Interconnect Design for Deep Submicron ICs

Jason Cong[*], Zhigang Pan, Lei He, Cheng-Kok Koh and Kei-Yong Khoo
Computer Science Department
University of California, Los Angeles, CA 90095 [†]

Abstract

Interconnect has become the dominating factor in determining circuit performance and reliability in deep submicron designs. In this embedded tutorial, we first discuss the trends and challenges of interconnect design as the technology feature size rapidly decreases towards below 0.1 micron. Then, we present commonly used interconnect models and a set of interconnect design and optimization techniques for improving interconnect performance and reliability. Finally, we present comparisons of different optimization techniques in terms of their efficiency and optimization results, and show the impact of these optimization techniques on interconnect performance in each technology generation from the 0.35μm to 0.07μm projected in the National Technology Roadmap for Semiconductors.

I. Interconnect Trends and Challenges

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) [1], it will further decrease at the rate of 0.7× per generation (consistent with Moore's Law) to reach 0.07 μm by 2010. Table I lists the main characteristics of each technology generation in the NTRS. 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× per generation to reach 800 millions in the 0.07 μm technology. Second, it implies that the circuit performance will be increasingly determined by the interconnect performance. The interconnect design will play the most critical role in achieving the projected clock frequencies in the NTRS. This paper presents the trends and challenges of interconnect design in current and future technologies and discusses the available solutions.

Tech. (μm)	0.35	0.25	0.18	0.13	0.10	0.07
Year	1995	1998	2001	2004	2007	2010
# transistors	12M	28M	64M	150M	350M	800M
$Clock$ (MHz)	300	450	600	800	1000	1100
Area (mm^2)	250	300	360	430	520	620
Wiring levels	4-5	5	5-6	6	6-7	7-8

TABLE I Summary of NTRS [1]

Tech.	0.35	0.25	0.18	0.13	0.10	0.07
Metal 1 Interconnect						
W	**0.40**	**0.30**	**0.22**	**0.15**	**0.11**	**0.08**
S	**0.60**	**0.45**	**0.33**	**0.25**	**0.16**	**0.12**
R	**0.15**	**0.19**	**0.29**	**0.82**	**1.34**	**1.34**
C	**0.17**	**0.19**	**0.21**	**0.24**	**0.27**	**0.27**
AR_M	**1.5:1**	**2:1**	**2.5:1**	**3:1**	**3.5:1**	**4:1**
AR_V	**2.5:1**	**3:1**	**3.5:1**	**4.2:1**	**5.2:1**	**6.2:1**
Metal 4 Interconnect						
W	1.00	0.76	0.56	0.38	0.28	0.20
S	1.50	1.125	0.825	0.625	0.40	0.30
R	0.04	0.050	0.076	0.22	0.35	0.36
Metal 4 with min. spacing and width						
C_a	0.031	0.025	0.021	0.018	0.018	0.017
C_f	0.046	0.042	0.040	0.040	0.042	0.037
C_x	0.056	0.072	0.086	0.090	0.107	0.119
Metal 4 with 2× min. spacing and 2× min. width						
C_a	0.061	0.051	0.044	0.038	0.041	0.035
C_f	0.066	0.060	0.055	0.054	0.054	0.051
C_x	0.020	0.031	0.041	0.045	0.056	0.063

TABLE II Interconnect parameters. W and S are the minimum width and spacing in μm, respectively. R and C are the unit-length resistance and total capacitance in Ω/μm and fF/μm, respectively. AR_M and AR_V are the aspect ratios of the metal and via, respectively. C_a, C_f and C_x are the area, fringe and coupling capacitances per unit length in fF/μm, respectively.

Tech.	0.35	0.25	0.18	0.13	0.10	0.07
C_g	0.225	0.158	0.097	0.050	0.031	0.016
R_d	18.7	22.8	23.8	29.4	31.1	28.3
T_x	0.113	0.084	0.057	0.031	0.020	0.011

TABLE III Device parameters. C_g and R_d are the input capacitance in fF and output resistance in kΩ of an unit-sized gate, respectively. T_x is the intrinsic delay of a gate in ns.

In order to better understand the significance of interconnect design in the future technology generations, we performed a number of experiments based on the interconnect parameters provided in the NTRS as shown in the bold face in Table II. Since the NTRS parameters are for the first metal (M1) layer only, which is usually not suitable for global interconnects, we also derived the interconnect parameters for the M4 layer,[1] which are also shown in Table II. Furthermore, we derived a set of device parameters as shown in Table III based on the data on processes and device in the NTRS. Using these sets of parameters, we carried out extensive simulations using HSPICE to quantitatively measure the interconnect performance and reliability in future technology generations and obtained the following results:

(1) Interconnect delay is clearly the dominating factor in determin-

[*] Email: cong@cs.ucla.edu

[†] This work is partially supported by the NSF Young Investigator Award MIP-9357582 and a grant from Intel under the California MICRO Program.

[1] We assume that the minimum width and spacing of M4 is 2.5 times those of M1. The aspect ratios AR_M and AR_V are used to determine the metal thickness and the dielectric thickness for all layers. For M1, we assume that the substrate and M2 are the ground planes; and for M4, we assume that M3 and M5 are the ground planes. The total capacitance, including the area capacitance, fringing capacitance, and coupling capacitance components, are obtained using the 3D field solver FastCap [2]. Based on these assumptions, our capacitance values for M1 closely match those given in the NTRS.

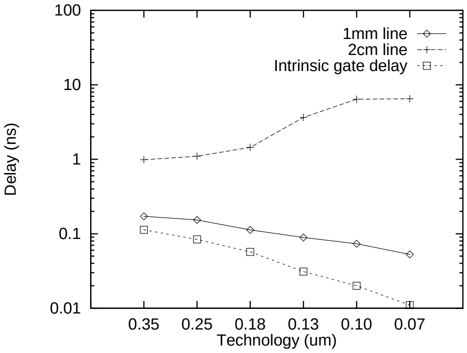

Fig. 1 Global and local interconnect delays versus gate delays.

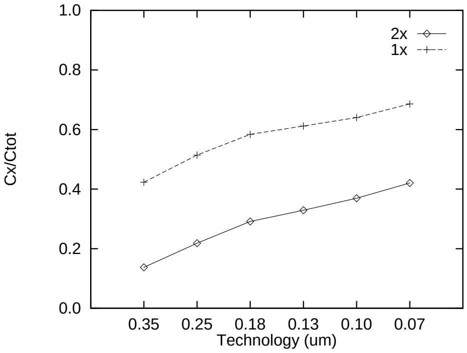

Fig. 2. Ratio of coupling capacitance to total capacitance of M4 interconnect with the minimum width and spacing (1×) and two times the minimum width and spacing (2×).

ing the circuit performance. As shown in Fig. 1, as 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[2]. Clearly, aggressive interconnect optimization is needed in order to achieve the clock frequencies projected in Table I. In Section IV, we shall show how various existing interconnect optimization techniques will limit the growth of interconnect delays.

(2) The coupling capacitance between adjacent lines will be a major component in the total capacitance due to the increase of wire aspect ratio and the decrease of the line spacing. But its value is very sensitive to spacing. As shown in Fig. 2, the ratio of the coupling capacitance to the total capacitance for a wire on M4 with the minimum spacing to its two neighbors increases from around 40% to around 70% when the technology progresses from 0.35 μm to 0.07 μm. When we increase the spacing to two times the minimum, the same ratio becomes from around 15% to around 40% for different technology generations. Therefore, proper spacing is very important in deep submicron interconnect designs.

(3) The coupling noise between adjacent wires will become a important factor in deep submicron designs due to the increase of coupling capacitance. Our experimental results in Fig. 3 shows that if we restrict the peak noise value to be $15\% Vdd$, the maximum allowable length on M4 using the minimum spacing decreases from over 4000 μm to almost 500 μm when the technology progresses from 0.35 μm to 0.07 μm. The same figure also shows the wire length limits under two times the minimum spacing and with $10\% Vdd$ peak noise tolerance.

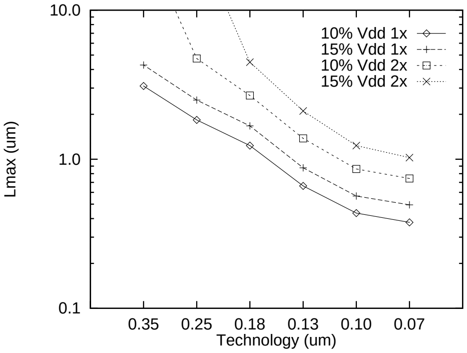

Fig. 3. Maximum allowable length (in log scale) for parallel M4 lines with the minimum width and spacing (1×) and two times the minimum width and spacing (2×) when the peak coupled noise is limited to 10% and 15% of the supply voltage.

Since most existing works have been on interconnect performance optimization, this tutorial covers only the modeling and optimization techniques for interconnect delay minimization. The remainder of this paper is organized as follows: Section II discusses commonly used interconnect and gate delay models for layout optimization. Sections III presents the techniques for interconnect layout design and optimization. Section IV compares a number of interconnect optimization techniques in terms of their efficiency and solution quality and shows their impact on interconnect delay reduction in each technology generation projected in the NTRS. Due to the page limitation, the authors are able to present only a small subset of results on the topics covered in this paper. A more comprehensive survey and bibliography is available in [3].

II. Delay Modeling

A. Interconnect Modeling

In order to consider both wire resistance and capacitance and model the distributive nature of the interconnects, a routing tree is usually modeled as an RC tree by dividing each long wire into a sequence of wire segments and modeling each wire segment as an L-type or π-type of RC circuit. The number of R, C elements can be large when the length of each segment is chosen to be small for a better approximation of the distributed nature of the interconnects or a greater degree of flexibility in wiresizing optimization. Therefore, a reduced-order RC model is often computed to approximate the large RC tree using the moment matching technique.

Let $h(t)$ be the impulse response at a node of a RC tree. The transfer function $H(s)$ of the circuit, which is the Laplace transform of $h(t)$, can be represented as

$$H(s) = \int_0^\infty h(t)e^{-st}dt = \sum_{i=0}^{\infty} \frac{(-1)^i}{i!} s^i \int_0^\infty t^i h(t)dt. \quad (1)$$

The i-moment of the transfer function m_i is defined to be the unsigned coefficient of the i-th power of s in Eqn. (1)

$$m_i = \frac{1}{i!}\int_0^\infty t^i h(t)dt. \quad (2)$$

Moments of an RC tree can be computed efficiently using recursive methods (see [3] for details).

The first moment $m_1 = \int_0^\infty t \cdot h(t)dt$, also called the *Elmore delay model* [4], is most commonly used for delay estimation in an RC tree. In essence, the Elmore delay model uses the mean of the impulse

[2] Both sets of interconnect delays are based on the assumption of the minimum wire width and two times minimum spacing on M4 with optimal driver sizing.

response $h(t)$ to approximate the 50% delay of the step response (under the step input), which corresponds to the median of the impulse response. It was shown that the Elmore delay from source s_0 to node i in an RC tree can be computed by the following simple equation [5]:

$$t(s_0, i) = \sum_{k \in Path(s_0, i)} R_k \cdot Cap(k), \tag{3}$$

where $Path(s_0, i)$ is the unique path from source s_0 to node i in an RC tree, R_k is the resistance at node k, and $Cap(k)$ is the total capacitance of the subtree rooted at node k. In general, the Elmore delay of a sink in an RC tree gives an upper bound on the actual 50% delay of the sink under the step input [6].

The Elmore delay allows us to explicitly express the signal delay as a simple algebraic function of the geometric parameters of the interconnect (the lengths and widths of wires), so that it can be easily used for interconnect optimization. It was shown that the Elmore delay model offers reasonably good *fidelity* for interconnect layout optimization, i.e., an optimal or near-optimal solution obtained under the Elmore delay model is also close to optimal according to actual (SPICE-computed) delays (see [3] for details). But the absolute value of Elmore delay may not be very accurate. So, it is not suitable to be used directly for accurate circuit timing analysis.

Higher order moments can be used for more accurate reduced-order RC models. The *Asymptotic Waveform Evaluation* (AWE) method [7] based on Padé approximation uses higher order moments to constructs a q-pole transfer function $\hat{H}(s)$, called the reduced-order *q-pole model*,

$$\hat{H}(s) = \sum_{i=1}^{q} \frac{k_i}{s - p_i}, \tag{4}$$

to approximate the actual transfer function $H(s)$, where p_i are poles and k_i are residues, all of which can be determined uniquely by matching the initial boundary conditions and the first $2q - 1$ moments of $H(s)$ to those of $\hat{H}(s)$ [7]. The response waveform in the time domain under the step input is given by

$$\hat{h}(t) = \sum_{i=1}^{q} k_i e^{p_i t}. \tag{5}$$

The choice of order q depends on the accuracy required but is usually much less than the order of the circuit. In practice, $q \leq 5$ is often used. It is difficult, however, to represent the poles and residues in $\hat{H}(s)$ explicitly in terms of design parameters of the interconnect in a closed-form expression, which makes the moment-matching method difficult to use for interconnect optimization directly[3]. Some delay metrics based on higher order moments, such as the central moments and the explicit RC delay using the first three moments, are summarized in [3]. Note that except for the Elmore delay model, which is defined for a monotonic response only, the techniques presented above still holds when interconnects are modeled as RLC trees.

Recent progresses on reduced-order models include the use of the PVL (Padé Via Lanczos) method for Padé approximation without direct moment computation [8, 9], the congruence transformations to create reduced RC networks which are guaranteed to be stable and passive [10], and the coordinate-transformed Arnoldi algorithm that can be applied to general RLC network [11]. The objective of these algorithms is to overcome the numerical instability of the AWE method.

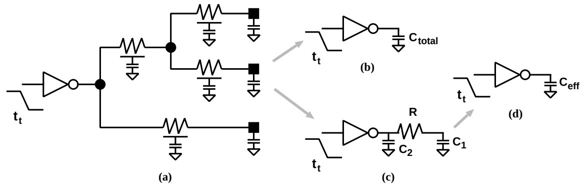

Fig. 4. (a) An inverter driving an RC interconnect. (b) The same inverter driving the total capacitance of the net in (a). (c) A π-model of the driving point admittance for the net in (a). (d) The same inverter driving the effective capacitance of the net in (a). The input signal has a transition time of t_t.

B. Driver Modeling

In this subsection, we collectively refer to gates, buffers, or transistors as drivers. We present two commonly used approaches to model the drivers for delay computation with interconnects. The first approach is a switch-resistor model comprised of an effective linear resistor driven by a voltage source (usually assuming a step input or ramped input). The effective resistance of a driver usually depends on the transition time of the input signal, the loading capacitance, and the size of the driver. For example, one can use a resistor of fixed value R_{eff} to model a driver by selecting an appropriate capacitance load C and matching the 50% delay of the driver driving the load with that of the equivalent RC circuit ($0.7 R_{eff} C$) under the step-input. A more accurate model, called the *slope model*, uses a one-dimensional table to compute the effective driver resistance based on the concept of rise-time ratio [12]. It first uses the output load and transistor size to compute the *intrinsic rise-time* of the driver, which is the rise-time at the output under the step input. The input rise-time of the driver is then divided by the intrinsic rise-time of the driver to produce the *rise-time ratio* of the driver. The effective resistance is represented as a piece-wise linear function of the rise-time ratio and stored in a one-dimensional table. Given a driver, one first computes its rise-time ratio and then calculates its effective resistance R_{eff} by interpolation according to its rise-time ratio from the one-dimensional table. Multi-dimensional tables can also be used for computing and storing the effective driver resistance as a function of the input slope, output load, etc. The switch-resistor model has the advantage that the coupling with the interconnect can be easily modeled by including the effective driver resistance in the interconnect RC tree for delay and/or waveform computation. But it may be difficult to model the non-linear behavior of the driver.

The second approach for driver modeling characterizes the behavior of a driver (such as the driver delay and the output transition time) using all relevant parameters of the input signal(s) and the output load. This allows for very accurate modeling, but the gate delay and the interconnect delay must be computed separately. For example, one can pre-characterize, the delay (t_d) and output transition times (t_f and t_r) of a driver in terms of the input transition time t_t and the total load capacitance C_L using accurate circuit simulation such as SPICE. The characterized results can then be stored in a *look-up table* where each entry is in the form: $\{t_t, C_L, (t_d, t_f, t_r)\}$. Such a model can be very accurate if one can afford the time and space to generate a detailed multi-dimensional table for each gate. Alternatively, one can store the characterization data much more compactly in the form of *k-factor equations* [13, 14], such as:

$$\begin{aligned} t_d &= (k_1 + k_2 \cdot C_L) \cdot t_t + k_3 \cdot C_L^3 + k_4 \cdot C_L + k_5 & (6) \\ t_f &= (k_1' + k_2' \cdot C_L) \cdot t_t + k_3' \cdot C_L^2 + k_4' \cdot C_L + k_5' & (7) \end{aligned}$$

where $k_{1 \cdots 5}$ and $k'_{1 \cdots 5}$ are determined based on linear regression or least square fits on the characterization data.

[3] Sensitivity-based methods have been proposed to use higher order moments for fast timing analysis to greedily guide the optimization process to a local optima.

C. Delay Computation

In general, we are interested to compute the total delay from the input of a driver to one of the sinks (an input to a gate in the next stage) in its output net, called the *stage delay*. When the interconnect is modeled as a lump capacitance (Fig. 4(b)) with no interconnect resistance, the computation of the stage delay is straightforward. Using the switched-resistor driver model, the stage delay is simply $R_d \cdot (C_L + C_I)$ (for a step voltage source) where C_L and C_I are the load capacitance and interconnect capacitance, respectively. Using a pre-characterized driver model, the stage delay can be obtained by table look-up and interpolation or computed from the k-factor equations directly.

When a distributed RC interconnect model is used in junction with a switch-resistor driver model, the stage delay can be easily computed by first constructing a new RC network that combines the interconnect model with the driver's effective resistance and then compute the delay through an RC network using the methods discussed in Section II.A. This shows the advantage of the switch-resistor driver model where the interaction between the driver and the interconnect can be easily modeled.

When a distributed RC interconnect model is used in junction with a pre-characterized driver model, the driver delay and the interconnect delay need to be computed separately and added up together to obtain the stage delay. Moreover, the interaction between the driver and the interconnect model should be considered during the driver pre-characterization. Since a distributed RC interconnect has many parameters, the information usually need to be ”compressed” for driver pre-characterization. For example, the π*-model* [15] was proposed to approximate the driving point (i.e, the output of the driver) admittance as shown in Fig. 4(c). The values of C_1, C_2 and R in a π-model (see Fig. 4(c)) can be computed by

$$C_1 = y_2^2/y_3, \;\; C_2 = y_1 - (y_2^2/y_3), \;\; R = -(y_3^2/y_2^3). \qquad (8)$$

where y_1, y_2 and y_3 are the first three moments of the driving point admittance, which can be computed recursively in a bottom-up fashion, starting from the sinks of the interconnect tree. In this case, the driver can be characterized using C_1, C_2 and R in addition to the input transition time, etc. for driver delay computation.

Since a very large look-up table or complex k-factor equations and very extensive simulations are needed to account for all possible combinations of C_1, C_2 and R in a π-model, the *effective capacitance model* [14] was proposed to allow drivers to be still pre-characterized in terms of a single load capacitance, even when used to drive distributed RC interconnects. The effective capacitance model first computes a π-model to approximate the driving point admittance, and then compute iteratively an “effective capacitance,” denoted C_{eff} as in Fig. 4(d), using the following expression:

$$\begin{aligned} C_{eff} \quad = \quad & C_2 + C_1 \cdot \Bigg[1 - \frac{R \cdot C_1}{t_D - t_x/2} + \\ & \frac{(R \cdot C_1)^2}{t_x(t_D - t_x/2)} \cdot e^{\frac{-(t_D - t_x)}{R \cdot C_1}} \cdot (1 - e^{\frac{-t_x}{R \cdot C_1}}) \Bigg] \qquad (9) \end{aligned}$$

where $t_D = t_d + t_t/2$ and $t_x = t_D - t_f/2$, and t_d and t_f are obtained from the k-factor equations in terms of the effective capacitance and the input transition t_t. The iteration starts with using the total interconnect and sink capacitance as the loading capacitance C_L to get an estimate of t_D and t_x through the k-factor equations. A new value of the effective capacitance is computed using Eqn. (9) and it is used as the loading capacitance for the next iteration of computation. The process stops when the value of C_{eff} does not change in two successive iterations. At the end of the iterative process, we also obtain t_d and t_f at the gate output. The effective capacitance, which is smaller than C_{total} in Fig. 4(b), captures the fact that not all the capacitance of the routing tree and the sinks is seen by the driver due to the effect of interconnect *resistance shielding*, especially in deep submicron design with fast logic gates of lower driver resistance. A so-called resistance model (*R-model*) was also proposed in [14] to better approximate the slow decaying tail portion of the response waveform when the driver is behaving like a resistance to ground. The model can be used to further account for the interaction between the RC interconnect and the driver when computing the interconnect delay [16]. These methods illustrate the complication of the interaction between the driver model and the interconnect model in the deep submicron design.

III. Interconnect Layout Optimization

Given the growing importance of interconnects, interconnect optimization needs to be considered in every step of the layout design process. We propose a performance-driven layout design flow as shown in Fig. 5, in which planning and optimization for global interconnects are carried out during the floorplan stage and further interconnect optimization is performed during global routing. In this section, we discuss various optimization techniques that can be applied in this flow for interconnect delay minimization, including wirelength minimization, device sizing, interconnect topology optimization, buffer insertion, optimal wiresizing, and simultaneous device and interconnect optimization.

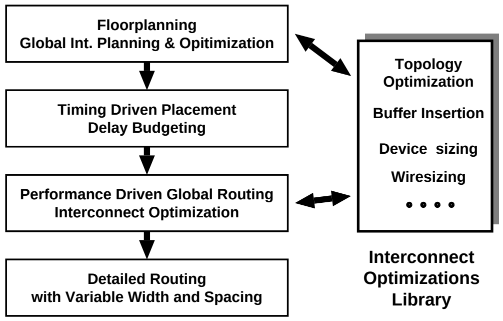
Floorplanning
Global Int. Planning & Opitimization
Timing Driven Placement
Delay Budgeting
Performance Driven Global Routing
Interconnect Optimization
Detailed Routing
with Variable Width and Spacing
Topology Optimization
Buffer Insertion
Device sizing
Wiresizing
• • • •
Interconnect Optimizations Library

Fig. 5 Layout design flow for deep submicron ICs.

A. Wirelength Minimization

A very effective way to reduce the interconnect delay is to minimize the wirelength of timing-critical nets, so that their total capacitances are reduced. Placement has the biggest impact on the wirelength. Timing-driven placement methods can be classified into the *net-based approaches* and *path-based approaches*. For net-based approaches, a delay budgeting algorithm is first applied on the netlist to compute the timing slack for each net (or two-terminal subnet) (e.g. [17]). These slacks are then translated into wirelength upper bound constraints (e.g. [18]) or the net weights in the optimization objective function used by the placement engine. Path-based approaches usually use mathematical programming techniques and consider the path-based timing constraints directly in the problem formulation (e.g. [19]). In both cases, the estimated wirelengths of the timing critical nets (often measured in terms of the half perimeter of the net bounding box) are minimized during the placement, possibly at the expense of the wirelengths of non-timing critical nets.

Wirelength minimization can also be carried out during global routing by constructing an optimal (or near-optimal) Steiner tree (OST) for each timing-critical net. The commonly used methods include iterative addition of Steiner points, optimal merging of edges of a minimum spanning tree (MST), or iterative refinement of an MST. These methods are surveyed in [3]. However, when the interconnect resistance needs to be considered as well, wirelength minimization alone during global routing may not lead to the minimum interconnect delay. Interconnect topology optimization needs to be considered.

B. Interconnect Topology Optimization

It was shown in [20] that when the *resistance ratio*, defined to be the driver effective resistance over the unit wire resistance, is small enough, both the total wirelength (i.e. the total interconnect capacitance) and interconnect topology will impact the interconnect delay. The first step in interconnect topology optimization is to minimize or control the path-lengths from the driver to the timing-critical sinks to reduce the interconnect RC delays. A number of algorithms have been developed to minimize both the path-lengths and the total wire-length in a routing tree. For example, the *bounded-radius bounded-cost* (BRBC) algorithm [21] bounds the radius (i.e. the maximum path-length between the driver and a sink) in the routing tree while minimizing its total wire-length. It first constructs an MST, then eliminates the long paths by adding 'short-cuts' into the MST and computing a shortest path tree of the resulting graph. Other algorithms in this class include the AHHK tree construction and the 'performance oriented spanning tree' construction, which are discussed in [22] and [3]. In particular, it was shown in [20] that a minimal length shortest path tree in the Manhattan plane (called the *A-tree*) can be constructed very efficiently using a bottom-up merging heuristic with sizable delay reduction yet only a small wire-length overhead compared to the OST. The A-tree construction method has been extended to signal nets with multiple drivers (as in signal busses) [23].

Further optimization of interconnect topology involves using more accurate delay models during routing tree topology construction. For example, the Elmore delay model was used in [24] and the 2-pole delay model was used in [25] to evaluate which node or edge to be added to the routing tree during iterative tree construction. Other methods, such as the alphabetical tree and P-tree construction are also summarized in [3].

C. Device Sizing

When we have a good estimate of the interconnect capacitive load of a net, the size of its driving gate can be optimized for delay minimization. For a heavy capacitive load, a chain of cascaded drivers is usually used. The *driver sizing* problem is to determine both the number of driver stages and the size for each driver. Using the simple switch-resistor RC model and ignoring the capacitance of the driver output and the wire connecting to consecutive drivers, one can show that if the loading capacitance is C_L and the stage number is N, the ratio of two consecutive drivers (called the stage ratio) should be a constant $(\frac{C_L}{C_0})^{1/N}$ in order to achieve the minimum delay. When N is not fixed, the optimal stage ratio $f = e$ and the stage number is $N = ln(\frac{C_L}{C_g})$. When the more accurate driver delay model is used with consideration of the driver input transition time and output capacitance, the result in [26] shows that 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. For the technology used in [26], α is about 1.35 and the optimal stage ratio is in the range of 3–5 instead of e.

In general, *transistor sizing* can be used to determine the optimal width for each transistor to optimize the overall circuit performance. This technique is often used in cell generation and full-custom layout. It is usually assumed that the transistor can be assigned a continuous width. The early work TILOS [27] used the simple switch-resistor model for transistors, formulated the transistor sizing problem as a posynomial program, and applied a greedy sensitivity based method. The sensitivity of a transistor is defined to be the delay reduction due to a unit increment of its size. The algorithm starts with a minimum-sized solution, and timing analysis is applied. The transistor with the largest sensitivity is increased by a user defined factor and then timing analysis is applied again. This procedure terminates when the timing specification is satisfied or all sensitivities are zero or negative. Recent advances in transistor sizing include the use of more accurate transistor delay model with consideration of the input waveform slope, and the use of linear programming, convex programming, or other non-linear programming techniques for computing a global optimal solution. Similar techniques have also been used for *discrete gate sizing* (also called *cell sizing*) in ASIC designs, which assumes that each gate has a discrete set of pre-designed implementations (cells) from a given cell library. The gate sizing algorithm chooses an appropriate cell for each gate for performance optimization. These techniques are summarized in [3].

D. Buffer Insertion

Buffer insertion (also called *repeater insertion*) is another common and effective technique to use active device areas to trade for reduction of interconnect delays. Since the Elmore delay of a long wire grows quadratically in terms of the wirelength, buffer insertion can reduce interconnect delay significantly.

A polynomial-time dynamic programming algorithm was presented in [28] 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, and maximizes the required arrival time at the source. The algorithm includes both bottom-up synthesis of possible buffer assignment solutions at each node and top-down selection of the optimal solution. In the bottom-up synthesis procedure, for each legal position i for buffer insertion, a set of possible buffer assignments, called *options*, in the subtree T_i rooted at i is computed. For a node k which is the parent of two subtrees T_i and T_j, the list of options for T_k is generated from the option lists of T_i and T_j based on a merging rule and a pruning rule, so that the number of options for T_k is no more than the sum of the numbers of options for T_i and T_j plus the number of possible buffer assignments in the edge coming to k. As a result, if the total number of legal positions is N and there is one type of buffer, the total number of options at the root of the entire 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 which maximizes the required arrival time at the source is selected. Then, a top-down back-tracing procedure is carried out to select the buffer assignment solution that led to the optimal option at the source.

E. Wiresizing Optimization

It was first shown in [20, 29] that when wire resistance becomes significant, as in the deep submicron design, proper wire-sizing can effectively reduce the interconnect delay. Assuming each wire has a set of discrete wire widths, their work presented an optimal wire-sizing algorithm for a single-source RC interconnect tree to minimize the sum of weighted delays from the source to timing-critical sinks under the Elmore delay model. They showed that an optimal wiresizing solution satisfies the monotone property, the separability, and the dominance property. Based on the dominance property, the lower (or upper) bounds of the optimal wire widths can be computed efficiently by iterative local refinement, starting from a minimum-width solution (or maximum-width solution for computing upper bounds). Each local refinement operation refines the width of an edge in the routing tree assuming all other edge widths are fixed. The lower and upper bounds usually meet, which leads to an optimal wiresizing solution. Otherwise, a dynamic programming based method is used to compute the optimal solution within the lower and upper bounds. This method is very efficient, capable of handling large interconnect structures, and leads to substantial delay reduction. It has been extended to optimize the routing trees with multiple drivers, routing trees without a priori segmentation of long wires, and to meet the target delays using Lagrangian relaxation. The reader may refer to [3] for more details.

An alternative approach to wiresizing optimization computes an optimal wiresizing solution using bottom-up merging and top-down selection [30] in a very similar way as the buffer insertion algorithm presented in the preceding subsection. At each node v, a set of irredundant wiresizing solutions of the subtree rooted at v is generated by merging and pruning the irredundant wiresizing solutions of the subtrees rooted at the children nodes of v. Eventually, a set of irredundant wiresizing solutions is formed at the driver for the entire routing tree, and an optimal wiresizing solution is chosen by a top-down selection process. The approach has the advantages that the optimization is targeted at meeting the required signal arrival times at sinks directly, and it can be easily extended to be combined with routing tree construction and buffer insertion as shown in the next section.

Further studies on wiresizing optimization include using more accurate delay models, such as higher-order RC delay models [31] and lossy transmission line models [32], and understanding the optimal wire shape under the assumption that non-uniform continuous wiresizing is allowed to each wire segment [33]. These results are discussed in more details in [3]. All these algorithms, however, optimize the wire widths of a single net and ignore the coupling capacitance between adjacent nets, which can be significant in deep submicron designs. Recently, an efficient algorithm named GISS (global interconnect sizing and spacing) was developed to optimize the widths and spacings for multiple nets simultaneously with consideration coupling capacitance for delay minimization [34]. It reported substantial further delay reduction compared to the single net wire sizing algorithms.

F. Simultaneous Device and Interconnect Optimization

The most effective approach to performance optimization is to consider the interaction between devices and interconnects, and optimize both of them at the same time. Two approaches are discussed in this subsection.

F.1. Simultaneous Device and Wire Sizing

The simultaneous driver and wire sizing (SDWS) problem was studied in [35] and later generalized to simultaneous buffer and wire sizing (SBWS) in a buffered routing tree [36]. In both cases, the switch-resistor model is used for the driver and the Elmore delay model is used for the interconnects modeled as RC trees. The objective function is to minimize the sum of weighted delays from the first stage of the cascaded drivers through the buffered routing tree to timing-critical sinks. It was shown that the dominance property still holds for SDWS and SBWS problems and the local refinement operation, as used for wiresizing, can be used iteratively to compute tight lower and upper bounds of the optimal widths of the driver, buffers, and wires efficiently, which often leads to an optimal solution. Dynamic programming or bounded enumeration can be used to compute the optimal solution within the lower and upper bounds when they do not meet. This approach has been shown to be very effective for optimizing very large buffered trees, yielding substantial reduction on both delay and power dissipation compared to manual designs.

In fact, it was recently shown in [37] that the dominance property holds for a large class of objective functions called *general CH-posynomials*. Based on this general result, the work in [37] is able to perform simultaneous transistor and wire sizing efficiently given a general netlist (not limited to buffered trees). A significant advantage of the CH-posynomial formulation is that it can handle more accurate transistor models, including both simple analytical models or more accurate table-lookup based models obtained from detailed simulation to consider the effect of the waveform slope, which leads to better optimization results. Other studies on simultaneous device and wire sizing include using higher order RC delay models for the interconnect by either matching to the target moments or using a q-pole transfer function for sensitivity analysis. The reader may refer to [3] for more details.

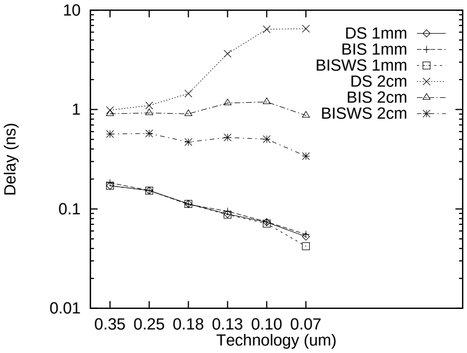

Fig. 6. Delays of 1 mm and 2 cm M4 lines under driver sizing only (DS), buffer insertion and sizing (BIS) and buffer insertion and sizing and wiresizing (BISWS).

F.2. Simultaneous Topology Construction with Buffer and Wire Sizing

The *wiresized buffered A-tree (WBA-tree)* algorithm was proposed [38] for simultaneous routing tree topology construction, buffer insertion and wiresizing. It naturally combines the A-tree construction algorithm [20] and the simultaneous buffer insertion and wiresizing algorithm [30], as both use bottom-up construction techniques. The WBA algorithm includes a bottom-up synthesis procedure and a top-down selection procedure. During the bottom-up synthesis procedure, it selects two subtrees for merging with consideration of both minimization of wirelength and maximization of the estimated arrival time at the source. As a result, it is able to achieve both *critical path isolation* and a *balanced load decomposition*, as often used for fanout optimization in logic synthesis. The WBA algorithm has been extended recently to explore multiple interconnect topologies at each subtree and use high-order RLC delay models based on efficient incremental moment computation in partially constructed routing trees [39].

Other methods have also been proposed for simultaneous topology construction and wire sizing, including a greedy dynamic wire sizing during iterative routing tree construction and use of link insertion with dynamic wire sizing to create non-tree topologies. These algorithms are summarized in [3].

IV. Optimization Results and Comparative Studies

A. Impact of Interconnect Optimization on Future Technology Generations

We applied three interconnect optimization techniques for interconnect delay minimization of a 2 cm global interconnect and a 1 mm local interconnect for each technology generation in NTRS. The three optimization algorithms include (i) optimal driver sizing (DS), (ii) optimal buffer insertion and sizing (BIS), and (iii) optimal buffer insertion, sizing and wiresizing (BISWS). The delays of the optimized interconnect structures in each technology generation are shown in Fig. 6, and detailed description of the optimization results by BISWS

Tech.		0.35	0.25	0.18	0.13	0.10	0.07
2 cm line	t_d (ns)	0.57	0.57	0.47	0.53	0.50	0.34
	$\#B$	2	3	3	6	9	11
	ABS	565	992	1370	2237	3094	4535
	AWS (μm)	6.00	5.83	6.03	6.10	5.72	5.25
	$\%WS$	91.4	96.2	97.6	99.3	99.6	99.8
1 mm line	t_d (ns)	0.17	0.15	0.11	0.09	0.07	0.04
	$\#B$	1	1	1	1	1	2
	ABS	71	91	111	161	231	711
	AWS (μm)	1.00	0.76	0.61	0.68	0.73	1.45
	$\%WS$	0	0	17	72	84	96

TABLE IV Results of Buffer Insertion and Sizing and Wiresizing. $\#B$ is the number of buffers inserted. ABS is the average buffer size normalized to minimum feature size. AWS is the average wire size. $\%WS$ is the percentage of wire segments with sizing larger than minimum width.

are shown in Table IV. We have several observations from this set of results.

1. The impact of buffer insertion and buffer/wire sizing for local interconnects is minimal after proper driver sizing, even for the technologies below 0.1 μm.
2. Buffer insertion/sizing and wire sizing have very significant impact for global interconnects, especially as the technology progresses to very deep submicron designs. In the 0.07 μm technology, BIS reduces the interconnect delay by almost a factor of 10. When wiresizing is allowed, BISWS further reduces the interconnect delay by 40% to 50%.
3. Interconnect design will be highly complex in deep submicron technologies. For example, the optimization result of the 2 cm global interconnect by BISWS contains 11 buffers with 99.8% wires being sized above the minimum width. Clearly, a global interconnect is no longer a simple metal line. It becomes a complex circuitry with optimized devices and wires in deep submicron designs! Considering the fact that there will be over 800 million transistors and 7-8 routing layers, with an estimated total wire length over 10 kilometers per chip in the 0.07 μm technology, we need highly efficient and scalable layout systems to support the various interconnect optimization techniques discussed in this paper.
4. Although the best interconnect optimization technique (BISWS) is able to reduce the global interconnect delay by up to 20× compared with the un-optimized designs in the same technology generation, if we compare the delays of best optimized global interconnects in different technology generations, it only decreases slightly by about 40% from 0.35 μm to 0.07 μm. This clearly indicates that such optimization alone will not achieve over $3\times$ performance increase from the 0.35 μm to 0.07 μm technologies as expected in Table I. Therefore, innovations in system architectures, interconnect architectures, and interconnect technologies are needed to achieve the predicted performance targets in NTRS.

B. Comparisons of Various Interconnect Optimization Algorithms

In this subsection, we provide a comparative study of a number of interconnect optimization algorithms presented in Section III in terms of their efficiency and optimality, so that one can make proper choices for his or her optimization needs in practice. We use the interconnect optimization package developed in our group at UCLA in the past five years, named TRIO (Tree, Repeater, and Interconnect Optimization) for this set of experiments. The TRIO package includes many interconnect optimization algorithms presented in Section III and also offers the capability to combine them in different ways to provide a wide spectrum of interconnect optimization solutions. In particular, we shall compare the following four optimization strategies:

- **T+B+W**: A-tree construction (Section III.F.2), followed by optimal buffer insertion and sizing (Section III.F.1) with B=10 buffer sizes, then followed by optimal wiresizing using bundled local refinement [40] based on the dominance property (Section III.E) with W=18 wire widths.
- **TB+SBWS**: simultaneous topology and buffer optimization (Section III.F.2) with B=3 followed by simultaneous buffer and wiresizing (Section III.F.1) with B=40 and W=18.
- **Tbw+SBWS**: simultaneous topology, buffer insertion and sizing, and wiresize optimization (Section III.F.2) with very limited choices of buffer sizes and wire widths (B=3 and W=3), followed by simultaneous buffer and wire sizing (SBWS in Section III.F.1) with B=40 and W=18.
- **TBW**: simultaneous topology construction, buffer insertion and sizing, and wiresize optimization (Section III.F.2) with B=10 and W=8.

These algorithms are applied to three sets of randomly generated multi-terminal nets of 5, 10 and 20 pins, respectively, with pins uniformly distributed within a 10 mm by 10 mm area. Each set contains three instances. The optimization results are shown in Table V based on the 0.18 μm technology. We have several observations:

1. Simultaneous device and interconnect optimization by TBW usually produces the better results compared to other separate optimizations, with up to 20% additional delay reduction compared to T+B+W.
2. The bottom-up dynamic programming technique used in TBW can be very timing consuming (even run in polynomial time) with large number of choices of buffer sizes and wire widths (up to 6 minutes on the average for 20-pin nets).
3. For buffer or/and wire sizing, local refinement based optimization (SBWS) using the dominance property is much more efficient than the bottom-up dynamic programming technique used in TBW. SBWS can handle a large number of buffer sizes and wire widths in a fraction of a second. Therefore, proper combination of TBW and SBWS provides a good trade-off of efficiency and optimality. Our results show that Tbw+SBWS has less than 1% difference compared to TBW in terms of solution quality, but runs more than 10 times faster. Therefore, Tbw+SBWS is our recommended solution for most interconnect optimization applications.

The UCLA TRIO package also includes a number of other interconnect optimization routines, such simultaneous transistor and wire sizing (STIS), global interconnect sizing and spacing (GISS), etc. whose results are not able to be included here due to the space limitation. The TRIO package can accommodate a number of layout constraints, such as the upper and lower bounds of each wire segments, allowed buffer locations, etc. It also interfaces with a 2.5D capacitance extractor and can produce the optimization results directly into the HSPICE netlist format for detailed timing simulation. All the delay results reported in this paper are obtained by HSPICE simulations.

V. CONCLUSIONS

In this tutorial, we have shown the trends and challenges of interconnect design as the technology feature size decreases to below 0.1 μm based on the data in NTRS. We presented a set of commonly

		Algorithms			
		T+B+W	TB+SBWS	Tbw+SBWS	TBW
5 pins	t_d (ns)	0.40	0.39	0.35	0.34
		0.47	0.48	0.38	0.38
		0.42	0.41	0.36	0.35
	CPU (s)	0.1	0.1	1.4	15
10 pins	t_d (ns)	0.42	0.37	0.34	0.33
		0.56	0.56	0.44	0.44
		0.47	0.45	0.38	0.38
	CPU (s)	0.8	1.0	6.4	76
20 pins	t_d (ns)	0.45	0.43	0.38	0.39
		0.54	0.48	0.42	0.41
		0.46	0.43	0.38	0.38
	CPU (s)	1.6	4.0	27.6	350

TABLE V Comparison of Algorithms. t_d is the average delay time for each net (each row is one net) and CPU is the average running time on a Sun Ultra2 workstation with 256 Mbytes of memory.

used interconnect and driver models and presented a set of interconnect design and optimization techniques which have proven to be very effective for improving interconnect performance and reliability. Our experimental results show that these optimization techniques have a very significant impact on the performance of the global interconnects, with different degree of efficiency and optimality.

The research on interconnect modeling and optimization have been focused mainly on interconnect delay minimization in the past several years. Given the growing importance of coupling noise as discussed in Section 1 and other concerns on signal reliability, we expect to see much more research on modeling and optimization on signal reliability of interconnects in the near future.

References

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

[2] 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*, vol. 10, pp. 1447–1459, Nov. 1991.

[3] J. Cong, L. He, C.-K. Koh, and P. H. Madden, "Performance optimization of VLSI interconnect layout," *Integration, the VLSI Journal*, vol. 21, pp. 1–94, 1996.

[4] W. C. Elmore, "The transient response of damped linear networks with particular regard to wide-band amplifiers," *Journal of Applied Physics*, vol. 19, pp. 55–63, Jan. 1948.

[5] 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*, vol. CAD-2, pp. 202–211, July 1983.

[6] R. Gupta, B. Tutuianu, B. Krauter, and L. T. Pillage, "The Elmore delay as a bound for RC trees with generalized input signals," in *Proc. Design Automation Conf*, pp. 364–369, June 1995.

[7] L. T. Pillage and R. A. Rohrer, "Asymptotic waveform evaluation for timing analysis," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, pp. 352–366, Apr. 1990.

[8] P. Feldmann and R. W. Freund, "Efficient linear circuit analysis by Padé approximation via the Lanczos process," in *Proc. European Design Automation Conf.*, 1994.

[9] P. Feldmann and R. W. Freund, "Reduced-order modeling of large linear subcircuits via a block Lanczos algorithm," in *Proc. Design Automation Conf*, pp. 474–479, 1995.

[10] K. J. Kerns, I. L. Wemple, and A. T. Yang, "Stable and efficient reduction of substrate model networks using congruence transforms," in *Proc. Int. Conf. on Computer Aided Design*, pp. 207–214, 1995.

[11] L. M. Silveira, M. Kamon, I. Elfadel, and J. White, "A coordinate-transformed Arnoldi algorithm for generating guaranteed stable reduced-order models for RLC circuits," in *Proc. Int. Conf. on Computer Aided Design*, pp. 288–294, 1996.

[12] J. K. Ousterhout, "Switch-level delay models for digital MOS VLSI," in *Proc. Design Automation Conf*, pp. 542–548, 1984.

[13] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: a Systems Perspective.* Addison-Wesley, second ed., 1993.

[14] 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*, vol. 13, pp. 1526–1535, Dec. 1994.

[15] P. R. O'Brien and T. L. Savarino, "Modeling the driving-point characteristic of resistive interconnect for accurate delay estimation," in *Proc. Design Automation Conf*, pp. 512–515, Nov. 1989.

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

[17] P. S. Hauge, R. Nair, and E. J. Yoffa, "Circuit placement for predictable performance," in *Proc. Int. Conf. on Computer Aided Design*, pp. 88–91, 1987.

[18] W. Swartz and C. Sechen, "Timing driven placement for large standard cell circuits," in *Proc. Design Automation Conf*, pp. 211–215, 1995.

[19] A. Srinivasan, K. Chaudhary, and E. S. Kuh, "RITUAL: Performance driven placement algorithm for small cell ics," in *Proc. Int. Conf. on Computer Aided Design*, pp. 48–51, 1991.

[20] J. Cong, K. S. Leung, and D. Zhou, "Performance-driven interconnect design based on distributed RC delay model," in *Proc. Design Automation Conf*, pp. 606–611, 1993.

[21] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong, "Provably good performance-driven global routing," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, pp. 739–752, June 1992.

[22] A. B. Kahng and G. Robins, *On Optimal Interconnections for VLSI.* Kluwer Academic Publishers, 1994.

[23] J. Cong and P. H. Madden, "Performance driven routing with multiple sources," in *Proc. IEEE Int. Symp. on Circuits and Systems*, pp. 1.203–1.206, 1995.

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

[25] D. Zhou, F. Tsui, and D. S. Gao, "High performance multichip interconnection design," in *Proc. 4th ACM/SIGDA Physical Design Workshop*, pp. 32–43, Apr. 1993.

[26] N. Hedenstierna and K. O. Jeppson, "CMOS circuit speed and buffer optimization," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. CAD-6, pp. 270–281, Mar. 1987.

[27] J. P. Fishburn and A. E. Dunlop, "TILOS: A posynomial programming approach to transistor sizing," in *Proc. Int. Conf. on Computer Aided Design*, pp. 326–328, 1985.

[28] 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*, pp. 865–868, 1990.

[29] J. Cong and K. S. Leung, "Optimal wiresizing under the distributed Elmore delay model," in *Proc. Int. Conf. on Computer Aided Design*, pp. 634–639, 1993.

[30] 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*, pp. 138–143, Nov. 1995.

[31] 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*, pp. 418–425, 1994.

[32] 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.*, pp. 117–121, 1996.

[33] C.-P. Chen, H. Zhou, and D. F. Wong, "Optimal non-uniform wire-sizing under the Elmore delay model," in *Proc. Int. Conf. on Computer Aided Design*, pp. 38–43, 1996.

[34] 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*, 1997.

[35] J. Cong and C.-K. Koh, "Simultaneous driver and wire sizing for performance and power optimization," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 2, pp. 408–423, Dec. 1994.

[36] 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*, pp. 271–276, Aug. 1996.

[37] J. Cong and L. He, "An efficient approach to simultaneous transistor and interconnect sizing," in *Proc. Int. Conf. on Computer Aided Design*, pp. 181–186, Nov. 1996.

[38] T. Okamoto and J. Cong, "Buffered Steiner tree construction with wire sizing for interconnect layout optimization," in *Proc. Int. Conf. on Computer Aided Design*, pp. 44–49, Nov. 1996.

[39] J. Cong and C.-K. Koh, "Interconnect layout optimization under higher-order RLC model," in *Proc. Int. Conf. on Computer Aided Design*, 1997.

[40] J. Cong and L. He, "Optimal wiresizing for interconnects with multiple sources," *ACM Trans. on Design Automation of Electronics Systems*, vol. 1, pp. 478–511, Oct. 1996.