A Heuristic for Optimizing Stochastic Activity Networks with Applications to Statistical Digital Circuit Sizing

Seung-Jean Kim[1] Stephen P. Boyd[1] Sunghee Yun[1]
Dinesh D. Patil[2] Mark A. Horowitz[2]

Feburary 2005

[1]Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA 94305. ({sjkim,boyd,sayunint}@stanford.edu)

[2]Computer Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA 94305. ({ddpatil,horowitz}@stanford.edu)

Abstract

A deterministic activity network (DAN) is a collection of activities, each with some duration, along with a set of precedence constraints, which specify that activities begin only when certain others have finished. One critical performance measure for an activity network is its makespan, which is the minimum time required to complete all activities. In a stochastic activity network (SAN), the durations of the activities and the makespan are random variables. The analysis of SANs is quite involved, but can be carried out numerically by Monte Carlo analysis.

This paper concerns the optimization of a SAN, *i.e.*, the choice of some design variables that affect the probability distributions of the activity durations. We concentrate on the problem of minimizing a quantile (*e.g.*, 95%) of the makespan, subject to constraints on the variables. This problem has many applications, ranging from project management to digital integrated circuit (IC) sizing (the latter being our motivation). While there are effective methods for optimizing DANs, the SAN optimization problem is much more difficult; the few existing methods cannot handle large-scale problems.

In this paper we introduce a heuristic method for approximately optimizing a SAN, by forming a related DAN optimization problem which includes extra margins in each of the activity durations to account for the variation. Since the method is based on optimizing a DAN, it readily handles large-scale problems. To assess the quality of the resulting suboptimal designs, we describe two widely applicable lower bounds on achievable performance in optimal SAN design.

We demonstrate the method on a simplified statistical digital circuit sizing problem, in which the device widths affect both the mean and variance of the gate delays. Numerical experiments show that the resulting design is often substantially better than one in which the variation in delay is ignored, and is often quite close to the global optimum (as verified by the lower bounds).

Key words: activity network, circuit optimization, design centering, design for manufacture, design for yield, makespan, project network, robust optimization, statistical circuit design.

1 Introduction

A deterministic activity network (DAN) is a collection of activities (jobs or tasks), each with some duration or processing time, along with a set of precedence constraints which specify that activities can begin only when others have finished. The minimum time required for all activities to finish is called the *makespan* of the network. In a stochastic activity network (SAN), the durations of the activities are random variables, as is the makespan. Activity network models have been used in many fields, *e.g.*, project management [29, 34, 35], parallel processing systems [77], and statistical timing analysis of digital circuits [5, 51]. In digital circuit analysis, the durations are usually called delays, and the makespan is called the *cycle time* or *total delay* of the circuit.

This paper concerns the optimization of a SAN, *i.e.*, the choice of some design variables (or allocation of some resources) which affect the durations of the activities in a DAN, or their probability distributions in a SAN. While there are many possible choices of objectives in a SAN, we concentrate on a typical one, an upper quantile (*e.g.*, 95%) of the makespan. This corresponds to risk-averse allocation of resources, or to a *robust design* in a design context. Designing a circuit that meets a timing specification with high probability, despite statistical variation, is called *statistical circuit design*. It is closely related to *design for yield* (DFY), *design for manufacture* (DFM), and *design centering*.

Deterministic and stochastic activity network optimization problems arise in many applications, ranging from project management [72, 84, 86] to digital circuit sizing [22]. There are many exact or direct methods for solving DAN optimization problems, even for large-scale problems. There are far fewer methods for solving SAN optimization problems, and these are computationally far more demanding (or intractable), even for small problems. Moreover, the designs obtained depend on details of the duration distributions, which are often not well known in practical applications.

In this paper we introduce a heuristic method for approximately solving a SAN optimization problem, by reducing it to a related DAN optimization problem which includes extra margins in each of the durations to account for the variation. Since the method is based on optimizing a DAN, it readily handles large-scale problems. To assess the quality of the resulting suboptimal designs, we describe two widely applicable lower bounds on achievable performance in optimal SAN design.

We demonstrate the method on a simplified statistical digital integrated circuit (IC) sizing problem, in which the device widths affect both the mean and variance of the gate delays. Numerical experiments show that the resulting designs are often substantially better than one in which the variation in delay is ignored, and often quite close to the global optimum. Our example problem uses simplified models for delay and parameter variation, since our goal in this paper is to describe the basic method, algorithms, and analysis. We have also applied our method to circuit design problems with more complex models, and have found similar improvements in the designs (verified also by high fidelity circuit simulation). These results are reported in [69].

While our focus in this paper is not on the details of statistical digital circuit sizing, we make a few comments about its growing importance. For current IC technologies, statistical

uncertainty and process variation can be indirectly handled by incorporating simple margins in the timing and other critical constraints, or by a post-design step that does centering or yield improvement; see, *e.g.*, [1, 2, 7, 55, 64, 87, 96]. As device dimensions shrink, however, growing (relative) statistical uncertainty and process variation will require an approach where design and yield optimization are combined [48, 92, 95]. We believe that the method described here is good candidate for such a method.

This paper is organized as follows. We describe the deterministic activity network model in §2, and the stochastic activity network model in §3. The corresponding optimization problems are covered in §4. In §5, we introduce our heuristic method for approximately solving the SAN optimization problem and give some lower bounds on achievable performance in SAN optimization. In §6, we illustrate the heuristic method on a simplified statistical digital circuit sizing problem. We give the details of our statistical circuit design example in §7, and further numerical results, in §8, showing that our method is quite insensitive to model assumptions. We give our conclusions in §9.

2 Activity networks

We consider a set of n activities, labeled $1, \ldots, n$, which are carried out under some given precedence constraints among them, *i.e.*, each activity starts only when certain others have finished. We represent the precedence constraints by a directed acyclic graph (DAG) $G = (N, A)$, called the *precedence graph*, with nodes $N = \{1, \ldots, n\}$ and arcs $A \subseteq \{(i, j) : i, j = 1, \ldots, n\}$. Each arc in the precedence graph represents a precedence constraint: $(i, j) \in A$ means that activity j cannot start until activity i ends. For each activity $k \in N$, we denote the set of its *predecessor nodes* (called *fan-in* in IC design) as $\mathbf{Pred}(k)$, and the set of its *successor nodes* (called *fan-out* in IC design) as $\mathbf{Succ}(k)$:

$$\mathbf{Pred}(k) = \{i \in N : (i, k) \in A\}, \qquad \mathbf{Succ}(k) = \{j \in N : (k, j) \in A\}.$$

Input (also called *source* or *starting*) nodes are those with no predecessors, and output (*sink* or *ending*) nodes are those with no successors. We let $d_i > 0$ denote the duration (or processing time or delay) of activity i. The pair (G, d), *i.e.*, a DAG along with a duration for each node, is referred to as an *activity network*.

2.1 Path delay, completion time, and makespan

A path on the DAG G represents a sequence of activities that must occur in order. Suppose p is a path of length k, *i.e.*, $p = (i_1, i_2, \ldots, i_k)$, where $(i_j, i_{j+1}) \in A$ for $j = 1, \ldots, k-1$. We will denote the length of the path as $l(p) = k$. The *duration* or *delay* of p is the sum of the durations of the activities on the path, and is denoted $d(p)$:

$$d(p) = d_{i_1} + \cdots + d_{i_k}.$$

For activity i we define its *completion time* c_i as the maximum of the durations of all paths finishing at i:

$$c_i = \max\{d(p) \ : \ p \text{ is a path finishing at } i\}.$$

The completion time can be interpreted as follows. Suppose the input activities begin at time $t = 0$, and all other activities begin as soon as all of their predecessor activities have finished. Then the completion time c_i defined above is exactly the time at which activity i finishes.

The maximum of the durations of all paths, which is also the maximum of the completion times for all activities, is called the *makespan* of the activity network, and is denoted by $c_{\max}$:

$$c_{\max} = \max\{d(p) \ : \ p \in \mathcal{P}\} = \max_{i \in N} c_i,$$

where $\mathcal{P}$ denotes the set of all paths in the DAG. The makespan is also called the total delay, total processing time, or total duration. The makespan is exactly the total time for all activities to complete, when the starting activities start at $t = 0$, and each activity starts when its predecessors have finished. To indicate that the completion times and makespan are functions of the durations, we sometimes write $c_i(d)$ and $c_{\max}(d)$.

Simple examples show that an activity network with n nodes can have a total number of paths that grows exponentially with n. This suggests that computing the completion times c_i or makespan $c_{\max}$ might be difficult. Using a simple recursion, however, we can evaluate the completion times and the makespan efficiently, without enumerating all the paths. We start with the source nodes and assign $c_i = d_i$ for them. We then choose an ordering of the nodes that respects the ordering on the DAG (*i.e.*, one that takes a node after all its predecessors), and successively set

$$c_i = d_i + \max\{c_j \ : \ j \in \mathbf{Pred}(i)\}.$$

In other words, we compute the completion time of activity i as the maximum of the completion times of its predecessors, plus its own duration. Once we have found all the completion times, we can compute the makespan as

$$c_{\max} = \max_{i \in N} c_i = c_{\max} = \max\{c_i \ : \ i \text{ a sink node}\}. \tag{1}$$

The number of operations required for this recursion grows linearly in the total number of nodes and arcs.

2.2 Critical paths and activities

Any path with duration $c_{\max}$ is said to be *critical*, *i.e.*, it is a longest path. The difference between the makespan and its duration, $c_{\max} - d(p)$, gives a measure of how close to critical the path is: if the difference is zero the path is critical; if it is small we say it is nearly critical; if it is large, the path is noncritical.

An activity is said to be critical if it is on a critical path. A common measure of criticality of an activity i is its *float time*, which is the amount we can increase d_i without affecting the makespan [73]. An activity is critical if and only if its float time is zero.

The recursion for computing c_i and $c_{\max}$ described above is readily modified to yield a critical path, by keeping track of a predecessor node which determines the completion time for each node. We then choose an output node whose completion time is equal to the makespan, and follow it backward through the predecessors to an input node. This path is critical. Thus, we can efficiently find a critical path, even for a large network with an enormous number of paths.

More generally, we can mark for each node *all* the predecessors which determine the completion time for the node. The marked edges and nodes determine a subgraph of the original DAG. We then delete edges and nodes in the subgraph that are not connected to an input or output node. The resulting DAG gives *all* critical paths in the network. (Since there can be an enormous number of critical paths, we may wish to leave the set of critical paths described as a subgraph, and not enumerate its paths.) A more sophisticated method can be used to identify the k most critical paths [36].

2.3 Activity network representation of digital circuits

In the remainder of this section we show how activity networks can be used to (approximately) model delay in a digital combinational logic circuit. We consider a circuit consisting of n gates labeled $1, \ldots, n$. We associate each gate with a node of an activity network; the dependency DAG $G = (N, A)$ with $N = \{1, \ldots, n\}$ is simply the signal flow graph of the combinational logic circuit. In an accurate static timing analysis of a real digital circuit, we would distinguish between rising and falling delays at the output of each gate, use different delays for each gate input/output transition, and take into account effects of signal slopes, distributed loads, false paths, and so on. (Extensions of activity network models can easily handle these effects.) But to keep things simple, we consider here a single delay for each gate. The delay d_i represents the delay of gate i, from any of its inputs to any of its outputs, including the effects of capacitive loading from wires and fanout gates. With these simplifying assumptions, the circuit can be represented by the DAN (G, d). The makespan of the DAN is called the *cycle time*, since it is the maximum delay of the combinational logic block, from the time its primary inputs become valid until all outputs are valid. The cycle time tells us the maximum speed at which the clock can operate, assuming the inputs to the combinational logic circuit are driven from latches and the outputs drive latches.

As a very simple example, consider a chain of four inverters, shown in figure 1. We represent this by the series network, with four activities, as shown in figure 1. For this simple example, there is only one path, all activities are critical, and the cycle time is the sum of the four gate delays: $c_{\max} = d_1 + \cdots + d_4$.

A slightly more complicated example, with two inverters, two NAND, and two NOR gates, is shown in figure 3. This combinational logic block can be represented by the activity network shown in figure 4. This circuit (or activity network) has 6 paths; its cycle time $c_{\max}$

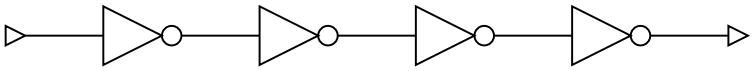

Figure 1: A chain of 4 inverters.

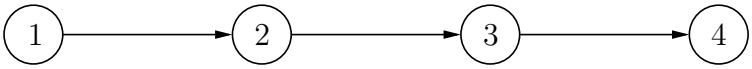

Figure 2: Activity network representation of the inverter chain in figure 1.

can be expressed as

$$c_{\max} = \max\{y_1, \ldots, y_6\},$$

where the path delays are

$$\begin{aligned} y_1 &= d_1 + d_4 + d_6, \\ y_2 &= d_1 + d_4 + d_7, \\ y_3 &= d_2 + d_4 + d_6, \\ y_4 &= d_2 + d_4 + d_7, \\ y_5 &= d_2 + d_5 + d_7, \\ y_6 &= d_3 + d_5 + d_7. \end{aligned}$$

3 Stochastic activity networks

A stochastic activity network (SAN) is an activity network with random activity durations. A SAN is described by the pair $(G, \mathbf{D})$, where the DAG $G = (N, A)$ is the precedence graph, and $\mathbf{D} = (\mathbf{D}_1, \ldots, \mathbf{D}_n)$ is the vector of random activity durations. In a SAN, all quantities that depend on the activity durations are random variables. In particular, the delay of any path, completion time at any node, and the makespan are all random variables. We will denote the makespan (random variable) as $\mathbf{C}_{\max}$.

Stochastic activity networks were originally developed in the late 1950s for analyzing project management and scheduling [71]. Most work on project management has focused on project scheduling techniques such as *project evaluation and review techniques* (PERT) and the *critical path method* (CPM) (see, *e.g.*, [29, 34, 35]).

We define the *criticality index* of a path as the probability that it is critical, *i.e.*, that its delay is greater than or equal to that of every other path (see, *e.g.*, [18, 33, 39]). In a similar way, the criticality index of an activity is defined as the probability that a critical path passes through the activity; it is the sum of the criticality indices of the paths containing the activity. Identifying activities and paths with large criticality indices is a well studied problem (see, *e.g.*, [18, 25, 31, 56]).

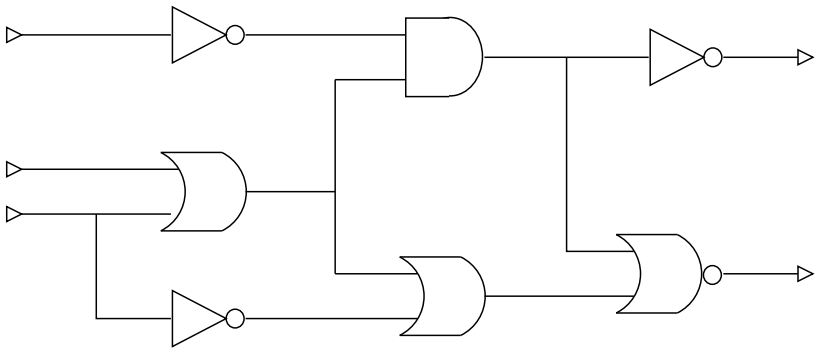

Figure 3: Schematic of a digital circuit.

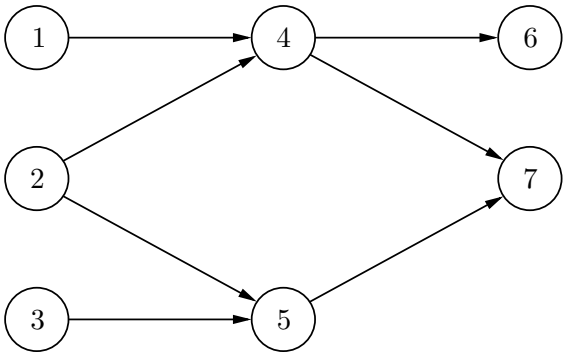

Figure 4: Activity network representation of the digital circuit in figure 3.

Several scalar performance measures can be used to characterize the makespan $\mathbf{C}_{\max}$ of a SAN. Examples include:

- Expected makespan: $\mathbf{E}(\mathbf{C}_{\max})$.
- Probability of missing a deadline T_0: $\mathbf{prob}(\mathbf{C}_{\max} \geq T_0)$.
- Expected tardiness: $\mathbf{E}(\max\{\mathbf{C}_{\max} - T_0, 0\})$.
- α-quantile of makespan: $q_\alpha(\mathbf{C}_{\max}) = \inf\{t : \mathbf{prob}(\mathbf{C}_{\max} \leq t) \geq \alpha\}$.

In project management, the expected makespan and expected tardiness have been widely used; quantiles are commonly used in finance and risk management [74, 78]. In digital circuit design problems, the probability of missing a deadline (and therefore also the quantiles of $\mathbf{C}_{\max}$) can be related to *yield*. In this context, the statistical variation in the durations is due to manufacturing variation, and the deadline T_0 represents a timing constraint. The probability of missing the deadline is then the fraction of manufactured devices that will fail to meet the timing constraint. In circuit design, a quantile can be interpreted as the worst-case delay of the circuit, with a given reliability.

3.1 Performance bounds

Exact analysis of SANs is very difficult, except in a few special cases. The recursive representation of the makespan given in (1) (or its formula as a maximum over a set of sums) shows that both addition and maximum of random variables is involved. While there are common families of distributions that are closed under addition, and others that are closed under maximum (or minimum), no obvious family of distributions is closed under both. (And in any case, the typical distributions for $\mathbf{D}$ are not.)

We can, however, say many things about the distribution of $\mathbf{C}_{\max}$. For example, if $\mathbf{D}$ is Gaussian, then $\mathbf{C}_{\max}$ is the maximum of a number of correlated Gaussian random variables (the path delays), and there are many bounds and asymptotics known for such distributions [82]. A general and common approach is based on approximating or bounding the distributions of the completion times, by recursively bounding (or approximating) the distribution of completion time of activity i, given bounds (or approximations) of the distributions of the completion times of its predecessors. Examples can be found in, *e.g.*, [30, 32, 43, 53, 56, 76, 83, 94]. Lower or upper bounds on the expected makespan or expected tardiness in a SAN is considered in, *e.g.*, [13, 17, 66, 93].

3.1.1 The nominal DAN and Jensen's inequality

The DAN $(G, \mathbf{ED})$, with (deterministic) activity durations equal to the means of the durations in the SAN $(G, \mathbf{D})$, is called the *nominal* DAN associated with the SAN $(G, \mathbf{D})$. Roughly speaking, the nominal DAN is the SAN, when we ignore or remove the statistical variation in the activity durations. When the distribution of $\mathbf{D}$ is very tight, *i.e.*, $\mathbf{D}$ is very

close to its mean **ED** with high probability, we expect the makespan of the nominal DAN to give a good approximation of the SAN makespan $\mathbf{C}_{\max}$.

In fact, the makespan of the nominal DAN is always an underestimator of the makespan of the SAN, in the following sense:

$$c_{\max}(\mathbf{ED}) \leq \mathbf{E}(\mathbf{C}_{\max}) = \mathbf{E}(c_{\max}(\mathbf{D})). \tag{2}$$

This holds for any distribution on $\mathbf{D}$. One interpretation of this inequality is that by adding zero-mean statistical variation to the durations of any DAN, we can only increase the expected value of the makespan. This inequality is a direct application of Jensen's inequality, along with the observation that $c_{\max}$ is a convex function of the activity durations ([21]). Convexity follows since $c_{\max}$ is the maximum of a set of sums of components of d, which are linear functions of d ([21]).

Equality holds in (2) if and only if there is a unique path with criticality index one [45]. If there are relatively few paths with high criticality indices, the difference between the left and righthand sides of in (2) can be relatively small. In other cases, *e.g.*, when all activities have low criticality index, the difference can be relatively large.

3.1.2 Performance bounds via stochastic dominance

A scalar random variable $\mathbf{X}$ is said to be *stochastically less than or equal to* another scalar random variable $\mathbf{Y}$ (denoted by $\mathbf{X} \leq_{\mathbf{st}} \mathbf{Y}$) if $F_{\mathbf{X}}(t) \geq F_{\mathbf{Y}}(t)$ holds for all t. (Here $F_{\mathbf{U}} = \mathbf{prob}(\mathbf{U} \geq t)$ denotes the cumulative distribution function of the scalar random variable $\mathbf{U}$.) Stochastic inequality can also be expressed in terms of quantiles: $\mathbf{X} \leq_{\mathbf{st}} \mathbf{Y}$ if and only if $q_\alpha(\mathbf{X}) \leq q_\alpha(\mathbf{Y})$, for all $\alpha \in (0,1)$. (Here $q_\alpha(\mathbf{U}) = \inf\{t : \mathbf{prob}(\mathbf{U} \leq t) \geq \alpha\}$ denotes the α-quantile of $\mathbf{U}$.) All of the performance measures described in §3 are monotone with respect to stochastic dominance. For example, if $\mathbf{X} \leq_{\mathbf{st}} \mathbf{Y}$, the expected tardiness of $\mathbf{X}$ is no more than the expected tardiness of $\mathbf{Y}$.

A basic result is that for any random variables $\mathbf{X}_1, \ldots, \mathbf{X}_p$, no matter what their joint distribution is, their maximum is always stochastically greater than or equal to each of them:

$$\mathbf{X}_i \leq_{\mathbf{st}} \max\{\mathbf{X}_1, \ldots, \mathbf{X}_p\}, \quad i = 1, \ldots, p.$$

Since the makespan $\mathbf{C}_{\max}$ is the maximum of all path delays, it is always stochastically greater than or equal to the delay of any path, no matter what the duration distributions are, and whether or not they are independent. We conclude that for each of the performance measures described in §3, the maximum of the performance measure over all paths is a lower bound on the performance measure of $\mathbf{C}_{\max}$. For example, the α-quantile of the makespan satisfies

$$\max_{p \in \mathcal{P}} q_\alpha(\mathbf{D}_p) \leq q_\alpha(\mathbf{C}_{\max}), \tag{3}$$

where $\mathcal{P}$ is the set of all paths, and $\mathbf{D}_p$ is the delay of path p. This gives us a method for obtaining a lower bound on a performance measure when the durations are Gaussian. In this case, each path delay is Gaussian, and its performance measure can be calculated exactly

as a function of its mean and variance. By taking the maximum of these measures over all of the paths (or a subset) we obtain a lower bound on the performance measure for $\mathbf{C}_{\max}$. (Unfortunately, there is no simple recursion, like the one for calculating $c_{\max}$ in a DAN, for calculating the maximum of the α-quantile over all paths.)

For future use, we give a very simple lower bound on quantiles of the makespan. By Jensen's inequality, the makespan of the nominal DAN is a lower bound on the expected value of the makespan of the SAN. Now we make the following assumption: for every path, its expected delay is less than or equal to its α-quantile:

$$\mathbf{E}(\mathbf{D}_{i_1} + \cdots + \mathbf{D}_{i_k}) \leq q_\alpha(\mathbf{D}_{i_1} + \cdots + \mathbf{D}_{i_k}), \qquad \forall\, p = (i_1, \ldots, i_k) \in \mathcal{P}. \tag{4}$$

If the duration distributions are Gaussian, then the path delay distributions are Gaussian, and this assumption holds for any $\alpha \geq 0.5$. We would argue that in all cases of interest, the assumption will hold for values of α of interest, such as $\alpha = 0.95$. Of course there are distributions for which the mean exceeds the 0.95-quantile, but we feel that these distributions are exceedingly unlikely to arise as path delays in any practical SAN.

When this assumption holds, the makespan of the nominal DAN $(G, \mathbf{ED})$ is a lower bound on the α-quantile of the makespan of the SAN:

$$c_{\max}(\mathbf{ED}) = \max_{p=(i_1,\ldots,i_k)\in\mathcal{P}} \{\mathbf{ED}_{i_1} + \cdots + \mathbf{ED}_{i_k}\} \leq q_\alpha(\mathbf{C}_{\max}). \tag{5}$$

3.1.3 Performance bounds via surrogate DANs

The DAN $(G, \tilde{d})$ with $\tilde{d}_i = \mathbf{ED}_i + \kappa_i \sigma(\mathbf{D}_i)$, where $\sigma(\mathbf{U})$ denotes the standard deviation of a scalar random variable $\mathbf{U}$, is called a *surrogate DAN* of the SAN $(G, \mathbf{D})$. We call $\kappa_i \geq 0$ the *margin coefficients*. Note that the nominal DAN is the surrogate DAN with all margin coefficients zero. Surrogate DANs will play a central role in this paper.

We can derive some bounds on the quantiles (or other measures) of the makespan of a SAN from the makespan of its surrogate DAN (for proper choice of margin coefficients). We consider the case in which the duration distributions are independent and Gaussian. The delay of path $p = (i_1, \ldots, i_k)$ is also Gaussian, and its α-quantile can be expressed as

$$q_\alpha(\mathbf{D}_p) = \mathbf{ED}_{i_1} + \cdots + \mathbf{ED}_{i_k} + \Phi^{-1}(\alpha)\left(\sigma(\mathbf{D}_{i_1})^2 + \cdots + \sigma(\mathbf{D}_{i_k})^2\right)^{0.5}$$

where

$$\Phi(\alpha) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\alpha} e^{-t^2/2}\, dt$$

is the cumulative distribution function of a unit Gaussian variable.

We will now relate $q_\alpha(\mathbf{D}_p)$ to the delay of a path in a surrogate DAN, using the Cauchy-Schwartz inequality

$$(u_1^2 + \cdots + u_k^2)^{0.5} \geq a_1 u_1 + \cdots + a_k u_k, \tag{6}$$

provided $a_1^2 + \cdots + a_k^2 \leq 1$. This gives

$$q_\alpha(\mathbf{D}_p) \geq \mathbf{ED}_{i_1} + \cdots + \mathbf{ED}_{i_k} + \Phi^{-1}(\alpha)(a_1\sigma(\mathbf{D}_{i_1}) + \cdots + a_k\sigma(\mathbf{D}_{i_k})),$$

provided $a_1^2 + \cdots + a_k^2 \leq 1$. Note that the righthand side here is the delay of the path p, in the surrogate DAN with $\kappa_i = \Phi^{-1}(\alpha)a_i$.

We can make several simple choices of the a_i so that the requirement $a_1^2 + \cdots + a_k^2 \leq 1$ holds for every path. One simple choice is

$$a_i = l_{\max}^{-1/2},$$

where $l_{\max}$ is the maximum length of any path in the DAG. Another choice is

$$a_i = l_i^{-1/2},$$

where

$$l_i = \max\{l(p) \ : \ \text{path } p \text{ contains activity } i\}$$

is the length of the longest path that contains activity i.

The quantity $l_{\max}$ is readily computed using the recursion given in §2.1, by evaluating the makespan of the DAN with DAG G and all activity durations equal to one. A variation on this recursion can be used to efficiently calculate the quantities l_i. We use a recursion to compute the length of the longest path from a source to node i, and another recursion, starting at the sink nodes and working backward, to find the length of the longest path from each node i to a sink. We add these two quantities at each node to obtain l_i.

The bounds above, along with (3), imply that the makespan of the surrogate DAN $(G, \tilde{d})$, with the choice of margin coefficients

$$\kappa_i = \Phi^{-1}(\alpha){l_{\max}}^{-1/2}, \tag{7}$$

or the more sophisticated choice

$$\kappa_i = \Phi^{-1}(\alpha)l_i^{-1/2}, \tag{8}$$

is a lower bound on α-quantile of the makespan of the SAN. Note that the makespan of a surrogate DAN can be computed efficiently using the recursive method given in (1).

The same lower bound on the α-quantile of the makespan of the SAN holds with correlated Gaussian duration distributions, provided that the covariance of any two durations is nonnegative. When this is the case, the standard deviation of the delay of any path is less than or equal to the standard deviation when the durations have the same standard deviations, but are uncorrelated.

We also note that it is also possible to obtain *upper bounds* on a performance measure, such as the α-quantile, of $\mathbf{C}_{\max}$, that have a similar form. As an example, we consider the SAN in which the durations are Gaussian and can be correlated. Let $|\mathcal{P}|$ be the total number of paths from sources to sinks and $\mathbf{Y}_1, \ldots, \mathbf{Y}_{|\mathcal{P}|}$ denote the delays of all such paths, *i.e.*,

$$\mathbf{C}_{\max} = \max\{\mathbf{Y}_1, \ldots, \mathbf{Y}_{|\mathcal{P}|}\}.$$

Let $\mathbf{Z}_1, \ldots, \mathbf{Z}_{|\mathcal{P}|}$ denote *independent* random variables whose distributions are identical to those of $\mathbf{Y}_1, \ldots, \mathbf{Y}_{|\mathcal{P}|}$, respectively, that is,

$$\tilde{\mathbf{C}}_{\max} = \max\{\mathbf{Z}_1, \ldots, \mathbf{Z}_{|\mathcal{P}|}\}.$$

A basic result on stochastic comparison between random variables shows that

$$\mathbf{C}_{\max} \leq_{\text{st}} \tilde{\mathbf{C}}_{\max}.$$

It follows that the α-quantile of $\mathbf{C}_{\max}$ is less than or equal to that of $\tilde{\mathbf{C}}_{\max}$:

$$q_\alpha(\mathbf{C}_{\max}) \leq q_\alpha(\tilde{\mathbf{C}}_{\max}).$$

Since $\mathbf{Z}_i$ are independent of each other, the righthand side can be expressed as

$$q(\tilde{\mathbf{C}}_{\max}) = \inf\left\{t \ : \ \prod_{i=1}^{|\mathcal{P}|} F_{\mathbf{Z_i}}(t) \geq \alpha\right\}. \tag{9}$$

Let t be given by

$$t = \max_{i=1,\ldots,|\mathcal{P}|} \mathbf{E}\mathbf{Z}_i + \Phi^{-1}(\alpha^{1/|\mathcal{P}|})\sigma_i(\mathbf{Z}_i).$$

Since $\mathbf{Z}_i$ are Gaussian, we have $F_{\mathbf{Z_i}}(t) \geq \alpha^{1/|\mathcal{P}|}$, and hence $\prod_{i=1}^{|\mathcal{P}|} F_{\mathbf{Z_i}}(t) \geq \alpha$. This along with (9) leads to the inequality

$$q(\tilde{\mathbf{C}}_{\max}) \leq \max_{i=1,\ldots,|\mathcal{P}|} \mathbf{E}\mathbf{Z}_i + \Phi^{-1}(\alpha^{1/|\mathcal{P}|})\sigma_i(\mathbf{Z}_i).$$

We now relate the righthand side of the inequality above to the delay of a path in a surrogate DAN. Let $\mathbf{Z}_i$ be the delay of a path consisting of nodes $i_1, \ldots, i_k$:

$$\mathbf{Z}_i = \mathbf{D}_{i_i} + \cdots \mathbf{D}_{i_k}.$$

Since $(x_1^2 + \cdots x_k^2)^{0.5} \leq x_1 + \cdots + x_k$ for $x_i \geq 0$, we have

$$\begin{aligned}
\mathbf{E}\mathbf{Z}_i + \Phi^{-1}(\alpha^{1/|\mathcal{P}|})\sigma_i(\mathbf{Z}_i) &= \sum_{j=1}^{k} \mathbf{E}\mathbf{D}_{i_j} + \Phi^{-1}(\alpha^{1/|\mathcal{P}|})\left(\sum_{j=1}^{k} \sigma_{i_j}(\mathbf{D}_{i_j})^2\right)^{0.5} \\
&\leq \sum_{j=1}^{k}\left(\mathbf{E}\mathbf{D}_{i_j} + \Phi^{-1}(\alpha^{1/|\mathcal{P}|})\sigma_{i_j}(\mathbf{D}_{i_j})\right).
\end{aligned}$$

The righthand side is the delay of the path $p = (i_1, \ldots, i_k)$ in the DAN with activity durations $\mathbf{E}\mathbf{D}_i + \Phi^{-1}(\alpha^{1/|\mathcal{P}|})\sigma_i(\mathbf{D}_i)$. We therefore have the bound

$$q_\alpha(\mathbf{C}_{\max}) \leq \gamma,$$

where γ is the makespan of the surrogate DAN with activity durations $\mathbf{E}\mathbf{D}_i + \kappa\sigma_i(\mathbf{D}_i)$, where

$$\kappa = \Phi^{-1}(\alpha^{1/|\mathcal{P}|}).$$

For a SAN where the number of all paths is 3000, the choice of $\kappa = 4.14$ gives an upper bound for its 0.95-quantile. As in the lower bound analysis above, we can obtain a tighter

upper bound on the α-quantile, by using a surrogate DAN (G, κ) with activity durations $\mathbf{ED}_i + \kappa_i \sigma_i(\mathbf{D}_i)$ where κ_i are chosen depending on l_i.

To summarize the results of this section, we have described some computationally efficient and relatively simple performance bounds. These bounds require only the means and variances of the activity durations, and may not be as accurate as others that have been proposed, *e.g.*, the one given in [13], which rely on the same information. But the main purpose of the bounds given in this section is not *analysis* of a SAN (which can be done efficiently and accurately by Monte Carlo analysis). The bounds given in this section will be used to provide bounds on suboptimality of SAN *design*. This will be demonstrated in §6 with a statistical circuit design example, which will demonstrate the computational merits of our bounds, and that the bounds can be tight in practice.

3.2 Monte Carlo analysis

While exact analysis of the distribution of $\mathbf{C}_{\max}$ is very difficult, Monte Carlo (MC) methods can be used to approximately compute the distribution, along with the performance measures or other quantities of interest in the SAN. In this section we describe basic Monte Carlo analysis of a SAN; more sophisticated methods can be used to get higher accuracy with fewer samples, or combine Monte Carlo simulation with bounding methods (see, *e.g.*, [24, 75, 85, 88]). More discussion of sampling strategies for Monte Carlo analysis of SANs can be found in [6, 30, 43, 56, 76, 94].

In basic Monte Carlo analysis, we draw M independent samples $d^{(1)}, \ldots, d^{(M)}$ from the distribution of the random duration vector $\mathbf{D}$, and create the M DANs $(G, d^{(1)}), \ldots, (G, d^{(M)})$. For each realization of the SAN $(G, \mathbf{D})$, we can efficiently evaluate its makespan, using the recursion described in §2.1, to obtain $c_{\max}^{(1)}, \ldots, c_{\max}^{(M)}$. (We might also keep track of a critical path for each of the DANs.) The sampled makespans $c_{\max}^{(1)}, \ldots, c_{\max}^{(M)}$, are, of course, independent samples from the distribution of $\mathbf{C}_{\max}$.

3.2.1 PDF estimation

A nonparametric kernel estimate $\hat{f}_{\mathbf{C}_{\max}}$ of the probability density function (PDF) of the makespan takes the form

$$\hat{f}_{\mathbf{C}_{\max}}(t) = \frac{1}{M} \sum_{k=1}^{M} \frac{1}{h} K\left(\frac{t - c_{\max}^{(k)}}{h} \right),$$

where $K : \mathbb{R} \to \mathbb{R}$ is a probability density function with mean 0 and variance 1, *e.g.*, the standard normal distribution, and h is a positive constant, called the *bandwidth* of the estimator [90]. Our extensive computational experience with this kernel estimator suggests that a sample of several thousand realizations very often suffices to obtain an accurate estimate of the PDF of $\mathbf{C}_{\max}$, even for large SANs with thousands of activities, provided the activity durations are well shaped, *e.g.*, Gaussian or uniform.

3.2.2 Quantile estimation

To estimate $q_\alpha(\mathbf{C}_{\max})$ from the samples $c_{\max}^{(1)}, \ldots, c_{\max}^{(M)}$, we first re-order them so that

$$c_{\max}^{(1)} \leq \cdots \leq c_{\max}^{(M)}.$$

(Once re-ordered, $c_{\max}^{(k)}$ is called the kth *order statistic* of $\mathbf{C}_{\max}$.) A simple estimate $\hat{q}_\alpha(\mathbf{C}_{\max})$ of the quantile is given by

$$\hat{q}_\alpha(\mathbf{C}_{\max}) = c_{\max}^{(\lceil M\alpha+1 \rceil)}$$

where $\lceil x \rceil$ denotes the integral part of $x \in \mathbb{R}$. This estimate is asymptotically consistent and its variance is inversely proportional to M under the mild assumption of $f_{\mathbf{C}_{\max}}(q_\alpha(\mathbf{C}_{\max})) > 0$, where $f_{\mathbf{C}_{\max}}$ is the PDF of $\mathbf{C}_{\max}$ [90].

3.2.3 Criticality index estimation

Under mild assumptions, a realization of the SAN $(G, \mathbf{D})$ has a unique critical path, with probability one. The criticality index of an activity or a path from an input node to an ending node can be estimated by counting the fraction of the realizations in which it is critical. These estimates are consistent, and the variance of the estimation error is inversely proportional to M [42]. More discussion on estimating the criticality indices of activities and paths can be found in [18, 33].

3.3 Statistical timing analysis of digital circuits

In statistical timing analysis of a digital circuit, the gate delays are not fixed constants but random variables, possibly dependent. The increasing importance of process variation explains the growing interest in process variation modeling and statistical timing analysis of digital circuits; see, *e.g.*, [4, 16, 23, 44, 46, 49, 65, 66]. While there have been several theoretical and empirical studies of statistical variation in circuit parameters, the statistical modeling of circuit parameter variation is still an active research area; so far, no consensus has emerged as to what the best models are.

3.3.1 Example: 32-bit Ladner-Fisher adder

As an example, we carry out MC timing analysis of the 32-bit Ladner-Fisher adder [50], which is used throughout the paper to illustrate our main points. This circuit consists of 459 gates ($n = 459$), 64 input gates, 32 output gates, and 1714 arcs. The associated DAG $G = (N, A)$ has a total of 3214 paths from input nodes to output nodes. Its depth (*i.e.*, the maximum number of gates on a path) is 12. The delay and statistical models are simplified; more details are given in §7. In the sequel we will consider several instances of the adder, with different choices of device widths, but here we focus on one instance, designed to minimize the *nominal cycle time* (*i.e.*, the cycle time of the circuit when the statistical uncertainty and variation is ignored) under constraints on area and device sizes. For this choice of device sizes, there are around 1000 critical paths.

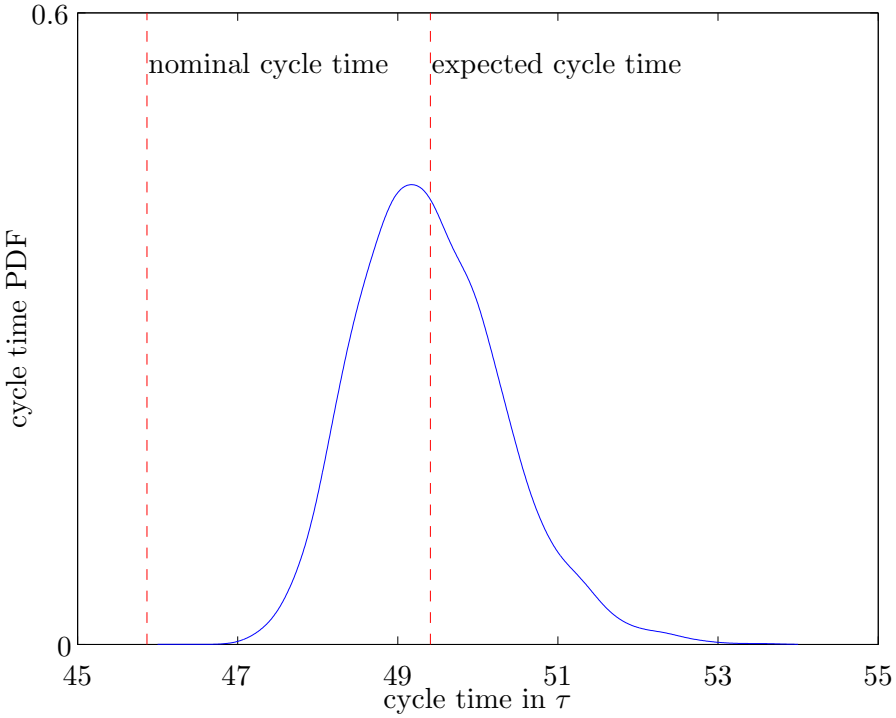

Figure 5: Estimate of cycle time PDF for the Ladner-Fisher adder circuit, obtained by Monte Carlo simulation.

The time unit τ we use throughput the paper denotes the delay of a minimum sized inverter driving no load. Here (and throughput the paper) we estimate the PDF of cycle times using the nonparametric kernel estimation method with a Gaussian kernel, with bandwidth $h = 0.2\tau$, and sample size $M = 5000$ realizations. We have carried out much larger MC simulations, which show that the PDFs estimated from 5000 samples are good enough. The MC simulation time is under one second on a 1.2 GHz Pentium IV PC.

The MC simulation of the cycle time PDF is shown in figure 5, along with the nominal cycle time and the expected cycle time (the latter found by MC simulation). As required by Jensen's inequality, the expected cycle time is larger than the nominal cycle time. We can also observe that the cycle time distribution is skewed to the right. This is to be expected: $\mathbf{C}_{\max}$ is the maximum of 3214 path delays, and the maximum of a large number of random variables is typically right skewed.

Monte Carlo simulation gives the (accurate) estimate for the 95% confidence cycle time $q_{0.95}(\mathbf{C}_{\max}) = 51.1$. The makespan of the nominal DAN is 45.9. In this example the delay distributions are Gaussian and uncorrelated, so the lower bounds via surrogate DANs described in §3.1.3 can be used. The lower bound obtained using the Cauchy-Schwartz inequality with l_i given by (7) is $q_{0.95}(\mathbf{C}_{\max}) \geq 48.7$; with the choice given by (8) we have $q_{0.95}(\mathbf{C}_{\max}) \geq 48.9$. The lower bound obtained by (3), *i.e.*, the maximum 0.95-quantile over all 3214 paths, is $q_{0.95}(\mathbf{C}_{\max}) \geq 49.0$. For this example, these lower bounds are much tighter

than the makespan of the nominal DAN.

4 Optimal activity network design

We now consider the problem of optimizing an activity network. We have a set of decision or design variables $x \in \mathbb{R}^m$, which affect the activity durations (for a DAN) or the distributions of the activity durations (for a SAN).

4.1 Optimal DAN design

We first consider a DAN. Taking the makespan as the objective, we have the optimization problem

$$\begin{array}{ll} \text{minimize} & c_{\max} \\ \text{subject to} & x \in \Omega. \end{array} \tag{10}$$

Here Ω represents the constraint set. In this problem the design variable x affects the activity durations, which in turn determine the makespan $c_{\max}$. There are many other formulations; we could, for example, impose an upper bound on the makespan, and minimize some other objective function that depends on x.

There are several approaches to solving (or approximately solving) the DAN optimization problem (10). Several heuristic approaches are based on shifting resources from noncritical paths to critical paths, *i.e.*, modifying x so as to maintain feasibility, while decreasing delay along critical paths.

A standard dynamic programming method can be used to avoid enumerating expressions for the path delays. Using the recursion (1), we write the problem (10) as

$$\begin{array}{ll} \text{minimize} & c_{\max} = \max\{c_i \ : \ i \text{ a sink node}\} \\ \text{subject to} & c_i = \max\limits_{j \in \mathbf{Pred}(i)} c_j + d_i, \quad i \text{ not a source node} \\ & c_i = d_i, \quad i \text{ a source node} \\ & x \in \Omega. \end{array}$$

Here x directly affects the activity durations d_i, which in turn affect the completion times c_i and the makespan $c_{\max}$. In this formulation, the completion times are also optimization variables, constrained by the recursion equations. This formulation avoids enumerating the paths, but introduces new variables (the completion times) and new equality constraints.

For most numerical methods, these equality constraints are difficult to handle, since the maximum destroys differentiability. A standard trick can be used to convert the problem to one involving inequality constraints, which typically is easier to solve. We replace the equality constraints above with inequalities and introduce new variables $t_1, \ldots, t_n$ and s,

which serve as upper bounds on the completion times and makespan, respectively:

$$\begin{array}{ll} \text{minimize} & s \\ & t_j \leq s, \quad j \text{ a sink node} \\ & t_j + d_i \leq t_i, \quad j \in \mathbf{Pred}(i), \quad i \text{ not a source node} \\ & d_i \leq t_i, \quad i \text{ a source node} \\ & x \in \Omega. \end{array} \tag{11}$$

In this formulation, the variables t_i represent upper bounds on the completion times; at the optimal solution, these bounds are tight for any activity along any critical path. It follows that any optimal solution of this problem yields an optimal solution of the original problem (10), and vice versa.

One advantage of the formulation (11) is that the constraint functions are differentiable if the durations d_i are. Although we have introduced $n + 1$ new variables, the constraints are relatively sparse (at least in the variables t_i and s), and this can be exploited by a solver for efficiency; see, *e.g.*, [11, 28, 62, 91].

In many cases we have to settle for a locally optimal solution of the DAN optimization problem (10). But in some cases we can solve the problem exactly, *i.e.*, globally. If the durations are convex functions of x, then the problem (11) is a convex optimization problem, and so can be (globally) solved very efficiently; see [10, 12, 21, 60, 59]. Another case in which we can solve the problem globally is when the durations are posynomials, or generalized posynomials. In this case the problem (11) is a geometric program (GP), and can be efficiently solved (globally) even for large networks [21, 20]. When the durations are convex, or generalized posynomial, a DAN optimization problem with a thousand variables is easily solved; if the problem is sparse, far larger problems, involving say 10^5 variables, can be solved.

4.2 Example: cycle time minimization of digital circuits

A wide variety of optimization methods have been developed for device and wire sizing in digital circuits; see, *e.g.*, [26, 27, 38, 79, 80, 81]. The cycle time minimization problem can be cast as a DAN optimization problem of the form (10), or an extension that handles rising and falling delays, signal slopes, and so on. See [19] for more on GP-based sizing of digital circuits.

As an example, we consider the 32-bit Ladner-Fisher adder circuit. Each of the 459 gates can be scaled by a multiplier or scale factor $x_i \geq 1$, which are the optimization variables. These scale factors can be thought of as scaling the widths of all devices in each gate. Each scale factor therefore affects the drive strength of its gate as well as the gate input capacitance. Thus, the scale factor affects the gate delay, as well as the delays of the gates that drive it (through capacitive loading). We impose lower bounds on the scale factors, $x_1 \geq 1$, as well a maximum on the total area for the circuit, $A \leq 12000$. (The details of the models are given in §7.)

We solved the associated DAN optimization problem, and compare it to the simple choice in which all scale factors are equal (with the same total area). Figure 6 compares the path

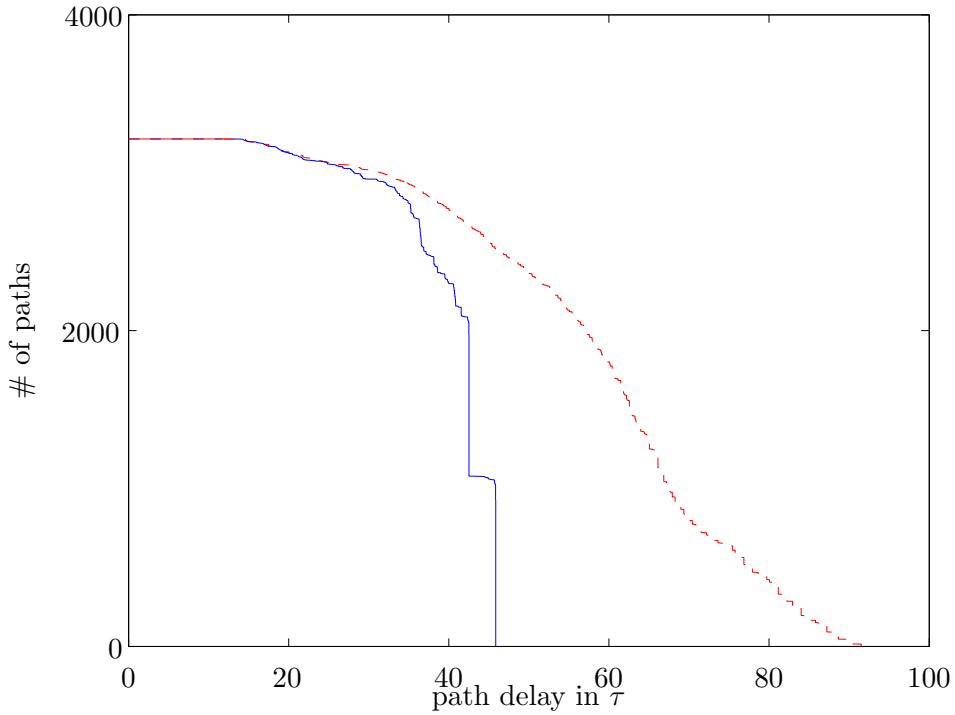

Figure 6: Path delays for optimal (solid) and uniform (dashed) gate scale factors in the 32-bit Ladner-Fisher adder, with no statistical variation.

delays for the optimal and the simple choice of scale factors. The optimal choice of scale factors reduces the cycle time by around 50%. The figure shows that optimizing cycle time results in a large number of critical paths (around 1000). This is a general phenomenon: optimizing a DAN generally results in many critical paths. The steep edge of the path delay distribution plot near the cycle time is sometimes referred to as a cliff, or wall.

4.3 Optimal SAN design

We now consider a SAN in which the distributions of the activity durations (and therefore the distribution of the makespan $\mathbf{C}_{\max}$) are affected by the choice of the design variable $x \in \mathbb{R}^m$. As the objective we can take a scalar performance measure of the makespan, such as expected tardiness or a quantile of the makespan:

$$\begin{array}{ll} \text{minimize} & q_\alpha(\mathbf{C}_{\max}) \\ \text{subject to} & x \in \Omega. \end{array} \tag{12}$$

This stochastic programming problem arises in a variety of applications areas, from project management to digital circuit design; see, *e.g.*, [37, 57, 72, 84]. Unfortunately, in all but a few trivial cases, it is very difficult to solve exactly.

Numerous methods have been developed for stochastic programming, including random search methods [40], methods based on a stochastic model of the objective [74], and sample-

path optimization methods [42]. Although they enjoy some theoretical convergence properties under certain conditions, these methods do not scale well to problems involving thousands or more optimization variables and activities.

5 Heuristic SAN design

Since it is extremely difficult to find the exact solution of a general SAN optimization problem (12), we resort to a heuristic method. We propose optimizing a surrogate DAN, with activity durations

$$d_i(x) = \mu_i(x) + \kappa_i \sigma_i(x), \quad i \in N,$$

with margin coefficients $\kappa_i \geq 0$, where $\mu_i(x)$ and $\sigma_i(x)$ denote the mean and standard deviation, respectively, of the ith activity duration. In other words, we consider the DAN optimization problem

$$\begin{array}{ll} \text{minimize} & c_{\max}(d(x)) \\ \text{subject to} & x \in \Omega \end{array} \tag{13}$$

as a heuristic for solving the SAN optimization problem (12). The DAN optimization problem (13) is readily solved. If $\mu_i(x)$ and $\sigma_i(x)$ are convex (or generalized posynomial) functions, then the surrogate DAN problem (13) is convex (or a generalized GP), and so can be efficiently and globally solved, even for large-scale problems.

This proposed heuristic method is very natural. In order to take into account statistical variation in the activity durations, we simply add a multiple of the standard deviation to the mean, and optimize the resulting surrogate DAN. We have already encountered this idea in §3.1.3, where surrogate DANs are used to find bounds on the performance of the SAN. Here, however, we propose using surrogate DAN optimization as a method for obtaining a suboptimal design for the SAN optimization problem; in §3.1.3, the surrogate DAN is used to find a lower bound on a performance measure such as a quantile.

This simple heuristic method is similar in spirit to the general method of *regularization* [61], in which an extra penalty term is added to a problem to approximately account for some variation in the problem data. A more sophisticated and recent approach to handling uncertainty and variation in optimization problem data is *robust optimization* [8, 9, 14, 21, 15, 41, 47]. In robust optimization, an explicit model of data uncertainty is used, and the objective is taken to be the average or worst-case value of the objective, over the parameter variation set. Some robust optimization problems are tractable; for others there are good heuristics [21].

The margin coefficients κ_i (whose choice is discussed below) can be related to the probability that $\mathbf{D}_i$ exceeds $\mu_i + \kappa_i \sigma_i$. In the general case, Chebyshev's inequality [3] tells us that

$$\mathbf{prob}\left(\mathbf{D}_i(x) \leq \mu_i(x) + \kappa_i \sigma_i(x)\right) \geq \frac{\kappa_i^2}{1+\kappa_i^2},$$

no matter what the distribution of $\mathbf{D}_i(x)$ is. For $\kappa = 3$, for example, we have

$$\mathbf{prob}\left(\mathbf{D}_i(x) \leq \mu_i(x) + 3\sigma_i(x)\right) \geq 0.9,$$

so the surrogate duration $\mu_i + 3\sigma_i$ is an upper bound on the 0.9-quantile of $\mathbf{D}_i$, valid for any distribution. If $\mathbf{D}_i(x)$ is Gaussian, then we have

$$\mathbf{prob}\left(\mathbf{D}_i(x) \leq \mu_i(x) + \kappa_i\sigma_i(x)\right) = \Phi(\kappa_i).$$

For $\mathbf{D}_i(x)$ Gaussian, the 0.9-quantile of $\mathbf{D}_i$ is given by the surrogate duration $\mu_i + \Phi^{-1}(0.9)\sigma_i = \mu_i + 1.282\sigma_i$.

Evidently κ_i controls the weight given to the statistical variation in the duration of activity i, compared to its mean, in the surrogate DAN. In particular when all κ_i are set to zero (*i.e.*, the statistical variation is ignored), the resulting design reduces to the nominal optimal design, which ignores the statistical variation. As we increase the margin parameters, we obtain a more and more risk-averse design, *i.e.*, one that trades off nominal performance for a reduction in variance. Whenever $\kappa_i \geq \Phi^{-1}(\alpha^{1/|\mathcal{P}|})$, the optimal objective for the surrogate DAN design optimization problem is an upper bound on the optimal objective for the original SAN optimization problem.

We have experimented with many schemes for choosing the margin weights. The simplest is to set them all equal to some constant κ, which would typically be chosen between 1 and 3. Another simple scheme comes from the lower bounding method (see (8)): we set $\kappa_i = \eta l_i^{-1/2}$, where l_i is the length of the longest path that contains activity i, and η is a constant. In both of these schemes, we have to choose a single constant. One good method for choosing the constant is to carry out the optimization on the surrogate DAN, for some set of values of the constant, and then subject the resulting designs to Monte Carlo analysis, to estimate the objective values (*e.g.*, $q_\alpha(\mathbf{C}_{\max})$). We then choose the constant corresponding to the best objective value. Our computational experience suggests that the choice of the constant has an effect on the objective, but is not critical. We have found that this method, though simple, seems to give very good results.

We have also experimented with more sophisticated methods, which alternate between a design step, in which a surrogate DAN is optimized, and a Monte Carlo analysis step, in which the objective, as well as other relevant quantities (such as criticality indices) are estimated. Based on Monte Carlo analysis, we then update the margin coefficients, and repeat the process. An effective margin coefficient update rule should decrease the margin coefficients for the less critical activities (*i.e.*, those with smaller criticality indices), and increase the coefficients for the highly critical activities. Another approach is suggested by the Cauchy-Schwartz inequality (6): we update the margin coefficients so as to make the Cauchy-Schwartz inequality tight for highly critical paths, for example by setting κ_i proportional to σ_i.

While these iterative methods can improve the performance over the simple schemes based on varying one parameter, we have found that the improvement is often small. In addition, we have not found an update scheme that is effective on a wide variety of problems.

5.1 Lower bounds on achievable performance

To assess the performance of the heuristic SAN design method, we derive some bounds on achievable performance in the optimal SAN design. The lower bounds are based on the lower

	nominal delay	$\mathbf{ED}$	$\sigma_{\mathbf{D}}$	$\mathbf{Q}^{.95}(\mathbf{D})$
nominal design	45.9	49.4	0.91	51.1
robust design	46.5	47.6	0.29	48.1

Table 1: Comparison of nominal and robust designs.

bounds given in §3, and the solution of another surrogate DAN optimization problem (13).

We first mention a simple lower bound that is widely applicable. The inequality (2) shows that no matter what the distributions are, and whether or not they are independent, we have

$$\min_{x\in\Omega} c_{\max}(\mathbf{ED}(x)) \leq \min_{x\in\Omega} \mathbf{EC}_{\max}(x).$$

The lefthand side is the optimal value of the nominal DAN optimization problem, and is easily computed (at least, when μ_i are convex or posynomial). Assuming the distributions of the path delays have their α quantiles larger than their means, we have

$$c_{\max}(\mathbf{ED}(x)) \leq q_\alpha(c_{\max}(\mathbf{D}(x))), \quad \forall\, x \in \Omega. \tag{14}$$

It follows that the optimal value of the SAN optimization problem (13) is at least the optimal value of the nominal DAN problem.

For independent Gaussian activity durations, we can find sharper lower bounds by solving the surrogate DAN problem with margin coefficients given by (7) or (8). The optimal value of the surrogate DAN optimization problem gives a lower bound on the optimal value of the SAN optimization problem (13). (This lower bound holds with correlated Gaussian duration distributions, provided that the covariance of any two durations is nonnegative.)

In summary, by solving a surrogate DAN problem with appropriately chosen margin coefficients, we obtain a lower bound on the optimal value of the (very difficult) SAN optimization problem (13).

6 Application to statistical digital circuit sizing

In this section we illustrate the heuristic method on a simplified statistical digital circuit sizing problem, in which the device widths affect both the mean and variance of the gate delays. The gate delays are independent and Gaussian, and depend on the gate scale factors x_i. We impose the same limits on the scale factors and total area as those in the deterministic digital circuit sizing example on page 16.

The corresponding optimization results are shown in table 1 and figure 7, for the heuristic method with $\kappa_i = 1.5$. The heuristic robust design is far superior to the nominal optimal design: At the cost of a slight increase in the nominal cycle time, over the nominal optimal design, it gives a much tighter distribution for $\mathbf{C}_{\max}$, and a much smaller 0.95-quantile. The robust design has a nominal cycle time (*i.e.*, a cycle time ignoring statistical variation) of 46.9, only 1.3% more than the optimal nominal design. When we take statistical variation

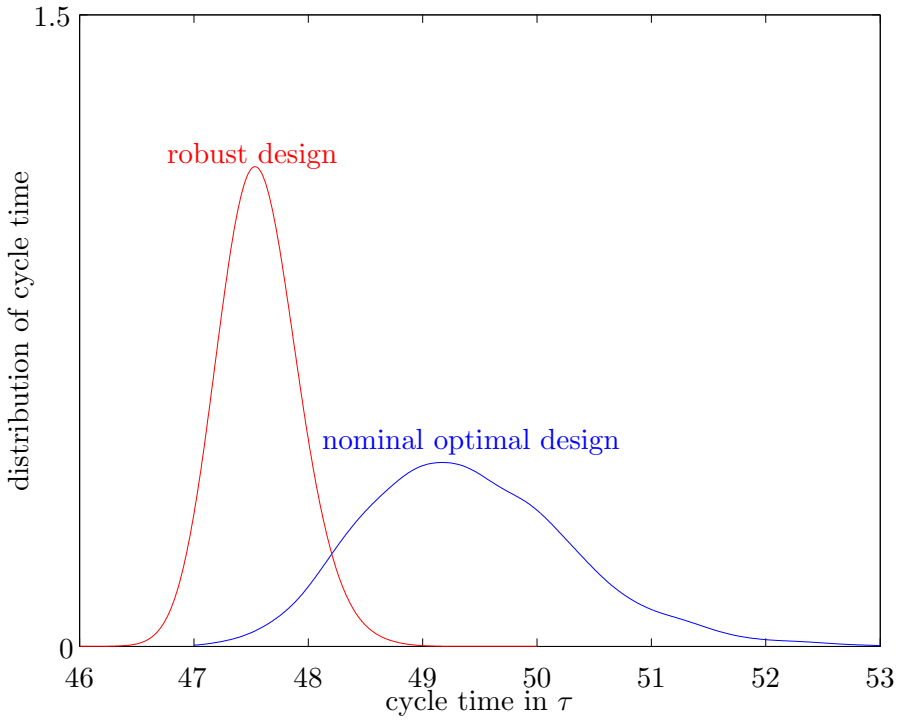

Figure 7: Distribution of path delays for robust and nominal optimal designs.

into account, however, the two designs are quite different. The 95% cycle time for the nominal optimal design is 51.1, which is 11.3% more than the nominal cycle time. For the robust design, the 95% cycle time is 48.1, which is only 3.4% more than the nominal optimal cycle time. Thus, the robust design has reduced the effect of the statistical variation by a factor of around 4, compared to the nominal optimal design.

Since the nominal optimal cycle time is a lower bound on the optimal value of the 95% cycle time, this implies in particular that our heuristic robust design is at most 4.8% suboptimal. A tighter lower bound can be found by minimizing the cycle time of the surrogate DAN with the choice of margin coefficients in (7), which yields 46.7. This means that our heuristic robust design is at most 3% suboptimal, compared to the globally optimal design (which is very difficult to compute).

Figure 8 shows a plot of $q_{0.95}(\mathbf{C}_{\max})$ versus κ. The lefthand side, $\kappa = 0$, corresponds to the nominal optimal design; as κ is increased above 0, the 0.95-quantile decreases. The minimum occurs around $\kappa = 1.5$, but a good design is obtained for κ between around 1 and 2.5. Thus, the choice of κ is not particularly critical.

6.1 Comparison of nominal optimal and robust designs

In this section we examine more carefully the differences between the nominal optimal and robust design (with $\kappa = 1.5$). The two plots in figure 9 show the distribution of scale factors

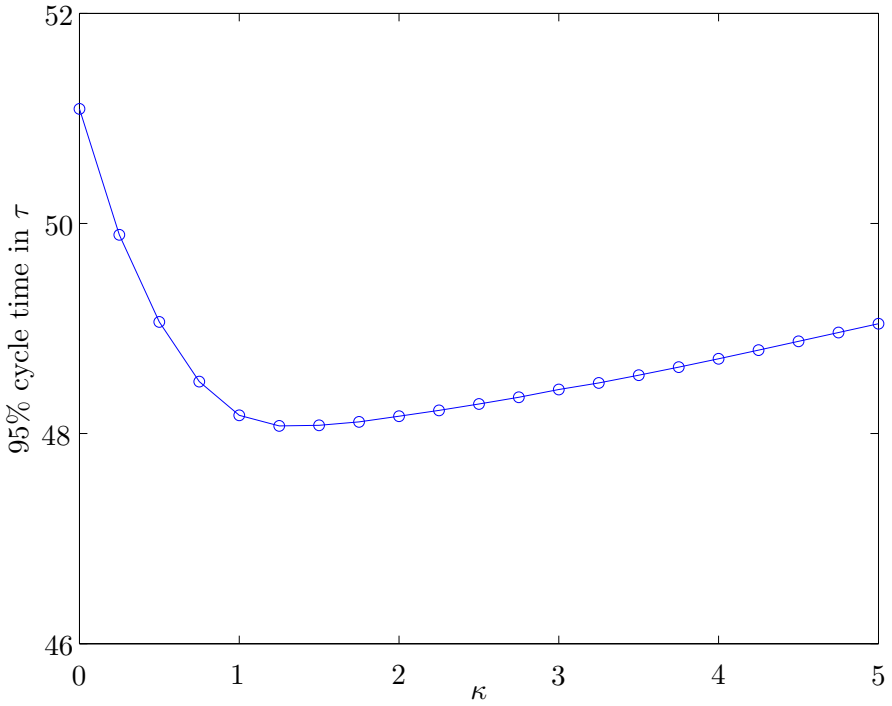

Figure 8: Plot of $q_{0.95}(\mathbf{C}_{\max})$ versus κ. The nominal optimal design corresponds to $\kappa = 0$; the best value of κ is near 1.5.

in the nominal optimal design and the robust design. (Recall that in both designs the total area is fixed to be 12000.) The only significant difference seems to be that the nominal optimal design has more scale factors that are minimum (*i.e.*, have value one) than the robust design. Figure 10 shows a scatter plot of the scale factor for the nominal optimal design versus the one for the robust design. This plot shows that the difference between the two designs seems to be subtle; most of the scale factors in the robust design are within 20% of the corresponding scale factors in the nominal optimal design. One noticeable difference is that many small gates in the nominal optimal design are up-sized in the robust design.

Figure 11 shows the distribution of expected path delays in the nominal optimal and robust design (with $\kappa = 1.5$). Compared with the nominal optimal design, the robust design significantly reduces the number of paths with expected delays very near the nominal cycle time, at the cost of a small increase in the nominal cycle time (the maximum expected path delay over all paths). In other words, the so-called wall in the plot is softened.

But 'softening the wall' only partly explains why the robust design handles gate delay variation more gracefully than the nominal optimal design. Figure 12 shows scatter plots of the mean versus standard deviation for all paths, for the nominal optimal and robust design (with $\kappa = 1.5$), respectively. The plots show that in the nominal optimal design, some paths with large expected delays have large variances (which directly gives $\mathbf{C}_{\max}$ a large right skew). In the robust design, however, the variances of the paths with large expected delays are smaller; paths with relatively small expected delays, however, have relatively larger variances.

More insight into the performance of the robust design can be found in figure 13, which shows the distribution of criticality indices for the nominal optimal and robust designs. In the nominal optimal design there is no node with criticality index larger than 0.3. This is expected, since there is a large number of critical paths. With statistical variation in the activity durations, the large number of nearly critical paths leads to a large right skew in $\mathbf{C}_{\max}$.

In the robust design, in contrast, a relatively small set of nodes have high criticality indices. This means that in the robust design, a relatively smaller set of paths is highly critical. This is the condition for Jensen's inequality (2) to be relatively tight, so we conclude that the nominal value $c_{\max}$ and the expected makespan $\mathbf{E}\mathbf{C}_{\max}$ are close. This suggests the distribution of $\mathbf{C}_{\max}$ is tight, which, in turn, suggests the design is good, *i.e.*, $q_{0.95}(\mathbf{C}_{\max})$ is not too much larger than the nominal cycle time.

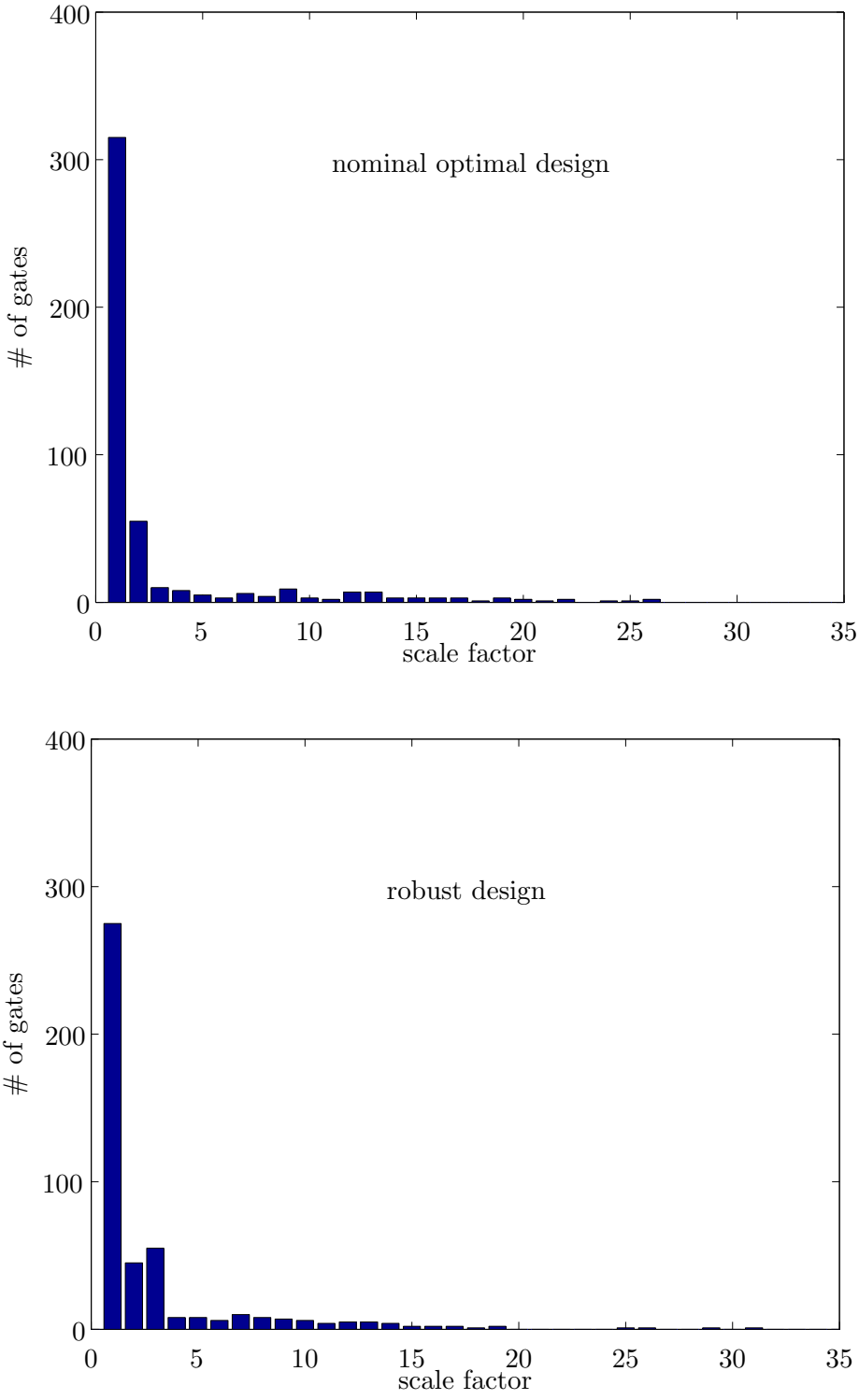

Figure 9: Distribution of scale factors in nominal optimal design (top) and robust design (bottom).

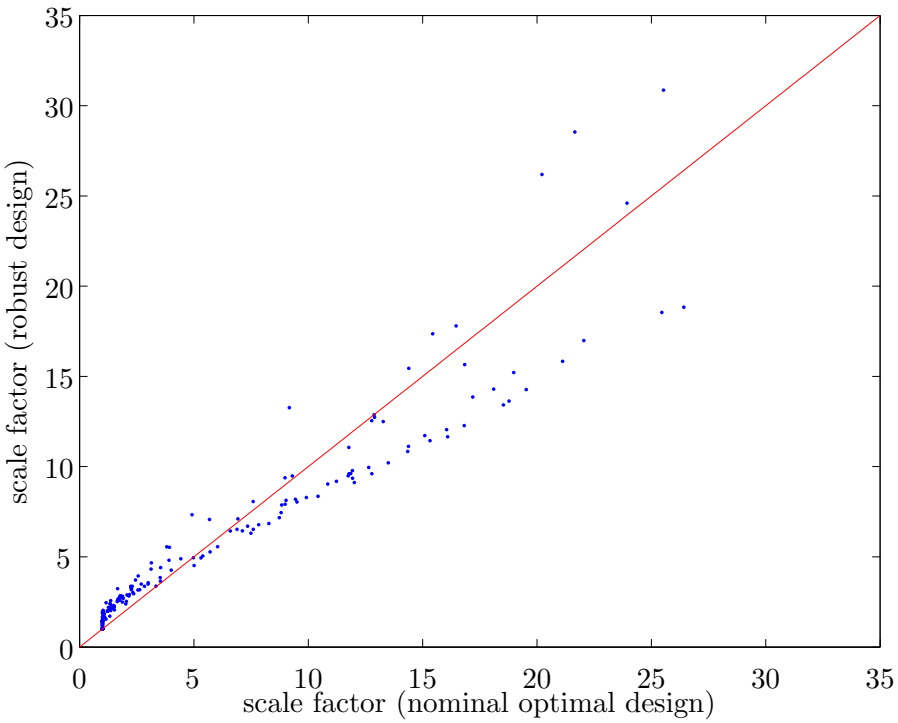

Figure 10: Scatter plot of nominal optimal design scale factor versus robust design scale factor.

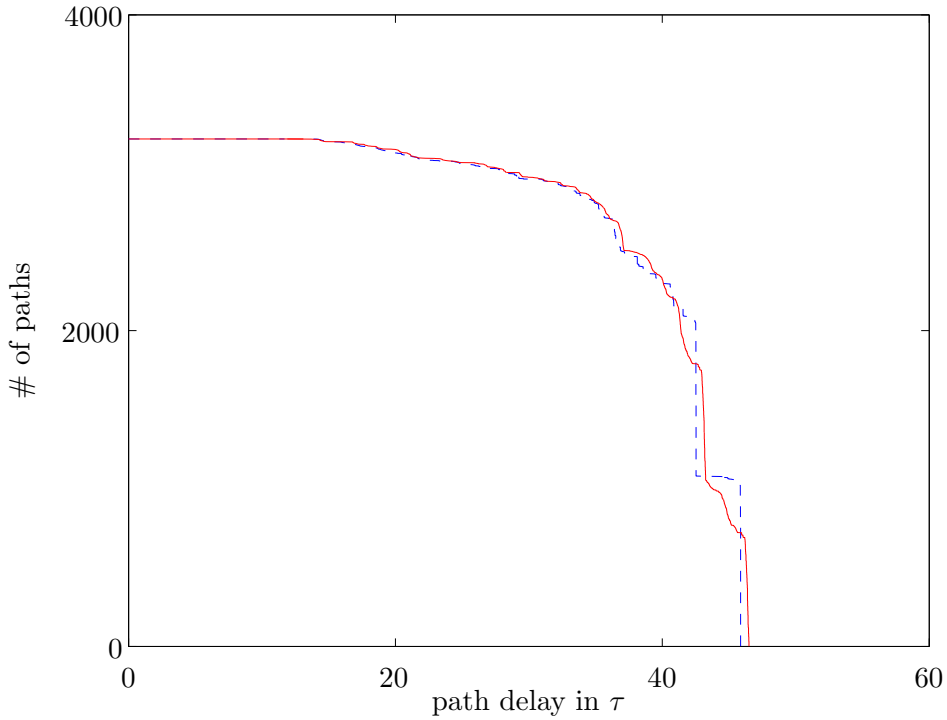

Figure 11: Expected path delays for nominal optimal design (dashed line) and robust design (solid line).

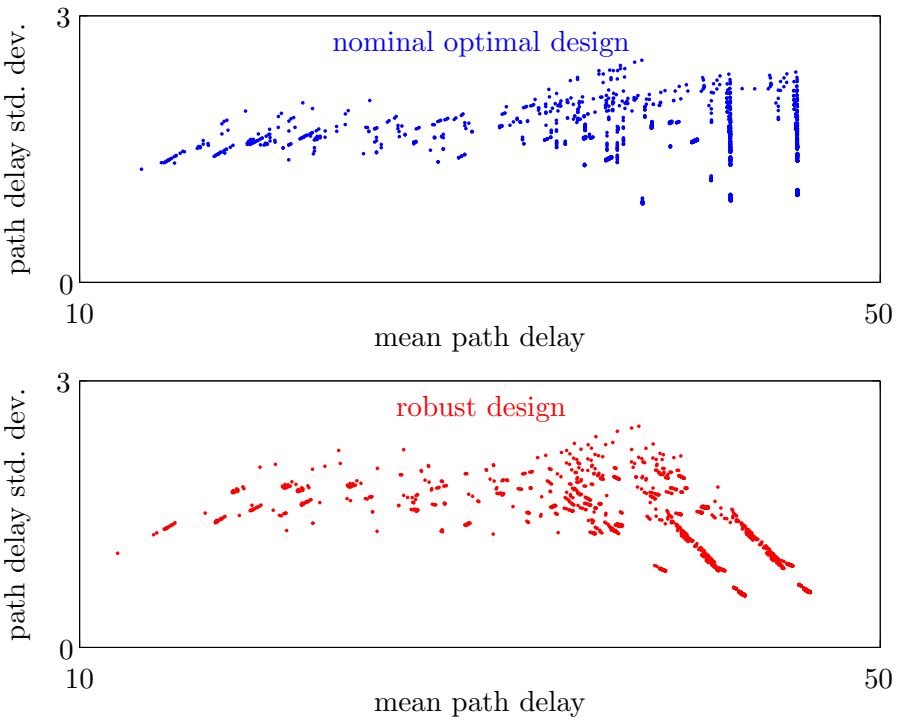

Figure 12: Scatter plots of path delay mean versus path delay standard deviation, for the nominal design (top) and statistical design (bottom).

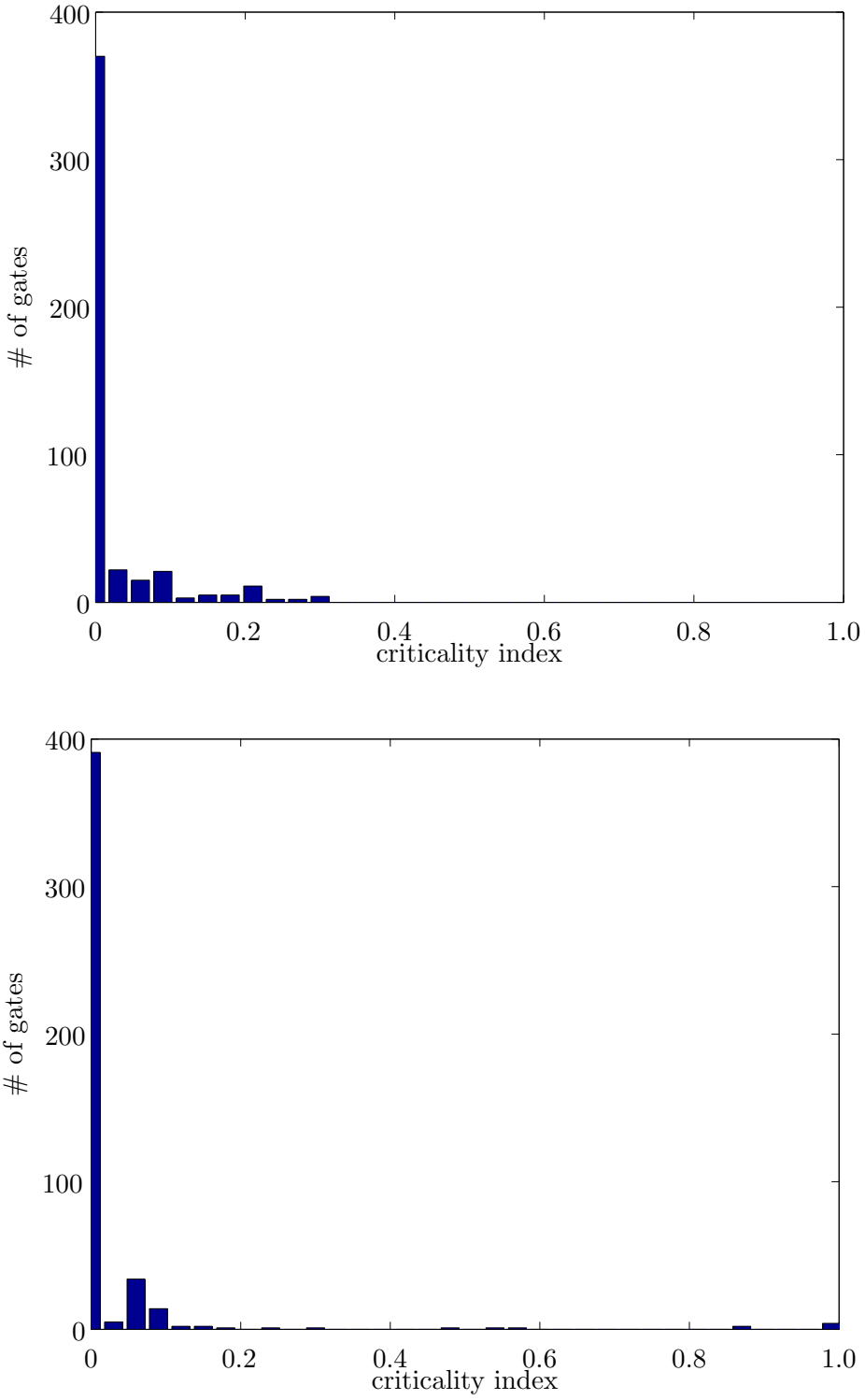

Figure 13: Distribution of criticality indices of gates for nominal optimal design (top) and robust design (bottom).

6.2 Trade-off analysis

In this section we vary the constraint that limits the total of the scale factors (which corresponds to the total circuit area) to carry out an area-delay trade-off analysis. In the design described above, the total area is limited to 12000; here we vary the limit $A^{\max}$ between 4000 and 15000. For each value of the limit on total area, we carry out a nominal optimal design and a robust design. For the robust designs, we take $\kappa = 1.5$ for each of the designs.

Figure 14 summarizes the trade-off analysis results. The bottom curve shows the nominal cycle time versus the area limit $A^{\max}$. The nominal optimal value is, of course, a bit smaller than the robust design in all cases. The top plot shows how the designs compare when the effects of duration statistics are included. We see that the robust design performs consistently well, compared with the nominal design. Note also that adding more area does not particularly improve the nominal optimal design, compared to the robust design. This shows that simply 'throwing more area' at the nominal design method does not lead to dramatic improvements in $q_{0.95}(\mathbf{C}_{\max})$. We can also compare the nominal optimal and robust designs in terms of area required to reach a yield of 95% for a cycle time of 48.1 (which is what our robust design with $A = 12000$ achieves). By following a horizontal line at this 95% cycle time, we find that the nominal optimal design still does not reach this yield, even with 3 times as much area.

7 Details of the statistical circuit sizing example

In this section we give the details of the statistical circuit sizing example used to illustrate our main points throughout the paper. As mentioned above, we use a simplified static timing model, with a single delay for each gate (ignoring differing rise and fall times, different delays for different gate transitions, and the effects of signal slopes). With each gate we associate a *scale factor* or *normalized size* $x_i \geq 1$ which scales the widths of the devices used to form the gate and therefore affects its drive strength, input capacitance, and area. (The same method can be applied to a full custom design, in which each device is sized individually; see [69].) 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.

Gate i has three parameters: an input capacitance $\bar{C}_i^{\text{in}}$, an intrinsic or internal capacitance $\bar{C}_i^{\text{int}}$, and driving resistance $\bar{R}_i^{\text{int}}$. The input and intrinsic 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. The driving resistance R_i 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.

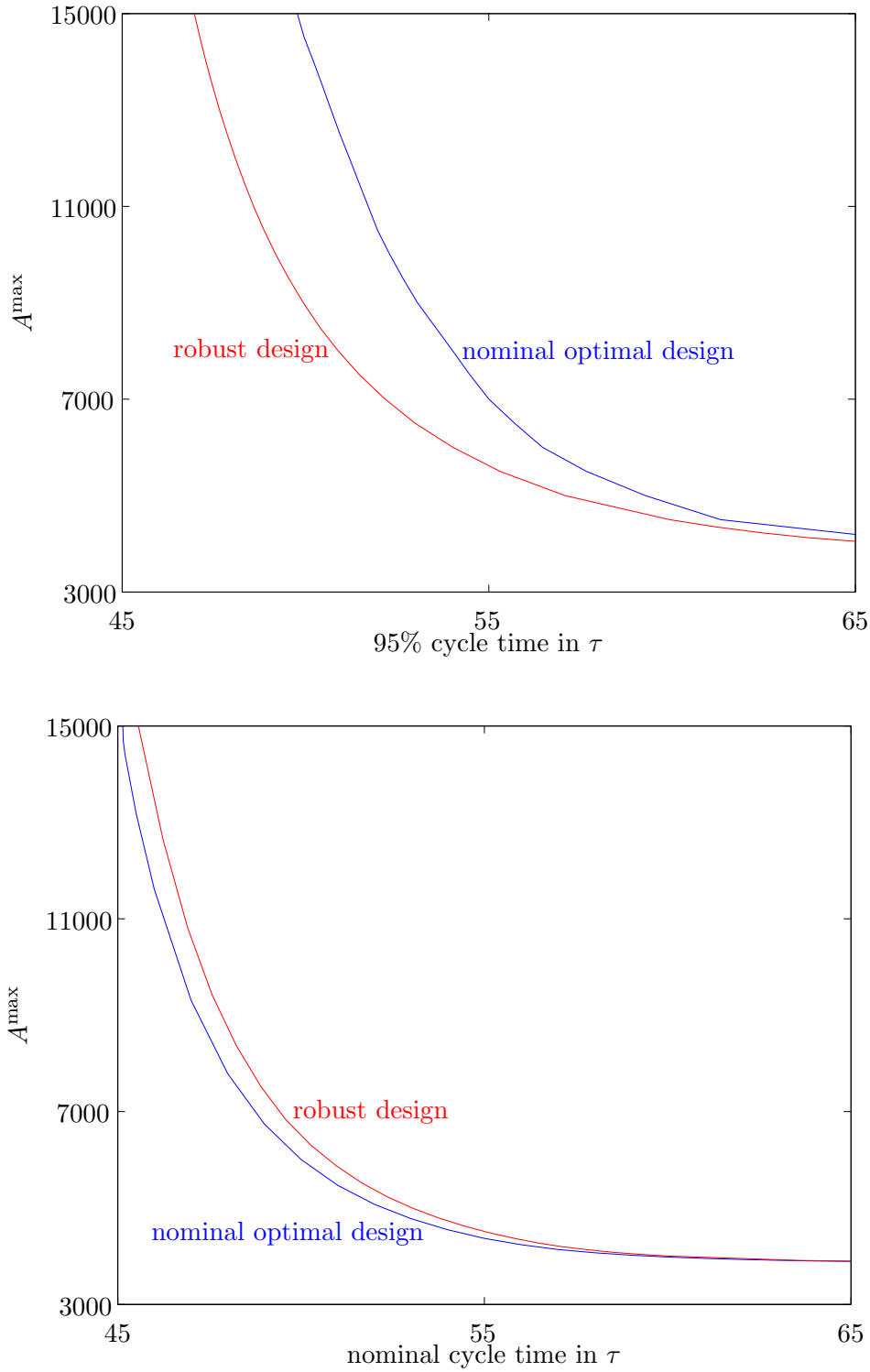

Figure 14: Tradeoff analysis between area and cycle time.

gate type	function	$\bar{C}^{\text{in}}$	$\bar{C}^{\text{int}}$	$\bar{R}$	$\bar{A}$
INV	$\overline{A}$	3	3	0.48	3
NAND2	$\overline{AB}$	4	6	0.48	8
NOR2	$\overline{A+B}$	5	6	0.48	10
AOI21	$\overline{AB+C}$	6	7	0.48	17
OAI21	$\overline{(A+B)C}$	6	7	0.48	16

Table 2: The 5 gate types used in the Ladner-Fisher adder. The first column gives the gate name; the second column gives the logic function the gate implements, and the remaining 4 columns give the model parameters.

Let C_i^{L} be the load capacitance that gate i drives. Then, for a non-sink node,

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

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

$$\mu_i(x) = 0.69 R_i (C_i^{\text{int}} + C_i^{\text{L}}), \tag{15}$$

which is the time required for the output voltage of an RC circuit to reach the midpoint between the logic voltage levels. Since 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.

Another parameter of gate i is the area. We approximates 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 gate area is the total width of the devices in the gate (since the gate lengths are always chosen to be the smallest value allowed in the technology.)

The circuit is a 32-bit Ladner-Fisher adder, consisting of 459 gates, including 64 input gates and 32 output gates, and 1714 arcs. The associated DAG has 3214 paths from input nodes to output nodes. The maximum path length is 12. The Ladner-Fisher adder contains 5 types of gates, with associated functions and model parameters listed in table 2. The capacitance unit is the capacitance of the NMOS device in a unit scaled inverter, and the area unit is the width of the NMOS device in a unit scaled inverter. 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, denoted by τ. In fact the model parameters come from the *logical effort model*, described in [89].

The expression (15) gives the mean delay of a gate. We take the standard deviation of the gate delay to be

$$\sigma_i(x) = \gamma x_i^{-\alpha} \mu_i(x), \tag{16}$$

where $\alpha > 0$ and $\gamma > 0$ are parameters. This simple model scales the relative statistical variation in delay, relative to the mean delay, using a power law. It is inspired by process parameter variation models such as the empirical threshold voltage variation model [54] and Pelgrom model [70], which predict a decrease in device parameter variation with increasing device area, due to spatial averaging. The parameter γ gives the relative variation for a minimum sized gate (*i.e.*, $x_i = 1$), and the parameter α accounts for the space averaging effects of process and device parameter variations. We used the model parameters

$$\alpha = 1.0, \qquad \gamma = 0.15.$$

This means that for a minimum sized gate, the delay standard deviation is 15% of its mean, and that this ratio decreases with increasing gate size. In our example, the delay distributions are Gaussian. (Since $\sigma/\mu \leq 0.15$, the probability of a negative delay was vanishingly small.) Like our mean delay model, this statistical model is chosen for simplicity, not accuracy. In any case, modeling the statistics of gate delay is an area of active research, with many open issues; see, *e.g.*, [52, 63, 67, 68].

For the optimization problems, we imposed a constraint on the area, as well as lower bounds on the scale factors:

$$A \leq 12000, \qquad 1 \leq x_i, \quad i = 1, \ldots, n. \tag{17}$$

The minimum area, *i.e.*, the area of the adder with all gates minimum sized is 3842, so a uniform allocation gives $x_i = 12000/3842 = 3.12$. The load capacitance of each primary output is taken as $C_i^{\mathrm{L}} = 6$.

Our delay mean and variance are both posynomial functions of x, as are our constraint functions. It follows that the surrogate DAN optimization problem can be formulated as a (generalized) geometric program, and therefore solved globally and efficiently (see [20]). To give some idea of the efficiency, the resulting GPs have around 1000 variables and 3000 constraints, and are solved using MOSEK [58] in two seconds on a 1.2GHz Pentium IV PC.

8 More numerical experiments

In this section we carry out some more numerical experiments with our circuit design example to test how sensitive the robust design is to errors in the model and assumptions.

8.1 Gate delay distribution shape

The numerical results above were obtained with independent Gaussian variation in the gate delays. To see how sensitive our robust design is to the shape of the gate delay distribution, we use MC simulation to find the cycle time PDFs of the nominal optimal and the robust

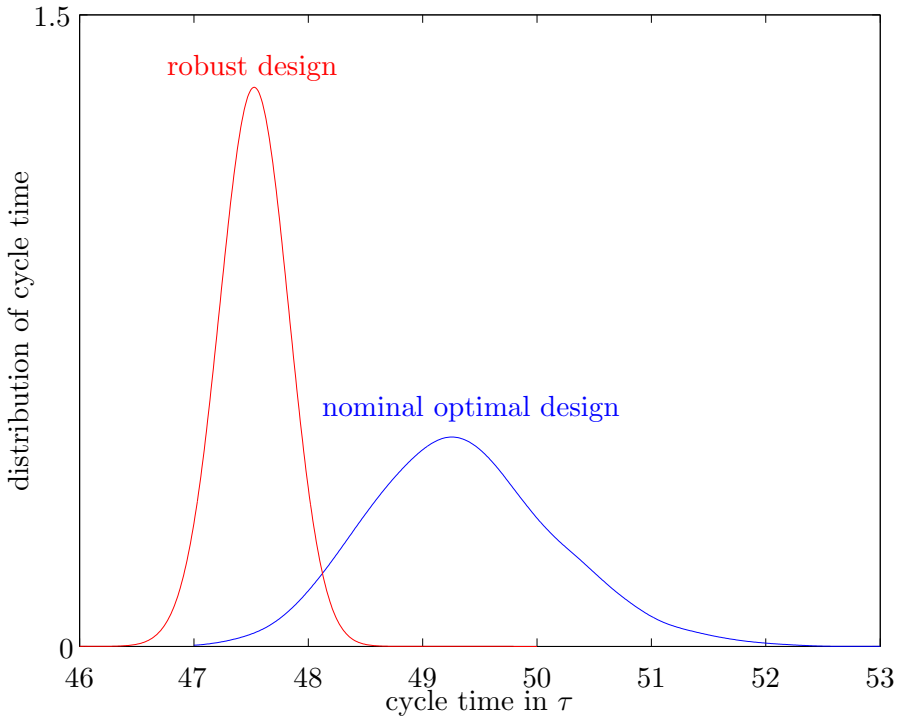

Figure 15: Cycle time PDF for nominal optimal design and robust design ($\kappa = 1.5$), with uniform gate delay distributions.

design with *uniform* gate delay distributions. (All other model assumptions and parameters are the same.) The results are shown in figure 15.

Compared to the results with Gaussian gate delays, shown in figure 7 (on page 21), we see that the nominal optimal design is not quite as bad. Still, the robust design is far superior to the nominal optimal design.

It is not difficult to see that uniform distributions meet the assumption for establishing the lower bound (5) with $\alpha = 0.95$. The small gap of 3.7% percent between the nominal optimal cycle time (45.9) and the 95% cycle time (47.9) of the robust design therefore shows that the robust design is within 4.36% percent of the global optimum.

8.2 Correlations among gate delays

In the results reported above, we assume that the gate delay distributions are uncorrelated. In fact, several mechanisms (spatial correlation, process parameter variation) can result in correlation among gate delays (see, *e.g.*, [52]). We have carried out many analyses of the nominal optimal and robust designs with different models of correlation, and found that correlations typically make the nominal design not as bad as in the uncorrelated case, but in all cases the robust design is substantially better.

We report here only the (typical) results for one such test. We use the following simple

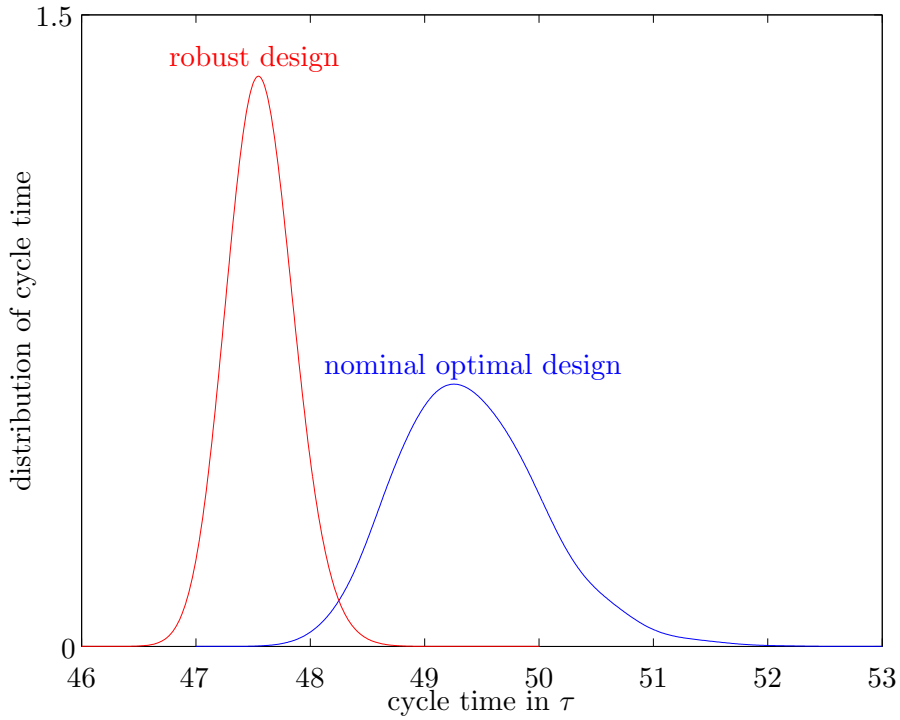

Figure 16: Cycle time PDF for nominal optimal design and robust design, with correlation among adjacent gates.

model: two gates, say, i and j, are correlated by 30% if they are adjacent (*i.e.*, one drives the other); otherwise they are uncorrelated. The results are shown in figure 16. As can be seen from figure 16, the difference between the nominal optimal delay and the 95% cycle time of the robust design is small, meaning that the robust design is quite close to the global optimum. (Since the correlation coefficients are positive, the nominal optimal delay is still a lower bound on the optimal 95% cycle time.)

9 Conclusions

We have proposed a heuristic method for approximately solving a SAN optimization problem, based on solving one or more surrogate DAN optimization problems, using Monte Carlo analysis to verify the performance of the designs, and choosing the best one. Our computational experience with the method so far suggests that the heuristic method

- is often far superior to the nominal optimal design (*i.e.*, the design obtained by ignoring statistical variation),
- is not very sensitive to the details of the activity duration distributions or correlations among them, and
- often yields a design that is provably close to the global optimum.

We certainly do not claim that the method works well in all cases; we are only claiming that it seems to work very well on the (circuit design) examples we have examined. One important future research topic is to identify general classes of SAN problems for which the method works well.

One natural question is why a method this simple should work as well as it does on the problem instances we have examined. We believe that the phenomenon is similar to Tikhonov regularization, which is another case in which a simple modification of an optimization problem yields a surprisingly robust result. In the most basic form of regularization, the goal is to estimate x, given a measurement $y \approx Ax$. The standard least-squares estimate, obtained by minimizing $\|Ax - y\|^2$, can be very sensitive to statistical errors in A. In Tikhonov regularization, we choose the estimate by minimizing $\|Ax - y\|^2 + \mu\|x\|^2$, where $\mu > 0$ is a parameter. This modified problem can be interpreted several ways; for example, it corresponds to minimizing $\mathbf{E}\|Ax-y\|^2$ when the entries of A are zero mean and uncorrelated, with variance μ/n (see, *e.g.*, [21]). Tikhonov regularization has some of the same features we have observed in our heuristic method [21, §6.4].

- It often works very well, compared to the simple least-squares estimator.
- The choice of the parameter μ is not particularly critical.
- The method often works well even when the underlying statistical assumptions are not correct (for example, there is some correlation among the entries of A).

We should also comment on a special case: when the standard deviation of each durations is proportional to the mean. In this case, the robust heuristic design, with constant κ, is the same as the nominal optimal design. In particular, the robust heuristic method cannot improve the performance over the nominal optimal design. Our computational experience so far suggests that in this case, the nominal optimal design (which is also the robust heuristic design) is quite robust to the variation in the duration distributions; we have observed in many cases that the lower bounds described above verify that the design is close to the global

optimum. But a more extensive comparison between the nominal optimal design and the true solution of the SAN optimization problem remains to be carried out.

For circuit design, we have already extended the method described here to problems with more accurate delay models, with different delay models for rising and falling signals, different input/output pairs for each gate, and effects of signal slope [69]. In the design problem we size individual devices (as opposed to whole gates as in the example considered here), and take into account power as well as area.

We mention one application in digital circuit design, suggested by Abbas El Gamal, that we will be exploring. In the simple model, the statistical variation can be thought of as coming from device parameter variations (which therefore decrease with increasing device sizes). El Gamal has suggested that we develop a statistical model of the effects of crosstalk and coupling (which comes from the interconnect, not the devices). We model the delay of a gate (and its output net) statistically; the variation in delay is due to crosstalk and coupling from other nets. In this case the variance of the delay depends on layout, and not just device sizes. The heuristic robust method then corresponds to a crosstalk-aware design method, which allocates a bit more margin to gates and paths that are near critical and have long nets, and so are likely to be victims of crosstalk.

Acknowledgments

This work was supported in part by the MARCO Focus Center for Circuit & System Solutions (C2S2, www.c2s2.org), under contract 2003-CT-888.

References

[1] H. Abdel-Malek and J. Bandler. Yield optimization for arbitrary statistical distributions: Part I-Theory. *IEEE Transactions on Circuits and Systems*, 27(4):245–253, April 1980.

[2] H. Abdel-Malek and J. Bandler. Yield optimization for arbitrary statistical distributions: Part II-Implementation. *IEEE Transactions on Circuits and Systems*, 27(4):253–262, April 1980.

[3] M. Abramowitz and I. Stegun, editors. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. John Wiley & Sons, New York, 1972.

[4] A. Agarwal, D. Blaauw, and V. Zolotov. Statistical timing analysis for intra-die process variations with spatial correlations. In *International Conference on Computer-Aided Design*, pages 900–907, San Jose, CA, USA, November 2003.

[5] A. Agarwal, V. Zolotov, and D. Blaauw. Statistical timing analysis using bounds and selective enumeration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(9):1243–1260, 2003.

[6] K. Anklesaria and Z. Drezner. A multivariate approach to estimating the completion time for PERT networks. *Journal of the Operational Research Society*, 37:811–815, 1986.

[7] X. Bai, C. Visweswariah, P. Strenski, and D. Hathaway. Uncertainty-aware circuit optimization. In *Proc. of 39th IEEE/ACM Proc. Design Automation Conference*, pages 58–63, New Orleans, LA, June 2002.

[8] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805, 1998.

[9] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25:1–13, 1999.

[10] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization. Analysis, Algorithms, and Engineering Applications*. Society for Industrial and Applied Mathematics, 2001.

[11] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, second edition, 1999.

[12] D. Bertsekas, A. Nedic, and A. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, 2003.

[13] D. Bertsimas, K. Natarajan, and C.-P. Teo. Probabilistic combinatorial optimization: Moments, semidefinite programming and asymptotic bounds. *SIAM Journal on Optimization*, 15(1):185–209, 2004.

[14] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming, Series B*, 98:49–71, 2003.

[15] D. Bertsimas and A. Thiele. A robust optimization approach to supply chain management. Manuscript, 2003.

[16] S. Bhardwaj, S. Vrudhula, and D. Blaauw. TAU: Timing analysis under uncertainty. In *International Conference on Computer-Aided Design*, pages 615–620, San Jose, CA, USA, November 2003.

[17] J. Birge and M. Maddox. Bounds on expected project tardiness. *Operations Research*, 43:838–850, 1995.

[18] R. Bowman. Efficient estimation of arc criticalities in stochastic activity networks. *Management Science*, 41(1):58–67, 1995.

[19] S. Boyd, S.-J. Kim, D. Patil, and M. Horowitz. Digital circuit optimization via geometric programming, 2005. Manuscript. Available from `www.stanford.edu/~boyd/~gp_digital_ckt.html`.

[20] S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi. A tutorial on geometric programming, 2004. Manuscript. Available from `http://www.stanford.edu/boyd/~gp_tutorial.html`.

[21] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[22] E. Bozorgzadeh, S. Ghiasi, A. Takahashi, and M. Sarrafzadeh. Optimal integer delay budget assignment on directed acyclic graphs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(8):1184–1199, August 2004.

[23] A. Brambilla and P. Maffezzoni. Statistical method for the analysis of interconnects delay in submicrometer layouts. *IEEE Transactions on Compute-Aided Design of Integrated Circuits and Systems*, 20(8):957–966, 2001.

[24] J. Burt and M. Garman. Conditional Monte Carlo: A simulation technique for stochastic network analysis. *Management Science*, 18(1):207–217, September 1971.

[25] A. Charnes, W. Cooper, and G. Thompson. Critical path analyses via chance constrained and stochastic programming. *Operations Research*, 12:460–470, 1964.

[26] A. Conn, P. Coulman, R. Haring, G. Morrill, C. Visweswariah, and C. Wu. JiffyTune: Circuit optimization using time-domain sensitivities. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(12):1292–1309, December 1998.

[27] A. Conn, I. Elfadel, W. Molzen Jr., P. O'Brien, P. Strenski, C. Visweswariah, and C. Whan. Gradient-based optimization of custom circuits using a static-timing formulation. In *Proc. of 36th IEEE/ACM Proc. Design Automation Conference*, pages 452–459, June 1999.

[28] A. Conn, N. Gould, and Ph. L. Toint. *LANCELOT: A Fortran package for large-scale nonlinear optimization (Release A)*, volume 17 of *Springer Series in Computational Mathematics*. Springer-Verlag, 1992.

[29] E. Davis. Resource allocation in project network models–A survey. *The Journal of Industrial Engineering*, 17(4), 1966.

[30] L. Devroye. Inequalities for the completion times of stochastic PERT networks. *Mathematics of Operations Research*, 4(4):441–447, 1979.

[31] B. Dodin. Determining the k most critical paths in PERT networks. *Operations Research*, 32:859–877, 1984.

[32] B. Dodin. Bounding the project completion time distribution in PERT networks. *Operations Research*, 33:862–881, 1985.

[33] B. Dodin and S. Elmaghraby. Approximating the criticality indices of the activities in PERT networks. *Management Science*, 31(2):207–223, Feb. 1985.

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

[35] S. Elmaghraby. *Project Planning and Control by Network Models*. John Wiley and Sons, 1977.

[36] D. Eppstein. Finding the k shortest paths. *SIAM J. Computing*, 28(2):652–673, 1998.

[37] J. Esary, F. Proschan, and D. Walkup. Association of random variables with applications. *Annals of Mathematical Statistics*, 38:71466–1474, 1967.

[38] J. Fishburn and A. Dunlop. TILOS: A posynomial programming approach to transistor sizing. In *IEEE International Conference on Computer-Aided Design: ICCAD-85. Digest of Technical Papers*, pages 326–328. IEEE Computer Society Press, 1985.

[39] S. Foldes and F. Sourmis. PERT and crashing revisited: Mathematical generalizations. *European Journal of Operational Research*, 64:286–294, 1993.

[40] B. Fox. Integrating and accelerating tabu search. *Annals of Operations Research*, 41:47–67, 1994.

[41] L. El Ghaoui and H. Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications*, 18(4):1035–1064, 1997.

[42] P. Glasserman. *Monte Carlo Methods in Financial Engineeting*. Springer-Verlag, 2003.

[43] J. Hagstrom and N. Jane. Computing the probability distribution of project duration in a PERT network. *Networks*, 20:231–244, 1990.

[44] H. Hashimoto and H. Onodera. A performance optimization method by gate sizing using statistical static timing analysis. In *Proc. of ACM International Symposium on Physical Design*, pages 111–116, San Diego, CA, April 2000.

[45] U. Heller. On the shortest overall duration in stochastic acyclic network. *Methods of Operations Research*, 42:85–104, 1981.

[46] C. Hongliang and S. Sapatnekar. Statistical timing analysis considering spatial correlations using a single PERT-like traversal. In *International Conference on Computer-Aided Design*, pages 621–625, San Jose, CA, USA, November 2003.

[47] K.-L. Hsiung, S.-J. Kim, and S. Boyd. Robust geometric programming via piecewise linear approximation, April 2004. Revised for publication in *Mathematical Programming*.

[48] J. Jess, K. Kalafala, S. Naidu, R. Otten, and C. Visweswariah. Statistical timing for parametric yield prediction of digital integrated circuits. In *Proc. of 40th Proc. IEEE/ACM Design Automation Conference*, pages 343–347, Anaheim, CA, June 2003.

[49] H.-F. Jyu, S. Malik, S. Devadas, and K. Keutzer. Statistical timing analysis of combinational logic circuits. *IEEE Transactions on VLSI Systems*, 1(2):126–137, 1993.

[50] S. Knowles. A family of adders. In *Proceedings of 14th IEEE Symposium on Computer Arithmetic*, pages 30–34. IEEE Computer Society Press, 1999.

[51] J. Kouloheris. *Empirical Study of the Effect of Cell Granularity on FPGA and Performance*. PhD thesis, Stanford University, 1993.

[52] J.-J. Liou, A. Krstić, Y.-M. Jiang, and K.-T. Cheng. Modeling, testing, and analysis for delay defects and noise effects in deep submicron devices. *IEEE Transactions on Compute-Aided Design of Integrated Circuits and Systems*, 22(6):756–769, 2003.

[53] A. Ludwig, R. Möhring, and F. Stork. A computational study on bounding the makespan distribution in stochastic project networks. *Annals of Operations Research*, 102:49–64, 2001.

[54] S. Ma, A. Keshavarzi, V. De, and R. Brews. A statistical model for extracting geometric sources of transistor performance variation. *IEEE Transactions on Electronic Devices*, 51(1):36–41, 2004.

[55] W. Maly. Computer-aided design for VLSI circuit manufacturability. *Proceedings of IEEE*, 78(2):356–392, February 1990.

[56] I. Mejilson and A. Nadas. Convex majorization with an application to the length of critical path. *Journal of Applied Probability*, 16:671–677, 1979.

[57] A. Mingozzi, V. Maniezzo, S. Ricciardelli, and L. Bianco. An exact algorithm for the resource constrained project scheduling problem based on a new mathematical formulation. *Management Science*, 44:714–729, 1998.

[58] MOSEK ApS. *The MOSEK Optimization Tools Version 2.5. User's Manual and Reference*, 2002. Available from `www.mosek.com`.

[59] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, Boston, 2003.

[60] Y. Nesterov and A. Nemirovsky. *Interior-Point Polynomial Methods in Convex Programming*, volume 13 of *Studies in Applied Mathematics*. SIAM, Philadelphia, PA, 1994.

[61] A. Neumaier. Solving ill-conditioned and singular linear systems: A tutorial on regularization. *SIAM Review*, 40(3):636–666, 1998.

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

[63] K. Okada, K. Yamaoka, and H. Onodera. A statistical gate delay model for intra-chip and inter-chip variabilities. In *Proc. of 40th IEEE/ACM Proc. Design Automation Conference*, pages 31–36, Anaheim, CA, June 2003.

[64] L. Opalski and M. Styblinski. Generalization of yield optimization problems: Maximum income approach. *IEEE Transactions on Computer-Aided Design*, 5(2):346–360, April 1986.

[65] M. Orshansky, J. Chen, and C. Hu. Direct sampling methodology for statistical analysis of scaled CMOS technologies. *IEEE Transactions on Semiconductor Manufacturing*, 12(4):403–408, November 1999.

[66] M. Orshansky and K. Keutzer. A general probabilistic framework for worst case timing analysis. In *Proc. of 39th. IEEE/ACM Design Automation Conference*, pages 556–561, New Orleans, LA, June 2002.

[67] M. Orshansky, L. Milor, P. Chen, K. Keutzer, and C. Hu. Impact of spatial intrachip gate length variability on the performance of high-speed digital circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(5):544–553, May 2002.

[68] M. Orshansky, L. Milor, and C. Hu. Characterization of spatial intrafield gate CD variability, its impact on circuit performance and spatial mask-level correction. *IEEE Transactions on Semiconductor Manufacturing*, 17(1):2–11, February 2004.

[69] D. Patil, Y. Yun, S.-J. Kim, S. Boyd, and M. Horowitz. A new method for robust design of digital circuitss, 2004. To be presented at *International Symposium on Quality Electronic Design (ISQED) 2005*.

[70] M. Pelgrom. Matching properties of MOS transistors. *IEEE Journal of Solid State Circuits*, 24(5):1433–1439, October 1989.

[71] M. Pich, C. Loch, and A. De Meyer. On uncertainty, ambiguity, and complexity in project management. *Management Science*, 48(8):1008–1023, August 2002.

[72] E. Plambeck, Fu B.-R, S. Robinson, and R. Suri. Sample-path optimization of convex stochastic performance functions. *Mathematical Programming*, 75(2):137–176, 1996.

[73] S. Pollalis. *Computer-aided Project Management: A Visual Scheduling and Control System*. Viewweg Verlag, Wiesbaden, Germany, 1993.

[74] A. Prekopa. *Stochastic Programming*. Kluwer Academic Publishers, 1983.

[75] J. Provan and M. Ball. The complexity of counting cuts and of the probability that a graph is connected. *SIAM Journal on Computing*, 12:777–788, 1983.

[76] P. Robillard and M. Trahan. The completion times of PERT networks. *Operations Research*, 25:15–29, 1976.

[77] J. Robinson. Some analysis techniques for asynchronous multiprocessor algorithms. *IEEE Transactions on Software Engineering*, 5(1):24–31, 1979.

[78] R. Rockafellar and S. Uryasev. Optimization of conditional Value-at-Risk criterion. *The Journal of Risk*, 2(3):21–41, 2000.

[79] S. Sapatnekar. Wire sizing as a convex optimization problem: Exploring the area-delay tradeoff. *IEEE Transactions on Computer-Aided Design*, 15:1001–1011, August 1996.

[80] S. Sapatnekar. Power-delay optimization in gate sizing. *ACM Transactions on Design Automation of Electronic Systems*, 5(1):98–114, 2000.

[81] S. Sapatnekar, V. Rao, P. Vaidya, and S. Kang. An exact solution to the transistor sizing problem for CMOS circuits using convex optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(11):1621–1634, November 1993.

[82] R. Serfling. *Approximation Theorems of Mathematical Statistics*. Wiley, New York, 1980.

[83] A. Shogan. Bounding distributions for a stochastic PERT network. *Networks*, 7:359–381, 1977.

[84] R. Slowinski. Two approaches to problems of resource allocation among project activities: A comparative study. *Journal of the Operational Research Society*, 31:711–723, 1980.

[85] R. Van Slyke. Monte Carlo methods and the PERT problem. *Operations Research*, 11:839–860, 1963.

[86] R. Van Slyke and R. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17:638–663, 1969.

[87] M. Styblinski and L. Opalski. Algorithms and software tools for IC yield optimization based on fundamental fabrication parameters. *IEEE Transactions on Computer-Aided Design*, 5(1):79–89, January 1986.

[88] R. Sullivan and J. Hayya. A comparison of the method of bounding distributions (MBD) and Monte Carlo simulation for analyzing stochastic acyclic networks. *Operations Research*, 28(3):614–617, May-Jun. 1980.

[89] I. Sutherland, B. Sproul, and D. Harris. *Logical Effort: Designing Fast CMOS Circuits*. Morgan Kaufmann Publishers, San Francisco, CA, 1999.

[90] A. van der Vaart. *Asymptotics Statistics*. Cambridge University Press, 1998.

[91] R. Vanderbei. LOQO user's manual. Technical Report SOL 92–05, Dept. of Civil Engineering and Operations Research, Princeton University, Princeton, NJ 08544, USA, 1992.

[92] C. Visweswariah. Death, taxes and failing chips. In *Proc. of 40th IEEE/ACM Design Automation Conference*, pages 343–347, Anaheim, CA, June 2003.

[93] S. Wallace. Bounding the expected time-cost curve for a stochastic PERT network from below. *Operations Research*, 8:89–94, 1989.

[94] G. Weiss. Stochastic bounds on distributions of optimal value functions with applications to PERT, network flows and reliability. *Operations Research*, 34(4):595–605, 1986.

[95] K. White, W. Trybula, and R. Athay. Design for semiconductor manufacturing-perspective. *IEEE Transactions on Components, Packaging, and Manufacturing Technology-Part C*, 20(1):58–72, January 1997.

[96] P. Yang, D. Hocevar, P. Fox, C. Machala, and P. Chatterjee. An integrated and efficient approach for MOS VLSI statistical circuit design. *IEEE Transactions on Computer-Aided Design*, 5(1):5–14, January 1986.