Statistical Static Timing Analysis using Markov Chain Monte Carlo

Yashodhan Kanoria*, Subhasish Mitra*† and Andrea Montanari*‡
Departments of Electrical Engineering*, Computer Science† and Statistics‡, Stanford University
Email: {ykanoria, subh, montanari}@stanford.edu

***Abstract*—We present a new technique for statistical static timing analysis (SSTA) based on Markov chain Monte Carlo (MCMC), that allows fast and accurate estimation of the right-hand tail of the delay distribution. A "naive" MCMC approach is inadequate for SSTA. Several modifications and enhancements, presented in this paper, enable application of MCMC to SSTA. Moreover, such an approach overcomes inherent limitations of techniques such as importance sampling and Quasi-Monte Carlo. Our results on open source designs, with an independent delay variation model, demonstrate that our technique can obtain more than an order of magnitude improvement in computation time over simple Monte Carlo, given an estimation accuracy target at a point in the tail. Our approach works by providing a large number of samples in the region of interest. Open problems include extension of algorithm applicability to a broader class of synthesis conditions, and handling of correlated delay variations. In a broader context, this work aims to show that MCMC and associated techniques can be useful in rare event analyses related to circuits, particularly for high-dimensional problems.**

I. Introduction

Increasing effects of process variations have strengthened the case for a move from static timing analysis (STA) to statistical static timing analysis (SSTA). SSTA aims to use statistical techniques to analyze the timing performance of a chip in the presence of statistical variations in underlying parameters. Various approaches for SSTA have been suggested. [1], [2] are recent surveys of the field. We defer a discussion of related work to Section V.

Analysis of timing performance boils down to a study of the distribution of the worst case circuit delay. An important feature of the delay distribution estimation problem is that we are typically interested in the right-hand tail of the distribution, i.e., the tail corresponding to large delay values (see Figure 1). Given a model, we seek statistical information about the 'bad' (timing violating) region in the parameter space. This information can be used for three purposes:

1) Yield calculation: Timing yield is the fraction of manufactured chips that work to frequency specification.
2) Critical path identification: Probabilistic timing critical paths are further used for: (i) Design improvement eg. resizing (ii) Test pattern generation. The *timing test* requires that a chip satisfies timing requirements for all test patterns in the test set.
3) Shipped-product Quality Loss (SPQL) estimation: SPQL is the fraction of chips that pass the timing test but, in fact, violate timing (see, for example, [3]).

For problems 2 and 3, especially, we aim for very high confidence so that very few bad chips are shipped (overall defective parts per million goals range from 5 to 200, so confidence must be $> 99.98\%$). Thus, very detailed and accurate information about the 'bad' regions is needed. Further, with rapid increase in both chip complexity and statistical variations, fine-grained statistical optimization (during design time or during system operation) of individual blocks may be necessary [4]. Potentially, even to obtain a reasonable yield target for a chip, one may have to study very rare events at the per block level. This is especially true with rapid increase in intra-die variations [1], leading to an increase in uncorrelated failure of blocks. For example, for a chip containing 100 blocks, which fail independently of each other, nearly 99.95% yield is needed at a per block level, in order to achieve 95% yield at the chip level. As such, there is much to be gained by viewing the SSTA problem as a large deviation problem, i.e., one in which a rare event (poor timing performance) is to be studied in terms of its probability and its causes.

Y. Kanoria is supported by a 3Com Corporation Stanford Graduate Fellowship. This research was partially supported by NSF, grants CCF-0743978 and CCF-0915145, and a Terman fellowship.

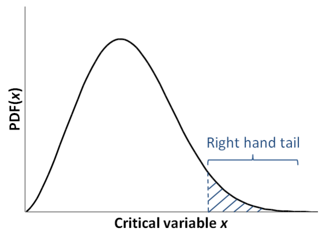


Fig. 1. Schematic showing 'right-hand tail' of distribution

The SSTA problem is (increasingly) high dimensional. Moreover, depending on the variation model, it can have a large number of 'bad' regions. For example, consider a model with independent gate delay variations. The number of important dimensions scales almost linearly with circuit size (one dimension for each gate that lies on *some* critical path). Further, there are a huge number of 'bad' regions, corresponding to different paths being critical, that are not connected (for disjoint paths) or very weakly connected (for overlapping paths) to each other. In worst case, this number of 'bad' regions can grow exponentially corresponding to the explosion of critical paths. Thus, the problem is 'hard', making it difficult to obtain large improvements over simple Monte Carlo sampling. Techniques such as Statistical Blockade (SB) [5] and importance sampling (e.g. [6]) used for other rare event problems, notably SRAM failure analysis, are not useful for SSTA as discussed in Section V.

Markov chain Monte Carlo is a popular approach for estimating distributions that are hard to compute exactly. In particular, it can be used effectively for solving a variety of rare event problems, including those with high dimensionality, and a multitude of 'bad' regions. We demonstrate the power of MCMC in this work by applying it to the SSTA problem. Our algorithm retains the well-documented advantages of Monte Carlo over other approaches for SSTA, while exploiting the large deviation nature of problem.

We work with a simplified model in which we assume the delays of gates vary independently at random. This allows for simplicity, while showing the power of our approach on the hardest (independent random) component of the statistical timing problem. This is discussed in Section V. Also, our work may be directly applicable to sub-threshold circuits, where independent V_T variations [7] dominate delay variation.

Our contributions are:

1) Introduction of MCMC as an effective technique for rare event problems in circuits, even those having high dimensionality and/or a large number of 'bad' regions
2) Development of a version of MCMC for SSTA when gates have independent delay variations, and showing over 10X reduction in computational effort over regular Monte Carlo for tail probability estimation in a specific setting (See Section V for a discussion on our inability to provide broad performance guarantees across settings).

The organization of the paper is as follows — Section II describes the model. We present our algorithm in Section III. Empirical results are stated in Section IV. Section V contains a discussion of our approach and related work. We conclude in Section VI. The Appendix briefly discusses MCMC and its use for rare event analysis.

II. Model

We consider a combinational logic circuit with independently varying gate delays (cf. Sections I, V). We assume that gate delays do not depend on the nature of the transition (rising or falling), or the input on which the transition is occurring. As with other Monte Carlo based approaches, we can use the core STA engine directly, so neither of these assumptions is crucial. We also assume the interconnect delays are negligible (we can easily account for non-zero delays including statistical variations).

We associate a 'gate delay' d_v with each gate v, the set of gates being denoted by $\mathcal{V}_{\text{g}}$. Denote by I the set of inputs and pseudo-primary inputs, and by O the set of outputs and pseudo-primary outputs. Let $\mathcal{V} = \mathcal{V}_{\text{g}} \cup I \cup O$. Let $d_v, v \in \mathcal{V}_{\text{g}}$ have a distribution $p_v(\cdot)$, denoted by $d_v \sim p_v(\cdot)$, and let $\mathbb{P}$ be the corresponding product distribution. We denote the set of allowed gate delay vectors $\bar{d} = (d_v, v \in \mathcal{V}_{\text{g}})$ by $\mathcal{S}_d$. We define arrival time D_v for each $v \in \mathcal{V}$ corresponding to the longest path arrival time at v.

Let D_{req} denote the required time for the entire circuit i.e. all outputs are assumed to have the same required time (add artificial delays to output nodes to achieve equality). Let D be the circuit arrival time i.e. the largest arrival time at an output. Thus, 'bad' events corresponding to the circuit not meeting timing are characterized by $(D > D_{\text{req}})$. Our algorithm estimates yields simultaneously for a range of D_{req} values corresponding to high yields.

III. Using Markov chain Monte Carlo

We begin by stating a baseline MCMC-based algorithm for SSTA called MCMC-B in Section III-A. A key fact is that both the 'tilted distribution' (with parameter β) and the algorithm MCMC-B presented in Section III-A are only particular choices from a very large class of possibilities for both the tilted distribution and the algorithm. In subsequent sections, we describe modifications to both MCMC-B and our tilted distribution, to speed up mixing in our Markov chain, and thus develop an effective algorithm for SSTA.

A. A baseline MCMC algorithm for SSTA

Let $\mathbb{Q}_X(\cdot)$ denote the density of any random variable X wrt distribution $\mathbb{Q}$. Associated uniquely with each $\bar{d}$ is a circuit arrival time $D(\bar{d}) = D$. We wish to accurately estimate the right-tail of $\mathbb{P}_D(\cdot)$. To achieve this, we want to sample from our state space $\mathcal{S}_d$ with a bias towards higher D configurations (the same philosophy leads to techniques like importance sampling). Hence, we choose a 'tilted' distribution $\pi(\cdot)$ with a bias towards larger values of D. A simple choice is

$$\pi_{\bar{d}}(\bar{x}) = \tfrac{1}{Z(\beta)}\mathbb{P}_{\bar{d}}(\bar{x})\exp(\beta D(\bar{d})), \qquad \text{for all } \bar{x} \in \mathcal{S}_d \tag{1}$$

for some $\beta > 0$, $Z(\beta)$ being a normalizing constant that can be computed as

$$Z(\beta) = \left(\int_{-\infty}^{\infty} \pi_D(x)\exp(-\beta x)\,\mathrm{d}x\right)^{-1} \tag{2}$$

where $\pi_D(\cdot)$ represents the density of D under $\pi(\cdot)$.

We have,

$$\pi_D(x) = \tfrac{1}{Z(\beta)}\mathbb{P}_D(x)\exp(\beta x) \tag{3}$$

so, the tail of $\mathbb{P}_D(\cdot)$ can be inferred from $\pi_D(\cdot)$ using

$$\mathbb{P}(D \geq D_0) = Z(\beta)\textstyle\int_{D_0}^{\infty} \pi_D(x)\exp(-\beta x)\,\mathrm{d}x \tag{4}$$

As an example, suppose $\mathbb{P}_D(\cdot)$ is normal with mean 10 and variance 1. Suppose $\beta = 2$. Then from Eq. (3), $\pi_D(\cdot)$ is also a normal distribution with mean 12 and variance 1. In fact, any normal distribution, when tilted as per Eq. (3), yields a normal shifted by β to the right.

(2) and (4) provide us with a way of estimating the tail of $\mathbb{P}_D(\cdot)$ provided we can sample from $\pi_D(\cdot)$. MCMC provides a means to sample from $\pi_D(\cdot)$.

We now state a particular simple MCMC algorithm for estimating the right-tail of $\mathbb{P}_D(\cdot)$ in Table I. The superscripts in the algorithm denote the time step.

TABLE I
A baseline MCMC-based algorithm for SSTA

	MCMC-B
1	Choose an arbitrary starting state $\bar{d}^0 \in \mathcal{S}_d$, $\beta > 0$ and N
2	Compute $D^0 = D(\bar{d}^0)$
3	**for t=1 to N**
4	Choose $v \in \mathcal{V}_{\text{g}}$ uniformly at random.
5	Choose new delay $d'_v \sim p_v(\cdot)$ #other gate delays stay unchanged
6	Compute D' due to change $d_v^{t-1} \rightarrow d'_v$
7	Draw $r \sim U[0,1]$
8	**if** $\left(r < \exp\left(\beta(D' - D^{t-1})\right)\right)$
9	$d_v^t \leftarrow d'_v$ # accept transition
10	**else**
11	$d_v^t \leftarrow d_v^{t-1}$ # reject transition
12	Compute $D^t = D(\bar{d}^t)$
13	**end**
14	Compute $\hat{\pi}_D(\cdot) =$ empirical dist. of $\left(D^0, \ldots, D^N\right)$
15	Compute tail of $\hat{\mathbb{P}}_D(\cdot)$ using Eqs. (2), (4)

MCMC-B is based on the popular Metropolis-Hastings algorithm (see [8]). Successive 'proposals' of the form $d_v \rightarrow d'_v$ for some $v \in \mathcal{V}_{\text{g}}$ are probabilistically accepted or rejected, so that the state progresses in a Markov chain fashion. The key idea is that the chosen Markov chain has a unique equilibrium distribution $\pi(\cdot)$, and can hence be used to draw approximate independent identically distributed (iid) samples from $\pi(\cdot)$ (cf. Appendix).

B. Level-wise updates

With MCMC-B, proposing a change in the delay of each gate at least once in steps 4 and 5 requires $\sim (|\mathcal{V}_g| \ln |\mathcal{V}_g|)$ steps. Moreover, even with a change in just one gate delay, arrival times for the entire fanout cone must be updated to compute D.

Instead, gates in $\mathcal{V}_g$ are assigned 'levels' by grouping them such that nodes with a level lower than l are predecessors and nodes with a level greater than l are successors of nodes at level l. The number of levels R is the depth of the circuit. Now, in steps 4 and 5 of MCMC-B, we select a level uniformly at random instead of a single gate, and propose new delays for all gates at that level, drawn independently as per their respective distributions. The probability of accepting a move remains $\min(1, \exp(\beta(D' - D)))$. The delay of exactly one gate along each path from input to output changes in each proposed move, hence the change $(D - D')$ is typically small, leading to a good acceptance probability. Computational effort per step is linear in circuit size, whereas typically $R \ll |\mathcal{V}_g| \Rightarrow R \ln R \ll |\mathcal{V}_g| \ln |\mathcal{V}_g|$. Hence, savings are large.

C. Parallel tempering

We create k tilted distributions as in Eq. (1).

$$\pi_i(\bar{d}) \propto \mathbb{P}(\bar{d}) \exp(f_i(D)) \qquad i = 1, \ldots, k \tag{5}$$

where $f_i(\cdot)$ are functions monotonic non-decreasing in D. Further, we ensure that for any $j > i$,

$$f_j(D_2) - f_j(D_1) \geq f_i(D_2) - f_i(D_1) \qquad \forall\, D_2 > D_1 \tag{6}$$

Informally, the exponent $f_i(\cdot)$ only gets 'steeper' as i increases. The acceptance probability for the Markov chain corresponding to $\pi_i(\cdot)$ is now $\min(1, \exp(f_i(D^{\text{new}}) - f_i(D^{\text{old}})))$.

Parallel tempering/simulated tempering (see [8]) allows us to run multiple Markov chains at different values of i, with regular 'swap' moves occurring between states of the different chains (see Figure 2). For simplicity, we describe a version of parallel tempering with just two chains running i.e. $k = 2$. At each time step, we choose either a 'normal' move or a 'swap' move If a normal move is chosen, each Markov chain has an update according to its own update rule, consistent with its particular $f_i(\cdot)$. If a swap move is chosen, the proposal is to move from $(\bar{d}_1, \bar{d}_2)$ to $(\bar{d}_2, \bar{d}_1)$. This is accepted with probability $\min(1, \exp(-(f_2(D_2) - f_2(D_1)) + (f_1(D_2) - f_1(D_1))))$.

Most importantly, the ith chain continues to have the target equilibrium distribution $\pi_i(\cdot)$. The 'swap' moves help tremendously with mixing, since the chains with flatter exponents (smaller values of i) move rapidly over the state space, and swaps with chains at steeper exponents(larger values of i) ensure that those chains do not get 'stuck' in local energy 'valleys'. In analogy with physics, small values of i correspond to higher temperatures, whereas large values of i correspond to lower temperatures. We denote this notion of temperature by T_i for the ith chain, with $T_1 > T_2 > \ldots > T_k$.

This can be generalized to $k > 2$ with swaps possible between chains having adjacent values of i or even more general swaps.

Figure 4(a) shows empirical distributions obtained from parallel tempering with 3 chains. A significant 'overlap' between the distributions at adjacent i's, as seen in this figure, ensures a good swap acceptance probability. Given a parallel tempering formulation that mixes, we can use empirical estimates $\hat{\pi}_{D,1}(\cdot), \hat{\pi}_{D,2}(\cdot), \ldots$ to derive estimates $\hat{Z}_1, \hat{Z}_2, \ldots$ and $\hat{\mathbb{P}}_D(\cdot)$ whose tail we care about.

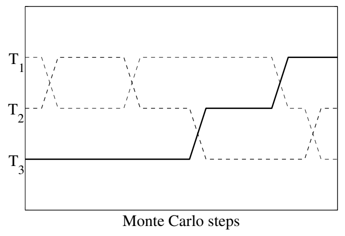


Fig. 2. Schematic showing evolution of Parallel tempering with 3 chains

D. Other tweaks

We saw in Section III-A that when a Gaussian distribution is tilted with a linearly increasing exponent βD, then the resulting distribution is also Gaussian. Now, in our empirical work, the distribution of D is found to have a heavier than Gaussian right-hand tail. Hence, tilting with a linear exponent leads to a broadening of the distribution, hurting mixing. Hence, we use concave exponents of the form $\beta \min(D, D_{\text{thresh}})$ instead of linear exponents.

Another important choice we make is $f_1(D) = 0$ in Eq. (5) i.e. $\pi_1(\cdot) = \mathbb{P}(\cdot)$. This enables us to draw independent identically distributed (iid) samples from the model distribution for the circuit for 1st chain, instead of moving in a non-trivial Markov chain. iid samples improve mixing since it is possible to move to an entirely new region in the state space in each step. Also, new samples at T_1 now only need to be drawn when a $(1, 2)$ swap is suggested, reducing computational effort. This is particularly useful since iid samples require more computational effort than Markov chain based samples (cf. Section IV-B).

Also, we allow 'burn in' (see [8]) i.e. equilibration of our Markov chain by dropping the first 10% of our samples.

IV. Empirical Results

A. Main results

We tested our approach on two circuits, the 'maccontrol' circuit from the Opencores Ethernet MAC design, and the 'fbfly' circuit which is a router design developed in the Concurrent VLSI Architecture Group at Stanford University [9].

Each of our circuits was synthesized using only three kinds of gates – NAND, NOR and INV from a 90 nm commercial standard cell library (see Section V for a description of less favorable performance seen in a different setting). Synthesis was carried out using the Synopsys Design Compiler tool.

The sizes of the synthesized circuits were:

Circuit	#Gates	# Links	Depth
Maccontrol	1666	2507	18
Fbfly	165602	248586	49

On each circuit, the performance of our algorithm (which we call MCMC-PT) was compared to the performance of simple MC. The core deterministic STA engine used was essentially the same for both algorithms, enabling a fair comparison. No pruning of the circuits was done. The rationale for this is that any pruning should yield the same improvement factor for both simple Monte Carlo (MC) and MCMC-PT, hence our comparative analysis will remain valid. The MCMC-PT algorithm parameters were tuned to enable fast computation of the yield at confidence points over 99.9%.

For each circuit, simple MC was run for 17000 steps (41s for 'maccontrol' and 72 mins for 'fbfly' using one core of an Intel Core 2 Duo E6850 3.00 GHz processor). This leads to an estimation error of nearly 25% at 99.9% yield, for example. Then MCMC-PT was run for *exactly the same CPU time* for each circuit. As explained above, the same core procedures were used for each algorithm. The results obtained are shown in Figure 3.

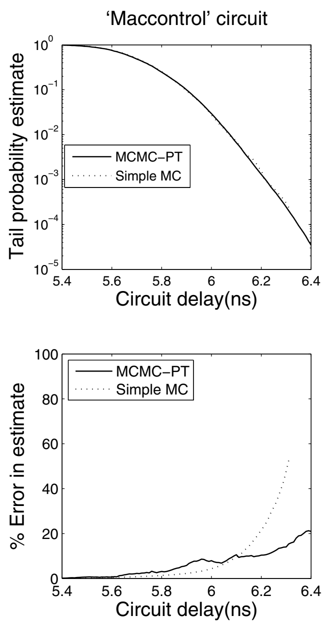


Fig. 3. Delay tail estimates for 'Maccontrol' and 'Fbfly' circuits. Note that % Yield = $100(1 - \text{tail prob.})$.

We see that MCMC-PT has slightly worse error performance as compared to simple MC for yields smaller than 99%, whereas it does much better for yields greater than 99%.

For each circuit, the yield estimate at the 99.9% yield point had an average error of about 10% with MCMC-PT compared to nearly 25% with simple MC. Results at the 99.99% yield point were even more striking, with an average error of $\approx 20\%$ for each circuit with MCMC-PT. No estimate was available from simple MC which has an average of 17000 total samples $\times\, 0.01\% = 1.7$ samples — it would require 15X more samples from simple MC to match the results from MCMC-PT i.e. we obtained a 15X reduction in computation. We call this ratio in computation required the *computation factor*. Conversely, We could sample 1400X more points from the tail ($D > 6.3$ns) for the 'maccontrol' circuit and 210X more points from the tail ($D > 15.6$ns) for the 'fbfly' circuit with MCMC-PT relative to simple MC, in the same computation time. We call this ratio of points sampled the *sampling factor*. The sampling factor represents an upper bound for the computation factor, that would be tight if MCMC-PT drew independent samples.

Most importantly for us, multiple checks demonstrate that our Markov chain is mixing (see Section IV-C). This allows for all the gains quoted above. Further, as explained in Section IV-C, the computation factor results obtained can be generalized to a large set of (yield,accuracy) pairs without multiple simulations.

A note on the gate delay model used — SPICE simulation was used to estimate delay distributions for individual NOR, NAND and INV gates. We assumed 10% normal (Gaussian) variation each in the gate threshold voltage, channel length and width.

It was found that a log-logistic distribution could be made to fit each of the empirical distributions very well, in contrast with other standard distributions like normal, skew normal, log normal, generalized gamma etc. Hence, we fitted log-logistic distributions to each of the three gates and used these fits in our delay model. Note that our method does not depend on the specific distribution used, as long as the inverse cumulative distribution can be computed. Inversion is always possible for any distribution with a table lookup.

B. MCMC-PT implementation

Our algorithm was implemented in the C++ Boost graph library. We used parallel tempering with three chains for the 'maccontrol' circuit, and two chains for the 'fbfly' circuit. The exponents used for the chains were (with D in ns):

Circuit	# chains	$f_1(D)$	$f_2(D)$	$f_3(D)$
Maccontrol	3	0	$9\min(D, 6.1)$	$18\min(D, 6.3)$
Fbfly	2	0	$11\min(D, 15.6)$	-

At each step of parallel tempering, we chose a $(1, 2)$ swap with probability 0.1, a $(2, 3)$ swap with probability 0.1 (if applicable), and a normal move otherwise. $20 - 30\%$ of the swap moves were found to be accepted across each of the transitions for each circuit.

A simple optimization was used in making the Markov chain updates. If a level i was chosen in a proposed move, computation of arrival times at levels $1, \ldots, i$ was avoided, since all arrival times here stay the same. We did nearly 30k steps of MCMC-PT for 'maccontrol' and nearly 50k steps for the 'fbfly' circuit in the time needed for 17k simple MC steps.

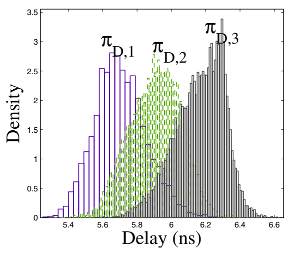
Our post Markov chain analysis used non-parametric unbiased estimators for simplicity and general applicability (independent of distribution). Our estimate of Z was based on a modification of Eq. (2).

C. Analyzing MCMC-PT performance

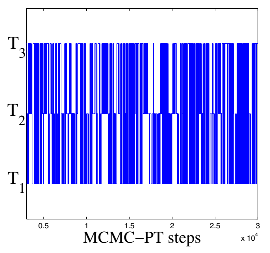
The empirical distributions obtained for the 'maccontrol' circuit can be seen in Figure 4(a). See how the systems at different 'temperatures' explore different delay regions in detail, and yet how adjacent temperature systems have significant overlap in distribution. Multiple runs starting from different seeds, and with different lengths, yielded consistent empirical distributions at each temperature — a signature of fast mixing. Also, it was found that systems made fast transitions across chains at different temperatures, as seen in Figure 4(b). We see that the worst case time gap between swaps is significantly smaller than the timescale over which MCMC-PT is run. This also strongly suggests mixing. Note that this test is stronger than the rule of thumb [10] 'the fraction of accepted swaps should be $\sim 20\%$'.

The error with MCMC-PT is harder to estimate than with simple MC. We ran our MCMC-PT procedure 10 times for each of the circuits, and estimated the statistical error. Note that the expected systematic error is very small, especially when we are convinced of mixing as above. Error estimation is also possible using a 'ground truth' derived from simple Monte Carlo, but would require several days of computation for the larger circuit.

For simple MC, standard analysis tells us that the relative standard deviation of the yield estimate at a point corresponding to failure probability p is $\sqrt{\frac{1-p}{Np}}$, where N is the number of samples. The error is due to the variability of the number of 'failure' samples, we call it *sampling variability*. Due to the tilt in distribution, sampling variability of MCMC increases slower than that of simple MC with increasing yield for a fast-mixing chain. With MCMC, there is an additional source of error — the error in the estimate of the normalizing factor Z. This error scales like $\frac{1}{\sqrt{N}}$. Armed with these facts, simple reasoning leads us to the following — *The computation factor at any particular target yield is nearly independent of desired accuracy. Further, the computation factor increases with target yield.* For example, we are assured gains of at least 15X with MCMC-PT



(a) Empirical distributions



(b) Movement across temperatures

Fig. 4. MCMC-PT empirical behavior for 'maccontrol'

over simple MC at any target yield of 99.99% or greater for the two circuits considered, for any desired accuracy.

V. Discussion and Related work

We begin with a look at related work. We then move to a discussion of our technique, including drawbacks and possibilities for improvement.

Tremendous progress has been made over the last few years in the design of algorithms for SSTA. [1] and [2] excellent recent surveys that provide detailed description and comparison of the important approaches for SSTA, and include exhaustive lists of references. Following [2], the dominant approaches for SSTA can be classified as 'Monte Carlo based' and the probabilistic 'Block-based' and 'Path-based' (see [1] and [2] for descriptions and references). Much progress has been made, particularly on block-based approaches.

The community has been leaning towards block-based approaches for practical SSTA chiefly because of seemingly faster runtime. However, Monte Carlo (MC) approaches offer significant advantages over block-based approaches, as is well documented (eg. [2], [11]). In addition to the commonly known fundamental advantages, we note that MC samples provide direct information about criticality of paths, making MC amenable to finding paths for testing. Also, if model errors are known they can be statistically included in MC sampling thus avoiding pessimism. Finally, MC methods can be parallelized. We note that MCMC is no exception — we can have up to $\sim \frac{N}{T}$ parallel threads, $T \ll N$ being the mixing time (cf. Appendix).

The chief concern with MC is the large number of samples needed. Multiple approaches have been suggested to address this. [11] suggests Quasi-Monte Carlo (QMC) for reduction of sample points needed after dimensionality reduction using Karhunen-Loeve expansion (KLE) models of spatial correlation. A potential problem here is that QMC cannot be used if the problem has large (> 12) 'effective' dimension [12]. Latin hypercube sampling (LHS) is an alternative that can be used for higher dimensional problems but with less variance reduction. [13] suggests using a combination of LHS and QMC along with 'stratification'. Stratification and QMC are used in a few 'critical' dimensions for large variance reduction. Importantly, the authors show how to achieve incrementality. For this work too, gains are dependent on the presence of a small number of 'critical' dimensions. [14] finds that LHS and traditional QMC are inefficient and proposes a QMC sampling technique for low L_2 discrepancy. [15] proposes the use of importance sampling and control variates — it is the only previous Monte Carlo-based approach for SSTA work that uses the large deviation nature of the problem. However, no empirical results are given.

MC-based approaches for other large deviation problems have been suggested. [6] uses importance sampling to reduce the number of samples needed to compute SRAM failure probability (failure being a 'rare event'). The 'norm-minimization' technique uses the fact that most rare events occur due to a small local region in parameter space for this problem. [5] proposes the Statistical Blockade (SB) technique to solve certain rare event problems. SB and the subsequently developed Recursive SB [16] speed up Monte Carlo sampling by building classifiers around bad regions. As such, they are applicable to problems with only a few 'bad' regions in the parameter space. Also, classification techniques are known to perform poorly when the number of 'important' dimensions becomes large [17].

Thus, MCMC proposed in this work is unique in two ways — it allows significant gains over simple MC even in a high dimensional problem with many 'bad' regions, and it is the first rare event sampling technique proposed for SSTA with demonstrated speedup.

A key feature of MCMC as a technique is that it provides immense freedom in designing an appropriate Markov Chain. Nevertheless, it requires careful use of intuition about the system to design a good Markov chain. Also, a Markov chain designed and tested for use in a specific setting, may not be useful in a modified setting [18]. In fact, in case of #P complete problems like the SSTA problem under consideration (see [19]), one can usually find counterexamples for any algorithm, where the algorithm performs poorly. In such a case, we usually attempt to design a different method for the newly encountered situation, for example, a new Markov chain.

Besides the results quoted in Section IV, we also tested our algorithm under different synthesis and modeling conditions, and found less favorable results. We used a more complete set of gate types instead of just 3 allowed gates, along with a better delay model including effects of input pin, load and slew. We found that for several circuits, improvement of MCMC-PT over simple MC was marginal (gains of 1.5X-5X at 99.95% yield), showing our algorithm is sensitive to changes in setup. This is a critical issue that needs resolution.

Another important question that needs to be addressed is whether our techniques can be suitably extended to the correlated variation setting. We believe such an extension should be natural and achieved as follows — the state space $\mathcal{S}_d$ now includes all problem parameters including global, spatially varying, and local (e.g. independent random variation) parameters. A Markov chain is designed to efficiently explore this state space, again focusing on the 'bad' regions. For example, if we add a global parameter to our current model, we could handle it by having two kinds of 'normal' moves – one corresponding to a change in the global parameter, the other corresponding to updating a 'level' (cf. Section III). Correlated variations between underlying parameters can typically be handled by reparametrization (see [8]).

An important practical drawback of our method is that it requires some parameters to be chosen 'by hand' depending on the circuit — the number of parallel tempering chains k and β, D_{thresh} for each chain. Moreover, this selection itself requires some preliminary sampling data and mixing checks. However, we found that $k = 2$ provides good performance over many circuits (including 'maccontrol', though we quote results for $k = 3$). Moreover, it seems likely that an efficient automated procedure can be designed to select β, D_{thresh} values with a small number of samples. Further investigation is needed here.

The coupled issues of deciding how many samples N are needed, and estimating the error in the current yield estimate (without multiple runs) also need to be addressed. Standard techniques (see [8]) might be useful here. Another concern is that our Markov chain may not perform well on some circuits, even with the current setup. Circuits with large depth, for example, may lead to poor performance due to too many levels. Also, mixing time may be large for some circuits.

For example, the mixing behavior is poor for two long disjoint paths. Parallel tempering helps in this context. However, it would be interesting to explore pruning and splitting techniques based on circuit topology to speed up mixing and improve results.

VI. Conclusion

This paper approaches SSTA as a rare event analysis problem, in contrast to existing approaches. We present a new Markov chain Monte Carlo based approach for SSTA when gate delays vary independently. The model studied falls in a 'hard' category with high dimensionality, and a large number of 'bad' regions in the parameter space. A specialized technique called parallel tempering is used along with other modifications to a baseline MCMC algorithm. We demonstrate over an order of magnitude improvement over simple MC and obtain accurate estimates of the right-tail of circuit delay distributions.

It is important to note that algorithm performance was found to be sensitive to changes in synthesis conditions. This needs to be addressed. Other future directions include handling of correlated delay variations including global and spatially varying parameters, incrementality, generation of test patterns using MCMC sampling results, and test set quality (i.e. SPQL) estimation.

Acknowledgments

We would like to thank Hakan Baba and Jie Zhang for help with producing empirical results. Jie Zhang and the anonymous reviewers provided numerous helpful comments and suggestions.

References

[1] C. Forzan and D. Pandini, "Statistical static timing analysis: A survey," *Integr. VLSI J.*, vol. 42, no. 3, pp. 409–435, 2009.

[2] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer, "Statistical timing analysis: From basic principles to state of the art," *IEEE Trans. CAD Integrated Circuits and Systems*, vol. 27, no. 4, 2008.

[3] J. Xiong, V. Zolotov, C. Visweswariah, and P. A. Habitz, "Optimal margin computation for at-speed test," in *DATE '08: Proceedings of the conference on Design, automation and test in Europe*, 2008, pp. 622–627.

[4] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *Proc. ISLPED '07*.

[5] A. Singhee and R. A. Rutenbar, "Statistical blockade: a novel method for very fast monte carlo simulation of rare circuit events, and its application," in *Proc. DATE*, 2007, pp. 1379–1384.

[6] L. Dolecek, M. Qazi, D. Shah, and A. Chandrakasan, "Breaking the simulation barrier: Sram evaluation through norm minimization," in *Proc. ICCAD*, 2008, pp. 322–329.

[7] N. Drego, A. Chandrakasan, and D. Boning, "Lack of spatial correlation in mosfet threshold voltage variation and implications for voltage scaling," *IEEE Trans. Semic. Manuf.*, vol. 22, no. 2, 2009.

[8] W. R. Gilks, S. Richardson, and D. Spiegelhalter, *Markov Chain Monte Carlo in Practice: Interdisciplinary Statistics*. Chapman & Hall/CRC, 1995.

[9] "Stanford cva group," 2009, http://cva.stanford.edu.

[10] A. Kone and D. A. Kofke, "Selection of temperature intervals for parallel-tempering simulations," *J. Chemical Physics*, vol. 122, no. 20, p. 206101, 2005.

[11] A. Singhee, S. Singhal, and R. Rutenbar, "Practical, fast monte carlo statistical static timing analysis: why and how." in *Proc. ICCAD*, 2008, pp. 190–195.

[12] P. Bratley, B. L. Fox, and H. Niederreiter, "Implementation and tests of low-discrepancy sequences," *ACM Trans. Model. Comput. Simul.*, vol. 2, no. 3, pp. 195–213, 1992.

[13] V. Veetil, D. Sylvester, and D. Blaauw, "Efficient monte carlo based incremental statistical timing analysis," in *Proc. DAC*, 2008.

[14] J. Jaffari and M. Anis, "On efficient monte carlo-based statistical static timing analysis of digital circuits," in *Proc. ICCAD*, 2008, pp. 196–203.

[15] S. Tasiran and A. Demir, "Smart monte carlo for yield estimation," in *Proc. Intl. Work. Timing Issues*, 2006.

[16] A. Singhee, J. Wang, B. H. Calhoun, and R. A. Rutenbar, "Recursive statistical blockade: An enhanced technique for rare event simulation with application to sram circuit design," in *Proc. VLSID*, 2008, pp. 131–136.

[17] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*. Springer, July 2003.

[18] E. Mossel, D. Weitz, and N. Wormald, "On the hardness of sampling independent sets beyond the tree threshold," *Prob. Th. and Related Fields*, vol. 143, no. 3-4, 2009.

[19] J. Hagstrom, "Computational complexity of pert problems," *Networks*, vol. 18, pp. 139–147, 1988.

Appendix

Markov chain Monte Carlo (MCMC) methods are often used for sampling from probability distributions ([8] is an excellent reference). We construct a Markov chain that has the desired distribution as its unique equilibrium distribution, usually by ensuring that the 'detailed balance' condition is satisfied, along with aperiodicity and irreducibility (see [8] for definitions). For a Markov chain to be practically useful, it must 'mix' i.e. the distribution should approach the equilibrium distribution in a sufficiently small number of steps. Most effort in designing an algorithm goes into ensuring mixing.

Consider a state space $\mathcal{S}$. The Metropolis-Hastings (MH) algorithm provides a way to construct a Markov chain having an equilibrium distribution $\pi(\cdot)$ given a proposal density $Q(\,\cdot\,;\,\cdot)$. $\pi(\cdot)$ and Q

TABLE II
The Metropolis-Hastings Algorithm

```
MH_π(Q)
1   Choose a starting state $\bar{x}^0 \in \mathcal{S}$ and $N$
2   for t=1 to N
3     Choose proposal $\bar{x}' \sim Q(\,\cdot\,; \bar{x}^{t-1})$
4     Draw $r \sim U[0,1]$
5     if $\left(r < \frac{\pi(\bar{x}')Q(\bar{x}^{t-1};\bar{x}')}{\pi(\bar{x}^{t-1})Q(\bar{x}';\bar{x}^{t-1})}\right)$
6       $\bar{x}^t \leftarrow \bar{x}'$          # accept transition
7     else
8       $\bar{x}^t \leftarrow \bar{x}^{t-1}$          # reject transition
9   end
10  Return $(\bar{x}^0, \ldots, \bar{x}^N)$
```

together determine the crucial mixing time T of the chain. See [8] for a discussion of methods for estimation of T and the number of samples N needed.

Now suppose we are given a distribution $\mathbb{P}(\bar{x}), \bar{x} \in \mathcal{S}$ and we want to analyze a large negative deviation event wrt $\mathbb{P}(\cdot)$ in an 'energy' function $E : \mathcal{S} \to \mathbb{R}$. We proceed as follows:

```
MCMC for Rare Event Analysis
Define 'tilted' target distribution $\pi(\bar{x}) \propto \mathbb{P}(\bar{x})\exp(f(E(\bar{x})))$
Choose suitable proposal density $Q(\bar{x}'; \bar{x}^t)$ for Markov chain transitions
Run MH(Q) to estimate the distribution $\pi_E(\cdot)$ of $E$ under $\pi(\cdot)$
Use $\hat{\pi}_E(\cdot)$ to accurately estimate left tail of $\mathbb{P}_E(\cdot)$
```

We are now in a position to understand MCMC-B stated in Table I. We have chosen our 'energy' variable $E = -D$ for SSTA. Accordingly, $\pi(\cdot)$ is defined in Eq. (1). MCMC-B uses $\mathrm{MH}_\pi(Q)$ to sample from $\pi(\cdot)$ with a very simple proposal density corresponding to steps 4 and 5 of MCMC-B. Also, the acceptance condition in line 8 of MCMC-B is identical to the one in line 5 of $\mathrm{MH}_\pi(Q)$.

Having chosen a value of β, we check if the chain is mixing fast. If not, we reduce β to 'flatten' the energy landscape, reducing the mixing time, and try again.