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 and Parents.

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. I believe that what I learned from him would guide me well beyond my graduation.

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

I thank Prof. Andrew B. Kahng for his help in my research. Main results presented in Chapter 5 are 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 appreciate the opportunity to work with Dr. Norman Chang, Dr. Shen Lin and Dr. O. Sam Nakagawa, during my research internship with Hewlett-Packard Laboratories. Discussing with them stimulated results presented in Chapter ??. I especially appreciate Dr. Chang's help and advice for my career planning.

My deep gratitude and appreciation also go to Prof. Lawrence Pileggi, Prof. Sachin Sapatnekar, and Prof. Martin D. F. Wong. They gave me invaluable help and advice on my career planning.

I dedicate this dissertation to my wife Shiping Xu and my parents. Their inspiration, and support have led me to go through the long way to finish this dissertation.

This work was supported in part by Defense Advanced Research Project Agency (DARPA), Electric Technology Office (ETO) under Contract DAAL01-96-K-3600, by DARPA under Contract 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. Prepared and conducted weekly discussion sections for thirty students; supervised computer sessions; commented on and evaluated term projects.
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. Prepared and conducted weekly tutorial sections for eighty students; assigned homework; held office hours and review sessions.

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 D 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 in 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 (*VLSI)* circuits move into the era of deep-submicron (*DSM*) technology and gigahertz frequency, the system performance has increasingly become dominated by the interconnect delay. This dissertation presents five related research topics on interconnect layout optimization, and interconnect extraction and modeling: the *multi-source wire sizing (MSWS)* problem, the *simultaneous transistor and interconnect sizing (STIS)* problem, the *global interconnect sizing and spacing (GISS)* problem, the *interconnect capacitance extraction* problem, and the *interconnect inductance extraction* problems.

Given a routing tree with multiple sources, the MSWS problem determines the optimal widths of the wire segments such that the delay is minimized. We reveal several interesting properties for the optimal MSWS solution, of which the most important is the bundled refinement property. Based on this property, we propose a polynomial time algorithm, which uses iterative bundled refinement operations to compute lower and upper bounds of an optimal solution. Since the algorithm often achieves identical lower and upper bounds in experiments, the optimal solution is obtained simply by the bound computation. Furthermore, this 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.

The STIS problem assigns optimal wire widths to interconnects and optimal sizes to transistors for minimizing the delay for multiple critical paths. To solve this problem with extremely high computational complexity, 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 leads immediately to an efficient polynomial-time algorithm, which in practice achieves the optimal solution to the STIS problem for large designs under accurate device model. More important, the formulation of CH-programs unifies solutions to several important problems for interconnect optimization.

The interconnect capacitance extraction problem extracts the capacitance values from complex 3-dimensional interconnect structures. It is needed for both interconnect optimization and interconnect verification, and is a difficult challenge for DSM VLSI circuits. To solve this problem, we first 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". We then present 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.

Our study of interconnect capacitance extraction shows that, rather than conventional ground capacitance, the coupling capacitance between neighboring wires becomes the dominant capacitance component for the DSM designs. Because the coupling capacitance is sensitive to spacing, it is important to study the

GISS problem. Given the topology for multiple nets, the GISS problem find the wire sizing and spacing solution simultaneously for all nets, with consideration of coupling capacitance between them. Our problem formulation is based on the concept of asymmetric wire sizing, and can be posed as a CH-program, which directly leads to an effective and efficient solution.

Due to increasingly wider and longer wire traces, faster clock frequencies and shorter rising times, inductance effects of on-chip interconnects no longer can be ignored. The interconnect inductance extraction problem computes the inductance values from three-dimensional interconnect structures. We propose an efficient yet accurate table-based approach based on 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 high-end microprocessor designs in Hewlett-Packard Company.

CHAPTER 1

Introduction

The driving force behind the impressive advancement of the VLSI circuit technology has been the constant reduction of the feature size, i.e., the minimum dimension of the transistor. It decreased from 2 μm in 1985 to 0.35 μm in 1996. For integrated circuits and systems using deep submicron (DSM, $\leq$ 0.35 μm) technologies, the interconnect modeling and optimization is one of the most important problems during layout design.

1.1 Trends in Deep Submicron Designs

The driving force behind the impressive advancement of the VLSI circuit technology has been the rapid scaling of the feature size, i.e., the minimum dimension of the transistor. It decreased from 2 μm in 1985 to 0.35 μm in 1996. According to the National Technology Roadmap for Semiconductors (NTRS) [97], it will further decrease at the rate of 0.7$\times$ per generation (consistent with Moore's Law) to reach 0.07 μm by 2010 (see Table 1.1). Such rapid scaling has two profound impacts. First, it enables much higher degree of on-chip integration. The number of transistors per chip will increase by more than 2$\times$ per generation to reach 800 millions in the 0.07 μm technology. Also, the density of interconnects will increase dramatically due to increased wiring layers and reduced wire width and spacing. The 0.07 μm technology will reach 7-8 wiring layers, 0.08 μm minimum wire width and 0.12 μm minimum wire spacing. Second, it implies that the circuit performance will be increasingly determined by the interconnect performance. As given in [35], when we advance from the 0.35 μm technology to the 0.07 μm technology, the intrinsic gate delay decreases from over 100 ps to around 10 ps, the delay of a local interconnect (1 mm) decreases from over 150 ps to around 50 ps, while the delay of a global interconnect (2 cm) increases from around 1 ns to over 6 ns. Therefore, interconnect will play the most critical role for performance optimization in deep submicron designs where the feature size is less than 0.35 μm. We propose to study the performance optimization problem for VLSI layout. We will focus more on the interconnect because of its dominating role in determining the circuit performance.

Tech. (μm)	0.35	0.25	0.18	0.13	0.10	0.07
Year	1995	1998	2001	2004	2007	2010
Area (mm^2)	250	300	360	430	520	620
transistor #	12M	28M	64M	150M	350M	800M
Wiring levels	4-5	5	5-6	6	6-7	7-8
Min. wire width	0.40	0.30	0.22	0.15	0.11	0.08
Min. wire spacing	0.60	0.45	0.33	0.25	0.16	0.12

Table 1.1: Summary of NTRS

1.2 Overview of this proposal

The remainder of the proposal includes the following parts:

In Chapter 2, we review previous work on device sizing and wire sizing. This chapter was presented as part of an invited survey paper [44].

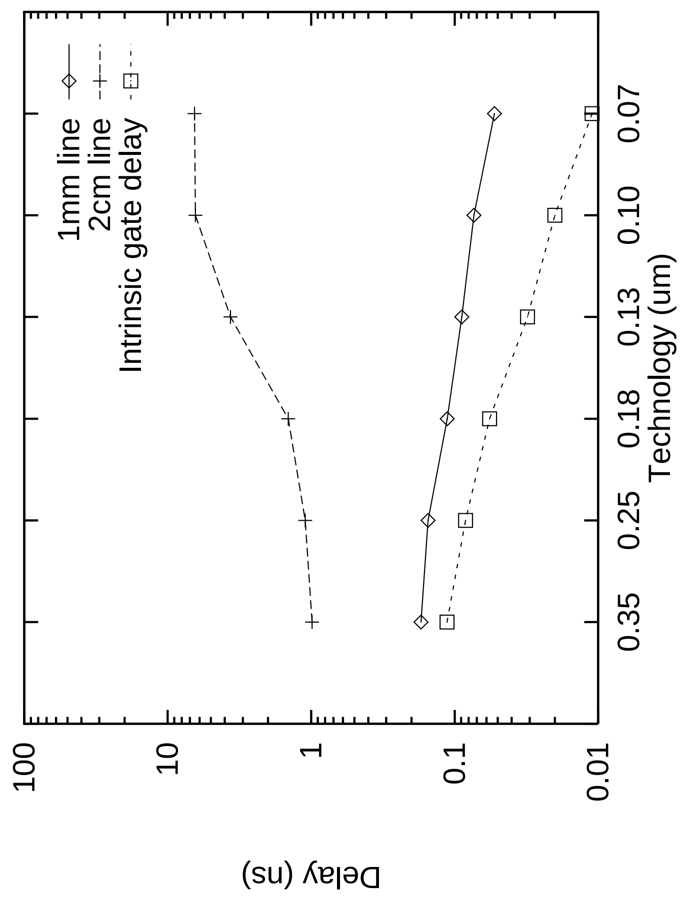


Figure 1.1: Global and local interconnect delays versus gate delays.

In Chapter 3, we study the optimal wiresizing problem for nets with multiple sources under the RC tree model and the Elmore delay model. Our contributions include:

- We decompose an MSIT into a source subtree (*SST*) and a set of loading subtrees (*LST*s), and 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 bundle refinement property which 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 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.

Results of this chapter was presented in [27, 28, 30].

In Chapter 4, we study the simultaneous transistor and interconnect sizing (*STIS*) problem. Our contributions include:

- We formulated the simultaneous transistor and interconnect sizing (STIS) problem. It supports the transistor sizing formulation to find an optimal size for each transistor, and gate sizing formulations to find either one optimal size or two optimal sizes for a gate. It also considers different transistor models, such as the simple step model used in the most previous work, and the much more accurate table-based model.

- We also formulated a class of optimization problems named CH-posynomial problems, and revealed that the dominance property holds for all CH-posynomial problems. Since STIS problems under different transistor models are all CH-posynomial problems, the local refinement operation previously used only for wire sizing problems can be used to compute a set of lower and upper bounds of optimal solutions to STIS problems in polynomial-time.

- Our experiments show that the bound computation algorithm often achieves identical lower and upper bounds and leads immediately to the optimal solution. Therefore, it is a very efficient and near-optimal algorithm in practice. We applied the algorithm to find the optimal size for each transistor and each $0.6\mu m$-long wire for a critical net containing 367 drivers/buffers and $59304\mu m$-long wire, and a 32bit-adder containing 1,026 transistors. Each took less than one minute in a SPARC-5 workstation.

- We observed the following results: (i) compared with separate driver sizing and wire sizing formulation for multi-source nets, the STIS formulation reduces the maximum delay by up to 17.7%, and more significantly, reduces the power dissipation by 61.7%; (ii) even for buffered trees where the simplest gates (inverters) are used, the simultaneous transistor and wire sizing formulation reduces the maximum delay by up to 8.2% under similar power dissipation when compared with the simultaneous gate and wire sizing formulation; (iii) again for buffered trees, the STIS formulation using the table-based model reduces the maximum delay by up to 15% when compared with the STIS formulation using the simple step model; (iv) the algorithm is also used to solve the transistor sizing problem for complex gates like full adders, it achieves a smooth area-delay trade-off.

Results of this chapter was presented in [29, 32, 34].

In Chapter 5, we conclude the proposal and propose to study interconnect optimization problem with consideration of coupling capacitance between neighboring wires for both delay minimization and signal integrity optimization. We will also study the signal integrity optimization problem in the context of mixed-signal designs.

CHAPTER 2

Previous Work on Device and Wire Sizing

Both device sizing and interconnect sizing can be used to reduce the delay. A larger driver/gate at the source of an interconnect tree has a stronger driving capability (or equivalently, smaller effective driver resistance), reducing the delay of this interconnect. But a larger driver/gate also means a heavier load (larger sink capacitance) to the previous stage and thus increases its delay. The *device sizing* problem is to determine the optimal size of each driver/gate to minimize the *overall* delay; this has been extensively studied in the past. Interconnect sizing, often called wire sizing, on the other hand, was investigated only recently. If the width of a wire is increased, the resistance of the wire will go down, which may reduce the interconnect delay, but the capacitance of the wire will go up, which may increase the interconnect delay. The *wire sizing* problem is to determine the optimal wire width for each wire segment to minimize the interconnect delay. When the interconnect resistance can be neglected as in the early days, the interconnect can be modeled as a lumped capacitor. In this case, the minimum wire width is preferred for delay minimization and only device sizing is necessary. But in the current deep submicron technology where the interconnect resistance can no longer be neglected, both device and wire sizing are needed to reduce the interconnect delay. Techniques for both device and wire sizing for delay minimization will be surveyed in this section. Sections 2.1 and 2.2 will present works on device sizing only and wire sizing only, respectively. Section 2.3 will focus on simultaneous device and wire sizing works, and Section 2.4 on simultaneous topology construction and sizing works. Because this proposal deals mainly with interconnect design and optimization, more emphasis will be given on wire sizing and simultaneous device and wire sizing.

2.1 Device Sizing

The device sizing problem is equivalent to determining the transistor channel width in CMOS logic since the transistor channel length is usually fixed to the minimum feature size. The following device sizing techniques are commonly used.

- Driver sizing: A chain of cascaded drivers is usually used at the source of an interconnect tree for heavy capacitive load. The *driver sizing* problem is to determine both the number of driver stages and the size for each driver.
- Transistor or gate sizing: The *transistor sizing* problem is to determine the optimal width, either continuous or discrete, for each transistor to optimize the overall circuit performance. Similarly, the gate sizing problem includes both the continuous and the discrete gate sizing problems. The *continuous gate sizing* problem assumes that all transistors in a gate can be scaled by a common factor, which is called the *size* of a gate. The *discrete gate sizing* problem assumes that each gate has a discrete set of pre-designed implementations (cells) as in a given cell library, and one needs to choose an appropriate cell for each gate for performance optimization.
- Buffer insertion: A buffer can be a pair of inverters or a single inverter[1], and they may have different sizes. The *buffer insertion* problem is to determine both the placement and the size of each buffer in a routing tree. In a

[1] For single-inverter buffers, the signal polarity needs to be considered during buffer insertion

uniform view, the driver sizing problem is a special case of buffer insertion with buffers only at the source of the routing tree.

2.1.1 Driver Sizing

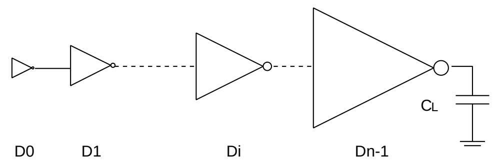


Figure 2.1: The cascaded drivers for a heavy capacitance loading.

For an interconnect tree with heavy load (due to large interconnect capacitance or/and sink capacitance), a chain of cascaded drivers is usually used at the source. The 0-th stage is a small, often minimum size, driver, and the driver size increases until the last stage is large enough to drive the heavy loading capacitance (see Figure 2.1). An early result on the optimal driver sizing problem was reported by Lin and Linholm [68]. Let D_i be the driver of the i-th stage, and C_i and R_i be its input gate capacitance and effective driver resistance, respectively. The stage ratio is defined to be $f_i = \frac{C_i}{C_{i-1}}(i > 0)$, it was shown that

Lin-Linholm Theorem: If the loading capacitance is C_L and the stage number is N, the optimal stage ratio at each stage is a constant $(\frac{C_L}{C_0})^{1/N}$ in order to achieve the minimum delay.

Let $\tau_0 = R_0 \cdot C_0$, where C_0 and R_0 are the input gate capacitance and the effective driver resistance for D_0, respectively. Under the constant stage ratio f and the switch-level driver model, we have $R_i = \frac{R_0}{f^i}$ and $C_i = C_0 \cdot f^i$. Therefore, every stage has the same delay $f\tau_0$, and the total delay of N stages is $t_d = Nf\tau_0$. When N is not fixed, the optimal stage number is $N = ln(\frac{C_L}{C_g})/lnf$. The total delay becomes $Nf\tau_0 = ln(C_L/C_g) \cdot \tau_0 \cdot f/lnf$. It is minimized when $\frac{f}{ln(f)}$ is minimum, which leads to $f = e$, the base of natural logarithms. This is the well known optimal stage ratio for delay minimization presented in most textbooks (such as [73]).

The output capacitance of a driver is not considered in the above derivation. Hedenstierna and Jeppson [54] developed a more accurate analytical delay formula with consideration of the input waveform slope and the output capacitance of the driver. Based on their delay formula, the optimal stage ratio f satisfies

$$f = e^{(\alpha+f)/f}$$

where α is the ratio between the intrinsic output capacitance and the input gate capacitance of the inverter. Since typical α is about 1.35 for the technology they used, the optimal stage ratio is in the range of 3–5 instead of e. It is easy to find that the optimal stage ratio is still e if $\alpha = 0$. The stage number N can be determined by the optimal stage ratio f as $N = ln\frac{C_L}{C_0}/lnf$. Then, f is used for all stages, except that the last stage has a little bit *larger* ratio for delay minimization [54].

Most recently, Zhou and Liu [112] discussed the optimal driver sizing for high-speed low-power ICs. The *increasing* stage ratios $f_i = f_0(1+\gamma)^i$ are used, where γ is a modification factor determined by the I-V curve of the transistor. The typical value of γ is around 0.2. The reason for the increasing stage ratio is the following: if the step waveform is applied at the input of the very first stage, the waveforms become increasingly "softer" at the subsequent stages, i.e., the input waveform to the following stage is no longer a step so an increasingly larger delay is expected for each following stage. Thus, an increasing stage ratio is applied to maintain equal delay in different stages. The authors derived an analytic relationship between signal delay, power dissipation, driver size and interconnect loading. They show that

$$f_0 = e^{\frac{\gamma}{2}+\sqrt{2\gamma\frac{C_L}{C_0}-1}} \quad and \quad f_i = f_0(1+\gamma)^i$$

are the optimal stage ratios for delay minimization. We would like to point out that all studies in [68, 54, 112] also discussed the optimal driver sizing for power minimization. Another study on optimal driver sizing for low-power can be found in [104].

2.1.2 Transistor and Gate Sizing

In addition to sizing drivers which usually drive global interconnects, the sizes of all transistors and gates in the entire circuit or a sub-circuit can also be adjusted properly according to their capacitive loads for performance or power optimization. The transistor sizing problem has been approached using both sensitivity based methods and mathematical optimization based methods. The gate sizing problem has been classified into both continuous and discrete gate sizing problems, and solved by different approaches.

2.1.2.1 Sensitivity Based Transistor Sizing

Fishburn and Dunlop [50] studied the transistor sizing problems for synchronous MOS circuits. Let $x_1, \cdots, x_i, \cdots, x_n$ be the transistor sizes, A the total active area of transistors and T the clock period. If K is a positive constant, there are three forms for the transistor sizing problem as follows:

1. Minimize A subject to the constraint $T < K$.
2. Minimize T subject to the constraint $A < K$.
3. Minimize AT^K.

Let a transistor be modeled by the switch level model, then the gate, source and drain capacitance are all proportional to the transistor size, and the effective resistance is inversely proportional to it. A CMOS gate will be modeled by a distributed RC network. The Elmore delay (Eqn. (3.1)) is used to compute the worst-case delay of the gate, which is the delay through the highest resistive path in the RC network. The delay of a PI-PO path is the sum of delays through all gates in the path. It is not difficult to verify that the delay of a PI-PO path can be written into this form

$$\sum_{1 \leq i,j \leq N} a_{ij} \frac{x_i}{x_j} + \sum_{1 \leq i \leq N} \frac{b_i}{x_i} \tag{2.1}$$

where the a_{ij} and b_i are nonnegative constants. In fact, a_{ij} is non-zero only when transistors i and j are dc-connected.

Furthermore, the authors of [50] show that Eqn. (2.1) and the area $A = \sum x_i$ are posynomials and the transistor sizing problems of the three forms are all posynomial programs.[2] Even though posynomial programming methods can be used to optimally solve the three forms of the transistor sizing problem, it is computationally expensive to be used for an entire circuit. Thus the transistor sizing tool TILOS (TImed LOgic Synthesizer) was developed to minimize A subject to $T < K$ based on the following scheme: First, the minimal size is assigned to all

[2] According to [48], a posynomial is a function of positive vector $\mathbf{X} \in \mathbf{R}^m$ having the form $g(\mathbf{X}) = \sum_{i=1}^{N} u_i(\mathbf{X})$ with

$$u_i(\mathbf{X}) = c_i x_1^{a_{i1}} x_2^{a_{i2}} \cdots x_m^{a_{im}}, \qquad i = 1, 2, \cdots, N$$

where the exponents a_{ij} are real numbers and the coefficients c_i are positive. A posynomial program is the following minimization problem:

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

where each g_k $(k = 0, 1, 2, \cdots, p)$ is a posynomial. The posynomial program has the important property that the local optimum is also the global optimum. In fact, the concepts of posynomial and posynomial program play an important role in many wire and device sizing works to be presented.

transistors. Then, timing analysis is performed to find the critical delay T. If T is larger than K, the sensitivities of all transistors related to the critical path will be computed. The sensitivity is defined as the delay reduction due to per transistor size increment. The size of the transistor with the largest sensitivity will be multiplied by a user defined factor (BUMPSIZE) and then the algorithm goes to the timing analysis again. This procedure will be terminated when the timing specification is satisfied or there is no improvement in the current loop, i.e., all sensitivities are zero or negative. The performance of TILOS is quite good. Circuits with up to 40,000 transistors have been tested. Based on the experiments, the results are reasonably close to the optimum under their delay model. However, it assumes that the effective resistance for a transistor is independent of the waveform slope of the input. But, in fact, the input slope has a significant effect on the transistor effective resistance. Another sensitivity based transistor sizing work is [93] which also performs iterative transistor sizing to reduce the critical path delay. In contrast to TILOS, it changes the size of more than one transistor in each iteration. In addition, a sensitivity-based transistor sizing is presented by Borah et al. [9] to minimize power consumption of CMOS circuit under delay constraint.

2.1.2.2 Mathematical Programming Based Transistor Sizing

Note that the method in [50] does not guarantee the optimality of the result. Studies have been done to formulate the transistor sizing problem as mathematical programming problems to obtain an optimal solution. Methods in [26, 55, 71] formulate the transistor sizing problem as nonlinear programs and solve them by the method of Lagrangian multipliers. Methods in [45, 56, 15] apply the following two-step iterations. First, the delay budget is distributed to each gate; Then, the transistors in each gate are sized optimally to satisfy the time budget.

Later, a two-phase algorithm was presented by Shyu *et al.* [98] to minimize the circuit area under timing constraints: first, TILOS [50] is used to generate an initial solution; then, a mathematic optimization is formulated and solved by using feasible directions to find the optimal solution. The variables in the optimization problem, however, are not sizes of *all* transistors in the circuit, but only sizes of those transistors that have been tuned by TILOS, thus it is still possible to lose the optimal solution with respect to the whole circuit. Experimental results of circuits with up to 500 transistors have been presented.

More recently, Sapatnekar [94] developed a transistor sizing tool iCONTRAST, again, to minimize the circuit area under timing constraints. It employs the analytical delay model developed in [54] which can consider the waveform slope of input signals to transistors, but assumes that the transition time is twice the Elmore delay of the previous stage. Under the delay model, the transistor sizing problem is a posynomial program that can be transformed into a convex program and the convex programming method [101] was implemented to solve the transformed problem. When using the simple delay model of TILOS [50], and the timing specification is loose, the area of the solution obtained by TILOS is close to that of the solution obtained by the iCONTRAST algorithm. However, as the time specification is tightened, the TILOS-solutions have larger area when compared with the iCONTRAST-solutions. Experimental results of circuits with up to 800 transistors have been presented.

2.1.2.3 Continuous Gate Sizing

The *continuous gate sizing* problem assumes that all transistors in a gate can be scaled by a common factor, which is called the *size* of a gate. In essence, it is very similar to the transistor sizing problem, but has much lower complexity for a given design, since all transistors in a gate are scaled by the same factor. Hoppe *et al* [59] developed analytical models for signal delay, chip area and dynamic power dissipation and formulated a nonlinear problem to minimize the weighted linear combination of delay, area and power. The nonlinear problem is solved by the Newton-Raphson algorithm. A 64K-SRAM was optimized on a mainframe computer in 2 hours.

In order to speed up the gate sizing problem, the linear programming (LP) formulation has been proposed. Berkelaar and Jess [6] used a piecewise-linear (PWL) function to linearize the delay function. More precisely, one divides the gate sizes into subranges so that the delay of a gate is a linear function of gate sizes within each subrange. Thus, the gate sizing problem can be formulated as a LP problem. Their LP formulation [6] is to minimize the power subject to a delay constraint. Experimental results on circuits with up to 500 gates were presented. Later on, their LP-based method was expanded [5] to compute the entire area or power-consumption versus delay trade-off curve. Results on MCNC'91 two-level benchmarks with up to 4,700 gates were reported. Recently, Tamiya, Matsunaga and Fujita [99] proposed another LP-based method where the latest and the earliest arrival times are introduced so that the setup and hold time constraints can be handled. The objective is to minimize the weighted linear combination of clock period, area and power. Result on a chip of 13,000 transistors was reported. Note that gate sizing works in [6, 6, 99] assume that the gate delay is a convex function of gate sizes, which is needed to make sure that the error introduced by the PWL approximation is small. However, the gate delay in fact is not a strict convex function.

More recently, Chen, Onodera and Tamaru [14] removed the convex delay model assumption in previous LP-based works. They also divided the the gate sizes into subranges, but different from the previous works [6, 5, 99] where only one LP problem is formulated over the whole gate size range with the delay being a PWL function in this LP formulation, a LP problem is formulated for *every* subrange with the delay being a linear function for each LP formulation. When the subrange is small enough, the error introduced by the non-convexity will be small. The linear programming is performed iteratively, and subranges of gate sizes are updated according to the result from the previous step. Experimental results for ISCAS85 benchmarks with up to 3,500 gates were reported.

2.1.2.4 Discrete Gate Sizing

The resulting optimized design by the continuous gate sizing formulation may be impractical or expensive to implement since a large number of manually-designed cells or a smart cell generator are needed. Thus, the discrete gate sizing problem is studied by assuming that each gate has a discrete set of pre-designed implementations (cells) as in a given cell library and one needs to choose an appropriate cell for each gate for performance optimization. In general, the discrete gate sizing problem is *NP*-complete: Chan [10] showed that the double sized discrete gate sizing problem to find discrete gate sizes to satisfy both maximum and minimum delay constraints is *NP*-complete, even without consideration of area minimization. Hinsberger and Kolla [57] proved the single-sided (with only maximum delay constraint) discrete gate sizing problem in a DAG (directed acyclic graph) is *NP*-complete under three objectives: to minimize the maximum delay, to minimize the maximum delay under an area constraint, and to minimize the area under a maximum delay constraint. Li [62] further showed that the discrete gate sizing problem under both area and maximum delay constraints is strongly

NP-hard even for a chain of gates.

The methods which are optimal for logic networks of certain structures have been proposed. For the double-sided problem, a branch and bound algorithm [10] was developed to find the optimal solution for tree structures. For the single-sided problem, an optimal dynamic programming method to minimize the maximum delay was proposed, again for tree structures [57]. It assumes that the delay for a gate could be determined *locally*, i.e., the delay could be determined only by the sizes of the gate and its fanout gates, and works in a bottom-up manner. Furthermore, an exact algorithm to minimize area subject to a maximum delay constraint (single-sided) was presented for series-parallel circuits [63]. A simple series circuit is solved by a dynamic programming method and a simple parallel circuit is solved by a number of transformations. All series-parallel circuits can be solved recursively.

Heuristics have been proposed to expand the optimal algorithms for trees or series-parallel circuits to the general cases in [10, 63]. Furthermore, the following methods have been developed: Lin, Marek-Sadowska and Kuh [69] use the weighted sum of sensitivity and criticality to choose cell sizes for standard-cell designs. The sensitivity of a cell is defined as $-\frac{\Delta delay}{\Delta area}$, where both *delay* and *area* are in terms of the cell. The criticality is inversely proportional to the *slack* of a cell so that a cell in a non-critical path will not be over-sized[3]. Chuang, Sapatnekar and Hajj [24, 25] presented a three-step method to minimize the area subject to the single-sided delay constraint. First, they formulate a linear programming (LP) problem to obtain a continuous solution. Then, they map the continuous solution onto the allowed discrete gate sizes; Finally, they adjust the gate sizes to satisfy the delay constraint. Also, the three-step algorithm was

[3]Since the method in [50] only sizes those transistors in the critical path based on their sensitivities, criticality has been considered implicitly.

modified in [23] to minimize the area under the double-sided delay constraint. It is worth mentioning that the work in [24, 25] further formulated gate sizing and clock skew optimization as a single LP problem not only to reduce the circuit area but also to achieve faster clocks. Another method to combine both gate sizing and clock skew optimization can be found in [95]. In addition, Chuang and Sapatnekar proposed another LP formulation to address the continuous gate sizing problem for power optimization in [22].

2.1.3 Buffer Insertion

Buffer (also called repeater) insertion is a common and effective technique to reduce interconnect delay. As the Elmore delay of a long wire grows quadratically in terms of the length of the wire, buffer insertion can reduce interconnect delay significantly. Bakoglu ([2]) gives a closed-form formula to determine the number and sizes of buffers (inverters) that are uniformly placed in a long interconnect line for delay minimization. Let k be the number of inverters and h the uniform size for every inverter, then the optimal values for an interconnect line of uniform wire width are the following:

$$\begin{aligned} k &= \sqrt{\frac{0.4 R_{int} C_{int}}{0.7 R_0 C_0}} \\ h &= \sqrt{\frac{R_0 C_{int}}{R_{int} C_0}} \end{aligned}$$

where R_{int} and C_{int} are the total resistance and capacitance for the interconnect line, respectively, and R_0 and C_0 the driver resistance and the input capacitance of the minimum-size inverter, respectively.

A polynomial-time dynamic programming algorithm was presented by van Ginneken [102] to find the optimal buffer placement and sizing for RC trees under the Elmore delay model. The formulation assumes that the possible buffer positions (called legal positions), possible buffer sizes, and the required arrival times at sinks are given. The optimal buffer placement and sizing is chosen so that the required arrival time at the source is maximized. For simplicity, the buffer of two inverters with the fixed size is used and the polarity of the signal can be ignored. Legal positions were assumed to be right after the branching points in the tree (see Figure 2.2.a).

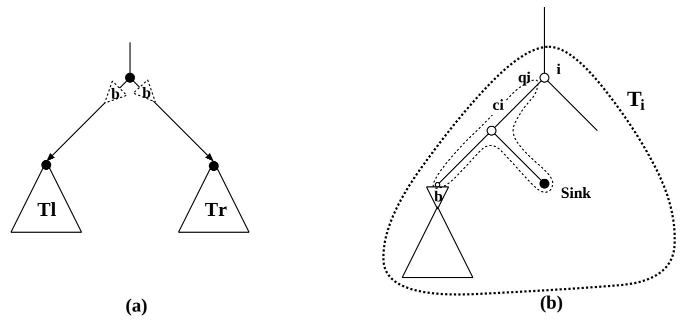


Figure 2.2: (a) Legal position for buffer insertion; (b) An option in a legal position.

The algorithm includes both bottom-up synthesis and top-down selection procedures. It begins with the bottom-up synthesis procedure, where for each legal position i for buffer insertion, a set of (q_i, c_i) pairs, called *options*, is computed for possible buffer assignments in the entire subtree T_i rooted at i. Each q_i is a required arrival time at i and c_i is the capacitance of of *dc-connected subtree*[4] rooted at i corresponding to q_i (Figure 2.2.b). Note that c_i is *not* the total capacitance in T_i.

A wire segment in the routing tree is modeled by a π-type circuit and only the wire area capacitance is considered. Recall that r and c_a are the resistance and the area capacitance for a unit-length wire, respectively. When a wire segment

[4] "dc-connected" means "directly connected by wires".

with upstream node k is added at i, an option (q_k, c_k) will be generated at k for *every* (q_i, c_i) at i as the following:

$$
\begin{aligned}
q_k &= q_i - r \cdot l \cdot (\frac{c \cdot l}{2} + c_i) \\
c_k &= c_i + c \cdot l
\end{aligned}
$$

where l is the length of the wire segment.

A buffer is modeled by the input gate capacitance C_{buf}, the driver resistance R_{buf} and the intrinsic delay D_{buf}. When a buffer with input node k is inserted at i, an option will be generated at k for *every* (q_i, c_i) at i as the following:

$$
\begin{aligned}
q_k &= q_i - D_{buf} - R_{buf} \cdot c_i \\
c_k &= C_{buf}
\end{aligned}
$$

When two subtrees T_i and T_j are merged at node k, for *every pair* of (q_i, c_i) and (q_j, c_j) (at i and j, respectively) an option (q_k, c_k) will be generated at k as the following:

$$
\begin{aligned}
q_k &= min(q_i, q_j) \\
c_k &= c_i + c_j
\end{aligned}
$$

The following *pruning rule* is used to prune a suboptimal option during the computation of options. For two options (q, c) and (q', c') in the same legal position, if $c' \geq c$ and $q' < q$ then (q', c') is suboptimal, thus, it can be pruned from the solution space. If the total number of legal positions is N, it was shown in [102] that the total number of options at the source of the whole routing tree is no larger than $N + 1$ even though the number of possible buffer assignments is 2^N.

After the bottom-up synthesis procedure, the optimal option is the one which has the maximum requirement time at the source pin of the *whole* interconnect tree. Then, the top-down selection procedure is carried out to trace back the buffer placement (in general, also the buffer sizes) which led to the optimal option. Several extensions can be made. It is easy to allow buffers of different types (sizes) to be placed. With different R_{buf}, C_{buf} and D_{buf} values for each type of buffer, there may be an extra option generated in every legal position for every extra buffer type. Let B be the number of buffer types and N, again, be the total number of legal positions, the total number of options at the root of the whole tree is bounded from above by $N + B$. In general, the time complexity of the algorithm is $O((N + B)^2 + k)$, where N is the total number of legal positions for buffer insertion, B the total number of buffer types and k the total number of sinks.

2.2 Wiresizing Optimization

It was first shown by Cong, Leung, and Zhou [42] that when wire resistance becomes significant, as in the deep submicron CMOS design, proper wire sizing can further reduce the interconnect delay. Their work presented an optimal wire sizing algorithm for a single-source RC interconnect tree to minimize the uniform upper bound of the delay [85]. Later on, single-source wire sizing algorithms were presented in [40, 41, 91, 107, 12, 11] using the Elmore delay model, in [75] using a higher-order RC delay model and in [109] using a lossy transmission line model. Furthermore, wire sizing was carried out simultaneously with device sizing in [38, 76, 74, 65] We classify the wire sizing works according to their objective functions and present them in Sections 2.2.1 and 2.2.2, and then discuss the simultaneous device and wire sizing in Section 2.3.

2.2.1 Wiresizing to Minimize Weighted Delay

In order to reduce the delays to multiple critical sinks in an interconnect tree with a single source, the wire sizing algorithms given by Cong and Leung [40, 41] minimize a weighted combination of Elmore delays from the single source to multiple critical sinks. Later on, Cong and He [28, 30] extended this formulation to the multiple-source net case, where the objective is to minimize the weighted combination of Elmore delays between multiple source-sink pairs. Wiresizing works in [40, 41] assumed that the wire widths are discrete and uniform within a wire segment or sub-segment. Most recently in [12], an optimal wire sizing formula was derived by Chen et al. to achieve the continuous and non-uniform wire width for each wire segment, again to minimize the weighted combination of Elmore delays from a single source to a set of critical sinks. All these works assume that the weights of the delay penalty between the source and each sink or each source-sink pair are given a prior.

2.2.1.1 Discrete Wiresizing for Single-Source RC Tree

In [42], Cong, Leung and Zhou modeled an interconnect tree as a distributed RC tree and applied the upper-bound delay model [85]. They showed that when the driver resistance is much larger than the wire resistance of the interconnect, the interconnect can be modeled as a lumped capacitor without losing much accuracy and that the conventional minimum wire width solution often leads to an optimal design. However, when the resistance ratio, i.e. the driver resistance versus unit wire resistance, is small, optimal wire sizing can lead to substantial delay reduction. In addition, they developed the first polynomial-time optimal wire sizing algorithm. Since the uniform upper bound delay model does not distinguish the delays at different sinks and may lead to over-sizing, Cong and

Leung [40, 41] extended the work to the Elmore delay formulation of Eqn. (3.1). Their formulation and method are summarized as follows.

Given a routing tree T, let $sink(T)$ denote the set of sinks in T, $\mathcal{W}$ be the wire sizing solution (i.e., wire width assignment for each segment of T) and $t_i(\mathcal{W})$ be the Elmore delay from the source to sink s_i under $\mathcal{W}$. The following weighted combination of delays is used as the objective function for wire sizing optimization.

$$t(\mathcal{W}) = \sum_{s_i \in sink(T)} \lambda_i \cdot t_i(\mathcal{W}) \tag{2.2}$$

where λ_i is the weight of the delay penalty to sink s_i. The larger λ_i, the more critical sink s_i is.

The following monotone property and separability were shown in [41].

Monotone Property: Given a routing tree, there exists an optimal wire sizing solution $\mathcal{W}$ such that $w_e \geq w_{e'}$ if segment $e \in Ans(e')$.

Separability: Given the wire width assignment of a path P originated from the source, the optimal wire width assignment for each subtree branching off from P can be carried out independently.

Based on these two properties, the optimal wire sizing algorithm (OWSA) was developed. It is a dynamic programming method based on the wire sizing solution for a *single-stem tree*, which is a tree with only one segment (called the *stem segment* of that tree) incident on its root (see Figure 2.3(a)). We use $sst(e)$ to denote the single-stem tree with stem e.

According to the separability, once e and every ancestor segment of e are assigned the appropriate widths, the optimal wire width assignment for the single-stem subtrees $sst(e_{c1})$, $sst(e_{c2})$, $\cdots$, $sst(e_{cb})$ of the tree $sst(e)$ (with respect to the width assignment of e and its ancestors) can be *independently* determined, where the segments $e_{c1}, \cdots, e_{cb}$ are the children of e. Therefore, given a set of possible widths $\{W_1, W_2, \cdots, W_r\}$, OWSA enumerates all the possible width assignments of e. For each possible width assignment W_k of e ($1 \leq k \leq r$), the optimal wire sizing is determined for each single-stem subtree $sst(e_{ci})$ ($1 \leq i \leq b$) of $sst(e)$ independently by recursively applying the same procedure to each $sst(e_{ci})$ with $\{W_1, W_2, \cdots, W_k\}$ as the set of possible widths (to guarantee the monotone property). The optimal assignment for e is the one which gives the smallest total delay.

If the original routing tree T is not a single-stem tree, we can decompose it into b single-stem trees, where b is the degree of the root of T, and apply the algorithm to each individual single-stem tree separately (see Figure 2.3(b)). The worst-case time complexity of OWSA is $O(n^r)$, which is much faster than brute-force enumeration $O(r^n)$, where n is the number of wire segments and r is the number of possible wire widths. However, OWSA algorithm can be slow when r is large.

In order to further speed-up the OWSA algorithm, the greedy wire sizing algorithm (GWSA) was developed based on the local refinement and the dominance property to compute the lower and upper bounds of the optimal wire widths.

Given two wire sizing solutions $\mathcal{W}$ and $\mathcal{W}'$, $\mathcal{W}$ is defined to *dominate* $\mathcal{W}'$ if $w_e \geq w'_e$ for every segment e. Given a wire sizing solution $\mathcal{W}$ for the routing tree, and any particular segment e in the tree, a *local refinement* on e is defined to be the operation to optimize the width of e while keeping the wire width assignment of $\mathcal{W}$ on other segments unchanged. The following dominance property was shown in [41]

Dominance Property: Suppose that $\mathcal{W}^*$ is an optimal wire sizing solution. If a wire sizing solution $\mathcal{W}$ dominates $\mathcal{W}^*$, then any *local refinement* of $\mathcal{W}$ still

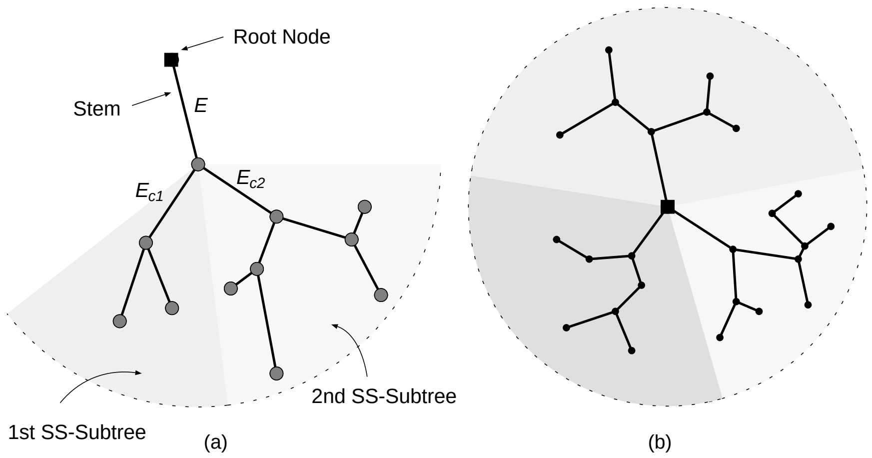


Figure 2.3: (a) A single-stem tree consists of a stem and a set of single-stem subtrees. In this example, e is the stem of the single-stem tree $sst(e)$, and $sst(e_{c1})$ and $sst(e_{c2})$ are the single-stem subtrees of $sst(e)$ (e_{c1} and e_{c2} are the children of e). (b) Any general tree T can be decomposed into a set of independent single-stem trees.

dominates $\mathcal{W}^*$. Similarly, if $\mathcal{W}$ is dominated by $\mathcal{W}^*$, then any *local refinement* of $\mathcal{W}$ is still dominated by $\mathcal{W}^*$.

The GWSA algorithm works as follows: starting with the minimum-width assignment, GWSA traverses the tree and performs a local refinement on each segment whenever possible. This process is repeated until no improvement is achieved on any segment in the last round of traversal. According to the dominance property, a lower bound of the optimal wire width on every segment is obtained. An upper bound of the optimal wire width assignment can be obtained similarly by starting with the maximum-width assignment. In most cases, GWSA obtains identical lower and upper bounds on all segments, which gives an optimal wire sizing solution. In cases when the lower and upper bounds do not meet on a few edges, the gaps are usually small and the OWSA algorithm can be applied very efficiently to obtain the optimal wire sizing solution. The worst-case time complexity of GWSA is $O(n^3 \cdot r)$. Experiments using SPICE simulation showed that, for the $0.5\mu m$ CMOS technology, the optimal wire sizing solution can reduce the maximum delay by up to 12.01% when compared with the minimum wire width solution.

2.2.1.2 Continuous and Non-uniform Wiresizing for Single-Source RC Tree

Another alternative to achieve non-uniform wire width within a segment is the optimal wire sizing formula proposed by Chen *at al* [12] very recently. Let $f(x)$ be the wire width at position x of a wire segment. When given the driver resistance and the loading capacitance for the wire segment, Chen et al. show that the Elmore delay through the wire segment is minimized when $f(x) = ae^{-bx}$ where a and b are constants. Furthermore, when the lower and upper bounds for the wire width of a wire segment are given, the optimal wire width function is one of the six truncated forms of ae^{-bx}. In all cases, formulas can be determined in constant time. However, it did not model the fringing capacitance.

In order to apply the optimal wire sizing formula to a routing tree, the authors propose to minimize the weighted combination of Elmore delays from the source to multiple sinks. A procedure like the GWSA algorithm developed in [41] is used. First, the minimum wire width is assigned to every segment. Then, the optimal wire sizing formula is *iteratively* applied to each wire segment until no improvement can be achieved. In contrast to the case of a single wire segment, the total upstream weighted resistance is used to replace the driver resistance, and the total downstream capacitance to replace the loading capacitance. The resulting wire width is continuous and non-uniform within a wire segment. Note that when a discrete wire sizing solution is needed, the mapping from a continuous solution to a discrete solution may lose its optimality.

2.2.2 Wiresizing to Minimize Maximum Delay or Achieve Target Delay

In addition to minimizing the weighted combination of delays, wire sizing methods have been developed to minimize the maximum delay or achieve a target delay. We will present first the wire sizing work [91] to minimize the maximum delay under the Elmore delay model, then the wire sizing work [75] to achieve the target delay under a higher-order RC delay model, and finally the wire sizing work [109] to minimize the maximum delay for a tree of transmission lines under a lossy transmission line model. Note that the Elmore delay model is suitable for formulations that minimize the weighted sum of delays for current CMOS designs, since it has high *fidelity* with respect to the SPICE-computed delay for the wire sizing optimization, which is verified by the experiments in [30] based on the 0.5 μm CMOS designs. On the other hand, in order to achieve the *target delay* or handle MCM designs, more accurate delay models are required as in [75, 109]. Furthermore, several iterations of the procedures to minimize the weighted delay can be used to minimize the maximum delay or achieve the target delay by adjusting the weight penalty assignment in practice. Particularly, the Lagrangian relaxation wire sizing work [11] proposes an optimal method to assign the weight penalty, which will be presented in Subsection 2.2.2.

2.2.2.1 Single-Source RC Tree under Elmore Delay Model

Sapatnekar [91] studied the wire sizing problem to minimize the maximum delay under the Elmore delay formulation of Eqn. (3.1). First, he showed that the separability no longer holds for minimizing the maximum delay. So, the dynamic programming based approach in [42, 40] does not apply. However, since the Elmore delay in an RC tree is a posynomial function of wire widths as first pointed out in [50], it has this property that the local optimum is also the global optimum; thus a sensitivity-based method like that used in [50] can be applied.

The algorithm in [91] goes through a number of iterations. In each iteration, the sink with the largest delay is identified and the sensitivity S_i given in the following is computed for each wire segment i on the path from the source to the identified sink:

$$S_i = \frac{Delay(F \cdot w_i) - Delay(w_i)}{(F-1) \cdot w_i}$$

where $Delay(w_i)$ is the delay from the source to the identified sink and F is a constant larger than 1 (set to 1.2 or 1.5 in [91]). Intuitively, the sensitivity is the delay reduction of unit wire area increment. For all wires on the path from the source to the identified sink, the width of the wire with the minimum negative sensitivity will be multiplied by $F > 1$. The iteration is stopped when no wire has a negative sensitivity or the delay specification is satisfied. Since a posynomial function can be mapped into a convex function, the convex programming technique developed in [101, 94] was also applied by Sancheti and Sapatnekar [90] to achieve the exact solution at higher computation costs.

Very recently, Kay *et al* [61] proposed a sensitivity-based two-phase wire sizing algorithm to minimize area under delay constraints for all sinks. In the first phase, they start with maximum wire width and iteratively de-size a wire segment to minimize the critical delay in a similar fashion as [91]. In the second phase, they iteratively de-size a wire segment such that the product of sum of slacks with respect to delay constraints (negative slacks are prohibited) and the area reduction is maximal. Furthermore, they also propose to iteratively de-size two wire segments each time in the second phase.

2.2.2.2 Single-Source RC Tree under Higher-Order RC Delay Model

Menezes *et al* [75] used a moment fitting approach to wiresize RC interconnect trees in order to achieve the target delays and slopes at critical sinks. Let *target moments* be moments for the 2-pole transfer functions that have the target delays and slopes at critical sinks, and *real moments* those for the transfer function under the current wire width assignment for the RC tree, Menezes et al. propose to modify the wire width assignment in the RC tree to match the real moments with the target moments so that the target delays and slopes will be obtained.

The sensitivities of real moments with respect to the wire widths are used to guide the search for the proper wire widths. The method works as follows: First, for each sink, a 2-pole transfer function is generated so that it has the target delay and slope at the sink. For each transfer function, the first four target moments are obtained. Then, the first four real moments are computed for each sink based on the recursive method developed in [82], which computes the higher moments from the lower moments, and a $O(MN^2)$ method is proposed to compute the sensitivities with respect to the wire widths for real moments, where M is the number of critical sinks and N the number of wire segments. Finally, such sensitivity values guide the search for wire widths to minimize the *mean square error* between the first four target moments and the first four real moments for *every* critical sink.

Furthermore, the following is proposed in order to achieve the solution with smaller area: each wire is assigned a weight in order to favor those wires which are related to the more critical sinks and those wires with respect to which the critical sinks exhibit larger Elmore delay sensitivities. Widening those wires has the maximum effect on delay with a minimal area penalty. Moreover, the delay sensitivity with respect to the driver area is also computed and compared with the delay sensitivity with respect to the interconnect area to empirically determine whether a larger driver should be used. The approach is extended in [76] to conduct simultaneous gate and interconnect sizing, which will be presented in Section 2.3. Note that the algorithm in [75], similar to [91, 90], assumes continuous wire width choices for their wire sizing solutions.

2.2.2.3 Single-Source Tree of Transmission Lines under Lossy Transmission Line Model

The wire sizing work by Xue and Kuh in [109] takes the wire inductance into account by modeling each wire segment as a lossy transmission line, and sizes the wire segments in an interconnect tree to minimize the maximum delay. The maximum delay and its sensitivities with respect to wire widths are computed via high order moments. Based on the exact moment matching method in [111], the higher moments and their sensitivities with respect to the wire widths are computed recursively from the lower moments and the sensitivities can be computed analytically. Thus, the maximum delay and its sensitivities with respect to the wire widths can be computed efficiently. The following procedure is repeated to reduce the maximum delay: First, one computes the high order moments, the maximum delay (t_d) and its sensitivity with respect to every wire width ($\frac{\partial t_d}{\partial w_i}$). Then, if a wire segment e_i has the maximum $\mid \frac{\partial t_d}{\partial w_i} \mid$, e_i will be assigned either the next larger or smaller wire width, based on the polarity of $\frac{\partial t_d}{\partial w_i}$. The procedure iterates until the sensitivities of the maximum delay becomes small.

[109] showed the following experimental results: The 2-pole transfer function with moments m_0, m_1 and $m_2 (m_0 = 0)$ is reasonably accurate when compared

with SPICE2. The approach can reduce the rising delay in the critical sink by over 60% with a small penalty in routing area [5]. The monotone property is still true under this lossy transmission line formulation. The final wire sizing solution reduces the overshoot and is more robust under parameter variation.

2.2.2.4 Weighted Delay Formulation versus Maximum Delay Formulation

All the wire sizing algorithms presented in Section 2.2.1 for minimizing the weighted sum of delays can be used to minimize the maximum delay by iteratively adjusting the weights so that the sinks with larger delays have higher weights. In particular, Chen *et al* [11] showed that for the continuous wire sizing formulation where the wire width can be any value between the lower and upper bounds, the weighted delay formulation is able to optimally minimize the maximum delay among all sinks. They formulated the following Lagrangian relaxation problem:

$$\begin{array}{ll} Minimize & t_{max} + \sum\limits_{s_i \in sink(T)} \lambda_i (t_i(\mathcal{W}) - t_{max}) \\ Subject\ to & t_i(\mathcal{W}) < t_{max} \end{array}$$

where $t_i(\mathcal{W})$ is the delay from the source to sink s_i under the current wire sizing solution $\mathcal{W}$ and t_{max} is the maximum delay from the source to all sinks.

The following two-level algorithm was proposed in [11]: in the outer loop, the weights associated with the delays from the source to sinks are dynamically adjusted, which are basically proportional to the delays at the sinks. In the inner loop, the continuous wire sizing solution is computed for the given set of weights,

[5] Note that the delay in a tree of transmission lines is the sum of flying time and the rising delay of the response waveform. Wiresizing only affects the rising delay, and the delay reduction means the reduction of the maximum rising delay at threshold voltage of 0.5Vdd

by the wire sizing algorithm [12] (Section 2.2.1) to minimize the weighted linear combination of delays. They showed that the Lagrangian relaxation iteration will converge to an optimal solution in terms of maximum-delay minimization. Moreover, the authors expanded their Lagrangian relaxation based algorithm to simultaneous wire and buffer sizing for buffered clock trees to minimize the weighted combination of delay, power and area minimization, and to address the problem of skew and sensitivity minimization for clock trees.

2.3 Simultaneous Device and Wire Sizing

The device sizing works presented in Section 2.1 model the interconnect as a lumped loading capacitor and do not consider the possibility of sizing the interconnect. On the other hand, the wiresizing works presented in Section 2.2 model the driver as a fixed effective resistor and do not consider the need to size the device again after interconnects have been changed. Both approaches may lead to suboptimal designs. As a result, a number of recent studies size both devices and interconnects simultaneously. These methods will be discussed in this subsection.

2.3.1 Simultaneous Driver and Wire Sizing

The simultaneous driver and wire sizing problem for delay minimization (SDWS/D problem) was studied by Cong and Koh [38]. The switch-level model is used for a driver and both the gate and the drain (output) capacitances of the transistor are taken into account, while the interconnect tree is modeled by a distributed RC tree as was used in [41]. The objective function is to minimize the summation of the delay for cascaded drivers and the weighted delay for the RC tree. The SDWS/D algorithm is based on the following important relation between the driver size and the optimal wire sizing:

Driver and Wire Sizing Relation [38]: Let R_d be the effective resistance for the last stage driver and $\mathcal{W}^*$ be the optimal wire sizing solution for driver resistance R_d. If R_d is reduced to R_d', the new corresponding optimal wiresizing solution $\mathcal{W}'^*$ dominates $\mathcal{W}^*$.

The core for the SDWS/D algorithm is the procedure to compute the optimal driver and wire sizing when given a stage number k, which works as follows. First, the algorithm starts with the minimum wire width assignment and computes the capacitive load of the routing tree. Then, it computes the optimal sizes of the k cascaded drivers based on Lin-Linholm Theorem in Section 2.1.1. Next, the optimal wiresizing algorithms (GWSA followed by OWSA) developed in [41] are performed on the routing tree based on the effective resistance of the last driver. If the wire width assignment changes, the new driver sizes are obtained according to Lin-Linholm Theorem. Then, the optimal wiresizing solution will be computed again based on the new size of the last driver. The process is repeated until the wire width assignments do not change in consecutive iterations. In this case, the lower bounds are obtained for the optimal sizes of both the drivers and the wire segments.

The upper bound for the optimal sizing solution can be obtained similarly by beginning with the maximum wire width assignments. If the lower and upper bounds meet, the optimal solution is achieved, which occurs in almost all cases as shown in the paper. Otherwise, the size of the last driver is enumerated between the lower and upper bounds. The corresponding optimal wire sizes and the first $(k-1)$ driver sizes are computed, and the optimal k-driver SDWS/D solution is selected for this set.

The overall SDWS/D algorithm computes the optimal number of stages by a linear search, increasing k starting with $k = 1$. The process terminates when stage k does not perform better than stage $k-1$ (i.e. when adding an additional driver actually slows down the circuit). Then, the optimal sizing solution for the $k-1$ stage drivers and the corresponding optimal wiresizing is the optimal SDWS/D solution. In practice, the runtime of SDWS/D is on the same order as k times the runtime of the GWSA algorithm followed by the OWSA algorithm to compute the optimal wiresizing algorithm. The simultaneous driver and wire sizing problem for power minimization was also studied in [38] and the efficient optimal algorithm was developed. Accurate SPICE simulation shows that the method reduces the delay by up to 12%–49% and power dissipation by 26%–63% compared with existing design methods. Very recently, Cong, Koh, and Leung [39] extended the work on SDWS to handle driver/buffer and wire sizing for buffered interconnects. However, both [38, 39] do not consider the waveform slope effect during the computation of the optimal driver/buffer sizes.

2.3.2 Simultaneous Gate and Wire Sizing

Recently, Menezes et al. [76, 74] studied the simultaneous gate and wire sizing problem for different objectives: to achieve the target delays in [76], and to find the minimal-area solution to satisfy the performance requirement in [74].

2.3.2.1 Simultaneous Gate and Wire Sizing to Achieve Target Delay

The algorithm proposed by Menezes *et al* in [76] is the extension of the moment fitting method for wiresizing [75] (Section 2.2.2.C) to the simultaneous gate and wire sizing problem. Again, let *target moments* be moments for the 2-pole transfer functions that has the target delays, and *real moments* those for the transfer function under the current widths of all wires and gates, the sensitivities of the real moments with respect to the wire and gate widths will guide the search for wire and gate widths to match the real moments and target moments.

A higher-order RC delay model is used for the interconnect tree as in [75]. Meanwhile, all transistors in a gate are assumed to scale by the same factor, which allows that a gate can be described by its "width" w_g. The gate is modeled by the single-resistor voltage-ramp model as proposed in [46], which can accurately estimate the driver delay as well as output waveform slope. The sensitives with respect to the gate and wire widths for real moments can be computed, which are used to guide the changes of gate and wire widths to achieve the target delay for a stage by the aforementioned moment-fitting method (in this work, a *stage* is a dc-connected path from the voltage source in the gate model to a sink).

Furthermore, the algorithm in [76] handles a *path*, which contains cascaded stages. It is also based on the sensitivity guided moment-fitting method. The following assumption is made to simplify the sensitivity computations: given two successive stages n and $n+1$ in a path, first, except the gate of stage $n+1$, no wire/gate in stages $n+1, n+2, \cdots$ affects the delay in stage n; Second, sizing the gate or a wire in stage n only affects the input transition time to the gate in stage $n+1$, not those in stages $n+1, n+2, \cdots$. In their experiment, the objective for each PI-PO path was a 50% delay reduction, through gate sizing only and simultaneous gate and wire sizing, respectively. It was shown that for larger delay reductions, simultaneous gate and wire sizing could achieve lower area and that gate sizing only could not reach 50% delay reduction because the path delay was dominated by the interconnect delay. The trade-off between the area and the delay reduction was shown as well.

2.3.2.2 Simultaneous Gate and Wire Sizing to Satisfy Performance Requirement

The simultaneous gate and wire sizing approach by Menezes *et al* [74] is aimed at finding the minimal-area solution to satisfy the performance requirement. First, the driver is modeled by a fixed resistance driven by a step waveform and the delay of the interconnect tree is modeled by the Elmore delay model. The path delay in this case is a posynomial function of both gate and wire widths and the simultaneous gate and wire sizing problem is a posynomial programming problem which can be transformed into a convex programming problem. The sequential quadratic programming (*SQP*) [6] is used to solve this transformed convex programming problem to achieve an optimal solution.

Then, the delay of the interconnect tree is modeled by the higher-order RC delay while the driver is modeled by a fixed resistance. Although the path delay is no longer a posynomial function of gate and wire widths, the authors assumed that it was near-posynomial so that the SQP method could be applied. A q-pole transfer function is used and the sensitivity computation of the poles and residues is conducted during the SQP procedure.

Finally, the driver is modeled by the more accurate single-resistor voltage-ramp model [46]. Again, the near-posynomial is assumed for path delay and the SQP method is applied. The sizing results showed that the fixed-resistance driver model could lead to undersized solutions. RC meshes (non-tree interconnects) can be handled by the SQP method, again under the assumption that the delay

[6] According to [83], the SQP method reduces the nonlinear optimization to a sequence of quadratic programming (QP) subproblems. At each iteration, a QP subproblem is constructed from a quadratic linearization of both the objective function and the constraints about the solution from the previous iteration. The solution of the current iteration is then used as an initial solution for the next iteration. The iteration converges to a solution for a convex programming problem.

formulation is near-posynomial.

2.3.3 Simultaneous Buffer Insertion and Wire Sizing

2.3.3.1 Discrete Formulation

The polynomial-time dynamic programming algorithm for the buffer insertion problem [102] was generalized by Lillis *et al* in [65] to handle the simultaneous wiresizing and buffer insertion for both delay and power minimization. The slope effect on the buffer delay was also taken into account. Only the delay minimization feature will be discussed in the following.

Different from [102], when a wire segment of length l (with upstream node k) is added at the root i of a dc-connected subtree, a new option (q_k, c_k) will be generated at k for every wire width choice w and every (q_i, c_i) at i as the following:

$$\begin{aligned} q_k &= q_i - \frac{r}{w} \cdot l \cdot (\frac{c_a \cdot w \cdot l}{2} + c_i) \\ C_k &= c_i + c_a \cdot w \cdot l \end{aligned}$$

The non-uniform wiresizing can be easily carried out by just introducing 2-degree Steiner points within a wire segment, and the other two bottom-up rules to compute new options (with extension to multiple inverter sizes and consideration of the signal polarity) and the rule to prune suboptimal options given in [102] can be applied without any modifications. The number of total options at the source of the routing tree is still polynomial bounded.

According to [54], the delay of an inverter is the delay under the step input plus an increment due to the input slope. The increment is proportional to the input waveform transition time. By assuming that the delay increment due to the input slope is proportional to the Elmore delay D_{prev} in the previous stage,

[65] further formulated the following buffer (inverter) delay for the downstream capacitance c_k.

$$buf\_delay_s(b, c_k) \quad = \quad buf\_delay(b, c_k) + \lambda_b D_{prev}$$

where buf_delay(b, c_k) equals to $D_{buf} + R_{buf} \cdot c_k$ with D_{buf} being the intrinsic delay of an inverter and D_{prev} being the Elmore delay of the previous wire path.

Because the dynamic programming works from the bottom-up and D_{prev} is unknown, the option is re-defined as (f, c) instead of (q, c) when considering the slope effect, where f is a piece-wise linear function and $f(x) = q$ is the optimal required arrival time q for the downstream capacitance c and $D_{prev} = x$. With this new definition for the option, the number of total options at the source of a routing tree is no longer polynomially bounded in the theoretical sense. However, it was observed in [65] that the run time of the new version is comparable to that of its simpler version assuming step-input to buffers.

Discrete buffer sizes and wire widths are used in [65]. Very recently, Chu and Wong [19] studied the simultaneous buffer insertion and wire sizing problem for continuous wire width and buffer size values. They assume that wire width is uniform within a wire segment, and derived closed-form formula to the following problems for a wire line: wire sizing using no buffers, wire sizing using given number of buffers, and wire sizing using optimal number of buffers. Their formula gives wire segmentation and wire widths, as well as buffer location and sizes. Other assumptions for their formula includes: there is no bounds on buffer sizes and wire widths, and they consider only area capacitance for wires. Under these assumptions, they proved an interesting result: buffers can be placed anywhere without affecting the delay value under their optimal buffer and wire sizing solutions. Extension to tree structures is planned as their future work.

2.4 Simultaneous Topology Construction and Sizing

All wire and device sizing works presented up to now assume that the topology of interconnects is given, which can be called *static* sizing. Recently, dynamic wiresizing has been studied, where the wiresizing is performed during interconnect construction. Furthermore, simultaneous interconnect construction, buffer insertion and sizing, and wiresizing has been studied in order to achieve even better designs.

2.4.1 Dynamic Wiresizing during Topology Construction

Hodes, McCoy and Robins [58] propose a method to do wiresizing *dynamically* during tree construction. They combine the Elmore Routing Tree (ERT) algorithm [8] and the GWSA algorithm [40] (Section 2.2.1) as follows: starting with a degenerate tree initially consisting of only the source pin, grow the tree at each step by finding a new pin to connect to the tree in order to minimize the Elmore delay in the current *wiresized* topology. In other words, in each step they invoke the GWSA algorithm for each candidate edge and add the edge that yields the wiresized tree with the minimal maximum delay. After the construction spans the entire net, the GWSA algorithm is invoked once more to wiresize the entire tree, starting with the minimal width.

Recently, Xue and Kuh [108, 107] propose insertion of multi-links into an existing routing tree and do dynamic wiresizing during the link insertion in order to minimize the maximum delay. The Elmore delay formulation for RC meshes in [106] is used. The algorithm works as follows: Given a routing tree with a performance requirement, the sink n_{max} with the maximum delay is identified. A wire link e is established between the source and n_{max}. While the performance requirement is not met and n_{max} remains the most critical (i.e., still has the max-delay), e is assigned with non-uniform wire width. Suppose n'_{max} becomes the most critical sink after wiresizing on e. If there is a direct link e' from source to n'_{max}, then the algorithm sizes the wire of e' instead until n'_{max} is no longer the most critical sink or the delay requirement is met. If there is no direct link e' from source to n'_{max}, e' will be established only if further wiresizing of e can not satisfy the performance requirement with less area than creating the new link e'. The wiresizing is formulated as a Sequential Quadratic Programming (SQP) problem. Moreover, non-uniform wiresizing is achieved by dividing every segment into a number of sub-segments defined by the user. Because the sink with the maximum delay also has the maximum skew, minimization of the maximum delay also minimizes the maximum skew.

2.4.2 Simultaneous Tree Construction, Buffer Insertion and Wiresizing

Most recently, Okamoto and Cong [79] study the simultaneous tree construction, buffer insertion and wiresizing problem[7]. The following techniques are combined to develop a *wiresized buffered A-tree (WBA-tree)* algorithm: the A-tree algorithm for tree construction [42], the simultaneous buffer insertion and wiresizing algorithm [102, 65], *critical path isolation*, and a *balanced load decomposition* used in logic synthesis. In logic synthesis, when one or several sinks are timing-critical, the critical path isolation technique (Figure 2.4(a)) generates a fanout tree so that the root gate drives only the critical sinks and a smaller additional load due to buffered non-critical paths. On the other hand, if required times at sinks are within a small range, balanced load decomposition (Figure 2.4(b)) is ap-

[7]An early version of this work considers only simultaneous topology construction and buffer insertion [80].

plied in order to decrease the load at output of root gate. These transformations are applied recursively in a bottom-up process from the sinks in the same manner as the A-tree and the simultaneous buffer insertion and wiresizing algorithms.

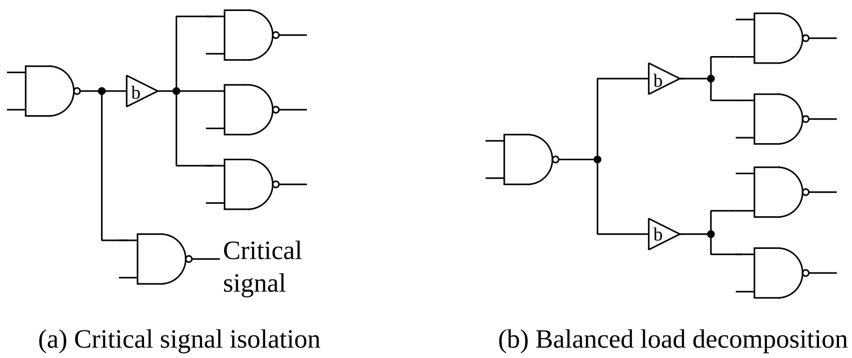


Figure 2.4: Fanout optimization in logic synthesis

As in the buffer insertion algorithm of [102] (Section 2.1.3), the WBA algorithm include two phases: the bottom-up synthesis procedure and the top-down selection procedure. Similar definitions of the option and the pruning rule are used. Recall the heuristic move in the A-tree algorithm [42] merges subtrees recursively in the bottom-up manner, starting from the set of subtrees, each containing a single sink. Let T_i be subtree rooted in node i, the following basic steps are iterated in the bottom-up synthesis procedure.

- Select v and w with considering critical path isolation and balanced load decomposition.
- Merge T_v and T_w to T_r, and compute a set of options at r.

In order to select the pair of v and w (equivalent to T_v and T_w) to merge, first, the following concepts are defined:

The distance between the source and the merging pair of v and w, denoted D_{vw}, is defined as $D_{vw} = \min(v_x,\ w_x) + \min(v_y,\ w_y)$. This definition is for the case that v and w are in the first quadrant with s_0 at the origin. Other cases can be defined in a similar way.

The maximum possible required time at the root r of subtree T_r generated by merging of T_v and T_w, denoted R_{vw}, is defined as $R_{vw} = \max_{z \in Z_r} q_z$, where r is the merging point of T_v and T_w, and Z_r is a set of options at r.

The maximum R_{vw} among all possible merging pairs v and w in the set of roots $ROOT$ of the current subtrees, denoted $R_{max}(ROOT)$, is defined as $R_{max}(ROOT) = \max_{v,w \in ROOT} R_{vw}$

The merging cost for v and w is defined as $merge\_cost(v, w, ROOT) = \alpha * R_{vw} + (1 - \alpha) * D_{vw}$ where α is a fixed constant with $0 \leq \alpha \leq 1.0$.

Then, the v and w pair with the maximum $merge\_cost$ is the one to be merged. The idea behind it is as follows: we want to maximize the required arrival time in the source pin so that we wish that the R_{vw} is as large as possible. Meanwhile, we want to minimize the total wire length, intuitively, we wish that D_{vw} is as large as possible. Note that, when $\alpha = 0$, it is equivalent to the heuristic move in [42].

The option computation and pruning can be carried out in a manner similar to [102, 65] after each merging of T_v and T_w. Overall, after the bottom-up synthesis procedure to construct tree and compute options, the top-down selection procedure is invoked. It chooses the option which gives the maximum required time and the minimum total capacitance at the source pin, then traces back the computations in the first phase that led to this option. During the back-trace, the buffer positions and wire width of each segments

Similarly, Lillis *et al* studied the simultaneous tree construction and wiresizing problem [66] and the simultaneous tree construction and buffer insertion problem [67], respectively. In fact, their method can generalized to handle the simultaneous tree construction, buffer insertion and wiresizing problem as well. In short, during the dynamic program scheme to construct a P-Tree [66] in a bottom-up manner for a given permutation, a set of options are computed for each subtree as in [102, 65] and the same option pruning rule is applied.

CHAPTER 3

Wiresizing Optimization for Nets with Multiple Sources

All wiresizing works [42, 41, 91, 107, 12, 11, 75, 109] and simultaneous device and wire sizing works [38, 74, 65] reviewed in Chapter 2 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 [90, 75, 74, 107, 11] or the sensitivity analysis [91, 109] 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 optimal wiresizing problem for MSITs under the RC tree model and the Elmore delay model. Our contributions include:

- We formulate the multi-source wiresizing (*MSWS*) problem. Decomposing an MSIT into a source subtree (*SST*) and a set of loading subtrees (*LST*s), 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 bundle refinement property which 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 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 [7]. 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 [28, 30]. 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 3.1, we present the formulation of the *MSIT* wiresizing problem. In Section 3.2 and 3.3, 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 3.4. Section 3.5 shows experimental results, including the fidelity study of the Elmore delay model. Section 3.6 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 [27].

3.1 Problem Formulation

3.1.1 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 [41] is not further divided and is simply treated as a uni-segment[1], and a segment in [38] 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 [41, 38] 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 3.3. For simplicity, we assume that an *a priori* fixed segment-division is given in this section.

The modeling technique similar to those used in [41] 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_0 and wire fringing capacitance c_1, then $r_E = r_0 \cdot \frac{l_E}{w_E}$ and $c_E = c_0 \cdot w_E \cdot l_E + c_1 \cdot l_E$ 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 [49] 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. (3.1) according to the Elmore delay

[1] we note that artificial degree-2 Steiner points can be introduced within a segment in [41] to achieve certain segment-division.

formulation for RC trees [85].

$$t^{ij}(MSIT, \mathcal{E}, \mathcal{W}) \;=\; \sum_{E \in P(N_i, N_j)} r_E \cdot (\frac{c_E}{2} + C_E) \tag{3.1}$$

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. (3.2).

$$t(MSIT, \mathcal{E}, \mathcal{W}) = \sum_{N_i \in src(MSIT)} \sum_{N_j \in sink^i(MSIT)} \lambda^{ij} \cdot t^{ij}(MSIT, \mathcal{E}, \mathcal{W}) \tag{3.2}$$

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 3.3.

3.1.2 Weighted Delay Formulation

For simplicity, we assume that all interconnects belong to the same layer and the assumption will be removed later in Section 3.2.3. 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:

$$
\begin{aligned}
t^{ij}(MSIT, \mathcal{E}, \mathcal{W}) \;=\; & \mathcal{K}_0^{ij} + \mathcal{K}_1^i \cdot \sum_{E \in MSIT} l_E \cdot w_E + \mathcal{K}_2 \cdot \sum_{E,E' \in MSIT} f^{ij}(E,E') \cdot \frac{l_E \cdot l_{E'} \cdot w_{E'}}{w_E} + \\
& \mathcal{K}_3 \cdot \sum_{E,E' \in MSIT} f^{ij}(E,E') \cdot \frac{l_E \cdot l_{E'}}{w_E} + \mathcal{K}_4 \cdot \sum_{E} g^{ij}(E) \cdot \frac{l_E}{w_E} + \mathcal{K}_5 \cdot \sum_{E \in MSIT} h^{ij}(E) \cdot \frac{l_E^2}{w_E}
\end{aligned}
\tag{3.3}
$$

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:

$$
\begin{aligned}
\mathcal{K}_0^{ij} &= R_d^i \cdot C_d^i + R_d^i \cdot \sum_{u \in sink^i(MSIT)} c_s^u + R_d^i \cdot \sum_{E \in MSIT} c_1 + \sum_{E \in P(N_i,N_j)} \frac{r_0 \cdot c_0}{2} \\
\mathcal{K}_1^i &= R_d^i \cdot c_0 \\
\mathcal{K}_2 &= r_0 \cdot c_0 \\
\mathcal{K}_3 &= r_0 \cdot c_1 \\
\mathcal{K}_4 &= r_0 \\
\mathcal{K}_5 &= \frac{r_0 \cdot c_1}{2}
\end{aligned}
$$

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.

$$
f^{ij}(E,E') = \begin{cases} 1 & if\ E \in P(N_i,N_j)\ and\ E' \in Des^i(E) \\ 0 & otherwise \end{cases} \tag{3.4}
$$

$$g^{ij}(E) = \begin{cases} \sum_{u \in sink^i(E)} c_s^u & if\ E \in P(N_i, N_j) \\ 0 & otherwise \end{cases} \tag{3.5}$$

$$h^{ij}(E) = \begin{cases} 1 & if\ E \in P(N_i, N_j) \\ 0 & otherwise \end{cases} \tag{3.6}$$

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. (3.2) becomes:

$$t(MSIT, \mathcal{E}, \mathcal{W}) = \mathcal{K}_0 + \mathcal{K}_1 \cdot \sum_{E \in MSIT} l_E \cdot w_E + \mathcal{K}_2 \cdot \sum_{E, E' \in MSIT} F(E, E') \cdot \frac{l_E \cdot l_{E'} \cdot w_{E'}}{w_E} + \mathcal{K}_3 \cdot \sum_{E, E' \in MSIT} F(E, E') \cdot \frac{l_E \cdot l_{E'}}{w_E} + \mathcal{K}_4 \cdot \sum_{E \in MSIT} G(E) \cdot \frac{l_E}{w_E} + \mathcal{K}_5 \cdot \sum_{E \in MSIT} H(E) \cdot \frac{l_E^2}{w_E} \tag{3.7}$$

where

$$\mathcal{K}_0 = \sum_{N_i \in src(MSIT)} \sum_{N_j \in sink^i(MSIT)} \lambda^{ij} \cdot K_0^{ij} \tag{3.8}$$

$$\mathcal{K}_1 = \sum_{N_i \in src(MSIT)} \sum_{N_j \in sink^i(MSIT)} \lambda^{ij} \cdot K_1^{i} \tag{3.9}$$

$$F(E, E') = \sum_{N_i \in src(MSIT)} \sum_{N_j \in sink^i(MSIT)} \lambda^{ij} \cdot f^{ij}(E, E') \tag{3.10}$$

$$G(E) = \sum_{N_i \in src(MSIT)} \sum_{N_j \in sink^i(MSIT)} \lambda^{ij} \cdot g^{ij}(E) \tag{3.11}$$

$$H(E) = \sum_{N_i \in src(MSIT)} \sum_{N_j \in sink^i(MSIT)} \lambda^{ij} \cdot h^{ij}(E) \tag{3.12}$$

Our MSWS problem is aimed to find the optimal w_E's to minimize the weighted delay formulation Eqn. (3.7). Although this weighted delay formulation for multiple sources and multiple sinks is very similar to that for the single source and multiple sinks in [41], 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 3.2.

3.2 Properties of Optimal MSWS Solutions

The single-source wiresizing problem (SSWS) under the an *a priori* fixed segment-division was studied in [41], 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 3.3 will enable us to apply the algorithms developed in [41] to the MSWS problem to certain extent and to develop even more efficient algorithms in Section 3.4.

3.2.1 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 3.1). 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.

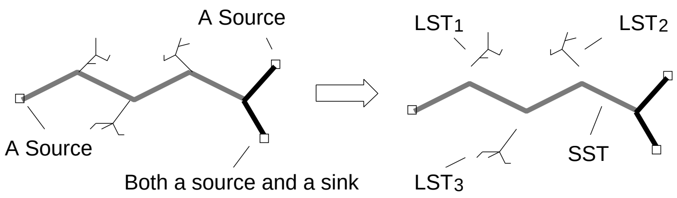


Figure 3.1: An $MSIT$ can be decomposed into the source subtree SST, and a set of loading subtrees (three LSTs here) branching off from the SST. The dark segments belong to the SST.

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

3.2.2 Properties of Optimal MSWS Solutions

A. LST Separability

Theorem 1 *Given the wire width assignment of the SST, the optimal width as-*

[2]Note that SST defined here is different from that defined in [41], where SST is used to denote a single stem tree.

signment 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 3.1, 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 [41] 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 3.4 is shown in Figure 3.2. 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.

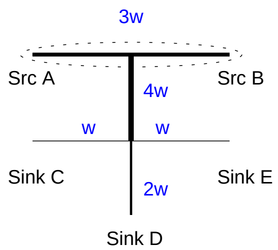


Figure 3.2: The optimal wire width assignments for a two-source net with W being the minimum wire width. The SST is surrounded by the dashed curve. Segments outside the curve belong to an LST. The wire width assignment is *monotone* within the LST. However, the root uni-segment of the LST is wider than uni-segments in the SST.

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. (3.10) is an invariant (denoted $F_l(S)$) if E_1 is left to E_2, and $F(E_1, E_2)$ is another invariant (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. (3.7) 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 [41] 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 [41] 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 3.3.2.

3.2.3 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 [41], 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 3.3, same as the dominance property, hold for any layer assignment and any allowable minimum or maximum wire width.

3.3 Properties of Optimal MSWS/E Solutions

Up to now, both the *MSWS* problem defined here and the *SSWS* problem studied in [41, 91, 75, 74, 65, 107, 109] 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 3.3.2 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.

3.3.1 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 (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.

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

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.*

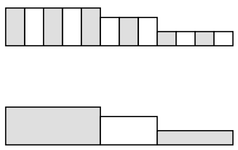


Figure 3.3: (a) The optimal wiresizing solution for segment S with twelve uni-segments under the finest segment-division $\mathcal{E}_F$. (b) Segment S contains only three bundled-segments which define a coarser segment-division with fewer computation costs to achieve the wiresizing solution same as that obtained by $\mathcal{E}_F$.

Figure 3.3 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 3.3.a), but just three bundled-segments (Figure 3.3.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.

3.3.2 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. (3.7) 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. (3.7) 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 3.4.1.

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 $\mathcal{W}^*$.*

3.4 Optimal MSWS Algorithm

3.4.1 Bundled Wiresizing Algorithm

Based on the dominance property (Theorem 4), the greedy wiresizing algorithm GWSA [41] 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 3.3) 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 3.1-3.3.

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.

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*

```
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 3.1: gBWSA$_L/U : Given the minimum uni-segment length$ minLength, $an segment-division$ E, $a lower/upper bound$ 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 3.2: 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`$(\mathcal{E}_0, minLength)$

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

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 3.3: 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.

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 3.5.2.

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 3.5.2 and [27]. 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 [91, 75, 74, 107, 109, 65] 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.

3.4.2 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 [41], can be used independently for every LST with respect to the given wire width assignments for the SST. However, 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 [41]. Experimental results with multi-layer MSIT designs will be presented in Section 3.5.

3.5 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 3.4. These parameters are based on the $0.5\mu m$ CMOS technology in North Carolina Microelectronic Center (MCNC) [72]. 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 3.4: Parameters based on MCNC $0.5\mu m$ submicron CMOS technology.

3.5.1 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 [72] 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 [53]. 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 [60] to route these nets. Table 3.5 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 3.5: 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. (3.7) is equivalent to the *average delay* among critical source-sink pairs. Comparisons between different wiresizing solutions are shown in Table 3.6. 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.

3.5.2 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 3.7, 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 3.4.1. 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].

3.5.3 Fidelity of the Elmore Delay Model

The concept of *fidelity* for the Elmore delay model was introduced by Boese *et al* in [7] 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 [32, 34].

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 [7]: 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 3.8. 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 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 [75, 74] does not consider the inductance, either.

3.6 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[41], 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 [40, 41], simultaneous driver and wire sizing [38], simultaneous transistor and interconnect sizing [34] and simultaneous buffer and wire sizing [39].

Simultaneous driver and wire sizing for multi-source nets has been solved by being posed as a CH-program [34], which will be presented in the next chapter. Other recent works on multi-source net optimization include the following: in [51], 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 [64], 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 [90, 74, 11] are able to achieve the optimal continuous wiresizing solution, and the approach in [65] is able to achieve the optimal discrete wiresizing solution. It is worthwhile to mention that the approach in [11] is based on a Lagrangian relaxation procedure to iteratively apply the weighted delay minimizations (same as that in [40, 41] 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 [90, 74, 11]. 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 [67, 79]. Also, a min-cost min-diameter A-tree algorithm has been proposed [43] 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.

Acknowledgments

The authors would like to thank Heming Chan at Intel Design Technology Department for providing the multiple source routing examples.

All Wires in 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	0.2004	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%)
X-axis Wires in M1 and Y-axis Wires in 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 3.6: Multi-source wiresizing results on several nets in an Intel microprocessor layout

	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 of BWSA-based algorithm	1	54.5	465.8	42.3	39.3	67.63

Table 3.7: Total 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 3.8: Average differences in ranking and SPICE-computed delay for Intel nets based on $0.5\mu m$ CMOS technology

CHAPTER 4

Theory and Algorithm of CH-Programs with Application 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 [38], 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 [41, 30]. In [32, 34], 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 [41, 38, 30], 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 [20]. 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 [74], 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 [65]. 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 [11] to optimally assign the weights for the sequence of weighted-delay minimizations. The simultaneous buffer and wire sizing problem was also solved [11].

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 presented for both device delay and interconnect capacitance in the this and next chapters, and then simultaneous device and wire sizing problem under those models are solved.

Our contributions in this chapter include:

- 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 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.

- We also show that the LR-based bound-computation algorithm is capable of solving many layout optimization problems in deep submicron IC and/or high-performance MCM/PCB designs. In particular, we apply the algorithm to the simultaneous transistor and interconnect sizing (*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% when compared with the solution given by manual optimization (reported in [17]). Furthermore, the solution obtained by the STIS algorithm under the accurate STL-model achieves up to 15.1% additional delay reduction when compared with the solution obtained by the STIS algorithm under the simple model.

In Chapter 5, we will describe an accurate 2-1/2 D capacitance model for interconnects in DSM designs. In Chapter **??**, we will then extend the STIS formulation to accomodate the concept of simultaneous interconnect sizing and spacing under the 2-1/2 D capacitance model. Note that the simultaenous interconnect sizing and spacing problem will be also solved by the LR-based bound computation algorithm. We believe that the CH-program formulations and the bound-computation algorithm can also be applied to other optimization problems in the CAD field.

The rest of the chapter is organized as follows: we first present the theory and algorithm of CH-programs in Section 4.1. We then describe the STIS problem formualtion in Section 4.2, and apply the LR-based algorithm to solving the STIS problem, for both simple and STL-bounded models. We finally present experimental results in Section 4.2.5, and conclude the chapter in Section 5.4. The proofs of Theorems 8 and 9 are presented in Appendix to this dissertation. Results of this chapter have been presented in conference papers [32, 34, 33], and the journal paper [37].

4.1 Theory and Algorithm for CH-Programs

4.1.1 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:

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

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 [32, 34]. It was shown that many previous works on device and interconnect sizing problems, including the single-source and multi-source wire sizing problems [41, 30], continuous wire sizing problem [13], and simultaneous driver and wire sizing problem [38], 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*:

[1] CH-function was called CH-posynomial in [34, 33]. We renamed it to show that it is not, in general, a posynomial.

Definition 7 (monotonically-constrained CH-function) *Eqn. (4.1) is a monotonically-constrained CH-function, if it satisfies the following monotonic 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 [33], 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 [33].

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 [48] is a function of a positive vector $\mathbf{X}$ having the form $g(\mathbf{X}) =$

[2]We saved the name "bounded" for the type of CH-function defined in Definition 8, which was called the general CH-posynomial in [33].

$\sum_{i=1}^{m} u_i(\mathbf{X})$ with

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

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.

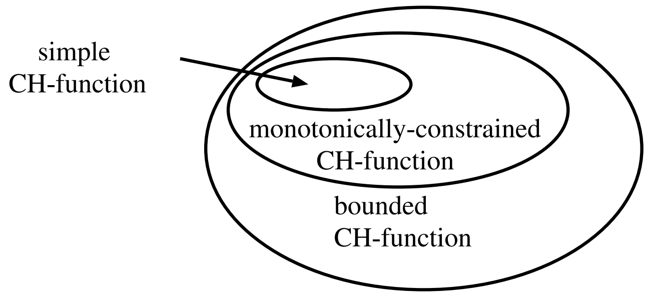


Figure 4.1: The simple CH-function is a subset of the monotonically-constrained CH-function, which is in turn a subset of the bounded CH-function.

4.1.2 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 simple, 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.

4.1.2.1 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

$$\begin{aligned} & g(\mathbf{X}) \\ = \; & \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} A_i(x_i) \cdot B_j(x_j) \end{aligned} \tag{4.6}$$

where $A_i(x_i)$ is a function depending only on x_i, and it increases with respect to an increase of x_i; $B_j(x_j)$ is a function depending only on x_j, and it decreases with respect to an increase of x_j. We have proved the following Lemma 1 in the technical report [31].

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* $\mathbf{X}^*$*.*

Based on Lemma 1, one is easy to verify the following dominance property for the simple CH-program:[3]

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 [41], and was extended to the multi-source wire sizing problem [30]. In [34], it was revealed that the dominance property holds for all simple CH-programs. It was also shown that both wire sizing problems [41, 30], 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.[4]

[3] Nevertheless, Lemma 1 also reveals that the dominance property holds for the monotonically-constrained CH-program when coefficients are functions of single variables, like the bounded-variation CH-program defined in [33]. The dominance property, however, may not hold for the new defined monotonically-constrained CH-program when coefficients are functions of solution vector $\mathbf{X}$.

[4] The SDWS algorithm for simultaneous driver and wire sizing problem in [38] is different from and less efficient than the LR-based algorithm in [34].

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.[5] The LR operation for other CH-programs may be less efficient, however. First, it might be no longer a posynomial program. An 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 the technical report [31]).

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.

4.1.2.2 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*

[5] According to [48], a posynomial program is the following minimization problem:

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

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 4.1.4.

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:

$$f(x_1, x_2) \quad = \quad \frac{a_1(x_1, x_2)}{x_1} \cdot (b_2(x_1, x_2) \cdot x_2^2) \tag{4.7}$$

$$+ \quad \frac{a_2(x_1,x_2)}{x_2} \cdot (b_1(x_1,x_2) \cdot x_1),$$

The pseudo-local refinement of x_1 with respect to $\mathbf{X}' = \{x_1', x_2'\}$ is

$$\tilde{x}_1^{PLR} = \sqrt{\frac{a_1(x_1',x_2') \cdot (b_2(x_1',x_2') \cdot x_2'^3)}{a_2(x_1',x_2') \cdot b_1(x_1',x_2')}}. \tag{4.8}$$

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

$$\tilde{x}_1^{ELR} = \sqrt{\frac{a_1^L \cdot (b_2^L \cdot x_2'^3)}{a_2^U \cdot b_1^U}}. \tag{4.9}$$

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 proved 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 $\mathbf{X}^*$.

The proof can be found in the technical report [31]. Because 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.

4.1.3 LR-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 **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.[6] 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 has been used for computing LR-tight lower and upper bounds for the exact wire sizing solution under fixed c_a and c_{ef} in [41, 30]. When the exact solution has the monotone property like those for the single-source and multi-source wire sizing problems (see Chapter **??**), the bundled refinement operation, which is also called the bundled-LR (*BLR*) operation [28], 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.

[6] 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 in Sections 4.2.5 and **??**.

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.

4.1.4 Comparison with the 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 5). 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 [50], 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 [94], wire sizing [92] and simultaneous gate and wire sizing [74], and was solved by being transformed into the convex program.[7] 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

[7] Same as the method in [74], methods in [94, 92] minimize the maximum delay.

and more general models for the interconnect capacitance and device delay.

Our LR-based algorithm is similar to the coordinate descent approach [70] 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.[8] Because the local optimum is unique for the posynomial program regarding continuous variables, one may even start with an *arbitrary* solution (see [21]) 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.[9] 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 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 **??** 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.

[8] An alternative method, called the steepest descent approach or the gradient method [70], 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 [70]. 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 [21].

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

4.2 Formulation and Solution of STIS Problem

4.2.1 Device and Interconnect Models

A. Simple Model

Almost all simultaenous device and wire sizing works [38, 74, 65] 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 formualtion 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 [81]. 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:

$$r_d = r_{d0}/x \tag{4.10}$$

$$c_g = c_{g0} \cdot x + c_{g1} \tag{4.11}$$

$$c_s = c_{s0} \cdot x + c_{s1} \tag{4.12}$$

$$c_d = c_{d0} \cdot x + c_{d1} \tag{4.13}$$

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.

Onverall, 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 [38, 74, 65]. 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 [30], 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

$$r = r_0/x \tag{4.14}$$

$$c = c_a \cdot x + c_f \tag{4.15}$$

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

B. 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 4.2. We apply HSPICE simulations, and use device parameters for the $0.18\mu m$ technology in Table 5 of the National Technology Roadmap for Semiconductors (NTRS) [97]. 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 4.2, r_0 is clearly *not* a constant. Its value may vary by a factor of 2.

We also computed the capacitance for the *basic geometric structure* (see Figure 5.6), 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 [77] to solve the structure,

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 4.2: Unit-size effective-resistance for n- and p-transistor

using interconnect geometric parameters for the $0.18\mu m$ technology in Table 22 of the NTRS.[10] Figure 4.3(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

[10] The NTRS gives capacitance values only for the minimum width and spacing. Our extracted capacitance values closely match those given in the NTRS (see [35]).

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 5.6) from the center of the victim to the edges of its neighboring wires is fixed. As shown in Figure 4.3(b) for two different center-to-edge spacing, c_{ef} is a not a constant either.

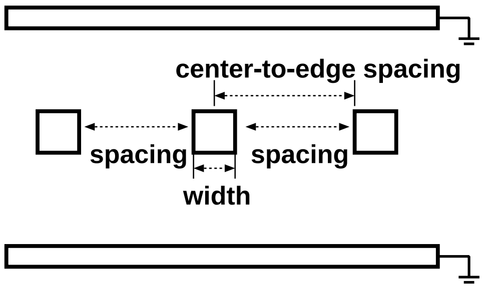
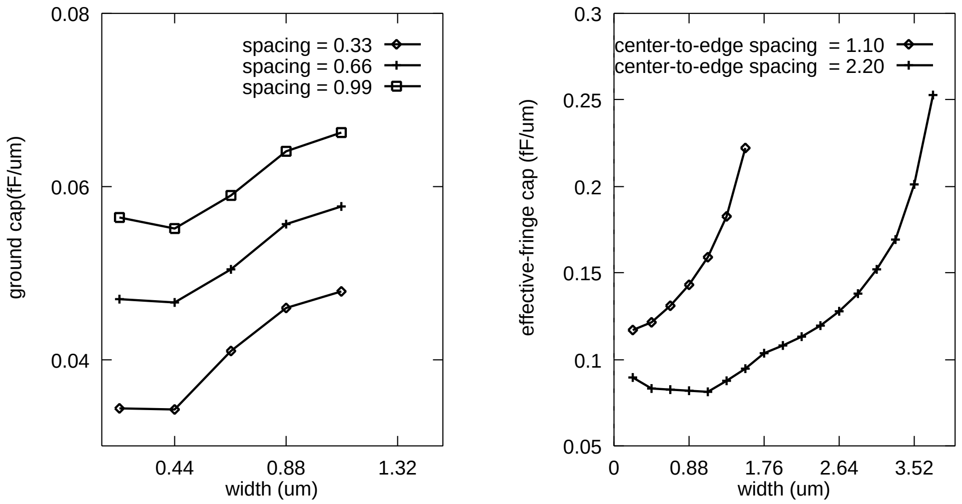


Figure 4.2: The basic geometric structure for capacitance extraction.

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. Little progress has been made for optimization beyond the simple models. The simultaneous buffer insertion and wire sizing algorithm [65] 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 [74] 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 [84]. The resulting sizing prob-



(a) (b)

Figure 4.3: (a) Ground capacitance and (b) effective-fringe capacitance for the central wire (the victim) in the basic geometric structure shown in Figure 5.6. 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.

lem, 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. One very recent works [103, 36] begins to consider coupling capacitance for multiple nets.[11] It allows variable c_{ef} but still assume that r_0 and c_a are constants. Even though all these algorithms still use the simple model for either device delay or interconnect capacitance, their runtime is already high.

We will call the device table, like Table 4.2, *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. In addition, a *WS-bounded* capacitance model will be presented in Chapter 5, where c_a and c_{ef} are determined by the width (w) and spacing (s), and their values are also bounded for any given ranges of w and s. We build tables for the STL-bounded device model via HSPICE simulations, and for the WS-bounded capacitance model via numerical capacitance extractions. These models are more accurate than the simple models, 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. We will first present a 2-1/2D capacitance model (i.e., WS-bounded capacitance model) in Chapter 5, then extend the STIS formulation and algorithm to consider WS-bounded capacitance model in Chapter **??**.

[11]The formulation in [103] is based on the dominant time constant, which is an approximation to the maximum delay among multiple sinks in a net. Because it is difficult to efficiently minimize the sum of the dominant time constants [103], the Elmore delay model (used in this dissertation) is more appropriate for path delay minimization.

4.2.2 Problem formulation

Our delay formulation is similar to those in [81], and is based on the Elmore delay formulation in [85] (see [34] for details). The delay is computed based on *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. (4.16) under the Elmore delay model.

$$\begin{aligned}
& t(P(N_s, N_t), \mathbf{X}) \\
= & \sum_{i,j} f(i,j) \cdot \frac{r_0(i)}{x_i} \cdot c_a(j) \cdot x_j + \sum_{i,j} f(i,j) \cdot \frac{r_0(i)}{x_i} \cdot c_{ef}(j) \\
+ & \sum_{i} g(i) \cdot \frac{r_0(i)}{x_i} + \sum_{i} r_0(i) \cdot h(i) + \sum_{i} h(i) \cdot \frac{r_0(i)}{x_i}
\end{aligned} \tag{4.16}$$

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. Coefficients $f(i,j), g(i)$ and $h(i)$ are determined by the transistor netlist and routing topology.

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

$$t(\mathbf{X}) = \sum_{P(N_s,N_t) \in \mathcal{P}} \lambda_{st} \cdot t(P(N_s, N_t), \mathbf{X}) \tag{4.17}$$

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. (4.17) can be re-written as

$$\begin{aligned}
& t(\mathbf{X}) \\
= & \sum_{i,j} F(i,j) \cdot \frac{r_0(i)}{x_i} \cdot c_a(j) \cdot x_j
\end{aligned}$$

$$
\begin{aligned}
&+ \sum_{i,j} F(i,j) \cdot \frac{r_0(i)}{x_i} \cdot c_{ef}(j) \\
&+ \sum_{i} G(i) \cdot \frac{r_0(i)}{x_i} + \sum_{i} H(i) \cdot \frac{r_0(i)}{x_i}
\end{aligned} \tag{4.18}
$$

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. (4.18) 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 [11]. 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.

4.2.3 Bound computation for the STIS problem

Under the simple models, r_0, c_a and c_{ef} are constants for each wire/transistor, and Eqn. (4.18) 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_{ef} are constants for each wire segment, but will remove the assumption in Chapter **??**.

In the STL-bounded model, r_0 is pre-computed and stored in tables (e.g., see Table 4.2) indexed by the size, input switching time (t_s), and output load (c_l). It could be very accurate depending on the table size.[12] 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-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 [41, 92, 30]) 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

[12]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 4.2.5. For simplicity, we assume that c_l is the lumped capacitance in this paper. Extension to the effective capacitance model [84] is ongoing work and will be discussed briefly in Section 5.4.

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 [30]. We assume that each segment always stays in the same layer, has the fixed r_0, c_a and c_{ef}, as well as same allowable wire widths.[13] 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 [31]. This theorem enables us to use the BLR operation [30] 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 [30].

4.2.4 Overall algorithm for the 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 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.[14]

Our experiments in Section 4.2.5.2 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

[13]Different segments may have different r_0, c_a and c_{ef} if they are in different layers, or have different spacings to neighboring wires.

[14]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 $\mathbf{L}'$.

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.

4.2.5 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.

4.2.5.1 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 [17]. 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.

	# of	wire length	max delay (ns)			average power (mW)		
net	drivers/buffers	(μm)	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 4.3: 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 4.3. 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.

4.2.5.2 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 [97] 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_{ef} values where the wire is $1.10\mu m$ wide and neighboring wires are $1.65\mu m$ away. 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 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 4.4 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 com-

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 4.4: Comparisons between different device and wire sizing formulations

putation 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 4.4 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 as shown in the experiment of Section 4.2.5.1. 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 $41518.2\mu m$ wires, when the transistor sizing formulation is used and wire segments are $10\mu m$ long. Therefore, our STIS algorithm is extremely efficient.

4.3 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 the bundled-LR (*BLR*) operation [28], 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.

We showed that the algorithm is very effective and efficient for many layout optimization problems in deep submicron (*DSM*) designs. It unifies solutions to several problems, including the single-source and multi-source wire sizing problems [41, 30], continuous wire sizing problem [13], and simultaneous driver/buffer and wire sizing problem [38, 11, 39]. Because these problems assume the simple models for the device delay and interconnect capacitance, they are all simple CH-program where the LR operation can be used for bound computations.

Furthermore, we applied the bound-computation algorithm to the simultaneous transistor and interconnect sizing (*STIS*) problem, We used tables precomputed from SPICE simulations to model devic delay, so that our device and interconnect models are much more accurate than many used in previous works. We first showed that the STIS problem is, in general, bounded CH-programs. We 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. Moreover, the algorithms are *extremely* efficient. It took less than 10 seconds to optimize the largest example in this chapter.

Note that the coupling capacitance is not considered in this chapter. We are going to present a 2-1/2D capacitance model with consideration of coupling capacitance in the next chapter, and then study the STIS problem with consideration of the coupling capacitance in Chapter **??**.

CHAPTER 5

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

This chapter addresses post-routing capacitance extraction during performance-driven layout. We first 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 then 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. We will also use this capacitance extraction methodology, i.e., the 2 1/2-D capacitance model, to study the interconnect sizing and spacing problem in the next chapter.

5.1 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 [88, 3, 16, 87, 89, 18, 47, 1, 105] as well as 2-D and 3-D field solvers [86, 100, 96, 78, 52, 77]. 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 capacitances, 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 [86] and Ansoft's products, or finite-difference as in [100, 96] and TMA Raphael) to solve Laplace's equation for potential distribution, then applies Gauss's theorem to yield charge distribution, then applies $Q = [C]V$ to determine self- and mutual capacitances of all conductors. Boundary-element numerical techniques [78] can exploit the flatness and rectilinearity of IC geometries. In addition, multipole algorithms [77] 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 (e.g., they have reached 1.0 in $0.50\mu m$ logic processes and 1.5 in $0.35\mu m$ logic processes, and will reach 2.5 in $0.18\mu m$ logic processes [97]) so that lateral and fringe couplings become much more significant. 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 [18, 4, 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 [110] 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 in 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.

The remainder of this paper is organized as follows. Section 5.2 uses 3-D numerical simulations based on representative 0.5 μm, 0.35 μm and 0.18 μm processes to justify these foundations. The Cadence SE 5.0 2.5-D RC extraction methodology, which is based on these foundations, is then described in Section 5.3; example structures are used to show the accuracy of the methodology. We conclude that the 2 1/2-D extraction method proposed in this paper is sufficient for delay calculation and estimation for performance-driver layout in deep submicron designs.

5.2 Foundations

5.2.1 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 [97]), 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 5.1); 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] 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.

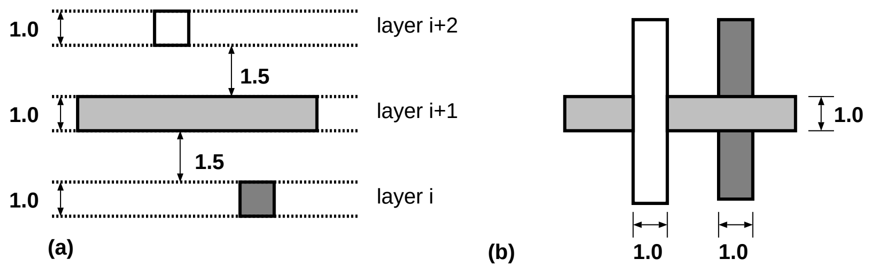


Figure 5.1: For the maximum-density local interconnect structure in $0.50\mu m$ process, we assume (a) the cross-section view – the thickness is 1.0 for all wires, and the height is 1.5 for all dielectrics (inter-layer spacings); (b) the top view – all wire have width 1.0.

[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 [97], 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.

We use an industrial-strength multipole-accelerated 3-D field solver, Fastcap[2] [77], 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.

5.2.2 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 5.2). 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 5.1 shows the following for $0.18\mu m$ process:

[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.

- 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.

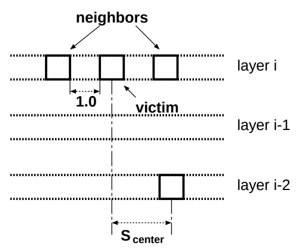


Figure 5.2: Cross-section of a pattern in the first experiment of Section 2.2.

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 5.3). We vary the number of crossunders and observe the change in both $C_{i,i}$ and the ratio $C_{i,i}/C_{i,i-2}$.

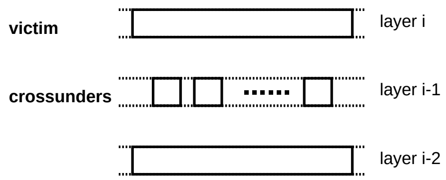


Figure 5.3: Cross-section for a pattern in the second experiment of Section 2.2.

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 capacitance values are given in Table 5.2, 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:

[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.

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.

5.2.3 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 on layer i and one *crossunder* of length 20 on layer $i-1$. The *real* pattern (see Figure 5.4) 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 5.3, 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 $C_{i,i-1}$.

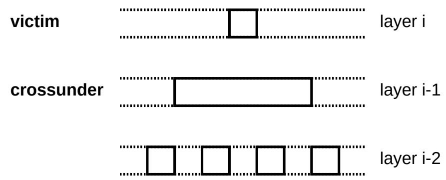


Figure 5.4: Cross-section of the real pattern in the first experiment of Section 2.3: the victim on layer i, one crossunder on layer $i-1$ and a number of wires on layer $i-2$. Layer $i-3$ is a ground plane, but is not shown in the figure.

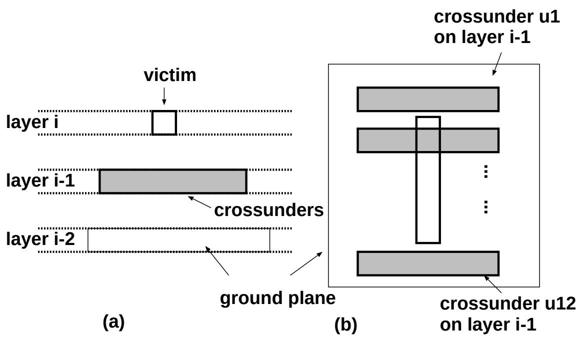


Figure 5.5: Layer $i-2$ is modeled as a ground plane: (a) the cross-section view; (b) the top view.

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 5.5). All wires on layers i and $i-1$ 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 $u1, \cdots, u12$. Table 5.4 shows 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 5.5), 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 $i-1$.

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

[4]Shorter wires mean that the coupling due to the front and end sidewalls, as well as wire

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.

Table 5.5 reports 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 $i \pm 2$.*

5.2.4 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 $i \pm 1$;

---

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.

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 5.6, 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 $r = \infty$ indicates no left or right neighbor). Simulation and derived results are given in Table 5.7. 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 5.8). 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.*

5.2.5 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 5.9). 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

[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.

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.*

5.3 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 [110]. For a *victim* wire on layer i, it looks into a neighborhood structure 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.

5.3.1 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.

5.3.1.1 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 5.6) 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.

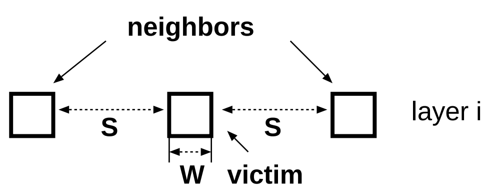


Figure 5.6: The geometric structure on layer i for lateral, area and fringe capacitance generation.

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

$$C_l = (C_{c1} + C_{c2})/(2 \cdot l) \tag{5.1}$$

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 separate the area and fringe components:

$$C_{self} - (C_{c1} + C_{c2}) = 2 \cdot l \cdot C_a + 2 \cdot (l + w) \cdot C_f \tag{5.2}$$

$$C'_{self} - (C'_{c1} + C'_{c2}) = 2 \cdot l \cdot C_a + 2 \cdot (l + w') \cdot C_f \tag{5.3}$$

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.

5.3.1.2 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 5.7(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

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

and s_c. We obtain the total capacitance C_{self} for the central wire (the victim) on layer i.

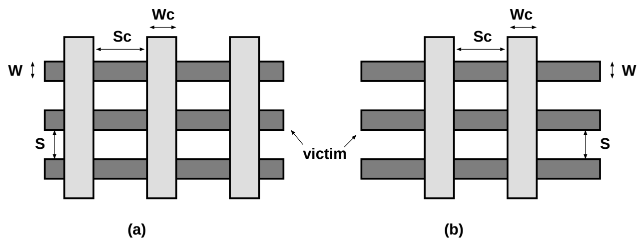


Figure 5.7: The geometric structure on layers i and $i+1$ for crossover correction capacitance generation.

Then, we run FastCap on the pattern in Figure 5.7(b). Difference from the pattern in Figure 5.7(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

$$C_{over} = (C_{self} - C'_{self})/4 \tag{5.4}$$

The division is necessary since C_{over} is defined for each corner, and added to the lumped capacitance for the vicitm corner by corner as shown in Table 5.10. The crossunder correction capacitance C_{under} can be generated similarly.

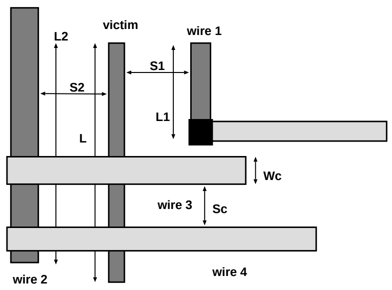


Figure 5.8: An example for 2 1/2-D capacitance analysis.

5.3.2 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 5.8. 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 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 5.10. 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 [18, 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.

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

5.3.3 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 5.11. The errors are 0.54% for the smaller net and 3.33% for the larger one. Moreover, our method 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.

5.4 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.

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

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. Typical extraction and performance verification flows often ignore one or more factors such as process variations, thermal gradients, errors in device and cell characterization, foundry insertion of dummy metal, etc. Since these errors can swamp our several percent error versus 3-D simulation, we conclude that the proposed 2 1/2-D methodology is sufficient for layout optimization. Our ongoing research and development seeks a more holistic integration of synthesis and analysis activities within a framework for constraint-driven design.

$0.50\mu m, C_{i,i}/C_{i,i-2}$				
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 5.1: $C_{i,i}/C_{i,i-2}$, 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 5.2: $C_{i,i}/C_{i,i-2}$ values for the second experiment of Section 2.2.

	$0.50\mu m, C_{i,i}/C_{i,i-1}$		$0.35\mu m, C_{i,i}/C_{i,i-1}$		$0.18\mu m, C_{i,i}/C_{i,i-1}$	
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 5.3: $C_{i,i}/C_{i,i-1}$, where $C_{i,i}$ is the total capacitance of the victim, and $C_{i,i-1}$ the coupling between the victim and the crossunder.

$0.50\mu m$	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
$0.35\mu m$	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
$0.18\mu m$	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 5.4: Total capacitance $C_{i,i}$ of the victim on layer i and couplings between the victim and crossunders on layer $i-1$.

$0.50\mu m$	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

$0.35\mu m$	s	$C_{i,i}$	$C_{i,l1}$	$C_{i,r1}$	$C_{i,u1}$	$C_{i,u2}$	$C_{i,u3}$	$C_{i,u4}$	$C_{i,u5}$	$C_{i,u6}$	$C_{i,o1}$	$C_{i,o2}$	$C_{i,o3}$	$C_{i,o4}$	$C_{i,o5}$	$C_{i,o6}$
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

$0.18\mu m$	s	$C_{i,i}$	$C_{i,l1}$	$C_{i,r1}$	$C_{i,u1}$	$C_{i,u2}$	$C_{i,u3}$	$C_{i,u4}$	$C_{i,u5}$	$C_{i,u6}$	$C_{i,o1}$	$C_{i,o2}$	$C_{i,o3}$	$C_{i,o4}$	$C_{i,o5}$	$C_{i,o6}$
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 5.5: 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 $i+1$.

$0.50\mu m$	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 5.6: 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-∞	∞-∞
	568.4	410.0	360.9	340.7	318.3										
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 5.7: 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 5.8: Total capacitances $C_{i,i}$ for the victim in case of different neighbor widths.

	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 5.9: 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.

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

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

    Endfor

    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

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

    Endfor

    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

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

    Endfor

  Endfor

  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

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

    Endfor

  Endfor

End

Table 5.10: 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 5.11: Comparison between 2 1/2-D analysis and 3-D simulation

CHAPTER 6

Appendix: Proofs for MSWS Properties

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 [41], the full proofs of these theorems are given in [27]. 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.

A. Properties of Coefficient Functions

Careful study of the definitions of f^{ij}, g^{ij} and h^{ij} in (3.4)–(3.6), as well as F, G and H in (3.10)–(3.12) reveals Lemma 1 presented in Section 3.2 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 (3.10)–(3.12), 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.

B. 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*

$$\begin{aligned} F(E_l, E) &= F(E_r, E) \quad if\ E \neq E_l \neq E_r \\ F(E, E_l) &= F(E, E_r) \quad if\ E \neq E_l \neq E_r \end{aligned}$$

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 $F(E, E_l) = F(E, E_r) = F(S', S)$. $\square$

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.*

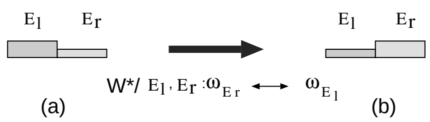


Figure 6.1: (a) The width assignments for E_l and E_r in the optimal solution $\mathcal{W}^*$. (b) The wiresizing solution obtained by swapping the width assignments for E_l and E_r.

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 (3.7) can be written as:

$$
\begin{aligned}
& t(MSIT, \mathcal{E}, \mathcal{W}) \\
= \quad & t(M, \mathcal{E}, \overline{\mathcal{W}}) \; + \; \mathcal{K}_1 \cdot (w_{E_l} \; + \; w_{E_r}) \; + \\
& \mathcal{K}_2 \cdot \sum_{E' \in MSIT} F(E_l, E') \cdot \frac{w_{E'}}{w_{E_l}} \; + \; \mathcal{K}_2 \cdot \sum_{E \in MSIT} F(E, E_l) \cdot \frac{w_{E_l}}{w_E} \; + \\
& \mathcal{K}_2 \cdot \sum_{E' \in MSIT} F(E_r, E') \cdot \frac{w_{E'}}{w_{E_r}} \; + \; \mathcal{K}_2 \cdot \sum_{E \in MSIT} F(E, E_r) \cdot \frac{w_{E_r}}{w_E} \; + \\
& \mathcal{K}_3 \cdot \sum_{E' \in MSIT} F(E_l, E') \cdot \frac{1}{w_{E_l}} \; + \; \mathcal{K}_4 \cdot G(E_l) \cdot \frac{1}{w_{E_l}} \; + \; \mathcal{K}_5 \cdot H(E_l) \cdot \frac{1}{w_{E_l}} \; + \\
& \mathcal{K}_3 \cdot \sum_{E' \in MSIT} F(E_r, E') \cdot \frac{1}{w_{E_r}} \; + \; \mathcal{K}_4 \cdot G(E_r) \cdot \frac{1}{w_{E_r}} \; + \; \mathcal{K}_5 \cdot H(E_r) \cdot \frac{1}{w_{E_r}}
\end{aligned}
\tag{6.1}
$$

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. 6.1), 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

$$G(E_l) = G(E_r) = G(S) \tag{6.2}$$

$$H(E_l) = H(E_r) = H(S) \tag{6.3}$$

$$F(E_l, E) = F(E_r, E) \quad if\ E \neq E_r \neq E_l \tag{6.4}$$

$$F(E, E_l) = F(E, E_r) \quad if\ E \neq E_r \neq E_l \tag{6.5}$$

thus,

$$\begin{aligned}
& t(MSIT, \mathcal{E}, \mathcal{W}^*/E_l, E_r : w_{E_l} \leftrightarrow w_{E_r}) - t(MSIT, \mathcal{E}, \mathcal{W}^*) \\
= \; & \mathcal{K}_2 \cdot F(E_l, E_r) \cdot (\frac{w^*_{E_l}}{w^*_{E_r}} - \frac{w^*_{E_r}}{w^*_{E_l}}) + \mathcal{K}_2 \cdot F(E_r, E_l) \cdot (\frac{w^*_{E_r}}{w^*_{E_l}} - \frac{w^*_{E_l}}{w^*_{E_r}}) + \\
& \mathcal{K}_3 \cdot F(E_l, E_r) \cdot (\frac{1}{w^*_{E_r}} - \frac{1}{w^*_{E_l}}) + \mathcal{K}_3 \cdot F(E_r, E_l) \cdot (\frac{1}{w^*_{E_l}} - \frac{1}{w^*_{E_r}}) \\
= \; & \{F(E_l, E_r) - F(E_r, E_l)\} \cdot \{w^*_{E_l} - w^*_{E_r}\} \cdot \{\mathcal{K}_2 \cdot \frac{(w^*_{E_l} + w^*_{E_r}) + \mathcal{K}_3}{w^*_{E_l} \cdot w^*_{E_r}}\}
\end{aligned} \tag{6.6}$$

We know that $\frac{(w^*_{E_l} + w^*_{E_r}) + \mathcal{K}_3}{w^*_{E_l} \cdot w^*_{E_r}} > 0$ and (6.6) ≥ 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.*

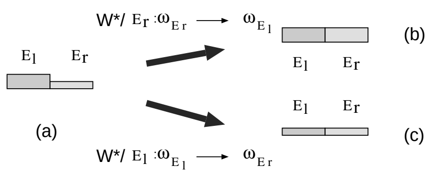


Figure 6.2: (a) The width assignments for E_l and E_r in the optimal solution $\mathcal{W}^*$. (b) The wiresizing solution obtained by replacing the width of E_r with that of E_l. (c) The wiresizing solution obtained by replacing the width of E_l with that of E_r.

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 6.2.b), by using (6.1), is:

$$\begin{aligned}
\Delta t1 &= t(MSIT, \mathcal{E}, \mathcal{W}^*/E_r : w_{E_r} \rightarrow w_{E_l}) \ - \ t(MSIT, \mathcal{E}, \mathcal{W}^*) \\
&= \mathcal{K}_1 \cdot (w_{E_l} - w_{E_r}) \ + \ \mathcal{K}_2 \cdot \sum_{E' \in MSIT} F(E_r, E') \cdot (\frac{w_{E'}}{w_{E_l}} - \frac{w_{E'}}{w_{E_r}}) \ + \\
&\quad \mathcal{K}_2 \cdot \sum_{E \in MSIT} F(E, E_r) \cdot (\frac{w_{E_l}}{w_E} - \frac{w_{E_r}}{w_E}) \ + \ \mathcal{K}_3 \cdot \sum_{E' \in MSIT} F(E_r, E') \cdot (\frac{1}{w_{E_l}} - \frac{1}{w_{E_r}}) \ + \\
&\quad \mathcal{K}_4 \cdot G(E_r) \cdot (\frac{1}{w_{E_l}} - \frac{1}{w_{E_r}}) \ + \ \mathcal{K}_5 \cdot H(E_r) \cdot (\frac{1}{w_{E_l}} - \frac{1}{w_{E_r}}) \\
&\geq 0
\end{aligned} \tag{6.7}$$

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 6.2.c) is:

$$\begin{aligned}
\Delta t2 &= t(MSIT, \mathcal{E}, \mathcal{W}^*/E_l : w_{E_l} \rightarrow w_{E_r}) \ - \ t(MSIT, \mathcal{E}, \mathcal{W}^*) \\
&= \mathcal{K}_1 \cdot (w_{E_r} - w_{E_l}) \ + \ \mathcal{K}_2 \cdot \sum_{E' \in MSIT} F(E_l, E') \cdot (\frac{w_{E'}}{w_{E_r}} - \frac{w_{E'}}{w_{E_l}}) \ + \\
&\quad \mathcal{K}_2 \cdot \sum_{E \in MSIT} F(E, E_l) \cdot (\frac{w_{E_r}}{w_E} - \frac{w_{E_l}}{w_E}) \ + \ \mathcal{K}_3 \cdot \sum_{E' \in MSIT} F(E_l, E') \cdot (\frac{1}{w_{E_r}} - \frac{1}{w_{E_l}}) \ +
\end{aligned}$$

$$
\begin{aligned}
& \mathcal{K}_4 \cdot G(E_l) \cdot (\frac{1}{w_{E_r}} - \frac{1}{w_{E_l}}) \ + \ \mathcal{K}_5 \cdot H(E_l) \cdot (\frac{1}{w_{E_r}} - \frac{1}{w_{E_l}}) \\
\geq \ & 0 \qquad (6.8)
\end{aligned}
$$

Recall (6.2)–(6.5), it is not difficult to verify the following:

$$\Delta t1 \ + \ \Delta t2 \ = \ 0 \qquad (6.9)$$

According to (6.7)–(6.9), $\Delta t1 \ = \ \Delta t2 = 0$. That is, both $\mathcal{W}^*/E_r : w_{E_r} \to w_{E_l}$ and $\mathcal{W}^*/E_l : w_{E_l} \to w_{E_r}$ 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,

B. 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:

$$
\begin{aligned}
\Psi(MSIT, \mathcal{E}, E, \mathcal{W}) \ = \ & \mathcal{K}_1 \cdot \sum_{E' \in MSIT-\{E\}} w_{E'} \ + \ \mathcal{K}_2 \cdot \sum_{E', E'' \in MSIT-\{E\}, E' \neq E''} F(E', E'') \cdot \frac{w_{E''}}{w_{E'}} \ + \\
& \mathcal{K}_3 \cdot \sum_{E', E'' \in MSIT-\{E\}, E' \neq E''} F(E', E'') \cdot \frac{1}{w_{E'}} \ + \\
& \mathcal{K}_4 \cdot \sum_{E' \in MSIT-\{E\}} G(E') \cdot \frac{1}{w_{E'}} \ + \ \mathcal{K}_5 \cdot \sum_{E' \in MSIT-\{E\}} H(E') \cdot \frac{1}{w_{E'}} \qquad (6.10)
\end{aligned}
$$

$$\Phi(MSIT, \mathcal{E}, E, \mathcal{W}) \ = \ \mathcal{K}_1 \ + \ \mathcal{K}_2 \cdot \sum_{E' \in MSIT-\{E\}} F(E', E) \cdot \frac{1}{w_{E'}} \qquad (6.11)$$

$$\Theta(MSIT, \mathcal{E}, E, \mathcal{W}) = \mathcal{K}_2 \cdot \sum_{E' \in MSIT-\{E\}} F(E, E') \cdot w_{E'} + \mathcal{K}_3 \cdot \sum_{E' \in MSIT-\{E\}} F(E, E') + \mathcal{K}_4 \cdot G(E) + \mathcal{K}_5 \cdot H(E) \tag{6.12}$$

Then, we can then rewrite the objective function (3.7) as follows:

$$t(MSIT, \mathcal{E}, \mathcal{W}) = \Psi(MSIT, \mathcal{E}, E, \mathcal{W}) + \Phi(MSIT, \mathcal{E}, E, \mathcal{W}) \cdot w_E + \Theta(MSIT, \mathcal{E}, E, \mathcal{W}) \cdot \frac{1}{w_E} \tag{6.13}$$

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':

[1] In general, under the modeling method used in this work and [41, 91], 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 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',

$$F(E_1, E') = F(E_2, E') = \cdots = F(E, E') = F_r(S)$$
$$F(E', E_1) = F(E', E_2) = \cdots = F(E', E) = F_l(S)$$

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 (6.13), 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. □

C. 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. $\square$

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] H. B. Bakoglu. *Circuits, Interconnections, and Packaging for VLSI*. Addison-Wesley, 1990.

[3] 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.

[4] M. Basel. Accurate and efficient extraction of interconnect circuits for full-chip timing analysis. In *Proc. WESCON*, pages 118–123, 1995.

[5] M. Berkelaar, P. Buurman, and J. Jess. Computing the entire active area/power consumption versus delay trade-off curve for gate sizing with a piecewise linear simulator. In *Proc. Int. Conf. on Computer Aided Design*, pages 474–480, 1994.

[6] M. Berkelaar and J. Jess. Gate sizing in MOS digital circuits with linear programming. In *Proc. European Design Automation Conf.*, pages 217–221, 1990.

[7] 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.

[8] K. D. Boese, A. B. Kahng, and G. Robins. High-performance routing trees with identified critical sinks. In *Proc. Design Automation Conf*, pages 182–187, 1993.

[9] M. Borah, R. M. Owens, and M. J. Irwin. Transistor sizing for minimizing power consumption of CMOS circuit under delay constraint. In *Proc. Int. Symp. on Lower Power Design*, pages 167–172, 1995.

[10] P. K. Chan. Algorithms for libaray-specific sizing of combinational logic. In *Proc. Design Automation Conf*, pages 353–356, 1990.

[11] 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.

[12] 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.

[13] 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.

[14] G. Chen, H. Onodera, and K. Tamaru. An iterative gate sizing approach with accurate delay evaluation. In *Proc. Int. Conf. on Computer Aided Design*, pages 422–427, 1995.

[15] H. Y. Chen and S. M. Kang. iCOACH: A circuit optimization aid for CMOS high-performance circuits. *Integration, the VLSI Journal*, 10:155–168, January 1991.

[16] 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.

[17] 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 $1.2\mu m$ CMOS. In *Proc. IEEE Int. Solid-State Circuits Conf.*, pages 30–31, 1994.

[18] 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.

[19] C. Chu and D. F. Wong. Closed form solution to simultaneous buffer insertion/sizing and wire sizing. In *Proc. Int. Symp. on Physical Design*, pages 192–197, 1997.

[20] 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.

[21] C. Chu and D. F. Wong. Greedy wire-sizing is linear time. In *Proc. Int. Symp. on Physical Design*, pages 39–44, 1998.

[22] W. Chuang and S. S. Sapatnekar. Power vs. delay in gate sizing: Conflicting objectives? In *Proc. Int. Conf. on Computer Aided Design*, pages 463–466, November 1995.

[23] W. Chuang, S. S. Sapatnekar, and I. N. Hajj. Delay and area optimization for discrete gate sizes under double-sided timing constraints. In *Proc. IEEE Custom Integrated Circuits conf.*, pages 9.4.1–9.4.4, 1993.

[24] W. Chuang, S. S. Sapatnekar, and I. N. Hajj. A unified algorithm for gate sizing and clock skew optimization to minimize sequential circuit area. In *Proc. Int. Conf. on Computer Aided Design*, pages 220–223, 1993.

[25] W. Chuang, S. S. Sapatnekar, and I. N. Hajj. Timing and area optimization for standard-cell VLSI circuit design. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 14(3):308–320, March 1995.

[26] M. A. Cirit. Transistor sizing in CMOS circuits. In *Proc. Design Automation Conf*, pages 121–124, 1987.

[27] J. Cong and L. He. Optimal wiresizing for interconnects with multiple sources. Technical Report 950031, UCLA CS Dept, August 1995.

[28] 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.

[29] J. Cong and L. He. Simulatenous transistor and interconnect sizing based on general dominance property. Technical Report 950046, UCLA CS Dept, September 1995.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] Z. Dai and K. Asada. MOSIZ: A two-step transistor sizing algorithm based on optimal timing assignment method for multi-stage complex gates. In *Proc. IEEE Custom Integrated Circuits Conf.*, pages 17.3.1–17.3.4, May 1989.

[46] F. Dartu, N. Menezes, J. Qian, and L. T. Pillage. A gate-delay model for high-speed CMOS circuits. In *Proc. Design Automation Conf*, pages 576–580, 1994.

[47] N. Delorme, M. Belleville, and J. Chilo. Inductance and capacitance analytic formulas for vlsi interconnects. *IEEE Electron Device Lett*, EDL-32:996–997, 1996.

[48] J. G. Ecker. Geometric programming: Methods, computations and applications. *SIAM Review*, 22(3):338–362, July 1980.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] H., Chan. *Private Communication*, 1994.

[54] N. Hedenstierna and K. O. Jeppson. CMOS circuit speed and buffer optimization. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, CAD-6(2):270–281, March 1987.

[55] K. S. Hedlund. AESOP: A tool for automatic transistor sizing. In *Proc. Design Automation Conf*, pages 114–120, 1987.

[56] L. S. Heulser and W. Fichtner. Transistor sizing for large combinational digital CMOS circuits. *Integration, the VLSI Journal*, 10:185–212, January 1991.

[57] U. Hinsberger and R. Kolla. A cell-based approach to performance optimization of fanout-free circuits. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 11(10):1317–1321, October 1992.

[58] T. D. Hodes, B. A. McCoy, and G. Robins. Dynamically-wiresized Elmore-based routing constructions. In *Proc. IEEE Int. Symp. on Circuits and Systems*, pages 463–466, 1994.

[59] B. Hoppe, G. Neuendore, D. Schmitt-Landsiedel, and W. Specks. Optimization of high-speed CMOS logic circuits with analytical models for signal delay, chip area and dynamic power dissipation. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 9(3):237–247, March 1990.

[60] 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.

[61] R. Kay, G. Bucheuv, and L. T. Pileggi. Ewa: Exact wireing-sizing algorithm. In *Proc. Int. Symp. on Physical Design*, pages 178–185, 1997.

[62] W. Li. Strongly NP-hard discrete gate-size problems. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 13(8):1045–1051, August 1994.

[63] W. Li, A. Lim, P. Agrawal, S. Sahni, and R. Kolla. On the circuit implementation problem. In *Proc. Design Automation Conf*, pages 478–483, 1992.

[64] 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.

[65] 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.

[66] J. Lillis, C. K. Cheng, and T. T. Y. Lin. Simultaneous routing and buffer insertion for high performance interconnect. In *Proc. the Sixth Great Lakes Symp. on VLSI*, 1996.

[67] 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.

[68] H. C. Lin and L. W. Linholm. An optimized output stage for MOS integrated circuits. *IEEE Journal of Solid-State Circuits*, SC-10(2):106–109, 1975.

[69] S. Lin, M. Marek-Sadowska, and E. S. Kuh. Delay and area optimization in standard-cell design. In *Proc. Design Automation Conf*, pages 349–352, 1990.

[70] David Luenberger. *Linear and Non-linear Programming*. Addison-Wesley, 1989.

[71] D. P. Marple. Transistor size optimization of digital VLSI circuits. Technical Report CSL-TR-86-308, Stanford Univ., October 1986.

[72] MCNC. *MCNC Designers' Manual*, 1993.

[73] C. Mead and L. Conway. *Introduction to VLSI Systems*. Addison-Wesley, 1993.

[74] 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.

[75] N. Menezes, S. Pullela, F. Dartu, and L. T. Pillage. RC interconnect synthesis — a moment fitting appraoch. In *Proc. Int. Conf. on Computer Aided Design*, pages 418–425, 1994.

[76] N. Menezes, S. Pullela, and L. T. Pileggi. Simultaneous gate and interconnect sizing for circuit-level delay optimization. In *Proc. Design Automation Conf*, pages 690–695, June 1995.

[77] 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.

[78] 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.

[79] 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.

[80] T. Okamoto and J. Cong. Interconnect layout optimization by simultaneous Steiner tree construction and buffer insertion. In *Proc. ACM/SIGDA Physical Design Workshop*, pages 1–6, April 1996.

[81] J. K. Ousterhout. Switch-level delay models for digital MOS VLSI. In *Proc. Design Automation Conf*, pages 542–548, 1984.

[82] L. T. Pillage and R. A. Rohrer. Asymptotic waveform evaluation for timing analysis. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 9(4):352–366, April 1990.

[83] M. J. D. Powell. A fast algorithm for nonlinear constrained optimization calculations. In G. A. Watson, editor, *Lecture Notes in Mathematics, no. 630*, pages 144–157. Springer Verlag, 1978.

[84] 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.

[85] 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.

[86] 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.

[87] T. Sakurai. Closed-form expressions for interconnect delay, coupling, crosstalk in VLSI's. *IEEE Trans. on Electron Devices*, ED-40:118–124, 1993.

[88] T. Sakurai and K. Tamaru. Simple formulas for two- and three- dimensional capacitances. *IEEE Trans. on Electron Devices*, ED-30:183–185, 1983.

[89] 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.

[90] P. K. Sancheti and S. S. Sapatnekar. Interconnect design using convex optimization. In *Proc. IEEE Custom Integrated Circuits Conf.*, pages 549–552, 1994.

[91] S. S. Sapatnekar. RC interconnect optimization under the Elmore delay model. In *Proc. Design Automation Conf*, pages 387–391, 1994.

[92] 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.

[93] S. S. Sapatnekar and V. B. Rao. iDEAS: A delay estimator and transistor sizing tool for CMOS circuits. In *Proc. IEEE Custom Integrated Circuits Conf.*, pages 9.3.1–9.3.4, May 1990.

[94] 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.

[95] H. Sathyamurthy, S. S. Sapatnekar, and J. P. Fishburn. Speeding up pipelined circuits through a combination of gate sizing and clock skew optimization. In *Proc. Int. Conf. on Computer Aided Design*, pages 467–470, 1995.

[96] 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.

[97] Semiconductor Industry Association. *National Technology Roadmap for Semiconductors*, 1994.

[98] J.-M. Shyu, A. L. Sangiovanni-Vincentelli, J. P. Fishburn, and A. E. Dunlop. Optimization based transistor sizing. *IEEE Journal of Solid-State Circuits*, 23(2):400–409, April 1988.

[99] Y. Tamiya, Y. Matsunaga, and M. Fujita. LP based cell selection with constraints of timing, area and power consumption. In *Proc. Int. Conf. on Computer Aided Design*, pages 378–381, November 1994.

[100] 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.

[101] P. M. Vaidya. A new algorithm for minimizing convex functions over convex set. In *Proc. IEEE Symp. on Foundations of Computer Science*, pages 338–343, October 1989.

[102] L. P. P. P. van Ginneken. Buffer placement in distributed RC-tree networks for minimal Elmore delay. In *Proc. IEEE Int. Symp. on Circuits and Systems*, pages 865–868, 1990.

[103] 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.

[104] H. J. M. Veendrick. Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits. *IEEE Journal of Solid-State Circuits*, SC-19(4):468–473, August 1984.

[105] 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.

[106] J. L. Wyatt. *Circuit Analysis, Simulation and Design – Part 2*, chapter 11. North-Holland, 1987.

[107] T. Xue and E. S. Kuh. Post routing performance optimization via multi-link insertion and non-uniform wiresizing. In *Proc. Int. Conf. on Computer Aided Design*, pages 575–580, 1995.

[108] T. Xue and E. S. Kuh. Post routing performance optimization via tapered link insertion and wiresizing. In *Proc. European Design Automation Conf.*, 1995.

[109] 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.

[110] S. Yen and N. Shirali. Capacitance extraction. Technical Report Application Note, Cadence Design Systems, 1995.

[111] Q. Yu and E. S. Kuh. Exact moment matching model of transmission lines and application to interconnect delay estimation. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 3(2):311–322, June 1995.

[112] D. Zhou and X. Y. Liu. On the optimal drivers for high-speed low power ICs. *International Journal of High Speed Electronics and System*, 7(2):287–303, June 1996.