect to the fixed initial central-lines, $c_{ef}^{0\uparrow}(i)$ is a function of x_i and x_j, $c_{ef}^{0\downarrow}(i)$ is a function of x_i and x_k.

Based on this representation for c_{ef}, the objective Eqn. (6.2) for the GISS problem can be re-written as

$$\begin{aligned}
t(\mathbf{X}) &= \sum_{i,j} F(i,j)\cdot\frac{r_0(i)}{x_i}\cdot c_a(j)\cdot x_j + \sum_{i,j} F(i,j)\cdot\frac{r_0(i)}{x_i}\cdot c_{ef}(j) \\
&+ \sum_i G(i)\cdot\frac{r_0(i)}{x_i} + \sum_i H(i)\cdot\frac{r_0(i)}{x_i}\cdot c_{ef}(i) \\
&= \sum_{i,j} F(i,j)\cdot\frac{r_0(i)}{x_i}\cdot c_a(j)\cdot x_j \\
&+ \sum_{i,j} F(i,j)\cdot\frac{r_0(i)}{x_i}\cdot\{c_{ef}^{0\uparrow}(j)\cdot(x_j + x_n^{\uparrow}(j)) + c_{ef}^{0\downarrow}(j)\cdot(x_j + x_n^{\downarrow}(j))\} \\
&+ \sum_i G(i)\cdot\frac{r_0(i)}{x_i} + \\
&+ \sum_i H(i)\cdot\frac{r_0(i)}{x_i}\cdot\{c_{ef}^{0\uparrow}(i)\cdot(x_i + x_n^{\uparrow}(i)) + c_{ef}^{0\downarrow}(i)\cdot(x_i + x_n^{\downarrow}(i))\}
\end{aligned} \tag{6.9}$$

where $x_n^{\uparrow}(i)$ and $x_n^{\downarrow}(i)$ are neighboring wires for wire E_i, and $x_n^{\uparrow}(j)$ and $x_n^{\downarrow}(j)$ are neighboring wires for wire E_j.

Because values for c_a and c_{ef}^0 are obviously bounded, we have the following Theorem 14:

Theorem 14 *The symmetric GISS problem under the WS-bounded capacitance model is a bounded CH-program.*

Based on this theorem, we may use the ELR operation to compute the lower and upper bounds for x_i^*, the optimal width for a wire segment E_i in the GISS problem. We use the same rules for c_a as those we used for the SISS problem: If we assume that $c_a \in [c_a^L, c_a^U]$ and E_i has two neighboring wires E_j and E_k, in an

ELR operation during the lower-bound computation for E_i, we use $c_a^U(i), c_a^U(j)$ and $c_a^U(k)$ instead of $c_a(i), c_a(j)$ and $c_a(k)$ for E_i, E_j and E_k, and use $c_a^L(d)$ instead of $c_a(d)$ for E_d that is a downstream segment of E_i, E_j, or E_k. Similarly, during the upper-bound computation for E_i, we use $c_a^L(i), c_a^L(j)$ and $c_a^L(k)$ for E_i, E_j and E_k, and $c_a^U(d)$ for downstream segment E_d. The following rules similar to those for c_a are used for c_{ef}^0: during the lower-bound computation, the upper bound of c_{ef}^0 will be used for E_i, E_j and E_k, and lower bound of c_{ef}^0 for downstream segment E_d; during the upper-bound computation, the lower bound of c_{ef}^0 will be used for E_i, E_j and E_k, and upper bound of c_{ef}^0 used for E_d[3].

6.2.3 Bound Computation for Asymmetric SISS and GISS Problems

In the following, we consider the asymmetric SISS and GISS problems. Because the SISS problem is a simpler version of the GISS problem, we present our algorithm in the following sections only using the GISS formulation. We first extend the dominance relation to consider the asymmetric wire sizing formulation. We say that the wire sizing solution $\mathbf{X}$ dominates another solution $\mathbf{X}'$ (denote as $\mathbf{X} \geq \mathbf{X}'$), if $(x_i^{\uparrow}, x_i^{\downarrow}) \geq (x_i'^{\uparrow}, x_i'^{\downarrow})$ (i.e., $x_i^{\uparrow} \geq x_i'^{\uparrow}$ and $x_i^{\downarrow} \geq x_i'^{\downarrow}$) holds for any wire segment E_i. A lower and upper bound of the exact solution to the asymmetric GISS problem will be determined according to the new definition of dominance relation.

We then solve the asymmetric SISS and GISS problems by augmenting the bound-computation algorithm presented in the above two sections. Each ELR or PLR operation gives only the total-width x_i, which is a lower or upper bound of

[3]Note that the monotonically constrained capacitance model can be extended with respect to the representation for c_{ef} given in Eqn. (6.6-6.8) so that the GISS objective Eqn. (6.2) becomes a monotonically constrained CH-function, and then the PLR operation can be used instead of the ELR operation.

the optimal total-width x_i^* for E_i. To obtain an asymmetric wire sizing solution, we need to separate x_i into $x_i^{\uparrow}$ and $x_i^{\downarrow}$, which are respective widths for the "two pieces" of wires around the initial central-line of E_i. This separation is equivalent to embed a wire with total-width x_i around the initial central-line of E_i. It also affects the ELR and PLR operations in the subsequent steps. We propose to perform a *conservative embedding* right after any ELR or PLR operation.

We assume that $x_i^* = (x_i^{\uparrow *}, x_i^{\downarrow *})$ is the width for E_i in the exact asymmetric solution. Let $x_i^{\uparrow L}$ and $x_i^{\uparrow U}$ be the lower and upper bounds for $x_i^{\uparrow *}$, and $x_i^{\downarrow L}$ and $x_i^{\downarrow U}$ the lower and upper bounds for $x_i^{\downarrow *}$. If we obtain a total-width x_i^L in the lower-bound computation, the conservative embedding (CE) operation computes $x_i^{\downarrow L} = x_i^L - x_i^{\uparrow U}$, which is a *conservative* lower-bound for $x_i^{\downarrow *}$. Similarly, $x_i^{\uparrow L} = x_i^L - x_i^{\downarrow U}$ is a conservative lower bound for $x_i^{\uparrow *}$. Note that the sum of $x_i^{\uparrow L}$ and $x_i^{\downarrow L}$ may be *less* than x_i^L in the CE operation. Symmetrically, for an upper-bound x_i^U, we compute $x_i^{\uparrow U} = x_i^U - x_i^{\downarrow L}$, and $x_i^{\downarrow U} = x_i^U - x_i^{\uparrow L}$. This augmented algorithm leads to the lower and upper bounds of the exact solution to the asymmetric GISS problem.

We also define a greedy embedding (GE) operation. Recall that neighboring wires of E_i have their lower- (upper-) bound widths during lower- (upper-) bound computation for E_i. If the lower or upper bound of wire width for E_i is x_i, we find $x_i^{\uparrow}$ and $x_i^{\downarrow}$ such that $x_i^{\uparrow} + x_i^{\downarrow} = x_i$ and the objective function Eqn. (6.2) is minimized with respect to the given neighboring wires. Different from the CE operation, the GE operation does not always lead to a lower or upper bound of the exact solution for the asymmetrical GISS problem. We will show, however, that the GE operation has a higher convergence rate than the CE operation in experiments, and achieves satisfactory experimental results in Section 6.3. Again, we say the computation on a wire segment is *convergent* if lower and upper bounds

are identical.

6.2.4 Overall Algorithm for Asymmetric SISS and GISS Problems

Our overall asymmetric GISS algorithm (denoted as GISS/ELR algorithm, see Table 6.1) consists of the following three steps. First, we compute the ELR-tight lower and upper bounds using iterative ELR operations and CE operations. Our ELR implementation invokes PLR operations when PLR operations assure the optimality. Then, if the resulting lower and upper bounds do not meet, we will use iterative LR operations and GE operations to further improve the lower and upper bounds. We carry out the LR operation and GE operations simultaneously as the following: for a wire segment, we enumerate width choices for two wire-pieces between lower and upper bounds, and the two widths that minimize our multiple-net objective function Eqn. (6.2) are the LR and GE result. Note that the first step guarantees the optimality in the sense that there exists a global exact solution within the resulting ELR-tight lower and upper bounds. However, this kind of optimality may not hold in the second step. Finally, for each net that still has non-convergent wire segments, we will assume that other nets have lower-bound wire widths, and invoke the dynamic-programming-based SISS algorithm presented in [23] to find the final sizing and spacing solution within its lower and upper bounds. This SISS algorithm combines the asymmetric wire sizing formulation and the wire sizing algorithm based on the bottom-up dynamic-programming technique [49]. We apply this SISS algorithm in the greedy order such that the more timing-critical net is processed earlier.

GISS/ELR Algorithm
1. Compute ELR-tight lower and upper bounds using iterative ELR operations and CE operations; 2. Compute LR-tight "lower" and "upper" bounds using iterative LR operations and GE operations; 3. For all non-convergent nets in the greedy order, invoke single-net dynamic-programming based algorithm within resulting lower and upper bounds.

Table 6.1: Asymmetric GISS algorithm based on ELR and LR operations

6.3 Experimental Results

We have implemented GISS/ELR algorithm using C++ programming language, and have tested the algorithm by a large number of examples. Experiment in this section use parameters based on the $0.18\mu m$ technology specified in NTRS'94 [77]. We assume the following: the sheet resistance of wire is 0.0638 $\Omega/\Box$, the minimum wire width is $0.22\mu m$, and minimum edge-to-edge spacing between neighboring wires is $0.33\mu m$. We define the min_pitch, as the sum of minimum spacing and minimum wire size, i.e., $0.55\mu m$. The allowable wire widths are from 0.22 to 1.1 μm, with the incremental step of 0.11 μm. The capacitance tables are generated using numerical capacitance extractions, as presented in Chapter 3.

6.3.1 Single-net Interconnect Sizing and Spacing

Our GISS/ELR algorithm and implementation can be used for both single-net and multi-net cases. Here, we apply this algorithm to find the optimal interconnect sizing and spacing solution for five nets extracted from an advanced indus-

trial micro-processor. The routing trees are the same as those used in Chapter 2. These trees originally have multiple sources, and we randomly assign one as the unique single source. We assume that each wire segment has neighboring wires at randomly-generated spacings from 1 to $5 \times min\_pitch$. We assign equal criticality for each sink so the weighted delay is equal to the average delay in this experiment.

We compare the average and maximum delays for the following solutions: minimum wire sizing solution (MIN); symmetric optimal wire sizing solution (OWS-S) without considering the coupling capacitance (but coupling capacitance is counted in HSPICE simulation); symmetric SISS solution (SISS-S) given by GISS/ELR algorithm and asymmetric SISS solution (SISS-A) given by GISS/ELR algorithm. We report delays via HSPICE simulations in Table 6.2. One can see from the table that SISS-A is consistently better than all other algorithms. In terms of the average delay, which is our objective function, the improvement is up to 51.6% over the MIN solution, 34.1% over OWS-S, and 32.7% over SISS-S.

Although the average delay is our objective, experimental results show that this formulation reduces the maximum delay as well. From the table, we can see that SISS-A is better than MIN, OWS-S and SISS-S by up to 47.6%, 38.0% and 32.3%, respectively. By assigning more weight to sinks with the most delays, we may further reduce the maximum delay.

We say that the computation for a wire segment is convergent if the resulting lower and upper bounds are identical. In Table 6.3, we report the convergence rate, which is the percent of wire segments that have identical lower and upper bounds after our bound computation algorithm. In the Table, ELR-S stands for our ELR-based algorithm under the symmetric SISS formulation, and ELR-A

	length	Average Delay (ns)				Maximum Delay (ns)			
	(mm)	MIN	OWS-S	SISS-S	SISS-A	MIN	OWS-S	SISS-S	SISS-A
Net1	3.6	0.08	0.08 (-0.00%)	0.08 (-0.00%)	0.08 (-0.00%)	0.14	0.14 (-0.00%)	0.14 (-0.00%)	0.14 (-0.00%)
Net2	6.6	0.31	0.19 (-38.7%)	0.18 (-41.9%)	0.15 (-51.6%)	0.34	0.22 (-35.3%)	0.22 (-35.3%)	0.19 (-44.1%)
Net3	10.07	0.78	0.71 (-9.0%)	0.71 (-9.0%)	0.61 (-21.8%)	1.03	0.96 (-6.8%)	0.95 (-7.8%)	0.83 (-19.4%)
Net4	10.57	0.54	0.41 (-24.1%)	0.39 (-27.8%)	0.27 (-50.0%)	0.84	0.71 (-15.5%)	0.65 (-22.6%)	0.44 (-47.6%)
Net5	31.98	3.19	2.57 (-19.4%)	2.57 (-19.4%)	1.73 (-45.8%)	4.92	4.26 (-13.4%)	4.27 (-13.2%)	3.26 (-33.7%)

Table 6.2: Comparison of the average and maximum delays using different algorithms.

Net	Convergence rate		Run time (s)			
	ELR-S	ELR-A	DP-S	ELR-S	DP-A	ELR-A
net1	100%	100%	0.13	0.02	5.58	0.04
net2	100%	100%	0.71	0.03	41.1	0.06
net3	100%	100%	0.80	0.05	48.9	0.11
net4	100%	100%	0.91	0.04	39.8	0.09
net5	100%	100%	3.62	0.07	980	0.8

Table 6.3: The convergence rates for the ELR-based bound computation under both symmetric and asymmetric SISS formulations. Also shown are the run-times for SISS algorithms based on dynamic-programming and ELR operations, respectively.

for this algorithm under the asymmetric SISS formulation. In all cases, 100% convergence rate is achieved. In [23], a dynamic-programming based SISS algorithm is developed[4]. In the same Table, we also report the running time comparison between dynamic-programming based algorithms and ELR-based algorithms. We represent the dynamic-programming based algorithms under symmetric and asymmetric SISS formulations by DP-S and DP-A, respectively. The runtime by LR-S is only about 1/50 to 1/6 of that by DP-S. For asymmetric case, the speed-up of LR-A versus DP-A is even more significant. Therefore, for the SISS problem in practice, we recommend to first compute the lower and upper bounds by LR-based algorithm, then apply the dynamic programming within the final lower and upper bounds if they are not identical.

[4]The implementation will be reported in Mr. Zhigang Pan's dissertation.

6.3.2 Multi-net Interconnect Sizing and Spacing

We have tested our GISS algorithm on a 16-bit parallel bus structure. In this bus, each bit is a $1cm$ line with a 119 Ω driver resistance and a 12.0fF sink capacitance. We assume that initially these lines are equally spaced. We will find an asymmetric wire sizing for every $500\mu m$-long wire segment.

pitch-spacing	Convergence		Average gap (μm)	
	ELR/CE	LR/GE	ELR/CE	LR/GE
2x	54%	100%	0.20	0.0
3x	42%	83%	0.33	0.037
4x	49%	100%	0.44	0.0
5x	45%	84%	0.55	0.048
6x	30%	100%	0.90	0.0

Table 6.4: Convergence of ELR/CE and LR/GE in GISS/ELR algorithm

We optimized the bus for different initial pitch-spacings, from 2x to 6x of the minimum pitch-spacing ($0.55\mu m$). Our GISS/ELR algorithm has two bound-computation phases, the first one using ELR/CE operations and the second one using LR/GE operations (see Table 6.1). As shown in Table 6.4, computations for from 30% to 54% wire segments are convergent, i.e., identical lower and upper bounds are achieved for these segments after the ELR/CE phase. The average gap after the ELR/CE phase is between 0.90 μm and 0.20 μm. Furthermore, the LR/GE phase increases the convergence rate to at least 83%.

We compare to an alternative GISS algorithm presented in [23]. Based on an effective-fringe property, it uses a bottom-up dynamic programming technique to compute lower and upper bounds for the global solution to the asymmetric GISS

pitch-spacing	Average Delay (ns)				Run Time (s)	
	SISS	GISS/FAF	GISS/VAF	GISS/ELR	GISS/VAF	GISS/ELR
2x	1.31	0.82(-37%)	0.82(-37%)	0.80(-39%)	183	0.69
3x	0.72	0.63(-13%)	0.56(-22%)	0.53(-27%)	189	1.23
4x	0.46	0.46(+0.0%)	0.45(-2.2%)	0.45(-2.2%)	511	1.82
5x	0.38	0.39(+2.6%)	0.37(-2.6%)	0.36(-5.3%)	1083	2.44
6x	0.35	0.36(+2.9%)	0.34(-2.9%)	0.32(-8.6%)	1379	2.28

Table 6.5: Comparison of different sizing solutions for the bus structure.

problem when c_a and c_f are constants. We call it GISS/FAF. The algorithm may be extended to use variable c_a and c_f under the WS-bounded capacitance model, and we call it GISS/VAF. In both cases, the exact solution may be *outside* the range defined by the resulting lower and upper bounds. Both GISS/FAF and GISS/VAF algorithms further use the SISS algorithm to obtain final solutions within the lower and upper bounds, whereas the GISS/ELR algorithm uses the lower bound as the final solution due to its high convergence. In addition, we also apply the SISS algorithm in a greedy order, which is equivalent to invoking only step 3 in the GISS/ELR algorithm (Table 6.1). The SISS algorithm obtains a local-optimal solution for the GISS problem.

We compare the average HSPICE delay for solutions given by these algorithms in Table 6.5 (average delay is our objective function). As seen from the table, the GISS/ELR algorithm always achieves results better than the SISS solutions, with up to 39% delay reduction. Therefore, it is important to find the globally optimal solution to the GISS problem. The improvement of the GISS/ELR algorithm over the SISS algorithm is reduced when the pitch spacing increases, due to the fact that the coupling capacitance is less significant for larger pitch spacings. Nevertheless, compared with the SISS algorithm, the GISS/ELR al-

gorithm still reduces the average delay by 8.6% in the case of maximum pitch spacing. Because neither c_a nor c_f is a constant in DSM designs, both GISS/ELR and GISS/VAF algorithms obtain better results than the GISS/FAF algorithm does. The GISS/ELR algorithm obtains an extra delay reduction of up to 17% when compared with the GISS/FAF algorithm. Furthermore, compared to the GISS/VAF algorithm, the extra delay reduction of the GISS/ELR algorithm is up to 7.1%. More significantly, the GISS/ELR algorithm runs 100x faster. It also uses much less memory. Very recently, it was brought to our attention that an alternative method to the GISS problem, named Bound Refinement (BR), has been developed. BR outperforms GISS/VAF in terms of solution quality and runtime. Preliminary experiment also indicates that for some examples, BR produces solutions with smaller delays but with longer runtime compared to the GISS/ELR algorithm[5]. Further comparison between BR and GISS/ELR algorithms is underway. The BR algorithm and more detailed experiment results will be included in Mr. Zhigang Pan's thesis.

6.4 Conclusions and Discussions

In this chapter, we have formulated the simultaneous interconnect sizing and spacing problems for single-net (denoted as SISS problem) and multiple nets (denoted as GISS problem), respectively. Given the topology for multiple nets, the GISS problem finds the wire sizing and spacing solution optimal for *all* nets, and considers coupling capacitance extracted during wire sizing and spacing procedure. The SISS problem is a simpler version of the GISS problem assuming that neighboring wires are fixed for the specific net. We pose both SISS and GISS problems as CH-programs, which directly leads to effective and efficient solutions

[5]Private communication with Zhigang Pan, 1999.

based on bound computation using different types of local-refinement operations.

We have convincingly shown that it is important for DSM designs to consider the coupling capacitance by using the simultaneous interconnect sizing and spacing formulation. Experiment showed that compared to the wire sizing formulation, which is not able to consider the coupling capacitance, the SISS formulation may achieve delay reduction by up to 38%. Furthermore, for multiple nets, the GISS formulation obtains up to 39% delay reduction when compared to the case iteratively applying SISS to these nets.

Our SISS and GISS algorithms based on different types of LR operations are extremely efficient, with no experiment in this chapter having a runtime of over 10 seconds. Note that SISS and GISS formulations can be treated as parts of the STIS formulation. Therefore, we have a unified formulation and solution to the problem of simultaneous device sizing, and wire sizing and spacing for multiple paths, under accurate models for device delay and interconnect coupling capacitance.

In the future, we plan to develop noise model with consideration of both coupling capacitance and coupling inductance, and extend our simultaneous device sizing, wire sizing and spacing algorithm to control noise. Buffer insertion will be taken into account as well.

CHAPTER 7

On-Chip Inductance Model and Its Applications

In this chapter, we study the inductance extraction problem for on-chip interconnects. Our contributions include:

- We propose and *theoretically* validate two foundations which allow us to reduce the problem size of inductance extraction without loss of accuracy, and present a table-based inductance extraction methodology directly based on the two foundations.
- We use this efficient inductance extraction methodology to generate distributed RLC models for on-chip interconnects with consideration of process variations, and also apply the resulting RLC model to optimize on-chip interconnects in the designs of the state-of-the-art microprocessors.

Part of preliminary result are first present in [44]. To the best of our knowledge, it is the first work that presents an efficient and accurate table-based inductance extraction method for on-chip interconnects. Very recently, we have extended the inductance extraction methodology to consider off-chip interconnects in MCM/PCB designs.

The remainder of this chapter is organized as follows: In section 7.1, we introduce the inductance extraction problem for on-chip interconnects. In section 7.2, we validate two foundations which allow us to reduce the problem size of inductance extraction without loss of accuracy. In section 7.3, we propose a table-based

inductance extraction methodology based on the two foundations. In section 7.4, we present two applications of the inductance extraction methodology: (i) to derive the effective (loop) inductance for a coplanar-waveguide; (ii) to be integrated with the statistically-based RC model generation in [5] to generate RLC models for on-chip interconnects. We also use the RLC model to optimize bus structures, using buffer insertion and shielding insertion. In addition, we briefly discuss extensions to model the power/ground planes and to consider inductance for MCM/PCB packages. Section 7.5 concludes this chapter.

7.1 Introduction

Precisely, the impedance of a wire is given by $R + j\omega L$, where R is the resistance of the wire, and L the inductance of the wire. However, the inductance was not considered traditionally for on-chip interconnect designs. According to a conservative rule the inductance is needed only when R is comparable to $j\omega L$. The RC model is used for the earlier chapters and for most IC designs in the reality, because the on-chip interconnect resistance R dominates the on-chip interconnect inductive impedance $j\omega L$ (less conservative figures of merit to characterize the importance of on-chip inductance can be found in [85, 45]).

The on-chip inductance are gaining increasingly importance for DSM designs for the following reasons:

- In VLSI circuits, global signal nets and clock nets often use large widths and are routed on the top layers that have high thicknesses. Moreover, lower resistive copper wires start to replace conventional aluminum wires. All of these factors lead to more on-chip interconnects that have relatively

low resistance and can exhibit significant inductive effects[1].

- As the clock frequency increases and the rising/falling time of the signal decreases, the signal has more and more high frequency components. It makes $j\omega L$ and therefore the inductance effect more significant. Note that the frequency ω in the inductive impedance $j\omega L$ is not given by 2π times the clock frequency, but more precisely by 2π times the *significant frequency*, which is larger than the clock frequency and is given by $0.17/t_r$ [52], where t_r is the rising/falling time of the signal.

- In addition, with the increase of chip size (see Table 1.1), there are increasingly more wires that are long and are running in parallel. It may lead to severe inductive coupling, and this type of coupling exists even for wires that are not directly adjacent.

The on-chip inductance is considered as a very difficult problem, because the inductance in essence is defined for the current loop but the current return path is not well defined for on-chip interconnects. A key idea to solve the problem to extract inductance from complex 3D interconnect structures is the concept of *partial inductance*. It is developed in [64] and was introduced to the circuit design field in [67, 68]. The partial inductance for a wire segment is defined as the portion of the loop inductance for the wire segment, assuming the wire segment forms a loop with the infinity. Based on the concept of the partial inductance (or called *partial element equivalent circuit (PEEC)* model as in [68]), the numerical field solvers are developed [68, 47, 34].

However, extraction of parasitic parameters (resistance, capacitance and in-

[1]Both resistance and inductance are reduced when the cross-section area becomes larger. While the resistance is reversely proportional to the cross-section area, the inductance is much less sensitive to the change of the cross-section area, as shown later on in (7.3) and (7.4).

ductance) via numerical field solvers is hard to support during iterative procedures of simulation and optimization for on-chip interconnects. Accurate and efficient extractions of resistance and capacitance without using numerical methods have been achieved recently. For example, we presented a 2 1/2-D capacitance extraction methodology in Chapter 3. Also, a fast generation of statistically-based worst-case RC models was implemented and used at Hewlett-Packard [5]. Both used the table-based approach, which is suitable for iterative simulation and optimization purposes. Yet, there is no report on table-based inductance extractions. In the following, we will present two foundations concerning the inductance extraction. These two foundations are based on the PEEC model, and will lead to an accurate and efficient table-based inductance extraction methodology.

7.2 Foundations for Inductance Extraction

7.2.1 Preliminaries

There are multiple metal layers in a VLSI technology. We assume that wire traces on adjacent layers are orthogonal, and extract the inductance for a block, which contains n parallel traces (T_1, T_2,..., T_n) of same lengths on the same layer (see Figure 7.1). In addition, we also assume that the two most outside traces, T_1 and T_n, are dedicated power/ground traces. When the block size is three, it is a coplanar-waveguide, which is one of the three basic forms for transmission line[2], and is often used for clock tree in high-speed designs. When the block size is large, it models the bus structure with outside power/ground traces that can be used for shielding only or for shielding and power supply at the same time. Because traces are orthogonal on adjacent layers, traces on layer $N+1$ and layer

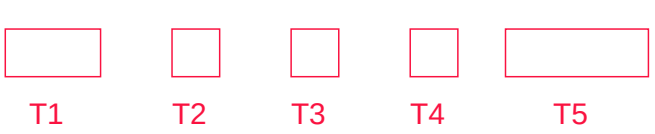
[2]The other two forms are the micro-strip line and strip line. Both will be discussed in Section 7.4.3.

$N-1$ will not affect the inductance of traces on the current layer N [51]. In section 7.5, we will discuss the impacts of layer $N+2$ and layer $N-2$.

Figure 7.1: The cross-section view for a block of n traces, where T_1 and T_n are dedicated power/ground traces. The widths for the traces are W_1, W_2, ..., and W_n, respectively. The spacings between them are S_1, S_2, ... and S_{n-1}, respectively.

Note that the capacitive effect is a "short-range" effect in the sense that for a block, only the mutual capacitance between adjacent traces are important, and the rest of the mutual capacitance can be ignored. Therefore, for any trace, it is sufficient to solve the trace and its two adjacent traces via numerical extraction (Foundation 4 in Chapter 3). In other words, we are able to reduce the n-trace capacitance problem to a number of 3-trace subproblems (see Chapter 3). The inductive effect, however, is a "long-range" effect. For example, in Figure 7.2, we compute the inductance for a block with size n=5 by assuming that the wire thickness is $2.0\mu m$, wire width W1=$4\mu m$, W2=W3=W4=$0.8\mu m$, W5=$2\mu m$, all spacings are $0.8\mu m$, and the length is $4000\mu m$. We specify that T_1 and T_5 are power/ground traces that serve as current return paths, and run a 3D inductance tool RI3 in Raphael [34] to compute loop inductance. The result is the loop inductance in a 3x3 matrix, where current is assumed to return via the two power/ground traces T_1 and T_5. In the matrix, diagonal elements are self inductance, and off-diagonal elements are mutual inductance. The mutual inductance between T_2 and T_4 can not be ignored even though there is T_3 between them.

In general, there is a significant mutual inductance between any traces within



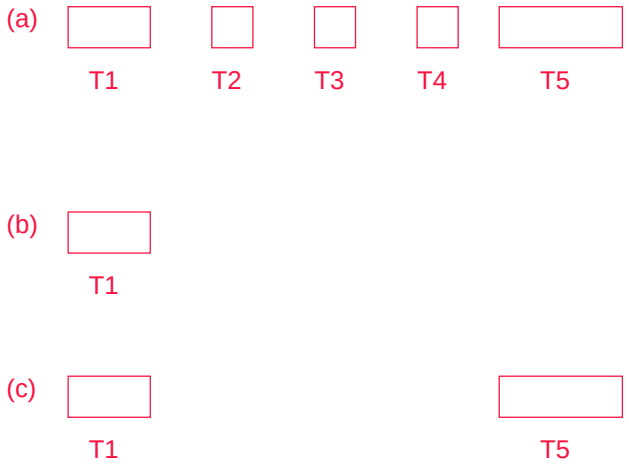
	T2	T3	T4
T2	1.73	1.15	0.53
T3	1.15	1.94	1.24
T4	0.53	1.24	1.92

Figure 7.2: Loop inductance (nH) by specifying that T_1 and T_5 are current return paths.

a block (e.g., even for a block of size n=32). Due to this "long-range" effect, even though we assume that all signal traces have an identical width, and the spacings are identical, the brute-force way to build inductance tables will have large table sizes. The table for self inductance has six dimensions: two widths for power/ground traces, one width for signal traces, the trace location, and uniform spacing and length. Note that the trace length is needed because the inductance is not a linear function of trace length. At the same time, the table for mutual inductance needs locations for two traces, which leads to seven-dimension tables. We may not afford to consider different widths and spacings for different traces.

Furthermore, the loop inductance in Figure 7.2 assumes that all current returns via the two power/ground traces, which may not be true for the high frequency. For example, when only one trace is switching, its current may return from adjacent quiet traces. The right way to extract inductance for a block is to run RI3 without specifying any power/ground traces as current return paths. Then, for the block in Figure 7.2, we obtain the partial inductance in a 5x5 matrix (see Figure 7.3(a)). Again, the diagonal elements are self inductance, and off-diagonal elements are mutual inductance. An important observation is that now the self inductance of a trace depends only on the trace itself, and the mutual inductance of two traces depends only on the two traces themselves. For example, in Figure 7.3(b), we compute the self inductance L_{11} for T_1 with other

traces removed, and obtain the same L_{11} as in Figure 7.3(a). In Figure 7.3(c), we compute the mutual inductance L_{15} for T_1 and T_5 with T_2, T_3 and T_4 removed, and obtain the same L_{15} as in Figure 7.3(a).



	T1	T2	T3	T4	T5
T1	6.17	5.43	5.12	4.89	4.66
T2	5.53	6.79	6.10	5.48	5.04
T3	5.12	6.10	6.79	6.10	5.33
T4	4.89	5.48	6.10	6.79	5.77
T5	4.66	5.04	5.33	5.77	6.50

	T1
T1	6.17

	T1	T5
T1	6.17	4.66
T5	4.66	6.50

Figure 7.3: Partial inductance (nH) without specifying any current return path: (a) a block of size $n = 5$, (b) only trace T_1 (with other traces removed), and (c) only two traces T_1 and T_5 (with traces T_2-T_4 removed).

When we do not specify which traces are power/ground traces, we compute partial inductance (denoted as L_p) under the PEEC model. In general, we have the following foundations:

Foundation 6 *Self L_p of a trace is solely decided by the trace (its length, width and thickness).*

Foundation 7 *Mutual L_p of two traces is solely decided by the two traces (their lengths, widths and thicknesses, and the spacing between them).*

7.2.2 Validation of foundations

In order to validate the two foundations, the following illustrates the inductance extraction procedure under the PEEC model. The PEEC model was introduced in [67, 68], and has been widely used in numerical inductance extraction tools (for example, [34, 47]). Because the inductance is defined only for closed loops, the partial inductance of a trace can be viewed as the inductance of the trace as it forms a loop with infinity. If the current density is uniform in traces T_k and T_m, according to [68], the mutual inductance $L_{p_{km}}$ under the PEEC model is:

$$L_{p_{km}} = \frac{\mu}{4\pi}\frac{1}{a_k a_m}\int_{b_k}^{c_k}\int_{a_k}\int_{b_m}^{c_m}\int_{a_m}\frac{dl_k \cdot dl_m}{r_{km}} da_k \cdot da_m \qquad (7.1)$$

where a_k and a_m are cross-sectional areas, b_k and b_m are starting points, c_k and c_m are ending points, all for traces T_k and T_m, respectively. In addition, r_{km} is the distance between dl_k and dl_m, which represent differential elements of length for traces T_k and T_m, respectively. When $k = m$, Eqn. (7.1) gives the self L_p of a trace.

In the case where the current is not uniform in a trace, a trace can be divided into rectangular filaments (see Figure 7.4). The current is assumed to flow along the length of each filament with a constant density within each filament. Therefore, Eqn. (7.1) may be used for each filament. It is easy to see that Foundations 6 and 7 hold for each filament with respect to (7.1). I.e., the self L_p of a filament is *solely* decided by the filament, and the mutual L_p between two filaments is *solely* decided by the two filaments. The conclusions hold for cases of a single trace and multiple traces.

If we assume that trace T_k has P filaments, and trace T_m Q filaments, then

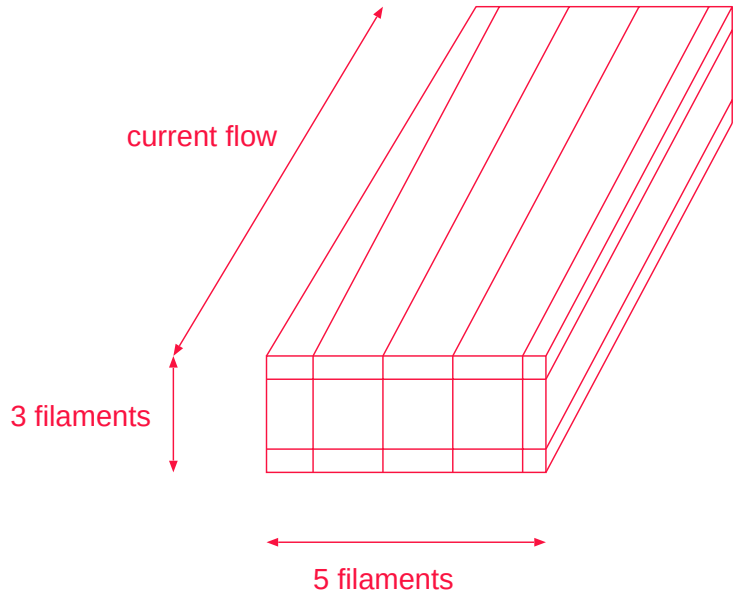


Figure 7.4: A trace is divided into 3×5 filaments. It is assumed that the current density is the same within a filament.

$L_{p_{km}}$ is given by

$$L_{p_{km}} = \sum_{i=1}^{P} \sum_{j=1}^{Q} L_{p_{ij}} \tag{7.2}$$

where $L_{p_{ij}}$ is the mutual L_p between filament i of T_k and filament j of T_m. Again, when $k = m$, Eqn. (7.2) computes the self L_p for a trace. Because Foundations 6 and 7 hold for $L_{p_{ij}}$, it is easy to see that Foundations 6 and 7 still hold after using Eqn. (7.2) to compute L_p for traces.

7.3 Table-based Inductance Extraction

The two foundations enable us to reduce the n trace inductance problem into 1-trace subproblems to solve the self L_p, and into 2-trace subproblems to solve the mutual L_p. There is no loss of accuracy during the reduction. As given in

[41], the self inductance may be solved by

$$L(nH) = 2l[\ln \frac{2l}{w+t} + 0.5 - k] \tag{7.3}$$

where $k = f(w,t)$ and $0 < k < 0.0025$, and l, w, and t are length, width and thickness of the trace in unit of cm. The mutual inductance for two traces of same width and length is

$$L(nH) = \frac{\mu_0 l}{2\pi}[\ln \frac{2l}{s} - 1 + \frac{s}{l}] \tag{7.4}$$

where s is spacing between two traces, again in unit of cm.

These equations give us two insights: First, the inductance for on-chip interconnects is not linearly scalable. Both self and mutual inductance are super-linear functions of the trace length. Secondly, because of the logarithmic operation of $\frac{2l}{w+t}$ and $\frac{l}{s}$, both mutual and self inductance is less sensitive to variations of trace width and spacing as the capacitance and resistance are. The two insights are also verified by experiments with numerical inductance tools. For example, in Table 7.1, we report both self and mutual inductance for two parallel wire traces with pitch-spacing being $3\mu m$. Columns 2-4 are inductance values for wire length of $1000\mu m$, and column 5-7 are inductance values for wire length of $4000\mu m$. For each length, three wire widths are assumed, namely, $1\mu m$, $1.2\mu m$ (i.e., the width has 20% variation compared to width=$1\mu m$), and $2\mu m$ (i.e., the wire is up-sized by 100% compared to width=$1\mu m$). One can see the following from the table: First, the inductance is not linearly scalable with respect to the wire length. For example, when the wire width is $1\mu m$ and the wire length increases from $1000\mu m$ to $4000\mu m$ (an increasing factor of 4x), the self inductance increases by a factor of 4.74x rather than 4x. Second, the inductance variation is only around 1% for the 20% variation of wire width (see columns 3 and 6). Therefore, there is no need to consider the impact of process variation for inductance extraction, even

length	$1000\mu m$			$4000\mu m$		
width(μm)	1	1.2	2	1	1.2	2
Self L_p(nH)	1.480(0.0%)	1.461(-1.2%)	1.400(-5.4%)	7.028(0.0%)	6.951(-1.1%)	6.709(-4.5%)
Mutual L_p(nH)	1.101(0.0%)	1.100(-0.1%)	1.096(-0.5%)	5.551(0.0%)	5.508(-0.1%)	5.490(-1.1%)

Table 7.1: On-chip self and mutual inductance computed via numerical extraction for two parallel wire traces with pitch-spacing being $3\mu m$, and the thickness being $1\mu m$. Columns 2-4 are inductance for wire length of $1000\mu m$, and columns 5-7 are inductance for wire length of $4000\mu m$. For each length, three wire widths are assumed, namely, $1\mu m$, $1.2\mu m$ (i.e., the width has 20% variation compared to width=$1\mu m$), and $2\mu m$ (i.e., the wire is up-sized by 100% compared to width=$1\mu m$). Percentages of inductance changes with respect to inductance for width=$1\mu m$ are also given.

though the impact must be considered for resistance and capacitance extractions as in [5]. Finally, when we up-size the wire width by 100% (see columns 4 and 7), the inductance changes by up to 5.4% for self inductance, and by 4.5% for mutual inductance. Therefore, interconnect sizing and spacing might *not* be an effective way to change inductance[3].

There are limitations of applying the two equations however. First, they do not consider the skin depth and internal inductance; Second, widths are not considered for mutual inductance. Therefore, we will propose to build tables via numerical inductance extraction for self and mutual inductance.

There are two parts in the table-based inductance extraction. One is to pre-compute inductance tables. We assume that each layer has a nominal thickness, and build tables for different layers. The self inductance table has two dimensions: width and length. The mutual inductance table has four dimensions: widths for two traces, the spacing between them, and the length. The 3D inductance extraction tool RI3 is invoked to solve a block of two traces for different combinations of lengths, widths, and spacings. The resulting self and mutual inductance is stored in tables. Note that only 2-trace subproblems need to be solved, because results to 1-trace subproblems are parts of results to 2-trace subproblems. In addition, the inductance depends on the skin depth, which is a function of frequency. We run RI3 under the significant frequency. The significant frequency is defined as $0.17/t_r$, where t_r is the minimum rising/falling time [52]. It captures the majority of high-frequency components for the signal with rising/falling time t_r.

The other part of the table-based inductance extraction is table lookup. For each trace in a block, we obtain a self inductance from tables for a given layer,

[3]On the other hand, because the coupling capacitance depends strongly on the spacing, the optimal interconnect sizing and spacing, as shown in Chapter 6, is effective to reduce RC delay (as well as capacitive coupling).

length and width. For any combination of two traces T_i and T_j, we obtain a mutual inductance from tables for a given layer, widths, and spacing between T_i and T_j. A bicubic spline algorithm [62] will be used to compute inductance that is not given in the table.

7.4 Applications of Inductance Model

7.4.1 L_{eff} for coplanar-waveguide

The coplanar-waveguide structure (a block of size $n = 3$, see Figure 7.5) is often used for on-chip clock trees in high-speed designs. Not to consider the inductive effect will lead to a significant underestimate of delay and noise. Therefore, the effective loop inductance (L_{eff}) of the signal trace needs to be computed in order to use the transmission line theory. In the following, we derive L_{eff} as a function of L_p for the three traces T_1, T_2 and T_3, where T_2 is the signal trace, and T_1 and T_3 are coplanar power/ground traces.

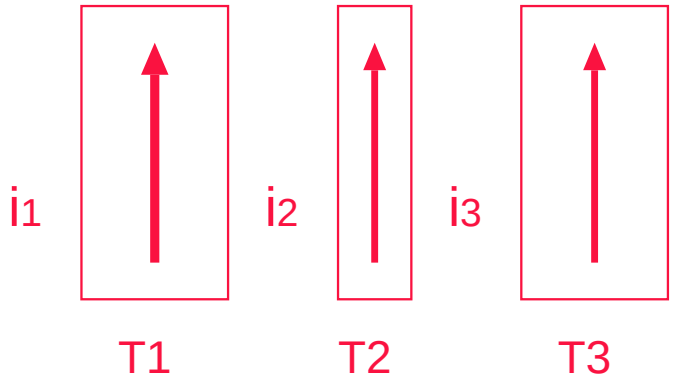


Figure 7.5: The top view of a coplanar-waveguide. T_1 and T_3 are dedicated power/ground traces.

L_{eff} is defined for the current loop that has two segments: the first segment

is current i2 through T_2; the second segment has two parallel branches, i.e., i1 through T_1 and i3 through T_3. According to the definition of L_{eff}, we have

$$\begin{aligned} \Delta V &= L_{eff}\frac{di_2}{dt} \\ &= L_{p_{22}}\frac{di_2}{dt} + L_{p_{21}}\frac{di_1}{dt} + L_{p_{23}}\frac{di_3}{dt} - L_{p_{11}}\frac{di_1}{dt} - L_{p_{12}}\frac{di_2}{dt} - L_{p_{13}}\frac{di_3}{dt} \end{aligned} \tag{7.5}$$

and the two power/ground traces have the same voltage drop

$$L_{p_{11}}\frac{di_1}{dt} + L_{p_{12}}\frac{di_2}{dt} + L_{p_{13}}\frac{di_3}{dt} = L_{p_{13}}\frac{di_1}{dt} + L_{p_{23}}\frac{di_2}{dt} + L_{p_{33}}\frac{di_3}{dt} \tag{7.6}$$

finally, the current are conservative according to KCL

$$i_1 + i_2 + i_3 = 0 \tag{7.7}$$

L_{eff} can be derived by simultaneously solving (7.5)-(7.5). When T_1 and T_3 are symmetric with respect to T_2, L_{eff} is

$$L_{eff} = L_{p_{22}} - 2L_{p_{23}} + \frac{L_{p_{11}}}{2} + \frac{L_{p_{13}}}{2} \tag{7.8}$$

Experiments show that (7.8) using our partial inductance tables gives results almost identical to L_{eff} obtained by 3D extractions using RI3. For example, we compute the L_{eff} for a coplanar-waveguide structure with all three traces $10\mu m$ wide, $2000\mu m$ long, and separated by $2\mu m$. The L_{eff} given by our formula (7.8) using table-based partial inductance is 0.839nH, while L_{eff} given by RI3 is 0.840nH. The difference can be almost ignored.

7.4.2 Bus optimization

We also have integrated the table-based inductance model with the statistically-based RC model [5] to obtain the RLC model for on-chip interconnects. As discussed in section III, we do not need to consider the impact of process variations for inductance, but we do consider the impact for resistance and capacitance

[5]. In addition, because of the assumption that traces on adjacent layers are orthogonal, there is no need to consider the mutual inductance between traces on layer N and those on layer $N+1$ or $N-1$. At the same time, trace densities on layers $N+1$ and $N-1$ are considered during capacitance extraction for traces on layer N. The resulting RLC model is represented as the SPICE netlist. The current return path will be determined automatically by SPICE.

In the following, we apply the RLC model to optimize a bus structure with 18 signal traces, and two fat power-traces outside the signal traces. The wire thickness is $2.0\mu m$, and spacing is $0.8\mu m$, both for all traces. The width is $0.8\mu m$ for all signal traces, and is $16\mu m$ for two fat power-traces. We assume the following signal pattern: all signal traces are simultaneously switching up with rising time of 80ps, except that one of the two central signal traces is the quiet victim. We also assume that all devices, including drivers, buffers and receivers, are 40x of the minimum inverter in a representative $0.18\mu m$ CMOS technology. Based on SPICE simulations, we will show how buffer insertion and shielding insertion reduce the noise under the RLC model.

A. Buffer insertion

It is well known that buffer insertion is effective to reduce the RC delay and capacitive noise. It is also very effective to reduce the inductive noise, since the original longer current return loop becomes shorter, as the current returns through the inserted buffers. Furthermore, as we discussed in the above (see Table 7.1), the inductance is a super-linear function of the wire length, more precisely, the DC-connected wire length between devices such as drivers, buffers and receivers. Therefore, while inserting a buffer at the middle point of a long wire reduces the coupling capacitance by a half for each DC-connected wire, it

reduces the mutual inductance by more than a half for each DC-connected wire.

In the following example via SPICE simulation under the RLC model, we assume that for all traces, the wire length is $4000\mu m$. We measure the noise at the far-end of the victim trace (the input node of receiver) for three cases: no buffer, one buffer, and three buffers inserted for each single trace, respectively. These buffers are inserted uniformly. Therefore, the DC-connected wire lengths are $4000\mu m$, $2000\mu m$ and $1000\mu m$ for three cases, respectively. As shown in Table 7.2, inserting one buffer reduces the noise by 21.1%, and inserting three buffers reduces the noise by 42.3%. Note that the noise is measure at the inputs of devices for the victim trace. Much less of noise will be observed if the measurement is made at the output of devices.

number of buffer inserted	0	1	3
DC-Connected wire length (μm)	4000	2000	1000
Noise (V)	0.71(0.0%)	0.56(-21.1%)	0.41(-42.3%)

Table 7.2: Comparison of noise between different buffer insertion solutions.

B. Shielding insertion

A shielding trace is a wire directly connected to power or ground networks. It can provide the dedicated current return path to reduce the inductive noise. In the following experiment, we assume that for all traces, the length is $4000\mu m$, and no buffers are inserted. We again measure the noise at the far-end of the victim trace (the input node of receiver) via SPICE simulation. We will insert shielding traces to make the far-end noise of the victim trace less than 0.25V.

When there is no shielding traces, the noise of victim trace is 0.71V. Then,

we insert a shielding trace for every six signal traces, and increase the width W_s for shielding traces from $0.8\mu m$ to $2.4\mu m$. The noise is 0.22V with $W_s = 2.4\mu m$. Finally, we insert a shielding trace for every three traces. The noise is 0.17V when $W_s = 0.8\mu m$. As shown in Table 7.3, there is a clear trade-off between area and noise: we may reduce the noise by a factor of 4.2x while the total width, the sum of the total wire width and total spacing, of the bus structure is increased by 13%, and the total wire width is increased by 8.8%.

N_s	W_s	Noise (V)	total width (μm)	wire width (μm)
18	–	0.71	61.6	46.4
6	0.8	0.38	64.8	46.0
6	1.6	0.27	64.4	49.6
6	2.4	0.22	68.0	51.2
3	0.8	0.17	69.6	50.4

Table 7.3: Comparison of noise between different shielding insertion solutions. Column one (N_s) is the number of signal traces between two shielding trace, and column 2 (W_s) is the width for the shielding traces. Column 3 is the total width (including the total spacing) of the the bus structure, and column 4 is the total wire width.

7.4.3 Extensions to Model Power/Ground Planes and MCM/PCB Designs

Same as the shielding trace, the power/ground plane can be used to provide the dedicated current return path and to reduce the inductance effect. Very recent, it is reported that the Alpha 21264 microprocessor uses entire metal layers as power/ground planes to supply power and reduce coupling [40]. Because using

entire layer is extremely expensive, partial metal layer may also be used. The on-chip interconnects become micro-strip lines if there is only one power/ground plane, or strip lines if there are two power/ground planes sandwiching interconnects.

Foundations 6 and 7 still valid if we *explicitly* model the power/ground planes using the PEEC model. It leads to a model with a high complexity however. Recently, the two foundations have been extended to compute the loop inductance for the micro-strip and strip lines in a similar fashion[4]. Because the loop inductance has taken the effect of power/ground planes into account, there is no need to explicitly model the inductance for power/ground planes. More detailed study on the characteristics and optimization of inductance for on-chip micro-strip and strip lines is planned as a future work.

The MCM/PCB design often uses the micro-strip and strip lines for off-chip interconnects. The extended methodology can be used to consider these off-chip interconnects. To illustrate the inductance characteristics for off-chip interconnects, we compute the loop inductance for the strip lines in the MMS MCM technology that is available through the MCM Interconnect Designer's Access Service from Information Science Institute. We use two wires between one power plane and one ground plane. Both wires are $4\mu m$ thick, and are $12\mu m$ and $21.5\mu m$ away from the power plane and ground plane, respectively. The pitch spacings between the two wires are $62\mu m$ and $75\mu m$, respectively. We report loop inductance under different wire widths and lengths in Table 7.4. Columns 2-4 are inductance for wires of 1cm, and columns 5-7 are inductance for wires of 4cm. We observe the following: First, the inductance is again not linearly scalable with respect to length. For example, when pitch-spacing is $62\mu m$, wire width is $19\mu m$

[4]Private discussion with Dr. Norman Chang and Dr. Shen Lin at Hewlett-Packard Laboratories.

pitch-spacing	length	$1cm$			$4cm$		
$62\mu m$	width(μm)	19	23	38	19	23	38
	Self L_{loop}(nH)	7.020(0.0%)	6.705(-4.5%)	6.115(-13%)	31.72(0.0%)	30.47(-4.1%)	28.11(-11%)
	Mutual L_{loop}(nH)	2.416(0.0%)	2.413(-0.1%)	2.401(-0.6%)	13.30(0.0%)	13.29(-0.1%)	13.24(-0.5%)
$75\mu m$	width(μm)	19	23	38	19	23	38
	Self L_{loop}(nH)	7.002(0.0%)	6.682(-5.0%)	6.094(-13%)	31.66(0.0%)	30.39(-4.0%)	28.03(-11%)
	Mutual L_{loop}(nH)	2.179(0.0%)	2.178(-0.1%)	2.174(-0.1%)	12.35(0.0%)	12.35(-0.1%)	12.33(-0.1%)

Table 7.4: Loop inductance computed via numerical extraction for two parallel wire traces in an MCM design. Both wires are $4\mu m$ thick, and are $12\mu m$ and $21.5\mu m$ away from the power plane and ground plane, respectively. The current is assumed to return from power/ground planes. Columns 2-4 are inductance for wire length of $1cm$, and columns 5-7 are inductance for wire length of $4cm$. For each length, three wire widths are assumed, namely, $19\mu m$, $23\mu m$ (i.e., the width has 20% variation compared to width=$19\mu m$), and $38\mu m$ (i.e., the wire is up-sized by 100% compared to width=$19\mu m$). Percentages of inductance changes with respect to inductance values for width=$19\mu m$ are also given.

and length increases from $1cm$ to $4cm$, the self inductance increases by a factor of 4.5x rather than 4x. Second, the mutual inductance is *not* sensitive to the change of the wire width for the given pitch-spacing, but does have certain change for different pitch-spacings. As shown in the table for pitch-spacing being $62\mu m$, a 100% increase of the wire width only leads to less than 1% change of the mutual inductance[5], but when we increase the pitch-spacing from $62\mu m$ to $75\mu m$, the mutual inductance decreases for about 10%. Finally, the self inductance is more sensitive to the wire width change, but not the pitch-spacing change. A 20% increase of the wire width reduces the self inductance by up to 4.5% and a 100% increase of the wire width reduces the self inductance by up to 13%, but when we increase the pitch-spacing from $62\mu m$ to $75\mu m$, the self inductance decreases only for about 1%.

7.5 Discussions and Conclusions

In this chapter, we have proposed two foundations that can be used to reduce the problem size for the on-chip inductance extraction problem. We have also presented a table-based inductance extraction methodology, which is efficient and accurate, and can be used for iterative layout optimization and verification procedure.

We have applied the table-based inductance model to compute L_{eff} for coplanar-waveguide, and to generate RLC models for on-chip interconnects. The RLC model has been used to optimize bus structures via SPICE simulations. Experiments have shown that both buffer insertion and shielding insertion are effective to reduce the inductive noise. In the future, we plan to develop optimal algo-

[5] It implies that the power/ground planes are effective to shield the coupling effect between parallel wires.

rithms for the optimal buffer insertion problem and the optimal shielding insertion problem, both under the distributed RLC model. Furthermore, the simultaneous buffer and shielding insertion problem will be studied to explore the trade-off between the device and interconnect costs to minimize the delay and noise under the distributed RLC model. Moreover, we will extend the simultaneous device sizing, and interconnect sizing and spacing problem, which is solved under the distributed RC model for performance optimization in Chapter 6, to the distributed RLC model for both performance and signal-integrity optimization.

Our inductance model in this chapter considers traces only on a single layer. It is assumed that for traces on layer N, the current return path via layer $N+2$ or $N-2$ has a very high impedance, therefore the lion's share of the current returns via loops in the same layer. In addition, layers $N+2$ and $N-2$ are statistically quiet. Therefore, it is acceptable to consider only layer N. Further study on the impact of layer $N+2$ or $N-2$ is planned.

Our Foundations 6 and 7 can be efficiently applied to model on-chip interconnects without the presence of power/ground planes. We have also briefly discussed the ongoing extensions to consider the power/ground planes. The extended results are applicable to both on-chip and off-chip interconnects.

CHAPTER 8

Conclusions and Discussions

As very large scale integrated (*VLSI*) circuits move into the era of deep-submicron (*DSM*) technology and gigahertz clock frequency, the system performance has increasingly become dominated by the interconnect delay. In this dissertation, we have presented five research topics on interconnect modeling and optimization, as well as optimization theory.

In Chapter 2, we have studied the multi-source wire sizing (*MSWS*) problem. Given a routing tree with multiple sources, the MSWS problem determines the optimal widths of the wire segments such that the weighted sum of delays between different source-sink pairs is minimized. We have revealed several interesting properties for the optimal MSWS solution, of which the most important is the bundled refinement property. Based on this property, we have proposed a polynomial-time algorithm, which uses iterative bundled-refinement operations to compute lower and upper bounds of an optimal solution. Because the algorithm often achieves identical lower and upper bounds in experiments, the optimal solution is obtained simply by the bound computation. Experiments based on SPICE simulations have convincingly shown that wire sizing is effective to reduce the interconnect delay for the routing tree with multiple sources. Furthermore, this new wire sizing algorithm can be used for single-source wire sizing problem and runs 100x faster than previous methods. It has replaced previous single-source wire sizing methods in practice.

In Chapter 3, we have solved the interconnect capacitance extraction problem to compute the capacitance values from complex 3-dimensional interconnect structures. We have shown how basic drivers in process technology (planarization and minimum metal density requirements) actually simplify the extraction problem; we do this by proposing and validating five "foundations" through detailed experiments with a 3-dimensional field solver on representative $0.50\mu m$, $0.35\mu m$ and $0.18\mu m$ process parameters. We have further presented a simple yet accurate 2 1/2-dimensional extraction methodology directly based on the foundations. This methodology has been productized and is being shipped with the Cadence Silicon Ensemble 5.0 product. Moreover, this methodology can also be used for MCM/PCB designs.

In Chapter 4, we have developed the theory and algorithm for the local-refinement based optimization. We have formulated three classes of optimization problems: the simple, monotonically constrained, and bounded CH-problems. We have revealed that the dominance property holds for those CH-problems under different types of local-refinement operations. This property immediately leads to an efficient polynomial-time algorithm, which provides a unified solution to a number of interconnect optimization problems, including the MSWS problem, as well as STIS and GISS problems presented in Chapters 5 and 6, respectively.

In Chapter 5, we have solved the STIS (i.e., simultaneous transistor and interconnect sizing) problem under the accurate device model. The problem assigns optimal wire widths to interconnects and optimal sizes to transistors for minimizing the delay for multiple critical paths. We have shown that the STIS problem is in general a bounded CH-program and can be solved by the local-refinement based algorithm presented in Chapter 4. According to SPICE simulations, significant delay reduction has been achieved by using the device

model more accurate than many used in existing interconnect optimization works.

Both the MSWS problem in Chapter 2 and the STIS problem in Chapter 5 are aimed to address the resistive effect of the interconnect. This effect is often ignored when the minimum transistor size is larger than the submicron that is $\leq 0.35\mu m$. Furthermore, our study of interconnect capacitance in Chapter 3 shows that, rather than conventional ground capacitance, the coupling capacitance between neighboring wires becomes the dominant capacitance component for the DSM designs. The dominant coupling capacitance has a great impact on both performance and signal integrity in the DSM designs.

In Chapter 6, we have studied the global interconnect sizing and spacing (GISS) problem to address the impact of the coupling capacitance on the interconnect delay. Given the topology for multiple nets, the GISS problem finds the wire sizing and spacing solution *simultaneously* for all nets, with consideration of coupling capacitance between them. We have formulated the GISS problem based on the concept of asymmetric wire sizing. The formulation can be posed as a CH-program, which directly leads to an effective and efficient solution. We have shown convincingly that the simultaneous wire sizing and spacing formulation and algorithm can significantly reduce the interconnect delay compared to the formulation using wire sizing only.

Note that the MSWS and GISS formulations can be viewed as parts of the STIS formulations, and the solutions to the three problems are all based on different types of local-refinement operations[1]. Therefore, we have a unified formulation and solution to minimize the interconnect delay under the distributed RC model with consideration of simultaneous device sizing, and wire sizing and spacing. We also use accurate models for both device delay and interconnect

[1]The bundled refinement operation for the MSWS algorithm is also a type of local-refinement operation as discussed in Chapter 4.

capacitance (including the coupling capacitance). Solutions to the three interconnect optimization problems, as well as the interconnect extraction problem, have been integrated in the TRIO package [24][2].

Due to increasingly wider and longer wire traces, faster clock frequencies and shorter rising times, the inductance effect of on-chip interconnects becomes an emerging problem for DSM designs with gigahertz clock frequencies. In Chapter 7, we have investigated the interconnect inductance extraction problem. It computes the inductance values from three-dimensional interconnect structures. We have proposed an efficient yet accurate table-based approach using the concept of partial inductance. This approach has been used to generate RLC models for on-chip interconnects with consideration of process variations, and is being used in the state-of-the-art microprocessor designs in Hewlett-Packard Company. We have also shown that both buffer insertion and shielding insertion are effective to reduce the inductive noise. However, the automatic interconnect optimization considering the inductance effect is still an open problem. To solve the problem, we plan to develop optimal algorithms under the distributed RLC model or the lossy transmission line model to optimize both performance and signal integrity.

[2]TRIO package is available at http://cadlab.cs.ucla.edu/ ~trio.

CHAPTER 9

Appendix: Proofs for Properties of Optimal MSWS Solutions

In this appendix, we present the proofs for properties of optimal multi-source wiresizing solutions. These properties are described in Chapter 2.

Since the proofs for the LST separability (Theorem 1), the LST monotone property (Theorem 2), the dominance property (Theorem 4) and Theorem 7 concerning the complexities of GWSA and BWSA algorithms are similar to those in [29], the full proofs of these theorems are given in [15]. In this appendix, we first discuss the properties for the coefficient functions and then prove the SST local monotone property (Theorem 3), the bundled refinement property (Theorem 5) and Theorem 6 concerning the optimality of the BWSA algorithm.

9.1 Properties of Coefficient Functions

Careful study of the definitions of f^{ij}, g^{ij} and h^{ij} in (2.4)–(2.6), as well as F, G and H in (2.10)–(2.12) reveals Lemma 1 presented in Section 2.2 and the following Lemmas 3 and 4 for the coefficient functions F, G and H.

Lemma 3 *Given an MSIT and a segment S in the MSIT, for any uni-segments E and E', if E is in segment S, and E' in segment $S'(\neq S')$, $F(E, E')$ is an invariant (denoted $F(S, S')$).*

Lemma 4 *Given an MSIT and a segment S in the MSIT, for any uni-segment E within segment S, G(E) and H(E) are invariants (denoted G(S) and H(S), respectively).*

Although the coefficient functions F, G and H are defined for uni-segments in (2.10)–(2.12), Lemmas 1–4 enable us to compute these functions based on segments rather than uni-segments. Because the number of segments in an MSIT may be much smaller than the number of uni-segments in the MSIT, we can compute these coefficient functions for much reduced costs. These coefficient functions will be computed before the wiresizing procedure and viewed as constants during the wiresizing procedure.

9.2 Proof of Theorem 3

We shall prove Theorem 3 (the SST local monotone property) based on a series of lemmas. For simplicity, we assume the finest segment division in this proof. Recall that a uni-segment under any valid segment-division corresponds to a single or multiple uni-segments in the finest segment-division, it is easy to verify that the local monotone property holds for any valid segment-division if and only if it holds for the finest segment-division.

Lemma 5 *Given an MSIT and a segment S in the MSIT, if E_l and E_r are two adjacent uni-segments within segment S and E_l is just left to E_r, then*

$$F(E_l, E) = F(E_r, E) \quad if\ E \neq E_l \neq E_r$$
$$F(E, E_l) = F(E, E_r) \quad if\ E \neq E_l \neq E_r$$

Proof: There are two cases for Lemma 5.

Case 1: E is in segment S, according to Lemma 1,

if E is right to E_l (as well as E_r), $F(E_l, E) = F(E_r, E) = F_l(S)$ and $F(E, E_l) = F(E, E_r) = F_r(S)$;

if E is left to E_l (as well as E_r), $F(E_l, E) = F(E_r, E) = F_r(S)$ and $F(E, E_l) = F(E, E_r) = F_l(S)$.

Case 2: E is in segment $S'(\neq S)$, according to Lemma, 3

$F(E_l, E) = F(E_r, E) = F(S, S')$ and $F(E, E_l) = F(E, E_r) = F(S', S)$. □

Lemma 6 *Given an MSIT and a segment S in the MSIT, let E_l and E_r be uni-segments in segment S with E_l just left to E_r, concerning the optimal wiresizing solution, if $F_l(S) > F_r(S)$, then E_l can not be narrower than E_r; if $F_l(S) < F_r(S)$, then E_l can not be wider than E_r.*

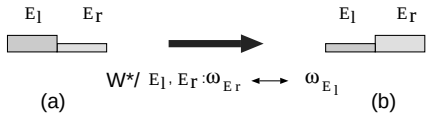


Figure 9.1: (a) The width assignments for E_l and E_r in the optimal solution $\mathcal{W}^*$. (b) The wiresizing solution obtained by swapping the width assignments for E_l and E_r.

Proof: Let $M = MSIT - \{E_l, E_r\}$ and $\overline{\mathcal{W}}$ be the wiresizing solution defined on M by $\mathcal{W}$. The objective function (2.7) can be written as:

$$
\begin{aligned}
& t(MSIT, \mathcal{E}, \mathcal{W}) \\
= \quad & t(M, \mathcal{E}, \overline{\mathcal{W}}) + \mathcal{K}_1 \cdot (w_{E_l} + w_{E_r}) + \\
& \mathcal{K}_2 \cdot \sum_{E' \in MSIT} F(E_l, E') \cdot \frac{w_{E'}}{w_{E_l}} + \mathcal{K}_2 \cdot \sum_{E \in MSIT} F(E, E_l) \cdot \frac{w_{E_l}}{w_E} + \\
& \mathcal{K}_2 \cdot \sum_{E' \in MSIT} F(E_r, E') \cdot \frac{w_{E'}}{w_{E_r}} + \mathcal{K}_2 \cdot \sum_{E \in MSIT} F(E, E_r) \cdot \frac{w_{E_r}}{w_E} + \\
& \mathcal{K}_3 \cdot \sum_{E' \in MSIT} F(E_l, E') \cdot \frac{1}{w_{E_l}} + \mathcal{K}_4 \cdot G(E_l) \cdot \frac{1}{w_{E_l}} +
\end{aligned}
$$

$$\mathcal{K}_5 \cdot H(E_l) \cdot \frac{1}{w_{E_l}} \ + \ \mathcal{K}_3 \cdot \sum_{E' \in MSIT} F(E_r, E') \cdot \frac{1}{w_{E_r}} \ + \ \mathcal{K}_4 \cdot G(E_r) \cdot \frac{1}{w_{E_r}} \ + \ \mathcal{K}_5 \cdot H(E_r) \cdot \frac{1}{w_{E_r}} \tag{9.1}$$

Let $\mathcal{W}^*$ be the optimal wiresizing solution. After swapping the width assignments for E_l and E_r with respect to $\mathcal{W}^*$ (see Fig. 9.1), we denote the resulted wiresizing solution $\mathcal{W}^*/E_l, E_r : w_{E_l} \leftrightarrow w_{E_r}$.

According to Lemmas 4 and 5

$$G(E_l) \ = \ G(E_r) \ = \ G(S) \tag{9.2}$$

$$H(E_l) \ = \ H(E_r) \ = \ H(S) \tag{9.3}$$

$$F(E_l, E) \ = \ F(E_r, E) \quad if \ E \ \neq E_r \ \neq E_l \tag{9.4}$$

$$F(E, E_l) \ = \ F(E, E_r) \quad if \ E \ \neq E_r \ \neq E_l \tag{9.5}$$

thus,

$$\begin{aligned} & t(MSIT, \mathcal{E}, \mathcal{W}^*/E_l, E_r : w_{E_l} \leftrightarrow w_{E_r}) \ - \ t(MSIT, \mathcal{E}, \mathcal{W}^*) \\ = \quad & \mathcal{K}_2 \cdot F(E_l, E_r) \cdot (\frac{w^*_{E_l}}{w^*_{E_r}} - \frac{w^*_{E_r}}{w^*_{E_l}}) \ + \ \mathcal{K}_2 \cdot F(E_r, E_l) \cdot (\frac{w^*_{E_r}}{w^*_{E_l}} - \frac{w^*_{E_l}}{w^*_{E_r}}) \ + \\ & \mathcal{K}_3 \cdot F(E_l, E_r) \cdot (\frac{1}{w^*_{E_r}} - \frac{1}{w^*_{E_l}}) \ + \ \mathcal{K}_3 \cdot F(E_r, E_l) \cdot (\frac{1}{w^*_{E_l}} - \frac{1}{w^*_{E_r}}) \\ = \quad & \{F(E_l, E_r) \ - \ F(E_r, E_l)\} \cdot \{w^*_{E_l} \ - \ w^*_{E_r}\} \cdot \\ & \{\mathcal{K}_2 \cdot \frac{(w^*_{E_l} \ + \ w^*_{E_r}) \ + \ \mathcal{K}_3}{w^*_{E_l} \cdot w^*_{E_r}}\} \end{aligned} \tag{9.6}$$

We know that $\frac{(w^*_{E_l} \ + \ w^*_{E_r}) \ + \ \mathcal{K}_3}{w^*_{E_l} \cdot w^*_{E_r}} > 0$ and $(9.6) \geq 0$ since $\mathcal{W}^*$ is the optimal solution. Clearly, if $F_l(S) > F_r(S)$, according to Lemma 1, $F(E_l, E_r) > F(E_r, E_l)$, then we have $w^*_{E_l} \geq w^*_{E_r}$. Similarly, if $F_l(S) < F_r(S)$, then $F(E_l, E_r) < F(E_r, E_l)$, so we have $w^*_{E_l} \leq w^*_{E_r}$. As a result, Lemma 6 holds. □

By applying Lemma 6 to any adjacent uni-segments in a segment, we obtain Lemma 7.

Lemma 7 *Given an MSIT and a segment S in the MSIT, concerning the optimal wiresizing solution, if* $F_l(S) > F_r(S)$*, then the wire widths within segment S decrease monotonically rightward. Similarly, they increase monotonically rightward if* $F_l(S) < F_r(S)$*.*

Lemma 8 *Given an MSIT and any segment S in the MSIT, if* $F_l(S) = F_r(S)$*, there exists an optimal wiresizing such that all uni-segments in segment S have the same wire width.*

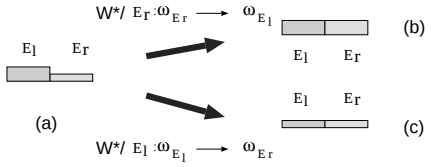


Figure 9.2: (a) The width assignments for E_l and E_r in the optimal solution $\mathcal{W}^*$. (b) The wiresizing solution obtained by replacing the width of E_r with that of E_l. (c) The wiresizing solution obtained by replacing the width of E_l with that of E_r.

Proof: Assume that Lemma 8 fails for an MSIT, then for any optimal solution $\mathcal{W}^*$, there must exist two uni-segments in a segment S of the MSIT such that E_l is just left to E_r and $w^*_{E_l} \neq w^*_{E_r}$. Since $\mathcal{W}^*$ is optimal, the increase in the objective function when we change the width of E_r in $\mathcal{W}^*$ from $w^*_{E_r}$ to $w^*_{E_l}$ (see Figure 9.2.b), by using (9.1), is:

$$\begin{aligned}
\Delta t1 &= t(MSIT, \mathcal{E}, \mathcal{W}^*/E_r : w_{E_r} \rightarrow w_{E_l}) - t(MSIT, \mathcal{E}, \mathcal{W}^*) \\
&= \mathcal{K}_1 \cdot (w_{E_l} - w_{E_r}) + \mathcal{K}_2 \cdot \sum_{E' \in MSIT} F(E_r, E') \cdot (\frac{w_{E'}}{w_{E_l}} - \frac{w_{E'}}{w_{E_r}}) + \\
&\quad \mathcal{K}_2 \cdot \sum_{E \in MSIT} F(E, E_r) \cdot (\frac{w_{E_l}}{w_E} - \frac{w_{E_r}}{w_E}) +
\end{aligned}$$

$$
\begin{aligned}
& \mathcal{K}_3 \cdot \sum_{E' \in MSIT} F(E_r, E') \cdot (\frac{1}{w_{E_l}} - \frac{1}{w_{E_r}}) \ + \\
& \mathcal{K}_4 \cdot G(E_r) \cdot (\frac{1}{w_{E_l}} - \frac{1}{w_{E_r}}) \ + \ \mathcal{K}_5 \cdot H(E_r) \cdot (\frac{1}{w_{E_l}} - \frac{1}{w_{E_r}}) \\
\geq \ & 0 \qquad (9.7)
\end{aligned}
$$

Similarly, the increase in the objective function when change the width of E_l in $\mathcal{W}^*$ from $w^*_{E_l}$ to $w^*_{E_r}$ (see Figure 9.2.c) is:

$$
\begin{aligned}
\Delta t2 \ &= \ t(MSIT, \mathcal{E}, \mathcal{W}^*/E_l : w_{E_l} \rightarrow w_{E_r}) \ - \ t(MSIT, \mathcal{E}, \mathcal{W}^*) \\
&= \ \mathcal{K}_1 \cdot (w_{E_r} - w_{E_l}) \ + \ \mathcal{K}_2 \cdot \sum_{E' \in MSIT} F(E_l, E') \cdot (\frac{w_{E'}}{w_{E_r}} - \frac{w_{E'}}{w_{E_l}}) \ + \\
& \quad \mathcal{K}_2 \cdot \sum_{E \in MSIT} F(E, E_l) \cdot (\frac{w_{E_r}}{w_E} - \frac{w_{E_l}}{w_E}) \ + \\
& \quad \mathcal{K}_3 \cdot \sum_{E' \in MSIT} F(E_l, E') \cdot (\frac{1}{w_{E_r}} - \frac{1}{w_{E_l}}) \ + \\
& \quad \mathcal{K}_4 \cdot G(E_l) \cdot (\frac{1}{w_{E_r}} - \frac{1}{w_{E_l}}) \ + \ \mathcal{K}_5 \cdot H(E_l) \cdot (\frac{1}{w_{E_r}} - \frac{1}{w_{E_l}}) \\
&\geq \ 0 \qquad (9.8)
\end{aligned}
$$

Recall (9.2)–(9.5), it is not difficult to verify the following:

$$\Delta t1 \ + \ \Delta t2 \ = \ 0 \qquad (9.9)$$

According to (9.7)–(9.9), $\Delta t1 \ = \ \Delta t2 = 0$. That is, both $\mathcal{W}^*/E_r : w_{E_r} \rightarrow w_{E_l}$ and $\mathcal{W}^*/E_l : w_{E_l} \rightarrow w_{E_r}$ are optimal.

Therefore, if there is an optimal solution $\mathcal{W}^*$ where the wire widths are not uniform in a segment S, let w^*_l be the wire width of the leftmost uni-segment in S, from left to right, we can successively replace the wire width for every uni-segment in S by w^*_l, without increase in the objective function. That is, the resulting wiresizing solution is an optimal wiresizing solution such that the wire widths are uniform in segment S. □

In conclusion, we obtain the SST local monotone property (Theorem 3) by combining Lemmas 7 and 8,

9.3 Proof of Theorem 5

In order to prove Theorem 5 (the bundled refinement property), we define the following equations with respect to any particular uni-segment E:

$$\begin{aligned}
\Psi(MSIT, \mathcal{E}, E, \mathcal{W}) &= \mathcal{K}_1 \cdot \sum_{E' \in MSIT - \{E\}} w_{E'} \; + \\
&\quad \mathcal{K}_2 \cdot \sum_{E', E'' \in MSIT - \{E\}, E' \neq E''} F(E', E'') \cdot \frac{w_{E''}}{w_{E'}} \; + \\
&\quad \mathcal{K}_3 \cdot \sum_{E', E'' \in MSIT - \{E\}, E' \neq E''} F(E', E'') \cdot \frac{1}{w_{E'}} \; + \\
&\quad \mathcal{K}_4 \cdot \sum_{E' \in MSIT - \{E\}} G(E') \cdot \frac{1}{w_{E'}} \; + \\
&\quad \mathcal{K}_5 \cdot \sum_{E' \in MSIT - \{E\}} H(E') \cdot \frac{1}{w_{E'}} \qquad (9.10) \\
\Phi(MSIT, \mathcal{E}, E, \mathcal{W}) &= \mathcal{K}_1 \; + \; \mathcal{K}_2 \cdot \sum_{E' \in MSIT - \{E\}} F(E', E) \cdot \frac{1}{w_{E'}} \qquad (9.11) \\
\Theta(MSIT, \mathcal{E}, E, \mathcal{W}) &= \mathcal{K}_2 \cdot \sum_{E' \in MSIT - \{E\}} F(E, E') \cdot w_{E'} \; + \\
&\quad \mathcal{K}_3 \cdot \sum_{E' \in MSIT - \{E\}} F(E, E') \; + \\
&\quad \mathcal{K}_4 \cdot G(E) \; + \; \mathcal{K}_5 \cdot H(E) \qquad (9.12)
\end{aligned}$$

Then, we can then rewrite the objective function (2.7) as follows:

$$\begin{aligned}
t(MSIT, \mathcal{E}, \mathcal{W}) &= \Psi(MSIT, \mathcal{E}, E, \mathcal{W}) \; + \; \Phi(MSIT, \mathcal{E}, E, \mathcal{W}) \cdot w_E \; + \\
&\quad \Theta(MSIT, \mathcal{E}, E, \mathcal{W}) \cdot \frac{1}{w_E} \qquad (9.13)
\end{aligned}$$

We show the following Lemma 9:

Lemma 9 *Given an MSIT, a segment-division $\mathcal{E}$ and a wiresizing solution $\mathcal{W}$, for any particular uni-segment E under $\mathcal{E}$, if we divide E into a sequence of uni-segments $E_1, E_2, \cdots$, and E_k, let each new uni-segment inherits the wire width*

assignment of E, and denote the resulting segment-division and wiresizing solution $\mathcal{E}'$ and $\mathcal{W}'$, respectively, then the following relations hold for any uni-segment E' other than E.

$\Psi(MSIT, \mathcal{E}, E', \mathcal{W}) = \Psi(MSIT, \mathcal{E}', E', \mathcal{W}')$

$\Phi(MSIT, \mathcal{E}, E', \mathcal{W}) = \Phi(MSIT, \mathcal{E}', E', \mathcal{W}')$

$\Theta(MSIT, \mathcal{E}, E', \mathcal{W}) = \Theta(MSIT, \mathcal{E}', E', \mathcal{W}')$ [1]

Proof: It is not difficult to verify that Lemma 9 is true if the follows hold:

$$F(E_1, E') = F(E_2, E') = \cdots = F(E, E')$$
$$F(E', E_1) = F(E', E_2) = \cdots = F(E', E)$$
$$G(E_1) = G(E_2) = \cdots = G(E)$$
$$H(E_1) = H(E_2) = \cdots = H(E)$$

Assuming uni-segment E is in segment S. There are two cases for uni-segment E':

Case 1: E' is also in the same segment S, according to Lemma 1, if E is left to E',

$$F(E_1, E') = F(E_2, E') = \cdots = F(E, E') = F_l(S)$$
$$F(E', E_1) = F(E', E_2) = \cdots = F(E', E) = F_r(S)$$

if E is right to E',

$$F(E_1, E') = F(E_2, E') = \cdots = F(E, E') = F_r(S)$$
$$F(E', E_1) = F(E', E_2) = \cdots = F(E', E) = F_l(S)$$

[1] In general, under the modeling method used in this work and [29, 73], for the Elmore delay t^{ij} between source N_i and sink N_j, we have $t^{ij}(\mathcal{E}, \mathcal{W}) = t^{ij}(\mathcal{E}', \mathcal{W}')$. I.e., the Elmore delay is independent of the segment-division when given the wiresizing solution.

Case 2: E' is in segment S' different from segment S, according to Lemma 3

$$F(E_1, E') = F(E_2, E') = \cdots = F(E, E') = F(S, S')$$
$$F(E', E_1) = F(E', E_2) = \cdots = F(E', E) = F(S', S)$$

Again assuming uni-segment E is in segment S, according to Lemma 4

$$G(E_1) = G(E_2) = \cdots = G(E) = G(S)$$
$$H(E_1) = H(E_2) = \cdots = H(E) = H(S)$$

As a result, Lemma 9 holds. □

Recall the definition for the local refinement operation, according to Lemma 9 and (9.13), we can conclude that the following Lemma 10 holds.

Lemma 10 *When given the wiresizing solution $\mathcal{W}$ and any particular uni-segment E, the local refinement result for E with respect to $\mathcal{W}$ is independent of the segment-division for uni-segments other than E.*

We give the following proof for Theorem 5 (the bundled refinement property).

Proof: For any particular uni-segment E under the current segment-division $\mathcal{E}$ in segment S of an $MSIT$, it may be divided into k uni-segments under the finest segment-division $\mathcal{E}_F$. From left to right, let them be $E_l = E_{F1}, E_{F2}, E_{F3}, ..., E_{Fk} = E_r$.

Without loss of generality, we assume $F_l(S) \geq F_r(S)$. For a wiresizing solution $\mathcal{W}$ which dominates the optimal solution $\mathcal{W}^*$, let $\mathcal{W}^b$ be the wiresizing solution after performing an BRU operation of $\mathcal{W}$ on E under $\mathcal{E}$. Then, we have

$w_l^b = w_{F2}^b = w_{F3}^b = ... = w_r^b$ according to the definition of the BRU operation. Meanwhile, in the optimal wiresizing solution $\mathcal{W}^*$, we have $w_l^* \geq w_{F2}^* \geq w_{F3}^* \geq ... \geq w_r^*$ according to the local monotone property.

According to Lemma 10, w_l^b is also the local refinement result for uni-segment E_l under the finest segment-division $\mathcal{E}_F$. Therefore, $w_l^b \geq w_l^*$. As a result, $w_l^b = w_{F2}^b = w_{F3}^b = ... = w_r^b \geq w_l^* \geq w_{F2}^* \geq w_{F3}^* \geq ... \geq w_r^*$. Recall that the bundled refinement of uni-segment E does not change the wire width in the wiresizing solution $\mathcal{W}$ (dominating $\mathcal{W}^*$) for any uni-segment E' other than $E_l, E_{F2}, E_{F3}, ..., E_r$. Thus, $\mathcal{W}^b$ still dominates $\mathcal{W}^*$.

The BRL case can be proved in a similar way. □

9.4 Proof of Theorem 6

Theorem 6 *The lower and upper bounds provided by BWSA are $\mathcal{E}_F$-tight.*

Proof: Let $\mathcal{E}$ be the wiresizing segment-division after BWSA. For any uni-segment E under $\mathcal{E}$, l_E is longer than $minLength$ (the length for all uni-segments under the finest segment-division $\mathcal{E}_F$) if and only if E is a *convergent* uni-segment, whose bounds can not be tightened any more.

If E is $minLength$ long, according to Lemma 10, its lower and upper bounds given by the bundled refinement operations are same as those given by local refinement operations under $\mathcal{E}_F$. Thus, if the bundled refinement operations can not tighten the lower and upper bounds, neither can the local refinement operations under $\mathcal{E}_F$.

In conclusion, Theorem 6 holds. □

References

[1] N. D. Arora, K. V. Raol, R. Schumann, and L. M. Richardson. Modeling and extraction of interconnect capacitances for multilayer vlsi circuits. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 15(1):58–67, Jan. 1996.

[2] E. Barke. Line-to-ground capacitance calculation for vlsi: A comparison. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 7(2):295–298, 1988.

[3] M. Basel. Accurate and efficient extraction of interconnect circuits for full-chip timing analysis. In *Proc. WESCON*, pages 118–123, 1995.

[4] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins. Rectilinear Steiner trees with minimum Elmore delay. In *Proc. Design Automation Conf*, pages 381–386, 1994.

[5] N. Chang, V. Kanevsky, O. S. Nakagawa, K. Rahmat, and S.-Y. Oh. Fast generation of statistically-based worst-case modeling of on-chip interconnect. In *IEEE ICCD*, 1997.

[6] C. P. Chen, Y. W. Chang, and D. F. Wong. Fast performance-driven optimization for buffered clock trees based on Lagrangian relaxation. In *Proc. Design Automation Conf*, pages 405–408, 1996.

[7] C. P. Chen, Y. P. Chen, and D. F. Wong. Optimal wire-sizing formula under the Elmore delay model. In *Proc. Design Automation Conf*, pages 487–490, 1996.

[8] C. P. Chen and D. F. Wong. A fast algorithm for optimal wire-sizing under elmore delay model. In *Proc. IEEE Int. Symp. on Circuits and Systems*, pages 412–415, 1996.

[9] J. Chern, J. Huang, L. Aldredge, P. Li, and P. Yang. Multilevel metal capacitance models for interconnect capacitances. *IEEE Electron Device Lett.*, EDL-14:32–43, 1992.

[10] C. Chien, P. Yang, E. Cohen, R. Jain, and H. Samueli. A 12.7Mchip/s all-digital BPSK direct sequence spread-spectrum IF transceiver in 1.2μm CMOS. In *Proc. IEEE Int. Solid-State Circuits Conf.*, pages 30–31, 1994.

[11] U. Choudhury and A. Sangiovanni-Vincentelli. Automatic generation of analytical models for interconnect capacitances. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 14(4):470–480, April 1995.

[12] C. Chu and D. F. Wong. A new approach to simultaneous buffer insertion and wire sizing. In *Proc. Int. Conf. on Computer Aided Design*, pages 614–621, 1997.

[13] C. Chu and D. F. Wong. Greedy wire-sizing is linear time. In *Proc. Int. Symp. on Physical Design*, pages 39–44, 1998.

[14] J. Cong and L. He. Optimal wiresizing for interconnects with multiple sources. In *Proc. Int. Conf. on Computer Aided Design*, pages 568–574, November 1995.

[15] J. Cong and L. He. Optimal wiresizing for interconnects with multiple sources. Technical Report 950031, UCLA CS Dept, August 1995.

[16] J. Cong and L. He. Optimal wiresizing for interconnects with multiple sources. *ACM Trans. on Design Automation of Electronics Systems*, 1(4):478–511, October 1996.

[17] J. Cong and L. He. Theory and algorithm of local refinement based optimization with application to transistor and interconnect sizing. Technical Report 970034, UCLA CS Dept, September 1997.

[18] J. Cong and L. He. Simultaneous transistor and interconnect sizing based on the general dominance property. In *Proc. ACM SIGDA Workshop on Physical Design*, pages 34–39, April 1996.

[19] J. Cong and L. He. An efficient technique for device and interconnect optimization in deep submicron designs. In *Proc. Int. Symp. on Physical Design*, pages 45–51, April 1998.

[20] J. Cong and L. He. An efficient approach to simultaneous transistor and interconnect sizing. In *Proc. Int. Conf. on Computer-Aided Design*, pages 181–186, Nov. 1996.

[21] J. Cong, L. He, A. B. Kahng, D. Noice, N. Shirali, and S. H.-C. Yen. Analysis and justification of a simple, practical 2 1/2-d capacitance extraction methodology. In *Proc. Design Automation Conf*, pages 627–632, 1997.

[22] J. Cong, L. He, K.Y. Khoo, C.K. Koh, and Z. Pan. Interconnect design for deep submicron ICs. In *Proc. Int. Conf. on Computer Aided Design*, pages 478–485, 1997.

[23] J. Cong, L. He, C.K. Koh, and Z. Pan. Global interconnect sizing and spacing with consideration of coupling capacitance. In *Proc. Int. Conf. on Computer Aided Design*, pages 628–633, 1997.

[24] J. Cong, L. He, C.K. Koh, and Z. Pan. *User Manual for TRIO – UCLA Interconnect Optimization Package*. UCLA CS Dept, 1998 (available at http://cadlab.cs.ucla.edu/ ~helei/publications.html).

[25] J. Cong and Lei He. Theory and algorithm of local-refinement based optimization with application to device and interconnect sizing. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 18(4):406–420, April 1999.

[26] J. Cong and C.-K. Koh. Simultaneous driver and wire sizing for performance and power optimization. In *Proc. Int. Conf. on Computer Aided Design*, pages 206–212, November 1994.

[27] J. Cong, C.-K. Koh, and K.-S. Leung. Simultaneous buffer and wire sizing for performance and power optimization. In *Proc. Int. Symp. on Low Power Electronics and Design*, pages 271–276, August 1996.

[28] J. Cong and K. S. Leung. Optimal wiresizing under the distributed Elmore delay model. In *Proc. Int. Conf. on Computer Aided Design*, pages 634–639, 1993.

[29] J. Cong and K. S. Leung. Optimal wiresizing under the distributed Elmore delay model. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 14(3):321–336, March 1995.

[30] J. Cong, K. S. Leung, and D. Zhou. Performance-driven interconnect design based on distributed RC delay model. In *Proc. Design Automation Conf*, pages 606–611, 1993.

[31] J. Cong and P. H. Madden. Performance driven routing with multiple sources. In *Proc. IEEE Int. Symp. on Circuits and Systems*, pages 1.203–1.206, 1995.

[32] J. Cong and Z. Pan. Interconnect estimation and planning for deep submicron designs. In *Proc. Design Automation Conf*, pages 507–510, 1999.

[33] Jason Cong, Lei He, Cheng-Kok Koh, and Patrick H. Madden. Performance optimization of VLSI interconnect layout. *Integration, the VLSI Journal*, 21:1–94, 1996.

[34] Avant! Corporation. *Raphael User Manual*.

[35] N. Delorme, M. Belleville, and J. Chilo. Inductance and capacitance analytic formulas for vlsi interconnects. *IEEE Electron Device Lett*, EDL-32:996–997, 1996.

[36] J. G. Ecker. Geometric programming: Methods, computations and applications. *SIAM Review*, 22(3):338–362, July 1980.

[37] W. C. Elmore. The transient response of damped linear networks with particular regard to wide-band amplifiers. *Journal of Applied Physics*, 19(1):55–63, January 1948.

[38] J. P. Fishburn and A. E. Dunlop. TILOS: A posynomial programming approach to transistor sizing. In *Proc. Int. Conf. on Computer Aided Design*, pages 326–328, 1985.

[39] Y. Gao and D. F. Wong. Optimal shape function for a bi-directional wire under elmore delay model. In *Proc. Int. Conf. on Computer Aided Design*, pages 622–627, 1997.

[40] B. A. Gieseke and et al. A 600mhz superscalar risc microprocessor with out-of-order execution. In *Proc. IEEE Int. Solid-State Circuits Conf.*, pages 176–177, 1997.

[41] F. W. Grover. *Inductance Calculations: working formulas and tables*. Dover Publications.

[42] R. Guerrieri and A. Sangiovanni-Vincentelli. Three-dimensional capacitance evaluation on a connection machine. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 7:1125–1133, 1988.

[43] H., Chan. *Private Communication*, 1994.

[44] L. He, N. Chang, S. Lin, and O. S. Nakagawa. An efficient inductance modeling for on-chip interconnects. In *Proc. IEEE Custom Integrated Circuits Conference*, pages 457–460, May 1999.

[45] Y. I. Ismail, E. G. Friedman, and J. L. Neves. Figures of merit to characterize the importance of on-chip inductance. In *Proc. Design Automation Conf*, pages 560–565, 1998.

[46] A. B. Kahng and G. Robins. A new class of iterative Steiner tree heuristics with good performance. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 11(7):893–902, July 1992.

[47] M. Kamon, M.J. Tsuk, and J. White. Fasthenry: a multipole-accelerated 3d inductance extraction program. *IEEE Trans. on MIT*, 1994.

[48] J. Lillis and C.-K. Cheng. Timing optimization for multisource nets: characterization and optimal repeater insertion. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 18:322–31, March 1999.

[49] J. Lillis, C. K. Cheng, and T. T. Y. Lin. Optimal wire sizing and buffer insertion for low power and a generalized delay model. In *Proc. Int. Conf. on Computer Aided Design*, pages 138–143, November 1995.

[50] J. Lillis, C. K. Cheng, T. T. Y. Lin, and C. Y. Ho. New performance driven routing techniques with explicit area/delay tradeoff and simultaneous wire sizing. In *Proc. Design Automation Conf*, pages 395–400, June 1996.

[51] J. Lillis, C.K. Cheng, S. Lin, and N. Chang. *High-performance interconnect analysis and synthesis*. John Wiley, 1999.

[52] J. Lillis, C.K. Cheng, S. Lin, and N. Chang. *High-performance interconnect analysis and synthesis*. John Wiley, to appear in 1999.

[53] David Luenberger. *Linear and Non-linear Programming*. Addison-Wesley, 1989.

[54] MCNC. *MCNC Designers' Manual*, 1993.

[55] N. Menezes, R. Baldick, and L. T. Pileggi. A sequential quadratic programming approach to concurrent gate and wire sizing. In *Proc. Int. Conf. on Computer Aided Design*, pages 144–151, 1995.

[56] N. Menezes, S. Pullela, F. Dartu, and L. T. Pillage. RC interconnect synthesis — a moment fitting approach. In *Proc. Int. Conf. on Computer Aided Design*, pages 418–425, 1994.

[57] G. E. Moore. Cramming more components onto integrated circuits. *Electronics Magazine*, 38:114–117, 1965.

[58] K. Nabors and J. White. Fastcap: A multipole accelerated 3-d capacitance extraction program. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pages 1447–1459, November 1991.

[59] Z. Ning and P. M. Dewilde. Spider - capacitance modeling for vlsi interconnections. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 7(2):1221–1228, Dec. 1988.

[60] T. Okamoto and J. Cong. Buffered Steiner tree construction with wire sizing for interconnect layout optimization. In *Proc. Int. Conf. on Computer Aided Design*, pages 44–49, November 1996.

[61] J. K. Ousterhout. Switch-level delay models for digital MOS VLSI. In *Proc. Design Automation Conf*, pages 542–548, 1984.

[62] W. Press, S. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992.

[63] J. Qian, S. Pullela, and L. T. Pileggi. Modeling the "effective capacitance" for the RC interconnect of CMOS gates. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 13(12):1526–1535, December 1994.

[64] E. Rosa. The self and mutual inductance of linear conductors. *Bulletin of the National Bureau of Standards*, pages 301–344, 1908.

[65] J. Rubinstein, P. Penfield, Jr., and M. A. Horowitz. Signal delay in RC tree networks. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, CAD-2(3):202–211, July 1983.

[66] A. E. Ruehli and P. A. Brennan. Efficient capacitance calculations for three-dimensional multiconductor systems. *IEEE Trans. on Microwave Theory Tech.*, MTI-21:76–82, 1973.

[67] A.E. Ruehli. Inductance calculation in a complex integrated circuit environment. *IBM Journal of Res. and Dev.*, 1972.

[68] A.E. Ruehli. Equivalent circuit models for three-dimensional multiconductor systems. *IEEE Trans. on MIT*, 1974.

[69] T. Sakurai. Closed-form expressions for interconnect delay, coupling, crosstalk in VLSI's. *IEEE Trans. on Electron Devices*, ED-40:118–124, 1993.

[70] T. Sakurai and K. Tamaru. Simple formulas for two- and three- dimensional capacitances. *IEEE Trans. on Electron Devices*, ED-30:183–185, 1983.

[71] G. S. Samudra and H. L. Lee. A set of analytical formulas for capacitance of vlsi interconnects of trapezium shape. *IEEE Trans. on Electron Devices*, ED-41:1467–1469, 1994.

[72] P. K. Sancheti and S. S. Sapatnekar. Interconnect design using convex optimization. In *Proc. IEEE Custom Integrated Circuits Conf.*, pages 549–552, 1994.

[73] S. S. Sapatnekar. RC interconnect optimization under the Elmore delay model. In *Proc. Design Automation Conf*, pages 387–391, 1994.

[74] S. S. Sapatnekar. Wire sizing as a convex optimization problem: exploring the area-delay tradeoff. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 15(8):1001–1011, August 1996.

[75] S. S. Sapatnekar, V. B. Rao, P. M. Vaidya, and S. M. Kang. An exact solution to the transistor sizing problem for CMOS circuits using convex optimization. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 12(11):1621–1634, November 1993.

[76] A. Seidl, A. Svoboda, J. Oberndorfer, and W. Rosner. Capcal - a 3d capacitance solver for support of cad systems. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 7(11):549–556, 1988.

[77] Semiconductor Industry Association. *National Technology Roadmap for Semiconductors*, 1994.

[78] Semiconductor Industry Association. *National Technology Roadmap for Semiconductors*, 1997.

[79] R. H. Uebbing and M. Fukuma. Process-based three-dimensional capacitance simulation–triceps. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 5:215–210, 1986.

[80] L. Vandenberghe, S. Boyd, and A. E. Gamal. Optimal wire and transistor sizing for circuits with non-tree topology. In *Proc. Int. Conf. on Computer Aided Design*, pages 252–259, 1997.

[81] F. C. Wu, S.C. Wong, P. S. Liu, D. L. Yu, and F. Lin. Empirical models for wiring capacitances in vlsi. In *Proc. IEEE Int. Symp. on Circuits and Systems*, 1996.

[82] T. Xue and E. S. Kuh. Post routing performance optimization via multilink insertion and non-uniform wiresizing. In *Proc. Int. Conf. on Computer Aided Design*, pages 575–580, 1995.

[83] T. Xue, E. S. Kuh, and Q. Yu. A sensitivity-based wiresizing approach to interconnect optimization of lossy transmission line topologies. In *Proc. IEEE Multi-Chip Module Conf.*, pages 117–121, 1996.

[84] S. Yen and N. Shirali. Capacitance extraction. Technical Report Application Note, Cadence Design Systems, 1995.

[85] D. Zhou, F. P. Preparata, and S. M. Kang. Interconnection delay in very high-speed vlsi. *IEEE Trans. on CAS*, July 1991.