High-Sigma Monte Carlo for High Yield and Performance Memory Design

Solido Design Automation Inc.

ABSTRACT

Today's mobile devices demand large memories with stringent power and speed constraints. Cutting-edge process nodes are needed to meet target bit densities; but these nodes have extreme statistical process variation, hurting yield on these high-volume chips. To design for yield, one must be able to measure it in the design phase. This in turn requires estimation of bitcell and sense amp yields, for which a simple Monte-Carlo approach would need billions of simulations. High-Sigma Monte Carlo (HSMC) is a commercially-available technology that enables fast, accurate, scalable, and verifiable estimation of such yields. HSMC can be used both to improve feedback within the iterative design loop, as well as for comprehensive verification of high-sigma designs. This paper reviews other proposed methods, then presents the HSMC method, and presents example results for representative high-sigma designs, revealing some of the key traits that make the HSMC technology effective.

1 INTRODUCTION

To enable Mbit or Gbit memory systems, the cells that store the bits are replicated millions or even billions of times. Sense amps, column selectors, and other support circuitry assist in writing to and reading from the bitcells. In a simple architecture, even a single bitcell failure means that the overall memory will fail. Redundant columns, error correction, and the like partially compensate, but at the core the individual bitcell failure rates must be extremely low so that the overall memory does not fail. A part having "low probability of failure" is equivalent to having "high sigma", they are merely different units for yield. For example (two-tailed) *sigma* of 6.0 is a p_{fail} of 1.97e-9, and a *yield* of 99.9999998027%.

High-sigma parts are inherently difficult to design and verify because it is difficult to measure the effects of variation on high-sigma designs quickly and accurately. With only a few defects in a very large number of samples, Monte Carlo (MC) sampling takes prohibitively long to obtain accurate information in the extreme tail of the distribution where the defects occur. Other methods, such as extrapolating from fewer MC samples or importance sampling, have other drawbacks, such as long runtimes, poor accuracy, or that they are only effective for trivial examples and do not scale to the needs of production designs.

The result is that the actual sigma of high-sigma designs is often unknown, so designers add additional margin to compensate for this uncertainty. This in turn sacrifices power, performance, and area. Still, some designs may fail to meet their high-sigma goals, resulting in poor yields and expensive re-spins.

What is required is a tool having the following attributes:

Fast: Runs fast enough to facilitate both iterative design and verification within production timelines.

Accurate: Provides SPICE-accurate information in the extreme tails of the high-sigma distribution, from real Monte Carlo samples.

Scalable: Applicable to production-scale high-sigma designs with hundreds of process variables. Since modern, accurate models of process variation can have 10 or more process variables per device, typical rare-failure event circuits have 50 to 200 process variables.

Verifiable: Results can be understood, pragmatically verified, and trusted.

Usable: The technology must be either easy to implement correctly within the design flow or available in a high-quality, reliable tool.

Solido High-Sigma Monte Carlo (HSMC) is an effective high-sigma design tool that has been designed to meet the above requirements. This paper presents an overview of existing high-sigma techniques, then presents HSMC, including an overview of the technical approach

Solido Design Automation, Inc.
111 North Market Street, Suite 300
San Jose, CA 95113
info@solidodesign.com +1 408 332 5811
http://www.solidodesign.com

and sample results on real production designs.

2 Building Intuition on the Problem

To start to get a feel for the problem, let's simulate 1 million Monte Carlo (MC) samples for a 6 transistor bitcell, measure the read current, and examine the distribution. The bitcell has reasonable device sizings. Some simple yet popular models of local process variation, such as one ΔV_{th} per device, are not accurate [1]. Modern models of statistical process variation are more accurate, having 5, 10, or more local process variables per device. For our examples, we use a 45nm industrial process, with 5 process variables per device. The bitcell has 30 process variables total.

Figure 1 illustrates the distribution of bitcell read current (*cell_i*), in normal quantile (NQ) plot form, where each dot is an MC sample point. NQ plots make it easier to see the tails of a distribution. In a NQ plot, the x-axis is the circuit output and the y-axis is the cumulative distribution function (CDF) scaled exponentially. In a circuit with linear response of output to process variables, the NQ curve will be linear – a straight line of dots from the bottom left to the top right. Nonlinear responses give rise to nonlinear NQ curves.

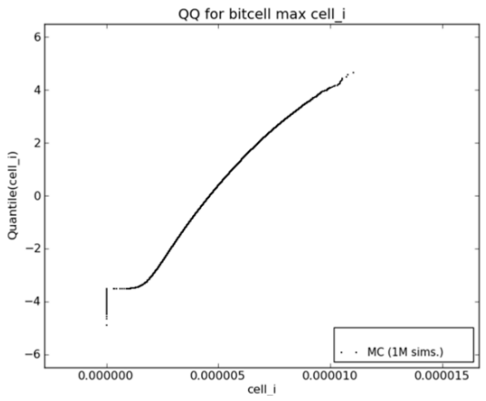

Figure 1: NQ plot for bitcell read current, with 1M MC samples simulated

In the bitcell NQ plot, the bend in the middle of the curve indicates a quadratic response in that region. The sharp dropoff in the bottom left shows that for process points in a certain region, the whole circuit shuts off, for a current of 0. The curve's shape clearly indicates that any method assuming a linear response will be extremely inaccurate, and even a quadratic response will suffer.

Figure 2 shows the NQ plot for delay of a sense amp, having 125 process variables. The three vertical "stripes" of points indicate three distinct sets of values for delay -- a trimodal distribution. The jumps in between the stripes indicate discontinuities: a small step in process variable space sometimes leads to a giant change in performance. Such strong nonlinearities will make linear and quadratic models completely fail; in this case they would completely miss the mode at the far right at delay of about 1.5e-9 s.

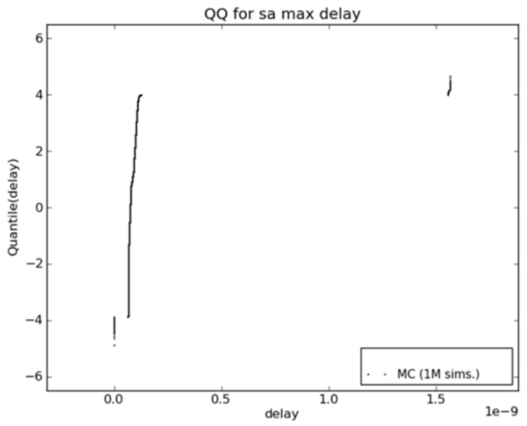

Figure 2: NQ plot for sense amp delay, with 1M MC samples simulated

In this analysis, we have shown the results of simulating 1M MC samples. But that can be very expensive, taking hours or days. And 1M MC samples only covers circuits to about 4 sigma. To find on average a single failure in a 6 sigma circuit, one would need to do about 1 billion MC samples. (Where a failure is either failing a spec, or failing to simulate which also implies failing spec.) Figure 3 illustrates, showing the mapping from process variable space (top) to output space (bottom). Of course, it is not feasible to simulate 1 billion MC samples, unless someone has a spare century!

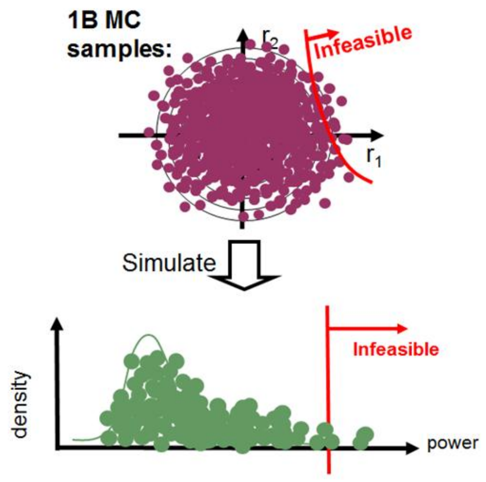

Figure 3: 1 billion simulated MC samples *will* find failures

But of course, taking fewer MC samples mean there will be no failures found, as Figure 4 illustrates.

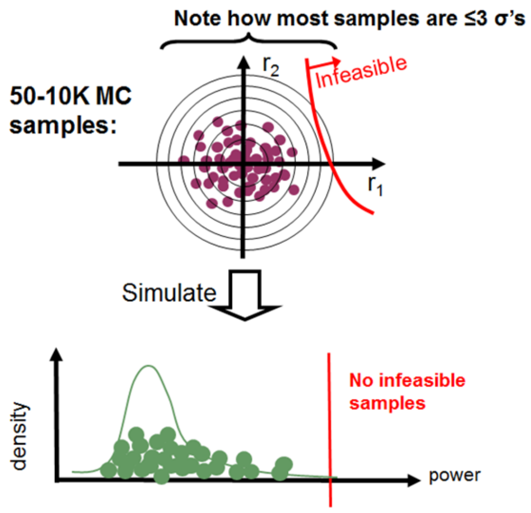

Figure 4: 10K simulated MC samples will not find failures on a high-sigma circuit

3 REVIEW OF HIGH-SIGMA APPROACHES

We aim to design and verify memory circuits with rare failure events. The approaches need to be fast, accurate, scalable, and verifiable to be applicable to production designs. Engineers and researchers have investigated a number of approaches for rare-event verification. This section summarizes some of the popular approaches and highlights challenges with these methods.

Monte Carlo (MC). As described in the previous section, MC would require 100s of millions or billions of samples in order to produce a handful of failures; and simulating fewer samples means that no failures would be found. Positive attributes include its quantifiable accuracy, results that are trustworthy, and scalability that is independent of dimensionality. This latter attribute is a truly remarkable property: MC accuracy α $1/\sqrt{N}$, where N is the number of MC samples; it is not related dimensionality or size of the circuit at all!

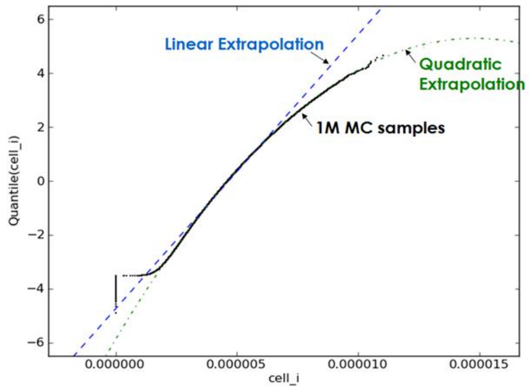

Figure 5: Extrapolation from 1M simulated MC samples on bitcell read current

MC with Extrapolation: This approach runs a large, but feasible number of MC simulations (e.g. 100K or 1M), then extrapolates the results to the region of interest. Extrapolation is typically done using curve fitting or density estimation. The benefits of this approach are that it is simple to understand and implement, and the results are at least trustworthy within the sampling region. Unfortunately, it is time-consuming to run 100K or 1M samples, and extrapolation assumes that the behavior in the extreme tails of the distribution is consistent with that observed at lower sigma. This assumption can be misleading, as there may be drop-offs or discontinuities in the extreme tails; for example, if a device goes out of saturation when a given level of variation is applied. Figure 5 and

Figure 6 show extrapolation on 1M MC samples, for the bitcell and sense amp examples given previously. Clearly, extrapolation fails. The failure of quadratic on the sense amp is almost humorous: the curve starts bending downwards which is mathematically impossible. So much for extrapolation!

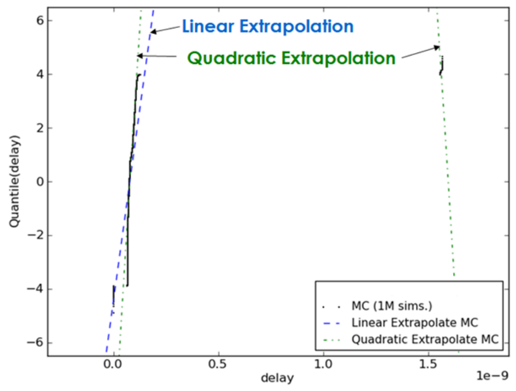

Figure 6: Extrapolation from 1M simulated samples for sense amp delay

Manual Model: In this approach, one manually constructs analytical models relating process variation to performance and yield. However, this is highly time-consuming to construct, is only valid for the specific circuit and process, and may still be inaccurate. A change to the circuit or process renders the model obsolete.

The previous three approaches are traditional industrial approaches. The approaches that follow are more recent.

Quasi Monte Carlo (QMC). Also called "Low Discrepancy Sampling" [1], this variance-reduction technique draws samples from the process distribution with better spread, which in turn reduces the number of samples to get the same accuracy as MC. However, this does not solve the core problem of handling rare failures: for a 1-in-a-billion chance of failure, even a perfect QMC approach will need on average 1 billion MC simulations to get 1 failure.

Direct Model-based. This approach uses models to evaluate a sample's feasibility far faster than simulation. Approach [4] adaptively builds a piecewise-linear model; it starts with a linear regression model, and at each iteration, it chooses a higher-probability process point with known modeling error, simulates, and adds another "fold" to the model. Approach [5] is similar to [4], but uses a classification model rather than a regression model. The general problem of model-based approaches is that the model must be trustworthy, and there is no pragmatic method for verifying the accuracy of a high-sigma model; if the model is inaccurate, then the results will be inaccurate. Even if the model error is just 1%, that 1% error can translate directly to improperly labeling a feasible point as infeasible, or vice-versa. Furthermore, these approaches have only been demonstrated on problems of just 6-12 variables; producing a reliable model for 60-200 variables is far more difficult.

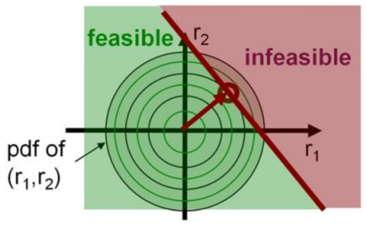

Figure 7: Linear Worst-Case Distance (WCD) approach

Linear Worst-Case Distance (WCD). This approach [6] is a specific instance of the "Direct Model-based" approach, where each output has a linear model mapping process variables to performance (or feasibility). The *n*-dimensional model is constructed from *n+1* simulations: one simulation at nominal, and one perturbation for each of *n* process variables. Figure 7 illustrates the approach's linear model to separate feasible from infeasible regions, in the input process variable space. The biggest (and obvious) limitation is that these models are linear; whereas real-world high-sigma problems including bitcells are often highly nonlinear. As an example, consider the NQ plot in Figure 1. The mapping would be linear only if the samples followed a straight line; but we see that the samples follow a quadratic curve, and in fact drop off on the bottom left when the transistor switches off (a very strong nonlinearity). Or in Figure 2, a linear mapping will not capture the sharp discontinuities between the vertical "stripes". A linearity assumption can lead to estimates of yield that are dangerously optimistic.

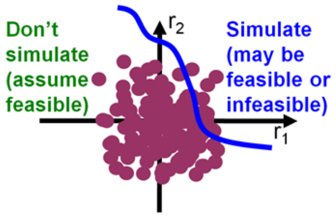
Don't simulate (assume feasible)
Simulate (may be feasible or infeasible)
r2
r1

Figure 8: Rejection Model-Based (Statistical Blockade) Approach

Rejection Model-Based (Statistical Blockade). This approach [3] draws MC samples, but uses a classifier to "block" MC samples that are not in the 97th percentile tails. It simulates the remaining samples, and uses the ones beyond the 99th percentile to estimate yield or construct a tail distribution. The extra 2 percent is a safety margin to account for classifier error, which avoids the need for perfectly accurate models, via the safety margin. It avoids designers' possible distrust of sampling from alternate distributions by drawing directly from the process distribution. Figure 8 illustrates. A problem is that the classifier model could easily have >2% error (the approach has no way to guarantee this), which means it could inadvertently block samples that are in the tail, and there is no effective method to detect this failure condition. Furthermore, this method was demonstrated using problems with just 6-12 variables; not the 60-200 needed for industrial practice, which is far more difficult to do.

Control Variate Model-Based (CV). This variance-reduction technique uses the assistance of a "lightweight model" to reduce the variance of the yield estimate. It combines the model predictions with simulated-value predictions. Because of this, CV models have no minimum-accuracy needs, unlike the above model-based approaches. Rather, CV approaches merely get faster with more accurate models. However, like QMC, control variates do not solve the core problem of handling rare failures.

Markov Chain Monte Carlo (MCMC). The MCMC approach recognizes that we do not need to draw samples directly from the distribution; instead, we can create samples that are infeasible more often, so that decent information is available at the tails. The MCMC approach derives from the famous Metropolis-Hastings algorithm [6]. In MCMC [8], the sampling distribution adaptively tilts towards the rare infeasible events, and then stochastically uses or rejects each subsequent sample in the "chain", based on a threshold. Unfortunately, a stable "well-mixed" chain of MCMC samples is difficult to achieve reliably in practice, especially for non-experts in MCMC (i.e. tool users). Even more troublesome is the arbitrariness of the sampling pdf: in real-world problems with dozens or hundreds of random process variables, it is difficult for users to gain insight into the nature of the sampling distribution, and therefore harder to trust the results or to know when MCMC may have failed.

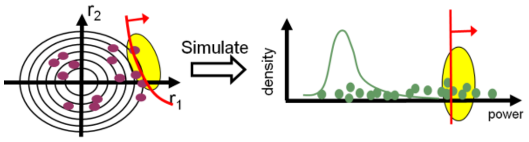

Figure 9: Importance sampling step 1: find region of failure

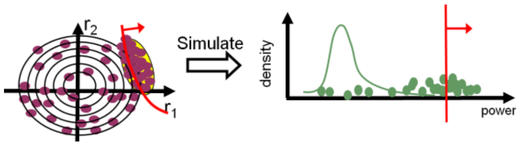

Figure 10: Importance sampling step two: sample in region of failure

Importance Sampling (IS). In this approach [9][10], the general idea is to change the sampling distribution so that more samples are in the region of failure. It is typically composed of two steps, as shown in Figure 9 and Figure 10. The first step finds the new sampling region (yellow oval), which may be via uniform sampling, a linear / WCD approach, or a more general optimization approach. The step typically finds a "center" which is simply a new set of mean values for the sampling distribution. The second step continually draws and simulates samples from the new distribution; it calculates yield by assigning a weight to each sample based on the point's probability density on the original and new sampling distributions.

Archetypical IS approaches for circuit analysis are [11] and [12], which use "centers". In [11], centers are computed by drawing samples are drawn from a uniform distribution in [-6, +6] standard deviations for each process parameter, and keeping the first 30 infeasible samples. The approach [12] finds a center via a spherical sampling technique, followed by solving an optimization formulation (find the process point that minimizes the distance to nominal, subject to being infeasible).

Both [11] and [12] were demonstrated on problems of 6-12 random process variables. But recall that for accurate industrial models of process variation, there are 5, 10, or more process variables per device. This means that even for a 6T bitcell, there are $\geq$30 process variables; and our sense amp problem has 125 process variables.

While IS has strong intuitive appeal, it turns out to have very poor scalability in the number of process variables, causing inaccuracy. Here's why: step one needs to find the most probable points that cause infeasibility; if it is off even by a bit then the average weight of the infeasible samples will be too low, giving estimates of yield that are far too optimistic. For example, in running [11] on a 185-variable flip flop problem, we found that the weights of infeasible samples were < 1e-200, compared to feasible sample weights of 1e-2 to 1e-0. This resulted in an estimated probability of failure of $\approx$1e-200, which is obviously wrong compared to the "golden" probability of failure of 4.4e-4 (found by a big MC run).

To reliably find the most probable points amounts to a global optimization problem, which has exponential complexity in the number of process variables – it can handle 6 or 12 variables (search space of $\approx 10^6$ or 10^{12}), but not e.g. 30 or 125 as in the industrial bitcell and sense amp problems given before (space of 10^{30} or 10^{125}). Finally, IS shares an issue with MCMC: a different sampling distribution hurts the transparency of the technique and makes it harder to trust.

Worst-Case Distance + Importance Sampling. This approach is a variant of importance sampling, where the "centers" are chosen in a style similar to the "Linear "Worst-Case Distances" approach, or local quadratic optimization variants. Its issues are in accuracy and trustworthiness. Specifically, it assumes that the slope at the nominal process point will lead to the most probable region of failure, which can easily be wrong, and leads to overoptimistic estimates of yield. Trustworthiness is hindered in the same fashion as all IS approaches: one has no guidance at all on whether the approach has succeeded or failed.

None of the approaches described here are simultaneously meet the target attributes of speed, accuracy, scalability, and verifiability.

4 HSMC METHOD

4.1 An Idea to Break the Complexity Barrier

While IS sounds promising, its reliability is hindered by the need to solve a global optimization problem of order 10^{30} to 10^{125}.

Perhaps we can then *reframe* the problem and associated complexity, by operating on a *finite set of MC samples*. If we have 1B MC samples, then that is an upper complexity of 10^9. While "just" 10^9 is much better than the 10^{125} complexity of IS, it is still too expensive to simulate 1B MC samples. But what if we were sneaky about which MC samples we actually simulated? Let us use an approach that prioritizes simulations towards the most-likely-to-fail cases. It never does an outright rejection of samples in case they cause failures; it merely de-prioritizes them. It can learn how to prioritize using modern machine learning, adapting based on feedback from SPICE. By never fully rejecting a sample, it is not susceptible to inaccurate models; model inaccuracy simply adds some noise to convergence, as we shall see later.

These are the central ideas behind the High-Sigma Monte Carlo (HSMC) approach.

4.2 HSMC Overview

Solido High-Sigma Monte Carlo (HSMC) is a fast, accurate, scalable, and verifiable tool for verifying high-sigma memory designs.

HSMC works by generating a large number of MC samples, ordering the samples, then running the worst-case samples until all failures are found or until the extreme tails of the distribution are well established. This both generates a SPICE-accurate view of the extreme tail and enables an accurate prediction of the sigma value for the design.

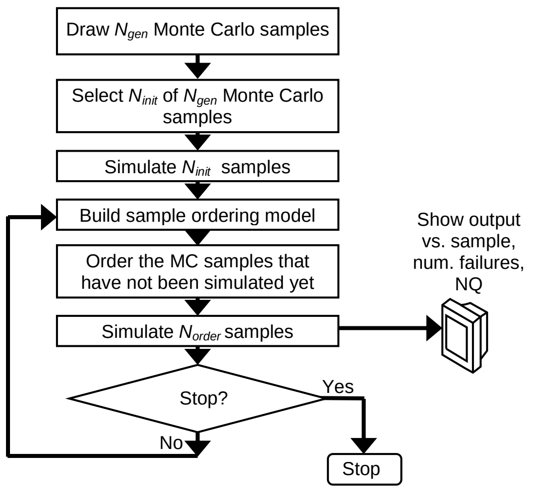
Draw N_{gen} Monte Carlo samples
Select N_{init} of N_{gen} Monte Carlo samples
Simulate N_{init} samples
Build sample ordering model
Order the MC samples that have not been simulated yet
Simulate N_{order} samples
Show output vs. sample, num. failures, NQ
Stop?
Yes
No
Stop

Figure 11: High-level HSMC algorithm

Figure 11 shows the high-level HSMC algorithm, which is summarized as follows. The engine inputs N_{gen}, the number of samples to generate. The algorithm draws N_{gen} samples from the process distribution (but does not simulate them yet). From these samples, it selects a subset of N_{init} samples and SPICE-simulates them. The algorithm constructs a model having the N_{init} points as training inputs, and the corresponding N_{init} performance values as training outputs. The candidate MC samples are from the N_{gen} MC samples, the ones not simulated yet. The algorithm simulates each point on the model to get predicted output values, then orders in ascending (or descending) order of output value. The algorithm then starts to simulate the samples in that order. Periodically, the algorithm will update the model with training data, and re-order the remaining candidate samples. The algorithm either stops manually when the user hits the "stop" button; or automatically when a stop criterion is hit, such as detected all failures found, or ran 5000 simulations.

A more detailed explanation of each of the steps:

Draw samples: HSMC generates a large number of MC samples, enough to estimate yield to the target sigma level. The number of samples to generate is determined using a look-up table based on the target sigma value for the design.

Build sample ordering model: From the MC samples generated, a subset is selected for use in building an ordering model. The subsampling method selects samples from the large number of generated MC samples, emphasizing samples are farthest from nominal. This subsampling method implicitly searches for directions of failures, with uniform bias to different directions. The subset of samples is then simulated using SPICE. A regression model is then created using the patent-pending technology FFX [13], which leverages advances in machine learning to handle arbitrary nonlinearities and high dimensionality. A separate ordering model is produced for each circuit specification.

Order the samples: Using the ordering model generated in the previous step, the full set of samples generated in the first step is ordered, from the worst case to the best case. This is done for each specification. At this point, the predicted order of the samples, from worst to best, is known for each specification. Figure 12 illustrates.

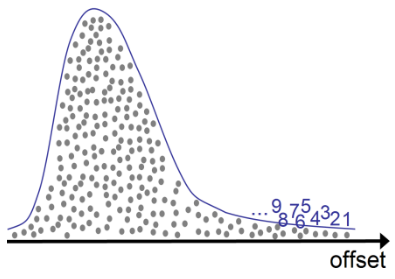

Figure 12: HSMC starts by ordering the samples by predicted output value: 1, 2, 3, … .

Simulate all failure cases: The samples are then simulated using SPICE in predicted order, starting from the worst case. For each specification, the worst sample is run, then the second worst, and so on, until all failure cases are detected. The SPICE simulator provides the real value for each sample, and so the real order of samples being run is also known. Differences between the predicted order and the actual order are used to calculate the error in ordering. Using this error, it is possible to calculate with 95% statistical confidence when all failures to meet specification are found. The simulations stop when all failures are found. If the error in the predicted order is too great, the algorithm rebuilds the model with the new samples added, re-orders the remaining samples, and runs additional simulations. This iterative step can help to correct problems with the ordering model. Figure 13 illustrates this step.

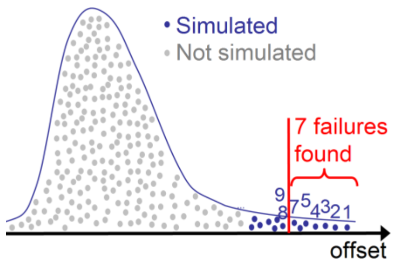

Figure 13: HSMC simulates samples in worst-case first order 1, 2, 3, …, until all failures are found.

Predict sigma and yield: Assuming that all failures are found, sigma and yield can be predicted accurately. This is done by assuming that all samples that were not simulated meet specification. A 95% confidence interval on the yield or sigma value can also be generated based on the number of samples generated using Wilson's confidence score for a binomial distribution [14].

This method then produces, typically in hundreds or a few thousand simulations:

- An accurate view of the extreme tail of the output distributions (e.g. in NQ form), using real MC samples and SPICE-accurate results. Since this gives a tradeoff between yield and spec, one may get accurate yield estimate for a given spec; or spec estimates for a given target sigma (yield).
- A set of MC and SPICE-accurate worst-case high-sigma corners, which can be subsequently for rapid design iterations.

4.3 Sizing Circuits with the Help of HSMC

The idea of "corners" has been around for a long time, and used broadly in many classes of circuit design including memory. Typically one thinks of corners as PVT corners: a modelset value such as FF or SS, and environmental conditions like voltage and temperature. The idea behind corners is sound: find some representative points that bound the distribution of performance, and design against those. This enables fast design iterations without having to do a full statistical analysis at each candidate design. The problem, of course, is that while FF/SS corners bracket digital device performance fairly well, which propagates to digital circuit performances of speed and power fairly well (at least traditionally). But FF/SS corners do not do a good job of bracketing the distributions of memory performances.

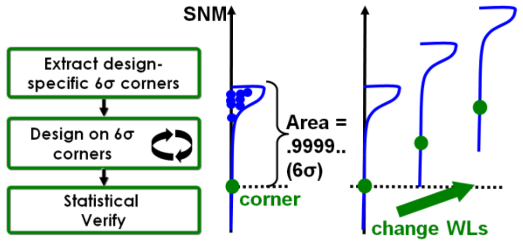

Figure 14: A six-sigma sizing flow

Let's take the idea of "corners" – bracketing performance for rapid design iterations – and make it more general than PVT corners in order to *accurately* bracket memory performance distributions. Figure 14 illustrates the flow.

The first step is to *extract* 6-sigma corners. This is done by simply running HSMC, opening the resulting NQ plot, selecting the point at the 6-sigma mark, and saving it as a corner.

In the next step, bitcell / SA designs with different candidate sizings are tried, using whatever methodology the designer prefers. For each candidate design, one only needs to simulate on the corner(s) extracted in the first step. The output performances are "at 6 sigma yield". The distribution of the output (SNM in the figure) is *implicitly* improved. This allows exploration of tradeoffs among different performances, at 6-sigma yield, but only having to do a handful of simulations (one per corner).

In the final step, one verifies the yield by doing another run of HSMC. The flow concludes if there was not significant interaction between process variables and outputs. Sometimes there is significant interaction, in which case a re-loop is done: grabbing a new corner, designing against it, and verifying. Typically only one re-loop at most is needed, because the design changes in the re-loop are smaller.

4.4 Global vs. Local Variation

Here, we discuss how global (die-to-die, wafer-to-wafer) reconciles with local (within-die) statistical process variation in an HSMC context. We consider five different approaches.

Nested MC. This is given in Table 4.1. An outer loop simulates different wafers or dies being manufactured, and an inner loop simulates the variation within the die. The approach is simple, and handles all nonlinear responses and interactions among global and local. But of course, it is far too slow, needing billions of simulations.

Table 4.1 Nested MC for global + local

```
For each of 100-1000 global MC samples
    Draw a global process point
    Run MC: For each of 1M-1B local MC samples, draw a local process point and simulate netlist at {global, local}.
```

No global + Local HSMC. The idea here is to simply ignore global variation, by setting its variables to nominal; then to run HSMC with local variation. It is simple, fast, convenient, and actually has many good use cases. But of course it (obviously) ignores global variation.

Global FF/SS Corner + Local HSMC. The idea here is to set global variation to a digital modelset value such as FF or SS. This is also simple, fast, and convenient, but is not a fully accurate reflection of the effect of global variation.

Global 3-Sigma Performance Corner + Local HSMC. First, a 3-sigma corner is extracted on global variation, e.g. using Solido Run Monte Carlo+ tool, which gives a process point with output value at the 3-sigma percentile in performance space. Then, HSMC is run where the global variation is set to the values of the 3-sigma corner.

This is simple, fast, and convenient, and a much better reflection of the effect of global variation. It handles nonlinear responses to global variation and to local variation, and interactions between the global process *point* and local variations. Its relatively minor drawback is that it assumes that local variation does not affect the choice of global process corner. This is a safe assumption for getting a {global + local} process for rapid design iterations, but may not be as safe for a final verification step.

Nested HSMC. This approach, given in Table 4.2, is just like nested MC, except the inner MC loop is replaced by HSMC. The approach is simple, and handles all nonlinear responses and interactions among global and local. Its drawback is that it takes about 100x more simulations than a single HSMC run. However, given that typical HSMC runs are about 1000 simulations, then 100K simulations are often justifiable for a final detailed verification step.

Table 4.2: Nested HSMC for global + local

```
For each of 100-1000 global MC samples
    Draw a global process point
    Run HSMC across local, using global process point just drawn.
```

The previous section described how HSMC is used in the context of a rapid-iteration design flow, via corner extraction. This flow reconciles with global + local variation, as Figure 15 illustrates. The first step, of corner extraction, uses the "Global 3-Sigma Corner" approach. The final verification step uses "Global 3-Sigma Corner" too, or "Nested HSMC", depending on the user's time constraints versus design aggressiveness.

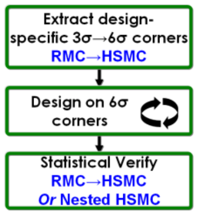

Figure 15: A six-sigma sizing flow, that reconciles global variation

Solido Design Automation, Inc.
111 North Market Street, Suite 300
San Jose, CA 95113
info@solidodesign.com +1 408 332 5811
http://www.solidodesign.com

5 HSMC: Illustrative Results

This section demonstrates HSMC's behavior on a suite of designs. The purpose of this section is to show how HSMC works in practice on actual designs, and purposely includes both cases where HSMC works very effectively and cases where HSMC is less effective.

We demonstrate HSMC on five different high-sigma problems: three circuits, one with a single output and two which have two outputs each. We test on a bitcell and a sense amp, which are representative memory circuits. We also test on a flip flop, which is a representative digital standard cell. The circuits have reasonable device sizings. The device models used are from a modern industrial 45nm process, having approximately 5-10 local process variables per device. The bitcell has 30 variables, the sense amp has 125 variables, and the flip flop has 180 variables.

The experimental methodology is as follows. For each problem, we drew $N \approx 1$M Monte Carlo samples and simulated them. These form our "golden" results. We set the output specification such that 100 of the N samples fail spec. Then, we ran HSMC on the problem, with $N_{gen} = N$, using the same random seed so that it has exactly the same generated MC samples. HSMC ran for 20K simulations. We repeat the procedure with specs set such that 10 of the N_{gen} samples fail spec. $N = 1.5$M for the bitcell, 1M for the sense amp, and 1M for the flip flop.

5.1 Bitcell Results

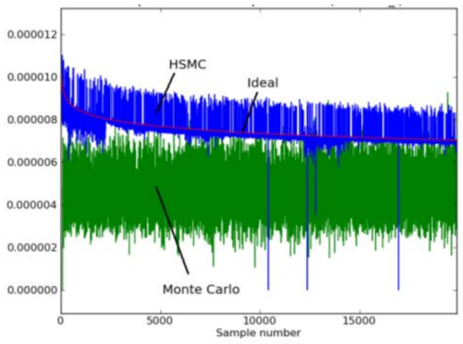

Figure 16: Bitcell cell_i - output vs. sample number (max-first)

Figure 16 shows the bitcell output current "cell_i" versus the sample number. HSMC's aim is to find maximum-valued outputs first, as they are the worst values in this example. We compute the ideal curve by calculating the actual value of all the 1.5M output values, then sorting them and displaying the first 20K sorted values, which is shown by the line marked "Ideal" in Figure 16. By definition, the ideal sorting order monotonically decreases. The MC sampling curve behaves randomly. Having no bias, MC's output values distribute across the whole range, so MC is very slow at finding the worst-case values.

The HSMC curve has a general downward trend starting at the worst-case value, with some noise. The trend shows that HSMC has captured the general relation from process variables to output value. The noise indicates that the HSMC model has some error, which is expected. The lower the modeling error, the lower the noise, and the faster that HSMC finds failures. This highlights HSMC low sensitivity to the accuracy of the model – a key attribute in its effectiveness on industrial-scale problems.

At about 2000 samples, the lower-range values for HSMC jump upwards. This is because HSMC has rebuilt its model using more data, and it has made the ordering more accurate. In a few cases, such as at about 10,500 samples, HSMC predicted that some points would have extreme-maximum values, but when simulated they had extreme minimum values – that is acceptable because HSMC's success is not dependent on getting every sample predicted within an error tolerance. HSMC's success is based on how quickly it finds the worst cases.

The HSMC curve of Figure 16 provides transparency into the behavior of HSMC, to understand how well HSMC is performing in finding failures. The width of the noise area shows how much margin should be given prior to concluding that all failures have been found for a given specification value. The clear trend shows that HSMC is working correctly and is generally outlining the tail of the distribution.

HSMC's effectiveness in finding failures depends on the target specification. A correct setup would typically include fewer than 100 failures to meet specification within the number of samples generated. If there are more failures, then the design is either not meeting its target sigma, or there were too many samples generated for the target sigma. Similarly, if there are no failures to meet specification, then the design either is over-margined or there were not enough samples generated to verify to the target sigma. Therefore, HSMC only needs to be able to find up to a hundred failures to meet specification, allowing a tolerance for significant ordering error while still working within acceptable simulations budgets. In the bitcell case, HSMC finds the first 100 failures within its first 5000 predicted samples (see Figure 17). Note that with 1.5 million samples containing 100 failures, MC will typically not find a single failure within 5000 samples.

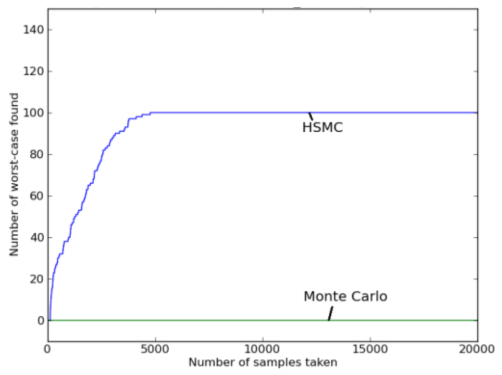

Figure 17: Bitcell cell_i – number of failures found vs. sample number (100 failures exist)

The bitcell results demonstrate one of the key strengths of HSMC, which is its resilience to order prediction error. Since the context only requires finding up to 100 failures to meet specification, the ordering model does not need to be perfectly accurate in order to deliver MC and SPICE accurate results in the extreme tails of a high-sigma distribution within a reasonable number of simulations.

5.2 Sense Amp Results

Figure 18 and Figure 19 show HSMC behavior on the sense amp's power output. Its behavior is similar to those seen in the bitcell, but it finds all 100 failures within its first 1000 samples (see Figure 7). The effect of the ordering model can be seen in Figure 18, as the amount of noise shown in the HSMC curve is clearly lower relative to the sampling region.

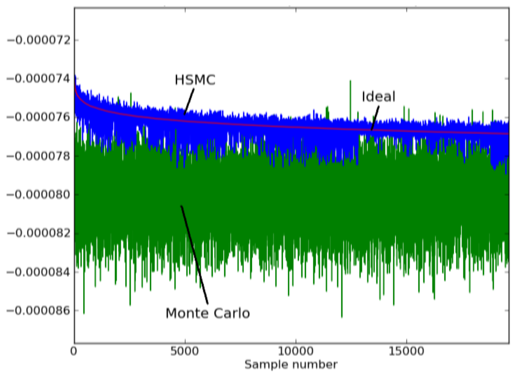

Figure 18: Sense amp power - output vs. sample number (max-first)

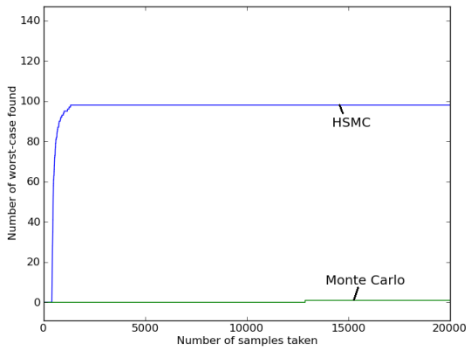

Figure 19: Sense amp power - number of failures vs. sample number (100 failures exist)

The sense amp power example illustrates how HSMC gains efficiency with a better ordering model.

Figure 20 and Figure 21 show HSMC performance on the sense amp's delay output. As we saw in Figure 2, this output has a trimodal distribution. Most sample values being about 0.1e-9 s, and failure cases having a value of about 1.5e-9 s. We set the spec in between; of the 1 million MC samples, there are 61 failing samples (rather than 100). Figure 20 shows that HSMC finds all failures within its first 9000 samples. HSMC's behavior is to find failures with highest frequency in the earlier samples, with decreasing frequency. We can see visually on the output vs. sample plot in Figure 20 that the ordering model is good because the frequency of failures is high at first, then drops off. We can also see that all failures are likely found because there are no new failures found over a large range of samples (i.e. from sample #9000 to #15000). Figure 21 further demonstrates this behavior; we see that HSMC finds all 61 failures within 9K samples, and that it finds most of the failures within the first 1000 samples.

In the case of a trimodal (or bimodal) output distribution, HSMC's behavior is still essentially the same as in the previous cases, though the noise propagates in a different manner. In this case, the upward trend in the output vs. sample number plot is replaced with a view of frequency of failures. Note that the frequency of failures drops off as more samples are run, and that it becomes clear that it is unlikely to find additional failures beyond 10K samples.

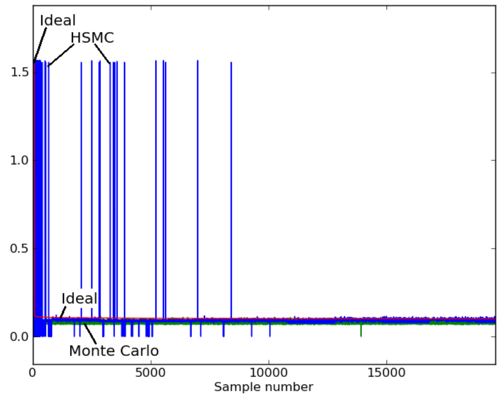

Figure 20: Sense amp delay/1e9 - output vs. sample number (max-first)

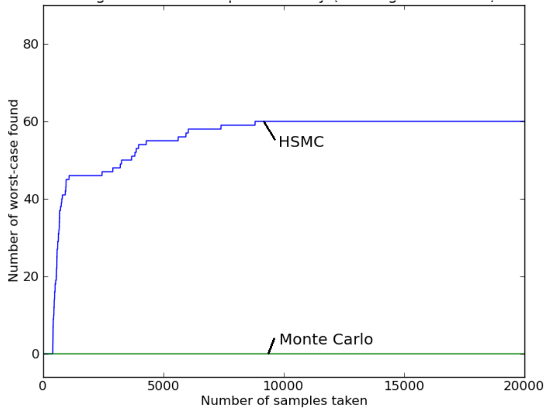

Figure 21: Sense amp delay - number of failures vs. sample number (61 failures exist)

5.3 Flip Flop Results

Figure 22 and Figure 23 show HSMC's behavior on the flip flop's V_h output. We see that HSMC performs near-ideally in output vs sample convergence, and HSMC finds 100/100 failures in less than 500 samples.

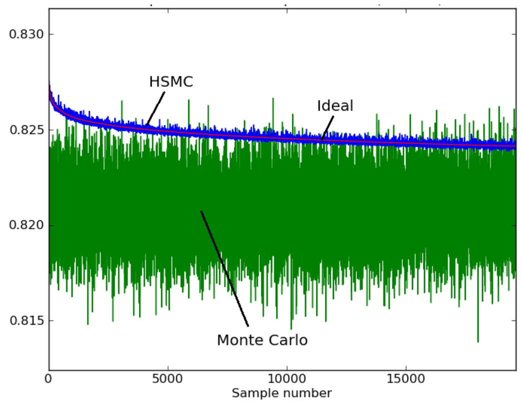

Figure 22: Flip flop V_h - output vs. sample number (max-first)

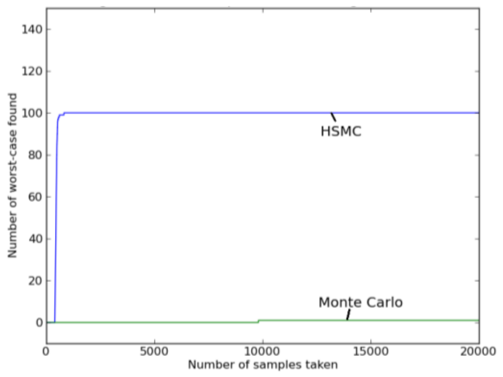

Figure 23: Flip flop V_h - number of failures vs. sample number (100 failures exist)

Note again how visibly tight the amount of noise is relative to the sampling region. Again, the amount of noise is a good indicator of the effectiveness of the sample ordering model.

Figure 24 and Figure 25 show HSMC's behavior on the flip-flop's I_d output. Figure 24 shows that, while the HSMC curve biases towards the extreme maximum, it has a high degree of noise. This means the underlying model is capturing the global trend, but it has significant error in capturing local trends. Despite this significant error, it is still finding failures with reasonable efficiency. Figure 25 shows that after 20K

Solido Design Automation, Inc.
111 North Market Street, Suite 300
San Jose, CA 95113
info@solidodesign.com +1 408 332 5811
http://www.solidodesign.com

simulations, HSMC has found 26/100 failures.

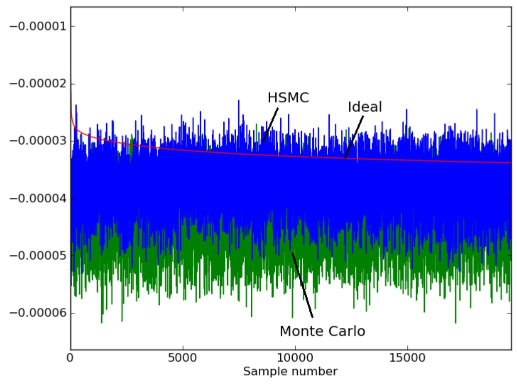

Figure 24: Flip flop I_d - output vs. sample number (max-first)

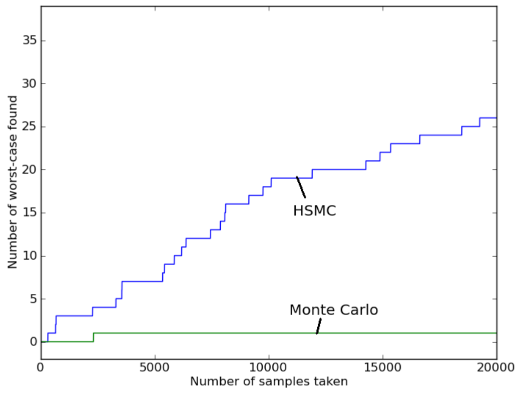

Figure 25: Flip flop I_d - number of failures vs. sample number (100 failures exist)

This example shows how HSMC is self-verifying at runtime, and how even with a very poor ordering model, HSMC can still produce useful results. In this case, the designer would be able to clearly see that HSMC is not producing dependable results within 20K simulations. Given this, the designer could opt to either run additional simulations to gain more resolution, to compliment the HSMC verification with another technique, or to design with some added margin to account for the uncertainty. The designer can also use high-sigma corners discovered here to design against in a subsequent iteration. The key is that HSMC is not misleading due to the inherent quality that it is largely self-verifying.

6 HSMC Vs. Extrapolated MC

As discussed, a common method for verifying high-sigma designs is to take a large number of MC samples, then extrapolate the output distribution it to estimate the extreme tails. The main problems with this method are:

- It takes a long time to run enough samples to have a good chance at extrapolating into the tails.
- Extrapolation quality depends on the extrapolation technique chosen.
- Circuit behaviours can change at higher sigma, and so the assumption that the behavior at low sigma extrapolates gracefully is inherently risky.

This section compares the speed, accuracy, and verifiability of extrapolating 1 million MC samples with HSMC. These experiments use the same bitcell, sense amp, and flip flop circuits examined in the previous section.

All results are presented on a NQ plot to facilitate extrapolation. The plots compare the distributions estimated by 1 million MC samples with the worst 100 from 5500 HSMC simulations on 100 million generated samples. Note that the points appear to form lines due to their density, though they are all in fact individual points representing individual simulations.

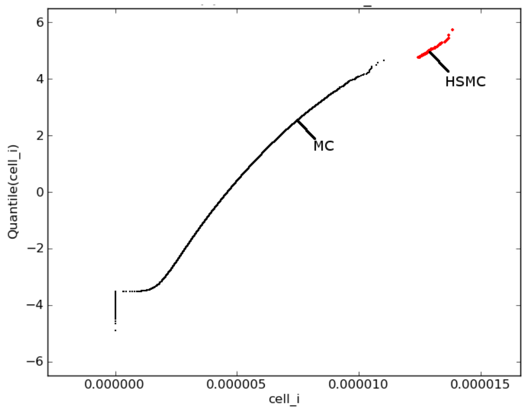

Figure 26: NQ plot for bitcell cell_i: 1M MC samples and 5500/100M HSMC samples

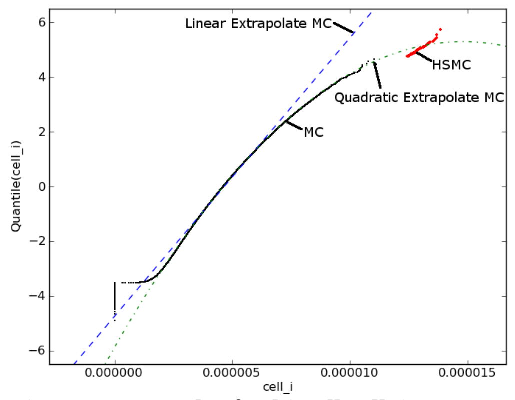

Figure 27: NQ plot for bitcell cell_i: 1M MC samples and 5500/100M HSMC samples, with linear and quadratic extrapolation curves

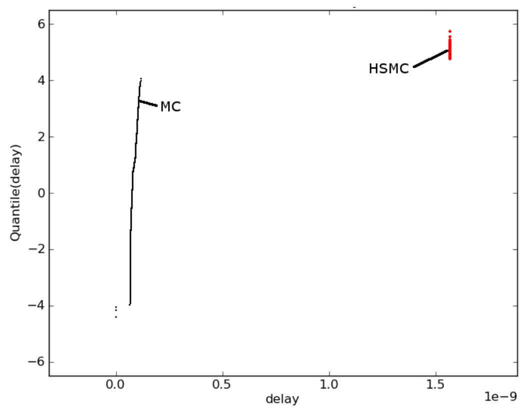

Figure 30: NQ plot for sense amp delay: 100K (not 1M) MC samples and 5500/100M HSMC samples

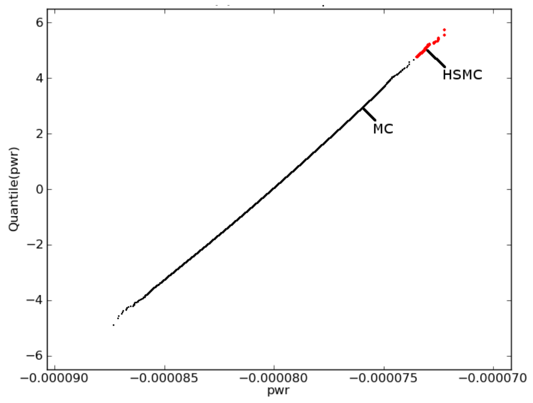

Figure 28: NQ plot for sense amp power: 1M MC samples and 5500/100M HSMC samples

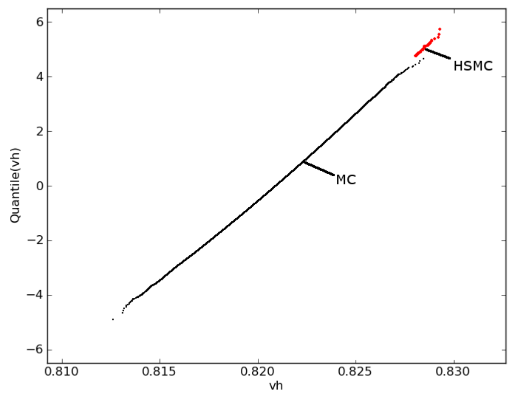

Figure 31: NQ plot for flip flop V_h: 1M MC samples and 5500/100M HSMC samples

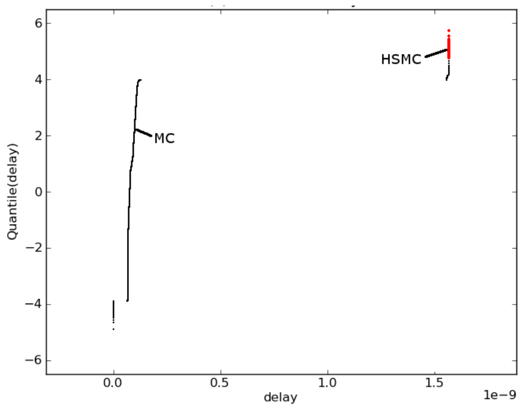

Figure 29: NQ plot for sense amp delay: 1M MC samples and 5500/100M HSMC samples

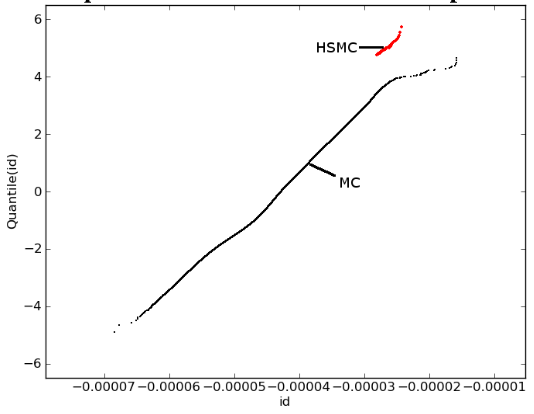

Figure 32: NQ plot for flip flop I_d: 1M MC samples and 5500/100M HSMC samples

Solido Design Automation, Inc.
111 North Market Street, Suite 300
San Jose, CA 95113
info@solidodesign.com +1 408 332 5811
http://www.solidodesign.com

The results shown in Figure 26 to Figure 32 demonstrate three main extrapolation cases:

MC results that extrapolate well: For the cases in Figure 28, Figure 29, and Figure 31, MC extrapolates well and is an effective predictor of the extreme tails of the distribution.

MC results that extrapolate questionably: In the case shown in Figure 26, MC provides an idea of what the extreme tail looks like, although the curve at the end of the MC data does not suggest a definitive extrapolation. In this case, any kind of prediction made will have some amount of error, which will depend on the extrapolation technique used. For example, Figure 27 shows the same data, but with linear and quadratic extrapolation; both extrapolations capture the tail poorly. Similarly, for the case of Figure 32, the MC data does not provide a clear indication of how to extrapolate. Note that the HSMC results in Figure 32 are off: HSMC had missed some failures, and a NQ plot needs all failures to have correct x-axis values. This inaccuracy would be revealed to the user by the amount of noise shown in the output vs. sample number plot (see Figure 24).

MC results that do not estimate the extreme tails: Some MC results simply do not serve as an effective estimator of the extreme tails. For example, consider the bimodality shown in Figure 29. Imagine if, instead of 1 million samples, only 100,000 samples were run and the second mode was not revealed. Figure 30 illustrates this. It would not be possible to even know about the second mode, much less to extrapolate from it. The challenge here is that there is no way to know if this type of case will happen, and similarly no way to know how many MC samples need to be run in order to estimate it. HSMC captures these cases.

In summary, extrapolating MC results is best reserved for cases where HSMC produces very high noise in its output vs. sample plot, such as in the flip flop I_d case. Otherwise, HSMC is faster because it requires fewer simulations, more accurate because it produces results in the extreme tails, and more verifiable because it self-verifies its model of the extreme tails at runtime.

6.1 HSMC Convergence / Limitations

HSMC's effectiveness and accuracy are primarily limited by its ability to generate an accurate ordering model. This section first discusses the implications of the accuracy of the ordering model on the quality of results, as well as other secondary limitations.

In the theoretical case where HSMC completely fails to order samples, HSMC degrades to MC speed and accuracy. In this case, where there is zero correlation between HSMC's ordered samples and the actual sample order, HSMC's samples are plain, unordered, random MC samples by definition. In this case, HSMC's error detection indicates that not all failures are found and it will continue running samples until asked to stop, producing a set of MC results, which may still be useful, though certainly not what HSMC is designed to deliver.

At the other extreme, where the ordering model is perfect, HSMC finds the actual, precise set of MC failure cases sequentially with SPICE accuracy. In this case, HSMC finds the exact tail of the failure region of the output distributions without wasting any samples, aside from those run to generate the ordering model.

The typical case is where HSMC's predicted sample ordering model has some amount of correlation, but it is not perfect. In this case, the order of predicted samples in the tails of the distributions will be imperfect. The tail of the distribution will be found, though not with optimal efficiency due to some error in the order. One essential attribute of HSMC is that, since it is running its predicted samples using SPICE and getting perfect accuracy, the amount of error is discovered at runtime. HSMC then self-calibrates to compensate for the amount of error, running samples until the amount of margin between the specification line and the output values is greater than the amount of error, then stopping. At this point, there is a very high probability that all failures have been found prior to stopping. Therefore, a poor correlation model implies that more simulations will be required to find all of the failures, and not that the accuracy of the result will be poor. Since a good high-sigma design with a well-chosen number of samples will have a small number of failures (e.g. <100), some inaccuracy still allows HSMC to complete verification of high-sigma designs in hundreds to thousands of simulations.

Other notable limitations of HSMC are as follows:

Number of MC samples: Since HSMC is generating and sorting real MC samples, the overhead becomes significant as the number of samples becomes large. At the time of writing, the overhead is insignificant for 100 million or fewer samples. The overhead begins to become a contributor to overall runtime past that. However, HSMC leverages parallel processing not only for simulation, but also for generating and sorting the samples. On modern machine(s) with 10 or more cores total, total runtime even for 5 billion samples remains below 20 minutes.

Number of process variables: For reasons similar to the number of MC samples, the number of process variables is significant because each sample must generate a random number for each process variable, and each process variable is considered when sorting. Memory consumption also becomes a factor as the number of process variables increases. We have found that HSMC works well for hundreds of process variables, and recommend using it on designs with <1000 process variables. We have industrial users pushing HSMC to 10x+ larger problems as well, where they are willing to wait the additional time for simulation and for management.

Not finding all failures: In some cases, HSMC may detect that it has found all failure cases, but there is still the possibility that additional failure cases are within those samples; it cannot be known for sure whether this has occurred. Fortunately, HSMC provides transparency: missed failures are more likely when the ordering model is poor, as shown by the noise in the output value vs. sample curve. For example, Figure 22 is a good curve, and Figure 24 is a poor curve. Even in a case where HSMC finds only half of the true failures, it nonetheless provides a reasonable indication of the distribution tail's behavior. Yield prediction is reasonable; for example, reporting 20 failures per billion versus the actual 40 per billion translates to a sigma difference of just 0.1 or so, which is still useful and sufficiently accurate for most practical purposes. In addition, HSMC has a feature that identifies "non-conforming points": sampled process points that are out of sync with the ordering model's prediction.

In summary, HSMC's limitations are noteworthy, but do not preclude its use with production designs.

7 HSMC: Discussion

This paper presented HSMC's method, as well as demonstrated its behavior through several real circuit examples. We now examine HSMC in terms of set of qualities required for a high-sigma verification technology outlined in the introduction.

Fast: HSMC typically consumes <5000 simulations, and often <1000. These simulations parallelize on a cluster nearly as well as regular MC samples. Runtime is typically 5-20 minutes, depending how many cores are used. To speed up the design loop further, high-sigma corners found using HSMC can be used to design against iteratively. These high-sigma corners provide accurate information about the behavior of the evolving design at the extreme tails with just a few simulations, which in turn reduces design iterations and removes the need to over-margin.

Accurate: HSMC works by leveraging the same trusted technologies used for lower-sigma verification, MC sampling and SPICE simulation. By generating a large number of real MC samples, then simulating only the samples out at the extreme tails, HSMC produces MC and SPICE accurate information at the high-sigma region of interest in the distribution. By revealing any inaccuracy at runtime by comparing the predicted order with the actual order, HSMC's predictions are reliable because it is transparent when there is inaccuracy in the sorting order.

Scalable: HSMC works for tens, hundreds and even thousands of variables, which is large enough to work with production memory designs such as bit cells, sense amps; and non-memory designs including digital standard cells and large custom digital blocks. HSMC can verify out to true 6 sigma with MC and SPICE accuracy, which is a reasonable practical limit for high-sigma designs.

Verifiable: Since HSMC's samples are all MC samples from the true output distribution, the technology can be verified comprehensively against MC samples if the same random seed is used to generate the samples. This is useful for verifying the technology against smaller numbers of samples. HSMC also self-verifies at runtime by comparing its predicted sample order against the actual sample order, as verified using SPICE. By knowing the exact amount of error in the sample order prediction, and the implications of the error in terms of output value variance, HSMC can automatically tune itself to account for the error at runtime, and in doing so, it can run enough samples to find all failure cases. When HSMC fails to order samples effectively, it reveals the error, preventing misconceptions about the accuracy of the approach.

Usable: HSMC is implemented as a high-quality tool for the Solido Variation Designer platform. It requires <1 minute to set up, and automatically post-processes all data at runtime to progressively show yield, sigma, and confidence intervals. The app also can determine at runtime when all failures are found with 95% statistical confidence. The output value vs. sample number plots used in this paper to reveal accuracy are also auto-generated at runtime. High-sigma corners found using HSMC can be saved and designed to using other Variation Designer apps.

In summary, to our knowledge, HSMC is the only method for verifying high-sigma designs that is simultaneously fast, accurate, scalable, and verifiable. Its availability as a high-quality product makes this effective technique easy to adopt and apply in an industrial design environment.

8 CONCLUSION

Semiconductor profitability hinges on high yield, competitive design performance, and rapid time to market. For the designer, this translates to the need to manage diverse variations (global and local process variations, environmental variations, etc.), and to reconcile yield with performance (power, speed, area, etc.), while under intense time pressures.

With high-sigma designs, where failures are one in a million or a billion, previous approaches to identifying failures and verifying those designs were either extremely expensive, inaccurate, or not trustworthy.

The HSMC tool wraps a fast, accurate, scalable, and verifiable high-sigma verification technique into an easy-to-use, reliable, and supported commercial tool. Using the HSMC tool enables rapid high-sigma design by enabling high-sigma feedback within the design loop and by making high-sigma corners available to design against. The HSMC tool also serves as an excellent mechanism for verifying high-sigma designs, reducing verification time and improving accuracy over conventional methods. This in turn promotes the reliable development of more competitive and more profitable products that rely on high-sigma components.

8.1 About Solido Variation Designer

Solido Variation Designer is a comprehensive set of tools for variation-aware design. It allows users to handle memory, analog, custom digital under PVT or statistical variation, following a unified corner-based design methodology. For each problem type, Variation Designer offers easy-to-use tools to quickly verify the design, extract corners, and design against those corners.

References

[1] P. G. Drennan, C. C. McAndrew, "Understanding MOSFET Mismatch for Analog Design," IEEE J. Solid State Circuits, 38(3), March 2003, pp. 450-456

[2] H. Niederreiter. Random Number Generation and Quasi-Monte Carlo Methods. Society for Industrial and Applied Mathematics, 1992

[3] A. Singhee, R. A. Rutenbar, "Statistical Blockade: very fast statistical simulation and modeling of rare circuit events, and its application to memory design", IEEE Transactions on CAD, 28(8), Aug 2009, pp 1176-1189

[4] J. Wang, S. Yaldiz, X. Li and L. Pileggi, "SRAM Parametric Failure Analysis," Proc. ACM/IEEE Design Automation Conference, June 2009

[5] C. Gu and J. Roychowdhury, "An efficient, fully nonlinear, variability-aware non-Monte-Carlo yield estimation procedure with applications to SRAM cells and ring oscillators," Proc.2008 Asia

and South Pacific Design Automation Conference, 2008, pp. 754-761

[6] F. Schenkel et al, “Mismatch Analysis and Direct Yield Optimization by Spec-Wise Linearization and Feasibility-Guided Search,” Proc. Design Automation Conference, 2001, pp. 858-863

[7] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, “Equations of State Calculations by Fast Computing Machines,” Journal of Chemical Physics 21 (6), 1953, pp. 1087–1092

[8] Y. Kanoria, S. Mitra and A. Montanari, “Statistical Static Timing Analysis using Markov Chain Monte Carlo”, Proc. Design Automation and Test Europe, March 2010

[9] T.C. Hesterberg, Advances in importance sampling. Ph.D. Dissertation, Statistics Dept., Stanford University, 1988

[10] D.E. Hocevar, M.R. Lightner, and T.N. Trick, "A Study of Variance Reduction Techniques for Estimating Circuit Yields", IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems 2 (3), July 1983, pp. 180-192

[11] R. Kanj, R.V. Joshi, S.R. Nassif, “Mixture importance sampling and its application to the analysis of SRAM designs in the presence of rare failure events,” Proc. Design Automation Conference 2006, pp. 69-72

[12] M. Qazi, M. Tikekar, L. Dolecek, D. Shah, and A. Chandrakasan, “Loop Flattening & Spherical Sampling: Highly Efficient Model Reduction Techniques for SRAM Yield Analysis,” Proc. Design Automation and Test in Europe, March 2010

[13] T. McConaghy, “High-dimensional statistical modeling and analysis of custom integrated circuits,” Proc. Custom Integrated Circuits Conference (CICC), Sept. 2011. (Invited paper)

[14] E.B. Wilson, “Probable inference, the law of succession, and statistical inference,” Journ. American Statist. Assoc. 22, 1927, pp. 209-212

Solido Design Automation, Inc.
111 North Market Street, Suite 300
San Jose, CA 95113
info@solidodesign.com +1 408 332 5811
http://www.solidodesign.com