Fast Performance-Driven Optimization for Buffered Clock Trees Based on Lagrangian Relaxation *

Chung-Ping Chen, Yao-Wen Chang, and D. F. Wong
Department of Computer Sciences, University of Texas, Austin, Texas 78712

Abstract

Delay, power, skew, area, and sensitivity are the most important concerns in current clock-tree design. We present in this paper an algorithm for simultaneously optimizing the above objectives by sizing wires and buffers in clock trees. Our algorithm, based on Lagrangian relaxation method, can optimally minimize delay, power, and area simultaneously with very low skew and sensitivity. With linear storage overall *and linear runtime* per iteration, *our algorithm is extremely economical, fast, and accurate; for example, our algorithm can solve a 6201-wire-segment clock-tree problem using about 1-minute runtime and 1.3-MB memory and still achieve pico-second precision on an IBM RS/6000 workstation.*

1 Introduction

Delay, skew, power, area, and skew sensitivity are the most important concerns in current clock-tree design. With the increasing complexity of synchronous ASICs, clock skew and clock-signal delay have become important factors in determining circuit performance [2, 4, 10, 17]. Wire width process variations during fabrication can significantly impact the delay and skew; thus it is important to consider the sensitivity of a design to inter-chip process variations [13]. As reported in [7], power dissipation of a clock tree play a key role in overall chip's power dissipation. Therefore, it is desirable to simultaneously consider delay, skew, power, area, and sensitivity in clock-tree design.

Algorithms for routing-tree optimization have been proposed in much of the literature recently [3, 4, 5, 12, 13, 15, 17]. The works in [3, 5, 12, 15] are designed for general routing tree, hence they cannot handle clock tree issues such as skew and sensitivity. Although [4, 13, 14, 17] consider sensitivity, skew, and/or delay, most of these algorithms only size wires and do not minimize power and area. Moreover, existing algorithms suffer long runtime and large storage requirements. For example, [13, 17] convert the skew minimization problem into the least-squares minimization problem. However, due to the storage and inversion of large gradient matrices, their respective runtimes *per iteration* and storage requirements are about cubic and quadratic in the problem size.

We present in this paper an algorithm for simultaneously optimizing the above-mentioned objectives by sizing wires and buffers in clock trees. Our algorithm, based on the Lagrangian relaxation method, can simultaneously optimize delay, power, and area with very low skew and sensitivity; it relaxes the constraints scaled with Lagrangian multipliers into its objective function and then iteratively solve the subproblems resulted from dynamically adjusting the Lagrangian multipliers. Our algorithm is extremely fast, economical, and accurate; it requires only linear storage *overall* and linear runtime *per iteration* for adjusting wire and buffer sizes. For example, we can solve a 6201-wire-segment clock-tree problem in about 1-minute runtime and 1.3-MB memory and still guarantee pico-second precision on an IBM RS/6000 workstation.

*This work was partially supported by the Texas Advanced Research Program under Grant No. 003658459.

2 Preliminaries

We use the following notations in this paper.

- T: A clock tree with a driver w_0 at the root (source) and a set of s sinks $\{N_1, N_2, ..., N_s\}$.
- w_i: i-th wire segment or buffer. w_i is a wire segment when $1 \le i \le n$, or a buffer when $n+1 \le i \le n+m$ or $i = 0$.
- x_i, l_i: Size and length of w_i, respectively.
- $\mathbf{x}$: $\mathbf{x} = (x_0, x_1, x_2, \ldots, x_{n+m})$ is a wire- and buffer-sizing solution.
- ρ_i: Resistance of wire per unit length at unit width, when $1 \le i \le n$; resistance of unit-size buffer, when $i = 0$ or $n+1 \le i \le n+m$.
- ε_i: Area capacitance of wire per unit square, when $1 \le i \le n$; capacitance of unit-size buffer, when $i = 0$ or $n+1 \le i \le n+m$.
- r_i: Resistance of w_i. $r_i \approx \rho_i l_i / x_i$, when $1 \le i \le n$; $r_i \approx \rho_i / x_i$, when $n+1 \le i \le n+m$ or $i = 0$.
- c_i: Capacitance of w_i. $c_i \approx \varepsilon_i l_i x_i$, when $1 \le i \le n$; $c_i \approx \varepsilon_i x_i$, when $n+1 \le i \le n+m$ or $i = 0$.
- U_i, L_i: Upper bound and lower bound of the size of w_i, respectively, i.e., $L_i \le x_i \le U_i$, $0 \le i \le n+m$.
- P_i: All wires and buffers on the path from the source to sink N_i (including N_i).
- T_i: All wires and buffers in the subtree of T rooted at w_i (excluding w_i).
- $parent(w_i)$: Parent of w_i.
- $Child(w_i)$: *Set* of w_i's children.
- $Ans(w_i)$: All wires or buffers on the path from w_i to the nearest upstream buffer or the root (excluding w_i).
- $Dec(w_i)$: All wires, buffers, or sinks on the paths from w_i to the neighboring downstream buffers or sinks (excluding w_i).
- R_i: Upstream resistance of w_i; $R_i = \sum_{w_j \in Ans(w_i)} r_j$.

- C_i: Downstream capacitance of w_i; $C_i = \sum_{w_j \in Dec(w_i)} (C_j + c_j) + \sum_{N_j \in Dec(w_i)} \tilde{c_j}$, where $\tilde{c_j}$ is the capacitance of sink N_j, $1 \le j \le s$.
- A: Area of a clock tree; $A = \sum_{i=1}^{n} w_i l_i$.

See Figure 1 for an illustration of R_i and C_i.

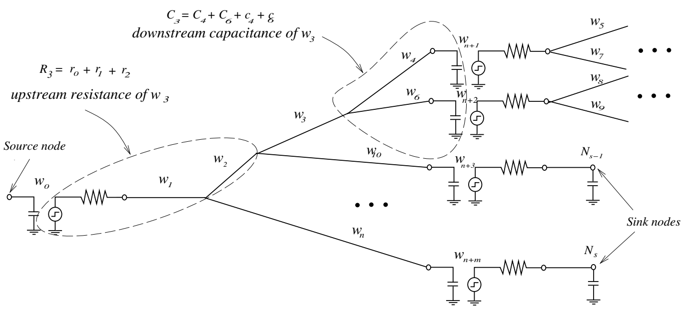

Figure 1: Upstream resistance and downstream capacitance.

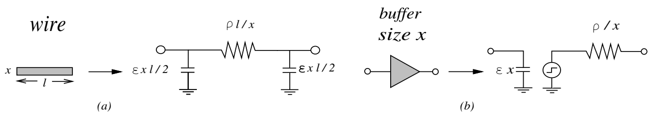

Figure 2: RC model for wire and buffer.

We use a distributed resistance-capacitance (RC) segment to represent a branch of a clock tree (see Figure 2(a)). The distributed RC segment can be modeled as an equivalent lumped π-circuit. The lumped resistance and capacitance of the π-model of an RC segment w_i are approximated by $\rho_i l_i / x_i$ and $\varepsilon_i x_i l_i$, respectively. We use the switch-resistor model to compute buffer delays (see Figure 2(b)) and apply the Elmore delay model [8] to approximate signal delays in a subtree. Given a distributed RC routing tree T, its signal delay at sink N_i is computed by $D_i = \sum_{w_j \in P_i, 1 \le j \le n} r_j (C_j + \frac{c_j}{2}) + \sum_{w_j \in P_i, n+1 \le j \le n+m} r_j C_j + r_0 C_0$.

In practical CMOS applications, capacitive dissipation (due to charging and discharging of load capacitances) usually dominates the other types of power dissipation [5]. Hence, we consider only the capacitive dissipation in this paper. Given a clock tree, its power dissipation P can be approximated by $P \approx f C_{tot} V_{dd}^2$, where f is the clock frequency and C_{tot} is the total capacitance of the tree.

Clock skew is defined as the maximum difference in the delays from the clock source to clock sinks; that is, the skew of a clock tree, $S = \max_{i,j} |D_i - D_j|$. Given wire width w, the *skew sensitivity*, Δ, is defined as the maximum difference between skews under varying values of w due to process variations [4]. The goal of sensitivity minimization is to find an optimal w such that Δ is minimized.

This paper addresses the clock-tree wire- and buffer-sizing problem, targeting multiple objectives such as delay, skew, power, area, and sensitivity. We give the formulation for the wire- and buffer-sizing problem as follows:

- The *Clock-Tree Wire- and Buffer-Sizing Problem*
  Given: A clock tree T with the source N_0 and sinks $\{N_1, N_2, \ldots, N_s\}$, wire segments $\{w_1, w_2, \ldots, w_n\}$, buffers $\{w_0, w_{n+1}, w_{n+2}, \ldots, w_{n+m}\}$, upper bounds $\{U_0, U_1, \ldots, U_{n+m}\}$, and lower bounds $\{L_0, L_1, \ldots, L_{n+m}\}$.
  Objective: Find an $\mathbf{x}$ that minimizes $\max_{1 \le i \le s} D_i$, S, P, A, and/or Δ.

3 Delay/Power/Area Minimization

We formulate the wire- and buffer-sizing problem for simultaneous delay, power, and area minimization as follows:

$$\begin{aligned} \mathcal{M}: \textit{Minimize} \quad & \alpha D_{max} + \beta P + \gamma A \\ \textit{Subject to} \quad & D_i(\mathbf{x}) \le D_{max},\ 1 \le i \le s, \\ & L_i \le x_i \le U_i,\ 0 \le i \le n+m, \\ & D_{max} > 0, \end{aligned}$$

where α, β, and γ are given constants. Note that D_{max} is a variable we introduced to minimize maximum delay. As shown above, there are two sets of inequalities. The first set of s inequalities is used to ensure that every sink satisfies its delay constraint. The second set of inequalities is used to ensure that the size of every wire segment and buffer satisfies its size constraints.

By dividing both sides of the delay, lower bound, and upper bound constraints by D_{max}, x_i, and U_i, respectively, we can rewrite these constraints as $\frac{D_i(\mathbf{X})}{D_{max}} \le 1$, $\frac{L_i}{x_i} \le 1$, and $\frac{x_i}{U_i} \le 1$. Hence $\mathcal{M}$ becomes a geometric programming problem which can be reduced to a convex programming problem by an exponential transformation [6]. However, since general geometric programming solvers usually involve gradient matrices inversions, their storage and runtime requirements are at least quadratic and cubic in the problem size, respectively. Therefore, it is desirable to develop an efficient algorithm for solving this problem.

Our approach for solving $\mathcal{M}$ is based on Lagrangian relaxation [1, 9]. We relax the delay constraints into the objective function by introducing Lagrange multipliers λ_i's, $1 \le i \le s$, one for each delay constraint $D_i(\mathbf{x}) \le D_{max}$. We have the Lagrangian-relaxation subproblem for $\mathcal{M}$ as follows:

$$\begin{aligned} \mathcal{M}': \textit{Minimize} \quad & \alpha D_{max} + \beta P + \gamma A + \sum_{i=1}^{s} \lambda_i (D_i(\mathbf{x}) - D_{max}) \\ \textit{Subject to} \quad & L_i \le x_i \le U_i,\ 0 \le i \le n+m, \\ & D_{max} > 0. \end{aligned}$$

For each λ, let $\mathcal{L}(\lambda)$ be the optimal objective function value of $\mathcal{M}'$. It is well known that $\mathcal{L}(\lambda)$ is a lower bound of the optimal objective value of $\mathcal{M}$ [1, 9]. On the other hand, any feasible solution of $\mathcal{M}$ is an upper bound of the optimal objective value. Hence, we can use these two bounds to evaluate the quality of a current solution and to determine the termination criteria. By the Kuhn-Tucker theory [11] and the fact that $\mathcal{M}$ is equivalent to a convex programming problem, we have the following theorem.

Theorem 1 *$(\mathbf{x}^*, D_{max}^*)$ is an optimal solution if and only if there exists a vector $\lambda^* = (\lambda_1^*, \lambda_2^*, \ldots, \lambda_s^*)$ such that*

(1) $\sum_{i=1}^{s} \lambda_i^* = \alpha$;

(2) $\lambda_i^* (D_i(\mathbf{x}^*) - D_{max}^*) = 0, 1 \le i \le s$;

(3) $D_i(\mathbf{x}^*) - D_{max}^* \le 0, 1 \le i \le s$;

(4) $\lambda_i^* \geq 0, 1 \leq i \leq s;$

(5) $x_i^* = min\,(U_i, max\,(L_i, \Phi_i)),$
where $\Phi_i = \sqrt{\frac{\rho_i \mu_i C_i}{\beta p_0 + \gamma + \varepsilon_i \sum_{w_j \in Ans(w_i)} r_j \mu_j}},$
$p_0 = f\varepsilon_i V_{dd}^2$, *and* $\mu_i = \sum_{N_j \in T_i} \lambda_j,\ 0 \leq i \leq n+m.$

Based on the above analyses, we need to find $\mathbf{x}^*$ and λ^* to solve Problem $\mathcal{M}$. Once λ_i's are assigned, we can compute $\mathbf{x}^*$ based on Theorem 1(5). Hence we can adopt a two-level approach to solve this problem: In the outer loop, we dynamically adjust sink weights λ_i's; weight associated with each sink is proportional to the signal delay of the sink. In the inner loop, we find an optimal wire- and buffer-sizing solution for the given λ_i's. With this in mind, we present the Lagrangian-relaxation-based algorithm shown in Figure 3; the algorithm iteratively adjusts the multipliers based on the delay information associated with sinks and solves the corresponding Lagrangian relaxation subproblems. Our algorithm runs in $O(pqn)$ time using $O(n)$ storage, where p is the number of iterations (**A3**–**A6**) in OWBA, and q is the number of iterations (**S2**–**S3**) in LRS. Empirically, the overall runtime approaches linear. We have the following theorem.

Theorem 2 *Algorithm OWBA converges to a global optimal solution.*

Algorithm: OWBA (Optimal Wire- and Buffer-sizing Algorithm)
A1. Let $k = 0$, $x_i = L_i$, $0 \leq i \leq n+m$.
A2. $\lambda_i = 1/s$, $1 \leq i \leq s$.
A3. Call Subroutine LRS.
A4. Recursively compute all sink delays D_i's; let $D_{max} = \max_i(D_i(\mathbf{x}))$.
A5 Adjust sink weights λ_i's according to the formula
$\lambda_i = \lambda_i + \theta_k(D_i(\mathbf{x}) - D_{max})$, $1 \leq i \leq s$,
where stepsize θ_k satisfies $\lim_{k\to\infty} \theta_k = 0$ and $\sum_{j=1}^{k} \theta_j \to \infty$.
A6. $k = k+1$.
A7. Repeat **A3**–**A6** until $D_{max} - \mathcal{L}(\lambda) \leq$ error bound.

Subroutine: LRS (Lagrangian-Relaxation Subroutine)

S1. Compute all the wire-segment weights in a bottom-up manner using the formula: $\mu_i = \sum_{w_j \in Child(w_i)} \lambda_j$.
S2. Compute the downstream capacitance in a bottom-up manner using the formula $C_i = \sum_{w_j \in Child(w_i)} (C_j + c_j)$.
S3. Traverse the clock tree in the dept-first-search order; During visiting w_i, keeping other wire and buffer sizes fixed, compute $R_i' = R_{parent_i} + \mu_{parent_i} r_{parent_i}$, $C_i' = \mu_i C_i$, and
$x_i = min\left(U_i, max\left(L_i, \sqrt{\frac{\rho_i C_i'}{\beta p_0 + \gamma + \varepsilon_i R_i'}}\right)\right)$
S4. Repeat **S2**–**S3** until no improvement.

Figure 3: The optimal wire- and buffer-sizing algorithm.

4 Skew and Sensitivity Minimization

By definition, clock skew $S = max_{i,j}|D_i - D_j|$. To reduce clock skew, we need not only to reduce signal delays but also to balance delays. We have the following formulation to minimize clock skew:

$$\begin{array}{lll} \mathcal{M}1: & Minimize & \alpha D_{max} + \beta P + \gamma A + \delta(D_{max} - D_{min}) \\ & Subject\ to & D_i(\mathbf{x}) \leq D_{max},\ 1 \leq i \leq s, \\ & & D_i(\mathbf{x}) \geq D_{min},\ 1 \leq i \leq s, \\ & & L_i \leq x_i \leq U_i,\ 0 \leq i \leq n+m, \\ & & D_{max} > 0,\ D_{min} > 0. \end{array}$$

Since $\mathcal{M}1$ introduces negative coefficients, it is no longer a geometric programming problem and hence there is no guaranty of convexity. For a non-convex problem, global optimal solution may not easily be found. We resort to the the following heuristic approach.

Following the Lagrangian relaxation procedure, we relax the delay constraints by bringing them into the objective function with associated Lagrange multipliers λ_i's and σ_i's, $1 \leq i \leq s$, where λ_i and σ_i are the Lagrange multipliers associated with the delay constraint $D_i(\mathbf{x}) \leq D_{max}$ and $D_i(\mathbf{x}) \geq D_{min}$, respectively. We have the Lagrangian relaxation subproblem for $\mathcal{M}1$ as follows:

$$\begin{array}{lll} \mathcal{M}1': & Minimize & \alpha D_{max} + \beta P + \gamma A + \delta(D_{max} - D_{min}) + \\ & & \sum_{i=1}^{s} \lambda_i(D_i(\mathbf{x}) - D_{max}) + \sum_{i=1}^{s} \sigma_i(D_{min} - D_i(\mathbf{x})) \\ & Subject\ to & L_i \leq x_i \leq U_i,\ 0 \leq i \leq n+m, \\ & & D_{max} > 0,\ D_{min} > 0. \end{array}$$

Hence by repeatedly solving the Lagrangian relaxation subproblems, we can minimize clock skew.

Sensitivity is used to measure the influence of production variations. It can be measured by the first derivative of the signal delay with respect to wire or buffer size which can be shown to be $|\varepsilon_i R_i - \frac{\rho_i C_i}{x_i^2}|$. Restricting the sensitivity of every wire and buffer to be smaller than Δ_{max}, we get $|\varepsilon_i R_i - \frac{\rho_i C_i}{x_i^2}| \leq \Delta_{max}$. In our algorithm, we dynamically add the following constraints into Step **S3** of LRS during execution:

$$x_i \leq min\left(U_i, max\left(L_i, \sqrt{\frac{\rho_i C_i}{\varepsilon_i R_i - \Delta_{max}}}\right)\right), if\ \varepsilon_i R_i - \frac{\rho_i C_i}{x_i^2} \geq 0,$$
$$x_i \geq min\left(U_i, max\left(L_i, \sqrt{\frac{\rho_i C_i}{\varepsilon_i R_i + \Delta_{max}}}\right)\right), if\ \varepsilon_i R_i - \frac{\rho_i C_i}{x_i^2} < 0.$$

While the above approaches reduce skew and sensitivity, they also tend to increase delay, power, area, and runtime at the same time. In fact, we observe that Algorithm OWBA already significantly reduces skew and sensitivity while optimizing delay, power, and/or area. Since Algorithm OWBA tends to allocate higher weights to sinks with longer delay and smaller weights to the ones with shorter delay. Consequently, the longer paths get more resources than the shorter ones. This effect directly balances the delays between different sinks and hence reduces clock skew. We observed that OWBA is a good heuristic for sensitivity minimization as well. To see this, let us consider delay minimization (i.e. $\alpha = 1, \beta = \gamma = 0$). Our algorithm essentially iteratively sizes all buffers and wire segments, one at a time (in Step S3 of LRS) while keeping the sizes of all other buffers/wire segments fixed. It can be proved that S3 not only optimally size a buffer/wire segment, it also simultaneously minimizes the sensitivity with respect to average delay.

Ckt	# Nodes	Delay (ns) Initial	Delay (ns) Final	Delay (ns) Reduce%	Skew (ps) Initial	Skew (ps) Final	Skew (ps) Reduce%	Δ_{max} $(10^{-15} sec/\mu m)$ Initial	Δ_{max} Final	Δ_{max} Reduce%	Runtime (sec)	Storage (kbytes)	Error (ps)
r1	533	0.775	0.161	481	64	16	400	7.96	0.53	1501	3.50	148	0.2
r2	1195	2.108	0.379	556	221	12	1842	15.86	0.65	2436	13.38	280	0.4
r3	1723	3.376	0.572	590	154	36	427	20.58	0.68	3039	17.25	388	0.6
r4	3805	9.087	1.376	660	716	92	778	42.13	1.48	2850	54.87	812	1.4
r5	6201	15.864	2.312	686	974	102	955	63.51	2.06	3085	67.04	1300	2.3
Avg	-	-	-	595	-	-	691	-	-	2582	-	-	-

Table 1: Experimental results in delay, skew, and sensitivity.

5 Experimental Results

We implemented our algorithm and tested on the five circuits $r1$–$r5$ used in [16] on an IBM RS/6000 workstation. The per micron resistance and capacitance used are $3m\Omega$ and $0.02fF$, respectively. The lower and upper bounds for wire widths are $1\mu m$ and $10\mu m$, respectively. Table 1 lists the names of the circuits, numbers of wire segments in the circuits, delays, skews, sensitivity, runtimes, and storage requirements. It shows that our algorithm, on the average, reduced the respective delay, skew, and sensitivity by 595%, 691%, and 2582% after wire-sizing. Further, our algorithm is extremely fast and economical. For example, for the circuit r5 with 6201 wire segments, our algorithm needed only 67-second runtime and 1.3-MB storage to achieve 2.3-ps precision. In Figure 4(a) (Figure 4(b)), the runtime (storage) requirement (represented by the vertical axis) is plotted as a function of the number of wire segments in a circuit (denoted by the horizontal axis). It shows that the runtime and storage requirements of our algorithm approach linear in the number of wire segments. Figure 5 shows the relationship among the maximum delays (D_{max}), the value of the $\mathcal{L}(\lambda)$, and clock skew at each iteration. The horizontal axis and the vertical axis represent the number of iterations and D_{max}, $L(\lambda)$, and skew (in pico second), respectively. The gap between D_{max} and $\mathcal{L}(\lambda)$ is the error bounds of our algorithm.

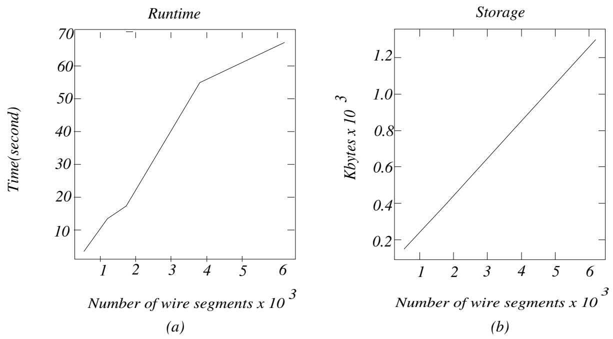

Figure 4: Runtime and Storage requirements.

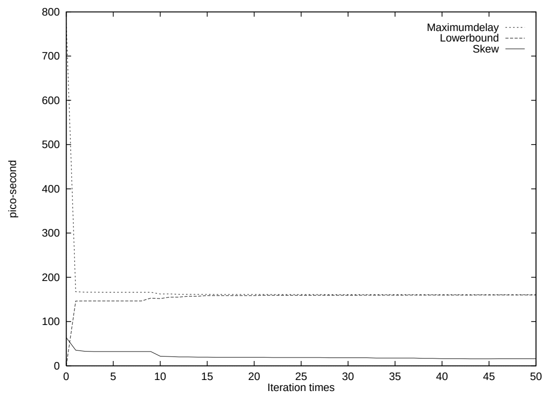

Figure 5: The values of D_{max} (upper bound), $\mathcal{L}(\lambda)$ (lower bound), and clock skew during execution on r1.

6 Acknowledgment

The authors thank Prof. Leon S. Lasdon, Prof. Patrick Jaillet, Prof. Ross Baldick, Prof. Jonathan Bard, and Prof. Jayant Rajgopal for their invaluable help and comments.

References

[1] Leon S. Lasdon, *Optimization Theory for Large Systems*, Macmillan Publishing Co., INC. 1970.

[2] H. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley Pub. Company Inc., 1990.

[3] Chung-Ping Chen, D. F. Wong. "A fast algorithm for optimal wire-sizing under Elmore delay model" *Proc IEEE ISCAS*, 1996.

[4] J. Chung and C.-K. Cheng, "Skew sensitivity minimization of buffered clock tree," *Proc. ICCAD*, pp. 280–283, 1994.

[5] J. Cong and K.-S. Leung, "Optimal wiresizing under elmore delay model," *IEEE TCAD* 14(3), pp. 321–336, 1995.

[6] R. J. Duffin, E. L. Peterson, and C. Zener, *Geometric Programming—Theory and Application*, John Wiley & Sons, Inc, 1967.

[7] D. Dobberpuhl and R. Witek, "A 200MHz 64B dual-issue CMOS microprocessor," *Proc. IEEE ISSCC,* pp. 106–107, 1992.

[8] W. C. Elmore, "The transient response of damped linear networks with particular regard to wide band amplifiers," *J. Applied Physics*, 19(1), 1948.

[9] M. L. Fisher, "An applications oriented guide to Lagrangian relaxation," *Interfaces*, 15:2, pp. 10-21, March-April 1985.

[10] M. A. B. Jackson, et al., "Clock routing for high performance ICs," *Proc. DAC*, pp. 573–579, June 1990.

[11] D. G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley Pub. Company Inc., 1984.

[12] N. Menezes, R. Baldick, and L. T. Pillage, "A sequential quadratic programming approach to concurrent gate and wire-sizing," *Proc. ICCAD*, 1995.

[13] N. Menezes, S. Pullela, F. Dartu, and L. T. Pillage, "RC interconnect syntheses—a moment fitting approach," *Proc. ICCAD*, Nov. 1994.

[14] S. Pullela, N. Menezes, and L. T. Pillage, "Reliable non-zero skew clock trees using wire width optimization," *Proc. 30th ACM/IEEE Design Automation Conference*, pp. 165–170, June 1993.

[15] S. S. Sapatnekar, "RC interconnect optimization under the Elmore delay model," *Proc. ICCAD*, pp. 387–391, 1994.

[16] R.-S. Tasy, "Exact zero skew," *IEEE TCAD*, 1993.

[17] Q. Zhu, W. W.-M. Dai, and J. G. Xi, "Optimal sizing of high-speed clock networks based on distributed RC and lossy transmission line models," *Proc. ICCAD*, pp. 628–633, 1993.