Modeling and Layout Optimization of VLSI Devices and Interconnects In Deep Submicron Design*

Jason Cong
Department of Computer Science
University of California, Los Angeles, CA 90095
cong@cs.ucla.edu

Abstract — This paper presents an overview of recent advances on modeling and layout optimization of devices and interconnects for high-performance VLSI circuit design under the deep submicron technology. First, we review a number of interconnect and driver/gate delay models, which are most useful to guide the layout optimization. Then, we summarize the available performance optimization techniques for VLSI device and interconnect layout, including driver and transistor sizing, transistor ordering, interconnect topology optimization, optimal wire sizing, optimal buffer placement, and simultaneous topology construction, buffer insertion, buffer and wire sizing. The efficiency and impact of these techniques will be discussed in the tutorial.

I. Introduction

The driving force behind the rapid growth of the VLSI technology has been the constant reduction of the feature size of VLSI devices. The feature size decreased from about $2\mu m$ in 1985 to 0.35-$0.5\mu m$ today (1996). Such continual miniaturization of VLSI devices has strong impact on the VLSI technology in several ways. First, the device density increases rapidly – the total number of transistors on a single VLSI chip has increased from less than 500,000 in 1985 to over 10 million today. Second, the interconnect delay becomes much more significant. According to the simple scaling rule described in [1], when the devices and interconnects are scaled down in all three dimensions by a factor of S, the intrinsic gate delay is reduced by a factor of S, the delay of local interconnects (such as connections between adjacent gates) remains the same, but the delay of global interconnects increases by a factor of S^2. As a result, the interconnect delay has become the dominating factor in determining system performance. In many systems designed today, as much as 50% to 70% of clock cycle are consumed by interconnect delays. This percentage will continue to rise as the feature size decreases further.

Not only do interconnects become more important, they also become much more difficult to model and optimize in the deep submicron VLSI technology, as the *distributed nature* of the interconnects has to be considered. For the conventional technology with the feature size of $1\mu m$ or above, the interconnect resistance in most cases is negligible compared to the driver resistance. In this case, the interconnect delay is determined by the driver resistance times the total interconnect and loading capacitance. Therefore, conventional optimization techniques focus on reducing the driver resistance using driver, gate, and transistor sizing, and minimizing the interconnect capacitance by minimum-length, minimum-width routing. In the deep submicron design technology, however, the interconnect resistance is comparable to the driver resistance in many long signal nets. Therefore, the interconnect has to be modeled as a distributed RC or RLC circuit. Techniques such as optimal wire sizing, optimal buffer placement, and simultaneous driver, buffer, and wire sizing have become necessary and important.

This paper presents an overview of the available techniques for device and interconnect layout for performance optimization in deep submicron design. Section II discusses interconnect and gate delay models used for layout optimization. Sections III and IV present the techniques for device and interconnect layout optimization, respectively. Section V presents recent advances on simultaneous device and interconnect layout optimization. Due to the page limitation, the author has to selectively present only a subset of results on the topics covered in this paper. A much more comprehensive survey and a complete bibliography will soon be available as an Integration Report [6].

II. Device and Interconnect Delay Models

A. Interconnect Delay Models

As VLSI design reaches deep submicron technology, the delay model used to estimate interconnect delay in interconnect design has evolved from the simplistic lumped RC model to sophisticated high order moment matching delay models.

In the *lumped RC model*, "R" refers to the resistance of the driver and "C" refers to the sum of the total interconnect capacitance and the total loading gate capacitance. This model assumes that wire resistance is negligible, which is generally true for designs with feature sizes of $1.2\mu m$ and above since the driver resistance is substantially larger than the total wire resistance for most on-chip wires. Assuming a step input, the delay for a gate to switch to 50% of its final value can be estimated by $0.7RC$ [1], which is used to estimate the delay from the input transition of a gate to the input transition of each of its loading gates.

However, as the feature size decreases to the submicron dimension, the wire resistance is no longer negligible. In order to consider both wire resistance and capacitance, the interconnect is usually modeled as an RC tree. The *Elmore delay model* [14] is the most commonly used for delay estimation in an RC tree. Under this delay model, the signal delay from source s_0 to node i in an RC tree is given by:

$$t(s_0, i) = \sum_{k \in Path(s_0, i)} R_k \cdot Cap(k), \quad (1)$$

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 essence, the Elmore delay model uses the mean of the impulse response $h(t)$, which can be easily represented by $\int_0^\infty t \cdot h(t)dt$, to approximate the 50% delay of the step response (under the step input), which corresponds to the median of the

*This work is supported by NSF Young Investigator Award MIP9357582.

ASP-DAC '97
0-89791-851-7$5.00

impulse response. 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 [17].

The main advantage of the Elmore delay is that it provides a simple closed-form expression, with much improved accuracy for delay measure compared to the lumped RC model, which allows us to express the signal delay as a simple algebraic function of the geometric parameters of the interconnect (the lengths and widths of wires) and parasitic constants (such as the sheet resistance and unit area and fringing capacitances of the interconnect). Many recent interconnect layout optimization algorithms first represent the routing tree as an RC tree by modeling each wire segment as an L-type or π-type of RC circuit, and then use the Elmore delay as the objective function to optimize interconnect layout parameters. It was shown that the Elmore delay model offers a high degree of *fidelity* for interconnect layout optimization, i.e., an optimal or near-optimal solution obtained under the Elmore delay model is also nearly optimal according to actual (SPICE-computed) delays (see [6] for details).

The Elmore delay model suffers a few disadvantages. First, 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. Also, it cannot handle the inductive effect as the Elmore delay is defined for a monotonic response. The Elmore delay is in fact the first moment of the interconnect under the impulse response when it is modeled as an RC tree. More accurate delay estimation can be obtained by modeling the interconnect as a RLC tree and using the higher orders of the moments. Let $h(t)$ be the impulse response at a node of an interconnect, which is modeled as an RC, RLC, or distributed-RLC circuit. The transfer function $H(s)$ of the circuit, which is the Laplace transform of $h(t)$, can be represented as

$$H(s) = \int_0^\infty h(t)e^{-st}dt = \sum_{i=0}^{\infty} \frac{(-1)^i}{i!} s^i \int_0^\infty t^i h(t)dt. \qquad (2)$$

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. (2)

$$m_i = \frac{1}{i!} \int_0^\infty t^i h(t)dt. \qquad (3)$$

The Elmore delay model is the first moment $m_1 = \int_0^\infty t \cdot h(t)dt$ of the impulse response $h(t)$. Higher order moments of an RLC tree can be computed efficiently using the recently proposed recursive methods (see [6] for details).

Higher order moments can be used to achieve more accurate delay estimation. The *Asymptotic Waveform Evaluation* (AWE) method [27] uses higher order moments to constructs a q-pole transfer function $\hat{H}(s)$, called the *q-pole model*,

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

to approximate the actual transfer function $H(s)$, where p_i are poles and k_i are residues to be determined. The corresponding time domain impulse response is

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

The poles and residues in $\hat{H}(s)$ can be determined uniquely by matching the initial boundary conditions, denoted m_{-1}, and the first $2q-1$ moments m_i of $H(s)$ to those of $\hat{H}(s)$ [27]. The choice of order q depends on the accuracy required but is always much less than the order of the circuit. In practice, $q \leq 5$ is commonly used. In general, however, it is not possible 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 AWE method difficult to use for interconnect optimization directly.[1] Therefore, the researchers have recently focused on the simple case of $q = 2$, known as the *two-pole model*, to approximate the delay explicitly using the first three moments m_0 (which is normalized), m_1, and m_2. The reader may refer to [6] for more detailed discussions. Note that the two-pole model can capture the basic inductive effect. The expressions used in the two-pole model are usually much more complex than the Elmore delay model. However, it is possible to use the two-pole model for interconnect optimization with similar degree of efficiency but higher accuracy than the Elmore delay model. Other 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 [6].

B. Device Delay Models

Given an input signal, we are interested in modeling the response waveform of a gate, buffer, or transistor at its output. In this subsection, we collectively refer to gates, buffers, or transistors as drivers. Although the delay models presented here calculate the fall time of the output signal of a driver, defined as the time for the output to fall from 90% to 10% of its steady-state value, and the delay time for the falling signal, defined as the time from 50% input transition to 50% output transition, these models can be used to compute the rise time and rise time delay in a similar way.

We first use a transistor to illustrate the *simple switch-level RC model*, where a transistor is modeled as an effective resistor discharging or charging a capacitor. We normalize the transistor size such that a minimum-size transistor has the unit size. For an n-transistor of size $d \geq 1$, assuming a step input, the fall time of the signal at the gate output is given by [30]:

$$t_f = k \cdot \frac{C_L}{\beta^n_{min} \cdot d \cdot V_{DD}}, \qquad (6)$$

where k is typically in the range of 3 to 4 for values of V_{DD} in the range of 3 to 5, β^n_{min} is the gain factor for the minimum n-transistor, and C_L is the loading capacitance driven by the transistor. The delay time for the falling signal can be approximated to be $t_{df} = t_f/2$ [30]. Since the effective resistance R_d is proportional to $1/\beta_{min} \cdot d$, t_f or t_{df} is proportional to $R_d \cdot C_L$. Similar discussion can be applied to a p-transistor. The simplicity of this model makes it easy to use for device layout optimization, especially for device sizing. A serious limitation of this model, however, is that it does not consider the shape of the input waveform, while in fact that the transition rate of the input signal has a significant impact on the driver delay. The delay models presented in the remaining of this section overcome this limitation with various degree of accuracy and efficiency.

An analytical expression with consideration of the input waveform slope was proposed in [18] for the delay time of a falling signal:

$$t_{df} = t_f/2 + \frac{t_t}{6} \cdot (1 + 2\frac{V^n_{th}}{V_{DD}}), \qquad (7)$$

where t_t is the input transition time (more specifically, the input rise time in this case) and V^n_{th} is the threshold voltage of n-transistor. Since this model provides a closed-form expression, it has also been widely used for device layout optimization.

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

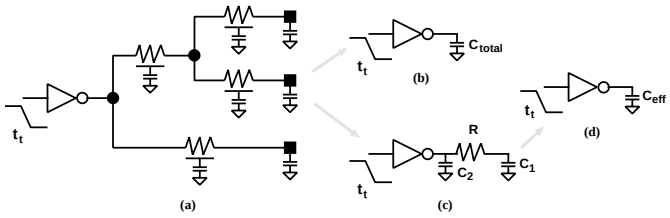

Fig. 1. (a) An inverter driving an RC interconnect. (b) The same inverter driving the total capacitance of the net in (a). (c) A π-model of the driving point admittance for the net in (a). (d) The same inverter driving the effective capacitance of the net in (a). The input signal has a transition time of t_t.

The *slope model* uses a one-dimensional table to compute the effective driver resistance based on the concept of rise-time ratio [26]. The effective resistance of a driver depends on the transition time of the input signal, the loading capacitance, and the size of the driver. In this model, the output load and transistor size are first combined into a single value called 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_d by interpolation according to its rise-time ratio from the one-dimensional table. The driver rise-time delay is computed by multiplying the effective resistance with the total capacitance.

Another commonly used driver delay model pre-characterizes the driver delay of each type of drivers in terms of the input transition time t_t, and the total load capacitance C_L in the form of *k-factor equations* [30, 29], such as:

$$t_{df} = (k_1 + k_2 \cdot C_L) \cdot t_t + k_3 \cdot C_L^3 + k_4 \cdot C_L + k_5, \quad (8)$$

$$t_f = (k'_1 + k'_2 \cdot C_L) \cdot t_t + k'_3 \cdot C_L^2 + k'_4 \cdot C_L + k'_5, \quad (9)$$

where $k_{1\cdots5}$ and $k'_{1\cdots5}$ are determined based on accurate circuit simulation (e.g. using SPICE) and linear regression or least square fits.

In general, a *look-up table* can be used to characterize the delay of each type of gate. A typical entry in the table can be of the following form: $\{t_t, C_L, (t_{df}, t_f)\}$. Given input transition time t_t and output loading capacitance C_L, the look-up table for a specific gate provides the delay and rise/fall time (t_{df}, t_f). The table look-up approach can be very accurate, if one can afford the time and space to generate a detailed multi-dimensional table for each gate.

All these driver delay models use the loading capacitance for delay computation. As the first order approximation, the loading capacitance can be simply computed as the total capacitance of the interconnect and the sinks (Fig. 1(a) and (b)). However, 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. The *effective capacitance model* was proposed to first use a π-*model* [24] (Fig. 1(c)) to better approximate the driving point admittance at the root of the interconnect (or equivalently, the output of the driver), and then compute iteratively the "effective capacitance" seen by the driver, denoted C_{eff}, using the *k*-factor equations.

The π-model of an interconnect is constructed using the first three moments y_1, y_2 and y_3 of the driving point admittance. The three moments of the driving point admittance are computed recursively in a bottom-up fashion, starting from the leaf nodes of the interconnect. The values of C_1, C_2 and R in a π-model (see Fig. 1(c)) can be computed as follows:

$$C_1 = y_2^2/y_3, \;\; C_2 = y_1 - (y_2^2/y_3), \;\; R = -(y_3^2/y_2^3). \quad (10)$$

After obtaining a π-model for the interconnect, the *"effective capacitance"* can be computed iteratively from R, C_1 and C_2 in the π-model (Fig. 1(c) and (d)) using the following expression:

$$C_{eff} = C_2 + C_1 \cdot \left[1 - \frac{R \cdot C_1}{t_D - t_x/2} + \frac{(R \cdot C_1)^2}{t_x(t_D - t_x/2)} \cdot e^{\frac{-(t_D - t_x)}{R C_1}} \cdot (1 - e^{\frac{-t_x}{R C_1}})\right], \quad (11)$$

where $t_D = t_{df} + t_t/2$ and $t_x = t_D - t_f/2$, and t_{df} and t_f can both be 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. (11) 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. A so-called resistance model (*R-model*) was also proposed in [29] to better approximate the slow decaying tail portion of the response waveform. This method illustrates the complication of the interaction between the drive model and the interconnect model in the deep submicron design.

III. Device Layout Optimization

In this section, we discuss the optimization techniques for device layout, including driver sizing, transistor and gate sizing, and transistor ordering.

A. 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. Using the simple switch-level RC model in Eqn. (6) 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 optimal stage ratio at each stage 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 in Eqn. (7) is used with consideration of the driver output capacitance, the result in [18] 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 [18], α is about 1.35 and the optimal stage ratio is in the range of 3–5 instead of e. In a recent work on driver sizing for both performance and power optimization [32], 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. Proper choice of increasing stage ratios can reduce power dissipation considerably with no or little loss on performance.

B. Transistor and Gate Sizing

The *transistor sizing* problem is 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 width may change continuously. The

early work TILOS [15] used the simple switch-level 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. These results are summarized in [6].

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. 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. As a result, more accurate delay models and global optimization techniques are often used for continuous gate sizing, as reviewed in [6]. This formulation is useful for parameterized cell generation.

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. This optimization is often performed as part of the technology mapping procedure in logic synthesis. But in deep submicron design it needs to be applied in connection with layout design so that a good estimation of the interconnect load is available. The discrete gate sizing problem has been shown to be NP-hard. Optimal solutions are only applicable to special circuits (such as trees and series-parallel circuits) based on dynamic programming. Heuristic methods have developed for general circuits based on sensitivity analysis and/or mathematical programming. The reader may refer to [6] for more details.

C. Transistor Ordering

The *transistor ordering* problem is to find the best ordering of (series-connected) transistors in each gate to minimize the delay and/or the power. This technique is useful as the signal arrival times at a set of equivalent input transistors of a gate may be different, and proper ordering can reduce the signal delay at the gate output. Both exhaustive methods and heuristic methods were proposed to search for an optimal ordering. When applied to the entire circuit, it needs to be combined with circuit timing analysis so that when a gate is optimized, the signal arrival times at its inputs are known. The reader may refer to [3] and [28] for more details. Transistor ordering has no (or little) area penalty, but the reduction on delay is limited (usually around 5%).

IV. Interconnect Layout Optimization

In this subsection, we summarize recent research results on interconnect layout optimization, including interconnect topology optimization and optimal wiresizing.

A. Interconnect Topology Optimization

When the interconnect resistance is negligible as for most nets in conventional design technology with large feature sizes, interconnect topology optimization focuses only on minimizing the total wire-length of the routing tree, as it minimizes the total interconnect capacitance which reduces the signal delay directly. Wire-length minimization is achieved by constructing an optimal (or near-optimal) Steiner tree (OST). 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 [6].

When the interconnect resistance needs to be considered, the first step is to minimize or control the path-lengths from the driver to timing-critical sinks to reduce the interconnect RC delay. A class 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 [7] 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 [19] and [6]. An extreme case of this class of algorithms is to construct a shortest path tree with the minimum wire-length. It was shown in [11], however, using a bottom-up merging heuristic, a minimal length shortest path tree in the Manhattan plane (also called the *A-tree*) can be constructed very efficiently 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) [12].

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 [2] and the 2-pole delay model was used in [33] to evaluate which node or edge to be added to the routing tree during iterative tree construction. Other methods, including the alphabetical tree and P-tree construction, have also been proposed. They are summarized in [6].

B. Wiresizing Optimization

It was first shown in [10, 11] 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 [6] for more details.

An alternative approach to wiresizing optimization computes an optimal wiresizing solution using bottom-up merging and top-down selection [20]. 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 [22] and lossy transmission line models [31], and understanding the optimal wire shape under the assumption that non-uniform continuous wiresizing is allowed to each wire segment [4]. These results are discussed in more details in [6].

V. Simultaneous Device and Interconnect Optimization

We feel that the most promising approach to performance optimization is to consider the interaction between devices and interconnects, and optimize both of them at the same time. This section discusses the recent advances in this area.

A. Simultaneous Device and Wire Sizing

The simultaneous driver and wire sizing (SDWS) problem was first studied in [8] and later generalized to simultaneous buffer and wire sizing (SBWS) in a buffered routing tree [9]. In both cases, the switch-level 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 [5] that the dominance property holds for a large class of objective functions called *general CH-posynomials*, which are defined as follows. A function $f(\mathbf{X})$ is a general CH-posynomial if it is of the following form:

$$\begin{aligned} f(\mathbf{X}) \quad &= \quad \sum_{p=0}^{m}\sum_{q=0}^{m}\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\frac{a_{pi}(x_i)}{x_i^p}\cdot b_{qj}(x_j)\cdot x_j^q \\ \textit{where} \quad &\quad a_{pi}(x_i)\geq 0 \textit{ and } b_{qj}(x_j)\geq 0, \\ &\quad \mathbf{0}<\mathbf{L}\leq\mathbf{X}\leq\mathbf{U}, \end{aligned} \tag{12}$$

and the coefficient functions satisfy the following conditions: (i) $a_{pi}(x_i)$ is a function of x_i. It monotonically increases with respect to an increase of x_i, but $\frac{a_{pi}(x_i)}{x_i^p}$ still monotonically decreases with respect to an increase of x_i. (ii) $b_{qj}(x_j)$ is a function of x_j. It monotonically decreases with respect to an increase of x_j, but $b_{qj}(x_j)\cdot x_j^q$ still monotonically increases with respect to an increase of x_i. When the coefficient functions in Eqn. (12) are constants, the function $f(\mathbf{X})$ is called a *simple CH-posynomial*. The class of simple CH-posynomials is a subset of posynomial functions defined in [13]. The optimization problem of minimizing a simple/general CH-posynomial is called a *simple/general CH-posynomial program*. It was shown in [5] that the dominance property holds for both simple and general CH-posynomial programs, and the local refinement operation can be applied to compute the lower and upper bounds of an optimal solution efficiently. Based on this general result, the work in [5] 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 [6] for more details.

B. Buffer Insertion

Buffer (also called repeater) insertion is a common and effective technique to use active device area to trade for reduction of 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.

A polynomial-time dynamic programming algorithm was presented in [16] 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.

C. Simultaneous Topology Construction with Buffer and Wire Sizing

Recently, the *wiresized buffered A-tree (WBA-tree)* algorithm was proposed [25] for simultaneous routing tree topology construction, buffer insertion and wiresizing. It naturally combines the A-tree construction algorithm [11] and the simultaneous buffer insertion and wiresizing algorithm, as both use bottom-up construction techniques. Similar to the buffer insertion algorithm presented in the previous section, the WBA algorithm includes a bottom-up synthesis procedure and a top-down selection procedure. However, 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 algo-

rithm enables us to study the interaction between topology optimization, buffer insertion, and wire sizing, and leads to many interesting observations. For example, we observed that the delay reduction due to wire sizing decreases as more buffers are inserted. Also, it is possible to construct a buffered routing tree first (with proper choice of buffer size), and then perform simultaneous buffer and wire sizing to produce a final routing solution with comparable signal delay but much shorter computation time as compared to an all integrated approach.

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 [6].

VI. Concluding Remarks

This paper has summarized the delay modeling techniques for VLSI devices and interconnects which have been useful to guide performance optimization in VLSI layout under the deep submicron technology. It has also classified and summarized various layout optimization techniques for delay minimization, with emphasis on recent advances on interconnect layout optimization, and simultaneous device and interconnect optimization. These modeling and optimization techniques are very important in developing new generation of timing-driven automatic layout systems for designing very large-scale ICs under the deep submicron technology. Although the commercial CAD systems for ASIC design today do not have most of the features and capabilities presented in this paper, we expect that they will be adopted soon by the turn of this century.

This paper has focused mainly on delay minimization for general signal nets. It did not address the layout optimization issues for special nets, such as skew minimization for clock nets and minimization of noise and voltage drop for power nets. A comprehensive survey of research results on clock skew optimization was presented in [6].

Another challenge for layout design under the deep submicron technology is modeling and minimization of crosstalk noise, which is becoming increasingly important because of reduced line-to-line spacing and change of aspect-ratio of metal lines. We see only a limited amount of research in this area, and did not include it in this tutorial. A list of related publications were cited in [6]. We believe that this is an important research area which will impact the design of future layout systems.

References

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

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

[3] B. S. Carlson and C.Y. Chen, "Performance Enhancement of CMOS VLSI Circuits by Transistor Reordering," *Proc. ACM/IEEE Design Automation Conf.*, 1993, pp. 361-366.

[4] C.-P. Chen, H. Zhou, and D. F. Wong "Optimal Non-Uniform Wire-Sizing under the Elmore Delay Model," *Proc. Int'l Conf. on Computer-Aided Design*, Nov. 1996, pp. 38-43.

[5] J. Cong and L. He, "An Efficient Approach to Simultaneous Transistor and Interconnect Sizing," *Proc. Int'l Conf. on Computer-Aided Design*, Nov. 1996, pp. 181–186.

[6] J. Cong, L. He, C.-K. Koh, and P. H. Madden, "Performance Optimization of VLSI Interconnect Layout," *Integration, the VLSI Journal*, Vol. 21, Nos. 1&2, November 1996, pp. 1–94.

[7] 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*, 11(6), June 1992, pp. 739-752.

[8] 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*, 2(4), December 1994, pp. 408-423.

[9] J. Cong, C.-K. Koh, and K.-S. Leung, "Simultaneous Buffer and Wire Sizing for Performance and Power Optimization," *Proc. Int'l Symp. on Low Power Electronics and Design*, August 1996, pp. 271–276.

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

[11] J. Cong, K. S. Leung, and D. Zhou, "Performance-Driven Interconnect Design Based on Distributed RC Delay Model," *Proc. ACM/IEEE Design Automation Conf.*, 1993, pp. 606-611.

[12] J. Cong and P. H. Madden, "Performance Driven Routing with Multiple Sources," *Proc. Int'l Symp. on Circuits and Systems*, 1995, pp. 1157–1169.

[13] J. G. Ecker, "Geometric Programming: Methods, Computations and Applications," *SIAM Review*, vol. 22, No. 3, July 1980, pp. 338-362.

[14] W. C. Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wide-Band Amplifiers," *Journal of Applied Physics*, vol. 19, no. 1, Jan. 1948, pp. 55–63.

[15] J. P. Fishburn and A. E. Dunlop, "TILOS: A Posynomial Programming Approach to Transistor Sizing," *Proc. Int'l. Conf. on Computer-Aided Design*, 1985, pp. 326-328.

[16] L. P. P. P. van Ginneken, "Buffer Placement in Distributed RC-tree Networks for Minimal Elmore Delay," *Proc. Int'l Symp. on Circuits and Systems*, 1990, pp. 865-868.

[17] R. Gupta, B. Tutuianu, B. Krauter, and L. T. Pillage, "The Elmore Delay as a Bound for RC Trees with Generalized Input Signals," *Proc. 32nd ACM/IEEE Design Automation Conf.*, June 1995, pp. 364–369.

[18] N. Hedenstierna and K. O. Jeppson, "CMOS Circuit Speed and Buffer Optimization," *IEEE Trans. on Computer-Aided Design*, 1987, pp. 270-281.

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

[20] J. Lillis, C. K. Cheng and T. T. Y. Lin, "Optimal Wire Sizing and Buffer Insertion for Low Power and a Generalized Delay Model," *Proc. IEEE Int'l. Conf. on Computer-Aided Design*, Nov. 1995, pp. 138-143.

[21] N. Menezes, R. Baldick, and L. T. Pileggi, "A Sequential Quadratic Programming Approach to Concurrent Gate and Wire Sizing," *Proc. Int'l Conf. on Computer-Aided Design*, 1995, pp. 144-151.

[22] N. Menezes, S. Pullela, F. Dartu and L. T. Pillage, "RC Interconnect Synthesis —- A Moment Fitting Approach," *Proc. Int'l Conf. on Computer-Aided Design*, 1994, pp. 418-425.

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

[24] P. R. O'Brien and T. L. Savarino, "Modeling the Driving-Point Characteristic of Resistive Interconnect for Accurate Delay Estimation," *Proc. Int'l Conf. on Computer-Aided Design*, Nov. 1989, pp. 512–515.

[25] T. Okamoto and J. Cong, "Buffered Steiner Tree Construction with Wire Sizing for Interconnect Layout Optimization," *Proc. Int'l Conf. on Computer-Aided Design*, Nov. 1996, pp. 44–49.

[26] J. K. Ousterhout, "Switch-Level Delay Models for Digital MOS VLSI," *Proc. 21st Design Automation Conf.*, 1984, pp. 542–548.

[27] L. T. Pillage and R. A. Rohrer, "Asymptotic Waveform Evaluation for Timing Analysis," *IEEE Trans. on Computer-Aided Design*, 9(4), Apr. 1990, pp. 352–366.

[28] S. C. Prasad and K. Roy, "Circuit Optimization for Minimization of Power Consumption under Delay Constraint," *Proc. Int'l workshop on Low Power Design*, April, 1994, pp. 15-20.

[29] 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), Dec. 1994, pp. 1526–1535.

[30] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: a Systems Perspective – 2nd ed*, Addison-Wesley, 1993.

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

[32] D. Zhou, and X. Y. Liu, "On the Optimal Drivers for High-Speed Low Power ICs," to appear in *International Journal of High Speed Electronics and System*, 1996.

[33] D. Zhou, F. Tsui, and D. S. Gao, "High Performance Multichip Interconnection Design," *Proc. 4th ACM/SIGDA Physical Design Workshop*, Apr. 1993, pp. 32–43.