# Yield-constrained Digital Circuit Sizing via Sequential Geometric Programming

Yu Ben, Laurent El Ghaoui, Kameshwar Poolla, Costas J. Spanos
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720 USA

## Abstract

Circuit design under process variation can be formulated mathematically as a robust optimization problem with a yield constraint. Existing methods force designers to either resort to overly simplified circuit performance model, or rely on simplistic variability assumptions. On the other hand, accurate yield estimation must incorporate a sophisticated variability model that recognizes both systematic and random components at various levels of hierarchy. Unfortunately, such models are not compatible with existing optimization solutions. To solve the problem, we propose the sequential geometric programming method, which consists of iterative usage of geometric programming and importance sampling, and is capable of handling an arbitrary variability model. The proposed method is shown to be able to achieve the desired yield without overdesign, and solve circuits with thousands of gates within reasonable amount of time.

## 1. Introduction

The shrinking technology nodes in the presence of persistent process variation are necessitating deeper interaction between design and manufacturing [1], [2]. Whereas chip manufacturers try to minimize the variation by way of process inspection, process control and equipment tune-ups, designers must design robust circuits mitigating the impact of process variation. This paper focuses on the latter approach. Traditionally, a robust design is achieved by considering the worst possible case arising from process variation and guaranteeing that the design meets specifications under that situation. This so-called *worst-case design* methodology leads to very conservative solutions. In reality, what matters is the yield, or the probability that a design meets specification. If we design a circuit so that the yield is greater than or equal to a certain pre-defined value, we can obtain, better results than worst-case design.

Formulating the circuit design problem as an optimization problem has been the topic of design automation for many years (see for example [3] and the references thereof). Recently, significant amount of effort has been spent on incorporating process variation in optimization [4]-[7]. While the majority of these works only dealt with the worst-case approach, many researchers have started to look at the problem with yield in mind [5], [6]. However, these methods either resort to overly simplified circuit performance formulae in order to conform to a particular optimization formulation, or rely on the assumption that the variability carries certain predefined distribution, typically a Gaussian. The latter is a particularly serious problem since a a predefined distribution can disagree tremendously with the remote tail of the actual distribution, which in fact is the most important region for yield estimation. A realistic variability model extracted from data [8] is often too complicated to be included in the existing optimization framework. Therefore, a generic optimization framework that is capable of utilizing more realistic and accurate variability model is desired.

In this report, sequential geometric programming (SGP) is introduced to solve the yield-constrained digital circuit sizing problem. This method fully explores the distribution of the perturbation by combining geometric programming with importance sampling. High efficiency of both geometric programming and importance sampling enables the method to efficiently solve circuits with thousands of gates. The proposed method can be used with any variability model despite the fact that in this paper it is illustrated via a two-tier hierarchical model. The yield constraint becomes active as the optimization concludes, which shows that the problem of overdesign in worst-case approach is eliminated.

This paper is organized as follows: the digital circuit sizing problem is formulated in section 2; the SGP method is described in detail in section 3; the simulation results are discussed in section 4.

## 2. Digital circuit sizing

### 2.1 Critical Delay

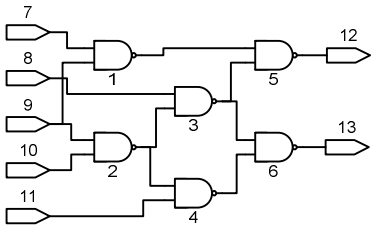


Fig. 1. An example circuit: ISCAS'85 benchmark circuit c17.

The digital circuit sizing problem has been studied by many authors (see for example [5], [9]), and we adopt the model as described in [9]. A combinational digital circuit is composed of a number of gates (e.g. Fig. 1), and the delay associated with gate $i$ is given by [10]

$$D_i = 0.69\frac{R_i}{x_i}\left(C_i^{int}x_i + \sum_{j\in \mathrm{FO}(i)} C_j^{in}x_j\right), \qquad (1)$$

where $i$=1…$m$ with $m$ being the number of gates; $x_i$'s are gate sizes (and $x$, without the index, is the respective column vector); $C_i^{int}$and $C_i^{in}$ are the intrinsic capacitance and input capacitance, respectively, of gate $i$ with unit size; $R_i$ is the unit size equivalent resistance of gate $i$ and it represents the driving strength of the gate; the symbol FO($i$) denotes the indices of the gates that connect to the output of the gate $i$, or

the fan-out of gate *i*. In order to simplify the notation in subsequent derivations, we rewrite (1) as

$$D_i = R_i C_i(x), \tag{2}$$

where

$$C_i(x) = 0.69 \frac{\left(C_i^{int} x_i + \sum_{j \in \mathrm{FO}(i)} C_j^{in} x_j\right)}{x_i}, \tag{3}$$

is a posynomial [11] of $x$. Starting from the input nodes, there are several paths to the outputs. The delay of a path $p$ is the sum of the delays of all gates belonging to the path:

$$D_p = \sum_{i \in \mathrm{path}(p)} D_i = \sum_{i \in \mathrm{path}(p)} R_i C_i(x). \tag{4}$$

The critical delay is the maximum of all path delays:

$$D_{\mathrm{critical}} = \max_p (D_p) = \max_p \left( \sum_{i \in \mathrm{path}(p)} R_i C_i(x) \right). \tag{5}$$

It is well known that the number of paths can grow exponentially as the circuit scales, therefore the path delay is used here only to introduce the concept of critical delay, while the actual calculation is carried out by way of node-based arrival time method as will be described in section 3.1.

In practice, these circuit blocks are placed between flip-flops, and the critical delay must be less than a value set by the clock frequency and the setup time of the flip-flop in order for the circuit to function properly. Suppose that the maximum allowable delay is $D_{\max}$, then the specification that the circuit must meet is $D_{\mathrm{critical}} \le D_{\max}$. We refer to the situation where $D_{\mathrm{critical}} > D_{\max}$ as a *failure event*.

## 2.2 Randomness in the inner optimization step

We assume that randomness comes from the variation in equivalent resistance $R_i$. This may reflect the variability arising from threshold voltage variation, random dopant fluctuation, gate length variation, etc. Suppose that $R_i$ is perturbed around its nominal value $R_i^0$ by a two-tier hierarchical variability [8], [12]:

$$R_i = R_i^0 + R_g \zeta_g + R_i^r \zeta_i, \tag{6}$$

where the global perturbation random variable $\zeta_g \in [-1,1]$; the deterministic constant $R_g \ge 0$ denotes the range of impact of global perturbation; the local perturbation random variables $\zeta_i \in [-1,1]$ are independent of each other and are independent of the global perturbation variable $\zeta_g$; the deterministic constant $R_i^r \ge 0$ characterizes the range of impact of the local perturbation. Information about intra-die spatial variability can be added if gate positions and the form of spatial variability are known.

Notice that the only assumption on $\zeta_g$ and $\zeta_i$ is that they take values on [-1,1]; otherwise we do not assume them to follow any particular distribution. This assumption is physically reasonable, if physical parameters under variation do not deviate too much from their nominal values. The constants $R_g$ and $R_i^r$ can be determined by way of measurements or by deducing them from other physical parameters. In our analysis, we assume that resistance $R_i$ is linked with the threshold voltage by the $\alpha$-power law [13]:

$$R_i \propto \left(V_{dd} - V_{th,i}\right)^{-\alpha}, \tag{7}$$

where $V_{dd}$ is the supply voltage, typically around 1 V; $\alpha$ is a parameter that typically equals 1.3; $V_{th,i}$ is the threshold voltage of gate *i*. We further assume that the threshold voltage $V_{th,i}$ is random and can be described as

$$V_{th,i} = V_{th,i}^0 + \sigma_g y_g + \sigma_i y_i, \tag{8}$$

where $V_{th,i}^0$ is the nominal threshold voltage; $\sigma_g$ characterizes the global fluctuation; $\sigma_i$ characterizes the local fluctuation range; $y_g$ and $y_i$'s are random variables whose distributions can be deduced from data. To find the coefficients $R_g$ and $R_i^r$ in (6), we simply let the global and local random variables in (8) to take their extreme values and estimate the change in threshold voltage, and consequently, in equivalent resistance through $\alpha$-power law. In case where the range of $y_g$ or $y_i$ is infinite, a practically "distant" value can be used instead (e.g. $3\sigma$ value in case of Gaussian). The goal is to obtain a rough estimate of the impact of the variability in threshold voltage on the equivalent resistance, through the coefficients $R_g$ and $R_i^r$. This can be achieved as long as a simulation can be carried out to calculate $V_{th,i}$.

In our example, we assume $y_g$ and $y_i$'s are independent and they follow the standard normal distribution $\mathcal{N}(0,1)$. To estimate the parameter $R_g$ and $R_i^r$, we let $y_g$ and $y_i$'s to take their $3\sigma$ value, i.e. $y_g$, $y_i$ =3, and arrive at the following formula:

$$R_g = R_i^0 \frac{\left(V_{dd} - V_{th,i}^0\right)^\alpha}{\left(V_{dd} - V_{th,i}^0 - 3\sigma_g\right)^\alpha} - R_i^0, \tag{9}$$

$$R_i^r = R_i^0 \frac{\left(V_{dd} - V_{th,i}^0\right)^\alpha}{\left(V_{dd} - V_{th,i}^0 - 3\sigma_i\right)^\alpha} - R_i^0. \tag{10}$$

The typical values of the parameters mentioned in this section are: $V_{dd}$=1 V, $V_{th,i}^0$=0.3 V, $\alpha$=1.3, $\sigma_g$=0.02 V, and $\sigma_i = 0.1\sigma_g$ Putting in the typical values, we have $R_g = 0.124 R_i^0$ and $R_i^r = 0.011 R_i^0$.

In practice, the capacitances also contribute to the variability of the circuit.. Those variations can be dealt with in the same manner since they are no more than another multiplication factor in front of a posynomial of $x$, and they are compatible with the methodology stated below. For simplicity (but without loss of generality), in this paper we only consider the variation arising from $R_i$.

## 2.3 Yield-constrained optimization problem

We assume that the design objective is the area of the circuit $A(x)$, and it is a linear function of the gate size vector $x$:

$$A(x) = a^T x, \tag{11}$$

where $a$ is a constant column vector of which the element denotes the area of the gate with unit size. If we specify that the yield must be greater than $1 - \delta$, then the probability of

the failure event should be no greater than $\delta$. Moreover, due to minimal feature size limitations, the gate sizes cannot be less than the unit value; so $x$ must satisfy $x \geq 1$, where the symbol "≥" is interpreted as "component-wise greater than or equal to". This leads to the problem

$$\begin{aligned} \text{minimize} \quad & A(x) = a^T x \\ \text{subject to} \quad & P_f = \Pr(D_{\text{critical}} > D_{\max}) \leq \delta \\ & x \geq 1, \end{aligned} \tag{12}$$

incorporating a yield constraint. The parameters used in the above formulation are from [9] as summarized in Table 1.

| Gate type | $C^{\text{in}}$ (fF) | $C^{\text{int}}$ (fF) | $R^0$ (kΩ) | $a$ |
|---|---|---|---|---|
| INV | 3 | 3 | 0.48 | 3 |
| NAND2 | 4 | 6 | 0.48 | 8 |
| NOR2 | 5 | 6 | 0.48 | 10 |

Table 1: Circuit parameters from [9].

## 3 Method

The presence of the yield constraint in (12) makes the problem non-convex, thus hard to solve in general. This is an active research area in robust convex optimization itself [14], [15]. One way is to replace the yield constraint with a tractable *robust* (or *safe*) approximation. This section first introduces the box approximation as a basis for the sequential methods that we are going to propose, the sequential geometric programming method is described in section 3.2, and another essential element of the method – confirmation through importance sampling – is explained in section 3.3.

### 3.1 Box approximation and its scaled version

An obvious way of replacing the yield constraint in (12) is to dictate the critical delay to be less than $D_{\max}$ for all possible perturbation values $\zeta_g, \zeta_i \in [-1,1]$. If we view $\zeta_g$ and $\zeta_i$'s as elements of a vector $\zeta = (\zeta_g, \zeta_i \ldots \zeta_m)^T$, then $\zeta$ takes values that is determined by the perturbation set $\mathcal{Z}_{\text{Box}} = \{\zeta : \|\zeta\|_\infty \leq 1\}$. A robust approximation to the yield constraint in (12) becomes

$$\begin{aligned} \max\left( \sum_{i \in \text{path}(p)} \left(R_i^0 + R_g\zeta_g + R_i^r\zeta_i\right) C_i(x) \right) \\ \leq D_{\max}, \\ \forall \zeta \in \mathcal{Z}_{\text{Box}}, p \in \text{All Paths}. \end{aligned} \tag{13}$$

Since $R_g, R_i^r \geq 0$, (13) amounts to

$$\begin{aligned} \sum_{i \in \text{path}(p)} \left(R_i^0 + R_g + R_i^r\right) C_i(x) \leq D_{\max}, \\ \forall p \in \text{All Paths}. \end{aligned} \tag{14}$$

Since the perturbation set $\mathcal{Z}_{\text{Box}}$ has the shape of a box, the approximation (14) is also called a *box* approximation [14]. The box approximation guarantees that the delay requirement is always satisfied under the uncertainty model described in (6); therefore it is a robust but overly conservative approximation of the yield constraint in (12). Since the worst possible situation is included, the design problem (12) with the yield constraint replaced by (14) is indeed a conservative, worst-case design.

Since the approximation deduced from the box perturbation set is more conservative than necessary, we can use a set that is smaller than $\mathcal{Z}_{\text{Box}}$, provided, of course, that the original yield constraint is never violated. One possible way is to scale the size of the box. Suppose that the size of the box is scaled by a scalar factor of $f_{\text{Box}}$, the scaled perturbation set is $\mathcal{Z}'_{\text{Box}} = \{\zeta : \|\zeta\|_\infty \leq f_{\text{Box}}\}$. If we dictate the critical delay to be less than $D_{\max}$ for all the perturbations defined by $\mathcal{Z}'_{\text{Box}}$, then the approximation (14) becomes

$$\begin{aligned} \sum_{i \in \text{path}(p)} \left(R_i^0 + f_{\text{Box}}(R_g + R_i^r)\right) C_i(x) \leq D_{\max}, \\ \forall p \in \text{All Paths}. \end{aligned} \tag{15}$$

We call (15) a *scaled box* approximation. Now the problem becomes determining the appropriate scale factor $f_{\text{Box}}$ that will help us achieve the desired yield without being too conservative. We address this problem by sequential geometric programming as will be described in section 3.2. But before that, (15) deserves further discussion.

The summation that appears in (15) is for all paths. This could pose a computational problem because the number of paths can grow exponentially with the number of gates; hence the number of constraints grows correspondingly. To circumvent this problem, one can adopt the dynamic programming formulation as described here. Suppose we define the delay of gate $i$ as $D_i$, then the maximum arrival time $T_i^D$ of the signal at the output of the gate satisfies the following inequality

$$T_i^D \geq T_j^D + D_i, \qquad \forall j \in \text{FI}(i), \tag{16}$$

where FI($i$) represents the indices of the gates that connect to the input of gate $i$, or the fan-in of gate $i$. In scaled box approximation (15), the term in the summation is the gate delay $D_i = \left(R_i^0 + f_{\text{Box}}(R_g + R_i^r)\right) C_i(x)$, and equation (16) can be written as

$$\begin{aligned} T_i^D \geq T_j^D + \left(R_i^0 + f_{\text{Box}}(R_g + R_i^r)\right) C_i(x), \\ \forall j \in \text{FI}(i). \end{aligned} \tag{17}$$

If we want the summation of the gate delays along all paths to be less than some value $D_{\max}$ as in (15), then we only need to look at the arrival time at the output nodes and set those to be less than $D_{\max}$. We can further simplify this by appending a single artificial output node with zero delay right after the real output nodes, with all real output nodes as its fan-in. If we denote this new artificial node as $o$, we can replace the inequalities involving all paths with a single inequality of $T_o^D$

$$T_o^D \leq D_{\max}. \tag{18}$$

The complete problem with scaled box approximation reads

$$\begin{aligned} \text{minimize} \quad & A(x) = a^T x \\ \text{subject to} \quad & T_i^D \geq T_j^D + \left(R_i^0 + f_{\text{Box}}(R_g + R_i^r)\right) C_i(x), \\ & \qquad \forall j \in \text{FI}(i) \\ & T_o^D \leq D_{\max} \\ & x \geq 1. \end{aligned} \tag{19}$$

This problem is recognized as a geometric programming problem and can solved very efficiently by off the shell solvers [11]. It should be obvious, however, that we have adopted many approximations in order to get to this point. These approximations, involving the shapes of the various distributions and the scaling factor $f_{\text{Box}}$ are only practical within the "inner" optimization step of our approach. An "outer" step that involves verification using the actual variability model is necessary and this is described next.

### 3.2 Sequential geometric programming

Let us denote the problem (19) as $p(f_{\text{Box}})$. If $f_{\text{Box}}$=1, then $p(1)$ is the conservative worst-case problem. If $f_{\text{Box}}$=0, the solution to the problem $p(0)$ is a risky design solution because no randomness is taken into account, and the resulting failure probability $P_f$ is bigger than any $\delta$ that one might be interested in. Based on these observations, we conclude that the best $f_{\text{Box}}$ that can make $P_f$ close to $\delta$ is bounded in [0,1]. Therefore, we can iteratively tune the scale factor $f_{\text{Box}}$, solve the associated problem $p(f_{\text{Box}})$, and, through simulation, calculate the failure probability $P_f$ corresponding to the solution at each iteration. If $P_f$ is close enough to $\delta$, then the algorithm exits; if not, a new value of $f_{\text{Box}}$ is generated for the next iteration.

The whole process is a line search for the best $f_{\text{Box}}$. The line search can be any generic line search algorithm, and we use bisection in our examples. The notion "close enough" will become concrete in section 4.3. Because the optimization problem being solved at each iteration is a geometric programming problem (19), we call this method sequential geometric programming (SGP).

In the current implementation of SGP, we use a single scalar parameter $f_{\text{Box}}$ to describe the set from which $\zeta$'s can take values. This limits the shape of the set. More parameters can be introduced to relax this limitation at the expense of more outer iterations.

It is understood that the geometric programming problem can be solved very efficiently in polynomial time as has been shown theoretically and observed experimentally [11]; the remaining question of SGP is how to efficiently calculate the failure probability. This could be done by direct Monte Carlo (MC) simulation which is robust and well understood. Unfortunately, the number of MC simulation runs can be prohibitively large if the failure probability is low (with the requisite requirement of high precision in its estimation). This problem can be overcome by adopting a variance reduction technique such as importance sampling [16-18]. The next subsection 3.3 explains the principle, and shows that the number of runs to achieve a pre-defined accuracy exhibits weak, if any, dependence on the probability being estimated.

### 3.3 Importance sampling

Suppose that the random variable is $y = (y_g, y_1 \dots y_m)^T \in \mathbb{R}^{m+1}$ as those described in (8), with a probability density function $g(y)$. The failure event that we are interested in is $\{y: D_{\text{critical}}(y) > D_{\max}\}$. Our objective is to evaluate $P_f$ = Prob($\{y: D_{\text{critical}}(y) > D_{\max}\}$). In MC [19], a random sequence of length $R$, $\{y_r\}_{r=1}^R$ is generated according to $g(y)$, and $D_{\text{critical}}$ is evaluated for all the $y_r$'s. $P_f$ is estimated as

$$\hat{P}_f = \frac{1}{R}\sum_{r=1}^{R} 1(D_{\text{critical}}(y_r) > D_{\max}), \tag{20}$$

where $1(D_{\text{critical}}(y_r) > D_{\max})$ is the indicator function of the failure event. The variance of the estimate is

$$\text{Var}(\hat{P}_f) = \frac{1}{R}(P_f - P_f^2). \tag{21}$$

In order to achieve acceptable accuracy, the standard deviation of the estimate must be small compared to the estimate itself. If the standard deviation is required to be less than $kP_f$, then the required number of runs $R$ is

$$R = \frac{1}{k^2}\frac{1-P_f}{P_f}. \tag{22}$$

It is evident from (22) that the number of runs to achieve certain accuracy will increase dramatically as $P_f$ becomes close to zero. The reason is that most of the runs are "wasted" in the region where we are not interested in.

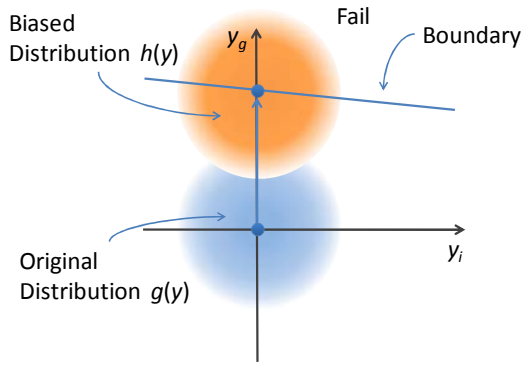


Fig. 2. Illustration of importance sampling in the context of local and global variability. The random sample is drawn from the biased distribution $h(y)$, which has a decent number of points falling in the region of interest.

Importance sampling [16-18] as a variance reduction technique was invented to circumvent this difficulty by biasing the original distribution such that there are sufficient number of trials falling into the important region, which in this case is the region close to the boundary. This is illustrated by the biased distribution $h(y)$ in Fig. 2. In importance sampling, the random sequence $\{y_r\}_{r=1}^R$ is generated according to the biased distribution $h(y)$, and

$$\hat{P}_f = \frac{1}{R}\sum_{r=1}^{R} w(y_r)1(D_{\text{critical}}(y_r) > D_{\max}), \tag{23}$$

with $w(y_r) = g(y_r)/h(y_r)$ as the unbiased estimator of $P_f$. The variance of this estimate is given by

$$\text{Var}(\hat{P}_f) = \frac{1}{R^2}\sum_{r=1}^{R}\left(w^2(y_r)1\left((D_{\text{critical}}(y_r) > D_{\max})\right) - RP_f^2\right). \tag{24}$$

If we want to achieve the same accuracy as that in (22), we can stop the simulation when the following criterion is met:

$$\frac{\sqrt{\mathrm{Var}(\hat{P}_f)}}{\hat{P}_f} \le k. \tag{25}$$

With a proper choice of $h(y)$, the variance given by (24) can be small, thus reducing the number runs required by (25).

The variance reduction achieved in importance sampling strongly depends on the choice of the biased distribution $h(y)$. Ideally, one wants to shift $g(y)$ to the place where failure is most likely to happen, or equivalently the point on the boundary that is closest to the center of the original distribution. The search for such points in a $(m+1)$-dimensional space, where m (the number of gates) could be in the thousands, can be extremely time consuming. The situation can become even worse when there are many vertices in the boundary due the fact that many paths are nearly critical, and small perturbations can cause the critical delay to shift from one path to another. Fortunately, the scaled hierarchical model (8) shows that the critical delay grows most rapidly along the direction of the global variability axis $y_g$ (since global variation is typically much larger than local), therefore the boundary of interest is nearly parallel to the local variation axes $y_i$. We can therefore largely ignore the boundary points on all local variation axes. Consequently, the closest boundary can be approximately found by shifting $g(y)$ along $y_g$ axis by an amount $y_g^{\text{shift}}$ to the boundary as shown in Fig. 2.

Following the model in (8), the original distribution is

$$g(y) = C \exp\left(-\frac{\sum_{i=1}^{m} y_i^2}{2}\right) \exp\left(-\frac{y_g^2}{2}\right), \tag{26}$$

and the biased distribution is

$$h(y) = C \exp\left(-\frac{\sum_{i=1}^{m} y_i^2}{2}\right) \cdot \exp\left(-\frac{\left(y_g - y_g^{\text{shift}}\right)^2}{2}\right), \tag{27}$$

where $C$ is a normalization constant. The function $w(y)$ as defined in (27) is

$$w(y) = \frac{g(y)}{h(y)} = \frac{\exp\left(-\frac{y_g^2}{2}\right)}{\exp\left(-\frac{\left(y_g - y_g^{\text{shift}}\right)^2}{2}\right)}. \tag{28}$$

The shift amount $y_g^{\text{shift}}$ is found using the Newton-Raphson method, where the iteration follows the following formula:

$$y_{g(n+1)} = y_{g(n)} - \frac{D_{\text{critical}}\left(y_{g(n)}\right) - D_{\max}}{\frac{\partial D_{\text{critical}}\left(y_{g(n)}\right)}{\partial y_{g(n)}}}, \tag{29}$$

and the stopping criterion of the search for $y_g^{\text{shift}}$ is

$$\left|D_{\text{critical}}\left(y_{g(n)}\right) - D_{\max}\right| \le 0.01 \text{ ps}. \tag{30}$$

Once $y_g^{\text{shift}}$ is found, hence the biased distribution $h(y)$ is determined, the failure probability is estimated by sampling at the biased distribution according to (23). It is shown in the next section that, with $k$=5%, the total number of simulations required in importance sampling is around $10^3$, almost independent of the probability that is being estimated.

## 4 Results and discussion

We first show the efficiency of geometric programming and importance sampling, which in turn legitimize the idea of SGP which combines these two. SGP is applied to several sample problems showing the capability of solving yield-constrained digital circuit sizing problem with thousands of gates. The simulations are done on a desktop with 2.4 GHz Intel Core 2 Quad processor and Matlab 7.4.0. The geometric programming problem is modeled and solved using the Matlab toolbox of MOSEK [20].

### 4.1 Geometric programming result

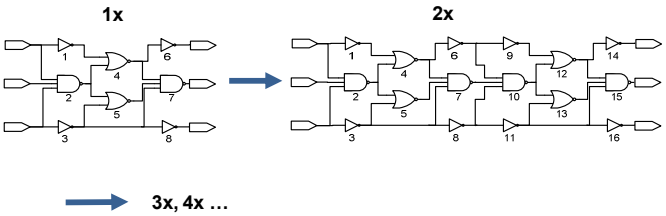


Fig. 3. Circuit cascade. The basic circuit is modified from the one used in [9].

To test how the geometric programming formulation of the circuit sizing problem can be solved efficiently, we form a test circuit and cascade it as shown in Fig. 3. As a result of cascading, the number of gates in the circuit grows linearly with the number of stages being cascaded. The parameter $D_{\max}$ is set to $N_{\text{stage}}$×12.5 ps, where $N_{\text{stage}}$ is the number of stages. The problem being solved is (19) with $f_{\text{Box}}$=1. The number of stages is changed from 1 to 180, and the number of gates changes from 6 to 1080. As shown in Fig. 4, the run-time to solve the geometric programming problem (19) grows roughly linearly with the number of gates. But the exact run-time can heavily depend on the architecture of the circuit, and can also depend on how the problem is formulated and processed in the solver. It is clearly seen that the run-time is at the order of seconds even for a circuit with thousands of gates.

### 4.2 Importance sampling result

Using Monte Carlo-based simulation to estimate probability, the accuracy is controlled by the parameter $k$ as defined in (25). For the rest of this paper, $k$ is set to 5%. We apply the importance sampling method outlined in section 3.3 to a sample circuit shown in Fig. 1. The circuit under test is set so that the size of all its gates is unit ($x$=1). For the purpose of comparison, standard MC is also carried out. The parameter $D_{\max}$ is changed from 14.8 ps to 16.2 ps. As a result of the increase in $D_{\max}$, the failure probability is going to drop. Both importance sampling and standard MC give the same result

as shown in the upper panel of Fig. 5. However, the number of simulation runs in Monte Carlo to achieve the accuracy set by $k$=5% grows exponentially, whereas the number of simulation runs in importance sampling to achieve the same accuracy shows much weaker, if any, dependence on $D_{\max}$, or equivalently on the probability being estimated. The number of simulation runs of importance sampling is roughly 1000. This can be understood by the following crude argument: the importance sampling biases the original distribution such that the effective $P_f$ under the biased distribution is around 0.5, making the term $P_f/(1-P_f)$ close to one; to achieve the accuracy determined $k$=5%, the number of runs determined by (22) is at the order of $10^3$. Even though the formula (22) is for pure MC, it still gives an approximate estimate of the required number of runs for importance sampling, since the method of importance sampling is indeed MC with a biased distribution.

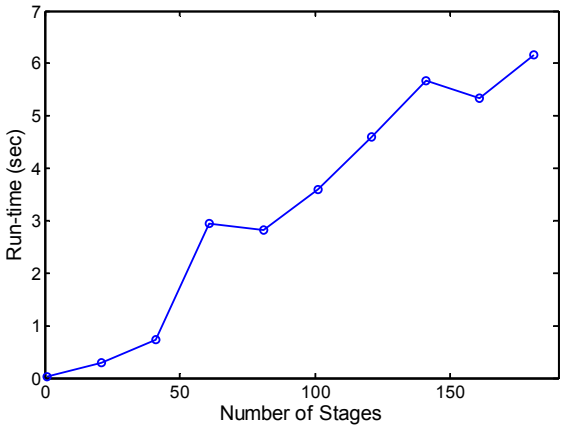


Fig. 4. GP Run-time vs. the number of stages in cascaded circuits. The run-time shows roughly a linear dependence of the size of the circuit. But the exact run-time depends on topology, formulation details.

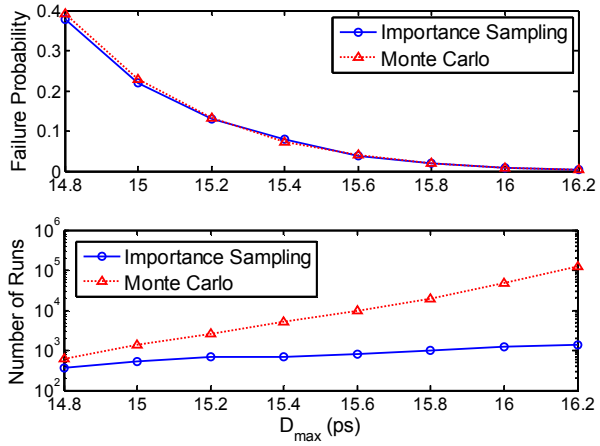


Fig. 5. Importance sampling and Monte Carlo simulation on unit-sized sample circuit shown in Fig. 1. Upper panel: the estimated failure probability vs. $D_{\max}$. Lower panel: number of runs vs. $D_{\max}$. The $k$ parameter as defined in (22) and (25) which determines the accuracy is set to 5%.

### 4.3 SGP result

Based on the results in 4.1 and 4.2, we observe that geometric programming can be solved very efficiently, and that the simulation runs required by importance sampling to estimate failure probability do not grow catastrophically large when the probability to be estimated is small. These observations show it is possible to combine these two elements in the iterative SGP method.

Due to the fact that the failure probability $P_f$ is estimated via stochastic methods, thus the estimate is random in itself, it would be unreasonable to demand the final $P_f$ to be infinitely close to $\delta$. Therefore, the notion "close enough to $\delta$" can be defined to be a region that is acceptable in practice. In our simulation, we use

$$P_f \in [\delta - 0.1\delta, \delta + 0.1\delta]. \qquad (31)$$

We apply both SGP and the worst-case approach to an 8-bit adder [21]. The parameter $\delta$ is set to 0.01, corresponding to 99% yield, and $D_{\max}$ is set to 70 ps. After the optimization, a standard 50,000-run MC simulation is applied to the optimized circuits to visualize the delay distribution, where the random numbers are generated according to (8). The simulated histogram is shown in the upper panel of Fig. 6, and the optimized circuit area is shown in the lower panel of Fig. 6. It is clearly seen that worst-case approach ends up with a solution of which the failure probability is much less than necessary; this comes with an area penalty. SGP gives better result by pushing the delay distribution closer to the success/failure boundary, such that the failure probability is very close to $\delta$; thus the yield constraint is active.

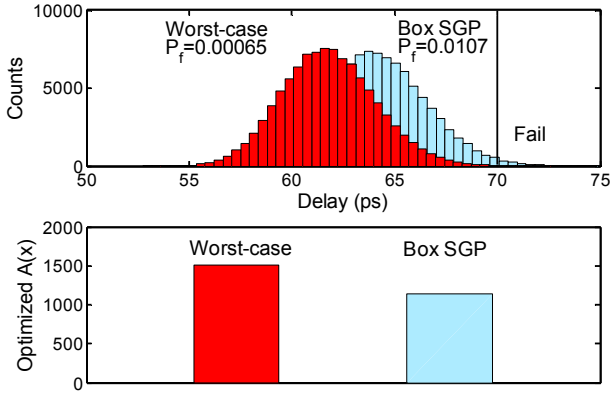


Fig. 6. SGP and worst-case approach on 8-bit adder. Upper panel: delay histogram from Monte Carlo simulation on the optimized circuits. Lower panel: optimized area given by different methods.

To see how the method behaves at different $\delta$'s, we change $\delta$ from $10^{-3}$ to $10^{-1}$, a typical tradeoff curve of optimized area vs. $\delta$ is shown in Fig. 7. It is interesting to see how many SGP iterations are needed to reach the stopping criterion in (31). The total run-time and SGP iterations at different $\delta$'s are shown in Fig. 8. It is seen that the total run-time is only dozens of seconds for this circuit, and the number of SGP iterations never exceeds 4 throughout the entire $\delta$ range that we simulated.

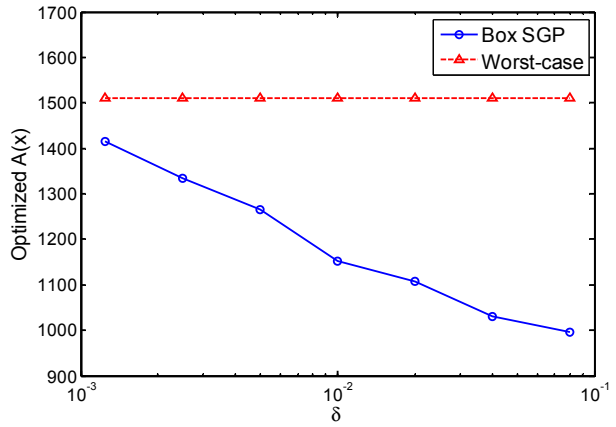


Fig. 7. Tradeoff curve of optimized area vs $\delta$.

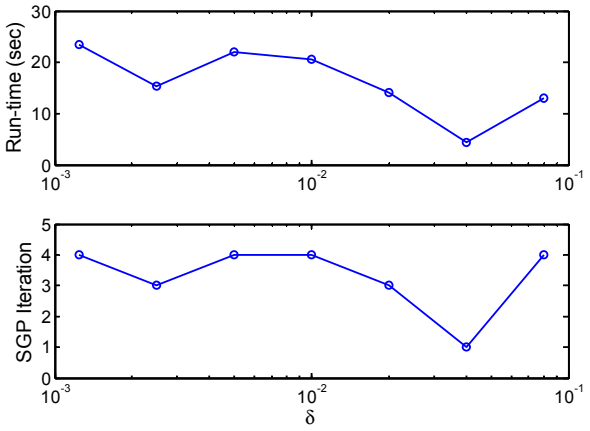


Fig. 8. Upper panel: runtime at different $\delta$'s. Lower panel: number of SGP iterations at different $\delta$'s.

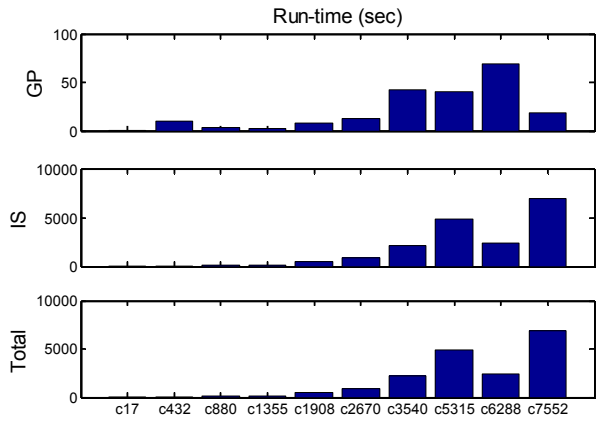


Fig. 9. Run-time breakdown for ISCAS'85 benchmark circuits.

We also applied the method to the ISCAS'85 benchmark circuit families, of which the largest circuit c7552 has 3,512 gates. The parameter $\delta$ is set to 0.01, and $D_{\max}$ is set to $N_{\text{logic}}\times 12.5$ ps, where $N_{\text{logic}}$ is the logic depth of the circuit. The run-time is summarized in Fig. 9. The total time to solve the problem is no more than a few hours. Compared to geometric programming, the importance sampling simulation takes most of the run-time. This is expected since importance sampling involves large number of repetitions and is implemented completely in Matlab which may add significant overhead. A better implementation exploiting the nature of the repetitive runs or even using a parallel architecture would have been far more efficient. However, even with the present code, the run-time of importance sampling can be well controlled as shown in Fig. 10. In Fig. 10, the number of simulation runs for importance sampling in each iteration is averaged and plotted for each benchmark circuit. The average simulation runs is around $10^3$, which conforms to the argument in section 3.3 that the simulation runs should be at the order of $10^3$ for $k$=5%. For the number of SGP iteration, as shown in Fig. 11, it is again below or equal to 4 for all the benchmark circuits we test.

We have explored several other formulations to handle the yield constraint, including the Ball and Bernstein approximations, etc [14]. These approximations work well if the perturbations are identically independently distributed for all gates. However, in reality the perturbation is hierarchical in nature as described in this paper, and this makes the coefficient in front of $\zeta_g$ in (13) significantly larger than those in front of $\zeta_i$'s. The total effect is that the perturbation is equivalently low-dimensional, making an approximation scheme such as Ball approximation not effective. One way to mitigate this problem in yield-constrained linear programming is to use the perturbation set that corresponds to the intersection of a ball and a box, but this formulation gives rise to constraints that are not consistent with geometric programming; therefore it is not readily convertible to a convex form, making it troublesome to solve.

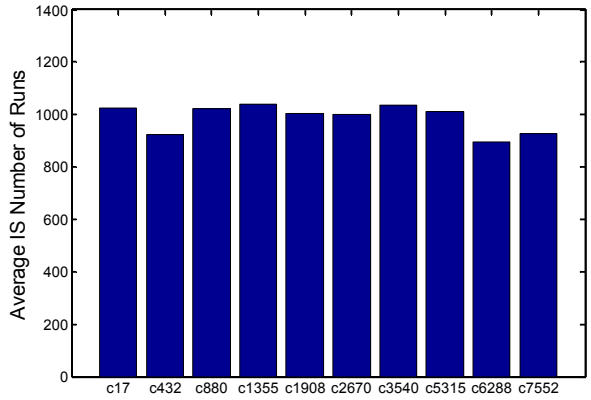


Fig. 10. Average number of runs for importance sampling at each iteration.

## 5 Conclusions

The method of sequential geometric programming (SGP) is introduced to solve yield-constrained digital circuit sizing problem. The proposed method consists of iterative usage of geometric programming and importance sampling. The fact that the yield constraint is handled by simulation makes the approach applicable to any variability model. The number of simulation runs in importance sampling is shown to be well controlled; the few SGP iterations are generally needed to achieve practical convergence. The method is applied to 8-bit adder and also ISCAS'85 benchmark circuit families. All of

them can be solved within reasonable amount of time using the proposed method.

Several aspects of the proposed method can be improved to further speed up the method. The geometric programming can be solved using large-scale technique such as preconditioned conjugate gradient [22], which allows the optimization problem in the inner loop of SGP to handle even larger problems. The importance sampling for probability estimation, if implemented within a programming tool, can be much faster provided the number of runs is well controlled.

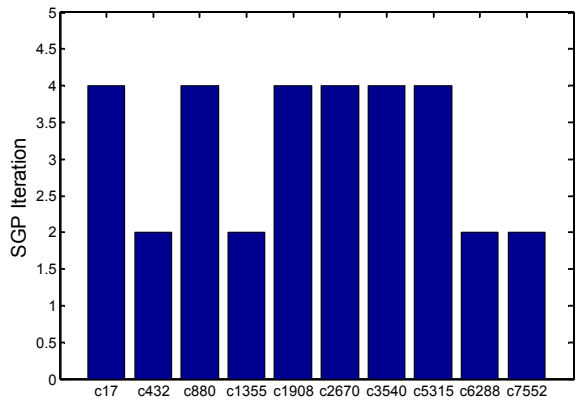


Fig. 11. The number of SGP iterations for ISCAS'85 benchmark circuits.

## Acknowledgements

This work is supported by the Integrated Modeling Process and Computation for Technology (IMPACT) project, which is funded by AMD, Applied Materials, ASML, Canon, Ebara, Hitachi, IBM, Intel, KLA-Tencor, Magma, Marvell, Mentor Graphics, Novellus, Panoramic, SanDisk, Spansion, Synopsys, Tokyo Electron Limited, and Xilinx, with donations from Photronics, Toppan and matching support by the U.C. Discovery Program.

## References

[1] L. Lieberman, "DfM, the teenage years," Proc of SPIE Vol. 6925, 692502 (2008)

[2] G. S. May and C. J. Spanos, "Semiconductor manufacturing and process control," John Wiley & Sons, Hoboken NJ (2006)

[3] M. D. M. Hershenson, S.P. Boyd and T. H. Lee, "Optimal design of a CMOS Op-Amp via geometric programming," IEEE Transactions on Computer-aided design of integrated circuits and systems, Vol. 20, No.1, (2001)

[4] X. Liu, W. Luk, Y. Song, P. Tang and X. Zeng, "Robust analog circuit sizing using ellipsoid method and affine arithmetic," ASP-DAC (2007)

[5] M. Mani and M. Orshansky, "A new statistical optimization algorithm for gate sizing," IEEE International Conference on Computer Design (2004)

[6] J. Singh, V. Nookala, Z. Luo and S. Sapatnekar, "Robust gate sizing by geometric programming," DAC (2005)

[7] E. Morifuji, D. Patil, M. Horowitz and Y. Nishi, "Power optimization for SRAM and its scaling," IEEE Trans. Electron Devices vol. 54, No. 4 (2007)

[8] K. Qian, B. Nikolic, and C. J. Spanos, "Hierarchical modeling of spatial variability with a 45nm example," Proc of SPIE, 727505 (2009)

[9] S. P. Boyd, S.-J. Kim, D. D. Patil, and M. A. Horowitz, "Digital circuit optimization via geometric programming," Oper. Res., vol. 53, no. 6, pp. 899–932 (2005)

[10] J. M. Rabaey, A. Chandrakasan and B. Nikolic, "Digital integrated circuits: a design perspective," second edition, Prentice Hall (2003)

[11] S. Boyd and L. Vandenberghe, "Convex optimization," Cambridge University Press (2004)

[12] P. Friedberg, W. Cheung, G. Cheng, Q. Y. Tang and C.J. Spanos, "Modeling spatial gate length variation in the 0.2um to 1.15mm separation range," SPIE Vol. 6521, 652119 (2007)

[13] T. Sakurai and A. Newton, "Alpha-power law MOSFET model and its application to CMOS inverter delay and other formulas," IEEE Journal of Solid-State Circuits 25(2), pp. 584–593 (1990)

[14] A. Ben-Tal, L. El Ghaoui and A. Nemirovski, "Robust Optimization," Princeton University Press (2009)

[15] A. Nemirovski and A. Shapiro, "Convex approximations of chance constrained programs," SIAM J. on Optimization, vol. 17, no. 4, pp. 969-996 (2006)

[16] R. Srinivasan, "Importance sampling: applications in communications and detection," Springer (2002)

[17] R. Kanj, R. Joshi and S. Nassif, "Mixture importance sampling and its application to the analysis of SRAM designs in the presence of rare failure events," DAC, 69-72 (2006)

[18] L. Dolecek, M. Qazi, D. Sha and A. Chandrakasan, "Breaking the simulation barrier: SRAM evaluation through norm minimization," ICCAD (2008)

[19] A. C. Davison, "Statistical models," Cambridge University Press, (2008)

[20] [Online] Available: http://www.mosek.com

[21] [Online] Available: http://www.pld.ttu.ee/testing/labs/adder.html

[22] S. Joshi and S. Boyd, "An efficient method for large-scale gate sizing," IEEE Tran. on Circuits and Systems I, 55(9): 2760–2773, (2008)