Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Digital Circuit Optimization via Geometric Programming

Stephen P. Boyd, Seung-Jean Kim, Dinesh D. Patil, Mark A. Horowitz,

To cite this article:
Stephen P. Boyd, Seung-Jean Kim, Dinesh D. Patil, Mark A. Horowitz, (2005) Digital Circuit Optimization via Geometric Programming. Operations Research 53(6):899-932. https://doi.org/10.1287/opre.1050.0254

OPERATIONS RESEARCH
Vol. 53, No. 6, November–December 2005, pp. 899–932
ISSN 0030-364X | EISSN 1526-5463 | 05 | 5306 | 0899

DOI 10.1287/opre.1050.0254

Digital Circuit Optimization via Geometric Programming

Stephen P. Boyd, Seung-Jean Kim, Dinesh D. Patil, Mark A. Horowitz
Department of Electrical Engineering, Stanford University, Stanford, California 94305-9510
{boyd@stanford.edu, sjkim@stanford.edu, ddpatil@stanford.edu, horowitz@stanford.edu}

This paper concerns a method for digital circuit optimization based on formulating the problem as a geometric program (GP) or generalized geometric program (GGP), which can be transformed to a convex optimization problem and then very efficiently solved. We start with a basic gate scaling problem, with delay modeled as a simple resistor-capacitor (RC) time constant, and then add various layers of complexity and modeling accuracy, such as accounting for differing signal fall and rise times, and the effects of signal transition times. We then consider more complex formulations such as robust design over corners, multimode design, statistical design, and problems in which threshold and power supply voltage are also variables to be chosen. Finally, we look at the detailed design of gates and interconnect wires, again using a formulation that is compatible with GP or GGP.

Subject classifications: programming: geometric; engineering: computer-aided design; digital circuit optimization.
Area of review: Invited Papers.
History: Received January 2004; revision received May 2005; accepted May 2005.

1. Introduction

1.1. Digital Circuit Sizing

The complexity of digital integrated circuits (ICs) has been increasing exponentially since around 1960, with the number of components or devices in a single IC more than doubling every 18 months. Some current ICs contain over 100 million devices and a similar number of wires connecting them. The design of such complex ICs relies heavily on electronic design automation (EDA) and computer-aided design (CAD) technologies.

In this paper, we focus on just one step in the design of a digital circuit: the selection of appropriate sizes for the devices, gates, and wires. These sizes can correspond to physical dimensions such as the width of a transistor channel or wire segment, or to more abstract parameters like the *drive strength* of a gate (which is closely related to the physical size of the gate). We will also consider extensions in which other design variables, such as threshold and power supply voltage, are also selected. The choice of these design variables can have a strong influence on the three primary top level objectives: the total area of the circuit, the total power it consumes, and the speed at which it can operate.

Our starting point is a given circuit topology that realizes the required circuit behavior. The circuit topology consists of an interconnection of *gates*, which are small circuits that carry out basic Boolean functions such as inversion (logical NOT), conjunction (logical AND), and disjunction (logical OR), and storage elements or *registers*, which are small circuits that store Boolean values at each clock cycle. The circuit topology can come directly from a circuit designer or from a *logic synthesis step*, in which the circuit topology is generated automatically from a high level description of the required behavior. The logic gates, storage elements, and their interconnections are fixed; what remains is to choose the size of each gate or, in custom design, each device in each gate. (There are many other components in digital integrated circuits that we do not consider, including, for example, circuitry for power and clock distribution, memory, and input/output functions.)

The circuit topology can be partitioned into a set of *combinational logic blocks*, which are subcircuits consisting of logic gates between registers. Each combinational logic block computes a particular Boolean function. For example, a 32-bit adder has 64 Boolean inputs, consisting of the Boolean representation of the two 32-bit numbers to be added, and 32 Boolean outputs, which is the Boolean representation of the sum (ignoring overflow).

We will focus on the sizing problem for combinational logic blocks, because a method for sizing individual blocks can be extended to one for sizing a set of blocks, i.e., a larger *sequential logic circuit* that includes registers. In the simplest case this can be done by designing each combinational logic block separately (for example, to minimize power or area, subject to some timing requirements); for more complex problem formulations it can also be done in a coordinated manner described in §2.5.

For general background on digital circuit design, we refer the reader to the recent books (Weste and Harris 2004,

Rabaey et al. 2002, Hodges et al. 2004) that describe the sizing problem and its context in detail. The influential book by Sutherland et al. (1999) is almost entirely devoted to the sizing problem. Sizing of digital circuits is a well-researched field, with hundreds of papers on the topic; see, e.g., the articles by Fishburn and Dunlop (1985), Passy (1998), Chen et al. (2004), Kim et al. (2004), Kasamsetty et al. (2000), Sapatnekar (1996), and Sapatnekar et al. (1993) and the references therein.

1.2. Sizing Optimization via Geometric Programming

In this paper, we focus on a particular approach, in which the sizing problem is modeled (at least approximately) as a *geometric program* (GP), a special type of mathematical optimization problem. We refer the reader to the paper, "A Tutorial on Geometric Programming" (Boyd et al. 2004) for an introduction to geometric programming, some of the basic tricks used to formulate problems in GP form, a number of examples, and an extensive list of references. We also give a very short introduction to GP in the Appendix.

GP-based circuit sizing is by no means new; it has been used for digital circuits since the 1980s. In 1985, Fishburn and Dunlop proposed a method for transistor and wire sizing, based on Elmore delay, that was later found to be a GP. Since then many digital circuit design problems have been formulated as GPs or related problems. Work on gate and device sizing (the main topics of this paper) can be found in, e.g., Chu and Wong (2001b), Passy (1998), Cong and He (1999), Kasamsetty et al. (2000), Matson and Glasser (1986), Pattanaik et al. (2003), Shyu et al. (1988), Sancheti and Sapatnekar (1996), Sapatnekar and Chuang (2000), and Sapatnekar et al. (1993). These are all based on gate delay models that are compatible with geometric programming; see Kasamsetty et al. (2000), Sakurai (1988), Sutherland et al. (1999), Rubenstein et al. (1983), and Abou-Seido et al. (2004) for more on such models. Work on interconnect sizing (also addressed in this paper) includes Alpert et al. (2001b), Cong and He (1996), Cong and Koh (1994), Cong et al. (1996), Cong and Leung (1995), Cong and Pan (2002), Chen et al. (2004), Chen and Wong (1999), Gao and Wong (1999), Kay and Pileggi (1998), Lee et al. (2002), Lin and Pileggi (2001), and Sapatnekar (1996); simultaneous gate and wire sizing is considered in Chen et al. (1999) and Jiang et al. (2000). In some of these papers, the authors develop custom methods for solving the resulting GPs instead of using general purpose interior-point methods (see, e.g., Chu and Wong 2001b, Ismail et al. 2000, Young et al. 2001). For some simple problems, analytic solutions are available (see, e.g., Chu and Wong 2001a, Gao and Wong 1999). Other problems in digital circuit design where GP plays a role include buffering and wire sizing (Alpert et al. 2004; Chu and Wong 1999, 2001a), sizing and placement (Chen et al. 2000), yield maximization (Kim et al. 2004, Patil et al. 2005), parasitic reduction (Qin and Cheng 2003), clock tree design (Vittal and Marek-Sadowska 1997), and routing (Borah et al. 1997). Geometric programming has also been used for the design of nondigital circuits, e.g., analog circuits (Dawson et al. 2001, Hershenson 2003, Hershenson et al. 1998, Mandal and Visvanathan 2001, Vanderhaegen and Brodersen 2004), mixed-signal circuits (Colleran et al. 2003, Hassibi and Hershenson 2002, Hershenson 2002), and RF (radio frequency) circuits (Hershenson et al. 1999; Mohan et al. 1999, 2000; Xu et al. 2004). Geometric programming has also been used in floorplanning, for both analog and digital circuits (Moh et al. 1996).

Our focus will not be on any particular sizing problem, and certainly not on the particular results of any of our numerical examples; instead our focus will be on the *modeling* of a variety of problems in GP form. There are several advantages to modeling a problem, at least approximately, as a GP. The first is computational: new methods can solve even large GP problems exactly, globally, and efficiently. Even if these new methods are not exploited to solve the problem, the knowledge that a problem is (approximately) a GP has utility. For example, it tells us that a particular logarithmic transformation of the variables and constraints yields a convex optimization problem, and this can be exploited to develop a more efficient solution method. In addition, we have the very useful conclusion that any local solution of the problem is in fact global. If an ad hoc solution method can be shown to find a local solution, we can conclude that it finds a global solution. (For more discussion of these issues, see Boyd et al. 2004, Boyd and Vandenberghe 2004.) Another advantage to expressing a sizing problem in GP form is conceptual: we claim that GP serves as a unifying standard form for circuit sizing problems, the same way that linear programming (LP) serves as a unifying standard form for a wide variety of simple resource allocation problems.

The traditional approach to solving the GPs that arise from digital circuit sizing problems is to use an ad hoc or custom method specially designed for the particular problem, which does not exploit the GP structure. These methods typically analyze the timing of a circuit, identify a critical path, and then resize one or more devices or gates along the critical path. This is repeated until no further improvement occurs. This approach was developed in the 1980s, when general purpose methods for solving GPs were slow and limited to small problems, and some of the problems were not yet recognized as GPs. While this traditional approach allows the solution of even very large circuit sizing problems, it can only handle simple problem formulations. Moreover, this approach requires the development (and tuning) of a new solution method for each new problem formulation.

In the mid 1990s *interior-point algorithms* for GP were developed, which can solve even large-scale GPs extremely efficiently and reliably (Boyd and Vandenberghe 2004, Nesterov and Nemirovsky 1994, Nocedal and Wright 1999,

and Ye 1997). This opens the possibility of formulating even large circuit sizing problems as large-scale GPs, and directly solving them using interior-point methods. This approach easily handles complex problem formulations and a wide variety of constraints and objectives.

Like all methods, the GP modeling approach has advantages and disadvantages. One advantage is that complex interactions between the optimization variables are easily accounted for, and additional constraints are easily added. As we will see, the method handles complex problems, such as joint optimization of devices sizes, threshold, and supply voltage; robust design over corners or taking statistical variations into account; and the design of circuits that operate in multiple modes (such as a low power and a high performance mode). When compared with other methods based on numerical optimization, methods based on GP (and interior-point solution methods) have the advantage of not needing an initial design, or any algorithm parameter tuning, and always finding the global solution.

We can also list a number of shortcomings of the approach. The method does not give much insight into *why* some set of specifications cannot be achieved, nor does it suggest how the designer might change the circuit topology to do better. While solving GPs is fast, it is not as fast as methods that choose sizes using simple rules, with a few passes over the circuit.

1.3. Outline

We start with the simplest possible setup: simple gate scaling, with delay modeled as a simple resistor-capacitor (RC) time constant. We then add various layers of complexity and modeling accuracy, such as accounting for different fall and rise times, effects of signal transition times, static and dynamic power, and so on. Next, we consider problems in which threshold and power supply voltage are also design variables to be chosen. Finally, we consider device sizing, accounting for internal device capacitances, and distributed wire capacitance. In all cases our focus is on formulating the problem as a GP, or an extension of GP called *generalized geometric programming* (GGP) (Boyd et al. 2004).

We consider design problems ranging from simple ones involving trade-offs among area, speed, and power, to more complex and interesting formulations such as robust design (i.e., a design that meets the specifications despite variations in supply voltage, temperature, and device parameters), statistical design (one that takes manufacturing and other statistical variations into account), and multimode designs (i.e., a design that is meant to operate in multiple modes).

1.4. Audience and Goals

This paper is meant to serve several purposes. First, it is a tutorial, aimed at those new to GP-based digital circuit sizing, that covers well-known GP-based methods, e.g., gate sizing with simple RC models and wire and device sizing using Elmore delay. At the same time we describe a number of new problems, and variations on known ones, that can be solved using GP modeling. For example, the observation that joint device sizing, threshold voltage, and supply voltage optimization can be carried out using GP is new, as far as we know. We hope that researchers in digital circuit sizing will find the more complex problem formulations and models interesting. For researchers in optimization, we hope that the paper will serve as an introduction to what we feel is a promising application area for optimization.

Our main goal is to convince the reader that GP modeling is widely applicable in digital circuit sizing optimization, both in the variety of phenomena and effects it can model (such as signal transition time, distributed RC loads, parasitic capacitance, and leakage power), and the number of problem formulations it lends itself to (such as robust, multimode, and statistical optimization), and therefore is worth studying.

2. Gate Scaling

2.1. Basic Gate Scaling with Simple Models

2.1.1. Circuit Topology. We consider a combinational logic circuit that consists of a set of logic gates between (edge-triggered) *flip-flop* registers, connected by some interconnect wires or nets. We refer to the outputs of the flip-flops that drive the combinational logic block as its *primary inputs*, and the inputs of the flip-flops that are driven by the combinational logic block as the *primary outputs*. (The input and output flip-flops do not have to be distinct; the output of a single flip-flop can drive the combinational logic block, and its input can be connected to the output of the combinational logic block.) For simplicity we assume that each gate has a single output, which is connected to the inputs of one or more gates, and possibly a primary output. Each input of each gate is connected to the output of another gate, or a primary input. We assume that the circuit is acyclic, i.e., it contains no feedback paths or cycles of gates connected to each other. The circuit topology is described by a *netlist*, or a directed acyclic graph (DAG) in which each node represents a gate or primary input and each arc represents a wire or net connection. The gates are labeled $1, \ldots, n$, and the primary inputs and outputs are labeled starting at $n + 1$. The set of primary inputs is denoted by PI. The fan-in of gate or primary output i, denoted $\mathrm{FI}(i)$, is the set of predecessors of i in the DAG, i.e., the gates or primary inputs which drive an input of gate i. The fan-out of gate or primary input i, denoted $\mathrm{FO}(i)$, is the set of successors of i in the DAG, i.e., the set of gates or primary outputs which gate i drives. (When i is a primary input, $\mathrm{FI}(i) = \varnothing$, and when i is a primary output, $\mathrm{FO}(i) = \varnothing$. When i is a primary output, $\mathrm{FI}(i)$ has the form $\{j\}$, where j is the gate that drives primary output i.) The output gates are those with no fan-out gates.

Figure 1. A combinational logic block with seven gates, labeled $1, \ldots, 7$, with primary inputs labeled 8, 9, 10 and primary outputs labeled 11, 12.

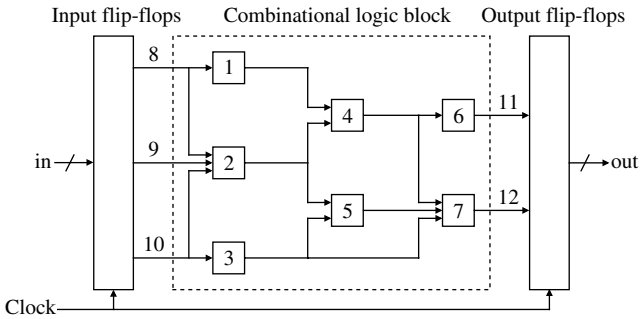


Figure 1 shows a small example with seven gates (labeled $1, \ldots, 7$), three primary inputs $(8, 9, 10)$, and two primary outputs $(11, 12)$. In this example, we have, for example,

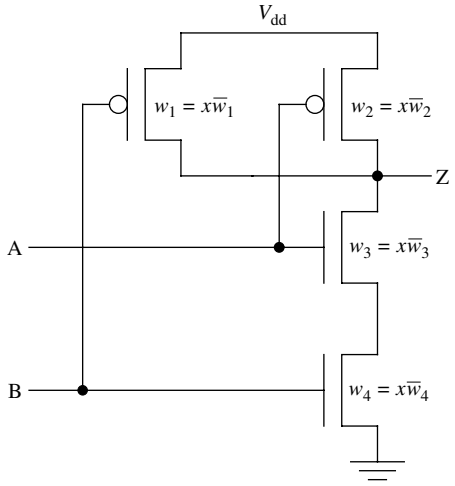
$$\mathrm{FI}(2) = \{8, 9, 10\}, \qquad \mathrm{FO}(2) = \{4, 5\},$$
$$\mathrm{FI}(6) = \{4\}, \qquad \mathrm{FO}(6) = \{11\}.$$

Each of the seven gates in this combinational logic block carries out some particular Boolean function of its inputs, but we do not specify these functions, because for the moment it does not matter. To make the example more concrete, the reader can imagine that gate 1 is an inverter (i.e., performs logical inversion), gate 2 is a three input AND gate (i.e., its output is the logical conjunction of its inputs), gate 3 is a buffer (i.e., its output is the same as its input), and so on. This combinational logic block computes a function from three Boolean variables (the primary inputs) to two Boolean variables (the primary outputs). We note that combinational logic blocks found in current ICs are far larger than this example, typically with many tens (or more) of inputs and outputs, and thousands (or more) gates.

2.1.2. Scale Factor. With each gate we associate a *scale factor* or *normalized size* $x_i \geqslant 1$ that scales the widths of the devices used to form the gate. The scale factor $x_i = 1$ corresponds to a minimum-sized gate, and a scale factor $x_i = 16$ (say) corresponds to a version of the gate in which all devices have width 16 times the widths of the devices in the minimum-sized gate. This is illustrated with the NAND gate shown in Figure 2, which computes the Boolean function $Z = \overline{AB}$. The minimum size NAND gate (corresponding to scale factor $x = 1$) has device widths $\bar{w}_1, \ldots, \bar{w}_4$. With scale factor x, the device widths are $x\bar{w}_1, \ldots, x\bar{w}_4$. The scale factors of the gates, which are the design variables to be chosen, affect the total circuit area, the power consumed by the circuit, and the delay of the circuit.

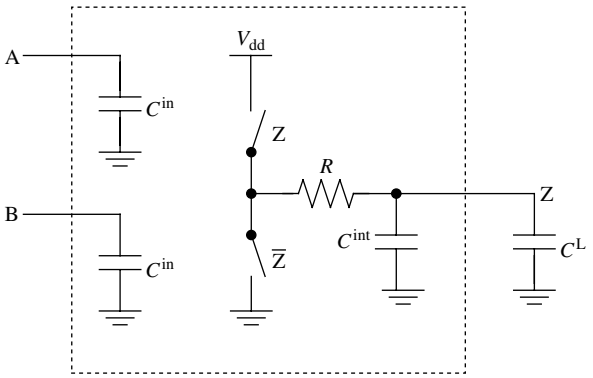
We will see later (in §2.2.2) that the gate scale factor need not have the literal meaning of a scaling applied to every device in a gate; in general, it is just a parameter meant to describe the gate's size or ability to drive a capacitive load (and, indeed, is sometimes referred to as the gate *drive strength*). For now, though, we keep the literal interpretation to derive simple models that relate scale factor to circuit performance. We also note that the scale factors x_i are often restricted to a finite set of allowed values, and are not allowed to take on a continuum of values. We will discuss both of these issues later.

Figure 2. Schematic diagram of a two-input NAND gate, with inputs A and B, output Z, and scale factor x.



2.1.3. RC Gate Delay Model. Each gate has a *delay*, which is the time it takes its output signal to transition to a new value, after its input values have transitioned to new values. We start with the simplest model of gate delay, based on the simple resistor-capacitor (RC) circuit shown in Figure 3. When the output of the gate is logical 1, the upper switch is closed, and the lower switch is open, so the output terminal Z is connected to the power supply voltage V_{dd}, which represents logical 1. When the output of the gate is logical 0, the upper switch is open and the lower

Figure 3. RC model of a CMOS gate with two inputs, A and B, and one output, Z.



switch is closed, so the output terminal Z is connected to ground, which represents logical 0.

Each gate has an input capacitance C^{in}, which is a load for the gate that drives it (or a primary input). Each gate also has an intrinsic or internal capacitance C^{int}, which appears as a load on the gate. These capacitances are modeled as linear functions of the scale factor,

$$C_i^{\text{in}} = \bar{C}_i^{\text{in}} x_i, \qquad C_i^{\text{int}} = \bar{C}_i^{\text{int}} x_i,$$

where $\bar{C}_i^{\text{in}}$ and $\bar{C}_i^{\text{int}}$ are the input capacitance and intrinsic capacitance of gate i with unit scaling.

If i represents a primary output, C_i^{in} denotes its input capacitance, i.e., the input capacitance of the associated flip-flop, which we assume is given. The load capacitance C_i^{L} that gate or primary input i drives is the sum of the input capacitances of the gates and primary outputs it drives:

$$C_i^{\text{L}} = \sum_{j \in \text{FO}(i)} C_j^{\text{in}}.$$

(When i represents a primary input, C_i^{L} is the input capacitance at the primary input which is a load to the flip-flop that drives it.) The load capacitance C_i^{L} is a linear function of the scale factors x, with positive coefficients, so it is a *posynomial function* of x. (See Boyd et al. 2004 for the definition of posynomial.)

Each gate has driving resistance R_i, which is inversely proportional to the scale factor,

$$R_i = \bar{R}_i / x_i,$$

where $\bar{R}_i$ is the driving resistance of gate i with unit scale factor. Thus, R_i is a *monomial function* of x_i. (Here we use the term "monomial" in the sense used in geometric programming, not the standard meaning in algebra; see Boyd et al. 2004.) Gate i thus has parameters $\bar{C}_i^{\text{in}}$, $\bar{C}_i^{\text{int}}$, and $\bar{R}_i^{\text{int}}$. These parameters depend on the particular process technology used to fabricate the circuit, the power supply voltage, and other parameters which for now are assumed known and fixed.

Using this simple RC model of a gate and its load, we can approximate the gate delay D_i as

$$D_i = 0.69 R_i \left(C_i^{\text{int}} + C_i^{\text{L}}\right),$$

which is the time required for the output voltage of an RC circuit to reach the midpoint between the logic voltage levels (i.e., 0 and V_{dd}). Because R_i, C_i^{L}, and C_i^{int} are posynomials of the scale factors, the delay of each gate is also a posynomial function of the gate scale factors.

2.1.4. Path and Circuit Delay. A path through the circuit is a sequence of gates, with each gate's output connected to the following gate's input. The delay of a path is the sum of delays of the gates on the path, and hence is posynomial. The circuit delay D (also called the worst-case or longest-path delay) is the maximum delay along any path through the circuit. The delay is meant to be the elapsed time when all primary outputs of the circuit are valid, after the primary inputs change. Because the worst-case delay is the maximum path delay, over all possible paths, it is a maximum of a set of posynomial functions, and therefore is a *generalized posynomial* of the scale factors. (See Boyd et al. 2004 for an extensive discussion of generalized posynomials.)

Some paths through the circuit correspond to logic transitions that cannot occur, i.e., there is no input transition that causes transitions in the outputs of all gates along the path. These paths are called *false paths*. It is possible to consider the "true" (worst-case) delay of the circuit, which is the maximum delay over all paths on which each gate actually transitions, for some input transition. This "true" delay is also a generalized posynomial, because it is the maximum over a set of path delays, each of which is a posynomial of the scale factors. Calculating delay ignoring the false path issue, which is called *static timing analysis*, is common practice because it leads to an efficient recursive formulation (which we describe later). The delay obtained from static timing analysis is in any case an upper bound on the true worst-case delay over all possible transitions.

As a specific example, the circuit shown in Figure 1 has only seven paths from primary inputs to primary outputs. The (worst-case, static-timing analysis) delay is given by

$$\begin{aligned} D = \max\{&D_1 + D_4 + D_6,\ D_1 + D_4 + D_7,\ D_2 + D_4 + D_6,\\ &D_2 + D_4 + D_7,\ D_2 + D_5 + D_7,\\ &D_3 + D_5 + D_6,\ D_3 + D_7\}. \end{aligned} \tag{1}$$

To find the true worst-case delay of this circuit we need to know the Boolean functions that gates $1, \ldots, 7$ compute. There are $2^3 = 8$ input states, and thus $8 \cdot 7 = 56$ possible input transitions. For each of these transitions, we determine the subset of the seven paths on which all gate outputs change value. (Brute force enumeration of the transitions is possible for a circuit with three primary inputs, but not in the general case, because there are $2^k(2^k - 1)$ transitions for a circuit with k primary inputs.)

A combinational logic block can be thought of as a *project network* or *activity network*, in which the gates represent tasks or activities to be carried out, and the wires in the circuit give the *precedence relations* among the tasks: the task of gate i is to compute its associated Boolean function of its inputs, once they have become valid. The delay of the circuit corresponds to the *makespan* of the associated project network, i.e., the time it takes to complete all tasks, assuming that each task starts as soon as it can (i.e., its precedents have finished). Static timing analysis corresponds closely to PERT (project evaluation and review techniques) (Davis 1966; Dodin 1984; Elmaghraby 1970, 1977; Robillard and Trahan 1976) methods for identifying one critical path (or K most critical paths) in a digital circuit; see, e.g., Blaauw et al. (2002).

2.1.5. Area and Power. In the simplest model, we approximate the (physical) area of gate i as proportional to the scale factor x_i, so the total area of the (combinational logic block) circuit has the form

$$A = \sum_{i=1}^{n} x_i \bar{A}_i,$$

where $\bar{A}_i$ is the area of gate i with unit scaling. The total circuit area is an affine function of the scale factors, with positive coefficients, and so is a posynomial function.

An output transition of gate i is associated with energy loss due to charging (or discharging) its intrinsic and load capacitances, given by

$$\left(C_i^{\mathrm{int}} + C_i^{\mathrm{L}}\right) V_{\mathrm{dd}}^2/2,$$

where V_{dd} is the circuit supply voltage. Similarly, a transition of primary input i is associated with the energy loss

$$C_i^{\mathrm{L}} V_{\mathrm{dd}}^2/2.$$

The total *dynamic power* dissipated by the circuit can be written as

$$P_{\mathrm{dyn}} = \sum_{i \in \mathrm{PI}} f_i C_i^{\mathrm{L}} V_{\mathrm{dd}}^2 + \sum_{i=1}^{n} f_i \left(C_i^{\mathrm{int}} + C_i^{\mathrm{L}}\right) V_{\mathrm{dd}}^2, \tag{2}$$

where f_i is the *activity frequency* of primary input i or the output of gate i. (The activity frequency is defined as the number of falling/rising cycles of the gate output per second, and so involves two transitions.) The parameters f_i are often given as a fraction of the clock frequency f_{clk}, as $f_i = \alpha_i f_{\mathrm{clk}}$. The fractions α_i are called *gate activity factors*. The activity factors are often guessed, or estimated from a behavioral simulation of the circuit. For fixed f_i and supply voltage, the dynamic power is a linear function of the gate scale factors, with positive coefficients, and therefore is a posynomial.

The *static power* or *leakage power* of a gate is the power dissipated by the circuit even when the inputs are constant. The static power of a gate is also approximately proportional to its scale factor (assuming constant supply voltage) and depends on the input state of the gate. If we have (either by assumption or simulation) the probability distribution of the gate input state values, we can form the average static leakage power for a gate, which is proportional to the scale factor. The total (average) static power of the circuit has the form

$$P_{\mathrm{stat}} = \sum_{i=1}^{n} \bar{I}_i^{\mathrm{leak}} x_i V_{\mathrm{dd}}, \tag{3}$$

where $\bar{I}_i^{\mathrm{leak}}$ is the leakage current of gate i with unit scaling. The static power is a linear function of the scale factors, with positive coefficients, and thus a posynomial.

The total power is

$$P = P_{\mathrm{dyn}} + P_{\mathrm{stat}},$$

which is a linear function of the scale factors, with positive coefficients, and therefore a posynomial. (For more details on dynamic and static power modeling, see Chandrakasan and Brodersen 1995, Pedram 1996, Roy et al. 2003.)

For the moment, we are assuming that power supply voltage and clock frequency (which scales the activity frequencies, assuming given activity factors) are constant. But if clock frequency or supply voltage are varied, the two components of the total power vary in different ways. (We consider this in more detail in later sections.)

2.1.6. Basic Gate Scaling Problem. We can now formulate a basic gate scaling problem: choose the scale factors to give minimum delay, subject to limits on the total area and power. This can be expressed as the optimization problem

$$\begin{array}{ll} \text{minimize} & D \\ \text{subject to} & P \leqslant P^{\max}, \quad A \leqslant A^{\max}, \\ & 1 \leqslant x_i, \quad i = 1, \ldots, n. \end{array} \tag{4}$$

Here, $P^{\max}$ and $A^{\max}$ are given limits on the total power and area, and the optimization variables are the scale factors $x_1, \ldots, x_n$. Because D is a generalized posynomial, and A and P are posynomials, this problem is a GGP and so can be solved very efficiently. The solution of problem (4) gives the fastest circuit, with given power and area budgets.

We can interpret the basic gate scaling problem (4) as a project network optimization problem. We can think of the vector of gate scale factors as an allocation of resources (scale factors) to the tasks (gates), subject to the resource limitations given by the power and area constraints. The objective is to allocate the resources in such a way that the time to complete all tasks is minimized. The interesting twist in the gate scaling problem is that the time to complete a task (i.e., a gate delay) depends not only on the amount of the resource allocated to it (i.e., its own gate scaling), but also the resources allocated to its successor tasks (i.e., its fan-out gates). When we include signal transition times in the delay model (the Appendix), the task completion time will also depend on the resources allocated to its predecessor tasks.

There are many variations on the basic gate scaling problem. For example, we can choose area or power as the objective to be minimized and impose a limit on delay. As a simple extension, we can formulate the problem as one of maximizing the circuit clock frequency f_{clk}, instead of minimizing the circuit delay. We assume that the circuit delay D must not exceed some fixed fraction, such as 80%, of the clock cycle time $1/f_{\mathrm{clk}}$. (The extra margin takes into account the delay of the flip-flops, as well as a required

setup time.) This can be expressed as $D \leqslant 0.8/f_{\mathrm{clk}}$, which can be expressed as the inequality

$$(1/0.8) f_{\mathrm{clk}} D \leqslant 1,$$

where the left-hand side is a generalized posynomial of the gate scale factors *and* the clock frequency. Assuming that the activity factors are fixed, the dynamic power scales linearly with f_{clk}, whereas the static power does not scale at all. Thus, the total power is a posynomial of the scale factors and the clock frequency. To maximize the clock frequency, we form the problem

$$\begin{array}{ll} \text{maximize} & f_{\mathrm{clk}} \\ \text{subject to} & (1/0.8) f_{\mathrm{clk}} D \leqslant 1, \\ & P \leqslant P^{\max}, \quad A \leqslant A^{\max}, \\ & 1 \leqslant x_i, \quad i = 1, \ldots, n. \end{array}$$

This is a GGP with variables $x_1, \ldots, x_n$ and f_{clk}.

2.1.7. Trade-Off Analysis and Optimal Sensitivities. We can minimize a composite objective such as a weighted sum or maximum, or a product of positive powers of delay, area, and power. This produces designs that are *Pareto optimal*, or *efficient*, with respect to the three objectives: delay, area, and power. (Solving the basic problem (4) also yields Pareto optimal points, provided the area and power constraints are active.) By generating many such designs, for various values of the weights (with a weighted sum objective), or limits $P^{\max}$ and $A^{\max}$ (when the basic problem (4) is solved) we find the globally *optimal trade-off surface* for delay, power, and area (see Boyd et al. 2004).

The optimal trade-off surface has many practical uses. If, for example, a great reduction in optimal delay can be obtained with only a modest increase in area and power (say), the designer may wish to increase the area and power limits. The optimal trade-off surface is also useful if many instances of the combinational logic block are to be used in a large circuit, or in several circuit designs, with (possibly) different power, area, and delay specifications. Finally, the optimal trade-off surface allows us to jointly optimize a set of combinational logic blocks in a sequential circuit (see §2.5).

When we solve the GGP (4), we get (at no additional computational cost) the *optimal sensitivities* or *Lagrange multipliers* for the power and area constraints. These numbers give local predictions of optimal delay, as $P^{\max}$ and $A^{\max}$ are changed a small amount. For example, if $S^{\mathrm{area}} = -2.3$, a 1% increase in allowed area $A^{\max}$ would lead to a reduction in delay of around 2.3%. These sensitivities are extremely useful because they quantify how "binding" the power and area constraints are. (For more discussion of optimal sensitivities, see Boyd et al. 2004, Boyd and Vandenberghe 2004.)

2.1.8. Other Constraints. There are many other constraints, compatible with geometric programming, that can be added. For example, we can impose a maximum on the scale factors, $x_i \leqslant x_i^{\max}$, or we can impose a maximum load capacitance at each primary input. (Because the load capacitance at each input is a sum of gate capacitances, it is a posynomial function of the scaling factors, and imposing a maximum is a posynomial inequality constraint.)

We also mention a few common constraints that are *not* compatible with GGP. One is the requirement that the gate scaling factors x_i lie in discrete sets, such as the set of integers, or powers of two. With constraints like these, the gate scaling problem (4) becomes a *mixed-integer generalized geometric program*. Dealing with discrete constraints in a mixed-integer GGP is sometimes called *snapping* (to the grid of allowed values). In theory, such problems are very difficult to solve exactly (whereas in theory, GGPs are easy to solve), but in practice there are several good heuristics for obtaining at least an approximate solution. The simplest snapping method is to solve the GGP, ignoring the discrete constraints, and then round each x_i to its nearest valid value; see §7.3 in Boyd et al. (2004) for more discussion. Provided we do not insist on always finding the true global minimum, these snapping heuristics can be quite effective.

Another constraint not compatible with GP is a *minimum delay constraint*. Such a constraint requires that the shortest path delay, on any path in the circuit, must exceed a given minimum value. In GGP we can only impose a minimum on a monomial function (see the Appendix), and the minimum path delay of a circuit is certainly not a monomial. However, such constraints can usually be dealt with after the design via GP is completed, by inserting extra delay elements in the paths that are too fast. An efficient design typically has few or no very short path delays, so little modification is needed to comply with short path constraints.

2.1.9. Dynamic Programming GP Formulation. The delay D is the maximum of a set of posynomials, i.e., the delays of all paths through the circuit. Even small circuits can have a very large number of paths, in which case it is not practical to form an expression for D by listing all the paths. A simple recursion for D can be used to avoid enumerating all paths. We define T_i as the maximum delay over all paths that start at a primary input and end with gate i. We can interpret T_i as the latest time at which the output of gate i can transition, assuming that the primary input signals transition at $t = 0$. For this reason T_i is sometimes called the (latest) *signal arrival time* at the output of gate i.

We can express T_i via the recursion

$$T_i = \max_{j \in \mathrm{FI}(i)} T_j + D_i, \tag{5}$$

with the starting conditions $T_i = 0$ for i corresponding to a primary input. This recursion states that the latest signal

arrival time at the output of a gate is equal to the maximum signal arrival time of its fan-in gates, plus its own delay. The delay D of the whole circuit is given by the maximum over all T_i, which is the same as the maximum over all output gates:

$$D = \max_{i=1,\ldots,n} T_i = \max\{T_i \mid i \text{ an output gate}\}.$$

The recursion (5) shows that each T_i is a generalized posynomial of the scaling factors, because generalized posynomials are closed under addition and maximum.

As a specific example, for the circuit shown in Figure 1, the recursion is

$$\begin{aligned}
&T_i = D_i, \quad i = 1, 2, 3,\\
&T_4 = \max\{T_1, T_2\} + D_4,\\
&T_5 = \max\{T_2, T_3\} + D_5,\\
&T_6 = T_4 + D_6,\\
&T_7 = \max\{T_3, T_4, T_5\} + D_7,\\
&D = \max\{T_6, T_7\}.
\end{aligned}$$

For this small example, the recursion gives no real savings over expression (1) above based on enumeration of the paths, but in larger circuits the savings can be dramatic.

This recursion for D allows us to reformulate problem (4), which is a generalized geometric program, as a geometric program. Problem (4) is equivalent to the GP

$$\begin{aligned}
&\text{minimize} \quad \overline{T}\\
&\text{subject to} \quad \overline{T}_j \leqslant \overline{T} \quad \text{for } j \text{ an output gate},\\
&\qquad \overline{T}_j + D_i \leqslant \overline{T}_i \quad \text{for } j \in \mathrm{FI}(i),\\
&\qquad P \leqslant P^{\max}, \quad A \leqslant A^{\max},\\
&\qquad 1 \leqslant x_i, \quad i = 1, \ldots, n,
\end{aligned} \tag{6}$$

with optimization variables x_i (the gate scale factors), $\overline{T}_i$ for i not a primary input (upper bounds on signal arrival times), and $\overline{T}$ (an upper bound on the overall delay). For primary inputs, we take $\overline{T}_i = 0$. Although we have introduced $n+1$ new variables, the constraints in (6) are *sparse*: each of the timing constraints involves only a few variables (assuming that fan-ins and fan-outs are not very large). This sparsity can be exploited by a GP solver to obtain great efficiency; we refer the reader to part III of Boyd and Vandenberghe (2004) for more on exploiting structure to obtain efficiency.

To give a rough idea of the current state of the art, a typical circuit with 1,000 gates can be solved in a few seconds on a small personal computer (see Kim et al. 2004). Our computational experience with larger problems suggests that it grows quite modestly in the total number of gates. For typical problems, the running time grows as n^α, with α a bit larger than one. (The growth exponent depends on the particular structure and algorithm used, but in any case is between 1 and 2.)

2.1.10. Ladner-Fischer Adder Design Example. In this section, we describe a simple numerical example, which we will use to illustrate various design problems throughout the paper. We consider a 32-bit adder, with a particular topology first given by Ladner and Fischer (see, for example, Weste and Harris 2004), and called a *Ladner-Fischer adder*. The adder circuit consists of 461 gates ($n = 461$), with 64 primary inputs and 32 primary outputs. The total number of paths, from primary inputs to primary outputs, is 3,214. The circuit has a *depth* of 8, i.e., the longest path in the circuit passes through 8 gates.

The Ladner-Fischer adder contains five types of gates, with associated functions and model parameters listed in Table 1. The last two gate types, AOI21 and OAI21 (which are acronyms for "and-or-invert" and "or-and-invert"), are *compound gates*. The model parameters come from the *logical effort model*, described in Sutherland et al. (1999). The drive strength value $\bar{R} = 0.48$ is chosen so that the delay of a unit size inverter with no load is $0.69 \cdot 0.48 \cdot 3 = 1$. In other words, the time unit is normalized to the delay of a unit scale inverter, with no load. (This time unit is often called τ.) For a current state-of-the-art IC process technology, this time unit is on the order of 15 ps.

The gate area is the total width of the devices in the gate (because the gate lengths are always chosen to be the smallest value allowed in the technology). The unit is the width of the NMOS device in a unit scaled inverter. The capacitance unit is the capacitance of the NMOS device in a unit scaled inverter. For a current IC process technology, these have values on the order of 100 nm, and a few fF, respectively. The average leakage current parameters are taken from a current IC process and scaled so that static power is around 10% of the dynamic power for a circuit operating near maximum speed at normal temperature.

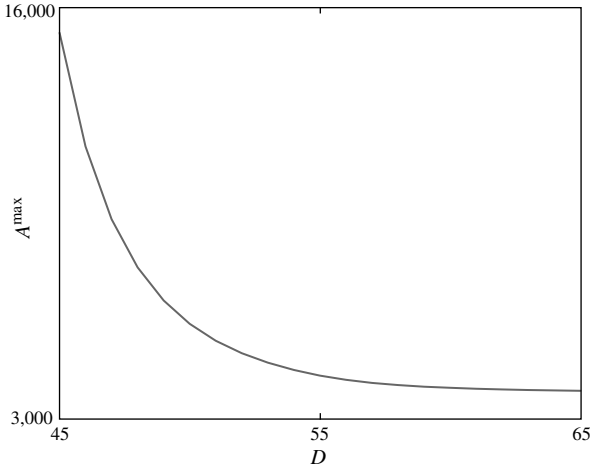
We take the load capacitance of the primary outputs to be $C_i^{\mathrm{L}} = 6$. We impose a limit on area but not on power, and solve the basic gate scaling problem (4) using the dynamic programming formulation (6). The resulting GP has about 1,000 variables and 4,000 constraints but is very sparse, with each constraint (except for the area constraint) depending on at most three variables. (The area constraint is not sparse, but sparse solvers can easily handle a small number of dense constraints; see Boyd and Vandenberghe 2004.)

Table 1. The five gate types used in the Ladner-Fischer adder.

Gate type	Function	$\bar{C}^{\mathrm{in}}$	$\bar{C}^{\mathrm{int}}$	$\bar{R}$	$\bar{A}$	$\bar{I}^{\mathrm{leak}}$
INV	$\overline{A}$	3	3	0.48	3	0.006
NAND2	$\overline{AB}$	4	6	0.48	8	0.007
NOR2	$\overline{A+B}$	5	6	0.48	10	0.009
AOI21	$\overline{AB+C}$	6	7	0.48	17	0.003
OAI21	$\overline{(A+B)C}$	6	7	0.48	16	0.003

Note. The first column gives the gate name; the second column gives the logic function the gate implements, and the remaining five columns give the model parameters.

Figure 4. Optimal trade-off curve of minimum delay D versus maximum area $A^{\max}$ for the 32-bit Ladner-Fischer adder circuit, with no power limit.



The GP is solved using a generic GP solver (that exploits sparsity) in around one second, on a basic PC. Figure 4 shows the optimal trade-off curve of minimum delay D versus maximum area $A^{\max}$.

2.1.11. Inverter Chain. In a few cases, the gate scaling problem (4) can be solved analytically. One famous example is a chain of N inverters driving a load capacitance C^{L}, shown in Figure 5, with no limit on the scale factors, area, or power, and a specified input capacitance C^{in} (Hodges et al. 2004, Rabaey et al. 2002). The input capacitance constraint fixes the first gate scaling to be $x_1 = C^{\mathrm{in}}/\bar{C}^{\mathrm{in}}$. Thus, the variables are $x_2, \ldots, x_N$.

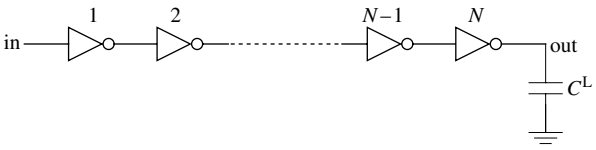
The load capacitance on gate i is the input capacitance of gate $i+1$, i.e., we have

$$C_i^{\mathrm{L}} = C_{i+1}^{\mathrm{in}} = x_{i+1}\bar{C}^{\mathrm{in}},$$

and the load capacitance on gate N is $C_N^{\mathrm{L}} = C^{\mathrm{L}}$. The delay D of the inverter chain is given by

$$\begin{aligned} D &= 0.69\sum_{i=1}^{N-1} R_i\left(C_i^{\mathrm{L}} + C_i^{\mathrm{int}}\right) + 0.69R_N\left(C^{\mathrm{L}} + C_N^{\mathrm{int}}\right) \\ &= 0.69\sum_{i=1}^{N-1}(\bar{R}/x_i)(x_{i+1}\bar{C}^{\mathrm{in}} + x_i\bar{C}^{\mathrm{int}}) \\ &\quad + 0.69(\bar{R}/x_N)(C^{\mathrm{L}} + x_N\bar{C}^{\mathrm{int}}) \\ &= 0.69\bar{R}\bar{C}^{\mathrm{in}}\left(N(\bar{C}^{\mathrm{int}}/\bar{C}^{\mathrm{in}}) + \sum_{i=1}^{N-1} x_{i+1}/x_i + (C^{\mathrm{L}}/\bar{C}^{\mathrm{in}})/x_N\right). \end{aligned}$$

Figure 5. A chain of N inverters driving a load C^{L}.



Setting the derivative with respect to x_i to zero (because we consider here the unconstrained problem), we obtain

$$-x_{i+1}/x_i^2 + 1/x_{i-1} = 0$$

for $i = 2, \ldots, N-1$. From this we conclude that $x_i = \sqrt{x_{i-1}x_{i+1}}$, i.e., that the optimal scale factor for each gate (except the first and last) is the geometric mean of the scale factors of the previous and next gates. It follows that the optimal scale factors form a geometric series, $x_i = f^{i-1}x_1$, where f is the constant factor between successive stages.

Setting the derivative of D with respect to x_N to zero, we obtain $1/x_{N-1} = (C^{\mathrm{L}}/\bar{C}^{\mathrm{in}})/x_N^2$. Substituting

$$\begin{aligned} x_{N-1} &= f^{N-2}x_1 = f^{N-2}(C^{\mathrm{in}}/\bar{C}^{\mathrm{in}}), \\ x_N &= f^{N-1}x_1 = f^{N-1}(C^{\mathrm{in}}/\bar{C}^{\mathrm{in}}) \end{aligned}$$

into this equation, we find that $f = (C^{\mathrm{L}}/C^{\mathrm{in}})^{1/N}$. The optimal scale factors are therefore

$$x_i = (C^{\mathrm{in}}/\bar{C}^{\mathrm{in}})(C^{\mathrm{L}}/C^{\mathrm{in}})^{(i-1)/N}, \quad i = 1, \ldots, N.$$

With these optimal scalings, the delay of each stage is the same

$$D_i = 0.69\bar{R}\bar{C}^{\mathrm{in}}((C^{\mathrm{L}}/C^{\mathrm{in}})^{1/N} + (\bar{C}^{\mathrm{int}}/\bar{C}^{\mathrm{in}})),$$

and the overall delay of the inverter chain is

$$D = 0.69N\bar{R}\bar{C}^{\mathrm{in}}((C^{\mathrm{L}}/C^{\mathrm{in}})^{1/N} + (\bar{C}^{\mathrm{int}}/\bar{C}^{\mathrm{in}})).$$

(This, in turn, can be minimized over N to find the optimal number of stages for a given ratio of load to input capacitance.)

The appearance of the geometric mean, and a geometric series, is not surprising. Indeed, GPs are solved by the change of variables $e^{y_i} = x_i$; the result above says that the optimal values of y_i (which are the logarithms of the scalings) are evenly spaced and that each y_i is the average of y_{i+1} and y_{i-1}.

2.1.12. Logical Effort Method. The inverter chain result above is the starting point for the logical effort method developed by Sutherland et al. (1999). The logical effort method starts with the same RC model described above, along with a specific method for deriving the RC model parameters from a schematic diagram of the gate. Sutherland et al. then develop a set of design rules for gate sizing, based on extensions of the inverter chain result above. The method can be thought of as a fast method for approximately solving problem (4) (without the scaling factor, area, or power constraints) using a few simple calculations and just one, or a few, passes over the circuit. Logical effort has been widely used for gate sizing; see, e.g., Ebergen et al. (2004) and Rezvani and Pedram (2003). Specific applications include adder design Seidel and Even (2004) and fast low power decoder design for

RAMs Bharadwaj and Horowitz (2000). (The circuit sizing problems formulated in these papers can all be cast as GPs.)

As Sutherland et al. (1999) make clear, the main point of the logical effort method is *not* to solve the delay minimization problem (4) exactly (which doesn't make much sense, because the RC delay model is itself an approximation) but instead to give the circuit designer intuition about the circuit, to help him or her choose an appropriate number of stages in the circuit, or find another circuit topology if needed. In contrast, the GGP formulation (4) or GP formulation (6) of the problem does not give the designer much insight.

On the other hand, the GP formulation has some advantages over the logical effort method. First, it always solves problem (4) exactly, taking into account multiple paths, branching and re-combining, as well as area and power constraints (or any other constraint compatible with GP). While solving the GP (6) is not as fast as a simple logical effort pass over the circuit, it is not much slower, provided the sparsity of the problem is exploited.

This suggests a combined approach. The choice of circuit topology, including, for example, the number of stages, can be guided by the logical effort method. Sizing based on logical effort can be used for the initial evaluation of candidate topologies; once a good candidate topology has been identified, GP can be used to choose the scalings, taking into account branching, power and area constraints, and more accurate models, our next topic.

2.2. Better GP Models

In this section, we show how the simple gate scaling problem (4) can be extended to use more accurate models for area, delay, and power, while retaining compatibility with GP (so the problem can still be solved very efficiently).

2.2.1. Generalized Posynomial Models. Our first observation is that we can replace the simple linear functions we used for the area and power of a gate with any generalized posynomial of the scale factors. These generalized posynomial functions might be found by a more refined analysis, or by fitting generalized posynomials to the actual cell areas obtained after layout (i.e., the detailed physical design of the logic gates), or the average power obtained by circuit simulation, for a number of values of the scaling parameter. (See Boyd et al. 2004 for a discussion of generalized posynomial fitting techniques.) The same observation holds for the driving resistance and input capacitance of a cell: The simple inverse proportionality and proportionality relations described above can be replaced with any generalized posynomial functions of the scaling.

The delay does not need to be the simple product of load capacitance and driving resistance: it can be any increasing generalized posynomial of load capacitance and driving resistance. Even more generally, we can dispense with the notion of drive resistance, and model gate delay directly as a generalized posynomial function of the load capacitance and the scale factor x.

2.2.2. Abstract and Multiple Scale Factors. When the area, power, input capacitance, and driving resistance are generalized posynomials of the scaling factor, the scaling factor x in effect becomes an *abstract parameter* that describes the size or strength of a gate, and need not have the strict interpretation given above of a scale factor for each device in the gate.

As a simplified example, consider a buffer that consists of a chain of two inverters. Suppose that x gives the scaling of the devices in the second inverter. Motivated by the inverter chain example described above, we might scale the devices in the first inverter by $\sqrt{x}$. Using very simple electrical models, then, the area and power of the gate would grow as $x + x^{1/2}$, the input capacitance would scale as $x^{1/2}$, and the drive resistance would scale as x^{-1}. The total delay of the buffer would have the form $a + bx^{1/2} + x^{-1}C^{\mathrm{L}}$, where the terms $a + bx^{1/2}$ give the delay of the first inverter, driving the second inverter. Each of the functions above is a posynomial, so this model gives a GP formulation of the scaling factor problem. But in this example, x does not have the strict interpretation of a scaling factor for all devices in the gate.

Even more generally, a gate can be parameterized by more than one parameter. In this case, x is a vector of parameters. Again, all that matters is that the area, power, input capacitance, and drive resistance should be generalized posynomial functions of the parameters. A simple example here is given by an inverter chain with some fixed number of inverters, with the stages scaled according to a geometric series (again, following the inverter chain example above). We describe the chain by *two* parameters, which give the input stage size and the output stage size. Here too the area, power, input capacitance, and drive resistance will be posynomials of the two parameters that describe the inverter chain.

In the extreme case, we can independently vary the width of every device in the gate, so the parameters are simply the individual device widths. This is essentially custom (device level) design, the topic of §4.

2.2.3. Parasitic Capacitance. If the parasitic capacitance of a net (to ground) is known, it can be added to the load capacitance seen by the gate driving the net. Adding such a term keeps D_i posynomial, so the scaling problem remains a GP. More generally, we can use estimates of parasitic capacitance, provided they are generalized posynomials of the scaling factors. (These estimates can involve factors such as the fan-out of a gate or might come from preliminary placement and routing.)

One interesting issue arises here. We get accurate values for the parasitic capacitances only after detailed layout, which occurs only after sizing the gates. When we reoptimize the gate scaling factors, we may need to modify the layout, which changes the parasitic capacitances. The general approach is to iterate between sizing, layout, and parasitic extraction (i.e., determining the parasitic capacitances), hoping for convergence. At each optimization step,

we use an estimate of the parasitic values, such as the values from the last iteration.

One method that enhances the probability of convergence is to add a cost term or constraint in the reoptimization steps that penalizes deviation between the previous design and the new one. This results in gate scalings that, one hopes, are not too different from the previous gate scalings. This in turn leads to layout and routing, and therefore parasitics, that are not too different from the previous design. By adding a penalty term of the form

$$\lambda \sum_{i=1}^{n} \max\left\{x_i/x_i^{\mathrm{prev}}, x_i^{\mathrm{prev}}/x_i\right\},$$

where $\lambda > 0$ is a parameter, and x_i^{prev} is the gate scaling in the previous design iteration, we obtain a new design in which many of the scalings are (exactly) the same as in the previous design. (The parameter λ controls the trade-off between optimizing the circuit and keeping many gates the same size.) This is the GP analog of adding a sum of absolute values penalty in convex optimization, which is a well-known heuristic for finding a sparse solution; see Boyd and Vandenberghe (2004, Ch. 6).

2.2.4. Clock Skew. Another simple extension that is readily handled via GP involves (known) clock skews at the input and output flip-flops. These can be incorporated in the circuit as an extra layer of fictitious gates at the primary inputs and outputs, with fixed delays that are independent of the scaling factors (and zero area and power). Equivalently, we can initialize the arrival times of the primary input signals using the clock skews of the input flip-flops and subtract the clock skews at the output flip-flops from the output gates.

2.2.5. Distinguishing Gate Inputs and Transitions. So far we have associated one delay for each logic gate, which is independent of the gate input, the particular transition, and so on. Greater accuracy can be obtained by distinguishing between rising and falling delays at the gate output, assigning different delays from each gate input to the output, or even, in the extreme case, modeling a different delay for every possible input transition.

One common approach is to replace the single gate delay D_i, with four different delays, for each input j to gate i:

$$D_{ji}^{\mathrm{rr}}, \quad D_{ji}^{\mathrm{rf}}, \quad D_{ji}^{\mathrm{fr}}, \quad D_{ji}^{\mathrm{ff}}.$$

The first, D_{ji}^{rr}, denotes the maximum possible delay from a rising transition on input j to a rising transition on the gate output. (The maximum is over all possible input transitions with input j rising and output i rising.) The other three delays are defined similarly. Depending on the gate logic, one or more of these transitions might be impossible, in which case we can leave the associated delay undefined. For example, a simple NAND gate is always inverting, so a falling output can only occur when one (or more) inputs are rising, and vice versa. For such a gate, we would only have two delays, D_{ji}^{rf} and D_{ji}^{fr}, for each input j.

We can form a simple recursion for propagating a bound on maximum rising and falling signal arrival times at the output of each gate. Let T_i^{r} (T_i^{f}) denote an upper bound on the rising (falling) signal arrival time at gate i, assuming the primary input transitions occur at $t = 0$. We can propagate these via the recursions

$$T_i^{\mathrm{r}} = \max_{j\in \mathrm{FI}(i)} \max\left\{T_j^{\mathrm{r}} + D_{ji}^{\mathrm{rr}}, T_j^{\mathrm{f}} + D_{ji}^{\mathrm{fr}}\right\},$$

$$T_i^{\mathrm{f}} = \max_{j\in \mathrm{FI}(i)} \max\left\{T_j^{\mathrm{r}} + D_{ji}^{\mathrm{rf}}, T_j^{\mathrm{f}} + D_{ji}^{\mathrm{ff}}\right\}.$$

In the maxima here we can ignore or remove any transitions that are not logically valid. As an example, suppose that a gate is always inverting, i.e., a rising input transition always yields a falling output transition (or no transition), and vice versa. Then, the recursions above can be simplified to

$$T_i^{\mathrm{r}} = \max_{j\in \mathrm{FI}(i)} \left(T_j^{\mathrm{f}} + D_{ji}^{\mathrm{fr}}\right), \qquad T_i^{\mathrm{f}} = \max_{j\in \mathrm{FI}(i)} \left(T_j^{\mathrm{r}} + D_{ji}^{\mathrm{rf}}\right).$$

The overall circuit delay can be taken to be the maximum over all rising and falling signal times:

$$D = \max\left\{\max\left\{T_i^{\mathrm{r}}, T_i^{\mathrm{f}}\right\} \mid i \text{ an output gate}\right\}.$$

As in the simple recursion of a single delay and arrival time, the resulting problem can be reformulated as a GGP, provided the delays are generalized posynomials. There can be false paths, but the delay D is a valid upper bound on the true maximum delay of the circuit over all input transitions.

Once we distinguish between rising and falling transitions, with different delays from each gate input to its output, we can just as well assign different input capacitance to each gate input, and a different energy loss for each transition. This makes the bookkeeping and notation more complex, but the problem remains a GGP.

2.2.6. Signal Transition Time. So far we have neglected the effect of input signal slope, or transition time, on the various quantities such as gate power, delay, and input capacitance. These effects are readily handled in a GP formulation. Signal transition time is typically defined as the time for the signal voltage to transition between 10% and 90% of supply voltage. For our purposes, however, the exact definition does not matter; signal transition time can be thought of as an abstract parameter associated with a signal that allows us to better predict the circuit behavior. (This is similar to the scaling parameter, which can be thought of as an abstract parameter that describes the size of a gate.)

We model the delay of gate (during a particular transition) as a function of its scale factor, load capacitance, and its input signal transition times,

$$D_i = f_i\left(x_i, C_i^{\mathrm{L}}, \tau_i^{\mathrm{in}}\right),$$

where τ_i^{in} is the vector of transition times of input signals to gate i. Input transition times affect the energy loss E_i in a transition (and therefore the total average power), through several mechanisms, e.g., short circuit current, so we model it as

$$E_i = g_i\left(x_i, C_i^{\text{L}}, \tau_i^{\text{in}}\right).$$

To propagate the signal transition time forward in the circuit, we model the output transition time of gate i, denoted τ_i^{out}, as

$$\tau_i^{\text{out}} = h_i\left(x_i, C_i^{\text{L}}, \tau_i^{\text{in}}\right).$$

A simpler model, based on the maximum of the transition times of the input signals, can be used. In this simpler model, τ_i^{in} is a scalar.

One simple and commonly used model that includes signal transition time is

$$f_i\left(x_i, C_i^{\text{L}}, \tau_i^{\text{in}}\right) = \bar{D}_i\left(x_i, C_i^{\text{L}}\right) + \kappa_i \tau_i^{\text{in}},$$
$$h_i\left(x_i, C_i^{\text{L}}, \tau_i^{\text{in}}\right) = \zeta_i \bar{D}_i\left(x_i, C_i^{\text{L}}\right),$$

where κ_i and ζ_i are positive constants and $\bar{D}_i$ is the gate delay when the input signals switch instantaneously. (Here we take τ_i^{in} to be a scalar.) In this model, nonzero input signal transition time adds (linearly) to the delay, and the output signal time is proportional to the gate delay.

The functions f_i, g_i, and h_i are monotone increasing functions of the signal transition times, i.e., delay, energy loss, and output signal transition time all increase when any input signal transition time is increased. Roughly speaking, circuit behavior degrades with increasing input signal transition times. This means that we can work with upper bounds on signal transition times and propagate them recursively in the same way we propagate (bounds on) signal arrival times. In such a formulation we associate four quantities with each gate output (i.e., each net): T^{f}, T^{r}, τ^{f}, and τ^{r}. These are interpreted as upper bound (or worst-case) values of rising and falling signal arrival time, and rising and falling transition times, respectively. Provided the functions f_i, g_i, and h_i are generalized posynomial (and monotone increasing in τ_i^{in}), we obtain a GP or GGP formulation of the gate scaling problem.

We mention here a related constraint that is *not* compatible with GP. In *signal integrity constraints*, we require that a particular gate output signal has a *minimum* transition time, i.e., that it cannot rise or fall too quickly. (A rapidly rising or falling signal can disrupt other signals whose wires are routed nearby.) These constraints can usually be dealt with just like minimum path delay constraints, after the initial design. Alternatively, they can be handled by changing the routing or spacing of the wires involved.

2.2.7. Design with a Standard Library. We now show how to apply GP to design with a *standard library*. A typical standard cell library contains a variety of gate types, and for each type, a number of different sizes (typically around 10 for small, common gate types, and three or four for gate types that are less frequently used). The different sizes are generally not obtained by simple scaling of all the gate devices by a common factor, but do follow the same general pattern, i.e., larger gates are able to drive larger loads with smaller delay. For each gate type, a standard library cell provides a table which, for both rising and falling transitions of each input pin, gives the input capacitance, delay, average energy loss per transition, and output transition time, in terms of the size (i.e., scale factor), load capacitance, and input transition time.

Now suppose that the circuit topology is fixed, and it remains to choose the size for each gate from among the choices in the library used. In general this is a combinatorial optimization problem, which is difficult to solve exactly. But the problem can be approached using GP, as we now outline.

The first step is to develop GP compatible models for each gate type in the library, for each input/output transition type (i.e., rising-rising, etc.). This can be done by fitting generalized posynomials to the data given in the library characterization tables. This step involves a few subtleties. The first is avoiding over-fit, i.e., fitting a model with too many parameters to a small data set (i.e., the given tables). The second is the choice of the size parameter (or parameters) for each type of cell. Often the cell name includes a number that is related to its size (e.g., 3NANDX32 might denote a three-input NAND gate with drive strength 32), and these can be used as the scale factor x. But in fact one can choose any values for the scale parameters because they are just abstract labels. Indeed, it is possible to use more than one scale parameter for each cell type, if necessary. After developing these GP models, we have a continuously parameterized family of gates of each type; the gates in the library correspond to some discrete values of the scaling parameter x_i. For each gate type we have generalized posynomial models of area, delay, input capacitance, output signal transition time, average energy lost per transition, and average leakage current, as functions of the scale factor, load capacitance, and input transition time. Using these fitted models, we can solve the sizing problem, which is a GP or GGP. We can then snap the resulting design back to the valid library values, using a simple rounding procedure, or a more sophisticated method such as branch and bound (see Boyd et al. 2004).

In this approach, we start with a combinatorial optimization problem, then form a related continuous problem, solve it, and then snap the resulting continuous design back to the original library. This might seem like a very roundabout way to (approximately) solve the original combinatorial optimization problem. The advantage of forming the intermediate continuous GP or GGP is that such problems can easily take into account the complex interactions among the choices being made on multiple constraints and

performance measures. The continuous problem is solved globally and reliably; its solution, we hope, gives a good starting point for choosing the discrete cell sizes.

2.3. Robust and Multimode Design

In the previous section, we explored more accurate models of delay, power, and other constraints that are compatible with a GP formulation. In this section, we show how GP formulations allow us to take into account process and parameter variation, as well as carry out designs that are meant to be operated in different modes.

2.3.1. Robust Design over Corners. We start with a GP formulation of a gate scaling problem. The various generalized posynomial models such as gate delay, power, and signal transition times are implicitly based on a particular set of process parameters (such as oxide thickness and threshold voltage) and environmental parameters (such as temperature and supply voltage). In a *corner-based robust design* approach, we identify a finite set of *corners* or *scenarios*, each of which consists of a particular combination of process and environmental parameters. For each corner, we develop generalized posynomial models for gate delay, power, and so on. This can be done analytically (e.g., using formulas that predict the variation in gate delay with supply voltage), or by fitting to data obtained from simulations or measurements.

In the simplest version of robust design over corners, we replicate the constraints for each of the corners, using the (slightly) different models for each of the corners. Consider, for example, constraints on delay and power, of the form

$$D(x) \leqslant D^{\max}, \qquad P(x) \leqslant P^{\max}.$$

If we have K corners, we have K different models of the delay and power, say,

$$D^{(1)}(x), \ldots, D^{(K)}(x), \qquad P^{(1)}(x), \ldots, P^{(K)}(x).$$

Here, $D^{(k)}(x)$ and $P^{(k)}(x)$ are the delay and total power dissipation of the circuit in scenario k, as functions of the gate scale factors x. These models of delay and power are different (but, presumably, not radically different) from each other. Now we form the GP

$$\begin{array}{ll} \text{minimize} & A \\ \text{subject to} & D^{(1)}(x) \leqslant D^{\max}, \ldots, D^{(K)}(x) \leqslant D^{\max}, \\ & P^{(1)}(x) \leqslant P^{\max}, \ldots, P^{(K)}(x) \leqslant P^{\max}, \\ & 1 \leqslant x_i, \quad i = 1, \ldots, n. \end{array} \tag{7}$$

The optimal solution of this problem gives a design that satisfies the specification on delay and power *for all corners*, and achieves the minimum area.

Note that if we can identify ahead of time a particular corner that is always "worst" for, say, delay, for all gates and all values of x, then we can just as well carry out an ordinary design using this corner for our delay model. Robust design with replicated constraints, as in (7), is useful in cases where the effect of the variations is not entirely clear when the parameters affect different objectives in different ways (for example, when one parameter change increases leakage power but decreases dynamic power, so its net effect on total power is not obvious).

There are many variations on the basic robust design problem (7). As an example, suppose that the delay D, which depends on the particular corner or scenario, is our objective. We can minimize the worst-case or maximum delay over the corners by taking

$$D^{\mathrm{wc}} = \max\{D^{(1)}(x), \ldots, D^{(K)}(x)\},$$

as the objective to be minimized, subject to a limit on area, and a limit on power for all corners. (D^{wc} is a generalized posynomial if $D^{(k)}$ is.) This is a minimax design: it satisfies the power specification for all corners and achieves the best possible worst-case delay over the corners.

We can also combine the objectives for different corners using a weighted sum. Suppose, for example, that $\pi_1, \ldots, \pi_K$ are positive and sum to one. (We can think of π_k as the probability that corner k occurs.) As objective we can take the weighted sum

$$\bar{D} = \pi_1 D^{(1)}(x) + \cdots + \pi_K D^{(K)}(x),$$

which is the expected or average value of the delay, over the scenarios. More discussion of general robust optimization can be found in Boyd and Vandenberghe (2004, Ch. 6).

Corner-based robust optimization yields designs that gracefully tolerate the process and environment parameter variations that are taken into account in the corners. For combinations of process and environment parameters not included in the list of corners, of course, there is no guarantee (in general). In most cases, it is not expected that the actual parameter variations will be one of the given corners; the method is used as a heuristic for generating designs that work over the K corners, and therefore are likely to work with other, similar combinations of parameter values.

There are some subtleties in forming a multicorner robust optimization problem when worst-case bounds are used in the formulation. Consider, for example, a delay that is formulated using the dynamic programming approach. The basic multicorner approach is to form K separate arrival times at the output of each gate, say,

$$T_i^{(k)}, \quad i = 1, \ldots, n,\ k = 1, \ldots, K,$$

and to propagate the delays separately for each scenario or corner, as in

$$T_i^{(k)} \geqslant \max_{j \in \mathrm{FI}(i)} \left(D_i^k + T_j^{(k)}\right). \tag{8}$$

In this case, $T_i^{(k)}$ is an upper bound on the signal arrival time at the output of gate i, in corner case k.

Another possibility is to have only one set of arrival time bounds, propagated as

$$T_i \geqslant \max_{j \in \mathrm{FI}(i),\, k=1,\ldots,K} \left(D_i^k + T_j\right). \tag{9}$$

In this formulation, T_i can be interpreted as an upper bound on the signal arrival time, valid even when each gate is associated with a different corner. This formulation is, of course, the same as carrying out a standard design using the worst-case delay model for each gate, i.e.,

$$D_i^{\mathrm{wc}} = \max_{k=1,\ldots,K} D^{(k)}.$$

Formulation (8) might be appropriate to model variations in external power supply voltage (but assuming all gates have the same supply voltage). Formulation (9) might be appropriate to model internal variations in power supply (due to IR drop, for example), in which each gate can have a different supply voltage.

Robust design problems such as (7) are typically very large but also very sparse, and so can be solved efficiently despite the large number of constraints. In particular, the running time grows modestly with K, the total number of corners considered. The running time depends on the problem structure but typically grows almost linearly in K.

2.3.2. Multimode Design. In *multimode design* or *multiscenario design*, the goal is to find one set of gate scalings that work well in two or more different modes or scenarios. Mathematically this is very close to corner-based robust design, where we seek a single set of scalings that work well for a set of different parameter values. In robust design, the variations are unintentional; in multimode design, however, they are intentional. In both robust and multimode design, we have K sets of models of gate delay, power, and signal transition time. In robust design, we impose the same constraints for each of the corners; in multimode design, however, we typically impose different constraints for the different scenarios.

We will explain multimode design with a more specific example. Consider a circuit that will be operated (at different times, and intentionally), in two modes: a high performance (fast) mode, and low power (slow) mode. These modes could involve different power supply voltages, different bulk biasing (to change threshold voltages), and different clock frequencies (and therefore different timing requirements). The goal is to find one set of gate scalings that work well in both the fast and slow operating modes. To do this, we start with two sets of models for delay, power, and signal transition times: one for the fast mode, and one for the slow mode. Thus, $D_i^{\mathrm{fast}}(x)$ is the delay of gate i, with scale factors x, when the circuit is operated in the fast mode; $D_i^{\mathrm{slow}}(x)$ is the delay of gate i, with scale factors x, when the circuit is operated in the slow mode. These models can be found using analytical formulas that predict delay and power when supply and threshold voltage vary (say), or they can be found by fitting data obtained from circuit simulation.

In the low power (slow) mode, the power and delay are required to satisfy

$$P^{\mathrm{slow}} \leqslant \bar{P}^{\mathrm{slow}}, \qquad D^{\mathrm{slow}} \leqslant \bar{D}^{\mathrm{slow}}.$$

In the high performance (fast) mode, we must have

$$P^{\mathrm{fast}} \leqslant \bar{P}^{\mathrm{fast}}, \qquad D^{\mathrm{fast}} \leqslant \bar{P}^{\mathrm{fast}}.$$

Here, $\bar{D}^{\mathrm{slow}}$ and $\bar{D}^{\mathrm{fast}}$ are typically different because the circuit operates at a different frequency in the two modes. The power limits, $\bar{P}^{\mathrm{slow}}$ and $\bar{P}^{\mathrm{fast}}$, are also (presumably) different.

Now we can form a simple two-mode design problem by minimizing area (which does not vary with scenario), subject to these constraints:

$$\begin{array}{ll} \text{minimize} & A \\ \text{subject to} & P^{\mathrm{slow}} \leqslant \bar{P}^{\mathrm{slow}}, \quad D^{\mathrm{slow}} \leqslant \bar{D}^{\mathrm{slow}}, \\ & P^{\mathrm{fast}} \leqslant \bar{P}^{\mathrm{fast}}, \quad D^{\mathrm{fast}} \leqslant \bar{D}^{\mathrm{fast}}, \\ & 1 \leqslant x_i, \quad i = 1, \ldots, n. \end{array} \tag{10}$$

In this problem, the variables are $x_1, \ldots, x_n$; the power and area limits are given problem parameters. Like a robust design, this design involves a complex *optimization* interaction between the two modes because the choice of each x_i has consequences for both the slow and fast modes of circuit operation. (There is, of course, no electrical interaction between the two modes.)

This multimode design problem is readily extended in several ways. For example, suppose that π^{slow} (π^{fast}) is the known probability or fraction of time the circuit operates in slow (fast) mode. We can minimize average power (over both modes of operation):

$$\begin{array}{ll} \text{minimize} & \pi^{\mathrm{slow}} P^{\mathrm{slow}} + \pi^{\mathrm{fast}} P^{\mathrm{fast}} \\ \text{subject to} & A \leqslant A^{\max}, \\ & D^{\mathrm{slow}} \leqslant \bar{D}^{\mathrm{slow}}, \quad D^{\mathrm{fast}} \leqslant \bar{D}^{\mathrm{fast}}, \\ & 1 \leqslant x_i, \quad i = 1, \ldots, n. \end{array} \tag{11}$$

In multimode design (as described in this section), we are still only optimizing scaling factors. Each scenario's supply voltage, threshold voltage, clock frequency, and so on, is fixed ahead of time. In §3, however, we will see that these can also be optimized in a multimode design.

2.4. Statistical Design

In statistical design, we take into account statistical variation in the device, process, and environment parameters *for each gate*. In other words, we consider local variations in the model parameters for gate delay, energy, and leakage current, with each gate having its own set of parameter

values drawn from some distribution. (This is in contrast to the framework for robust design over corners, where the gate model parameter variations are global, i.e., the same for all gates.) In the simplest case, which we consider here, the parameter values for each gate are modeled as independent random variables, but more sophisticated models can include correlation between the parameters associated with different gates. Another approach is to include two unknown terms in the parameters of each gate: one is a systematic one, global for the whole circuit (as in robust design over corners), and the other is the local uncertainty we consider here. This leads to a blend of robust design over corners (to handle global parameter variation) and statistical design (to handle local parameter variation). In this section, however, we consider pure statistical design.

The statistical variation in the gate parameter values induces statistical variation in the overall performance objectives such as power and delay, which are described by probability distributions (that depend on the choice of scaling factors, i.e., x). We can modify the basic problem by requiring that the power constraint should hold with some minimum probability (or reliability) such as 95%, and taking as objective the 95% quantile (say) of delay:

$$\begin{array}{ll}\text{minimize} & \mathbf{Q}^{0.95}(\mathbf{D}) \\ \text{subject to} & \mathbf{Q}^{0.95}(\mathbf{P}) \leqslant P^{\max}, \quad A \leqslant A^{\max}, \\ & 1 \leqslant x_i, \quad i = 1, \ldots, n.\end{array} \tag{12}$$

Here, $\mathbf{D}$ and $\mathbf{P}$ are random variables, and $\mathbf{Q}^{0.95}(\mathbf{X})$ denotes the 95% quantile of the random variable $\mathbf{X}$. In the basic statistical design problem formulation (12), we insist that 95% of the circuits meet the power constraint, and we judge a design by the 95% quantile of its delay. These statistical measures are closely related to the (parametric) yield of the circuit, i.e., the fraction of circuits that, when manufactured, meet the performance specifications. Formulation (12) is a stochastic optimization problem.

The statistical analysis and design of digital circuits is an area of growing interest and importance; see, e.g., Agarwal et al. (2003), Bhardwaj et al. (2003), Brambilla and Maffezzoni (2001), Jyu et al. (1993), Orshansky et al. (1999), and Orshansky and Keutzer (2002). This is still an active research area, and no consensus has emerged as to what the best statistical models are. In the following sections, we base our development on some simple models that have been used, to derive (approximate) GP formulations of the statistical design problem.

2.4.1. Statistical Power Constraint. The statistical power constraint $\mathbf{Q}^{0.95}(\mathbf{P}) \leqslant P^{\max}$ is not very difficult to handle, at least approximately, because the power is the sum of the gate powers. Assuming that the parameter variations are independent and the circuit contains a large number of gates (which is always the case in problems of interest), the power $\mathbf{P}$ has a standard deviation that is small relative to its mean $\mathbf{EP}$. Thus, we can use $\mathbf{EP} \leqslant P^{\max}$ as a reasonable approximation of $\mathbf{Q}^{0.95}(\mathbf{P}) \leqslant P^{\max}$. (A more sophisticated approximation can be made using a normal approximation of $\mathbf{P}$.) Because $\mathbf{P} = \sum_{i=1}^{n} \mathbf{P}_i$, our approximation of the statistical power constraint $\mathbf{Q}^{0.95}(\mathbf{P}) \leqslant P^{\max}$ reduces to

$$\sum_{i=1}^{n} \mathbf{EP}_i \leqslant P^{\max}.$$

Thus, we can take into account the effect of statistical variation on total circuit power by replacing the nominal gate power model with a mean gate power model (where the mean is over the parameter variation).

As an example we consider the leakage power of a gate, and variation in the threshold voltage V_{th}, a critical electrical parameter of the devices in a gate. One simple and commonly used model for threshold voltage variation is *Pelgrom's model*, which predicts that the threshold voltage of the devices in a gate has variance

$$\sigma_{V_{\text{th}}}^2 = \bar{\sigma}_{V_{\text{th}}}^2 x^{-1},$$

where $\bar{\sigma}_{V_{\text{th}}}^2$ is the variance for a unit-scaled gate (Pelgrom 1989). This model predicts that larger gates have smaller variation in threshold voltage than smaller ones. We will assume that V_{th} has a Gaussian distribution.

The leakage current for a gate has an exponential dependence on the threshold voltage,

$$I^{\text{leak}} \propto e^{-V_{\text{th}}/V_0},$$

where V_0 is a process-related constant. Because V_{th} has a Gaussian distribution, it follows that I^{leak} has a lognormal distribution. Its mean is, therefore,

$$\mathbf{E}I^{\text{leak}} = I^{\text{leak, nom}} e^{\sigma_{V_{\text{th}}}^2/(2V_0^2)} = I^{\text{leak, nom}} e^{\bar{\sigma}_{V_{\text{th}}}^2/(2V_0^2 x)},$$

where $I^{\text{leak, nom}}$ is the leakage current of the gate when statistical variation is ignored. This formula shows that the mean leakage current is the nominal leakage current multiplied by a factor that is always greater than one, and decreases as the gate scale factor x increases.

The extra term is readily approximated as a generalized posynomial in x, using

$$e^{\bar{\sigma}_{V_{\text{th}}}^2/(2V_0^2 x)} \approx \left(1 + \bar{\sigma}_{V_{\text{th}}}^2/\left(2V_0^2 xa\right)\right)^a,$$

where a is large. (See Boyd et al. 2004 for discussion of generalized posynomial approximation of an exponential.) Thus, to take statistical variation in V_{th} into account in power modeling, we simply use the generalized posynomial (of x),

$$I^{\text{leak, nom}}\left(1 + \bar{\sigma}_{V_{\text{th}}}^2/\left(2V_0^2 xa\right)\right)^a, \tag{13}$$

instead of the constant $I^{\text{leak, nom}}$ in leakage power calculations.

In summary, the statistical power constraint $\mathbf{Q}^{0.95}(\mathbf{P}) \leqslant P^{\max}$ can be (approximately) handled by simply substituting expression (13) into the leakage power expression (3), which results in a generalized posynomial constraint in the scale factor variables.

2.4.2. Statistical Delay Analysis. The effect of statistical variation in the gate delays is far more difficult to analyze than the effect of variation in the gate powers and is what makes the statistical gate sizing problem (12) challenging. Assuming that the variations in gate delays are independent, the delay of the paths, which are sums of gate delays, have reduced variance by a factor on the order of the number of gates on the path (with is typically not large, on the order of 10). Thus, the paths still exhibit substantial statistical variation in delay.

The overall delay of a circuit is the maximum of the path delays. The maximum of a set of random variables, however, behaves very differently from a sum. In particular, the maximum of a set of random variables can have a distribution with a strong right skew and a variance substantially larger than the variance of the individual variables.

We first start with an analysis of gate delay variation induced by variation in V_{th}. Gate delay varies with threshold voltage according to the *alpha-power law model* (see Sakurai and Newton 1990)

$$D \propto \frac{V_{\mathrm{dd}}}{(V_{\mathrm{dd}} - V_{\mathrm{th}})^{\alpha}},$$

where α is a parameter typically between 1.3 and 2. Assuming that the statistical variation in V_{th} is not too large, we have

$$\sigma \approx \left|\frac{\partial D}{\partial V_{\mathrm{th}}}\right| \sigma_{V_{\mathrm{th}}} = \frac{\alpha \bar{\sigma}_{V_{\mathrm{th}}}}{(V_{\mathrm{dd}} - V_{\mathrm{th}})} x^{-1/2} D, \tag{14}$$

where σ is the standard deviation of the gate delay D. This shows that the relative standard deviation of a gate decreases proportionally to the inverse square root of the gate scale factor. We note for future use that the gate delay standard deviation is a generalized posynomial of the scale factors, whenever D is.

The distribution of the overall circuit delay is found from the gate delay distributions, propagated through the function that maps gate delays into overall circuit delay, which is a maximum of a large number of sums. There is no simple analysis or description of this distribution, which, as mentioned above, can have a large right skew. For a fixed design (i.e., choice of x) we can find this distribution by Monte Carlo analysis, but problem (12) is in general a difficult stochastic optimization problem.

Statistical (static) timing analysis of a digital circuit is closely related to the stochastic PERT problem for the associated *stochastic project network*, a well-researched topic (Anklesaria and Drezner 1986, Bowman 1995, Devroye 1979, Hartley and Wortham 1966, Robillard and Trahan 1976, Van Slyke 1963). Indeed, many techniques for statistical timing analysis are based on those developed in the operations research and management science literature, e.g., criticality analysis (Bowman 1995) and distribution bounding (Kleindorfer 1971, Ludwig et al. 2001).

2.4.3. A Heuristic for Statistical Design. In this section, we describe a simple heuristic for the statistical design problem (12), which is compatible with GP and seems to give good results in practice. We start with a (deterministic) model for each gate, as described above. Here, we interpret the generalized posynomial D_i as giving the *mean* gate delay. To this mean gate delay we add a zero-mean random variable with variance σ_i^2, which represents the statistical fluctuation or uncertainty in the gate delay. We assume that σ_i is, like D_i, a generalized posynomial function of the scale factor, load capacitance, and input signal transition times. The ratio σ_i/D_i is a measure of the relative or percentage variation in the gate delay.

Now we can describe the heuristic method. We form a "surrogate delay" model

$$\tilde{D}_i = D_i + \kappa_i \sigma_i,$$

where κ_i are constants (often all the same), typically between 1 and 3, that are used to trade off mean and variance of the overall delay. The extra term $\kappa_i \sigma_i$ adds an extra margin in the gate delay model that scales with increasing gate delay variance. Now we optimize the circuit as usual, using the surrogate delays $\tilde{D}_i$ in place of the (mean) delays D_i. Note that when $\kappa_i = 0$, this method is the same as the standard (nonstatistical) design method. We can analyze the resulting design using Monte Carlo analysis, coupled with static timing analysis. We can then adjust the constants κ_i for optimum performance (for example, estimated yield). For example, designs can be carried out with all κ_i constant, and equal to the values $\kappa_i = 2$, $\kappa_i = 2.5$, and $\kappa_i = 3$; the best of the resulting three designs is taken as the final design.

This method looks simple but is more sophisticated than it appears because the extra margins are added on a gate-by-gate basis and not on a path-by-path basis. Our experience suggests that this heuristic method often yields designs that are far superior to those obtained by ignoring statistical variation. Moreover, the designs seem to be good despite the simplicity of the statistical model (which ignores gate distribution shape, correlations, and so on). In some cases, the method yields a design that is *provably* close to the global optimum of the (difficult) stochastic optimization problem. (See Kim et al. 2004 for more details.)

2.4.4. Ladner-Fischer Adder Statistical Design Example. We illustrate the heuristic method for statistical design on the Ladner-Fischer adder example, with the model parameters given in §2.1.10. The gate delays are taken to be independent and Gaussian. The mean delay model is the same as the delay model used in the basic gate scaling example, and the standard deviation model that follows is given by

$$\sigma_i = 0.15 x_i^{-0.5} D_i.$$

This means that for a minimum size gate, the delay has a relative standard deviation of 15%; as the gate is scaled up,

Table 2. Comparison of nominal and statistical designs.

	Nominal delay	**ED**	$\sigma_{\mathbf{D}}$	$\mathbf{Q}^{0.95}(\mathbf{D})$
Nominal design	45.1	48.9	0.88	50.4
Statistical design	45.7	47.5	0.47	48.2

the relative standard deviation drops scales as $x_i^{-1/2}$. The limit on area is taken as $A^{\max} = 15{,}000$; no limit is imposed on power. Monte Carlo analysis revealed that the heuristic statistical design method gave good results with $\kappa_i = 2$.

The performance of the statistical design, found by Monte Carlo analysis with 5,000 samples, is compared to the nominal design (i.e., the one obtained by ignoring the statistical variation) in Table 2. The robust design has a nominal circuit delay (i.e., a circuit delay ignoring statistical variation) of 45.7, only 1.3% more than the nominal design. When we take statistical variation into account, however, the two designs differ. The 95% circuit delay for the nominal optimal design is 50.4, which is 11.8% more than the nominal circuit delay. For the robust design, the 95% circuit delay is 48.2, which is only 5.6% more than the nominal optimal circuit delay. Thus, the statistical design has reduced the effect of statistical delay variation by a factor of around 2, compared to the nominal optimal design. The standard deviation of the statistical design is also reduced by a factor of around 2, compared to the nominal design. The distributions of the delay for the nominal and statistical designs (estimated by Monte Carlo simulation) are shown in Figure 6.

Some insight into why the statistical design performs better than the nominal design can be found in Figure 7, which shows scatter plots of mean delay versus standard deviation for all 3,214 paths in the Ladner-Fischer adder for the nominal and statistical designs. The problematic paths are the ones at the upper right, which represent paths that are

Figure 6. Distributions of circuit delay for nominal and statistical designs.

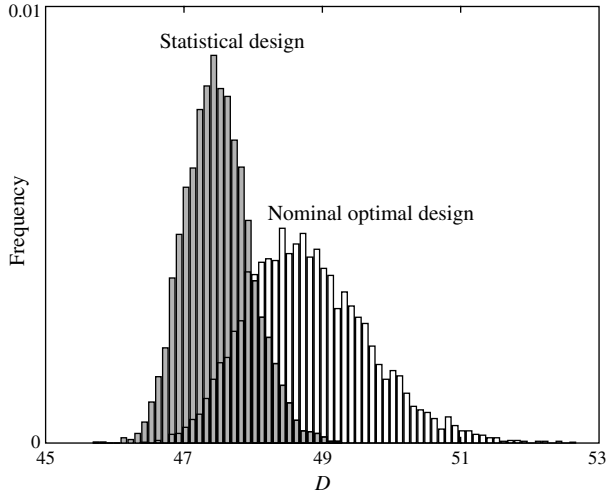
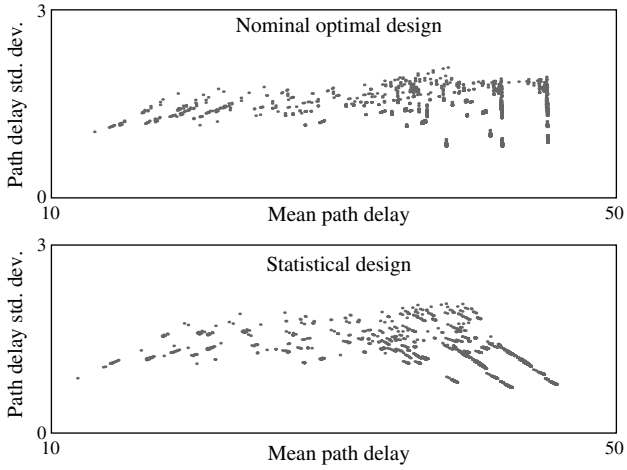


Figure 7. Scatter plots of path delay mean versus path delay standard deviation for the nominal design (top) and statistical design (bottom).



near critical and in addition have large standard deviation. The plots show that in the nominal design, a number of paths with large expected delays have large standard deviation; in the statistical design, however, the variances of the paths with large expected delays are smaller; paths with relatively small expected delays, however, have relatively larger variances.

2.5. Optimization of Multiple Combinational Logic Blocks

We now turn to the larger problem of sizing (the gates in) multiple combinational logic blocks. We start by assuming that the flip-flop registers between the blocks are fixed, and present constant capacitive loads at their inputs, and have fixed drive strength at their outputs. As a result, the performance of each block is locally determined, i.e., its delay, area, and power are independent of the gate scale factors of the other blocks. Indeed, the registers are precisely meant to provide electrical and timing decoupling between the circuits they separate.

For block i, we let D_i denotes its delay, A_i its area, P_i its power, and x_i (the vector of) its gate scale factors. We let $A = A_1 + \cdots + A_K$ denote the total area of the K blocks, and $P = P_1 + \cdots + P_K$ denote the total power. The functions A and P are block separable, i.e., they are sums of functions of the x_i.

Assuming that the clock frequency is fixed, the maximum delay of each block is fixed. The maximum delays for each block can be different, but for simplicity we will assume that each is given by $0.8/f_{\mathrm{clk}}$, i.e., 80% of the clock period. In the simplest formulation, the objective is area:

$$\begin{array}{ll}\text{minimize} & A = A_1 + \cdots + A_K \\ \text{subject to} & D_i \leqslant 0.8/f_{\mathrm{clk}}, \quad i = 1, \ldots, K,\end{array}$$

where the variables are the gate scale factors in each of the K blocks. (We do not show other constraints local to each

block, such as limits on the gate scale factors.) Because the constraints are decoupled, and the objective is separable, this problem can be solved by (separately) minimizing the area of each block, subject to its timing requirement. The same approach works for any other separable objective, e.g., power, or a linear combination of area and power.

If area or power is constrained, we introduce (optimization) coupling between the blocks. Suppose, for example, that the problem is

$$\begin{array}{ll} \text{minimize} & A = A_1 + \cdots + A_K \\ \text{subject to} & D_i \leqslant 0.8/f_{\text{clk}}, \quad i = 1, \ldots, K, \\ & P = P_1 + \cdots + P_K \leqslant P^{\max}. \end{array} \tag{15}$$

This problem is, of course, a large GGP (assuming that the individual block design problems are GGPs). It can be solved directly, as one very large GGP, or iteratively, using a method that carries out K separate block designs in each step.

One such method is a Lagrangian approach, in which the coupled power constraints are dualized. We form the problem

$$\begin{array}{ll} \text{minimize} & A + \lambda P \\ \text{subject to} & D_i \leqslant 0.8/f_{\text{clk}}, \quad i = 1, \ldots, K, \end{array} \tag{16}$$

where λ is a (positive) dual variable, the Lagrange multiplier associated with the total power limit. This problem is block separable and can be solved, for any particular λ, by optimizing each block separately. The iteration consists of adjusting λ so that the power constraint is just satisfied, or equivalently, to maximize the optimal value of the dualized problem (16). (This can be done by bisection because there is only one variable λ.) Because the large GGP problem is convex, after a log transformation, there is no duality gap here, assuming that a constraint qualification holds.

Another interesting approach can be taken when the goal is to obtain a portion of the optimal delay-area trade-off curve for the overall circuit. (This would be the case in the initial exploratory phase of design, when the exact power limit has not been fixed.) For each block, we find several designs on the optimal delay-area curve. We can then fit each of these curves by a generalized posynomial that characterizes the achievable area-power pairs for each block. (The optimal trade-off curve for a GGP can always be arbitrarily well approximated by a generalized posynomial.) Parameterizing (say) optimal power P_k^{opt} as a function of maximum allowed area $A_k^{\max}$, we obtain the GGP

$$\begin{array}{ll} \text{minimize} & A = A_1^{\max} + \cdots + A_K^{\max} \\ \text{subject to} & P = P_1^{\text{opt}} + \cdots + P_K^{\text{opt}} \leqslant P^{\max}, \end{array}$$

where the variables are $A_k^{\max}$, $k = 1, \ldots, K$. This extremely small GGP can be solved for several values of $P^{\max}$ to trace out the optimal area-delay trade-off curve for (15).

The same general approaches can be taken when more than one objective involves coupling between the blocks. For example, we can find the optimal trade-off surface for clock frequency, area, and power, for the whole circuit by first finding the optimal trade-off surface for each block and then solving (several times) a small GGP that combines them.

We mention several extensions and variations on multiblock optimization and design. First, a flip-flop does not completely isolate the circuitry at its input from the circuitry at its output. For example, the *setup time* for a flip-flop, which is the time before the clock that the input must be valid, is a mild function of the capacitive load the flip-flop drives. This constraint couples the timing of one block to the sizing of the input gates in another, but the problem remains a GP. Because the coupling is mild, iterative methods that ignore it when optimizing the individual blocks work well.

Another extension involves optimizing the sizes of the flip-flops as well as the gates in the combinational logic blocks between them. As the size of a flip-flop varies, its capacitive load, drive resistance, and timing parameters vary. This gives a strong coupling between the blocks, but the problem remains a GGP. One simple approach to jointly optimizing the flip-flops and gates is to alternate between fixing the flip-flops and optimizing the gates, and fixing the gates and optimizing the flip-flops.

3. Supply and Threshold Voltage Optimization

So far, we have considered threshold and power supply voltages as fixed parameters, or, in the case of robust, statistical, or multimode formulations, as parameters that vary in given ways beyond our control. In this section, we consider problem formulations in which threshold and power supply voltages are, along with device sizes, *design variables to be optimized*.

Supply and threshold voltage optimization, in the context of low power CMOS circuit design, is an area of active current research; see, e.g., Anis et al. (2003), Chabini et al. (2003), Chen et al. (2001), Kao et al. (2002), Pant et al. (2001), and Srivastava and Sylvester (2004). Many approaches to low power CMOS design have been proposed in the literature, including design with multiple supply and threshold voltages (Chang and Pedram 1997; Jung et al. 2003; Kim et al. 2003a, b; Krishnamurthy and Carley 1997; Marković et al. 2004; Sirichotiyakul et al. 2002; Srivastava and Sylvester 2004; Yeh et al. 2001), multiple threshold CMOS (MTCMOS) (Anis et al. 2002, 2003; Calhoun et al. 2004; Kao and Chandrakasan 2000), variable threshold CMOS (VTCMOS) via adaptive body biasing (Im et al. 2003, Kao et al. 2002, Tschanz et al. 2003, Yang et al. 1997), dynamic threshold CMOS (DTCMOS) (Assaderaghi et al. 1997), joint device sizing and $V_{\text{dd}}/V_{\text{th}}$ assignment (Augsburger and Nikolić 2002a, b; Chen and Sarrafzadeh

2002; Hung et al. 2004; Ketkar and Sapatnekar 2002; Karnik et al. 2002; Liu et al. 2004; Nguyen et al. 2003; Pant et al. 2001), dynamic frequency scaling (Lu et al. 2002), supply voltage scaling (Bellaouar et al. 1998), and transistor stacking (Johnson et al. 2002, Mukhopadhyay et al. 2003).

In this section, we show how GP-based approaches can be combined with the techniques above. In particular, the adaptive body biasing, DTCMOS, supply voltage scaling, and dynamic threshold voltage control problems are related to multimode design.

3.1. Generalized Posynomial Gate Delay Model

To model the delay of a gate as a function of threshold and power supply voltage as well as scale factor, load capacitance, and input signal transition time, we use the alpha-power law model (Sakurai and Newton 1990),

$$D = \frac{V_{\mathrm{dd}}}{(V_{\mathrm{dd}} - V_{\mathrm{th}})^{\alpha}} g(x, C^{\mathrm{L}}, \tau^{\mathrm{in}}), \tag{17}$$

where α is a constant, typically between 1.3 and 2, and g is a generalized posynomial. This gate delay model can be *exactly* handled by GP. We can replace the signal arrival time bound constraint in the dynamic programming formulation,

$$\overline{T}_j + D \leqslant \overline{T}_i,$$

with

$$\overline{T}_j + V_{\mathrm{dd}} z^{-\alpha} g(x, C^{\mathrm{L}}, \tau^{\mathrm{in}}) \leqslant \overline{T}_i, \qquad z + V_{\mathrm{th}} \leqslant V_{\mathrm{dd}}.$$

(Here the new variable z serves as a lower bound on $V_{\mathrm{dd}} - V_{\mathrm{th}}$, which is called the *overdrive voltage*.) These are generalized posynomial constraints and result in a GP formulation without further approximation. (See Boyd et al. 2004, §7.1 for more details on this trick.) We can also approximate the delay, using a truncated power series expansion, to get

$$\widehat{D} = V_{\mathrm{dd}}^{1-\alpha}(1 + V_{\mathrm{th}}/V_{\mathrm{dd}} + \cdots + (V_{\mathrm{th}}/V_{\mathrm{dd}})^5)^{\alpha} g(x, C^{\mathrm{L}}, \tau^{\mathrm{in}})$$

(the right-hand side is a generalized posynomial). The fractional error $(D - \widehat{D})/D$ of this approximation is smaller than 1% over the range of interest, $V_{\mathrm{dd}} \geqslant 2V_{\mathrm{th}}$, $1.3 \leqslant \alpha \leqslant 2$.

Many other gate delay models used in the literature (e.g., Chen et al. 1997, Wei et al. 1999) are compatible with GP. For instance, a gate delay model (Wei et al. 1999) of the form

$$D = \gamma\left(\frac{V_{\mathrm{dd}}}{(V_{\mathrm{dd}} - V_{\mathrm{th}})^{\alpha}} + \frac{\beta}{1.5V_{\mathrm{dd}} - V_{\mathrm{th}}}\right) g(x, C^{\mathrm{L}}, \tau^{\mathrm{in}}),$$

where β, γ are process dependent parameters, can also be exactly handled by GP.

Another GP-compatible delay model is the *velocity saturated model*, which has the form

$$D = \frac{(C^{\mathrm{L}} + C^{\mathrm{int}})V_{\mathrm{dd}}}{2I},$$

where the charging current is

$$I = \kappa \frac{x(V_{\mathrm{dd}} - V_{\mathrm{th}})^2}{V_{\mathrm{dd}} - V_{\mathrm{th}} + E_{\mathrm{c}} L},$$

with process dependent parameters κ, E_{c}, and L. Like the models above, this can be exactly handled by GGP.

3.2. Generalized Posynomial Power Model

The total dynamic power dissipated is given by formula (2), but with (possibly) different supply voltages for the gates,

$$P_{\mathrm{dyn}} = \sum_{i \in \mathrm{PI}} f_i C_i^{\mathrm{L}} V_{\mathrm{dd},\,i}^2 + \sum_{i=1}^{n} f_i \left(C_i^{\mathrm{int}} + C_i^{\mathrm{L}}\right) V_{\mathrm{dd},\,i}^2,$$

where $V_{\mathrm{dd},\,i}$ is the supply voltage of gate i. (For i corresponding to a primary input, $V_{\mathrm{dd},\,i}$ is the supply voltage of the flip-flop that drives the logic block.) This is evidently a posynomial of the device sizes and supply voltages.

The average static power is given by formula (3), but with (possibly) different supply voltages for each gate,

$$P_{\mathrm{stat}} = \sum_{i=1}^{n} \bar{I}_i^{\mathrm{leak}} x_i V_{\mathrm{dd},\,i}.$$

The average leakage current $\bar{I}_i^{\mathrm{leak}}$ is a function of both $V_{\mathrm{dd},\,i}$ and $V_{\mathrm{th},\,i}$. One standard model for the variation of leakage current with supply and threshold voltage is

$$\bar{I}_i^{\mathrm{leak}} \propto e^{-(V_{\mathrm{th},\,i} - \gamma_{\mathrm{D}} V_{\mathrm{dd},\,i})/V_0},$$

where γ_{D} is a constant, typically around 0.06, and V_0 is a constant, typically around 0.04 at room temperature. For more on leakage current modeling, see Weste and Harris (2004) and Marković et al. (2004), or the survey paper by Roy et al. (2003).

The next step is to develop a generalized posynomial approximation of the function

$$V_{\mathrm{dd}} e^{-(V_{\mathrm{th}} - \gamma_{\mathrm{D}} V_{\mathrm{dd}})/V_0} = V_{\mathrm{dd}} e^{\gamma_{\mathrm{D}} V_{\mathrm{dd}}/V_0} e^{-V_{\mathrm{th}}/V_0},$$

which can be done by separately approximating the function $V_{\mathrm{dd}} e^{\gamma_{\mathrm{D}} V_{\mathrm{dd}}/V_0}$ (of V_{dd}) and the function e^{-V_{th}/V_0} (of V_{th}).

The first function, $V_{\mathrm{dd}} e^{\gamma_{\mathrm{D}} V_{\mathrm{dd}}/V_0}$, can be very well modeled by a generalized posynomial because its logarithm is a convex function of $\log V_{\mathrm{dd}}$ (see Boyd et al. 2004). For example, the generalized posynomial approximation

$$V_{\mathrm{dd}}(1 + (\gamma_{\mathrm{D}}/V_0)V_{\mathrm{dd}}/500)^{500}$$

gives a relative error smaller than 1% over the range $0.8 \leqslant V_{\mathrm{dd}} \leqslant 2$ for $\gamma_{\mathrm{D}} = 0.06$ and $V_0 = 0.039$.

The second function, e^{-V_{th}/V_0}, *cannot* be well modeled by a generalized posynomial, at least over a large range, because its logarithm is not a convex function of $\log V_{\text{th}}$. Fortunately, an approximation that is accurate when e^{-V_{th}/V_0} is large will suffice, because when it is small, the dynamic power will dominate the total power. This can be done using a monomial approximation over the range where V_{th} is near its minimum value. For example, the monomial approximation

$$e^{-V_{\text{th}}/0.0039} \approx 0.002 V_{\text{th}}^{-6.16} \tag{18}$$

is accurate for V_{th} near and above 0.2.

Putting these approximations together, we get the leakage power model

$$P_{\text{stat}} = \sum_{i=1}^{n} a_i V_{\text{dd},i}(1 + (\gamma_{\text{D}}/V_0)V_{\text{dd},i}/500)^{500} V_{\text{dd},i} V_{\text{th},i}^{-6.16} x_i,$$

where a_i is a constant that depends only on process parameters and the particular gate topology. This is a posynomial of the gate sizes, threshold voltages, and supply voltages. (This model for static power is not accurate when the static power is small.) The total power, i.e., the dynamic plus leakage power, is thus a generalized posynomial.

As a specific example, consider a total power expression of the form

$$P = V_{\text{dd}}^2 + 30 V_{\text{dd}} e^{-(V_{\text{th}} - 0.06 V_{\text{dd}})/0.039}, \tag{19}$$

with the region of interest $1.0 \leqslant V_{\text{dd}} \leqslant 2.0$, $0.2 \leqslant V_{\text{th}} \leqslant 0.4$. This function is shown at the top of Figure 8. Over this region, for each fixed power supply voltage, the leakage power increases around 170 times as V_{th} varies from the maximum value to the minimum. The corresponding leakage power approximation is

$$\hat{P} = V_{\text{dd}}^2 + 0.06 V_{\text{dd}}(1 + 0.0031 V_{\text{dd}})^{500}(V_{\text{th}}/0.24)^{-6.16}. \tag{20}$$

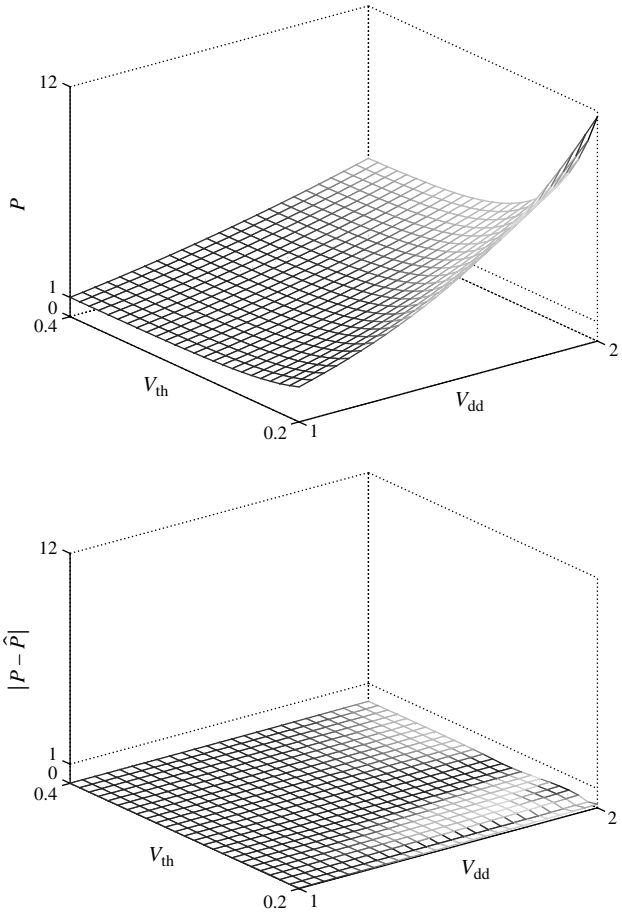
The difference between P and $\hat{P}$ is shown at the bottom of Figure 8. The associated fractional error $(\hat{P} - P)/P$ is below 2.7% over the entire region of interest.

3.3. Joint Optimization

Now we consider the case when gate supply voltages $V_{\text{dd},i}$, threshold voltages $V_{\text{th},i}$, and scale factors x_i are design variables to be chosen. Using the generalized posynomial models of gate delay and power derived above, we obtain the GGP

$$\begin{array}{ll} \text{minimize} & D \\ \text{subject to} & P \leqslant P^{\max}, \quad A \leqslant A^{\max}, \\ & V_{\text{th}}^{\min} \leqslant V_{\text{th},i} \leqslant V_{\text{th}}^{\max}, \quad i = 1, \ldots, n, \\ & V_{\text{dd}}^{\min} \leqslant V_{\text{dd},i} \leqslant V_{\text{dd}}^{\max}, \quad i = 1, \ldots, n. \end{array} \tag{21}$$

Figure 8. Approximation of the power function P as it is given in (19) by the generalized posynomial $\hat{P}$ given in (20). Total power P (top) and the associated absolute approximation error $|P - \hat{P}|$ (bottom).



Several constraints must be added to this problem to make it practical. The most obvious one is that only a handful of allowed supply and threshold voltages are used in practice. This addition makes the problem (21) a mixed-integer GGP. As with the gate scaling problem with discrete allowed scale factors, we can use the snapping heuristics in Boyd et al. (2004, §7.3) to approximately solve this mixed-integer GGP. Other constraints, which we describe below, limit the supply voltages of the gates depending on their fan-out gates. We note that even when these constraints are not added, problem (21) is useful, because it provides a lower bound or limit of performance on the delay that can be achieved in practice.

One extension of problem (21) is *gate clustering*, where the gates are partitioned into clusters, with a single pair of supply and threshold voltages for each cluster. This can be expressed using the monomial equality constraints

$$V_{\text{th},i} = V_{\text{th},j}, \qquad V_{\text{dd},i} = V_{\text{dd},j},$$

whenever gates i and j belong to the same cluster. Design with multiple supply voltages inside a combinational logic

block requires level conversion circuits (Ishihara et al. 2004, Kulkarni and Sylvester 2004). When there is only one cluster, i.e., all devices have the same supply and threshold voltages, then the problem reduces to optimal gate scaling, along with (common) supply and threshold voltage selection. (This problem has been studied by many researchers; see, e.g., Marković et al. 2004.)

Special asynchronous level conversion circuitry is needed whenever a gate drives another gate with larger supply voltage. This can be avoided by imposing the monomial inequality constraints

$$V_{\mathrm{dd},i} \geqslant V_{\mathrm{dd},j} \quad \text{for all } j \in \mathrm{FO}(i),$$

which require that all gates only drive gates with smaller (or equal) supply voltage. This was proposed by Usami and Horowitz (1995), who called it *clustered voltage scaling*. We can also add a regularization term of the form

$$\lambda \sum_{i=1}^{n} \sum_{j \in \mathrm{FO}(i)} \frac{V_{\mathrm{dd},i}}{V_{\mathrm{dd},j}}$$

to the objective, which acts as a heuristic for enforcing a small number of supply voltage changes (see, e.g., Boyd and Vandenberghe 2004, Ch. 6).

Multimode design is particularly effective when gate scalings, and supply and threshold voltage are jointly optimized, because supply voltage can be (intentionally) varied during operation of the circuit, and in some cases, so can the threshold voltage. Suppose, for example, that all gates have the same V_{dd} and V_{th}, which, however, can be changed with the particular operating mode or scenario. We then design one set of gate scalings and different supply voltages $V_{\mathrm{dd}}^{(k)}$ and threshold voltages $V_{\mathrm{th}}^{(k)}$ for the scenarios $k = 1, \ldots, K$. This was dubbed *dynamic threshold voltage* CMOS (DTCMOS) in Sirisantana and Roy (2004). The threshold voltage (common for all gates) is adapted to suit the operating state of the circuit through bulk (body or substrate) biasing. Another related topic is *supply voltage scaling*, in which the supply voltage (common for all gates) is adapted to the circuit performance requirement. Our point here is that such problems can be formulated and solved as GGPs.

Multiple threshold voltage design can be carried out via multiple doping concentrations, multiple channel lengths, multiple oxide thicknesses, and various combinations (Lee et al. 2004, Sirisantana and Roy 2004). (Recently, a multiple oxide thickness and dual supply voltage device approach has become popular in e-DRAMs to achieve high performance and low leakage power (Takahashi et al. 2000).) As with joint device sizing and supply and threshold voltage optimization, GP-based design can be carried out with multiple doping concentrations, channel lengths and oxide thicknesses, because models of gate delay as a function of doping concentration, channel length, and oxide thickness, as well as the device width, supply, and threshold voltage, can be well approximated as generalized posynomials. The models described in Bowman et al. (2001), Lindert et al. (1999), Yang et al. (2000), and Sirisantana and Roy (2004), for example, are all (approximately) compatible with GP-based design.

3.4. Examples

We consider the 32-bit Ladner-Fischer adder, where each gate has three design variables: scale factor x_i, threshold voltage $V_{\mathrm{th},i}$, and supply voltage $V_{\mathrm{dd},i}$. We use the same area, delay, and power models used in the simple gate scaling example, except that the drive resistance and average leakage current (for a unit scaled gate) vary with supply and threshold voltage as

$$\bar{R}_i = 0.3019 \frac{V_{\mathrm{dd}}}{(V_{\mathrm{dd},i} - V_{\mathrm{th},i})^{1.3}},$$

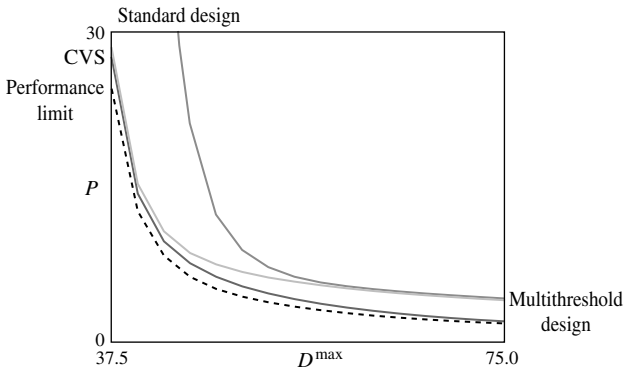
$$\bar{I}_i^{\mathrm{leak}} = \bar{I}_i^{\mathrm{leak,T}} e^{(V_{\mathrm{th},i} - 0.06 V_{\mathrm{dd},i})/0.039} / 0.00898,$$

where $\bar{I}_i^{\mathrm{leak,T}}$ is the average leakage current of gate i with unit scale factor, for the typical supply and threshold voltage, $V_{\mathrm{dd},i} = 1$ and $V_{\mathrm{th},i} = 0.3$. With these typical values, the delay and power models used here coincide with those used in the simple gate scaling example in §2.1.10. The activity frequencies of all gates and primary inputs are taken to be $f_i = 0.05/D^{\max}$.

Figure 9 compares the optimal power-delay trade-off curves for four designs, subject to an area limit $A^{\max} = 15{,}000$, and scale factor limit $x_i \geqslant 1$. The designs are described below.

- *Standard design.* The supply and threshold voltages are fixed, equal to the typical values $V_{\mathrm{dd},i} = 1.0$ and $V_{\mathrm{th},i} = 0.3$. (This is identical to the simple example given in §2.1.10.)
- *Multithreshold design.* The supply voltage is fixed at typical, $V_{\mathrm{dd},i} = 1$, and each gate has one of three possible threshold voltages, $V_{\mathrm{th},i} \in \{0.2, 0.3, 0.4\}$. (We solve the resulting mixed-integer GP using a simple snapping method.)

Figure 9. Power-delay trade-off curves for the 32-bit Ladner-Fischer adder circuit.



• *Clustered voltage scaling* (*CVS*). Here we have $V_{\mathrm{dd},i} \in \{0.6, 1.0\}$, $V_{\mathrm{th},i} \in \{0.2, 0.3, 0.4\}$, with the constraint that a gate cannot drive one with larger supply voltage.

• *Performance limit.* We solve the general problem, with $0.6 \leqslant V_{\mathrm{dd},i} \leqslant 1.0$, $0.2 \leqslant V_{\mathrm{th},i} \leqslant 0.4$. This design is not practical but serves as bound on the best achievable performance for design with supply and threshold voltages within the limits.

Figure 9 clearly shows that the multithreshold designs outperform the standard designs, which use only the typical threshold voltage, especially for small required delay. The CVS designs outperform the standard designs for all delay specifications. Moreover, the CVS designs, which include the requirement that gates cannot drive other gates with higher supply voltage, are not far from the lower bounds given by the solution of the general problem. This suggests that CVS is a rather good practical approach.

Figure 10 shows the change of the distribution of gate threshold voltages in the multithreshold designs, as the delay specification $D^{\max}$ varies. As $D^{\max}$ increases, the design uses more high threshold gates and fewer low threshold gates, which reduces leakage power. For small delay, however, many of the gates have low threshold voltage. One interesting observation is that over the whole range of delay specification, many gates use high threshold voltage. These are gates off the critical path; the higher threshold voltage reduces total leakage power.

Figure 11 shows the change of the distribution of gate supply voltages for the CVS designs, as the delay specification $D^{\max}$ varies. As $D^{\max}$ increases, more gates use low V_{dd}, which reduces power at the cost of increased delay. For $D^{\max} = 37.5$, about 60% of the gates have high supply voltage; for $D^{\max} \geqslant 55$, most gates have low supply voltage. The plot reveals three distinct regimes. For small delay, all critical path gates are assigned high supply voltage. For medium delays, some of the critical path gates are assigned low V_{dd}. For large delay, low power designs, all gates are assigned low V_{dd}.

Figure 10. Distribution of gate threshold voltages for multithreshold designs.

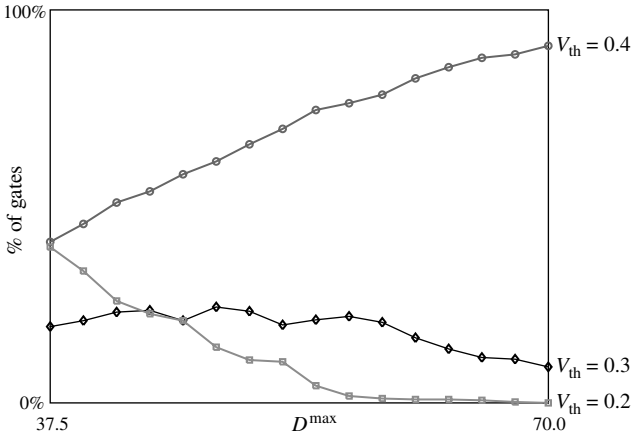
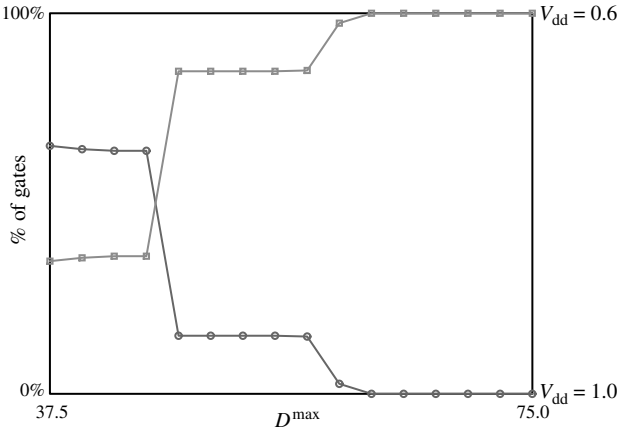


Figure 11. Distribution of gate supply voltages for CVS designs.



3.4.1. Multimode Design Example. In this section, we describe a multimode design example, with a single set of gate scalings, but different supply voltage $V_{\mathrm{dd}}^{(k)}$ and threshold voltage $V_{\mathrm{th}}^{(k)}$ for each of 10 scenarios. (The supply and threshold voltages are common to all gates.) These scenarios correspond to clock frequencies

$$f_{\mathrm{clk}}^{(1)} = 1/50,\ f_{\mathrm{clk}}^{(2)} = 0.8/50,\ \ldots,\ f_{\mathrm{clk}}^{(10)} = (0.8)^9/50.$$

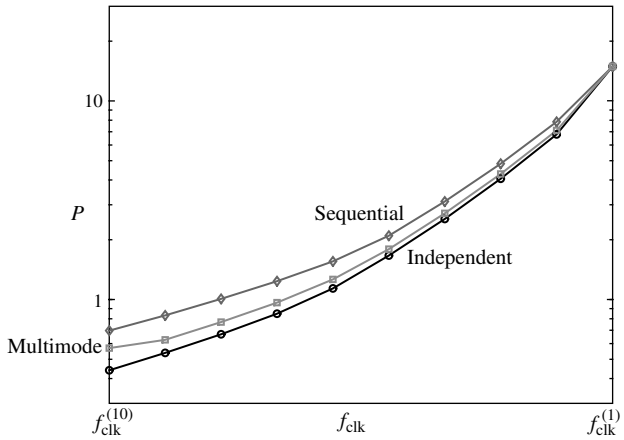
(Recall that the time unit is τ, the delay of a unit scaled unloaded inverter. For $\tau = 15$ ps, these frequencies correspond to $f^{(1)} = 1.33$ GHz, $f^{(20)} = 178.5$ MHz.) We require that the circuit delay $D^{(k)}$ in mode k must not exceed 80% of the clock period $1/f_{\mathrm{clk}}^{(k)}$. Thus, we have the constraints

$$D^{(k)} \leqslant 0.8/f^{(k)} = 50 \cdot 0.8^{(2-k)}, \quad k = 1, \ldots, 10.$$

We use the area constraint $A^{\max} = 15{,}000$, and take average power (over the 10 scenarios) as the objective. The resulting GP has on the order of 10^4 variables and 10^5 constraints, and is solved by a generic GP solver that exploits sparsity in a few minutes.

Figure 12 shows the power versus clock frequency for the multimode design. For comparison, two other plots are shown. One shows independent designs carried out (separately) for the 10 clock frequencies. The gap between these individually optimized designs and the multimode design shows the cost of insisting on one set of gate scalings to be used for all clock frequencies. The third curve shows a sequential design approach. Here we first design the circuit for the fastest clock frequency $f^{(1)}$ and then optimally choose $V_{\mathrm{dd}}^{(i)}$ and $V_{\mathrm{th}}^{(i)}$ for each of the other nine clock frequencies, with the gate scalings fixed. We call this the sequential design approach because the gate scalings are first designed for one clock frequency, and then the supply and threshold voltage is chosen as a second step. For the highest clock frequency $f^{(1)}$, the sequential and independent designs are the same, and give a negligible improvement in power over the multimode design. At all other clock frequencies, the multimode design gives lower power than the sequential design and is not too far from the independent design.

Figure 12. Power P versus clock frequency f_{clk} for multimode design, sequential design, and independent designs.



Note. Both axes are logarithmic.

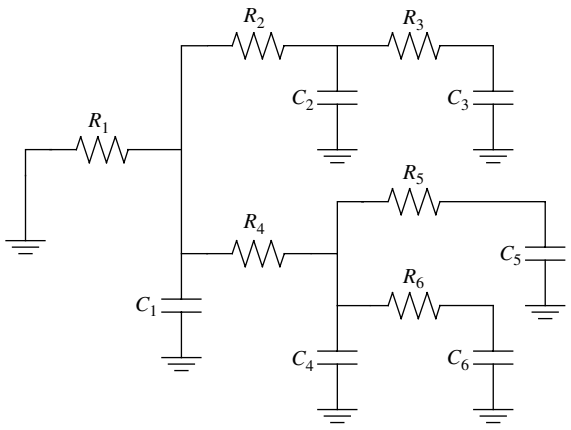
4. Wire Sizing and Gate Design

In this section, we consider three closely related topics. The first is wire sizing, in which the widths of wire segments are chosen. The second is gate design, where we choose the widths of the devices inside a gate, given specifications and objectives for the gate. The last topic is custom design, which combines gate scaling and gate design by optimizing the widths of the individual devices in all gates in a combinational logic block.

4.1. RC Tree Optimization

We start with the problem of optimizing an RC tree, which is the basis for wire sizing, gate design, and custom design. We consider an RC tree, an example of which is shown in Figure 13. An RC tree consists of a resistive tree network, with grounded root, and a capacitance to ground at each node. We label the nodes (not including ground) $1, \ldots, N$. Node i has a capacitance C_i to ground, and a resistance R_i that connects node i to its parent node. We use the term "upstream" to mean toward the root of the tree, and "downstream" to mean toward the leaves of the tree. We will see that an RC tree can serve as a model for an interconnect network, or as a model of the internal dynamics of a gate, that takes into account the nonzero resistance of wire segments and parasitic capacitance.

Figure 13. Example of an RC tree.



4.1.1. Elmore Delay. Let $v_i(t)$ denote the voltage at node i. Suppose that the capacitors in the tree are all initialized as $v_i(0) = 1$. Then, it can be shown that each voltage monotonically decreases to zero as $t \to \infty$. The *Elmore delay* at node i is defined as

$$D_i = \int_0^\infty v_i(t)\,dt,$$

the area under the voltage curve, when the voltages are initialized as $v_i(0) = 1$ (see Elmore 1948). The Elmore delay D_i is a reasonable measure of the time it takes the voltage at node i to decay, from an initial value of one.

The Elmore delay to a node can also be interpreted as a measure of delay during a rising transition. Suppose that the root of the tree is driven by a voltage source that switches from 0 to 1 at $t = 0$, and the voltages are all initialized at 0. Then, each $v_i(t)$ monotonically increases to 1 as $t \to \infty$, and we have

$$D_i = \int_0^\infty (1 - v_i(t))\,dt,$$

i.e., the Elmore delay is the area between the voltage response and the asymptotic value 1.

We define the (critical, or worst-case) Elmore delay D of the whole RC tree as the maximum Elmore delay to any of its nodes. (We will see that the maximum always occurs at one of the leaf nodes.)

There is an analytic expression for the Elmore delay D_i to node i. Let C_i^{tot} denote the total capacitance downstream from node i (including C_i). Let $d_i = R_i C_i^{\text{tot}}$. Then, the Elmore delay can be expressed as

$$D_i = \sum_{j \in \text{Path}(i)} d_j, \tag{22}$$

where the sum is over the unique path from node i to the root. (This shows that d_i can be interpreted as the "local" Elmore delay across resistor R_i; the total Elmore delay is the sum of these local delays along the path to the root.)

Equation (22) is easily understood. When all initial voltages are one, C_i^{tot} is the total charge stored on capacitors downstream from node i. Because this charge must flow through R_i, we see that $R_i C_i^{\text{tot}}$ is the integral of the voltage that will appear across R_i as the RC tree discharges. The sum in (22) follows because the voltage at node i is the sum of the voltages across the resistors on the path from the root to the node.

As an example, the Elmore delay to node 3 in the RC tree in Figure 13 is given by

$$\begin{aligned} D_3 &= R_1 C_1^{\text{tot}} + R_2 C_2^{\text{tot}} + R_3 C_3^{\text{tot}} \\ &= R_1(C_1 + \cdots + C_6) + R_2(C_2 + C_3) + R_3 C_3. \end{aligned}$$

For this example, the Elmore delay is $D = \max\{D_3, D_5, D_6\}$.

A similar analysis can be carried out when the initial voltages are not all equal to 1 (or 0, when the root of the tree is driven by a unit voltage source). The total initial charge stored on the capacitors downstream from node j is

$$Q_j^{\text{tot}} = \sum_{k \text{ downstream from } j} C_k v_k(0).$$

(This reduces to C_j^{tot} when the initial voltages are all one.) The charge Q_j^{tot} must flow through R_j, so the integral of the voltage across R_j is $R_j Q_j^{\text{tot}}$. It follows that

$$\int_0^\infty v_i(t)dt = \sum_{j \in \text{Path}(i)} R_j Q_j^{\text{tot}}. \tag{23}$$

It can be shown that when the initial voltages are nonnegative, the voltages are always nonnegative, in which case the integral gives a reasonable measure of the delay. Formula (23) is just like the formula for the Elmore delay, except that the initial voltages scale the capacitances. This modified version of the Elmore delay can be used when the initial voltages are not all equal to one.

There is a large literature on Elmore delay and related topics; see, e.g., Alpert et al. (2001a), Kashyap et al. (2004), Kahng and Muddu (1997), Gupta et al. (1997), Fishburn and Schevon (1995), and Horowitz (1984). Rubenstein et al. (1983) published the simple closed-form formula described above for computing the mean of the impulse response of RC interconnect trees. A general technique to compute higher-order moments was discovered a few years later in Pillage and Rohrer (1990). In particular, the authors showed how these moments can be used to approximate the poles of the circuit, which allows the time domain waveform to be computed under arbitrary inputs.

4.1.2. Energy Loss. Energy loss is another important quantity associated with an RC tree. When an RC tree with grounded root is discharged, the total energy lost in the resistors is the total initial energy stored in the capacitors,

$$E = \sum_{i=1}^N C_i v_i(0)^2/2,$$

which reduces to the simple expression $(C_1 + \cdots + C_N)/2$ when the initial voltages are one. When the root is driven by a unit voltage source, the total energy loss is

$$E = \sum_{i=1}^N C_i (v_i(0) - 1)^2/2,$$

which also reduces to $(C_1 + \cdots + C_N)/2$ when the initial voltages are zero.

4.1.3. Elmore Delay Optimization. Now suppose that the resistances and capacitances in an RC tree are generalized posynomial functions of some optimization variables x. The Elmore delay to any node is a sum of products of resistances and capacitances, and therefore is a generalized posynomial. The overall worst-case Elmore delay is the maximum of these over the leaf nodes, and so is also a generalized posynomial of the variables. The same observations apply if the initial voltages are not all equal to one (but nonnegative): the modified Elmore delay to any node and the maximum modified Elmore delay are generalized posynomials. The energy loss is also a generalized posynomial. We can minimize the worst-case Elmore delay, subject to a maximum energy loss E, and other generalized posynomial constraints, via GGP:

$$\begin{array}{ll} \text{minimize} & D \\ \text{subject to} & E \leqslant E^{\max}, \\ & f_i(x) \leqslant 0, \quad i = 1, \ldots, m. \end{array} \tag{24}$$

4.1.4. Dynamic Programming Formulation. We can use a recursive formulation to obtain a sparse form of the Elmore delay minimization problem (24). The downstream capacitance satisfies the recursion

$$C_i^{\text{tot}} = C_i + \sum_{j \in \text{Child}(i)} C_j^{\text{tot}},$$

where $\text{Child}(i)$ denotes the children of node i. The Elmore delays can be computed through the recursion

$$D_i = D_{\mathbf{Par}(i)} + R_i C_i^{\text{tot}},$$

where $\mathbf{Par}(i)$ denotes the (unique) parent of node i. Relaxing these to inequalities we obtain the equivalent GP:

$$\begin{array}{ll} \text{minimize} & s \\ \text{subject to} & s \geqslant D_i \quad \text{for } i \text{ an output gate}, \\ & C_i^{\text{tot}} \geqslant \sum_{j \in \text{Child}(i)} C_j^{\text{tot}} + C_i, \quad i = 1, \ldots, n, \\ & D_i \geqslant D_{\mathbf{Par}(k)} + R_i C_i^{\text{tot}}, \quad i = 1, \ldots, n, \\ & E \leqslant E^{\max}, \\ & f_i(x) \leqslant 0, \quad i = 1, \ldots, m. \end{array} \tag{25}$$

This problem is sparse (except for the energy loss constraint), and so can be very efficiently solved. Using this formulation, we can optimize a tree with more than tens of thousands of nodes.

4.1.5. Second Central Moment Optimization. Suppose that the initial capacitor voltages are zero, and a one-volt source is applied to the root of the tree. The resulting voltage at node i, $v_i(t)$ increases from zero to one as $t \to \infty$. Therefore, it can be interpreted as the cumulative distribution of a nonnegative random variable, with density function $h_i(t) = dv_i(t)/dt$. The function h_i is the *impulse response* at node i, i.e., the voltage at node i when a voltage unit impulse is applied to the root of the RC tree at $t = 0$.

The impulse response is positive and has integral one. The Elmore delay D_i at node i is the mean of the associated random variable,

$$D_i = \int_0^\infty t h_i(t)\,dt,$$

i.e., the first moment of the impulse response at node i. This has a natural interpretation as the delay to node i, as discussed above.

We can also work with the variance of the associated random variable, i.e., the second central moment of the impulse response,

$$\sigma_i^2 = \int_0^\infty t^2 h_i(t)\,dt - \left(\int_0^\infty t h_i(t)\,dt\right)^2 = \int_0^\infty t^2 h_i(t)\,dt - D_i^2.$$

The square root of the second central moment, i.e., the standard deviation σ_i of the associated random variable, has a natural interpretation as the transition time or rise time of the signal at node i; see, e.g., Kashyap et al. (2004) and Lin and Pileggi (2001).

Recently, Lin and Pileggi (2001) have shown that the second central moment σ_i^2 is, like the first moment D_i, a posynomial function of the resistances and capacitances in the RC tree. We can express the second central moments via the recursion

$$\sigma_i^2 = \sigma_{\mathbf{Par}(i)}^2 + 2R_i \sum_{j\in \mathrm{Child}(i)} C_j \sum_{k\in \mathrm{Path}(i,\,j)} d_k,$$

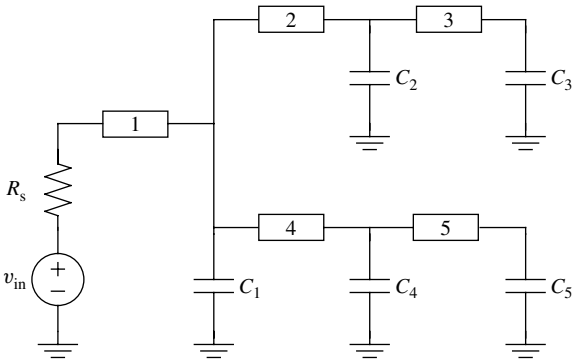
with the starting condition $\sigma_0^2 = 0$ for the root node. Here, $\mathrm{Path}(i, j)$ denotes the set of nodes on the path from i to j excluding i. (See Lin and Pileggi 2001 for the derivation of this recursion.) This recursion shows that σ_i^2 is a posynomial of the resistances and capacitances, so σ_i, which is a measure of signal rise time, is a generalized posynomial of the resistances and capacitances.

As a result, we can add maximum allowable rise time constraints to the RC tree optimization problem (24), which remains a GGP. Another option is to use a more refined measure of the delay to node i, of the form $\lambda_1\mu_i + \lambda_2\sigma_i$, where λ_i are positive constants. This expression is a generalized posynomial of the resistances and capacitances, so the RC tree optimization problem (24), with this model for delay, is still a GGP.

4.2. Wire Sizing

One application of RC tree optimization is interconnect wire sizing, i.e., the problem of determining the widths $w_1, \ldots, w_N$ of N wire segments in an interconnect network. The interconnect network forms a tree; its root is driven by the input signal, which is modeled as a voltage source and a series resistance. Each wire segment has a given capacitive load C_i connected to it. An example of a simple interconnect network is shown in Figure 14, where an ideal voltage source in series with a resistor drives a tree of five wire segments (shown as boxes labeled $1, \ldots, 5$), with capacitive loads $C_1, \ldots, C_5$.

Figure 14. An interconnect network consisting of an input (the voltage source and resistor) driving a tree of five wire segments (shown as boxes labeled $1, \ldots, 5$) and capacitive loads $C_1, \ldots, C_5$.



4.2.1. Wire Segment RC Model. We will use a simple π model for each wire segment, as shown in Figure 15. The wire resistance and capacitance are given by

$$R_i = \alpha_i \frac{l_i}{w_i}, \qquad \bar{C}_i = \beta_i l_i w_i + \gamma_i l_i,$$

where l_i and w_i are the length and width of the wire segment, and α_i, β_i, and γ_i are positive constants (that depend on the physical properties of the routing layer of the wire segment). Substituting this model for the wire segments, the interconnect network becomes an RC tree, where both the resistances and the capacitances are posynomial functions of the wire segment widths w_i.

4.2.2. Wire Sizing Problem. Now we can formulate the wire sizing problem, i.e., the problem of choosing the wire segment widths $w_1, \ldots, w_N$. We impose lower and upper bounds on the wire widths,

$$w_i^{\min} \leqslant w_i \leqslant w_i^{\max},$$

as well as a limit on the total wire area,

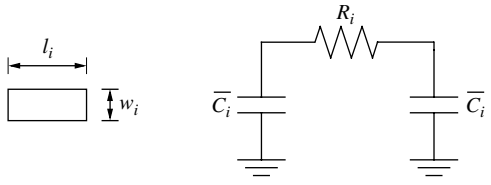
$$l_1 w_1 + \cdots + l_N w_N \leqslant A^{\max}.$$

Taking Elmore delay as objective, we obtain the problem

$$\begin{aligned} &\text{minimize} && D \\ &\text{subject to} && w_i^{\min} \leqslant w_i \leqslant w_i^{\max}, \quad i = 1, \ldots, N, \\ &&& l_1 w_1 + \cdots + l_N w_N \leqslant A^{\max}, \end{aligned} \tag{26}$$

with optimization variables $w_1, \ldots, w_N$. This is a GGP.

Figure 15. Wire segment with length l_i and width w_i (left) and its π model (right).



This formulation can be extended to use more accurate models of wire segment resistance and capacitance, as long as they are generalized posynomials of the wire widths. In addition, other GP-compatible constraints can be added, e.g., a limit on the energy loss, or signal rise times. Wire sizing using Elmore delay goes back to Fishburn and Dunlop (1985); for some more recent work on wire (and device) sizing via Elmore delay, see, e.g., Shyu et al. (1988), Sapatnekar et al. (1993), and Sapatnekar (1996). The state of the art in interconnect wire sizing is well summarized in the survey papers Cong et al. (1996), Sylvester and Hu (2001), and Ho et al. (2001). The survey paper by Cong et al. (1996) also summarizes many interconnect layout issues.

When the wire widths are restricted to take values in some discrete set, the wire sizing problem (26) becomes a mixed-integer GGP. As with the gate scaling problem with discrete scale factors, the snapping heuristics in Boyd et al. (2004, §7.3) can be used to find an approximate solution. The problem can also be solved exactly, using a dynamic-programming-based algorithm, which, however, has a long worst-case running time, $O(N^r)$, where r is the number of discrete wire sizes (Cong and Leung 1995).

4.2.3. Wire Sizing Example. As a simple example, we consider the interconnect network shown in Figure 14, with wire parameter values

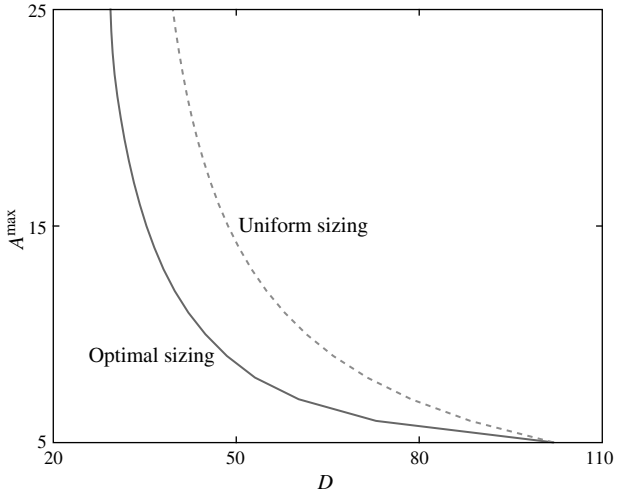
$$l_i = 1, \quad \alpha_i = 1, \quad \beta_i = 1, \quad \gamma_i = 1,$$

load capacitances

$$C_1 = 10, \quad C_2 = 10, \quad C_3 = 10, \quad C_4 = 10, \quad C_5 = 10,$$

and source resistance $R_s = 0.1$. The wire size limits are $w_i^{\min} = 1$ and $w_i^{\max} = 10$. Figure 16 shows the optimal trade-off curve of minimum Elmore delay versus maximum area $A^{\max}$. For comparison, we also show the area and Elmore delay obtained when all wire widths are equal, i.e., $w_i = A^{\max}/5$. Even for this very small example, optimally sizing the wire segment widths gives a substantial reduction in delay (for fixed area).

Figure 16. Optimal trade-off curve of Elmore delay D versus maximum area $A^{\max}$ for the interconnect network shown in Figure 14.



Note. The dashed curve shows the area and delay obtained when all wire widths are equal.

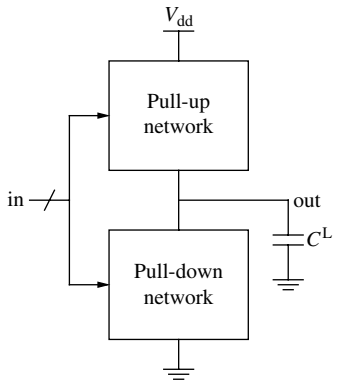
4.3. Gate Design

A (static CMOS) gate is a combination of a *pull-down network* of NMOS transistors and a *pull-up network* of PMOS transistors, as shown in Figure 17. The pull-up network provides a connection between the output and the power supply when the gate output is logical high (V_{dd}), and the pull-down network provides a connection between the gate output and ground when the gate output is logical low (0). A simple specific example is the two-input NAND gate shown in Figure 2. The pull-up network consists of a parallel connection of two PMOS devices, and the pull-down network consists of a series connection of two NMOS devices.

In this section, we consider the problem of gate design, i.e., choosing the widths $w_1, \ldots, w_p$ of the devices in the pull-up and pull-down networks in a gate, subject to constraints on gate area, delay, input capacitance, power, and so on. For the simple two-input NAND gate, for example, there are four device widths to choose. These device widths affect the gate area, input capacitance at the two inputs, gate delay, and dynamic and static power. We can also include other design variables in the gate design problem, e.g., threshold voltages for each of the devices. For simplicity, though, we consider device sizing only. Before going into the details of gate design, we sketch the basic approach.

We start with some constraints that are readily formulated. Manufacturing constraints specify minimum and sometimes also maximum allowable values for the device widths. The area of a gate is also easily handled, because

Figure 17. A CMOS logic gate formed from a pull-up network and a pull-down network.



it can be approximated as a linear function of the device widths, with positive coefficients, and so is a posynomial. (We can use a more refined generalized posynomial function of the device widths that takes into account more details of the actual gate physical layout.) Input capacitance is also straightforward to model because it can be approximated as a linear or affine function of the device widths.

Gate delay and power constraints are more complex, but also GP compatible when the models described below are used. (Several more sophisticated and accurate models are also GP compatible; see, e.g., Kasamsetty et al. 2000, Sapatnekar and Chuang 2000.) For the design of gates with not too many inputs, it is common to distinguish and model the gate behavior for every input transition. For each such transition, we model the energy loss and the delay to the output (provided the output changes during the input transition). In a similar way, we model the leakage current for each input state.

A basic gate design problem might have the form

$$\begin{array}{ll} \text{minimize} & D = \max\{D_1, \ldots, D_K\} \\ \text{subject to} & w_i^{\min} \leqslant w_i \leqslant w_i^{\max}, \quad i = 1, \ldots, p, \\ & A \leqslant A^{\max}, \\ & (1/K)\sum_{k=1}^{K} E_k \leqslant E^{\max}, \\ & C_i^{\text{in}} \leqslant C^{\text{in, max}}, \quad i = 1, \ldots, m, \end{array} \tag{27}$$

where A denotes the gate area, D_k denotes the delay for transition k, E_k denotes the energy loss during transition k, and C_i^{in} denotes the input capacitance at gate input i. (Thus, K is the total number of transitions, and m is the number of inputs for the gate.) Thus, we seek gate device widths that minimize (maximum) delay, subject to a gate area limit $A^{\max}$, an average energy loss (over all transitions) that does not exceed $E^{\max}$, and a maximum input capacitance (at any of the gate inputs) $C^{\text{in, max}}$. This problem is a GGP, provided D_k, E_k, and C_i^{in} are generalized posynomials.

There are many variations on the basic gate design problem. We can impose different delay or energy loss limits for different transitions, and different limits on input capacitance at different inputs. We can also impose limits on leakage power. We can form a robust design to take into account variation in the model parameters or load capacitance.

4.3.1. RC Gate Model. The first step in gate modeling is to substitute a *switch-level RC model* of each device into the gate schematic diagram. This results in a gate schematic diagram containing switches (which are open or closed depending on the inputs), and capacitors and resistors, whose values depend on the device widths. For each transition we obtain an RC circuit, typically a tree, with given initial conditions on the capacitors. The energy loss is modeled as the total energy loss in the RC circuit, and the gate delay is modeled as the Elmore delay to the gate output node. We will see that these are both posynomials of the device widths. It is also possible to model the leakage power of a gate as a function of the device widths, but it is quite involved, so we do not consider it here.

Figure 18. MOS transistors (left) and the switch-level RC model (right).

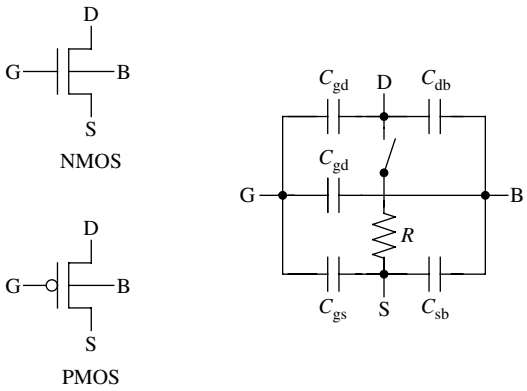


Figure 18 shows a simple switch-level RC circuit model for a transistor or device. It consists of a source-to-drain resistance R, which models the channel resistance when the device is on, and five parasitic capacitances between the four terminals, bulk (B), gate (G), source (S), and drain (D). In our discussion we will ignore the gate-drain capacitance, for simplicity, but it can be handled, in a GP-compatible way, via the Miller effect. We refer the reader to Hodges et al. (2004) for detailed discussion of switch-level RC device models and their use in delay modeling. We assume that each of these parameters is a posynomial function of the device width. In one simple model, for example, the resistance is inversely proportional to width, and the capacitances are linear (or affine) in the width.

Substituting this switch-level RC model for each device in a gate, with the NMOS bulk terminals connected to ground and the PMOS bulk terminals connected to V_{dd}, we obtain a switched RC circuit model of a gate. For the purposes of delay and energy analysis, this resulting RC circuit is equivalent to one in which the capacitances connected to V_{dd} are connected to ground. For each input transition, we obtain an RC circuit, with known initial conditions.

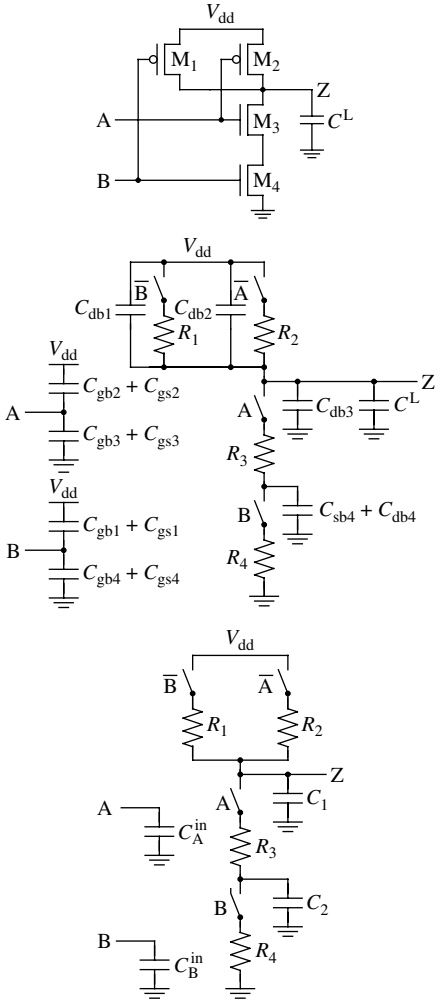
To illustrate this approach we consider a simple two-input NAND gate driving a load capacitance C^{L}, shown on the top of Figure 19. Using the switch-level RC device model, it can be modeled as the RC circuit in the middle, which is electrically equivalent to the RC circuit on the bottom, where

$$C_1 = C_{\text{db1}} + C_{\text{db2}} + C_{\text{db3}} + C^{\text{L}}, \qquad C_2 = C_{\text{sb3}} + C_{\text{db4}}.$$

The input capacitances at input pins A and B are

$$C_{\text{A}}^{\text{in}} = C_{\text{gb2}} + C_{\text{gs2}} + C_{\text{gb3}} + C_{\text{gs3}},$$
$$C_{\text{B}}^{\text{in}} = C_{\text{gb1}} + C_{\text{gs1}} + C_{\text{gb4}} + C_{\text{gs4}}.$$

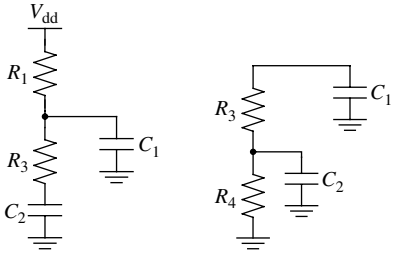
Figure 19. Two-input NAND gate (top), its switched RC circuit model (middle), and an electrically equivalent switched RC circuit model (bottom).



Because the device resistance and capacitances are assumed to be posynomial functions of the widths, the resistances and capacitances in the switched RC circuit are all posynomials of the device widths. Now we can analyze each transition separately. There are four input states, and therefore 12 possible transitions. Of these 12 transitions, three transitions cause the output Z to rise from 0 to V_{dd}, and three cause the output Z to fall from V_{dd} to 0.

To illustrate the delay and energy model for a rising transition, consider the input transition in which gate input B falls from V_{dd} to zero, while gate input A remains at V_{dd}. Before the transition, the switches labeled A and B are closed, and the switches labeled $\bar{A}$ and $\bar{B}$ are open. The output Z is at zero, and the capacitors C_1 and C_2 have zero initial charge. When the gate input B falls to zero, the switch labeled B opens, and the switch labeled $\bar{B}$ closes. This leaves us with the RC circuit shown on the left-hand side of Figure 20, with C_1 and C_2 initially uncharged. The Elmore

Figure 20. RC models of the two-input NAND gate for two input transitions: A remains at V_{dd}, and B falls from V_{dd} (left); A and B rise from zero to V_{dd} (right).



delay to the output node is

$$D = R_1(C_1 + C_2),$$

and the energy dissipated is

$$E = (C_1 + C_2)V_{dd}^2/2.$$

These are posynomials of the device sizes.

As another example, we consider the input transition in which both gate inputs A and B rise from zero to V_{dd}, which causes the output Z to fall to zero. In this transition, the switches labeled A and B close, and the switches labeled $\bar{A}$ and $\bar{B}$ open. This results in the RC tree shown on the right-hand side of Figure 20. For this transition, the initial voltage v_2^{init} on C_2 is not well defined; it can be any value between 0 and V_{dd}. The (modified) Elmore delay to the output node Z is

$$D = R_3C_1 + R_4\left(C_1 + \left(v_2^{init}/V_{dd}\right)C_2\right),$$

and the energy dissipated (during this transition) is

$$E = C_1V_{dd}^2/2 + C_2\left(v_2^{init}\right)^2/2.$$

Using the worst-case value $v_2^{init} = V_{dd}$, we obtain delay and energy (bounds)

$$D = R_3C_1 + R_4(C_1 + C_2), \qquad E = (C_1 + C_2)V_{dd}^2/2,$$

which are posynomials of the device sizes.

The delay and energy expressions for the six input transitions which result in an output transition are given in Table 3. For the last two transitions, the exact expression depends on v_2^{init}, so the worst-case bound is given. The third transition also deserves comment. When both inputs fall from one to 0, the resulting RC circuit is not a tree, because *two* resistors (R_1 and R_2) connect to V_{dd}. As a result, the delay expression is not a posynomial of the R_i. In this case, however, the delay is *always* smaller than the delay when only input A falls from one to 0 (in which R_2 alone connects to V_{dd}), so the (nonposynomial) delay for this transition can be ignored. The maximum delay (over all transitions) is

$$D = \max\{R_3C_1 + R_4(C_1 + C_2), R_1(C_1 + C_2), R_2C_1\},$$

which is a generalized posynomial. The energy models for the six input transitions listed in Table 3 are also generalized posynomials.

Table 3. Delay and energy expressions for the two-input NAND gate for the six input transitions that yield an output transition.

A	B	Z	Delay	Energy dissipated
$1\to1$	$1\to0$	$0\to1$	$R_1(C_1+C_2)$	$(C_1+C_2)V_{\mathrm{dd}}^2/2$
$1\to0$	$1\to1$	$0\to1$	R_2C_1	$C_1V_{\mathrm{dd}}^2/2$
$1\to0$	$1\to0$	$0\to1$	$C_1R_1R_2/(R_1+R_2)$	$C_1V_{\mathrm{dd}}^2/2$
$1\to1$	$0\to1$	$1\to0$	$R_3C_1+R_4(C_1+C_2)$	$(C_1+C_2)V_{\mathrm{dd}}^2/2$
$0\to1$	$1\to1$	$1\to0$	$R_3C_1+R_4C_1$	$(C_1+C_2)V_{\mathrm{dd}}^2/2$
$0\to1$	$0\to1$	$1\to0$	$R_3C_1+R_4(C_1+C_2)$	$(C_1+C_2)V_{\mathrm{dd}}^2/2$

4.3.2. Example. In this section, we give a simple numerical example of the design of the simple two-input NAND gate. Of course, this problem is simple enough that it could be solved by exhaustive search over the four widths. The gate design problem becomes more challenging for more complex gates, with 10 or more devices, in which case the GGP formulation becomes valuable.

The switch-level RC device model parameters for NMOS devices are

$$R = 0.4831/w, \qquad C_{\mathrm{db}} = C_{\mathrm{sb}} = 0.6w, \qquad C_{\mathrm{gb}} = C_{\mathrm{gs}} = w,$$

and for PMOS devices,

$$R = 2\cdot 0.4831/w, \qquad C_{\mathrm{db}} = C_{\mathrm{sb}} = 0.6w, \qquad C_{\mathrm{gb}} = C_{\mathrm{gs}} = w,$$

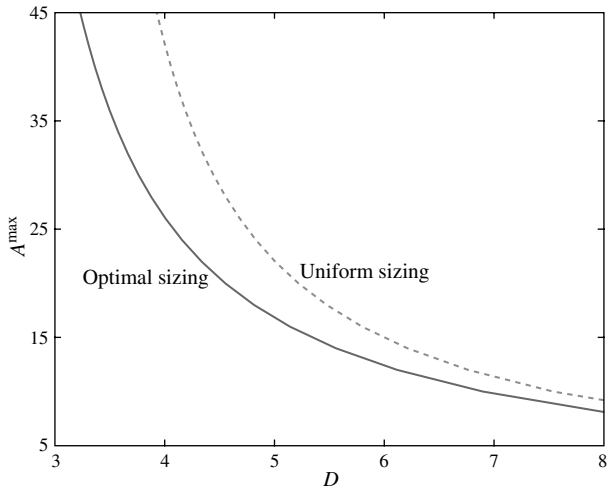
which are monomials of device sizes.

We solve the gate design problem (27), with minimum device width $w_i^{\min} = 1$, and varying maximum area (taken as the sum of the widths) $A^{\max}$. The load capacitance is $C^{\mathrm{L}} = 12$. Figure 21 shows the optimal trade-off curve of minimum delay D versus area limit $A^{\max}$. For comparison we also show the delay obtained when all device sizes are equal, i.e., $w_i = A^{\max}/4$. Even for this very small example, optimally sizing the device widths gives a substantial reduction in delay, compared to the design with all gate widths equal. For $A^{\max} = 24$, the optimal widths are

$$w_1 = 6.68, \qquad w_2 = 5.43, \qquad w_3 = 5.03, \qquad w_4 = 6.86,$$

which are not far from equal, but give a good reduction in delay.

Figure 21. Delay versus area for the design with $w_i = A^{\max}/4$ (uniform sizing) and optimal designs.



4.4. Custom Design

The ideas of gate design and gate scaling are readily combined to form the *custom design problem*. In custom design, the gates forming a combinational logic block are optimized for overall power, area, and delay, exactly as in gate scaling; the difference is that in custom design, the optimization variables are the individual devices within each gate, instead of a single scaling parameter per gate.

If the gate design formulation and the gate scaling formulation are both GP compatible, the overall custom design problem is also GP compatible. We start with a gate scaling problem formulation that connects the properties of each gate, such as delay, power, area, and input capacitance, to the combinational logic block delay, power, and area. In gate scaling, the gate properties are generalized posynomials of a single scaling parameter; in custom design, we substitute the more complex generalized posynomial expressions for the gate properties, in terms of the gate device widths. The result is a very large GGP. This works out because in the gate scaling formulation, the gate properties can be *any* generalized posynomial of *any* design variables. In simple scaling, the gate properties are posynomials of a single scale parameter x; in custom design, they are generalized posynomials of the gate device widths $w_1, \ldots, w_p$.

Indeed, *all* of the problem formulations described above can be combined. We can include interconnect wire sizing, as well as the choice of supply and threshold voltage, in the custom design problem. We can form robust or multimode versions of the resulting problems. These problems will be GP compatible, because, roughly speaking, GP is preserved under composition and hierarchy.

The natural problem hierarchy arising in the custom design problem can be exploited computationally in several ways. The most straightforward is to simply form a very large scale GP, and let an interior-point solver exploit the resulting sparsity each time the large problem is solved. In this approach we formulate the problem in a hierarchical way, then solve the problem as one large sparse GP.

Another interesting approach is to design a family of gates, and then form a reduced order model of the family that can be used in the larger combinational logic block problem. Consider, for example, a gate with 10 devices. We design, say, 10,000 different versions of the gate, each Pareto optimal with respect to input capacitance, energy

loss, average leakage power, delay, area, and so on, with varying load capacitance. (This is possible because the gate design problem is small, and can be solved extremely quickly.) We then fit two- or three-parameter generalized posynomial models of the properties of the gates in this family and use this reduced order model in the larger problem, instead of the full 10-parameter model that would be used in custom design.

The theoretical justification of this approach is as follows. First consider a convex optimization problem. Its optimal value, as a function of the right-hand sides of the inequalities, is always a convex function. This basic result transposes to GGPs via the logarithmic transformation used to transform a GGP or GP into a convex problem: the optimal value of a GGP, as a function of the right-hand sides of the constraints, is a function that can be arbitrarily well approximated by a generalized posynomial. If, for example, we design a gate to minimize delay subject to limits on area, energy loss, leakage power, and input capacitance, with a given load capacitance, then the resulting minimum delay, as a function of these limits and the load capacitance, can be arbitrarily well approximated as a generalized posynomial.

5. Conclusions

In the preceding sections, we have listed a wide variety of digital circuit sizing problems that can be cast, at least approximately, as geometric programs or generalized geometric programs. These problems range from simple ones, involving gate scaling to achieve a trade-off among area, power, and delay, to complex formulations involving multiple corners, multiple scenarios, joint device sizing and threshold voltage optimization, and so on.

The knowledge that a given design problem can be approximately cast as a GGP can be used in several ways. The most obvious is to solve the design problem using a generic interior-point method for GP. But even if an ad hoc method is used to solve the problem, the knowledge is useful because it tells us that a local solution of the problem is also a global solution. And it seems to us that reducing a design problem to a GP is conceptually useful, even if methods specific to GP are not used to solve the problem.

When a design problem involves discrete constraints, the resulting optimization problem is a mixed-integer or discrete GP. Such problems are difficult to solve exactly, but examples show that even relatively simple heuristics, based on solving one or more (continuous) GPs, seem to work well in practice. In any case, the relaxed GP formulation provides a lower bound against which the heuristic can be judged.

We should make some comments on the various models described above. Our focus is on GP-compatibility, and not on accuracy. At the gate or device levels, there are two extreme approaches to GP modeling. One is based on more refined analysis and is essentially equation based. The other is based on fitting GP-compatible functions to characterized, simulated, or measured data. Each method has its advantages and disadvantages. An equation-based method, for example, can handle many more varying parameters; a fitting method, on the other hand, gives a simple method for achieving higher accuracy.

Considering the approximations made in GP modeling, fitting errors, and ignored constraints, it should be borne in mind that the final GP formulation is only an approximation of the original circuit design problem. Its practical value is in getting close to a good solution; this can be followed by a final "polishing" of the design, starting from the GP solution obtained, and using a simple local method, based on more accurate (but not GP compatible) models.

Appendix. Generalized Geometric Programming

In this section, we give a very brief description of GP and GGP. Let $x = (x_1, \ldots, x_n)$ be a vector of n real, positive variables. A function of the form $cx_1^{a_1}x_2^{a_2}\cdots x_n^{a_n}$, where $c > 0$, is called a *monomial* function; a sum of monomial functions is called a *posynomial* function. A *generalized posynomial* function is any function obtained from posynomials using addition, multiplication, pointwise maximum, and raising to constant positive power. The functions

$$0.23, \qquad 1.5\sqrt{x_1/x_2},$$

for example, are monomials; the functions

$$2x_1 + x_3, \qquad 1 + 3x_1^{1.3}x_2^{-0.4}$$

are posynomials; and the function

$$\max\{3x_1 + x_2/x_3, 2x_1^{-0.3} + (x_1 + x_3)^{1.2}\}$$

is a generalized posynomial.

Any monomial is also a posynomial and a generalized posynomial; and any posynomial is also a generalized posynomial. Monomials are closed under product and any power (such as squareroot or inverse). Posynomials are closed under product and sum. Generalized posynomials are closed under product, sum, positive powers, and pointwise maximum.

A geometric program is an optimization problem of the form

$$\begin{aligned} &\text{minimize} && f_0(x) \\ &\text{subject to} && f_i(x) \leqslant 1, \quad i = 1, \ldots, m, \\ &&& g_i(x) = 1, \quad i = 1, \ldots, p, \end{aligned} \tag{28}$$

where $f_0, \ldots, f_m$ are posynomial functions and $g_1, \ldots, g_p$ are monomials. (The optimization variables are $x_1, \ldots, x_n$.) If the functions $f_0, \ldots, f_m$ are generalized posynomials, problem (28) is called a generalized geometric program.

Several extensions are readily handled. For example, we can maximize a monomial by minimizing its inverse (which is also a monomial, hence a posynomial). A constraint of the form $f(x) \leqslant g(x)$, where f is a generalized posynomial and g is a monomial, can be handled by expressing it as $f(x)/g(x) \leqslant 1$ (because f/g is a generalized posynomial). A constraint of the form $g(x) = \tilde{g}(x)$, where g and $\tilde{g}$ are monomials, can be handled by expressing it as $g(x)/\tilde{g}(x) = 1$ (because $g/\tilde{g}$ is a monomial).

Acknowledgments

This work was carried out with support from C2S2, the MARCO Focus Center for Circuit and System Solutions, under MARCO contract 2003-CT-888, and Barcelona Design, Inc. The authors thank Soo Hee Han, Sachin Sapatnekar, and Vladimir Stojanović for helpful comments and suggestions. They also thank Jeong Taek Kong for suggesting some of the topics explored in this paper.

References

Abou-Seido, A., B. Nowak, C. Chu. 2004. Fitted Elmore delay: A simple and accurate interconnect delay model. *IEEE Trans. VLSI Systems* **12**(7) 691–696.

Agarwal, A., V. Zolotov, D. Blaauw. 2003. Statistical timing analysis using bounds and selective enumeration. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **22**(9) 1243–1260.

Alpert, C., A. Devgan, C. Kashyap. 2001a. RC delay metrics for performance optimization. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **20**(5) 571–582.

Alpert, C., A. Devgan, J. Fishburn, S. Quay. 2001b. Interconnect synthesis without wire tapering. *Integration, the VLSI J.* **20** 90–114.

Alpert, C., C. Chu, G. Gandham, M. Hrkic, J. Hu, C. Kashyap, S. Quay. 2004. Simultaneous driver sizing and buffer insertion using a delay penalty estimation technique. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **23**(1) 136–141.

Anis, M., M. Allam, M. Elmasry. 2002. Energy-efficient noise-tolerant dynamic styles for scaled-down CMOS and MTCMOS technologies. *IEEE Trans. Very Large Scale Integration* (*VLSI*) *Systems* **10**(2) 71–78.

Anis, M., S. Areibi, M. Elmasry. 2003. Design and optimization of multithreshold CMOS (MTCMOS) circuits. *IEEE Trans. Comput.-Aided Design of Integrated Circuits Systems* **22**(10) 1324–1342.

Anklesaria, K., Z. Drezner. 1986. A multivariate approach to estimating the completion time for PERT networks. *J. Oper. Res. Soc.* **37** 811–815.

Assaderaghi, F., D. Sinitsky, S. Parke, J. Bokor, P. Ko, C. Hu. 1997. Dynamic threshold-voltage MOSFET (DTMOS) for ultra-low voltage VLSI. *IEEE Trans. Electron Devices* **44**(3) 414–422.

Augsburger, S., B. Nikolić. 2002a. Combining dual-supply, dual-threshold and transistor sizing for power reduction. *Proc. IEEE Internat. Conf. Comput. Design: VLSI in Computers and Processors*. Cambridge, MA, 316–321.

Augsburger, S., B. Nikolić. 2002b. Reducing power with dual-supply, dual-thresholds and transistor sizing. *Proc. IEEE Internat. Conf. Comput. Design: VLSI in Computers and Processors.* Cambridge, MA, 16–18.

Bellaouar, A., A. Fridi, M. Elmasry, K. Itoh. 1998. Supply voltage scaling for temperature insensitive CMOS circuit operation. *IEEE Trans. Circuits Systems II: Analog and Digital Signal Processing* **45**(3) 415–417.

Bharadwaj, B., M. Horowitz. 2000. Speed and power scaling of SRAMs. *IEEE J. Solid-State Circuits* **35**(2) 175–185.

Bhardwaj, S., S. Vrudhula, D. Blaauw. 2003. TAU: Timing analysis under uncertainty. *Internat. Conf. Comput.-Aided Design*. San Jose, CA, 615–620.

Blaauw, D., V. Zolotov, S. Sundareswaran. 2002. Slope propagation in static timing analysis. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **21**(10) 1180–1195.

Borah, M., R. Owens, M. Irwin. 1997. A fast algorithm for minimizing the Elmore delay to identified critical sinks. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **16**(7) 753–759.

Bowman, R. 1995. Efficient estimation of arc criticalities in stochastic activity networks. *Management Sci.* **41**(1) 58–67.

Bowman, K., L. Wang, X. Tang, J. Meindl. 2001. A circuit-level perspective of the optimum gate oxide thickness. *IEEE Trans. Electron Devices* **48**(8) 1800–1810.

Boyd, S., L. Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press, Cambridge, UK.

Boyd, S., S.-J. Kim, L. Vandenberghe, A. Hassibi. 2004. A tutorial on geometric programming. *Optim. Engrg.* Forthcoming. Available from www.stanford.edu/boyd/˜gp_tutorial.html.

Brambilla, A., P. Maffezzoni. 2001. Statistical method for the analysis of interconnects delay in submicrometer layouts. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **20**(8) 957–966.

Bryant, R., K.-T. Cheng, A. Kahng, K. Keutzer, W. Maly, R. Newton, L. Pileggi, J. Rabaey, A. Sangiovanni-Vincentelli. 2001. Limitations and challenges of computer-aided design technology for CMOS VLSI. *Proc. IEEE* **89**(3) 341–365.

Calhoun, B., F. Honore, A. Chandrakasan. 2004. A leakage reduction methodology for distributed MTCMOS. *IEEE J. Solid-State Circuits* **39**(5) 818–826.

Chabini, N., I. Chabini, E. Aboulhamid, Y. Savaria. 2003. Methods for minimizing dynamic power consumption in synchronous designs with multiple supply voltages. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **22**(3) 346–351.

Chandrakasan, A., R. Brodersen. 1995. Minimizing power consumption in digital CMOS circuits. *Proc. IEEE* **83**(4) 498–523.

Chang, J.-M., M. Pedram. 1997. Energy minimization using multiple supply voltages. *IEEE Trans. Very Large Scale Integration Systems* **5**(4) 436–443.

Chen, C., M. Sarrafzadeh. 2002. Simultaneous voltage scaling and gate sizing for low-power design. *IEEE Trans. Circuits Systems II: Analog and Digital Signal Processing* **49**(6) 400–408.

Chen, C., A. Srivastava, M. Sarrafzadeh. 2001. On gate level power optimization using dual-supply voltages. *IEEE Trans. Very Large Scale Integration Systems* **9**(5) 616–629.

Chen, C.-P., D. Wong. 1999. Greedy wire-sizing is linear time. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **18**(4) 398–405.

Chen, C.-P., C. Chu, D. Wong. 1999. Fast and exact simultaneous gate and wire sizing by Lagrangian relaxation. *IEEE Trans. Comput.-Aided Design of Integrated Circuits Systems* **18**(7) 1014–1025.

Chen, K., H. Hu, P. Fang, M. Lin, D. Wollesen. 1997. Predicting CMOS speed with gate oxide and voltage scaling and interconnect loading effects. *IEEE Trans. Electron Devices* **44**(11) 1951–1957.

Chen, T.-C., S.-R. Pan, Y.-W. Chang. 2004. Timing modeling and optimization under the transmission line model. *IEEE Trans. Very Large Scale Integration Systems* **12**(1) 28–41.

Chen, W., C.-T. Hseih, M. Pedram. 2000. Simultaneous gate sizing and placement. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **19**(2) 206–214.

Chu, C., D. Wong. 1999. An efficient and optimal algorithm for simultaneous buffer and wire sizing. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **18**(9) 1297–1304.

Chu, C., D. Wong. 2001a. Closed form solutions to simultaneous buffer insertion/sizing and wire sizing. *ACM Trans. Design Automation of Electronic Systems* **6**(3) 343–371.

Chu, C., D. Wong. 2001b. VLSI circuit performance optimization by geometric programming. *Ann. Oper. Res.* **105** 37–60.

Colleran, D., C. Portmann, A. Hassibi, C. Crusius, S. Mohan, S. Boyd, T. Lee, M. Hershenson. 2003. Optimization of phase-locked loop circuits via geometric programming. *Proc. Custom Integrated Circuits Conf.* (*CICC*). Orlando, FL, 326–328.

Cong, J., H. He. 1999. Theory and algorithm of local-refinement-based optimization with application to device and interconnect sizing. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **18**(4) 406–420.

Cong, J., L. He. 1996. Optimal wire sizing for interconnects with multiple sources. *ACM Trans. Design Automation Electronic Systems* **1**(4) 478–511.

Cong, J., C.-K. Koh. 1994. Simultaneous driver and wire sizing for performance and power optimization. *IEEE Trans. Very Large Scale Integration Systems* **2**(4) 408–423.

Cong, J., K.-S. Leung. 1995. Optimal wiresizing under Elmore delay model. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **14**(3) 321–336.

Cong, J., Z. Pan. 2002. Wire width planning for interconnect performance optimization. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **21**(3) 319–329.

Cong, J., L. He, C.-K. Koh, P. Madden. 1996. Performance optimization of VLSI interconnect layout. *Integration, the VLSI J.* **21** 1–94.

Davis, E. 1966. Resource allocation in project network models—A survey. *J. Indust. Engrg.* **17**(4) 77–187.

Dawson, J., S. Boyd, M. Hershenson, T. Lee. 2001. Optimal allocation of local feedback in multistage amplifiers via geometric programming. *IEEE Trans. Circuits Systems I* **48**(1) 1–11.

Devroye, L. 1979. Inequalities for the completion times of stochastic PERT networks. *Math. Oper. Res.* **4**(4) 441–447.

Dodin, B. 1984. Determining the K most critical paths in PERT networks. *Oper. Res.* **32** 859–877.

Ebergen, J., J. Gainsley, P. Cunningham. 2004. Transistor sizing: How to control the speed and energy consumption of a circuit. *Proc. 10th Internat. Sympos. Asynchronous Circuits Systems*. Crete, Greece, 51–61.

Elmaghraby, S. 1970. *Some Network Models in Management Science*. Springer-Verlag, New York.

Elmaghraby, S. 1977. *Project Planning and Control by Network Models*. John Wiley and Sons, New York.

Elmore, W. 1948. The transient response of damped linear networks with particular regard to wideband amplifiers. *J. Appl. Phys.* **19**(1) 55–63.

Fishburn, J., A. Dunlop. 1985. TILOS: A posynomial programming approach to transistor sizing. *IEEE Internat. Conf. Comput.-Aided Design: ICCAD-85. Digest Tech. Papers*. IEEE Computer Society Press, Santa Clara, CA, 326–328.

Fishburn, J., C. Schevon. 1995. Shaping a distributed-RC line to minimize Elmore delay. *IEEE Trans. Circuits Systems I: Fundamental Theory Appl.* **42**(12) 1020–1022.

Gao, Y., D. Wong. 1999. Optimal shape function for a bidirectional wire under Elmore delay model. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **18**(7) 994–999.

Gupta, R., B. Tutuianu, L. Pileggi. 1997. The Elmore delay as a bound for RC trees with generalized input signals. *IEEE Trans. Comput.-Aided Design of Integrated Circuits Systems* **16**(1) 95–104.

Hartley, H., A. Wortham. 1966. A statistical theory for PERT critical path analysis. *Management Sci.* **12**(6) 469–481.

Hassibi, A., M. Hershenson. 2002. Automated optimal design of switched-capacitor filters. *Design, Automation and Test in Europe Conference and Exhibition*. Paris, France, 1111.

Hershenson, M. 2002. Design of pipeline analog-to-digital converters via geometric programming. *Proc. IEEE/ACM Internat. Conf. Comput. Aided Design*. San Jose, CA, 317–324.

Hershenson, M. 2003. Analog design space exploration: Efficient description of the design space of analog circuits. *Proc. 40th Design Automation Conf.* Anaheim, CA, 970–973.

Hershenson, M., S. Boyd, T. Lee. 1998. GPCAD: A tool for CMOS op-amp synthesis. *Proc. IEEE/ACM Internat. Conf. Comput. Aided Design*. San Jose, CA, 296–303.

Hershenson, M., A. Hajimiri, S. Mohan, S. Boyd, T. Lee. 1999. Design and optimization of LC oscillators. *Proc. IEEE/ACM Internat. Conf. Comput.-Aided Design*. San Jose, CA, 65–69.

Ho, R., K. Mai, M. Horowitz. 2001. The future of wires. *Proc. IEEE* **89**(4) 490–504.

Hodges, D., H. Jackson, R. Saleh. 2004. *Analysis and Design of Digital Integrated Circuits*, 3rd ed. McGraw-Hill, New York.

Horowitz, M. 1984. Timing models for MOS circuits. Ph.D. thesis, Stanford University, Stanford, CA.

Hung, W., Y. Xie, N. Vijaykrishnan, M. Kandemir, M. Irwin, Y. Tsai. 2004. Total power optimization through simultaneously multiple-V_{dd} multiple-V_{th} assignment and device sizing with stack forcing. *Proc. Internat. Sympos. Low Power Electronics and Design* (*ISLPED*). Newport Beach, CA, 144–149.

Im, H., T. Inukai, H. Gomyo, T. Hiramoto, T. Sakurai. 2003. VTCMOS characteristics and its optimum conditions predicted by a compact analytical model. *IEEE Trans. Very Large Scale Integration* (*VLSI*) *Systems* **11**(5) 755–761.

Ishihara, F., F. Sheikh, B. Nikolić. 2004. Level conversion for dual-supply systems. *IEEE Trans. Very Large Scale Integration* (*VLSI*) *Systems* **12**(2) 185–195.

Ismail, Y., E. Friedman, J. Neves. 2000. Equivalent Elmore delay for RLC trees. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **19**(7) 83–97.

Jiang, I., Y. Chang, J. Jou. 2000. Crosstalk-driven interconnect optimization by simultaneous gate and wire sizing. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **19**(9) 999–1010.

Johnson, M., D. Somasekhar, L.-Y. Chiou, K. Roy. 2002. Leakage control with efficient use of transistor stacks in single threshold CMOS. *IEEE Trans. Very Large Scale Integration* (*VLSI*) *Systems* **10**(1) 1–5.

Jung, S., K. Kim, S. Kang. 2003. Noise constrained power optimization for dual V_t domino logic. *IEEE Trans. Very Large Scale Integration* (*VLSI*) *Systems* **10**(5) 532–541.

Jyu, H.-F., S. Malik, S. Devadas, K. Keutzer. 1993. Statistical timing analysis of combinational logic circuits. *IEEE Trans. VLSI Systems* **1**(2) 126–137.

Kahng, A., S. Muddu. 1997. An analytical delay model for RLC interconnects. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **16**(12) 1507–1514.

Kao, I., M. Miyazaki, A. Chandrakasan. 2002. A 175mv multiply-accumulate unit using an adaptive supply voltage and body bias architecture. *IEEE J. Solid-State Circuits* **37**(11) 1545–1554.

Kao, J., A. Chandrakasan. 2000. Dual-threshold voltage techniques for low-power digital circuits. *IEEE J. Solid-State Circuits* **35**(7) 1009–1018.

Karnik, T., Y. Ye, J. Tschanz, L. Wei, S. Burns, V. Govindarajulu, V. De, S. Borkar. 2002. Total power optimization by simultaneous dual-V_{th} allocation and device sizing in high performance microprocessors. *Proc. 39th IEEE/ACM Design Automation Conf.* New Orleans, LA, 486–491.

Kasamsetty, K., M. Ketkar, S. Sapatnekar. 2000. A new class of convex functions for delay modeling and its application to the transistor sizing problem. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **19**(7) 779–788.

Kashyap, C., C. Alpert, F. Liu, A. Devgan. 2004. Closed-form expressions for extending step delay and slew metrics to ramp inputs for RC trees. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **23**(4) 509–516.

Kay, R., L. Pileggi. 1998. EWA: Efficient wiring-sizing algorithm for signal nets and clock nets. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **17**(1) 40–49.

Ketkar, M., S. Sapatnekar. 2002. Standby power optimization via transistor sizing and dual threshold voltage assignment. *Proc. IEEE/ACM Internat. Conf. Comput.-Aided Design*. San Jose, CA, 375–378.

Kim, C., K. Kim, S. Kang. 2003a. Energy efficient skewed static logic design with dual V_{th}: Design and synthesis. *IEEE Trans. Very Large Scale Integration* (*VLSI*) *Systems* **11**(1) 64–70.

Kim, K.-W., S.-O. Jung, P. Saxena, C. Liu, S. M. Kang. 2003b. Coupling delay optimization by temporal decorrelation using dual threshold voltage technique. *IEEE Trans. Very Large Scale Integration* (*VLSI*) *Systems* **11**(5) 879–887.

Kim, S.-J., S. Boyd, S. Yun, D. Patil, M. Horowitz. 2004. A heuristic for optimizing stochastic activity networks with applications to statistical digital circuit sizing. Submitted to *Optim. Engrg.* Available from www.stanford.edu/~boyd/heur_san_opt.html.

Kleindorfer, G. 1971. Bounding distribution for stochastic acyclic networks. *Oper. Res.* **19** 1586–1601.

Kong, J.-T. 2004. CAD for nanometer silicon design challenges and success. *IEEE Trans. Very Large Scale Integration* (*VLSI*) *Systems* **12**(11) 1132–1147.

Krishnamurthy, R., L. Carley. 1997. Exploring the design space of mixed swing quadrail for low-power digital circuits. *IEEE Trans. Very Large Scale Integration Systems* **5**(4) 389–400.

Kulkarni, S., D. Sylvester. 2004. High performance level conversion for dual V_{dd} design. *IEEE Trans. Very Large Scale Integration* (*VLSI*) *Systems* **12**(9) 926–936.

Lee, D., D. Blaauw, D. Sylvester. 2004. Gate oxide leakage current analysis and reduction for VLSI circuits. *IEEE Trans. Very Large Scale Integration Systems* **12**(2) 155–166.

Lee, Y.-M., C. Chen, D. Wong. 2002. Optimal wire-sizing function under the Elmore delay model with bounded wire sizes. *IEEE Trans. Circuits Systems I: Fundamental Theory Appl.* **49**(11) 1671–1677.

Lin, T., L. Pileggi. 2001. RC(L) interconnect sizing with second order considerations via posynomial programming. *Proc. ACM/SIGDA Internat. Sympos. Physical Design*. Sonoma, CA, 16–21.

Lindert, N., T. Sugii, S. Tang, C. Hu. 1999. Dynamic threshold pass-transistor logic for improved delay at lower power supply voltages. *IEEE J. Solid-State Circuits* **34**(1) 85–89.

Liu, M., W.-S. Wang, M. Orshansky. 2004. Leakage power reduction by dual-V_{th} designs under probabilistic analysis of V_{th} variation. *Proc. Internat. Sympos. Low Power Electronics and Design* (*ISLPED*). Newport Beach, CA, 2–7.

Lou, J., W. Chen, M. Pedram. 1999. Concurrent logic restructuring and placement for timing closure. *Proc. IEEE/ACM Internat. Conf. Comput.-Aided Design*. San Jose, CA, 31–35.

Lu, Y.-H., L. Benini, G. De Micheli. 2002. Dynamic frequency scaling with buffer insertion for mixed workloads. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **21**(11) 1284–1305.

Ludwig, A., R. Möhring, F. Stork. 2001. A computational study on bounding the makespan distribution in stochastic project networks. *Ann. Oper. Res.* **102** 49–64.

Mandal, P., V. Visvanathan. 2001. CMOS op-amp sizing using a geometric programming formulation. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **20**(1) 22–38.

Marković, D., V. Stojanović, B. Nikolić, M. Horowitz, R. Brodersen. 2004. Methods for true energy-performance optimization. *IEEE J. Solid-State Circuits* **39**(8) 1282–1293.

Matson, M., L. Glasser. 1986. Macromodeling and optimization of digital MOS VLSI circuits. *IEEE Trans. Comput.-Aided Design Integrated Circuits and Systems* **5**(4) 659–678.

Moh, T.-S., T.-S. Chang, S. Hakimi. 1996. Globally optimal floorplanning for a layout problem. *IEEE Trans. Circuits Systems I: Fundamental Theory Appl.* **43**(29) 713–720.

Mohan, S., M. Hershenson, S. Boyd, T. Lee. 1999. Simple accurate expressions for planar spiral inductances. *IEEE J. Solid-State Circuit* **34**(10) 1419–1424.

Mohan, S., M. Hershenson, S. Boyd, T. Lee. 2000. Bandwidth extension in CMOS with optimized on-chip inductors. *IEEE J. Solid-State Circuits* **35**(3) 346–355.

Mukhopadhyay, S., C. Neau, R. Cakici, A. Agarwal, C. Kim, K. Roy. 2003. Gate leakage reduction for scaled devices using transistor stacking. *IEEE Trans. Very Large Scale Integration* (*VLSI*) *Systems* **11**(4) 716–730.

Nesterov, Y., A. Nemirovsky. 1994. *Interior-Point Polynomial Methods in Convex Programming*, Vol. 13. Studies in Applied Mathematics. SIAM, Philadelphia, PA.

Nguyen, D., A. Davare, D. Chinnery, B. Thompson, M. Orshansky, K. Keutzer. 2003. Minimization of dynamic and static power through joint assignment of threshold voltages and sizing optimization. *Proc. Internat. Sympos. Low Power Electronics and Design* (*ISLPED*). Seoul, Korea, 158–163.

Nocedal, J., S. J. Wright. 1999. *Numerical Optimization*. Springer Series in Operations Research. Springer, New York.

Orshansky, M., K. Keutzer. 2002. A general probabilistic framework for worst case timing analysis. *Proc. 39th IEEE/ACM Design Automation Conf.* New Orleans, LA, 556–561.

Orshansky, M., J. Chen, C. Hu. 1999. Direct sampling methodology for statistical analysis of scaled CMOS technologies. *IEEE Trans. Semiconductor Manufacturing* **12**(4) 403–408.

Pant, P., M. Roy, A. Chattejee. 2001. Dual-threshold voltage assignment with transistor sizing for low power CMOS circuits. *IEEE Trans. Very Large Scale Integration Systems* **9**(2) 390–394.

Passy, U. 1998. Theory and algorithm of local refinement based optimization. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **18**(4) 406–420.

Patil, D., Y. Yun, S.-J. Kim, S. Boyd, M. Horowitz. 2005. A new method for robust design of digital circuits. *Proc. Internat. Sympos. Quality Electronic Design* (*ISQED*). San Jose, CA, 676–681.

Pattanaik, M., S. Banerjee, B. Bahinipati. 2003. GP based transistor sizing for optimal design of nanoscale CMOS inverter. *IEEE Conf. Nanotechnology*. San Francisco, CA, 524–527.

Pedram, M. 1996. Power minimization in IC design: Principles and applications. *ACM Trans. Design Automation of Electronic Systems* **1**(1) 3–56.

Pelgrom, M. 1989. Matching properties of MOS transistors. *IEEE J. Solid State Circuits* **24**(5) 1433–1439.

Pillage, L., R. Rohrer. 1990. Asymptotic waveform evaluation for timing analysis. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **9**(4) 352–366.

Qin, Z., C.-K. Cheng. 2003. Realizable parasitic reduction using generalized Y-Δ transformation. *Proc. 40th IEEE/ACM Design Automation Conf.* Anaheim, CA, 220–225.

Rabaey, J., A. Chandrakasan, B. Nikolić. 2002. *Digital Integrated Circuits: A Design Perspective*, 2nd ed. Prentice-Hall, Englewood Cliffs, NJ.

Rezvani, P., M. Pedram. 2003. A fanout optimization algorithm based on the effort delay model. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **22**(12) 1671–1678.

Robillard, P., M. Trahan. 1976. The completion times of PERT networks. *Oper. Res.* **25** 15–29.

Roy, K., S. Mukhopadhyay, H. Mahmoodi-Meimand. 2003. Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits. *Proc. IEEE* **91**(2) 305–327.

Rubenstein, J., P. Penfield, M. Horowitz. 1983. Signal delay in RC tree networks. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **2**(3) 202–211.

Sakurai, T. 1988. Approximation of wiring delay in MOSFET LSI. *IEEE J. Solid-State Circuits* **18**(4) 418–426.

Sakurai, T., A. Newton. 1990. Alpha-power law MOSFET model and its application to CMOS inverter delay and other formulas. *IEEE J. Solid-State Circuits* **25**(2) 584–593.

Sancheti, P., S. Sapatnekar. 1996. Optimal design of macrocells for low power and high speed. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **15**(9) 1160–1166.

Sapatnekar, S. 1996. Wire sizing as a convex optimization problem: Exploring the area-delay tradeoff. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **15** 1001–1011.

Sapatnekar, S., W. Chuang. 2000. Power-delay optimization in gate sizing. *ACM Trans. Design Automation of Electronic Systems* **5**(1) 98–114.

Sapatnekar, S., V. Rao, P. Vaidya, S. Kang. 1993. An exact solution to the transistor sizing problem for CMOS circuits using convex optimization. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **12**(11) 1621–1634.

Sathyamurthy, H., S. Sapatnekar, J. Fishburn. 1998. Speeding up pipelined circuits through a combination of gate sizing and clock skew optimization. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **17**(2) 173–182.

Seidel, P., G. Even. 2004. Delay-optimized implementation of IEEE floating-point addition. *IEEE Trans. Comput.* **53**(2) 97–113.

Shyu, J., A. Sangiovanni-Vincetelli, J. Fishburn, A. Dunlop. 1988. Optimization-based transistor sizing. *IIEEE J. Solid-State Circuits* **23**(2) 400–409.

Singh, D., J. Rabaey, M. Pedram, F. Catthoor, S. Rajgopal, N. Sehgal, T. Mozdzen. 1995. Power conscious CAD tools and methodologies: A perspective. *Proc. IEEE* **83**(4) 570–594.

Sirichotiyakul, S., T. Edwards, O. Chanhee, R. Panda, D. Blaauw. 2002. Duet: An accurate leakage estimation and optimization tool for dual-V_{th} circuits. *IEEE Trans. Very Large Scale Integration Systems* **10**(2) 79–90.

Sirisantana, N., K. Roy. 2004. Low-power design using multiple channel lengths and oxide thicknesses. *IEEE Design Test Comput.* **21**(1) 56–63.

Srivastava, A., D. Sylvester. 2004. Minimizing total power by simultaneous V_{dd}/V_{th} assignment. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **23**(5) 665–677.

Sutherland, I., B. Sproull, D. Harris. 1999. *Logical Effort: Designing Fast CMOS Circuits*. Morgan Kaufmann Publishers, San Francisco, CA.

Sylvester, D., C. Hu. 2001. Analytical modeling and characterization of deep submicron interconnect. *Proc. IEEE* **89**(5) 634–666.

Takahashi, O., S. Dhong, M. Ohkubo, S. Onishi, R. Dennard, R. Hannon, S. Crowder, S. Iyer, M. Wordeman, B. Davari, W. Weinberger, N. Aoki. 2000. 1GHz fully pipelined 3.7ns address access time 8k 1024 embedded DRAM macro. *ISSCC Digest of Tech. Papers*. San Francisco, CA, 396–397.

Tschanz, J., S. Narendra, Y. Ye, B. Bloechel, S. Borkar, V. De. 2003. Dynamic sleep transistor and body bias for active leakage power control of microprocessors. *IEEE J. Solid-State Circuits* **38**(11) 1838–1845.

Usami, K., M. Horowitz. 1995. Clustered voltage scaling technique for low-power design. *Proc. 1995 Internat. Sympos. Low Power Design*. San Diego, CA, 3–8.

Van Slyke, R. 1963. Monte Carlo methods and the PERT problem. *Oper. Res.* **11** 839–860.

Vanderhaegen, J., R. Brodersen. 2004. Automated design of operational transconductance amplifiers using reversed geometric programming. *Proc. 41st IEEE/ACM Design Automation Conf.* IEEE/ACM, San Diego, CA, 133–138.

Vittal, A., M. Marek-Sadowska. 1997. Low-power buffered clock tree design. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **16**(9) 965–975.

Wei, L., Z. Chen, K. Roy, M. Johnson, Y. Ye, V. De. 1999. Design and optimization of dual-threshold circuits for low-voltage low-power applications. *IEEE Trans. Very Large Scale Integration Systems* **7**(1) 16–24.

Weste, N., D. Harris. 2004. *CMOS VLSI Design*, 3rd ed. Addison Wesley, Boston, MA.

Xu, Y., L. Pileggi, S. Boyd. 2004. ORACLE: Optimization with recourse of analog circuits including layout extraction. *Proc. 41st IEEE/ACM Design Automation Conf.* IEEE/ACM, San Diego, CA, 151–154.

Yang, I., C. Vieri, A. Chandrakasan, D. Antoniadis. 1997. Back-gated CMOS on SOIAS for dynamic threshold voltage control. *IEEE Trans. Electron Devices* **44**(5) 822–831.

Yang, N., W. Henson, J. Wortman. 2000. A comparative study of gate direct tunneling and drain leakage currents in N-MOSFETs with sub-2100-nm gate oxides. *IEEE Trans. Electronic Devices* **47**(8) 1636–1644.

Ye, Y. 1997. *Interior Point Algorithms: Theory and Analysis*. Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley, New York.

Yeh, Y.-J., S.-Y. Kuo, J.-Y. Jou. 2001. Converter-free multiple-voltage scaling techniques for low-power CMOS digital design. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **20**(1) 172–176.

Young, F., C. Chu, W. Luk, Y. Wong. 2001. Handling soft modules in general nonslicing floorplan using Lagrangian relaxation. *IEEE Trans. Comput.-Aided Design Integrated Circuits Systems* **20**(5) 687–692.