Stochastic Testing Method for Transistor-Level Uncertainty Quantification Based on Generalized Polynomial Chaos

Zheng Zhang, Tarek A. El-Moselhy, Ibrahim (Abe) M. Elfadel, and Luca Daniel

***Abstract*—Uncertainties have become a major concern in integrated circuit design. In order to avoid the huge number of repeated simulations in conventional Monte Carlo flows, this paper presents an intrusive spectral simulator for statistical circuit analysis. Our simulator employs the recently developed generalized polynomial chaos expansion to perform uncertainty quantification of nonlinear transistor circuits with both Gaussian and non-Gaussian random parameters. We modify the nonintrusive stochastic collocation (SC) method and develop an intrusive variant called stochastic testing (ST) method. Compared with the popular intrusive stochastic Galerkin (SG) method, the coupled deterministic equations resulting from our proposed ST method can be solved in a decoupled manner at each time point. At the same time, ST requires fewer samples and allows more flexible time step size controls than directly using a nonintrusive SC solver. These two properties make ST more efficient than SG and than existing SC methods, and more suitable for time-domain circuit simulation. Simulation results of several digital, analog and RF circuits are reported. Since our algorithm is based on generic mathematical models, the proposed ST algorithm can be applied to many other engineering problems.**

***Index Terms*—Uncertainty quantification, stochastic circuit simulation, generalized polynomial chaos, stochastic testing method, variation analysis.**

I. Introduction

VARIATION has become a major concern in today's nanometer integrated circuit design [1]. It is well known that the uncertainties of transistor threshold voltages have significantly limited the scaling down of the supply voltage in low-power design [2], [3]. Meanwhile, manufacturing uncertainties can remarkably influence the performance of on-chip interconnects [4]–[11], leading to timing variations [12], [13]. These device-level uncertainties can propagate to the circuit or system level, and finally influence chip performance and yield [14]. Therefore, new electronic design automation (EDA) tools are highly desirable to model and simulate the uncertainties at different levels [4]–[9], [15]–[19].

One bottleneck lies in propagating the effect of uncertainties from the device level to the circuit or system level. Such uncertainty quantification (UQ) problems require specialized stochastic solvers to estimate the underlying statistical information by detailed transistor-level simulation. The mainstream transistor-level simulators such as PSpice [20], Cadence Spectre [21], and Synopsys HSPICE [22] utilize the well-known Monte Carlo (MC) algorithm [23] to perform a statistical characterization. Unfortunately, MC must run repeated transistor-level simulations at a huge number of sampling points due to its slow convergence rate. Although some improvements have been proposed (such as Quasi-Monte Carlo and Latin Hypercube samplings [24]–[26]), MC simulation is still inefficient for many circuit UQ problems.

As an alternative, spectral methods based on polynomial chaos (PC) expansions have been proposed to accelerate the UQ of circuits with Gaussian random parameters [4]–[9], [27], [28]. Spectral methods represent the circuit uncertainties by truncated Hermite-chaos polynomial [29] [which is abbreviated to polynomial chaos (PC)] series expansions, and they compute the PC coefficients by a stochastic Galerkin (SG) [30] or stochastic collocation (SC) [31] approach. The intrusive SG method solves a coupled deterministic equation by modifying an existing deterministic solver to directly compute the PC coefficients. Alternatively, the nonintrusive SC scheme solves a set of decoupled equations at some sampling points by repeatedly calling an existing deterministic solver, followed by a numerical procedure to reconstruct the PC coefficients. Since the truncated PC expansion converges very fast when the solution dependence on the random parameters is smooth, spectral methods have shown remarkable speedup over MC when the number of parameters is small or medium. In the context of EDA, most work has been focused on applying SC and SG to solve the linear stochastic equations arising from interconnect analysis [4]–[9], whereas only a limited number of publications have discussed nonlinear circuits [27], [28]. In [27], SC is combined with PC to simulate RF circuits with Gaussian variations. Later, [28] developed a SPICE-type stochastic simulator for nonlinear circuits. The key idea is to construct some stochastic library models for both linear and nonlinear devices by linearization and Galerkin projection. However, one has to reconstruct these library models for different uncertainty specifications and bias conditions, and thus industrial semiconductor device models cannot be feasibly integrated with this PC-based simulator.

There often exist non-Gaussian variations in practical circuit design, which cannot be easily handled by existing PC-based UQ tools. Due to the development of generalized Polynomial

This work was supported by the Cooperative Agreement Between the Masdar Institute of Science and Technology, Abu Dhabi, UAE and the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, Reference No.196F/002/707/102f/70/9374.

Z. Zhang and L. Daniel are with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (MIT), Cambridge, MA 02139, USA. E-mail: z_zhang@mit.edu, luca@mit.edu.

T. El-Moselhy is with the Department of Aeronautics and Astronautics, MIT, Cambridge, MA 02139, USA. E-mail: tmoselhy@mit.edu

I. M. Elfadel is with Masdar Institute of Science and Technology, Abu Dhabi, United Arab Emirates. E-mail: ielfadel@masdar.ac.ae.

Chaos (gPC) [32]–[34], spectral methods can now be applied to physical models with non-Gaussian variations, and extensive results have been reported [34]–[42]. Unfortunately, there seems to be limited research investigating the application of gPC to EDA problems. In [41], gPC was employed with SC to construct linear stochastic models for electromagnetic devices. Later, gPC-based SC and SG were applied to the UQ of linear circuits with Gaussian and non-Gaussian variations [38]. However, directly applying existing SG or SC methods to circuit problems can be inefficient, as will be discussed in Section IV and demonstrated by the examples in Section V.

Among various SC methods, there exists a special kind of SC scheme [35]–[38][1]. Different from the mainstream SC methods using sparse grids or tensor rules, this SC scheme uses the same number of basis functions and sampling nodes to construct a coupled deterministic equation. The resulting equation can be decoupled *a-priori* with a transformation [35] and then solved by repeatedly calling existing deterministic solvers. Combined with gPC, this nonintrusive method has been used for the UQ of the nonlinear dynamic systems arising from multibody problems [35] and of linear differential algebraic equations (DAEs) from linear circuit analysis [38]. In [38], the tensor product rule is used to construct the basis functions and sampling nodes for SC, leading to some computational overhead.

Our Contribution. In this paper, we propose a gPC-based **intrusive** simulator, called **stochastic testing (ST)**, for the UQ of transistor-level simulation. This work is a variant of the interpolation-based SC [35], [38]. Our work uses a collocation testing method to set up a coupled equation, and decoupling is used to accelerate the computation. However, our ST simulator differs from the previous work in the following aspects:

1) Different from the nonintrusive SC in [35], [38], our proposed method is an intrusive simulator: the resulting coupled equation is solved directly to obtain the spectral coefficients, **without** decoupling *a-priori*. To distinguish our simulator with the intrusive SG and nonintrusive sampling-based SC, we call it "stochastic testing" (ST).
2) ST uses fewer testing nodes than the mainstream SC algorithms [31] (which use sampling nodes from tensor products or sparse grids) and the recent work in [38], leading to remarkable computational speedup. ST provides extra speedup in time-domain simulation, since the intrusive nature of ST allows adaptive time stepping.
3) Decoupling is applied **inside** the intrusive solver. This makes our solver much more efficient over existing intrusive solvers such as SG without sacrificing flexible time stepping controls.

Our algorithm is implemented in a SPICE-type stochastic simulator and integrated with several semiconductor device models for algorithm verification. The proposed method can be applied to many general engineering problems as the mathematical derivation is very generic and does not make any restrictive assumptions in the stochastic DAE's.

Paper Organization. In section II we review MC, the existing spectral methods for stochastic circuit simulation, as well as gPC. Section III presents our intrusive ST simulator and its numerical implementation. In Section IV, gPC-based SG and SC are briefly extended to nonlinear circuits and compared with ST, and we further classify various stochastic simulators. Section V provides some circuit simulation results and discusses the speedup of ST over SC in detail.

[1]The authors would like to thank the anonymous reviewer for pointing out the related work in the mathematical community, specifically Ref. [35], [38].

II. Review: Stochastic Simulators and GPC

Let us consider a general nonlinear circuit with random parameters. Applying modified nodal analysis (MNA) [43], we obtain a stochastic Differential Algebraic Equation (DAE):

$$\frac{d\vec{q}\left(\vec{x}\left(t,\vec{\xi}\right),\vec{\xi}\right)}{dt}+\vec{f}\left(\vec{x}\left(t,\vec{\xi}\right),\vec{\xi}\right)=B\vec{u}\left(t\right) \tag{1}$$

where $\vec{u}(t)\in\mathbb{R}^m$ is the input signal, $\vec{x}\in\mathbb{R}^n$ denotes nodal voltages and branch currents, $\vec{q}\in\mathbb{R}^n$ and $\vec{f}\in\mathbb{R}^n$ represent the charge/flux term and current/voltage term, respectively. Vector $\vec{\xi}=[\xi_1;\xi_2;\ \cdots\ \xi_l]$ denotes l random variables describing the device-level uncertainties assumed mutually independent. In this paper, the port selection matrix B is assumed independent of the random parameters $\vec{\xi}$. We focus on how to solve (1) to extract some statistical information such as mean, variance and probability density function (PDF) of the state vector $\vec{x}\left(t,\vec{\xi}\right)$.

A. Monte Carlo Method

Monte Carlo (MC) is the most widely used UQ tool, and it is implemented in almost all commercial circuit simulators. In MC, N_s samples $\vec{\xi}^1,\cdots,\vec{\xi}^{N_s}$ are first generated according to $\mathrm{PDF}(\vec{\xi})$, the joint Probability Density Function (PDF) of $\vec{\xi}$. Any available deterministic solver is then called to run a simulation at each sample, generating a set of deterministic solutions. Finally, all deterministic solutions are utilized to compute the statistical characterization of interest. The error of MC is proportional to $\frac{1}{\sqrt{N_s}}$. Very often, a huge number (thousands to millions) of samples are required to obtain the desired level of accuracy even when improvements on sampling point selection, such as Mixture Importance Sampling, Quasi-Monte Carlo and Latin Hypercube sampling [24]–[26], are used. The excessive number of samples render the repeated simulation prohibitively expensive in many cases.

B. PC-based SG and SC Methods

In the EDA community, most existing spectral stochastic simulators focus on linear circuits with Gaussian parameters [4]–[9] by considering the following linear stochastic DAE

$$E\left(\vec{\xi}\right)\frac{d\vec{x}\left(t,\vec{\xi}\right)}{dt}+A\left(\vec{\xi}\right)\vec{x}\left(t,\vec{\xi}\right)=Bu\left(t\right). \tag{2}$$

Since $\vec{\xi}$ contains only Gaussian parameters, $\vec{x}\left(t,\vec{\xi}\right)$ can be well approximated by a truncated Hermite expansion

$$\vec{x}\left(t,\vec{\xi}\right)\approx\sum_{k=1}^{K}\hat{x}_k(t)H_k(\vec{\xi}) \tag{3}$$

TABLE I
UNIVARIATE GPC POLYNOMIAL BASIS OF DIFFERENT RANDOM PARAMETERS [34].

Distribution of ξ_k	PDF of component ξ_k $[\rho_k(\xi_k)]$[1]	univariate gPC basis function $\phi_{i_k}(\xi_k)$	Support of ξ_k
Gaussian	$\frac{1}{\sqrt{2\pi}}\exp\left(\frac{-\xi_k^2}{2}\right)$	Hermite-chaos polynomial	$(-\infty,+\infty)$
Gamma	$\frac{\xi_k^{\gamma-1}\exp(-\xi_k)}{\Gamma(\gamma)}$, $\gamma>0$	Laguerre-chaos polynomial	$[0,+\infty)$
Beta	$\frac{{\xi_k}^{\alpha-1}(1-\xi_k)^{\beta-1}}{\mathrm{B}(\alpha,\beta)}$, $\alpha,\beta>0$	Jacobi-chaos polynomial	$[0,1]$
Uniform	$\frac{1}{2}$	Legendre-chaos polynomial	$[-1,1]$

[1] $\Gamma(\gamma)=\int_0^\infty t^{\gamma-1}\exp(-t)\,dt$ and $\mathrm{B}(\alpha,\beta)=\int_0^1 t^{\alpha-1}(1-t)^{\beta-1}\,dt$ are the Gamma and Beta functions, respectively.

where $H_k(\vec{\xi})$ is an orthonormal multivariate Hermite polynomial [30], and $\hat{x}_k(t)$ the PC coefficient. If the total polynomial order is p, then the above Hermite expansion uses

$$K=\left(\begin{array}{c}p+l\\ p\end{array}\right)=\frac{(p+l)!}{p!l!} \tag{4}$$

basis functions in total to approximate $\vec{x}(\vec{\xi},t)$.

In the intrusive SG method [30], the Hermite expansion (3) is first substituted into (2). Applying Galerkin testing, SG sets up a coupled equation of dimension nK. The PC coefficients are then directly computed by solving this coupled equation.

The nonintrusive SC method [31] first selects a set of sampling points according to some rules (such as Gauss-quadrature tensor product rule or sparse grid rule). At each sampling point, (2) is solved as a deterministic equation to get a deterministic solution. After that, a post-processing step such as numerical integration is applied to get the PC coefficients.

C. Generalized Polynomial Chaos (gPC)

Generalized polynomial chaos (gPC) [32]–[34] is a generalization of the original Hermite-type PC [29], and it can handle both Gaussian and non-Gaussian random parameters efficiently. A multivariate gPC basis function $H_{\vec{i}}(\vec{\xi})$ reads

$$H_{\vec{i}}\left(\vec{\xi}\right)=\prod_{k=1}^{l}\phi_{i_k}(\xi_k), \tag{5}$$

where $\phi_{i_k}(\xi_k)$ is a univariate orthonormal polynomial of degree i_k. The specific form of $\phi_{i_k}(\xi_k)$ depends on the density function of ξ_k. Table I lists the correspondence between some typical univariate gPC polynomial basis $\phi_{i_k}(\xi_k)$ and the probability distributions of ξ_k [34].

In the stochastic space Ω, the inner product of any two general functions $y_1\left(\vec{\xi}\right)$ and $y_2\left(\vec{\xi}\right)$ is defined as

$$\left\langle y_1\left(\vec{\xi}\right),y_2\left(\vec{\xi}\right)\right\rangle=\int_\Omega \mathrm{PDF}\left(\vec{\xi}\right)y_1\left(\vec{\xi}\right)y_2\left(\vec{\xi}\right)d\vec{\xi}. \tag{6}$$

The normalized gPC bases have the the orthogonality property

$$\left\langle H_{\vec{i}}\left(\vec{\xi}\right),H_{\vec{j}}\left(\vec{\xi}\right)\right\rangle=\delta_{\vec{i},\vec{j}}.$$

With gPC, one can also approximate a second-order stochastic process $\vec{x}(\vec{\xi},t)$ by an order-p truncated series

$$\vec{x}(t,\vec{\xi})\approx\sum_{|\vec{i}|\le p}\tilde{x}_{\vec{i}}(t)H_{\vec{i}}(\vec{\xi}) \tag{7}$$

which has totally K basis functions [K is given in (4)]. In (7), $\vec{i}=[i_1;i_2;\cdots;i_l]$ is the index vector with $|\vec{i}|=\sum_{k=1}^{l}|i_k|$, integer i_k the highest order of ξ_k in $H_{\vec{i}}(\vec{\xi})$. The mean value and standard deviation of $\vec{x}(\vec{\xi},t)$ are easily calculated as:

$$\begin{aligned}&\mathrm{E}\left(\vec{x}\left(t,\vec{\xi}\right)\right)=\tilde{x}_{\vec{i}}(t),\ |\vec{i}|=0\\ &\sigma\left(\vec{x}\left(t,\vec{\xi}\right)\right)\approx\sqrt{\sum_{|\vec{i}|=1}^{p}|\tilde{x}_{\vec{i}}(t)|^2}.\end{aligned} \tag{8}$$

In PC and gPC, the random parameters are assumed mutually independent. For general cases with arbitrary probability measures, constructing orthogonal basis functions is much more involved. A nice approach is proposed in [44], however, its numerical implementation is not trivial. In this paper, we keep the assumption that all random parameters are mutually independent, and we apply gPC to develop more efficient UQ tools for nonlinear transistor-level circuit analysis.

III. STOCHASTIC TESTING SIMULATOR

Since there is a one-to-one correspondence between $1\le k\le K$ and the index vector $\vec{i}$, we denote

$$\hat{x}(t,\vec{\xi})=\sum_{k=1}^{K}\hat{x}_k(t)H_k(\vec{\xi}) \tag{9}$$

for convenience. Now $H_k(\vec{\xi})$ denotes the k-th multivariate orthonormal gPC basis function of (7). Replacing the exact solution $\vec{x}\left(t,\vec{\xi}\right)$ in stochastic DAE (1) with the above truncated gPC expansion yields a residual function

$$\mathrm{Res}(\vec{X}(t),\vec{\xi})=\frac{d\vec{q}\left(\hat{x}(t,\vec{\xi}),\vec{\xi}\right)}{dt}+\vec{f}\left(\hat{x}(t,\vec{\xi}),\vec{\xi}\right)-B\vec{u}(t). \tag{10}$$

Now the unknown vector reads

$$\vec{X}(t)=[\hat{x}_1(t);\ \cdots;\ \hat{x}_K(t)]\in\mathbb{R}^N,\ \text{with } N=nK. \tag{11}$$

A. Basic Idea of the ST Method

In order to compute $\vec{X}(t)$, ST starts from (10) and sets up a larger-size determined equation by collocation testing. Specifically, ST selects K testing (or collocation) points $\vec{\xi}^1,\cdots,\vec{\xi}^K$, then it enforces the residual function to be zero at each point, leading to the following deterministic DAE:

$$\frac{dQ\left(\vec{X}(t)\right)}{dt}+F\left(\vec{X}(t)\right)=\tilde{B}\vec{u}(t) \tag{12}$$

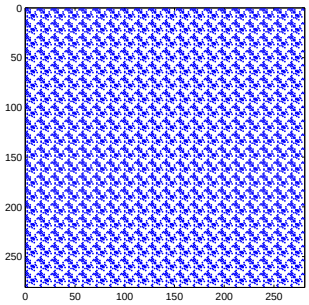


Fig. 1. Structure of the Jacobian matrix in ST-based simulator, with $l = p = 3$ and $K = 20$.

with

$$Q\left(\vec{X}\left(t\right)\right) = \begin{bmatrix} \vec{q}\left(\hat{x}(t,\vec{\xi}^1),\vec{\xi}^1\right) \\ \vdots \\ \vec{q}\left(\hat{x}(t,\vec{\xi}^K),\vec{\xi}^K\right) \end{bmatrix},\ \tilde{B} = \begin{bmatrix} B \\ \vdots \\ B \end{bmatrix}$$
$$F\left(\vec{X}\left(t\right)\right) = \begin{bmatrix} \vec{f}\left(\hat{x}(t,\vec{\xi}^1),\vec{\xi}^1\right) \\ \vdots \\ \vec{f}\left(\hat{x}(t,\vec{\xi}^K t),\vec{\xi}^K\right) \end{bmatrix}. \tag{13}$$

The collocation testing used in ST is the same with that used in collocation-based integral equation solvers [45]. However, in stochastic computation, "stochastic collocation" means a different sampling-based method (c.f. Section IV-B). Therefore, we name our proposed method as "stochastic testing".

There remain two important issues, and how to address them distinguishes our ST solver with the nonintrusive stochastic solvers in [35], [38]. The first issue is how to solve the resulting coupled DAE. ST directly solves (12) by an intrusive solver. As a result, the gPC coefficients can be directly computed and adaptive time stepping [46] can be used. The second issue is how to select the testing nodes. ST selects K testing points from some candidate nodes, whereas $(p+1)^l \gg K$ nodes are used in [38] to make the transformation matrix invertible.

B. Intrusive Decoupled Solver

ST is an intrusive simulator: the coupled DAE is passed into a specialized transient solver to directly compute the gPC coefficients, and matrix structures are exploited inside Newton's iterations to obtain simulation speedup. As a demonstration, we consider backward-Euler integration. Other types of numerical integration schemes (e.g., Trapezoidal or Gear-2 method) are implemented in a similar way inside ST.

Let $\vec{X}_k = \vec{X}\left(t_k\right)$ and $\vec{u}_k = \vec{u}\left(t_k\right)$. In the transient solver, DAE (12) is discretized, leading to an algebraic equation

$$\mathrm{R}(\vec{X}_k) = \alpha_k(Q(\vec{X}_k) - Q(\vec{X}_{k-1})) + F(\vec{X}_k) - \tilde{B}\vec{u}_k = 0$$

with $\alpha_k = \frac{1}{t_k - t_{k-1}}$. The time step size is adaptively selected according to the local truncation error (LTE) [20], [46]. Starting from an initial guess $\vec{X}_k^0$, $\vec{X}_k$ is computed using Newton's iterations

$$\begin{aligned} &\text{solve } \mathcal{J}\left(\vec{X}_k^j\right)\Delta\vec{X}_k^j = -\mathrm{R}\left(\vec{X}_k^j\right), \\ &\text{update } \vec{X}_k^{j+1} = \vec{X}_k^j + \Delta\vec{X}_k^j, \end{aligned} \tag{14}$$

until convergence. Here $\mathcal{J}(\vec{X}_k^j)$ is the Jacobian matrix of $\mathrm{R}(\vec{X}_k^j)$. Fig. 1 shows the structure of $\mathcal{J}(\vec{X}_k^j)$ from a CMOS low-noise amplifier (LNA) with n=14, l=p=3 and K=20. Clearly, all off-diagonal blocks are filled with non-zero sub-matrices. As a result, directly using a matrix solver to compute $\Delta\vec{X}_k^j$ can be inefficient. If a direct matrix solver is employed, the linear system solution costs $O(N^3) = O(K^3n^3)$; when an iterative method is applied, the cost is $\hat{m}O(K^2n)$ where $\hat{m}$ is the number of iterations.

The coupled linear equation in (14) is instead solved in a decoupled manner. We rewrite the Jacobian matrix in (14) as

$$\mathcal{J}(\vec{X}_k^j) = \tilde{\mathcal{J}}(\vec{X}_k^j)M. \tag{15}$$

Matrix $\tilde{\mathcal{J}}(\vec{X}_k^j)$ has a block-diagonal structure:

$$\tilde{\mathcal{J}}(\vec{X}_k^j) = \begin{bmatrix} J(\vec{X}_k^j,\vec{\xi}^1) & & \\ & \ddots & \\ & & J(\vec{X}_k^j,\vec{\xi}^K) \end{bmatrix}. \tag{16}$$

Let $\hat{x}_{n_2}^{k,j}$ denotes the n_2-th gPC coefficient vector in X_k^j, then

$$J(\vec{X}_k^j,\vec{\xi}) = \alpha_k\frac{\partial\vec{q}(\vec{x},\vec{\xi})}{\partial\vec{x}} + \left.\frac{\partial\vec{f}(\vec{x},\vec{\xi})}{\partial\vec{x}}\right|_{\vec{x}=\sum_{n_2=1}^{K}\hat{x}_{n_2}^{k,j}H_{n_2}(\vec{\xi})}. \tag{17}$$

The matrix M is

$$M = \Phi\otimes I_n,\ \Phi = \begin{bmatrix} H_1(\vec{\xi}^1) & \cdots & H_K(\vec{\xi}^1) \\ \vdots & \ddots & \vdots \\ H_1(\vec{\xi}^K) & & H_K(\vec{\xi}^K) \end{bmatrix} \tag{18}$$

where $\otimes$ denotes the Kronecker product operation. The Vandermonde-like matrix $\Phi \in \mathbb{R}^{K\times K}$ only depends on the testing points and basis functions. The inverse of M is

$$M^{-1} = \Phi^{-1}\otimes I_{n\times n} \tag{19}$$

which can be easily computed because: 1) Φ is of small size; and 2) fast inverse algorithms exist for Vandermonde-like matrices [47]. Both Φ and Φ^{-1} are calculated only once and then reused for all time points.

Finally, the linear equation in (14) is solved as follows:

1) Solve $\tilde{\mathcal{J}}(\vec{X}_k^j)\Delta z = -\mathrm{R}(\vec{X}_k^j)$ for Δz. Due to the block-diagonal structure, this step costs only $KO\left(n^3\right)$ for a direct solver or $\hat{m}KO(n)$ for an iterative solver.
2) Calculate the sparse matrix-vector product $\Delta\vec{X}_k^j = M^{-1}\Delta z$. Since the closed form of M^{-1} is ready, the matrix-vector multiplication costs only $O(nK)$.

The computational cost of ST solver now has only a linear dependence on K, as contrasted with the cubic or quadratic dependence when directly solving the coupled linear equation.

The ST solver can be easily implemented inside a commercial circuit simulator. Inside each Newton's iteration, one can convert $\vec{X}_k^j$ to a deterministic state variable and then evaluate the corresponding Jacobian and function values for

a testing node. Repeating this procedure for all nodes, $\tilde{\mathcal{J}}(\vec{X}_k^j)$ and $\mathrm{R}(\vec{X}_k^j)$ can be obtained. After that, all blocks are solved independently to obtain Δz and then $\Delta\vec{X}_k^j$. If the Newton's iterations get converged, the local truncation error (LTE) is checked by an existing estimator [20], [46]. The solution is accepted and ST proceeds to the next time point if the LTE is below a threshold; otherwise, the time step size is reduced and $\vec{X}_k$ is recomputed. Since the function/Jacobian evaluation and linear system solutions are well decoupled, ST can be easily implemented on a parallel computing platform.

C. Testing Node Selection

The testing nodes in ST are selected by two steps. First, $(p+1)^l$ candidate nodes are generated by a Gaussian-quadrature tensor product rule. Next, only K nodes (with $K \ll p+1)^l$) are selected from the candidate nodes and used as the final testing nodes. Note that $(p+1)^l$ sampling nodes are used in [38], which are exactly the candidate nodes of ST.

1) Candidate Node Generation: Let $\xi_k \in \Omega_k$ be a random parameter and $\rho_k(\xi_k)$ the corresponding PDF. Gaussian quadrature can be used to evaluate a 1-D stochastic integral:

$$\int_{\Omega_k} g(\xi_k)\,\rho_k(\xi_k)\,d\xi_k \approx \sum_{j=1}^{\hat{n}} g\left(\xi_k^j\right) w_k^j \tag{20}$$

where ξ_k^j denotes the j-th quadrature point and w_k^j the corresponding weight. The choice of a Gaussian quadrature rule depends on the support Ω_k and the PDF $\rho_k(\xi_k)$.

With the computed 1-D quadrature points and weights for each ξ_k, one can construct multi-dimensional quadrature points to calculate the multivariate stochastic integral

$$\int_{\Omega} g\left(\vec{\xi}\right) \mathrm{PDF}\left(\vec{\xi}\right) d\vec{\xi} \approx \sum_{j=1}^{\hat{N}} g\left(\vec{\xi}_j\right) w^j \tag{21}$$

by a tensor product or sparse grid technique [34], [39]. In this work, we set $\hat{n} = p+1$ [p is highest total polynomial order in (7)] and then use a tensor product rule to construct $\hat{N} = \hat{n}^l$ quadrature nodes in the l-D stochastic space. For convenience, we define an index matrix $\mathcal{I} \in \mathbb{Z}^{l\times\hat{N}}$, the j-th column of which is decided according to the constraint

$$1 + \sum_{k=1}^{l} (\hat{n}-1)^{k-1} (\mathcal{I}(k,j)-1) = j. \tag{22}$$

Then the j-th quadrature node in Ω is

$$\vec{\xi}_j = [\xi_1^{\mathcal{I}(1,j)}, \cdots, \xi_l^{\mathcal{I}(l,j)}], \tag{23}$$

where $1 \le \mathcal{I}(k,j) \le \hat{n}$ indicates the index of the quadrature point in Ω_k. The corresponding weight of $\vec{\xi}_j$ is computed by

$$w^j = \prod_{k=1}^{l} w_k^{\mathcal{I}(k,j)}. \tag{24}$$

Algorithm 1 Testing Node Selection.

1: Construct $\hat{N}$ l-D Gaussian quadrature nodes and weights;
2: $[\vec{w}, \mathrm{ind}]$=sort($\vec{w}$, 'descend'); % reorder the weights
3: $V = \vec{H}\left(\vec{\xi}_k\right)/||\vec{H}\left(\vec{\xi}_k\right)||$, with $k = \mathrm{ind}(1)$;
4: $\vec{\xi}^1 = \vec{\xi}_k$, $m = 1$; % the 1st testing node
5: **for** $j = 2, \cdots, \hat{N}$ **do**
6: $k = \mathrm{ind}(j)$, $\vec{v} = \vec{H}\left(\vec{\xi}_k\right) - V\left(V^T\vec{H}\left(\vec{\xi}_k\right)\right)$;
7: **if** $||\vec{v}||/||\vec{H}\left(\vec{\xi}_k\right)|| > \beta$
8: $V = [V; \vec{v}/||\vec{v}||]$, $m = m+1$;
9: $\vec{\xi}^m = \vec{\xi}_k$; % select as a new testing node.
10: **if** $m \ge K$, break, **end**;
11: **end if**
12: **end for**

2) Selecting Testing Nodes: K testing nodes are selected from the $(p+1)^l$ candidate nodes based on two criteria:

1) We prefer those quadrature nodes that are statistically "important", i.e., those nodes with large weight values;
2) The matrix Φ should be full-rank and well conditioned.

The Matlab pseudo codes of selecting the final testing nodes are provided in Algorithm 1. In Line 7, $\beta > 0$ is a threshold scalar. The input vector in Line 2 is $\vec{w}$=$[|w^1|, |w^2|, \cdots, |w^{\hat{N}}|]$, and the vector-valued function $\vec{H}(\xi) \in \mathbb{R}^{K\times 1}$ is

$$\vec{H}(\vec{\xi}) = [H_1(\vec{\xi}), H_2(\vec{\xi}), \cdots, H_K(\vec{\xi})]^T. \tag{25}$$

The basic idea of Algorithm 1 is as follows. All candidate nodes and their weights are reordered such that $|w^j| \ge |w^{j+1}|$, and the first node is selected as the first testing node $\vec{\xi}^1$. Then, we consider the remaining candidate nodes from the "most important" to the "least important". Assuming that $m-1$ testing nodes have been selected, this defines a vector space

$$V = \mathrm{span}\left\{\vec{H}(\vec{\xi}^1), \cdots, \vec{H}(\vec{\xi}^{m-1})\right\}. \tag{26}$$

The next "most important" candidate $\vec{\xi}_k$ is selected as a new testing node if and only if $\vec{H}(\vec{\xi}_k)$ has a large enough component orthogonal to V. This means that the dimensionality of V can be increased by adding $\vec{\xi}_k$ as a new testing point.

When l is large, generating and saving the candidate nodes and index matrix $\mathcal{I}$ become expensive. A solution is to select the testing nodes without explicitly generating the candidate nodes or $\mathcal{I}$. First, we generate weight w^j's and the corresponding index j's according to (24) and (22), respectively. In the k-th step, we find the k-th largest weight w^j and its corresponding index j. According to (22), the j-th column of the index matrix $\mathcal{I}$ can be calculated, and then we can construct candidate node $\vec{\xi}_j$. Finally $\vec{\xi}_j$ is selected as a new testing node if $\vec{H}(\vec{\xi}_j)$ has a large enough component orthogonal to V, otherwise it is omitted and not stored.

There exist other possible ways to select the testing nodes. A recent progress is to generate the nodes by Leja sequences, a greedy approximation to Fekete nodes [37]. How to select the optimal testing nodes is still an open problem.

IV. Comparison with Other Stochastic Solvers

This section briefly extends the gPC-based SG and SC to nonlinear circuit problems and compares them with our proposed ST algorithm. After that, a high-level classification of the mainstream stochastic solvers is presented.

A. Comparison with Stochastic Galerkin (SG) Method

1) SG for Nonlinear Circuits: Similar to ST, SG starts from the residual function (10), but it sets up a deterministic DAE in the form (12) by Galerkin testing. Specifically, SG enforces the residual function to be orthogonal to each gPC basis function:

$$\left\langle \mathrm{Res}\left(\vec{X}(t),\vec{\xi}\right), H_k\left(\vec{\xi}\right)\right\rangle = 0, \text{ for } k=1,\cdots,K. \quad (27)$$

Now $Q(\vec{X}(t))$, $F(\vec{X}(t))$ and $\tilde{B}$ in (12) have the block form

$$\begin{aligned} Q\left(\vec{X}(t)\right) &= \begin{bmatrix} Q_1\left(\vec{X}(t)\right) \\ \vdots \\ Q_K\left(\vec{X}(t)\right) \end{bmatrix}, \quad \tilde{B} = \begin{bmatrix} B_1 \\ \vdots \\ B_K \end{bmatrix} \\ F\left(\vec{X}(t)\right) &= \begin{bmatrix} F_1\left(\vec{X}(t)\right) \\ \vdots \\ F_K\left(\vec{X}(t)\right) \end{bmatrix}, \end{aligned} \quad (28)$$

with the n_1-th block defined by

$$\begin{aligned} Q_{n_1}\left(\vec{X}(t)\right) &= \left\langle \vec{q}\left(\hat{x}(t,\vec{\xi}),\vec{\xi}\right), H_{n_1}(\vec{\xi})\right\rangle, \\ F_{n_1}\left(\vec{X}(t)\right) &= \left\langle \vec{f}\left(\hat{x}(t,\vec{\xi}),\vec{\xi}\right), H_{n_1}(\vec{\xi})\right\rangle, \\ B_{n_1} &= \left\langle B, H_{n_1}(\vec{\xi})\right\rangle. \end{aligned} \quad (29)$$

To obtain the above inner product, one can use numerical quadrature or Monte Carlo integration [48].

2) ST versus SG: Both of them are intrusive solvers, and the coupled DAEs from ST and SG have the same dimension. However, SG is much more expensive compared to ST.

First, SG must evaluate multivariate stochastic integrals, hence functions $\vec{q}$ and $\vec{f}$ must be evaluated at many quadrature or sampling nodes. This step is not cheap because evaluating a semiconductor device model (e.g., BISM3 model) at each node involves running tens of thousands of lines of codes.

Second, the linear system solution inside the Newton's iteration of SG is much more expensive. Assume that Gaussian quadrature is applied to calculate the inner products in (29), then the Jacobian $\mathcal{J}(\vec{X}_k^j)$ has the following structure

$$\mathcal{J}\left(\vec{X}_k^j\right) = \begin{bmatrix} \mathcal{J}_{1,1}\left(\vec{X}_k^j\right) & \cdots & \mathcal{J}_{1,K}\left(\vec{X}_k^j\right) \\ \vdots & \ddots & \vdots \\ \mathcal{J}_{K,1}\left(\vec{X}_k^j\right) & \cdots & \mathcal{J}_{K,K}\left(\vec{X}_k^j\right) \end{bmatrix}, \quad (30)$$

and the submatrix $\mathcal{J}_{n_1,n_2}\left(\vec{X}_k^j\right) \in \mathbb{R}^{n\times n}$ is calculated by

$$\mathcal{J}_{n_1,n_2}\left(\vec{X}_k^j\right) = \sum_{q=1}^{\hat{N}} w^q H_{n_1}\left(\vec{\xi}^q\right) H_{n_2}\left(\vec{\xi}^q\right) J\left(\vec{X}_k^j,\vec{\xi}^q\right).$$

Here $\vec{\xi}^q$ is the q-th Gaussian quadrature node and w^q the corresponding weight, $J\left(\vec{X}_k^j,\vec{\xi}^q\right)$ is calculated according to the definition in (17). The Jacobian in SG cannot be decoupled. Therefore, solving the resulting DAE of SG requires $O(N^3) = O(K^3n^3)$ at each time point if a direct solver is used (or $\hat{m}O(K^2n)$ if $\hat{m}$ iterations are used in an iterative solver), much more expensive compared to ST.

B. Comparison with Stochastic Collocation (SC) Method

1) SC for Nonlinear Circuits: Unlike ST and SG, SC starts from the original stochastic DAE (1) without using gPC approximation *a-priori*. SC first selects $\hat{N}_s$ samples $\vec{\xi}^1,\cdots,\vec{\xi}^{\hat{N}_s}$ and solves (1) at each sample to obtain a deterministic solution $\vec{x}(t,\vec{\xi}^k)$. The gPC coefficients are then computed using a post-processing step. For example, one can select the sample $\vec{\xi}^k$ and weight w^k by a Gauss-quadrature tensor product rule or sparse grid technique, and then compute the gPC coefficient by

$$\hat{x}_j(t) = \left\langle \vec{x}(t,\vec{\xi}), H_j(\vec{\xi})\right\rangle \approx \sum_{k=1}^{\hat{N}_s} w^k H_j(\vec{\xi}^k)\vec{x}(t,\vec{\xi}^k). \quad (31)$$

2) ST versus SC: Like MC, SC is a sampling-based simulator. Therefore, the cost of SC has a linear dependence on $\hat{N}_s$, the number of samples. However, SC uses more sampling nodes than ST (c.f. Section V-F). Furthermore, SC is not as efficient as ST in time-domain simulation. To reconstruct the gPC coefficients of time-domain solutions, SC must use the same time grid for simulating all deterministic DAEs. Since it is difficult to preselect an adaptive time grid, a small fixed step size is normally used, leading to excessive computational cost. In contrast, ST can use any standard adaptive step stepping to accelerate the time-domain simulation since it directly computes the gPC coefficients. It seems that SC can use adaptive time stepping to simulate each deterministic DAE, and then uses interpolation at the time points where solutions are missing. Unfortunately, the errors caused by such interpolations are much larger than the threshold inside Newton's iterations, causing inaccurate computation of higher-order gPC coefficients. However, SC indeed can use adaptive time stepping if one is not interested in the statistical information of the time-domain waveforms.

C. Classification and Summary

Fig. 2 shows the classification of different stochastic solvers, which is detailed below.

- MC and SC are nonintrusive (or sampling-based) solvers. They both start from the original stochastic equation (1) and compute the deterministic solutions at a set of sampling points. The main difference of MC and SC lies in how to select the samples. MC draws the samples randomly according to $\mathrm{PDF}(\vec{\xi})$, whereas SC selects the samples by a tensor-product (TP) numerical quadrature or sparse grid (SP) technique. After repeatedly simulating each deterministic equation, MC provides the statistical information such as distribution or moments, whereas SC reconstructs the gPC coefficients by a post-processing step such as numerical integration.
- SG and ST are intrusive solvers as they both directly compute the gPC coefficients by simulating a larger-size

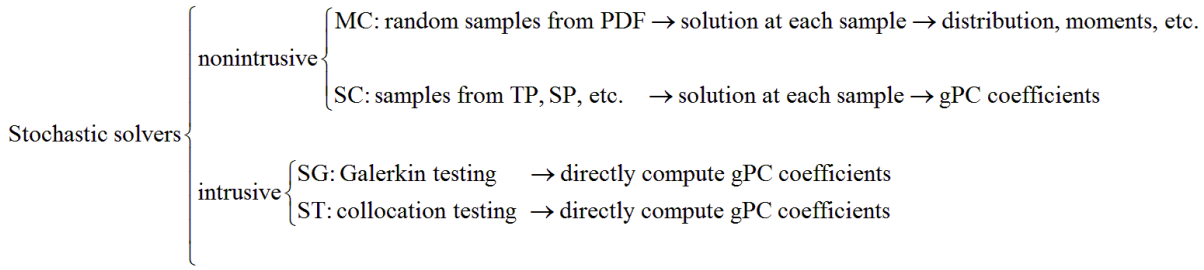


Fig. 2. The classification of MC, SG, SC and ST methods.

TABLE II
COMPARISON OF DIFFERENT SPECTRAL METHODS.

Method	Type	Decoupled?	Adapt. step size?
SC	nonintrusive	√	×
SG	intrusive	×	√
ST	intrusive	√	√

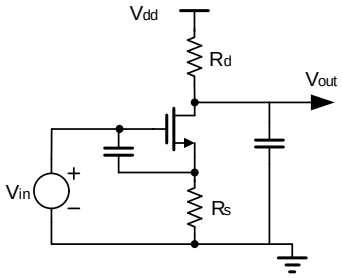


Fig. 3. Schematic of the common-source amplifier.

coupled DAE only once. With gPC approximations, they both start from the residual function (10). SG sets up a larger-size coupled deterministic model by Galerkin testing, whereas ST uses a collocation testing technique.

The spectral methods ST, SC and SG are further compared in Table II. ST allows both adaptive time stepping and decoupled simulation, therefore, it is more efficient over SC and SG.

V. NUMERICAL RESULTS

This section presents the simulation results of some analog, RF and digital integrated circuits. Our ST algorithm is implemented in a MATLAB prototype simulator and integrated with several semiconductor device models for algorithm verification. In this work, Level-3 MOSFET model and Ebers-Moll BJT model are used for transistor evaluation [49]. The TSMC 0.25μm CMOS model card [50] is used to describe the device parameters of all MOSFETs. SC, SG and Monte Carlo (MC) methods are implemented for comparison and validation. In SG and ST, step sizes are selected adaptively according to the local truncation error (LTE) [46] for time-domain simulation. In contrast, uniform step sizes are used for both MC and SC since we need to obtain the statistical information of time-domain solutions. In our experiments, all candidate nodes of ST are generated by Gaussian quadrature and tensor-product rules. The cost of generating the candidate nodes and selecting testing nodes is several milliseconds, which is negligible. For all circuit examples, SC and SG use the samples from a tensor-product rule. The sparse-grid and tensor-product SC methods are compared with ST in detail in Section V-F.

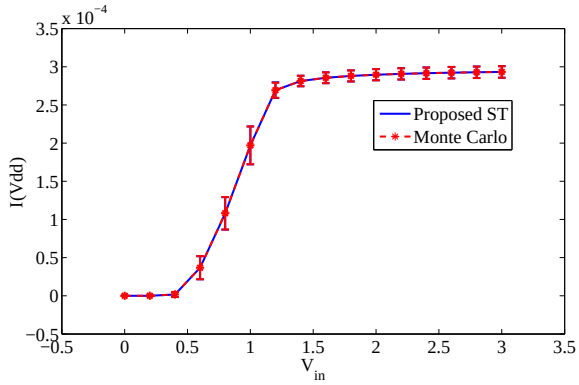


Fig. 4. Error bars showing the mean and s.t.d values from our ST method (blue) and Monte Carlo method (red) of $\mathrm{I}(\mathrm{V_{dd}})$.

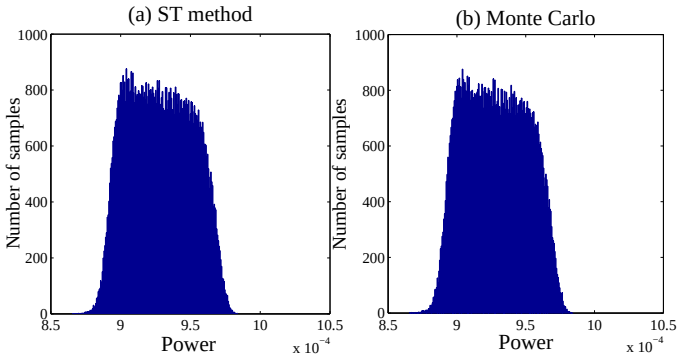


Fig. 5. Histograms showing the distributions of the power dissipation at $V_{\mathrm{in}} = 1.4$V, obtained by ST method (left) and Monte Carlo (right).

A. Illustrative Example: Common-Source (CS) Amplifier

The common-source (CS) amplifier in Fig. 3 is used to compare comprehensively our ST-based simulator with MC and other spectral methods. This amplifier has 4 random parameters: 1) $\mathrm{V_T}$ (threshold voltage when $\mathrm{V_{bs}} = 0$) has a normal distribution; 2) temperate T has a shifted and scaled Beta distribution, which influences $\mathrm{V_{th}}$; 3) $\mathrm{R_s}$ and $\mathrm{R_d}$ have Gamma and uniform distributions, respectively.

1) ST versus MC: ST method is first compared with MC in DC sweep. By sweeping the input voltage from 0 V up to

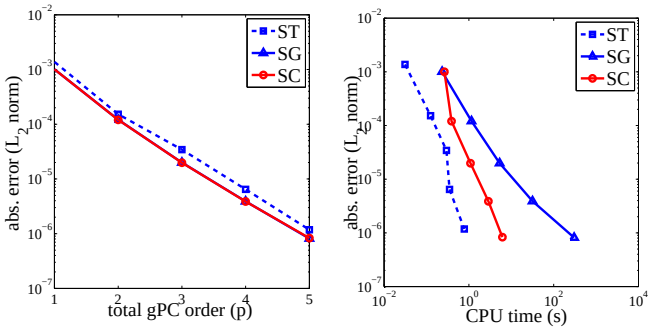


Fig. 6. Absolute errors (measured by L_2 norm) of the gPC coefficients for the DC analysis of the CS amplifier, with $V_{\rm in} = 1.6$V. Left: absolute errors versus gPC order p. Right: absolute errors versus CPU times.

TABLE III
COMPUTATIONAL COST OF THE DC ANALYSIS FOR CS AMPLIFIER.

gPC order (p)		1	2	3	4	5	6
ST	time (s)	0.16	0.22	0.29	0.51	0.78	1.37
	# nodes	5	15	35	70	126	210
SC	time (s)	0.23	0.33	1.09	2.89	6.18	11.742
	# nodes	16	81	256	625	1296	2401
SG	time (s)	0.25	0.38	5.33	31.7	304	1283
	# nodes	16	81	256	625	1296	2401

3 V with a step size of 0.2 V, we estimate the supply currents and DC power dissipation. In MC, 10^5 sampling points are used. In our ST simulator, using an order-3 truncated gPC expansion (with 35 gPC basis functions, and 35 testing nodes selected from 256 candidate nodes) achieves the same level of accuracy. The error bars in Fig. 4 show that the mean and s.t.d values from both methods are indistinguishable. The histograms in Fig. 5 plots the distributions of the power dissipation at $V_{\rm in} = 1.4$V. Again, the results obtained by ST is consistent with MC. The expected value at 1.4V is 0.928 mW from both methods, and the s.t.d. value is 22.07 μW from both approaches. Apparently, the variation of power dissipation is not a Gaussian distribution due to the presence of circuit nonlinearity and non-Gaussian random parameters.

CPU times: For this DC sweep, MC costs about 2.6 hours, whereas our ST simulator only costs 5.4 seconds. Therefore, a 1700$\times$ speedup is achieved by using our ST simulator.

2) ST versus SC and SG in DC Analysis: Next, ST method is compared with SG and SC. Specifically, we set $V_{\rm in} = 1.6$V and compute the gPC coefficients of all state variables with the total gPC order p increasing from 1 to 6. We use the results from $p = 6$ as the "exact solution" and plot the L_2 norm of the absolute errors of the computed gPC coefficients versus p and CPU times, respectively. The left part of Fig. 6 shows that as p increases, ST, SC and SG all converge very fast. Although ST has a slightly lower convergence rate, its error still rapidly reduces to below 10^{-4} when $p = 3$. The right part of Fig. 6 shows that ST costs the least CPU time to achieve the same level of accuracy with SC and SG, due to the decoupled Newton's iterations and fewer nodes used in ST.

CPU times: The computational costs of different solvers are summarized in Table III. The speedup of ST becomes more significant as the total gPC order p increases. We remark that the speedup factor will be smaller if SC uses sparse grids, as will be discussed in Section V-F.

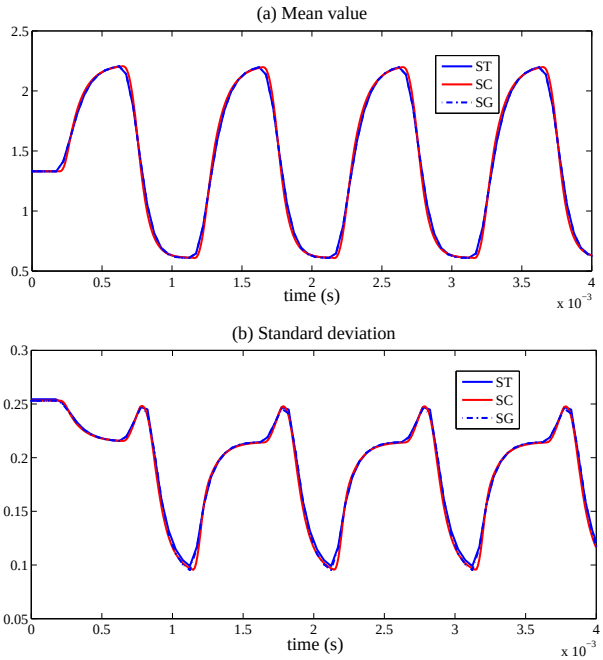


Fig. 7. Transient waveform of the output of the CS amplifier.

TABLE IV
COMPUTATIONAL COST OF TRANSIENT SIMULATION FOR CS AMPLIFIER.

Methods	ST	SG	SC
CPU times	41 s	$>$ 1 h	1180 s
# nodes	35	256	256
speedup of ST	1	$>$ 88	29

3) ST versus SC and SG in Transient Simulation: Finally, ST is compared with SG and SC in transient simulation. It is well known that the SG method provides an optimal solution in terms of accuracy [32]–[34], therefore, the solution from SG is used as the reference for accuracy comparison. The total gPC order is set as $p = 3$ (with $K = 35$ testing nodes selected from 256 candidate nodes), and the Gear-2 integration scheme [46] is used for all spectral methods. In SC, a uniform step size of 10μs is used, which is the largest step size that does not cause simulation failures. The input is kept as $V_{\rm in} =$ 1 V for 0.2 ms and then added with a small-signal square wave (with 0.2V amplitude and 1 kHz frequency) as the AC component. The transient waveforms of $V_{\rm out}$ are plotted in Fig. 7. The mean value and standard deviation from ST are almost indistinguishable with those from SG.

It is interesting that the result from ST is more accurate than that from SC in this transient simulation example. This is because of the employment of LTE-based step size control [46]. With a LTE-based time stepping [46], the truncation errors caused by numerical integration can be well controlled in ST and SG. In contrast, SC cannot adaptively select the time step sizes according to LTEs, leading to larger integration errors.

CPU times: The computational costs of different solvers are summarized in Table IV. It is noted that SC uses about 7$\times$ of nodes of ST, but the speedup factor of ST is 29. This

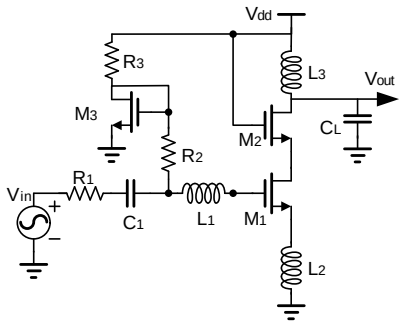


Fig. 8. Schematic of the LNA.

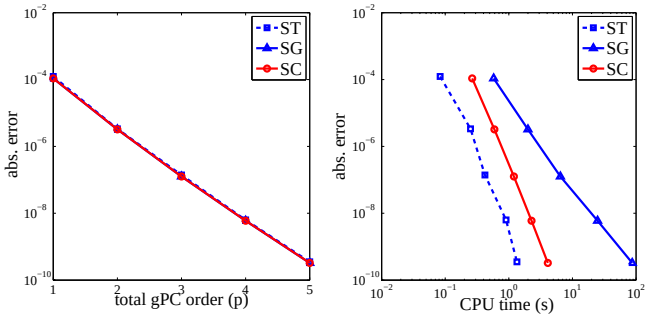


Fig. 9. Absolute errors (measured by L_2 norm) of the gPC coefficients for the DC analysis of LNA. Left: absolute errors versus gPC order p. Right: absolute errors versus CPU times.

is because the adaptive time stepping in ST causes an extra speedup factor of about 4. MC is prohibitively expensive for transient simulation and thus not compared here.

TABLE V
COMPUTATIONAL COST OF THE DC ANALYSIS FOR LNA.

gPC order (p)		1	2	3	4	5	6
ST	time (s)	0.24	0.33	0.42	0.90	1.34	2.01
	# nodes	4	10	20	35	56	84
SC	time (s)	0.26	0.59	1.20	2.28	4.10	6.30
	# nodes	8	27	64	125	216	343
SG	time (s)	0.58	2.00	6.46	24.9	87.2	286
	# nodes	8	27	64	125	216	343

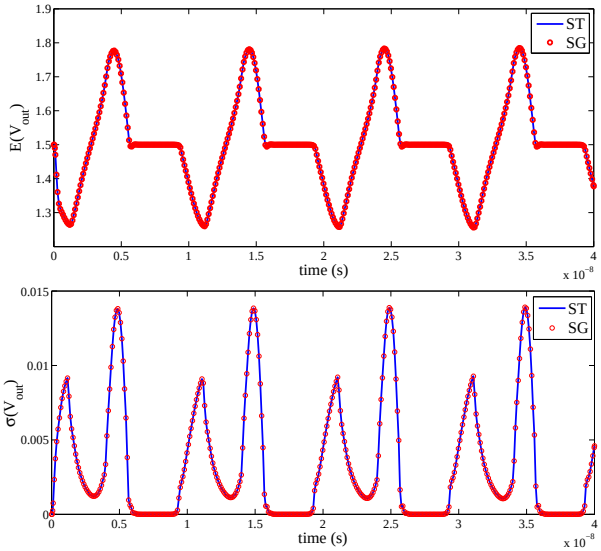


Fig. 10. Transient simulation results of the LNA. Upper part: expectation of the output voltage; bottom part: standard deviation of the output voltage.

B. Low-Noise Amplifier (LNA)

Now we consider a practical low-noise amplifier (LNA) shown in Fig 8. This LNA has 3 random parameters in total: resistor R_3 is a Gamma-type variable; R_2 has a uniform distribution; the gate width of M_1 has a uniform distribution.

DC Analysis: We first run DC analysis by ST, SC and SG with p increasing from 1 to 6, and plot the errors of the gPC coefficients of the state vector versus p and CPU times in Fig. 9. For this LNA, ST has almost the same accuracy with SC and SG, and it requires the smallest amount of CPU time. The cost of the DC analysis is summarized in Table V.

Transient Analysis: An input signal $V_{\mathrm{in}} = 0.5\sin(2\pi f t)$ with $f = 10^8$ Hz is added to this LNA. We are interested in the uncertainties of the transient waveform at the output. Setting $p = 3$, our ST method uses 20 gPC basis functions (with 20 testing nodes selected from 64 candidate nodes) to obtain the waveforms of the first 4 cycles. The result from ST is indistinguishable with that from SG, as shown in Fig. 10. ST consumes only 56 seconds for this LNA. Meanwhile, SG costs 26 minutes, which is $28\times$ slower compared to ST.

C. 6-T SRAM Cell

The 6-T SRAM cell in Fig. 11 is studied to show the application of ST in digital cell analysis. When the write line has a high voltage (logic 1), the information of the bit line can be written into the cell and stored on transistors $\mathrm{M}_1 - \mathrm{M}_4$. The 1-bit information is represented by the voltage of node Q. When the write line has a low voltage (logic 0), M_5 and M_6 turn off. In this case, $\mathrm{M}_1 - \mathrm{M}_4$ are disconnected with the bit line, and they form a latch to store and hold the state of node Q. Here V_{dd} is set as 1 V, while the high voltages of the write and bit lines are both set as 2 V.

Now we assume that due to mismatch, the gate widths of $\mathrm{M}_1 - \mathrm{M}_4$ have some variations which can be expressed as Gaussian variables. Here we study the influence of device variations on the transient waveforms, which can be further used for power and timing analysis. Note that in this paper we do not consider the rare failure events of SRAM cells [24]. To quantify the uncertainties of the voltage waveform at node Q, our ST method with $p = 3$ and $K = 35$ (with 35 testing nodes selected from 256 candidate nodes) is applied to perform transient simulation under a given input waveforms. Fig. 12 shows the waveforms of write and bit lines and the corresponding uncertainties during the time interval $[0, 1]\mu$s.

CPU times: Our ST method costs 6 minutes to obtain the result. SG generates the same results at the cost of several hours. Simulating this circuit with SC or MC is prohibitively expensive, as a very small uniform step size must be used due to the presence of sharp state transitions.

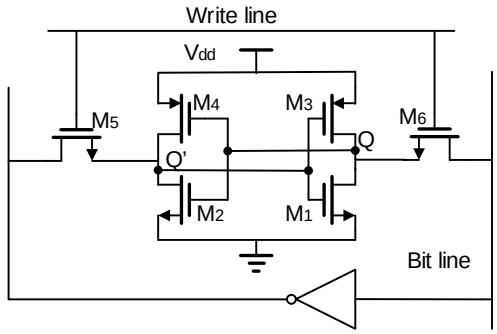


Fig. 11. Schematic of the CMOS 6-T SRAM.

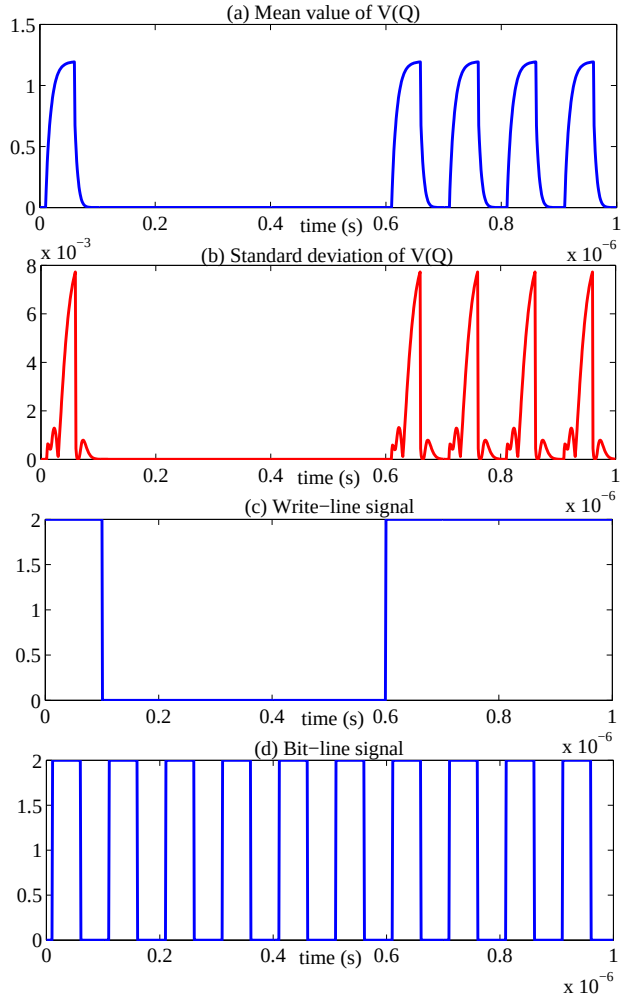


Fig. 12. Uncertainties of the SRAM cell. (a) and (b) shows the expectation and standard deviation of $V_{\rm out}$; (c) and (d) shows the waveforms of the write line and bit line, respectively.

D. BJT Feedback Amplifier

To show the application of our ST method in AC analysis and in BJT-type circuits, we consider the feedback amplifier in Fig. 13. In this circuit, R_1 and R_2 have Gamma-type uncertainties. The temperature is a Gaussian variable which significantly influences the performances of BJTs and diodes. Therefore, the transfer function from $V_{\rm in}$ to $V_{\rm out}$ is uncertain.

Using $p = 3$ and $K = 20$ (with 20 testing nodes selected

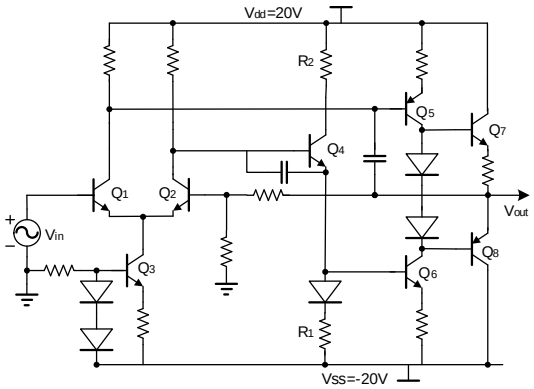


Fig. 13. Schematic of the BJT feedback amplifier.

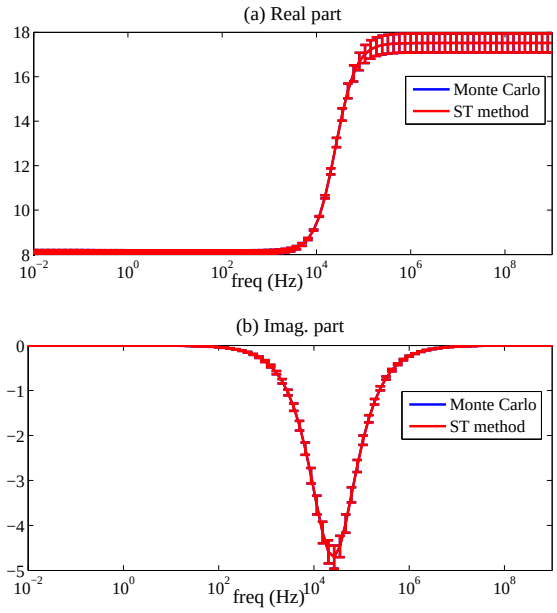


Fig. 14. Uncertainties of the transfer function of the BJT amplifier.

from 64 candidate nodes), our ST simulator achieves the similar level of accuracy of a MC simulation using 10^5 samples. The error bars in Fig. 14 show that the results from both methods are indistinguishable. In ST, the real and imaginary parts of the transfer functions are both obtained as truncated gPC expansions. Therefore, the signal gain at each frequency point can be easily calculated with a simple polynomial evaluation. Fig. 15 shows the calculated PDF of the small-signal gain at $f = 8697.49$ Hz using both ST and MC. The PDF curves from both methods are indistinguishable.

CPU times: The simulation time of ST and Monte Carlo are 3.6 seconds and over 2000 seconds, respectively.

E. BJT Double-Balanced Mixer

As the final circuit example, we consider the time-domain simulation of RF circuits excited by multi-rate signals, by studying the double-balanced mixer in Fig. 16. Transistors $\rm Q_1$ and $\rm Q_2$ accept an input voltage of frequency f_1, and $\rm Q_3 \sim Q_6$ accept the second input of frequency f_2. The output

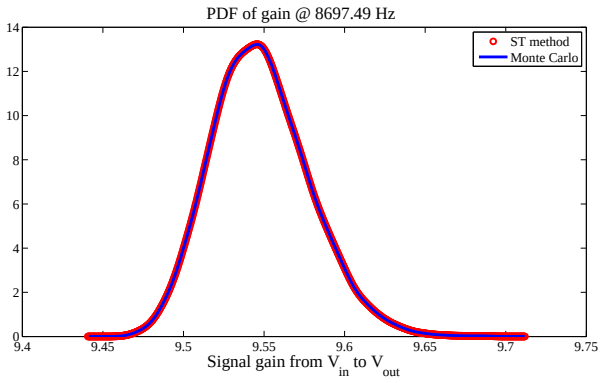


Fig. 15. Simulated probability density functions of the signal gain.

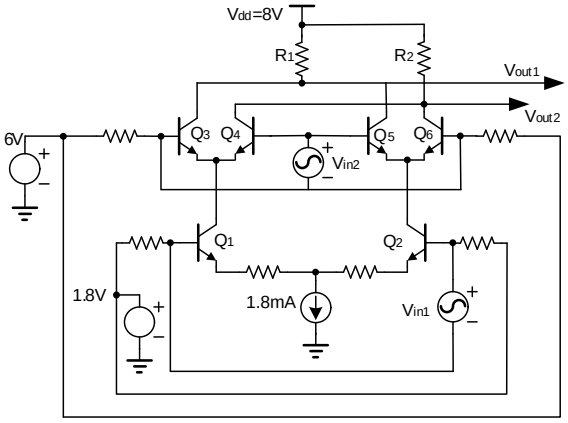


Fig. 16. Schematic of the BJT double-balanced mixer.

$v_{\rm out} = V_{\rm out1} - V_{\rm out2}$ will have components at two frequencies: one at $|f_1 - f_2|$ and the other at $f_1 + f_2$. Now we assume that R_1 and R_2 are both Gaussian-type random variables, and we measure the uncertainties of the output voltage. In our simulation, we set $V_{\rm in1} = 0.01\sin(2\pi f_1 t)$ with $f_1 = 4$ MHz and $V_{\rm in2} = 0.01\sin(2\pi f_2 t)$ with $f_2 = 100$ kHz. We set $p = 3$ and $K = 10$ (with 10 testing nodes selected from 16 candidate nodes), and then use our ST simulator to run a transient simulation from $t = 0$ to $t = 30\mu$s. The expectation and standard deviation of $V_{\rm out1} - V_{\rm out2}$ are plotted in Fig. 17.

CPU times: The cost of our ST method is 21 minutes, whereas simulating this mixer by SG, SC or MC on the same MATLAB platform is prohibitively expensive due to the presence of multi-rate signals and the large problem size.

F. Discussion: Speedup Factor of ST over SC

Finally we comprehensively compare the costs of ST and SC. Two kinds of SC methods are considered according to the sampling nodes used in the solvers [39]: SC using tensor product (denoted as SC-TP) and SC using sparse grids (denoted as SC-SP). SC-TP uses $(p+1)^l$ nodes to reconstruct the gPC coefficients, and the work in [38] belongs to this class. For SC-SP, a level-$p+1$ sparse grid must be used to obtain the p-th-order gPC coefficients in (31). We use the Fejèr nested sparse grid in [42], and according to [51] the total number of

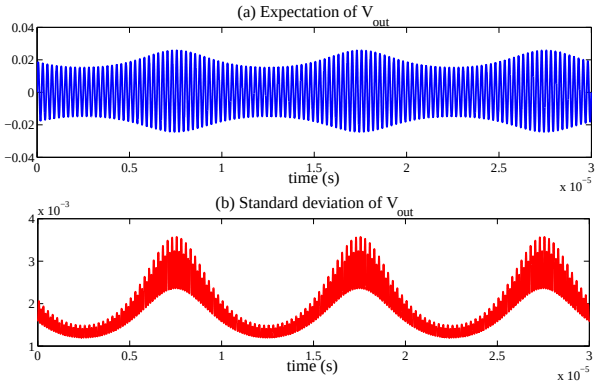


Fig. 17. Uncertainties of $V_{\rm out}$=$V_{\rm out1} - V_{\rm out2}$ of the double-balanced mixer.

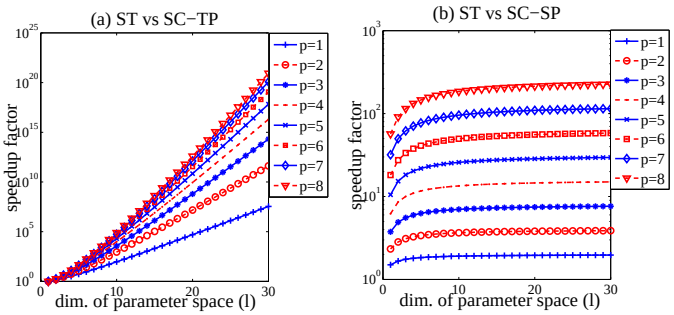


Fig. 18. The speedup factor of ST over SC caused by node selection: (a) ST versus SC-TP, (b) ST versus SC-SP. This is also the speedup factor in DC analysis.

nodes in SC-SP is estimated as

$$N_{\rm SC-SP} = \sum_{i=0}^{p} 2^i \frac{(l-1+i)!}{(l-1)!i!} \tag{32}$$

DC Analysis: In DC analysis, since both ST and SC use decoupled solvers and their costs linearly depend on the number of nodes, the speedup factor of ST versus SC is

$$\nu_{\rm DC} \approx N_{\rm SC}/K \tag{33}$$

where $N_{\rm SC}$ and K are the the numbers of nodes used by SC and ST, respectively. Fig. 18 plots the values of $N_{\rm SC}/K$ for both SC-TP and SC-SP, which is also the speedup factor of ST over SC in DC analysis. Since ST uses the smallest number of nodes, it is more efficient over SC-TP and SC-SP. When low-order gPC expansions are used ($p \leq 3$), the speedup factor over SC-SP is below 10. The speedup factor can be above 10 if $p \geq 4$, and it gets larger as p increases. In high-dimensional cases ($l \gg 1$), the speedup factor of ST over SC-SP only depends on p. It is the similar case if Smolyak sparse grids are used in SC [31]. For example, compared with the sparse-grid SC in [31], our ST has a speedup factor of 2^p if $l \gg 1$.

Transient Simulation: The speedup factor of ST over SC in a transient simulation can be estimated as

$$\nu_{\rm Trans} \approx (N_{\rm SC}/K) \times \kappa, \text{ with } \kappa > 1, \tag{34}$$

which is larger than $\nu_{\rm DC}$. The first part is the same as in DC analysis. The second part κ represents the speedup caused by adaptive time stepping in our intrusive ST simulator, which is

case dependent. For weakly nonlinear analog circuits (e.g., the SC amplifier in Section V-A), κ can be below 10. For digital cells (e.g., the SRAM cell in Section V-C) and multi-rate RF circuits (e.g., the double-balanced mixer in Section V-E), SC-based transient simulation can be prohibitively expensive due to the inefficiency of using a small uniform time step size. In this case, κ can be significantly large.

VI. Conclusion

This paper has proposed an intrusive-type stochastic solver, named stochastic testing (ST), to quantify the uncertainties in transistor-level circuit analysis. With gPC expansions, ST can handle both Gaussian and non-Gaussian variations. Compared with SG and SC, ST can simultaneously allow decoupled numerical simulation and adaptive step size control. In addition, multivariate integral calculation is avoided in ST. Such properties make ST method hundreds to thousands of times faster over Monte Carlo, and tens to hundreds of times faster than SG. The speedup of ST over SC is caused by two factors: 1) a smaller number of nodes required in ST; and 2) adaptive time stepping in the intrusive ST simulator. The overall speedup factor of ST over SC is normally case dependent. Various simulations (e.g., DC, AC and transient analysis) are performed on some analog, digital and RF circuits, demonstrating the effectiveness of our proposed algorithm.

References

[1] D. S. Boning, "Variation," *IEEE Trans. Semiconductor Manufacturing*, vol. 21, no. 1, pp. 63–71, Feb 2008.

[2] S.-W. Sun and P. G. Y. Tsui, "Limitation of CMOS supply-voltage scaling by MOSFET threshold-voltage variation," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 8, pp. 947–949, Aug 1995.

[3] N. Tega, H. Miki, F. Pagette, D. J. Frank, A. Ray, M. J. Rooks, W. Haensch, and K. Torii, "Increasing threshold voltage variation due to random telegraph noise in FETs as gate lengths scale to 20 nm," in *Proc. Intl. Symp. VLSI Technology*, Jun. 2009, pp. 50–51.

[4] T. Moselhy and L. Daniel, "Stochastic integral equation solver for efficient variation aware interconnect extraction," in *Proc. Design Auto. Conf.*, Jun. 2008, pp. 415–420.

[5] ——, "Stochastic dominant singular vectors method for variation-aware extraction," in *Proc. Design Auto. Conf.*, Jun. 2010, pp. 667–672.

[6] ——, "Variation-aware stochastic extraction with large parameter dimensionality: Review and comparison of state of the art intrusive and non-intrusive techniques," in *Proc. Intl. Symp. Quality Electronic Design*, Mar. 2011, pp. 14–16.

[7] T. A. El-Moselhy, "Field solver technologies for variation-aware interconnect parasitic extraction," PhD Dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 2010.

[8] D. V. Ginste, D. D. Zutter, D. Deschrijver, T. Dhaene, P. Manfredi, and F. Canavero, "Stochastic modeling-based variability analysis of on-chip interconnects," *IEEE Trans. Components, Packaging and Manufacturing Technology*, vol. 2, no. 7, pp. 1182–1192, Jul. 2012.

[9] I. S. Stievano, P. Manfredi, and F. G. Canavero, "Carbon nanotube interconnects: Process variation via polynomial chaos," *IEEE Trans. Electromagnetic Compatibility*, vol. 54, no. 1, pp. 140–148, Feb. 2012.

[10] Z. Zhang, I. M. Elfadel, and L. Daniel, "Model order reduction of fully parameterized systems by recursive least square optimization," in *Proc. Intl. Conf. Computer-Aided Design*, Jun. 2011, pp. 523–530.

[11] L. Daniel, C. S. Ong, S. C. Low, K. H. Lee, and J. K. White, "A multiparameter moment-matching model-reduction approach for generating geometrically parameterized interconnect performance models," *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 23, no. 5, pp. 678–693, May. 2004.

[12] V. Mehrotra, S. Nassif, D. Boning, and J. Chung, "Modeling the effects of manufacturing variation on high-speed microprocessor interconnect performance," in *Proc. Intl. Electron Devices Meeting*, Dec. 1998, pp. 767–770.

[13] K. Agarwal, D. Sylvester, D. Blaauw, F. Liu, S. Nassif, and S. Vrudhula, "Variational delay metrics for interconnect timing analysis," in *Proc. Design Auto. Conf.* New York, NY, Jun. 2004, pp. 381–384.

[14] C. H. Stapper and R. J. Rosner, "Integrated circuit yield management and yield analysis: development and implementation," *IEEE Trans. Semiconductor Manufacturing*, vol. 8, no. 2, pp. 95–102, May 1995.

[15] S. R. Nassif, "Modeling and analysis of manufacturing variations," in *Proc. Intl. Conf. Custom Integrated Circuits*, Sept. 2001, pp. 223 – 228.

[16] D. S. Boning and S. Nassif, "Models of process variations in device and interconnect," in *Design of High Performance Microprocessor Circuits*. IEEE Press, 2000.

[17] X. Li, "Finding deterministic solution from underdetermined equation: large-scale performance modeling of analog/RF circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 11, pp. 1661–1668, Nov 2011.

[18] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, S. Narayan, D. K. Beece, J. Piaget, N. Venkateswaran, and J. G. Hemmett, "First-order incremental block-based statistical timing analysis," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 2170 – 2180, Oct 2006,.

[19] W. Zhang, X. Li, F. Liu, E. Acar, R. Rutenbar, and R. Blanton, "Virtual probe: a statistical framework for low-cost silicon characterization of nanoscale integrated circuits," *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 30, no. 12, pp. 1814–1827, Dec 2011.

[20] P. W. Tuinenga, *Spice: A Guide to Circuit Simulation and Analysis Using PSpice*, 3rd ed. Upper Saddle River, NJ: Prentice Hall PTR, 1995.

[21] `http://www.cadence.com`.

[22] `http://www.synopsys.com/Tools/Verification`.

[23] N. Metropolis and S. Ulam, "The Monte Carlo method," *Journal of the American Statistical Association*, vol. 44, no. 247, pp. 335–341, 1949.

[24] R. Kanj, R. Joshi, and S. Nassif, "Mixture importance sampling and its application to the analysis of SRAM designs in the presence of rare failure events," in *Proc. Design Automation Conf.* San Francisco, CA, Jun 2006, pp. 69–72.

[25] A. Singhee and R. A. Rutenbar, "Statistical blockade: Very fast statistical simulation and modeling of rare circuit events and its application to memory design," *IEEE Trans. on CAD of Integrated Circuits and systems*, vol. 28, no. 8, pp. 1176–1189, Aug. 2009.

[26] ——, "Why Quasi-Monte Carlo is better than Monte Carlo or latin hypercube sampling for statistical circuit analysis," *IEEE Trans. CAD Integrated Circuits and Systems*, vol. 29, no. 11, pp. 1763–1776, Nov. 2010.

[27] J. Tao, X. Zeng, W. Cai, Y. Su, D. Zhou, and C. Chiang, "Stochastic sparse-grid collocation algorithm (SSCA) for periodic steady-state analysis of nonlinear system with process variations," in *Porc. Asia and South Pacific Design Automation Conf.*, 2007, pp. 474–479.

[28] K. Strunz and Q. Su, "Stochastic formulation of SPICE-type electronic circuit simulation with polynomial chaos," *ACM Trans. Modeling and Computer Simulation*, vol. 18, no. 4, Sep. 2008.

[29] N. Wiener, "The homogeneous chaos," *American Journal of Mathematics*, vol. 60, no. 4, p. 897936, Oct 1938.

[30] R. Ghanem and P. Spanos, *Stochastic finite elements: a spectral approach*. Springer-Verlag, 1991.

[31] D. Xiu and J. S. Hesthaven, "High-order collocation methods for differential equations with random inputs," *SIAM Journal on Scientific Computing*, vol. 27, no. 3, pp. 1118–1139, Mar 2005.

[32] D. Xiu and G. E. Karniadakis, "The Wiener-Askey polynomial chaos for stochastic differential equations," *SIAM Journal on Scientific Computing*, vol. 24, no. 2, pp. 619–644, Feb 2002.

[33] ——, "Modeling uncertainty in flow simulations via generalized polynomial chaos," *Journal of Computational Physics*, vol. 187, no. 1, pp. 137–167, May 2003.

[34] D. Xiu, "Fast numerical methods for stochastic computations: A review," *Communications in Computational Physics*, vol. 5, no. 2-4, pp. 242–272, Feb. 2009.

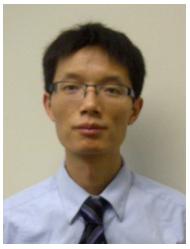
[35] A. Sandu, C. Sandu, and M. Ahmadian, "Modeling multibody systems with uncertainties. part I: Theoretical and computational aspects," *Multibody Syst. Dyn.*, vol. 15, pp. 373–395, Sept. 2006.

[36] A. Narayan and D. Xiu, "Stochastic collocation methods on unstructured grids in high dimensions via interpolation," *SIAM J. Scientific Computing*, vol. 34, no. 3, pp. 1729–1752, Mar 2012.

[37] ——, "Stochastic collocation with least orthogonal interpolant Leja sequences," in *SIAM Conf. Computational Science and Engineering*, Feb.-Mar. 2013.

[38] R. Pulch, "Stochastic collocation and stochastic Galerkin methods for linear differential algebraic equations," available at

http://www.imacm.uni-wuppertal.de/fileadmin/imacm/preprints/2012/imacm_12_31.pdf.
[39] J. Bäck, F. Nobile, L. Tamellini, and R. Tempone, "Stochastic spectral Galerkin and collocation methods for PDEs with random coefficients: A numerical comparison," *Spectral and High Order Methods for Partial Differential Equations, Lecture Notes in Computational Science and Engineering*, vol. 76, pp. 43–62, 2011.
[40] D. Xiu, *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton University Press, 2010.
[41] P. Sumant, H. Wu, A. Cangellaris, and N. R. Aluru, "Reduced-order models of finite element approximations of electromagnetic devices exhibiting statistical variability," *IEEE Trans. Antennas and Propagation*, vol. 60, no. 1, pp. 301–309, Jan. 2012.
[42] O. Le Maitre and O. Knio, *Spectral methods for uncertainty quantification: with application to computational fluid dynamics*. Springer, 2010.
[43] C.-W. Ho, A. Ruehli, and P. Brennan, "The modified nodal approach to network analysis," *IEEE Trans. Circuits and Systems*, vol. CAS-22, no. 6, pp. 504–509, Jun. 1975.
[44] C. Soize and R. Ghanem, "Physical systems with random uncertainties: Chaos representations with arbitrary probability measure," *SIAM Journal on Scientific Computing*, vol. 26, no. 2, p. 395410, Feb 2004.
[45] K. Nabors and J. White, "FastCap: a multipole accelerated 3-D capacitance extraction program," *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 10, no. 11, pp. 1447–1459, Nov. 1991.
[46] K. S. Kundert, *The Designers Guide to SPICE and Spectre*. Boston, MA: Kluwer Academic Publishers, 1995.
[47] D. Calvetti and L. Reichel, "Fast inversion of Vandermonde-like matrices involving orthogonal polynomials," *BIT Numerical Mathematics*, vol. 33, no. 3, pp. 473–484, 1994.
[48] S. Weinzierl, "Introduction to Monte Carlo methods," NIKHEF, Theory Group, Amsterdam, The Netherlands, Tech. Rep. NIKHEF-00-012, 2000.
[49] "Star-HSPICE users manual," avant! Corporation, Sunnyvale, CA, Feb. 1996.
[50] http://www.mosis.com/files/test_data/t14y_tsmc_025_level3.txt.
[51] H.-J. Bungartz and M. Griebel, "Sparse grids," *Acta Numerica*, vol. 13, pp. 147–269, 2004.



Zheng Zhang (S'09) received his B.Eng. degree from Huazhong University of Science and Technology, China, in 2008, and M.Phil. degree from the University of Hong Kong, Hong Kong, in 2010. He is a Ph.D student in Electrical Engineering at the Massachusetts Institute of Technology (MIT), Cambridge, MA. His research interests include uncertainty quantification, numerical methods for the computer-aided design (CAD) of integrated circuits and microelectromechanical systems (MEMS), and model order reduction.

In 2009, Mr. Zhang was a visiting scholar with the University of California, San Diego (UCSD), La Jolla, CA. In 2011, he collaborated with Coventor Inc., working on CAD tools for MEMS design. He was recipient of the Li Ka Shing Prize (university best M.Phil/Ph.D thesis award) from the University of Hong Kong, in 2011, and the Mathworks Fellowship from MIT, in 2010.

Tarek El-Moselhy received the B.Sc. degree in electrical engineering in 2000 and a diploma in mathematics in 2002, then the M.Sc. degree in mathematical engineering, in 2005, all from Cairo University, Cairo, Egypt. He received the Ph.D. degree in electrical engineering from Massachusetts Institute of Technology, Cambridge, in 2010.

He is a postdoctoral associate in the Department of Aeronautics and Astronautics at Massachusetts Institute of Technology (MIT). His research interests include fast algorithms for deterministic and stochastic electromagnetic simulations, stochastic algorithms for uncertainty quantification in high dimensional systems, and stochastic inverse problems with emphasis on Bayesian inference. Dr. El-Moselhy received the Jin Au Kong Award for Outstanding PhD Thesis in Electrical Engineering from MIT in 2011, and the IBM Ph.D Fellowship in 2008.

Ibrahim (Abe) M. Elfadel (SM'-02) received his Ph.D. from Massachusetts Institute of Technology (MIT) in 1993 and is currently Professor and Head of Microsystems Engineering at the Masdar Institute of Science and Technology, Abu Dhabi, UAE.

He has 15 years of industrial experience with IBM in the research, development and deployment of advanced computer-aided design (CAD) tools and methodologies for deep-submicron, high-performance digital designs. His groups research is concerned with several aspects of energy-efficient digital system design and includes CAD for variation-aware, low-power nano-electronics, power and thermal management of multicore processors, embedded DSP for mmWave wireless systems, modeling and simulation of micro power sources, and 3D integration for energy-efficeint VLSI design. Dr. Elfadel is the Director of the TwinLab/Abu Dhabi Center for 3D IC Design, a joint R & D program with the Technical University of Dresden, Germany.

Dr. Elfadel is the recipient of six Invention Achievement Awards, an Outstanding Technical Achievement Award and a Research Division Award, all from IBM, for his contributions in the area of VLSI CAD. He is currently serving as an Associate Editor for the IEEE Transactions on Computer-Aided Design for Integrated Circuits and Systems and the IEEE Transactions on Very-Large-Scale Integration.

Luca Daniel (S'98-M'03) received the Laurea degree (*summa cum laude*) in electronic engineering from the Universita di Padova, Italy, in 1996, and the Ph.D. degree in electrical engineering from the University of California, Berkeley, in 2003.

He is an Associate Professor in the Electrical Engineering and Computer Science Department of the Massachusetts Institute of Technology (MIT), Cambridge. His research interests include accelerated integral equation solvers and parameterized stable compact dynamical modeling of linear and nonlinear dynamical systems with applications in mixed-signal/RF/mm-wave circuits, power electronics, MEMs, and the human cardiovascular system.

Dr. Daniel received the 1999 IEEE TRANSACTIONS ON POWER ELECTRONICS best paper award, the 2003 ACM Outstanding Ph.D. Dissertation Award in Electronic Design Automation, five best paper awards in international conferences, the 2009 IBM Corporation Faculty Award, and 2010 Early Career Award from the IEEE Council on Electronic Design Automation