Dynamic-Range Estimation

Bin Wu, *Student Member, IEEE*, Jianwen Zhu, *Member, IEEE*, and Farid N. Najm, *Fellow, IEEE*

***Abstract*—It has been widely recognized that the dynamic-range information of an application can be exploited to reduce the datapath bitwidth of either processors or application-specific integrated circuits and, therefore, the overall circuit area, delay, and power consumption. While recent proposals of analytical dynamic-range-estimation methods have shown significant advantages over the traditional profiling-based method in terms of runtime, it is argued here that the rather simplistic treatment of input correlation and system nonlinearity may lead to significant error. In this paper, three mathematical tools, namely Karhunen–Loève expansion, polynomial chaos expansion, and independent component analysis are introduced, which enable not only the orthogonal decomposition of input random processes, but also the propagation of random processes through both linear and nonlinear systems with difficult constructs such as multiplications, divisions, and conditionals. It is shown that when applied to interesting nonlinear applications such as adaptive filters, polynomial filters, and rational filters, this method can produce complete accurate statistics of each internal variable, thereby allowing the synthesis of bitwidth with the desired tradeoff between circuit performance and signal-to-noise ratio.**

***Index Terms*—Bitwidth, correlation, dynamic range, independent component analysis (ICA), non-Gaussian, nonlinear, polynomial chaos expansion (PCE), Karhunen–Loève expansion (KLE).**

I. Introduction

TODAY'S application-specific integrated circuit (ASIC) designers start with a design specification handed off by system designers. Often in the form of C code, the algorithm-level design specification needs to be converted into register transfer level (RTL) design, typically in the form of hardware description languages. A crucial decision to be made during this process is the datapath bitwidth, including the bitwidths of different registers and functional units. An aggressively designed datapath often replaces floating-point arithmetic contained in the design specification by their fixed-point counterparts. In addition, the redundant bits that do not contribute much to the accuracy of the application are often eliminated. Such datapaths with minimal bitwidth always translate to superior circuit performance in terms of area, speed, and power consumption. To make this possible, the dynamic-range information of the application and, in the case of C code, the dynamic ranges of all declared variables and intermediate expressions (all referred to as signals or variables in the following text) have to be obtained.

Unfortunately, the best practice today for dynamic-range estimation is still profiling (also referred to as simulation), which works by instrumenting the original application with a code that can trace the value ranges at runtime. While this method can be made very accurate, the accuracy is achieved only by extremely long simulation since the design space to be explored could be large; worst of all, no confidence on the accuracy can be obtained. In contrast, analytical methods can avoid long simulation by analyzing the application at compile time. While many advances have been made on this front, the proposed methods have not been able to provide dynamic-range information as accurate and as complete as profiling. The accuracy problem can usually be attributed to either no treatment of signal correlation, as in the cases of bitwidth [1] or moment propagation methods [2], or inadequate treatment of signal correlation, as in the case of affine arithmetic method [3], [4]. The completeness problem can be attributed to the fact that these methods typically produce only value or error bounds instead of signal distribution, thereby limiting the scope of their application.

In this paper, we first propose a new analytical framework based on Karhunen–Loève expansion (KLE) and demonstrate its effectiveness in linear systems, which characterize a large class of useful applications. The KLE method allows us to decompose the system input, modeled as an arbitrary random process, into K number of deterministic signals, each multiplied by a random variable (RV). Exploiting the linearity of the system, we can therefore obtain the system response of a random input by combining system responses of K deterministic inputs, each multiplied by the corresponding RV.

While this method can capture spatial and temporal correlations effectively, it relies on the application of superposition, a property enjoyed only by linear systems. By definition, a system $y = f(x)$ is nonlinear if it does not satisfy the superposition property, that is, $f(a_1x_1 + a_2x_2) \neq a_1f(x_1) + a_2f(x_2)$. In practice, whenever operations such as multiplications, divisions, and conditionals are used in the corresponding C implementation, a system is likely to be nonlinear. Therefore, the majority of C applications are nonlinear. Even in the domain of digital signal processing (DSP), where linear filters are most widely used, nonlinear systems comprise a large important category of applications. For example, in image processing systems, it is typical to have applications that use logarithmic nonlinearity to model the human eye. Similarly, neural networks apply nonlinear operations to linear combinations of inputs to mimic the nervous system. Other common examples include median filters, where the output is the median of the input values in the corresponding neighborhood, polynomial filters,

Manuscript received September 23, 2004; revised February 1, 2005 and May 15, 2005. This work was supported by Micronet, with funding from Array Technologies Inc. (ATI) and from Altera Corp. This paper was recommended by Associate Editor R. Gupta.

The authors are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: bwu@eecg.toronto.edu; jzhu@eecg.toronto.edu; najm@eecg.toronto.edu).

Digital Object Identifier 10.1109/TCAD.2005.859507

where the output is a polynomial function of its inputs, rational filters, where the output is a ratio of two input polynomials, and adaptive filters, where the output is a weighed combination of inputs, and the weights themselves are functions of inputs. The readers are referred to books such as [5] for a detailed comprehensive treatment.

In this paper, we also propose a comprehensive solution for the difficult problem of dynamic-range estimation for nonlinear systems. The key idea of our solution is the introduction of a new statistical construct, called polynomial chaos (PC), which is a family of orthogonal polynomials in terms of Gaussian RVs. With a rigorous well-defined procedure, these polynomials are constructed with the powerful properties such that when an input signal is decomposed into a combination of PC, called PC expansion (PCE), it is easily propagated through a nonlinear system, and the full statistics of all signals can be derived accordingly. Intuitively, the accuracy of this method can be attributed to the fact that KLE can effectively capture the correlation of signals, and the further use of polynomials can effectively preserve the correlation even after nonlinear operations.

We demonstrate the following advantages over previous approaches. First, it is extremely fast compared to profiling, the only method with comparable accuracy. While profiling has to simulate the system for thousands or even millions of random input realizations, our method only needs a single run of a program transformed from the original system specification. Second, our method is constructive in the sense that the distribution of all signals can be obtained. This is in sharp contrast to other analytical methods that computes intervals or finite-order moments, which are only partial information about the signals. As a result, an optimal bitwidth can be synthesized by aggressively cutting the tails of the distribution under the given signal-to-noise ratio (SNR) constraint. In contrast, other recent analytical approaches, such as that in [3], can offer only a verification of error bound of a given bitwidth selection, while shedding no insight on the proper choice of bitwidth. Third, our method is highly accurate. It can capture not only the spatial correlation, but also the temporal correlation in the system input, which is not possible in the previous methods. As is shown later in this paper, this contributes significantly to the accuracy of the result. With our method, the accuracy can be maintained even for nonlinear systems, which is not the case for previous methods [2]–[4]. Finally, it is the first time that the complete treatment of nonlinear systems are given. In contrast, while the propagation of different signal representations through nonlinear operations are discussed in [2] and [3], no results on a complete nonlinear system have been demonstrated.

The remainder of this paper is organized as follows. Section II gives a brief account of the related literature. We then describe our treatment of linear systems in Section III and nonlinear systems in Section IV, with the following structure: For each case, we first introduce the underlying theoretical background for the modeling of system signals: KLE for linear systems (Section III-A) and PCE for nonlinear systems (Section IV-A). Assuming such models for system inputs are available, we then show how such models for system responses (Sections III-B and IV-B) and, in turn, their statistics (Sections III-C and IV-C) can be derived. To make the picture complete, we also show how models for system inputs can be obtained from either statistics or sample data supplied by the user (Sections III-D and IV-E). Detailed experimental results are shown in Section V before we draw conclusion in Section VI.

A word about notation would be useful. We will use bold font, such as $\mathbf{x}$, to denote an RV. Deterministic (i.e., nonrandom) variables will be denoted with regular italic font, such as x. In addition, some Greek symbols, such as ξ_i, μ_i, and Ψ_i, will be used to denote RVs, but sometimes Greek symbols like λ_i will denote deterministic variables. It will be clear from the context which variables are random and which are not. The notation $E[\mathbf{x}]$ will denote the mean or expected value of the RV $\mathbf{x}$ [6]. Two RVs $\mathbf{x}$ and $\mathbf{y}$ are said to be orthogonal if $E[\mathbf{xy}] = 0$. Two zero-mean RVs $\mathbf{x}$ and $\mathbf{y}$ are said to be orthonormal if they are orthogonal and each has a variance of one.

II. Related Work

Profiling or simulation-based approach [7]–[10] is used extensively to estimate the dynamic range of variables in a program. As mentioned earlier, this method is computationally expensive, since huge amounts of sample data need to be simulated.

The L_p norm method based on a transfer function is proposed in [11] and is applied in [12]. While theoretically well formulated, this method requires the explicit knowledge of the system transfer function, which may not be always available, and which may be difficult to extract from C code. In addition, it propagates only the maximum value of the data. Thus, the estimated dynamic range can be very conservative. It is also important to note that the L_p norm method is not applicable to nonlinear systems due to their absence of transfer functions.

A moment-based method is presented in [2]. This method models the input as an RV and propagates up to the seventh order of its moments through the system. The probability density function (pdf) of all variables can be constructed from the propagated moments. However, this method assumes that the input data is temporally independent, and that variables internal to the system (such as at the input to an arithmetic operator) are spatially independent. Both these assumptions are not true in practice, and can have significant impact on accuracy of the results, as we will demonstrate in Section V-C.

A bitwidth/interval propagation method is adopted in [1]. This method propagates the input bitwidth or interval through the system to obtain the dynamic range for intermediate variables. The estimated result from this method is overly pessimistic, since it always considers the worst case. If the system is large, these estimation errors could be accumulated rapidly to a large amount.

Affine arithmetic, an improvement on the interval propagation technique, is given in [3] and [4], where the spatial correlation between internal variables induced by the reconvergent fan-out structure in the system is taken into account. However, this method does not capture the inherent temporal correlation of the input signals. For many applications, spatial correlation

between variables is a direct result of the temporal correlation of the input signals. Therefore, ignoring the temporal correlation of the input automatically causes loss of true correlation information internally. In addition, because the information propagated is limited to affine, or linear form, accuracy has to be compromised when nonlinear terms are generated after nonlinear operations. As detailed later in Section V-C, the loss of accuracy can be significant.

Thus, all existing methods have certain shortcomings, especially in the way that correlation and nonlinearity are dealt with. In our paper, we take care of both temporal input correlation and spatial internal correlation even for nonlinear systems, and we provide not only ranges of the data, but its statistical distribution as well.

III. Linear Systems

Consider an application that can be modeled as a linear system with a single-input data channel. We will focus on this case in this paper. However, it is straightforward to extend the proposed method to linear systems with multiple input channels. In order to model the unknown input data, and since the size of the input space is typically huge, we will model the input data stream as a discrete-time random process, i.e., a sequence of RVs that are presented at the system input at discrete time steps. Correspondingly, the system internal state and output variables all become random processes. Given an input process model, the dynamic-range-estimation problem can be formulated as the determination of the statistics of the random processes corresponding to the system state and output variables. Typically, the variance may be sufficient to determine dynamic range. However, we will show that we are able to estimate the full pdf, if needed.

A. KLE

A random process $\mathbf{p}(t)$ defined over $[0, t_0]$ with zero mean and autocorrelation function $R(t_1, t_2)$, can be expressed using the following KLE [6]:

$$\mathbf{p}(t) = \sum_{i=0}^{\infty} \sqrt{\lambda_i} f_i(t) \mu_i \tag{1}$$

where $f_i(t)$ and λ_i are referred to as the eigenfunctions and eigenvalues, respectively, of the autocorrelation function. The eigenfunctions are also known to be orthonormal, i.e.,

$$\int_0^{t_0} f_i(t) f_j(t) dt = \delta_{ij} \tag{2}$$

where δ_{ij} is the Kronecker delta function

$$\delta_{ij} = \begin{cases} 0, & \text{if } i \neq j \\ 1, & \text{if } i = j \end{cases} \tag{3}$$

and where μ_i's are a set of zero-mean orthonormal RVs, which means

$$E[\mu_i \mu_j] = \delta_{ij}. \tag{4}$$

When the random process $\mathbf{p}(t)$ is Gaussian (i.e., its one-dimensional (marginal) and all joint pdfs are Gaussian), a model that is often useful in practice, it turns out that μ_is are all independent standard normal RVs. They may be referred to as a Gaussian basis. In cases where $\mathbf{p}(t)$ is near normal, μ_i's can still be approximately treated as Gaussian RVs in practice. In cases of large deviation from the Gaussian distributions, the Gaussian basis is not appropriate and one can then determine the nature of the μ_i basis (their distribution type) using standard KL techniques. The details are unimportant because one often does not need to know the exact nature of the μ_i basis, but only their moments. For our paper, the second-order moments would actually seem to be sufficient, but we will show how higher order moments can be generated as well. The literature on KL techniques and their practical implementations is quite extensive. Notice that, irrespective of the distribution of μ_i, the orthonormality property (4) guarantees that

$$E\left[\mu_i^2\right] = 1 \tag{5}$$

and this fact will be useful later on to compute the variance, without having to know exactly the distribution of the μ_i.

It can be shown that the KLE is optimal in the sense that the mean square error resulting from replacing the infinite summation in (1) by a truncated finite summation is minimal. Finally, one can obtain $f_i(t)$ and λ_i by solving

$$\int_0^{t_0} R(t_1, t_2) f_i(t_1) dt_1 = \lambda_i f_i(t_2). \tag{6}$$

The above results are applicable to a continuous-time process. If we consider the discrete-time random process $\mathbf{p}[k]$ to be defined on the discrete time domain $[0, n]$, then, as was done in [13], a KLE can be expressed in discrete time as

$$\mathbf{p}[k] = \sum_{i=0}^{n} \sqrt{\lambda_i} f_i[k] \mu_i, \qquad k = 0, 1, \ldots, n \tag{7}$$

where λ_i and $f_i[k]$ are the eigenvalues and eigenfunctions of the autocorrelation matrix of the discrete-time random process $\mathbf{p}[k]$

$$\begin{bmatrix} R(0,0) & R(0,1) & \cdots & R(0,n) \\ R(1,0) & R(1,1) & \cdots & R(1,n) \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & & \cdot \\ R(n,0) & R(n,1) & \cdots & R(n,n) \end{bmatrix} \begin{bmatrix} f_i[0] \\ f_i[1] \\ \cdot \\ \cdot \\ \cdot \\ f_i[n] \end{bmatrix} = \lambda_i \begin{bmatrix} f_i[0] \\ f_i[1] \\ \cdot \\ \cdot \\ \cdot \\ f_i[n] \end{bmatrix} \tag{8}$$

where $R(k_1, k_2) = E[\mathbf{p}[k_1]\mathbf{p}[k_2]]$ is the autocorrelation function. Here too, μ_i and $f_i[k]$ are orthonormal, and the summation can be truncated, yielding a least squares optimal expansion

$$\mathbf{p}[k] \approx \sum_{i=0}^{m} \sqrt{\lambda_i} f_i[k] \mu_i, \qquad k = 0, 1, \ldots, n. \tag{9}$$

In this case, it can be shown that the relative mean square error resulting from the truncation is given by

$$e = 1 - \frac{\sum_{i=0}^{m} \lambda_i}{\sum_{i=0}^{n} \lambda_i} \tag{10}$$

where $\lambda_0, \ldots, \lambda_m$ are the eigenvalues that are kept in the truncated KLE. We will refer to this error term as the truncation error of the KLE. In order to minimize error, the largest eigenvalues are always kept. Here, we assume λ_i's are ordered descendantly by magnitude.

B. Obtaining System Response

Let $\mathbf{u}[k]$ be the random process at the system input and $\mathbf{x}[k]$ be the random process at an arbitrary state variable or at a system output. Then, we can write

$$\mathbf{x}[k] = \mathcal{L}\left(\mathbf{u}[k]\right) \tag{11}$$

where $\mathcal{L}(\cdot)$ denotes the linear system operator that transforms $\mathbf{u}[k]$ to $\mathbf{x}[k]$. Let $\mathbf{u}[k]$ have the following KLE:

$$\mathbf{u}[k] = \sum_{i=0}^{m} \sqrt{\lambda_i} f_i[k] \mu_i = \sum_{i=0}^{m} u_i[k] \mu_i \tag{12}$$

where $u_i[k] = \sqrt{\lambda_i} f_i[k]$. Combining (11) and (12) gives

$$\mathbf{x}[k] = \mathcal{L}\left(\mathbf{u}[k]\right) = \mathcal{L}\left(\sum_{i=0}^{m} u_i[k] \mu_i\right). \tag{13}$$

By the superposition property of linear systems, it follows that

$$\mathbf{x}[k] = \mathcal{L}\left(\mathbf{u}[k]\right) = \sum_{i=0}^{m} \mathcal{L}\left(u_i[k]\right) \mu_i = \sum_{i=0}^{m} x_i[k] \mu_i \tag{14}$$

where $x_i[k] = \mathcal{L}(u_i[k])$.

In this way, we can obtain the KLE of the random process $\mathbf{x}[k]$ in terms of the system responses to each of the deterministic (nonrandom) functions $u_i[k]$ applied as input. In specific cases where a system transfer function is available, one can solve for $x_i[k]$ directly. However, in the more general case, and this is the approach that we take, even when the system is specified with a high-level behavioral description such as a C program, $x_i[k]$ can be obtained by simply simulating the system (e.g., executing the C program) with $u_i[k]$ as input. The order m of the KLE is typically small, as we will show in Section V, so that the complete statistics of the system internals and outputs may be obtained by simulating the system m times. The initial state of the simulation and the length of the simulation period are parameters that can be set depending on the particular situation. For example, in order to study the dynamic range during a system transient, the simulations can be performed starting from any desired initial state, and the largest variance of the resulting responses may be monitored. If the steady-state dynamic range is of interest, then the simulation period must be set long enough for the individual simulations of the components $u_i[k]$ to reach steady state. This, to some extent, depends on the statistics of the input process and on the system dynamics. Thus, a KLE is not a magic remedy that eliminates the need for simulation. Instead, KL is a way to drastically reduce the number of required simulations, compared to profiling-based methods, as we will demonstrate in Section V.

C. Obtaining System Statistics

Once the required m simulations are complete, we can assemble the complete KLE for any system variable or output response $\mathbf{x}[k]$ as

$$\mathbf{x}[k] = \sum_{i=0}^{m} x_i[k] \mu_i. \tag{15}$$

It is important to note that this is a complete statistical description of the process $\mathbf{x}[k]$. One can use this expansion to compute any desired probability associated with $\mathbf{x}[k]$, such as the probability that it would exceed a certain threshold value, or simply compute the overall distribution of $\mathbf{x}[k]$ from the known distributions of the μ_i RVs. Oftentimes, the moments of the distribution are very useful, and they are perhaps easiest to compute from this expansion. The first-order moment is the mean or expected value of the process, which is known to be zero because the input is zero mean. The second-order moment, which is required to compute the variance, is given by

$$E\left[\mathbf{x}[k]^2\right] = E\left[\left(\sum_{i=0}^{m} x_i[k] \mu_i\right)^2\right] = \sum_{i=0}^{m} x_i[k]^2 \tag{16}$$

which is true because μ_i have zero mean and unity variance [due to (5)].

If the system input is Gaussian, then all the internal and output responses are also Gaussian, so that the mean and variance are sufficient to capture the complete distribution. In the general case, higher order (say l-order) moments may be required and they may be obtained by similar, although slightly more involved, expansions that turn out to require terms such as $E[\mu_{j_1}^{l_1} \mu_{j_2}^{l_2} \cdots \mu_{j_q}^{l_q}]$, where $l_1 + l_2 + \cdots + l_q = l$. These terms may be obtained by using the KL transformation (9) in the reverse direction to compute samples of the RVs μ_i from the input data samples, and using these samples to compute terms like $E[\mu_{j_1}^{l_1} \mu_{j_2}^{l_2} \cdots \mu_{j_q}^{l_q}]$ by simple (Monte Carlo) averaging. The process is fairly straightforward but is omitted for brevity. Once the moments of the response $\mathbf{x}[k]$ are obtained, one can further estimate its pdf by any of the pdf expansion methods such as Gram–Charlier, Hermite, or Edgeworth expansion [14], [15], or by pdf fitting methods such as generalized lambda distribution [16]. These pdf estimation methods have been extensively applied in many research areas.

D. Input KLE-Model Extraction

The proposed method needs the KLE model for the primary input in the first place. The model can be extracted either from the input statistics or sample data supplied by the user.

The input statistics can be correlation functions, such as the $R(k_1, k_2)$ matrix in (8), and other moments of the input. If the input random process is Gaussian or nearly Gaussian, its mean

and correlation function are sufficient to compute the statistics of the system variables and outputs. In the general case, other moments may also be needed depending on the order of the required system variable statistics.

If input statistics are not available and sample data traces are provided instead, we first extract the mean and correlation matrix from the sample data. Extracting the mean is done by simple averaging. The correlation matrix is obtained by averaging the cross terms as follows:

$$\frac{1}{m}\begin{bmatrix} p_1[0] & p_2[0] & \cdots & p_N[0] \\ p_1[1] & p_2[1] & \cdots & p_N[1] \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & & \cdot \\ p_1[n] & p_2[n] & \cdots & p_N[n] \end{bmatrix} \begin{bmatrix} p_1[0] & p_1[1] & \cdots & p_1[n] \\ p_2[0] & p_2[1] & \cdots & p_2[n] \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & & \cdot \\ p_N[0] & p_N[1] & \cdots & p_N[n] \end{bmatrix} \tag{17}$$

where $[p_i[0]\ p_i[1]\ \cdots\ p_i[n]]$ is a sample trace input, and N is the total number of sample traces.

If the mean is not zero, then the input process is decomposed into a deterministic term whose value is equal to the mean, and another zero-mean random process resulting from subtracting the mean from the original process. The corresponding mean of every system response can be computed by one execution of the system behavior using the input mean function as excitation. Since the mean can be easily subtracted out up front, and the response to the mean function easily added later on, it is sufficient to focus the discussion on the zero-mean case.

Once the correlation matrix is available, we can extract the KLE model by solving the eigensystem problem for this matrix. Solution techniques of such systems are standard. After the eigenvalues and eigenfunctions are found, the KLE of the input random process is available. It can easily be truncated according to the truncation error specified by the user using (10). High-order moments (higher than second order) of the RVs μ_j can also be computed from the moments of the input random process if they are needed (as mentioned in the previous section).

IV. Nonlinear Systems

The behavior of nonlinear DSP applications can be captured at a high level of abstraction with a data-flow graph (DFG). Each node of the DFG represents a primitive operation, typically a multiplication or addition of real numbers, or a decision operation. Note that with a decision operation, the if–then–else branch that is typically captured in a control-flow graph (CFG) is implicitly captured in the DFG. The construction procedure for such a DFG is standard and has been reported in both the compiler [17] and CAD [18] communities.

As in the case of linear systems, we view the primary inputs of the DFG as random. The "source of randomness" in the system primary input is either a genuine uncertainty about what inputs to expect or a representation of a large population of possible inputs by means of their aggregate statistics. Since the principle of superposition can no longer be exploited here, we propose to use the PCE in place of KLE to model the system signals. The analysis can then be formulated as propagating the PCE coefficients through the DFG from the system primary inputs.

A. PCE

We first provide a brief review of the theory of PC and describe the PCE of an RV, based mainly on [19].

Let $\xi_1, \xi_2, \ldots, \xi_n$ be a sequence of zero-mean orthonormal Gaussian RVs. In other words, each ξ_i is Gaussian with mean zero and variance one (i.e., each has the standard normal distribution), and they are orthogonal, so that

$$E[\xi_i \xi_j] = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

It can be shown that ξ_i's are also independent. Let $\Gamma_p(\xi_1, \xi_2, \ldots, \xi_n)$ be a polynomial whose degree is p, defined on the orthonormal Gaussian RVs $\xi_1, \xi_2, \ldots, \xi_n$. If we consider different permutations and replacements of the arguments of $\Gamma_p(\cdot)$, which has at most p distinct arguments, we represent the resulting set of polynomials by

$$\{\Gamma_p(\xi_{i_1}, \xi_{i_2}, \ldots, \xi_{i_p})\} \tag{19}$$

where $i_k \in \{1, 2, \ldots, n\}$ for all $k \in \{1, 2, \ldots, p\}$. This set of polynomials is referred to as PC of order p. To be exact, PC is not just any set of polynomials; rather, it is a set with an important orthogonality property, defined by means of the following construction.

The zeroth-order PC $\{\Gamma_0(\cdot)\}$ is a constant, and is chosen to be one by construction

$$\Gamma_0() = 1. \tag{20}$$

The first-order PC is chosen so that every polynomial in $\Gamma_1(\xi_i)$ is orthogonal to every polynomial in $\{\Gamma_0(\cdot)\}$, where two distinct polynomials $\Gamma_q(\xi_{i_1}, \ldots, \xi_{i_q})$ and $\Gamma_r(\xi_{j_1}, \ldots, \xi_{j_r})$ are said to be orthogonal if

$$E\left[\Gamma_q(\xi_{i_1}, \ldots, \xi_{i_q})\Gamma_r(\xi_{j_1}, \ldots, \xi_{j_r})\right] = 0, \qquad \text{if } q \neq r. \tag{21}$$

Applying this orthogonality requirement leads to the following choice for the first-order PC:

$$\Gamma_1(\xi_i) = \xi_i. \tag{22}$$

The second-order PC is chosen so that it is orthogonal to both Γ_0 and Γ_1, leading to

$$\Gamma_2(\xi_{i_1}, \xi_{i_2}) = \xi_{i_1}\xi_{i_2} - \delta_{i_1 i_2} \tag{23}$$

where $\delta_{i_1 i_2}$ is the Kronecker delta as defined earlier.

Similar procedures give all higher order PCs, so that each is orthogonal to all lower order PCs. What is not obvious from the brief and informal presentation above is that even within a single $\{\Gamma_p\}$, it can be shown that the polynomials are all orthogonal. The concept of PC is useful in that it provides a basis for decomposition of a general RV, called a PCE, as follows.

If $\mathbf{x}$ is a square integrable RV, that is, $E[|\mathbf{x}|^2]$ is finite, then it can be shown that one can express it as a decomposition or

TABLE I
PC POLYNOMIALS $\Psi_0, \ldots, \Psi_{14}$ FOR $n = 2$ AND $p = 0, 1, 2, 3, 4$, WITH $E[\Psi_i^2]$ IN EACH CASE

Index	Order	Polynomial	$E[\Psi_i^2]$
0	0	1	1
1	1	ξ_1	1
2		ξ_2	1
3		$\xi_1^2 - 1$	2
4	2	$\xi_1\xi_2$	1
5		$\xi_2^2 - 1$	2
6		$\xi_1^3 - 3\xi_1$	6
7	3	$\xi_1^2\xi_2 - \xi_2$	2
8		$\xi_1\xi_2^2 - \xi_1$	2
9		$\xi_2^3 - 3\xi_2$	6
10		$\xi_1^4 - 6\xi_1^2 + 3$	24
11		$\xi_1^3\xi_2 - 3\xi_1\xi_2$	6
12	4	$\xi_1^2\xi_2^2 + \xi_1^2 - \xi_2^2 + 1$	4
13		$\xi_1\xi_2^3 - 3\xi_1\xi_2$	6
14		$\xi_2^4 - 6\xi_2^2 + 3$	24

expansion in terms of an underlying basis of PC. The number n of underlying RVs $\xi_1, \ldots, \xi_n$ is referred to as the dimension of the expansion, and, as above, the highest degree in the polynomials p is called the order. The exact PCE generally uses infinite dimension (n) and order (p). However, in practice, one can truncate the infinite series expansion based on a finite n and p, by neglecting high order terms and minor dimensions. Thus, the two-dimensional $(n = 2)$ PCE with order two $(p = 2)$ is

$$\mathbf{x} = a_0\Gamma_0 + a_1\Gamma_1(\xi_1) + a_2\Gamma_1(\xi_2) + a_{11}\Gamma_2(\xi_1, \xi_1) + a_{12}\Gamma_2(\xi_1, \xi_2) + a_{22}\Gamma_2(\xi_2, \xi_2) \quad (24)$$

where a_{ij}'s are constant coefficients. $\Gamma_2(\xi_2, \xi_1)$ is dropped here since it is identical to $\Gamma_2(\xi_1, \xi_2)$. In order to simplify the notation, the polynomials are sorted in a specific way and they are denoted $\Psi_0, \Psi_1, \Psi_2, \ldots, \Psi_m$; this is nothing more than a different numbering scheme. Given an order (p) and a dimension (n), one can quickly determine [19] exactly which polynomial each Ψ_i corresponds to under this numbering scheme. For example, Table I shows an example of the numbered polynomials for the two-dimensional case. With this simplified notation, each RV can be expressed as

$$\mathbf{x} = \sum_{i=0}^{m} x_i\Psi_i. \quad (25)$$

In general, m grows very quickly for larger n and p, but a PCE is typically useful even for small order and dimension, so that m remains tractable in practice. It can be proved that the number of n dimension PCs up to pth order $m + 1$ is

$$m + 1 = \sum_{i=0}^{p} \frac{(n + i - 1)!}{i!(n - 1)!} = \frac{(n + p)!}{n!p!} \quad (26)$$

where $(n + i - 1)!/i!(n - 1)!$ is the number of i-order PCs.

The key property of these polynomials that turns out to be very useful in practice is the following orthogonality property

$$E[\Psi_i\Psi_j] = \begin{cases} E\left[\Psi_i^2\right], & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}. \quad (27)$$

The values of $E[\Psi_i^2]$ can be computed easily based on the fact, which is easily proven, that if ξ is a zero-mean unity-variance Gaussian, then for any integer $k \geq 0$

$$E[\xi^{2k+1}] = 0 \quad \text{and} \quad E[\xi^{2k}] = \frac{(2k)!}{2^k k!}. \quad (28)$$

From above, it becomes possible to apply PCE to complex nonlinear systems of differential equations, such as in computational fluid dynamics [20], where random parameter perturbations have typically been expressed using PCE. As a result, an analysis of a system under random stimulus is replaced by repeated deterministic analyses of the system, from which coefficients of the PCE are solved for separately. Once the PCE coefficients are known, the distribution, the moments, or any statistics of the system response can be solved for.

B. Obtaining System Response

In the following, we will show how the output PCE can be derived from the input PCE, for various types of operations that one typically encounters in DFGs.

1) Scaling: This case is trivial. If $\mathbf{y} = a\mathbf{x}$, where a is a constant, and if a PCE is available for $\mathbf{x}$, so that $\mathbf{x} = \sum_{i=0}^{m} x_i\Psi_i$, then the PCE for $\mathbf{y}$ is $\mathbf{y} = \sum_{i=0}^{m} y_i\Psi_i$, where

$$y_i = ax_i. \quad (29)$$

2) Summation: Here too, the linearity of the summation operation makes this case very easy. If $\mathbf{z} = \mathbf{x} + \mathbf{y}$, and if a PCE expansion is considered for all three variables

$$\mathbf{x} = \sum_{i=0}^{m} x_i\Psi_i, \quad \mathbf{y} = \sum_{i=0}^{m} y_i\Psi_i, \quad \mathbf{z} = \sum_{i=0}^{m} z_i\Psi_i \quad (30)$$

then it is clear that

$$z_i = x_i + y_i. \quad (31)$$

3) Multiplication: The multiplication case is nontrivial, and it brings out the power of the PCE expansion. Suppose $\mathbf{z} = \mathbf{xy}$ and that a PCE expansion is considered for all three variables, so that

$$\sum_k z_k\Psi_k = \left(\sum_i x_i\Psi_i\right)\left(\sum_j y_j\Psi_j\right) = \sum_{i,j} x_iy_j\Psi_i\Psi_j. \quad (32)$$

In order to solve for z_k, we multiply both sides of the above equation by Ψ_k and take the expected value of both sides. Based on the orthogonality property (27), this leads to

$$z_k = \frac{1}{E\left[\Psi_k^2\right]} \sum_{i,j} x_iy_jE[\Psi_i\Psi_j\Psi_k]. \quad (33)$$

The three-way expectations on the right-hand side are actually trivial to compute for a given dimension and order of PCE, because of the independence of the ξ_i variables and due to (28).

As an example, in the case when the dimension is 1 ($n = 1$) and the order is 2 ($p = 2$), in which case the expansion consists of Ψ_0, Ψ_1, and Ψ_2 only, this type of analysis leads to

$$\begin{aligned} z_0 &= x_0y_0 + x_1y_1 + 2x_2y_2 \\ z_1 &= x_0y_1 + x_1y_0 + 2x_1y_2 + 2x_2y_1 \\ z_2 &= x_0y_2 + x_2y_0 + x_1y_1 + 4x_2y_2. \end{aligned} \tag{34}$$

Notice that there is no assumption of independence between $\mathbf{x}$ and $\mathbf{y}$. Indeed, the PC expansion offers a way by which correlations among different RVs can be captured via the coefficients of the various polynomials, so that correlation is taken into account quite naturally. For larger values of n and p, the expressions for z_i become more involved but they remain of this nature, meaning they consist of cross products of the x_i and y_j terms. If we go to large problems such as $n = 4$ and $p = 3$, these expressions can grow to include about 35 terms.

4) Division: Division being the inverse of multiplication, it is easy to see that equations such as (34) can be used in the reverse direction, to give values of x_i, given values of z_i and y_i for example. In the reverse direction, these equations constitute linear systems of simultaneous equations $Ax = b$ whose solution techniques (i.e., finding the inverse matrix of A) are standard. For example, in the case when $n = 1$ and $p = 2$, shown above, the resulting matrix equation is

$$\begin{bmatrix} y_0 & y_1 & 2y_2 \\ y_1 & y_0 + 2y_2 & 2y_1 \\ y_2 & y_1 & y_0 + 4y_2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} z_0 \\ z_1 \\ z_2 \end{bmatrix}. \tag{35}$$

For higher dimension and order, the equations remain linear, but the size of the matrix and the values of the matrix entries would change.

5) Multiplexing: We consider the case where the value of a variable depends on a "switch" or multiplexing decision based on another variable. Specifically, if $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{x}$ are RVs, and if (c_0, c_1) is a partition of the domain of $\mathbf{c}$, then we consider

$$\mathbf{x} = \begin{cases} \mathbf{a}, & \text{if } \mathbf{c} \in c_0 \\ \mathbf{b}, & \text{if } \mathbf{c} \in c_1 \end{cases}. \tag{36}$$

As before, we consider that each variable has a PCE, so that

$$\sum_i x_i\Psi_i = \begin{cases} \sum_i a_i\Psi_i, & \text{if } \mathbf{c} \in c_0 \\ \sum_i b_i\Psi_i, & \text{if } \mathbf{c} \in c_1 \end{cases}. \tag{37}$$

If we define a new set of RVs $\mathbf{x_i}$ by

$$\mathbf{x_i} = \begin{cases} a_i, & \text{if } \mathbf{c} \in c_0 \\ b_i, & \text{if } \mathbf{c} \in c_1 \end{cases} \tag{38}$$

then we can write

$$\sum_i x_i\Psi_i = \sum_i \mathbf{x_i}\Psi_i. \tag{39}$$

If we multiply both sides by Ψ_k and take the expected value of both sides, and applying the orthogonality property, this leads to

$$x_k = \frac{1}{E[\Psi_k^2]} \sum_i E[\mathbf{x_i}\Psi_i\Psi_k]. \tag{40}$$

We can further simplify the result by using conditional expectations

$$\begin{aligned} E[\mathbf{x_i}\Psi_i\Psi_k] = a_i E[\Psi_i\Psi_k|\mathbf{c} \in c_0]\,\mathcal{P}\{\mathbf{c} \in c_0\} \\ + b_i E[\Psi_i\Psi_k|\mathbf{c} \in c_1]\,\mathcal{P}\{\mathbf{c} \in c_1\}. \end{aligned} \tag{41}$$

$\mathcal{P}\{\mathbf{c} \in c_i\}$ denotes the probability of random event $\mathbf{c} \in c_i$, while $E[\Psi_i\Psi_k|\mathbf{c} \in c_i]$ denotes the conditional expectation under condition $\mathbf{c} \in c_i$. The conditional Expectations can be easily and quickly estimated by Monte Carlo techniques. Random samples of $\xi_1, \ldots, \xi_n$ are generated and classified as to whether they produce a sample of $\mathbf{c}$ in c_0 or c_1. The probabilities of the two outcomes are thus computed, and within each classification, the mean value of $\Psi_i\Psi_k$ is computed in the usual Monte Carlo fashion. Since the underlying variables are Gaussian, convergence is easily achieved.

6) Time Shift: The preceding operations were operations on RVs, without regard to the time dimension. In general, a behavioral description consists of a sequencing of the operations in time, so that the RVs considered are really signal values at specific time points on a discrete time scale: $\mathbf{x}[1], \mathbf{x}[2], \ldots$. The preceding analysis for scaling, summation, multiplication, division, and multiplexing were all relevant to a specific time point. In addition, it is trivial to handle a time-shift operation: if

$$\mathbf{y}[j] = \mathbf{x}[j - k] \tag{42}$$

then the resulting PCE coefficients are simply time shifted themselves

$$y_i[j] = x_i[j - k]. \tag{43}$$

C. Obtaining System Statistics

Once the PC expansion for an RV is available, one can use that PCE to compute statistics of various kinds for the variable. Moments of arbitrary order (mean, variance, etc.) can be easily generated [19]. Analytical approximations to the full distribution of that RV can then be obtained from the moments, using techniques such as the Edgeworth expansion [15] and others. This is the same as what we mentioned earlier for the KLE method.

For determining dynamic range and choosing the right bitwidth, it is important to have the cumulative distribution function (cdf), at least the tail of the cdf, in order to estimate the error incurred if bitwidth is reduced. Given that we have in hand a PCE for an RV, we have found it is very easy and practical to estimate the cdf by Monte Carlo techniques. Given a PCE expansion for an RV, we generate samples of the orthonormal basis $\xi_1, \ldots, \xi_n$ from which we obtain samples of the value of the RV itself, which we use to build an empirical approximation

to the cdf $F(x)$. It typically takes less than a second of CPU time to compute the cdf in this way.

D. Applicability

We have shown how a PCE can be propagated through typical operations found in a system and how system statistics can be obtained from the output PCE. While we have effectively defined the PCE "transfer functions" for these operations, what remains is the question as to how general can the scheme be applied. For given input PCEs, we could ask the following questions: First, do response PCEs exist for any system? Second, do PCEs exists for all operations we use to compose systems of interest? Third, can response PCEs be found for systems specified as network of basic operators, with structures including feedback?

It has been well established [19] that PCE representation can always be found and can be arbitrarily accurate for an RV that is square integrable (or its second moment exists). Note that this condition applies to most practical systems; in other words, their response PCEs exist. In particular, if a system/operation produces a bounded output, then the output is automatically square integrable.

We now consider a system composed of a network of elementary operations. Assuming that the PCE propagation formulas of all operations are known, assuming that the input PCEs for each time point are given, and assuming that the initial PCEs of state variables are given either as deterministic values or as random PCEs, we can find the system transient response, in the form of one PCE per time point, by applying the PCE propagation formulas of all operations in topological order at each time point. In the case where the network is cyclic due to the feedback structures, the cycles are first broken at delay elements, after which the topological application can be followed. This procedure is exact provided that the PCE order and dimension are infinite.

In practice, PCE truncations have to be used to keep the order and dimension finite, which inevitably introduce errors or noise. For feedback systems, it is possible that the error may become unbounded as it propagates through the system. It is well known that the "butterfly effect" exists in chaotic systems found in nature. However, well-designed engineering systems of interest are also stable and robust; in other words, the system itself tolerates small errors both in the system input and the system model, and we can approach the exact response as close as we want by reducing these errors to certain ranges. As we observe in experiments, errors in general can be dealt with by increasing PCE order and dimension. It is also important to note that feedback structures introduce correlation, for which the PCE method is particularly capable of capturing.

Since the same hardware resource will be used for system outputs at different time points, the response PCEs for all time points need to be combined for the purpose of dynamic-range estimation. It is also important to note that most practical systems are stationary; in other words, their statistics will converge after a finite number of iterations, in which case we can stop the PCE propagation process. It is up to the user whether only the steady state is important, in which case the converged PCE can be used, or the transients are important as well, in which case all the PCEs up to the steady state need to be combined. This can be achieved by using a method similar to the treatment of a multiplexor operation.

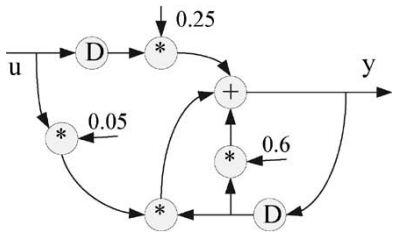


Fig. 1. Nonlinear DFG.

TABLE II
PCE COEFFICIENTS OF RECURSIVE NONLINEAR FILTER

Iter.	Ψ_0	Ψ_1	Ψ_2	Ψ_3	Ψ_4
1	0.002	0.004	0.002	0.002	0.001
4	0.00937	0.0188	0.00972	0.00963	0.00488
8	0.0112	0.0226	0.012	0.012	0.00603
12	0.0115	0.0231	0.0123	0.0121	0.00622
16	0.0115	0.0232	0.0124	0.0122	0.00625

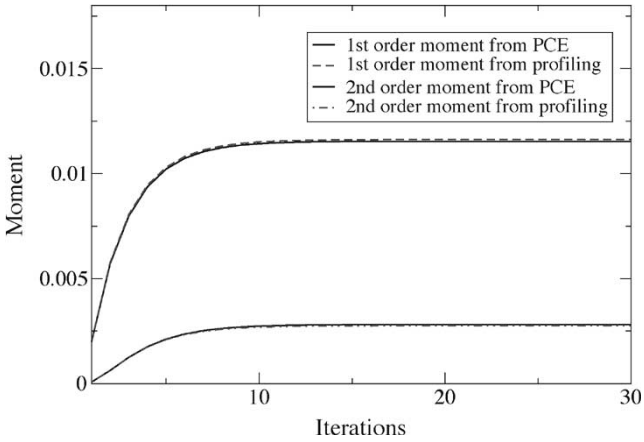


Fig. 2. Output moments of recursive nonlinear filter.

Consider a recursive nonlinear filter example shown in Fig. 1, which contains feedback structures. The D elements in the DFG denote time shift that causes one time unit delay. Assuming that the system input u is stationary and can be captured by a one-dimensional four-order PCE at all time points

$$u = 0.01\Psi_0 + 0.02\Psi_1 + 0.01\Psi_2 + 0.01\Psi_3 + 0.005\Psi_4. \tag{44}$$

By utilizing the PCE propagation method section, we are able to compute PCE for the DFG output y. Table II shows the PCE coefficients of y computed at different iterations. After 16 iterations, the PCE of y converges to steady status. The convergence process can be clearly shown in Fig. 2, where the first- and second-order moments are plotted for each iteration. Fig. 3 compares the cdf of y from the PCE with that produced by profiling with 30 000 samples. The two cdf plots match very well.

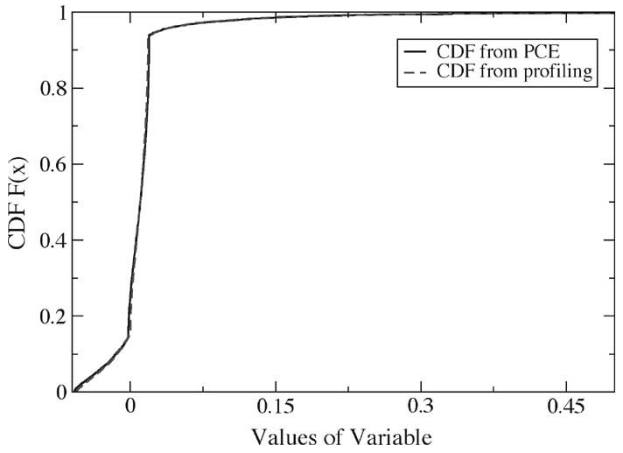


Fig. 3. Output cdf of recursive nonlinear filter.

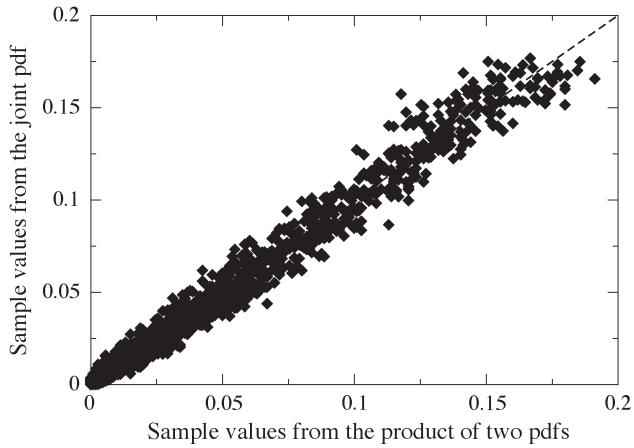


Fig. 4. Checking the independence assumption.

E. Input PCE Model Extraction

In order to propagate the PCE through a DFG, based on the above operations, one needs a PCE expression for the system input to begin with. We assume that the system has a single-input data channel, represented with a stochastic process $\mathbf{p}[k]$, $k = 0, 1, \ldots, N$. The extension to multiple inputs is not hard to do, but we will focus on the single-input case for clarity. At each time index k, the value $\mathbf{p}[k]$ is an RV. We will show how a KLE [6] for the stochastic process $\mathbf{p}[k]$ can be used to generate an input PCE expansion.

If μ_i is a Gaussian process, it can be shown that μ_i's are a set of independent standard Gaussian RVs (7). If $\mathbf{p}[k]$ is not Gaussian, then one can only guarantee that μ_i is an orthonormal set (4). Therefore, μ_i's are uncorrelated to each other, not necessarily independent. Uncorrelatedness is weaker than independence.

For a Gaussian random process, after KLE, it is already in the correct format of a PCE since the independent standard Gaussian RVs μ_i are exactly the first-order PC. No extra work is needed.

1) Nearly Gaussian Case: For a "nearly Gaussian" random process, even though RVs μ_i are only uncorrelated, strictly speaking, it is possible to approximate them as being independent and further expand every μ_i individually by PCE. The error caused by this assumption is tolerable since $\mathbf{p}[k]$ only slightly diverges from the Gaussian one.

From basic probability theory, it is known that if $\mathbf{x}$ is any RV, and if $F(\cdot)$ is its cdf, then $\mathbf{y} = F(\mathbf{x})$ is another RV whose cdf can be shown to be the uniform distribution on [0, 1]. As a corollary, if $\mathbf{u}$ is an RV that is uniform on [0, 1], and if $F(\cdot)$ is a valid cdf, then $\mathbf{x} = F^{-1}(\mathbf{u})$ is an RV with the cdf $F(\cdot)$. Let $\Phi(\cdot)$ be the cdf of the standard normal (zero mean, unity variance, Gaussian) distribution, and let $F_i(\cdot)$ be the cdf of μ_i (resulting from KLE). Therefore

$$\xi_i = \Phi^{-1}\left(F_i(\mu_i)\right) \tag{45}$$

is a standard normal RV (Gaussian with mean zero and variance one). We propose to use the resulting $\xi_1, \ldots, \xi_n$ as the basis for a PCE. The advantage in doing this is that we would have an easy way to compute the required PCE coefficients from the KLE for each μ_i. Before we can do that, however, we must first establish that the ξ_i's are orthonormal. Since they are standard normal RVs, orthonormality would be established if we could prove that they are pairwise uncorrelated so that $E[\xi_i\xi_j] = E[\xi_i]E[\xi_j] = 0$. However, uncorrelatedness is not guaranteed by the above construction of the ξ_i's.

One easy way to achieve uncorrelatedness is if it were the case that the μ_i's are independent. That would lead to the conclusion that the ξ_i's are independent, due to the construction (45), and therefore uncorrelated. However, KLE guarantees that μ_i's are independent only in the case when the process $\mathbf{p}[k]$ is itself Gaussian. Otherwise, KLE only guarantees that the μ_i's are uncorrelated, which is not enough to establish that the ξ_i's are uncorrelated.

If the input random process is not too far from the Gaussian model, we will assume that μ_i can be approximated as being independent. As a sanity check, we have checked the error of this approximation in the following way. We generated 80 000 random sample traces (realizations) of $\mathbf{p}[k]$, using a typical auto-regression moving average (ARMA) model of time series [21], which is chosen to be quite close to the Gaussian model. We then built the KLE (7) for $\mathbf{p}[k]$, so that we end up with 80 000 random samples of the vector $[\mu_1\mu_2\cdots\mu_n]$. This allows us to check on whether the joint distribution of μ_i and μ_j can be approximated as the product of their individual (marginal) distributions. If yes, then μ_i and μ_j may be assumed independent. The distributions were built from the data by simply approximating the pdf with a histogram (80 000 is a large-enough data set that this is very well justified). We show the results of this comparison in Fig. 4, where, for every observed (μ_i, μ_j) pair of values, the vertical axis shows the joint pdf of μ_i and μ_j evaluated at that point, $f_{\mu_i,\mu_j}(\mu_i, \mu_j)$, and the horizontal axis shows the product $f_{\mu_i}(\mu_i)f_{\mu_j}(\mu_j)$. Ideally, the data should be right on the diagonal line, but it is not, of course. However, it is close enough to the diagonal that we consider that this justifies the independence assumption.

With the ξ_i available as a basis, we can now proceed to obtain the PCE for $\mathbf{p}[k]$ as follows. For each μ_i, consider a one-dimensional PCE expansion, based on ξ_i, so that

$$\mu_i = \sum_j c_{ij}\Psi_j. \tag{46}$$

One can easily compute the values of the coefficients c_{ij}, as follows. To compute c_{ik}, multiply both sides of (46) by Ψ_k and take the expectation on both sides, leading to

$$c_{ik} = \frac{E[\mu_i \Psi_k]}{E\left[\Psi_k^2\right]}. \tag{47}$$

The right-hand side is a function of the statistics of μ_i and the ξ_i derived from it (recall that, in this one-dimensional PCE, Ψ_k is a function of only ξ_i). The denominator, for one thing, is very easy, and is available from the properties of the PC polynomials as we saw earlier in connection with (28). The expectation in the numerator can be computed by a simple Monte Carlo procedure, as follows. Given a large-enough number of user-supplied samples of the input process $\mathbf{p}[k]$ (or, given some probabilistic model of the input process, such an ARMA model, from which one can generate such samples), we can use the KLE (7) to generate samples of the μ_i's, from which the cdf of each μ_i can be approximated by its empirical cdf. The empirical cdf of an RV $\mathbf{x}$, evaluated at some value x_0, is given by the fraction of observed values of $\mathbf{x}$ that are less than or equal to x_0. Once these cdfs $F_i(\cdot)$ are available, we can use the construction (45) to get the corresponding sample values of the ξ_i's, from which Ψ_k can be computed, and the mean $E[\mu_i \Psi_k]$ computed by Monte Carlo.

2) Non-Gaussian Case: However, for a random process that is significantly non-Gaussian, the RVs μ_i cannot be treated as independent variables, otherwise, the resultant error is large, as we will show in Section V. Under this case, we will conduct a linear transformation on the random vector $\boldsymbol{\mu} = [\mu_0 \ \mu_1 \ \cdots \ \mu_n]^{\mathrm{T}}$ and the resulting transformed RVs will be mutually independent. The matrix defining this linear transformation can be found by independent component analysis (ICA). After we transform μ_i to a set of mutually independent RVs, the left work is exactly the same as the nearly Gaussian case. For the significantly non-Gaussian case, performing KLE is still necessary since KLE can effectively reduce the minor dimensions of the random process and is also needed by ICA as a preprocessing operation.

The problem of ICA is formulated as follows. Assume that we have a random vector $\boldsymbol{\mu} = [\mu_1 \ \mu_2 \ \cdots \ \mu_n]^{\mathrm{T}}$ where RVs $\mu_1, \mu_2, \ldots, \mu_n$ are not mutually independent. ICA tries to find an invertible square matrix $\mathbf{W}$, such that by performing the linear transformation

$$\mathbf{s} = \mathbf{W}\mu \tag{48}$$

the elements $(\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n)$ of the resulting random vector $\mathbf{s}$ are mutually independent or at least as independent as possible. Without loss of generality, it is typical to constrain the RVs $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n$ so that they each have a variance of one.

The main idea of the ICA method is: First, establish a quantitative metric of independence for random vector $\mathbf{s}$, and then find the matrix $\mathbf{W}$ that makes this independence metric reach its maximum or minimum value. Therefore, ICA is essentially an optimization problem.

It can be shown [22] that maximizing the independence of RVs $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n$, is equivalent to maximizing the sum of their negentropies: $\sum_i J(\mathbf{s}_i)$. Negentropy $J(\mathbf{s}_i)$ is a measure of non-Gaussianity of $\mathbf{s}_i$, which reaches its minimum value 0 if $\mathbf{s}_i$ is Gaussian. Therefore, in practice, we can make the RVs $\mathbf{s}_i$ independent by choosing a $\mathbf{W}$ to maximize their individual non-Gaussianity.

This fact can be intuitively explained by the central limit theorem. We want to discover a matrix $\mathbf{W}$ such that the uncorrelated RVs μ_i, forming the vector μ, can be expressed as

$$\mu = \mathbf{W^{-1}s} \tag{49}$$

where $\mathbf{s}$ is a vector of RVs $\mathbf{s}_i$ that are independent (or as independent as possible). Let $\mathbf{y} = \mathbf{A}\mu$, where $\mathbf{A}$ is a matrix. If/when $\mathbf{A} = \mathbf{W}$, then $\mathbf{y}_i = s_i$. When $\mathbf{A} \neq \mathbf{W}$, then $\mathbf{y}_i$ is some linear combination of some of the $\mathbf{s}_i$'s because $\mathbf{y} = \mathbf{AW}^{-1}\mathbf{s}$. The more $\mathbf{s}_i$'s are involved in this linear combination, the closer $\mathbf{y}_i$ is to being Gaussian, due to the central limit theorem. Therefore, the case $\mathbf{y}_i = \mathbf{s}_i$ (i.e., $\mathbf{A} = \mathbf{W}$) corresponds to maximizing the non-Gaussianity of $\mathbf{y}_i$. Thus, a row of $\mathbf{W}$, denoted by $\mathbf{W}_i$, may be obtained as the vector $\mathbf{W}_i$ that maximizes the non-Gaussianity of the RV $\mathbf{s}_i$. This leads to the intuition that non-Gaussianity can serve as the optimization metric for ICA.

The negentropies $J(\mathbf{s}_i)$ are difficult to estimate from samples of $\mathbf{s}_i$. Fortunately, we can use an alternative metric to approximate it and it does not matter whether this metric is an accurate approximation or not. For the purpose of optimization objective, it only needs to have the same trend as negentropy when non-Gaussianity of RVs $\mathbf{s}_i$ changes and reaches its maximum value at the same point as negentropy. For example, negentropy can be approximated by the absolute value of kurtosis

$$|\mathrm{kurt}(\mathbf{s}_i)| = \left| E\left[\mathbf{s}_i^4\right] - 3E\left[\mathbf{s}_i^2\right]^2 \right|. \tag{50}$$

Other approximations of negentropy can also be used as objective functions [22].

As we mentioned above, we can determine the ICA matrix $\mathbf{W}$ by finding its row vectors $\mathbf{W}_i$ one by one, and $\mathbf{W}_i$ is solved by maximizing the non-Gaussianity of $\mathbf{s}_i$. The optimum solution for $\mathbf{W}_i$ can be obtained by regular optimization algorithms, such as gradient methods. In order to reduce the computation, the original raw random vector will be first processed by KLE to make its elements (i.e., μ_i) zero mean, unity variance, and uncorrelated RVs. During the optimization process, all $\mathbf{W}_i$'s are also constrained to be orthogonal to each other to prevent two $\mathbf{W}_i$'s from converging to the same vector.

V. Results

We implemented the proposed methods by a software tool structured in Fig. 5. The input of the tool is the behavioral description of an application in the form of C code, and some information of the input data, which could either be its statistical model or sample data. The tool can build the appropriate models (either PCE or KLE) of all system variables, from which their pdfs and other statistics can be derived, and the minimum bitwidth under SNR constraints can be determined. In order to extract the input model automatically, different paths are followed depending on the linearity of the application and the Gaussianity of the input data.

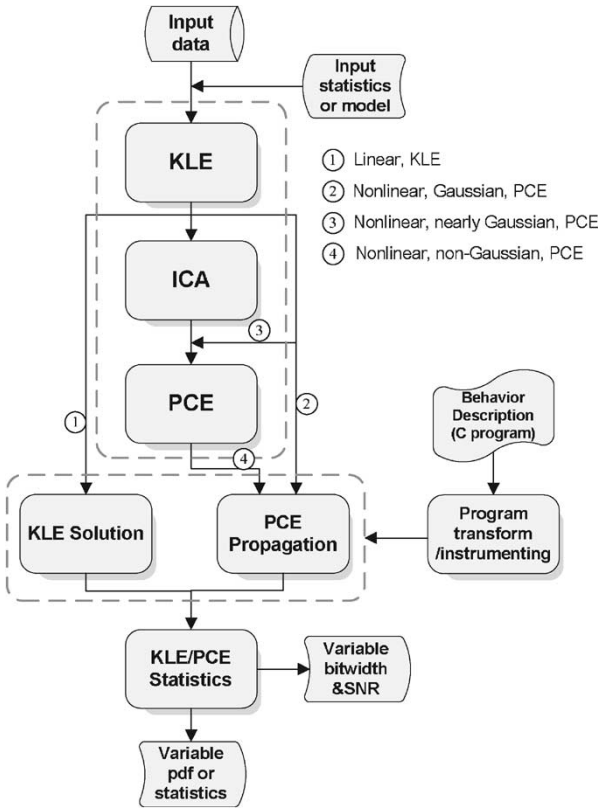


Fig. 5. Block diagram of PCE/KLE method.

```
void nelement()
{
  int i;
  float u, x, y, temp1, temp2;
  x=0;
  for(i=0;i<3;i++)
  {
    update(u);
    temp1=x*u;
    temp2=0.6*x;
    y=temp1+temp2;
    x=u;
  }
 ...................
}
```

Fig. 6. C program before conversion.

The behavioral description of the application under analysis, written in C, is transformed into another C program using compiler techniques. All program variables, including the intermediate variables, in the original program, are mapped to its corresponding analytical models in the transformed program. Likewise, all operators encountered in the original C program are translated into a code that propagates the analytical models, according to the technique described in Sections III-B and IV-B. The transformed program is then compiled and executed to propagate the extracted KLE/PCE model.

Figs. 6 and 7 show an example for program transformation. Fig. 6 is the original C program, while Fig. 7 is the program that performs the corresponding PCE computation. The deterministic variables (such as i in Fig. 6), which are not random, are left untouched. The RVs (such as the floating-point variables in Fig. 6), which are computed from the random input or related to the random input, are transformed into arrays of corresponding PCE coefficients of size `N_PCE`, a constant determined by the order and dimension of the chosen PCE. According to the PCE propagation formulas in Section IV-B, we implement a family of routines to perform PCE coefficient manipulations corresponding to all operations on the variables in the original programs. For example, in Fig. 7, `pce_assign_cnst()` corresponds to assigning a constant to an RV, while `pce_assign()` assigns an RV to the other; `pce_mult()` computes the PCE coefficients for the product of two RVs; `pce_scale()` does the similar operation for the product of a deterministic value and an RV; and `pce_add()` computes the PCE coefficients for the sum of two RVs.

```
void nelement()
{
  int i;
  float u[N_PCE], x[N_PCE], y[N_PCE],
        temp1[N_PCE], temp2[N_PCE];
  pce_assign_cnst(x,0);
  for(i=0;i<3;i++)
  {
    pce_update(u);
    pce_mult(temp1,x,u);
    pce_scale(temp2,0.6,x);
    pce_add(y,temp1,temp2);
    pce_assign(x,u);
  }
 ...................
}
```

Fig. 7. C program after conversion.

We construct a set of experiments first to quantify the effectiveness of the proposed methods in terms of their speed and accuracy, then to demonstrate the advantages of our approach over prior work by performing a "what-if" analysis to show the impact of ignoring correlation and nonlinearity, and finally to illustrate the application of our analysis result for making a tradeoff between bitwidth selection and SNR. All our experiments are conducted on an Ultra 80 Sun workstation.

A. Results for Linear Systems

The KLE-based method proposed in Section III is used to estimate the dynamic ranges of a set of linear benchmarks. The set of benchmarks includes FIR31, FIR63, CXFIR, IIR, IIR8, and FFT128. The first two, FIR31 and FIR63, are 31-order and 63-order finite impulse response (FIR) digital filters, respectively. CXFIR is a complex FIR filter, whose input and output are all complex numbers. IIR is a second-order infinite-impulse response (IIR) digital filter. IIR8 is an eight-order IIR digital filter. FFT128 performs a 128-point fast Fourier transform.

Sample sets of five input random processes are generated to conduct the experiments for linear systems. These were applied to the benchmarks as input signals. These processes fit the ARMA model of time series [21], which is extensively used in engineering and can cover quite a broad range of signals and data such as speech, image, and communication. Every sample set consists of 10 000 traces of 100 time points each, for a total of 1 million data points in each sample set. The

TABLE III
KLE EXTRACTION RESULT

Random Process	Truncation Error	Terms Kept	Computation Time (s)
rp1	2.91%	88	1.4482
rp2	2.97%	82	1.4519
rp3	2.99%	53	1.4439
rp4	2.91%	19	1.4462
rp5	2.92%	7	1.4489

TABLE IV
VARIANCE FROM PROFILING VERSUS VARIANCE FROM KLE, WITH SAMPLE DATA SET RP1 AS INPUT

Benchmark	KLE	Profiling	Difference
FIR31	0.050	0.050(0.044)	-0.027%(14%)
FIR63	0.114	0.114(0.14)	-0.009%(-18%)
CXFIR	0.168	0.168(0.146)	-0.009%(15%)
IIR	0.223	0.224(0.186)	-0.460%(20%)
IIR8	57.997	57.998(47.3)	-0.002%(23%)
FFT128	75.444	75.470(89.2)	-0.038%(-16%)

"randomness" of these sample sets were chosen to be different so as to provide extensive testing, as shown in Table III. Thus, rp1 is the most noisy sample set, while rp5 shows the most temporally correlated behavior. The "randomness" of data samples decreases from rp1 to rp5.

1) Speed and Accuracy of KLE Extraction: We first demonstrate the accuracy and speed with which we extract a KLE model of random processes from the sample data set. Table III shows these results. It can be observed from column 4 that all extractions can be accomplished within 1.5 s while retaining a truncation error of about 3%. Overall, the time complexity of our KLE extraction algorithm is linear in sample size.

It is interesting to observe from column 3 that the more correlated the sample set is, the fewer terms are needed to capture its behavior for a specific truncation error. This is because the energy of more correlated sample sets are concentrated on only a few random terms in the KLE. On the other hand, for sample data sets that show more random behavior in the time domain, their energy is spread over more terms. While the impact of the truncation on the estimation error is small and can be ignored, it can significantly reduce the computational work required.

2) Speed and Accuracy of Obtaining System Response: In this section, we demonstrate the accuracy and speed of the KLE method by comparing it against the traditional profiling-based method. The accuracy results are listed in Table IV. Note that these results are produced where all KLEs are truncated to retain more than 97% of its energy (a truncation error setting of less than 3%). The second and third columns list the variances of the system outputs obtained from the KLE- and profiling-based methods, respectively. Variance is equivalent to second moments when mean is zero. The last column shows the difference in percentage between variances from KLE and profiling. The results from KLE is consistently close to the results from profiling. Even the largest difference is within 0.5%. It is important to note that this difference is only an artifact of truncation during input KLE. If all terms in the KLE are kept, then the difference is essentially zero. In Table IV, the data in parenthesis are obtained from profiling when sample size is 1000; we can see that the large sample size is necessary to accurately estimate the distributions.

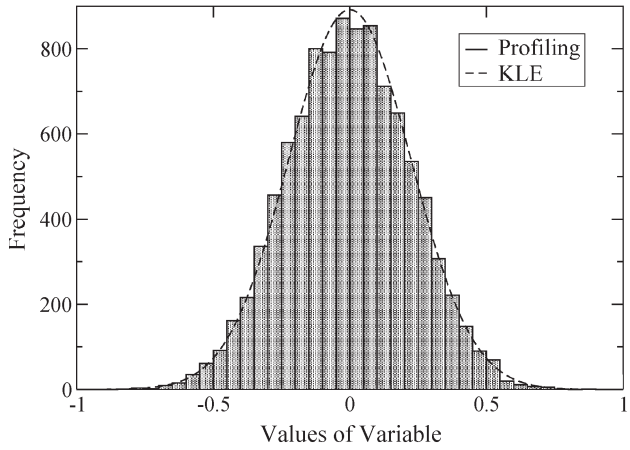


Fig. 8. Histogram from KLE versus histogram from profiling.

TABLE V
COMPUTATION TIME, PROFILING VERSUS KLE

Benchmarks	KLE time (s)	Profiling time (s)	Speedup
FIR31	5.3	578.5	110
FIR63	3.8	701.8	122
CXFIR	12.8	1204.5	125
IIR	4.5	512.9	113
IIR8	3.9	433.1	108
FFT128	4.3	405.2	95

Fig. 8 further shows the histogram of an output variable in benchmark FIR31 when rp1 is applied as input. The bar graph and curve correspond to the histogram from profiling and the estimated pdf from the KLE-based method, respectively. The total sample number for profiling is 10 000. It can be seen that the histogram and estimated pdf match very well. These results clearly verify that the KLE-based method is accurate and reliable. It is important to note that while we use the profiling result to verify the KLE method, it does not necessarily imply that profiling is more accurate than the KLE-based method. In fact, profiling can never reach the real theoretical values, because the simulated sample data can never be infinite. However, provided that the accurate statistics of input random processes are given, the exact statistics of state variables can always be obtained by KLE. For example, given the mean and correlation function of a Gaussian random process as input, then the exact models and statistics for state variables can be calculated.

Finally, Table V shows the computation time of our KLE-based method versus a profiling-based method. The KLE-based technique achieves about 100 times speedup.

B. Results for Nonlinear Systems

Eight benchmarks are selected from practical applications to conduct the experiments for nonlinear systems. Three benchmarks are polynomial filters. Among them, volterra is a second-order volterra filter, which derives its name from the Volterra expansion of general nonlinear functions [5]; teager implements a one-dimensional Teager's algorithm [5] and is commonly used for contrast enhancing in image and speech recognition;

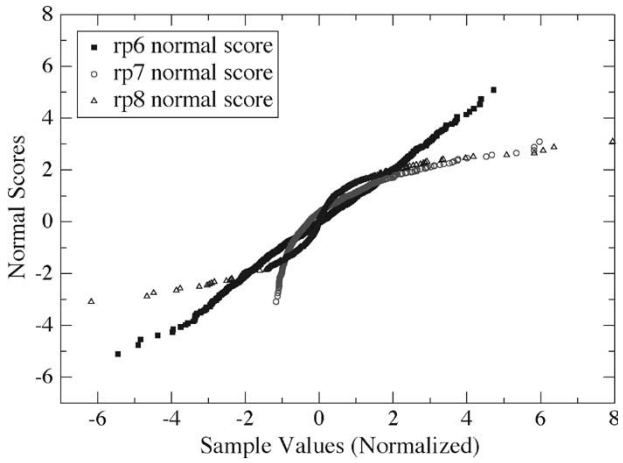


Fig. 9. Normal scores plot for rp6, rp7, and rp8.

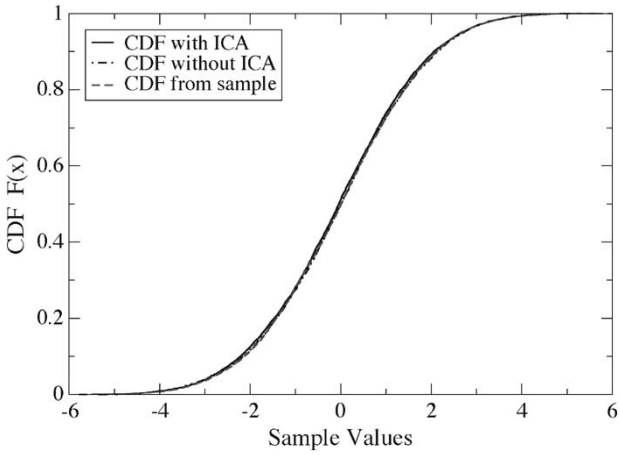


Fig. 10. Distribution functions for rp6.

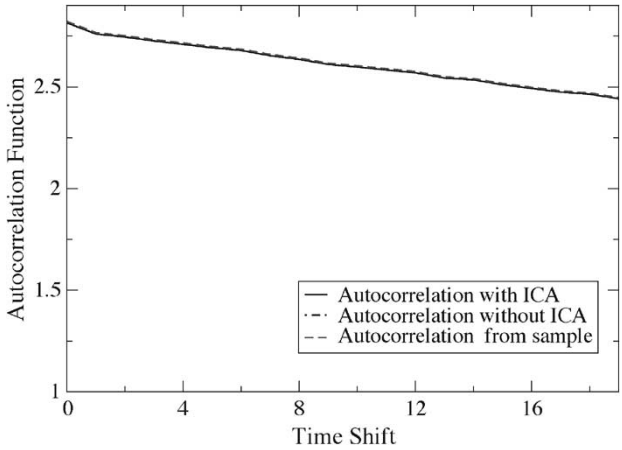


Fig. 11. Autocorrelation functions for rp6.

bilinear is a nonlinear filter whose output is linear with respect to every single system variable. Two benchmarks are adaptive filters, where adaptive1 is a regular LMS adaptive filter that adjusts its parameters in proportion to the computed error signal [5], and adaptive2 adjusts its parameter only according to the sign of the error signal. It is interesting to note that the latter includes conditionals. Another benchmark is a rational filter, which use both multiplication and division. Benchmark adpcm implements the 2-bit Adaptive Differential Pulse Code Modulation (ADPCM) algorithm, which includes a quantizer and several adaptive filters. Benchmark sensor is a high-reliability industry application that extracts up to seven high-frequency components and combine them to measure capacitance. Note that adpcm includes many nonlinear operations, and contain multiple conditionals. Also, note that sensor uses saturating addition, which inherently includes conditionals.

Sample sets of input random processes with different characteristics, speech (for adpcm), and sensor excitation data (for sensor) were used as inputs. Similarly as in the previous section, every sample set consists of 10 000 traces of the process. Such a large number of traces are necessary in order to avoid artifacts caused by finite sample size. This is especially important if we want to capture the tails of the distribution.

1) Speed and Accuracy of PCE Extraction: We construct three random processes, one (rp6) is close to Gaussian, while the other two (rp7 and rp8) are significantly non-Gaussian. Fig. 9 shows normal scores plots for rp6, rp7, and rp8, respectively. The normal scores plot is a commonly used statistical technique to verify whether a set of data samples come from an underlying Gaussian distribution. If Gaussian, the data should form a straight line. Minor deviations of the data from the linear fit is allowable, especially at the tails of the distribution, but strong deviations are a clear sign of a non-Gaussian distribution. From these three normal scores plots, it is clear that rp6 is close to Gaussian, while rp7 and rp8 significantly deviate from the Gaussian.

We use both KLE with ICA and KLE without ICA to extract PCEs from these random processes. Then, three cdfs are generated for every random process: two from the PCE models obtained by ICA and KLE and another one directly from the original sample data.

From Fig. 10, we can observe that if the random process is close to Gaussian, the PCE models with ICA and without ICA find similar distributions and both of them fit the original sample distribution very well. Alternatively, Fig. 11 shows the respective autocorrelation functions; it can be seen that they match each other very well. Therefore, we can conclude that for nearly Gaussian cases, not performing ICA is acceptable.

However, for significantly non-Gaussian cases, the distributions obtained with and without ICA are different, as shown in Fig. 12. Only those obtained with ICA fit the original cdfs well. Fig. 13 shows that even if the cdf without ICA deviates from the sample cdf, its corresponding autocorrelation function is still consistent with the sample autocorrelation function. This is because the autocorrelation function belongs to the second-order moments, and from the perspective of the second-order moments there is no difference between independence and uncorrelatedness. That is, PCEs extracted with ICA and without ICA reach the same results when they are used to estimated the second-order statistics of the sample (such as Fig. 13). However, for statistics higher than second order, PCE without ICA incurs a larger error. Although not shown in the paper, we observe the same for sample rp8. We can therefore conclude that for the general case of systems with non-Gaussian input, ICA is necessary.

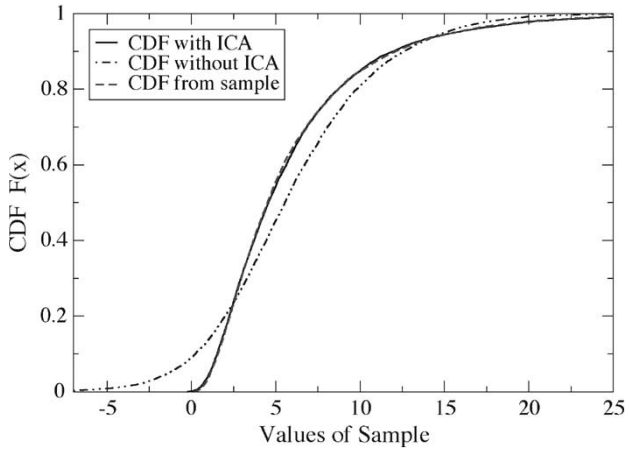


Fig. 12. Distribution functions for rp7.

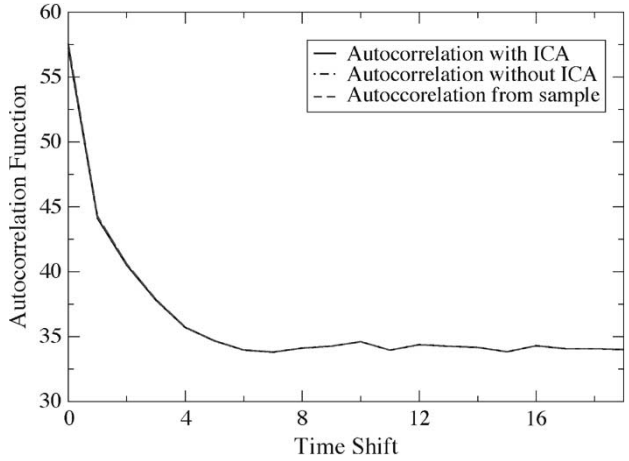


Fig. 13. Autocorrelation functions for rp7.

TABLE VI
PCE EXTRACTION TIME WITH ICA

Dimension & Order	Computation Time (s)
4D 3 order	0.8557
4D 5 order	0.8815
5D 3 order	1.2398
5D 5 order	1.2595
8D 3 order	4.4814
8D 5 order	4.5768

Table VI shows the time spent on performing input PCE extraction with ICA, for PCEs with different dimensions and orders. It can be seen that the computational cost of ICA is only a few seconds. It should also be noted that the PCE extraction time is a one-time cost—once a random process has been characterized, it can be reused for many times.

2) Speed and Accuracy of Obtaining System Response: We now demonstrate the accuracy of the ICA-based method for dynamic-range estimation. We obtain our result by applying rp7 to the benchmarks. Tables VII–IX show the first- (mean), second-, and third-order moments of benchmark output variables obtained by PCE with ICA, without ICA, and profiling, respectively. Tables VII–IX also show the relative differences of PCE results compared with profiling results. We can clearly see that the statistics obtained with ICA match the profiling results very well. However, the moments obtained without ICA are generally quite different from the profiling since the input random process is significantly non-Gaussian. The higher the order of moments, the larger the errors. To visualize the result, Fig. 14 shows the distribution functions for the rational benchmark. It is clear that the distribution function obtained without ICA causes large errors under this case.

TABLE VII
FIRST-ORDER MOMENTS (MEAN) ICA VERSUS PROFILING

Benchmark	Profiling	with ICA	without ICA
volterra	74.91	74.14	74.51
		-1.03%	-0.54%
teager	16.99	16.82	16.78
		-0.95%	-1.23%
adaptive1	86.09	86.15	85.84
		$+0.07\%$	-0.29%
adaptive2	117.7	118.3	125.8
		0.51%	6.84%
rational	1.579	1.574	1.620
		-0.32%	2.64%
bilinear	8.719	8.707	8.8
		-0.14%	0.92%
adpcm	1.43	1.43	1.47
		-0.13%	3%
sensor	1.123	1.123	1.11
		≈ 0	1.2%

TABLE VIII
SECOND-ORDER MOMENTS ICA VERSUS PROFILING

Benchmark	Profiling	with ICA	without ICA
volterra	30984	30967	18353
		-0.05%	-40.8%
teager	11482	11577	4012
		0.83%	-65.1%
adaptive1	10474	10457	10116
		-0.16%	-3.43%
adaptive2	20596	21145	25798
		2.66%	25.3%
rational	3.563	3.556	4.533
		-0.2%	27.2%
bilinear	140.9	137.9	153.7
		-2.15%	9.1%
adpcm	2.175	2.13	2.42
		-2.1%	11%
sensor	2.32	2.35	2.59
		1.3%	12%

TABLE IX
THIRD-ORDER MOMENTS ICA VERSUS PROFILING

Benchmark	Profiling	with ICA	without ICA
volterra	$2.096e7$	$2.069e7$	$6.21e7$
		-1.25%	-70.4%
teager	$5.719e6$	$5.627e6$	$5.543e5$
		-1.61%	-90.3%
adaptive1	$1.625e6$	$1.562e6$	$1.405e6$
		-3.83%	-13.5%
adaptive2	$4.976e6$	$5.250e6$	$6.813e6$
		5.51%	36.9%
rational	11.55	11.99	16.38
		3.82%	41.9%
bilinear	3850	3661	4742
		-4.9%	15.1%
adpcm	3.632	3.52	4.32
		-3%	19%
sensor	3.775	3.672	5.26%
		-2.7%	39%

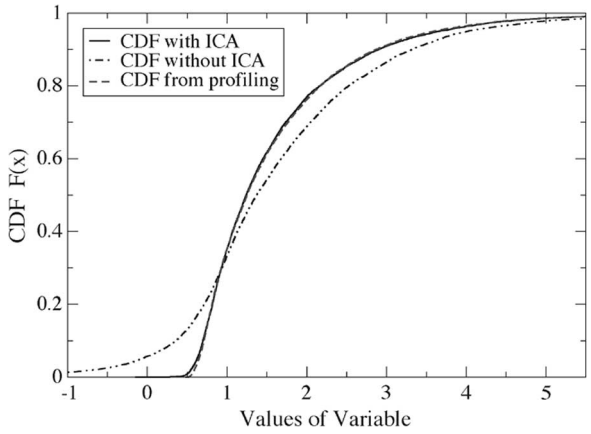


Fig. 14. Variable cdf of benchmark rational.

TABLE X
COMPUTATION TIME PCE VERSUS PROFILING

Benchmark	PCE (s)	Profiling (s)	Speedup
volterra	3	781	261
teager	1.3	154.6	117
adaptive1	1.6	495.5	318
adaptive2	5.5	665.3	120
rational	3.4	347.8	103
bilinear	4.8	1470	307
adpcm	11	1097	101
sensor	19	2507	129

Table X shows the total computation time of the proposed method. Compared with profiling, our method is orders of magnitude faster. Additionally, it should also be noted that in order to compute distributions of all signals, the storage requirement of the profiling method could be a thousand times larger than the PCE method. According to previous studies [7], [9], [10], the efficiency improvement (power or redundant bit) achieved by the accurate dynamic range is from 11% to 50%. The accuracy of the presented method is comparable to profiling. Therefore, our method enjoys the same improvement as profiling, but with much faster speed.

3) Impact of PCE Order and Dimension: In general, larger dimension and order imply better accuracy, but a more expensive analysis. The exact choice of n and p depends on the characteristics of both the input and the system.

To demonstrate the impact of dimension and order on the accuracy of our analysis, we apply two different types of ARMA processes to a third-order volterra filter for different dimension/order combinations. The output cdfs under different dimensions and orders for the first process, which is highly correlated, are shown in Fig. 15. It can be observed that a larger dimension number does not improve the accuracy much. This can be explained by the fact that for a highly correlated process, the energy of the input random process is concentrated only on a few dimensions, therefore, a small n dimension suffices. On the other hand, cutting the order does degrade the result, since significant non-Gaussian components are generated when the random process passes through the nonlinear volterra filter. The non-Gaussian components correspond to higher order PCs.

The output cdfs under different dimensions and orders for the second process, which is much less correlated, are shown

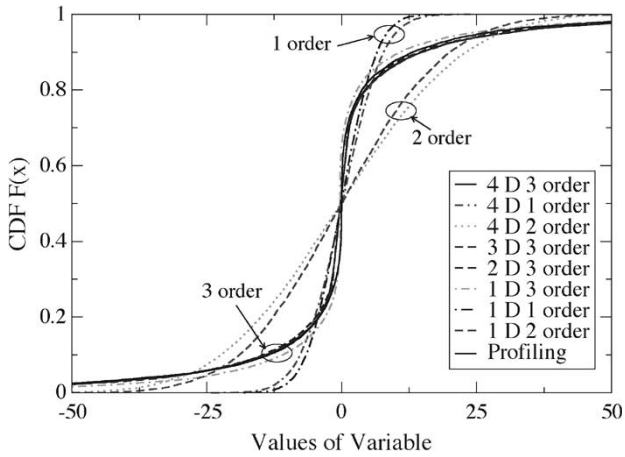


Fig. 15. Dimension/order impact for highly correlated input.

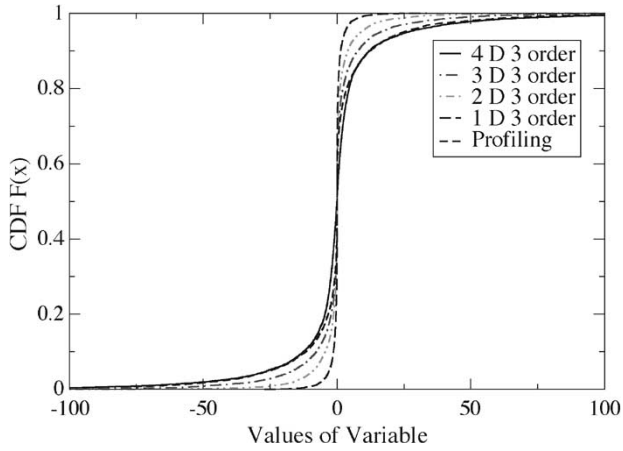


Fig. 16. Dimension/order impact for less correlated input.

in Fig. 16. It can be observed that reducing the dimension does affect the accuracy since signal energy is distributed more evenly.

In general, the necessary dimensions and orders are determined by both the system input and characteristics of the system. For the input signal, the dimension of PCE is determined by the correlatedness of the input signal. The more correlated the input, the fewer the necessary dimensions. The necessary order of PCE is essentially determined by the non-Gaussianity of the input. The more non-Gaussian the input, the higher the necessary order. For the system, the dimension of PCE is determined by the length of remembered history (memory). The longer the remembered history, the higher the necessary dimension. The order of PCE is determined by the nonlinearity of the system. The more nonlinear the system, the higher the necessary PCE order.

C. Impact of Correlations and Nonlinearity

Previous methods introduce simplified assumptions. In this section, we illustrate the impact of these simplifications on dynamic-range estimation by two examples, including both linear and nonlinear systems. The advantage of our proposed methods will be shown by comparing its results against those from previous methods.

TABLE XI
VARIANCES FROM KLE VERSUS CASES WHERE TEMPORAL CORRELATION IS SIMPLIFIED TO BE EITHER 1 (FULL) OR 0 (NONE), WITH SAMPLE DATA SET RP1 AS INPUT

Benchmarks	KLE	$\rho = 1$	$\rho = 0$
FIR31	0.050	0.224	0.012
FIR63	0.114	0.137	0.066
CXFIR	0.168	0.295	0.076
IIR	0.223	0.000	0.287
IIR8	57.997	173.124	14.784
FFT128	75.444	2218.93	17.335

1) Linear Systems With Different Correlation Assumption: We first show that the temporal correlation in the input process can significantly impact the accuracy of the result. This experiment is important because all prior analytical methods did not consider temporal correlation. In fact, prior studies either assumed that the input is a series of independent and identically distributed RVs (correlation coefficient $\rho = 0$), or a sequence of the same RV ($\rho = 1$). This effectively leads to two extreme assumptions on temporal correlation, both unfounded: The former assumes that the input is completely uncorrelated, while the latter assumes that the input is completely correlated.

In order to show that these oversimplified assumptions may cause large estimation errors, we have repeated our experiments and artificially introduced a $\rho = 0$ (independence) assumption in one case and a $\rho = 1$ (correlated) assumption in another. The results are shown in Table XI. All of the three kinds of inputs in this table have exactly the same distribution at any specific time point; the only difference between them is the temporal correlation. In Table XI, the second column shows the variance from the KLE-based method. They are the same as those in Table IV and verified by simulation. The variances listed in the third column are the results under the fully correlated assumption, while the variances in the fourth column are the results under the independent assumption. We can see that the results from different temporal correlation are quite different. For FFT128, the variance estimated from the fully correlated assumption is about 30 times larger than the result from the KLE-based method, while the variance from the independent assumption is about 4.4 times less than the KLE-based method.

Fig. 17 shows the pdf curves obtained by the KLE-based method and from the fully correlated case and the independent case, corresponding to the test case FIR31 in the second row of Table XI. The accuracy of the pdf from the KLE-based method has been verified by the histogram match in Fig. 8. The independence assumption gives a narrow distribution, while the extremely correlated assumption gives a flat distribution, both of which are quite wrong, as can be seen from the figure. The figure also shows what happens in the case when both temporal and spatial correlations are ignored, and the results are even worse in that case, as would be expected.

These results clearly demonstrate the advantages of the KLE approach and the need to maintain correlation information. Truncating one of the "wrong" pdfs in Fig. 8 would either give significant noise, or be very conservative. To illustrate this, we show the dynamic ranges under different range probabilities in Table XII. It can be observed that the extremely correlated assumption leads to a dynamic range five times wider than necessary—a pessimistic result—while the independent assumption leads to a dynamic range two times narrower than required—an overly optimistic result.

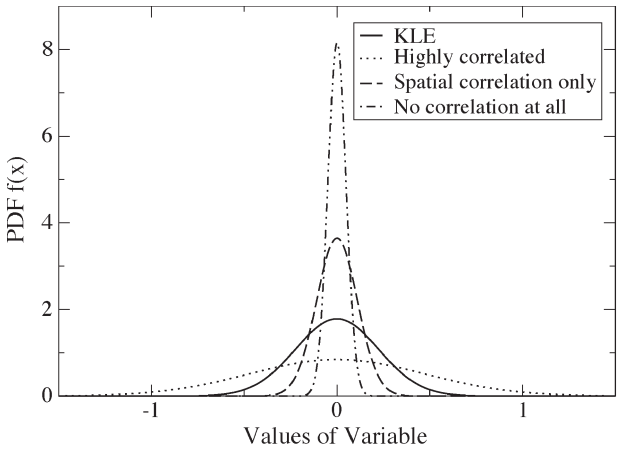


Fig. 17. The pdf under various assumptions related to correlation, compared to the pdf from KLE.

TABLE XII
DYNAMIC RANGE FROM KLE VERSUS CASES WHERE TEMPORAL CORRELATION IS SIMPLIFIED TO BE EITHER 1 (FULL) OR 0 (NONE)

Probability	KLE	$\rho = 1$	$\rho = 0$
99.98%	± 32.14	± 174.29	± 15.41
99.02%	± 22.41	± 121.53	± 10.74
95.00%	± 17.02	± 92.33	± 8.16
90.10%	± 14.33	± 77.72	± 6.87

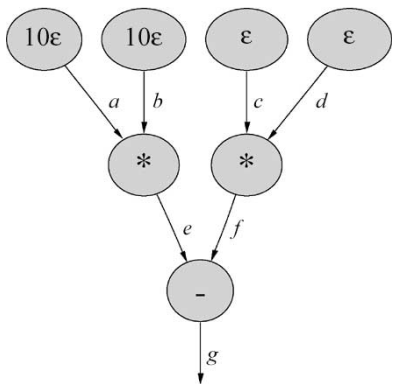


Fig. 18. DFG example.

2) Nonlinearity and Correlation in DFG: In this section, we extend the discussion in the previous section to the nonlinear case and include the treatment to nonlinearity. To quantify the advantages of the PCE method and offer insight on why it can perform better, we apply different analysis methods on a common example, extracted from the teager benchmark, and compare the statistics obtained against the exact solution.

As shown in Fig. 18, the DFG computes the equation $g = e - f = ab - cd$. Given an input random process x, we apply $x(n)$, its value at time point n to a and b, $x(n-1)$ to c, and $x(n+1)$ to d.

To show the impact of correlation and simplify the demonstration, we assume that the input process is highly correlated and $x(n)$, $x(n-1)$, $x(n+1)$ can be characterized by 10ε, ε,

and ε, respectively, where ε is an RV uniformly distributed over $[-1, 1]$. It is easy to see that the temporal correlation of the input is translated into the spatial correlation on the DFG, as a, b, c, and d can all be characterized by the same RV ε.

We now consider the exact statistics at the output. Since $g = e - f = ab - cd = (10\varepsilon)^2 - \varepsilon^2$, we have $e = 99\varepsilon^2$. Therefore, the exact statistics are

$$E[g] = E[99\varepsilon^2] = 33$$
$$E[g^2] = E\left[(99\varepsilon^2)^2\right] = \frac{99^2}{5} = 1960.2.$$

We now consider the moment propagation approach proposed in [19]. Since it always assumes that the inputs of an operator are independent, we have

$$\begin{aligned} E[g] &= E[a]E[b] - E[c]E[d] \\ &= E[10\varepsilon]E[10\varepsilon] - E[\varepsilon]E[\varepsilon] = 0 \\ E[g^2] &= E\left[(e-f)^2\right] = E[e^2 + f^2 - 2ef] \\ &= E[e^2] + E[f^2] - 2E[e]E[f] \\ &= E[10\varepsilon]^2 + E[\varepsilon]^2 = 3333.6667. \end{aligned}$$

It is apparent that both the mean and variance predicted by the moment propagation approach deviate significantly from the exact result.

We now consider the affine arithmetic approach proposed in [4], which is able to capture spatial correlation by representing each signal as a linear combination of uniformly distributed RVs. In the case of our example, a, b, c, and d are already in the affine form. This representation handles linear operations such as additions precisely; however, the captured correlation may be lost after the nonlinear operations. For example, when multiplications are applied, products of RVs may result, and to keep the result in affine form, a constant bound for each input has to be found, and an extra RV has to be introduced. For example, considering $e = ab$, according to [4], we have

$$\begin{aligned} e &= 10\varepsilon \times 10\varepsilon \\ &= B(10\varepsilon)B(10\varepsilon)\varepsilon_1 \\ &= 100\varepsilon_1 \end{aligned}$$

where ε_1 is a new RV independent of ε. Following the same procedure, we can derive $g = 100\varepsilon_1 - \varepsilon_2$, where ε_1 and ε_2 are independent RVs. It is evident that the original correlations are lost after the nonlinear operations. As a result, we obtain the statistics as follows:

$$E[g] = E[100\varepsilon_1 - \varepsilon_2] = 0$$
$$E[g^2] = 100^2 E\left[\varepsilon_1^2\right] + E\left[\varepsilon_2^2\right] = 3333.6667.$$

Another argument that affine arithmetic might not be adequate for nonlinear systems is the fact that a uniform distribution will become nonuniform after nonlinear operations, and it is theoretically impossible to represent the new distribution using a linear combination of uniform distributions.

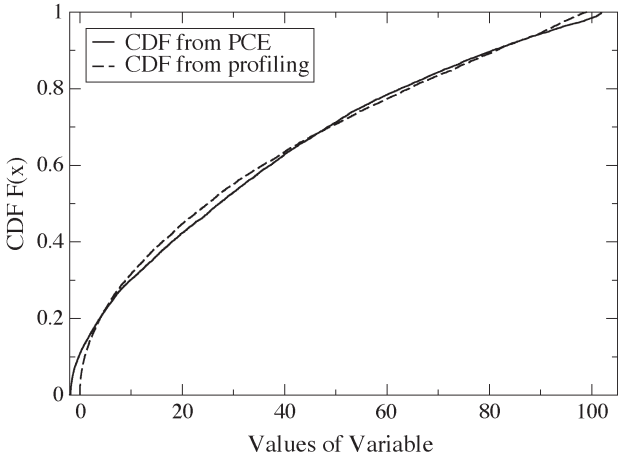


Fig. 19. The cdf obtained by PCE and simulation for example in Fig. 18.

The power of the PCE method lies in the fact that signals are represented as the combination of polynomials, therefore correlations can naturally survive nonlinearity. For the given example, we generate sample data for the uniform RV ε, from which a one-dimensional five-order PCE is extracted as

$$\begin{aligned} \epsilon = 0.002757\Psi_0 + 0.565888\Psi_1 - 0.001018\Psi_2 \\ - 0.048217\Psi_3 + 0.000139\Psi_4 + 0.003308\Psi_5. \end{aligned}$$

The PCE for a, b, c, and d can be obtained accordingly. Using the propagation method described in Section IV-B, we can finally obtain

$$\begin{aligned} g &= 33.2147\Psi_0 + 0.1182\Psi_1 + 18.3929\Psi_2 \\ &\quad - 0.0590\Psi_3 - 2.7226\Psi_4 + 0.0154\Psi_5 \\ E[g] &= 33.2147 \\ E[g^2] &= 1957.8. \end{aligned}$$

Compared with the exact result, the predicted statistics only have 0.65% and 0.12% error, respectively. Fig. 19 shows that the complete distribution obtained by the PCE method can match closely with the result of simulation.

D. Tradeoff Between Bitwidth and SNR

As an application of our paper, we will show how the distribution and statistics obtained by the KLE-based method can be used to make a tradeoff between bitwidth and the SNR. Due to lack of space, and since this is only for demonstration purposes, this will be very brief, will focus on the overflow error only, and will be done only for the case of benchmark FFT128. This can be extended to cover round-off error as well, and can be applied to all the other benchmarks. In order to fully determine the bitwidth of variables, in general, we need to determine the fraction bitwidth by round-off noise requirement or signal source precision (such as A/D resolution). This may require error or noise propagation. Because we focus on dynamic-range estimation in this paper, in our experiment, we assume the variable fraction bitwidth has been determined (for example,

TABLE XIII
BITWIDTH AND SNR

Bitwidth	Dynamic Range	Probability	SNR(dB)
14	± 8.191	65.44%	7.68
15	± 16.383	94.07%	17.95
16	± 32.767	99.98%	47.62
17	± 65.535	≈ 1	148.34
18	± 131.071	≈ 1	527.92

determined by A/D resolution). The SNR here is computed from the noise caused by overflow instead of round-off noise.

For a candidate bitwidth, one can easily compute the corresponding dynamic range, the range probability, (the probability for the value to fall within the range), and the SNR using [23]

$$\text{SNR} = 10 \log_{10} \left(\frac{\sigma_{\text{s}}^2}{\sigma_{\text{n}}^2} \right) \tag{51}$$

where σ_{s}^2 and σ_{n}^2 are the variances of signal and noise, respectively. Assuming the data is zero-mean Gaussian and is to be truncated to a dynamic range of $\pm z\sigma_{\text{s}}$, where $z > 0$, and ignoring discretization (round-off) error and focusing only on overflow error, it can be easily shown that the SNR is given by

$$\text{SNR} = -10 \log_{10} \left[2(1 + z^2)\Phi(-z) - 2z\phi(z) \right] \tag{52}$$

where $\Phi(\cdot)$ and $\phi(\cdot)$ are the cdf and the pdf of the standard normal distribution, respectively. With this, one can generate results such as those in Table XIII, which show the tradeoff between bitwidth and SNR. Notice that, for a bitwidth b, and allowing for one sign bit, the dynamic range is $\pm(2^{b-1}\delta - 1)$, where δ is the discretization step size, i.e., the value of the least significant bit. For the data in Table XIII, $\delta = 0.001$. Based on the computed variances from KLE for any/all signals, one can generate and use such tables to manage the tradeoff between bitwidth and SNR.

VI. CONCLUSION

In this paper, a new method for dynamic-range estimation is presented, which models the variables and intermediate results in the program by random processes. When input sample data are supplied, either a KLE or a PCE model, each of which can be considered as a combination of orthogonal RVs, or orthogonal polynomials in terms of RVs, can be obtained and propagated through the system. Full statistical information about the variables can in turn be obtained.

The PCE method has its own limitations. To expand an RV or a random process by PCE, they must be square integrable, that is, its second-order moment exists. Fortunately, for all practical systems, the variables are always square integrable. For a finite sample data set, its second-order moments (square sample mean) always exist. Most of the analytical RV models are also square integrable. However, in some corner cases, there are nonsquare-integrable RVs, which cannot be expanded by PCE. For example, RV $2^{2^{\xi}}$, where ξ is a standard Gaussian RV, does not have the second-order moment, and thus it is not expandable by PCE.

Compared with currently available methods, this approach has the following advantages. Firstly, more detailed information about the dynamic range or values of the variables is obtained. It includes the probability distributions. Designers can associate every choice of bitwidth or dynamic range with a value of the SNR, and thus make judicious tradeoffs between reliability and cost or power. Secondly, the proposed method fully considers both the spatial and temporal correlations, and is able to accurately handle the nonlinear structures. Thirdly, our method can construct random-process models from real sample data or empirical statistics, instead of using oversimplified models or assumptions. Our method can therefore compete with profiling for accuracy and flexibility. Finally, our method is computationally efficient. If one only considers the propagation of random processes, it can be several orders of magnitude faster than profiling very easily. For designers who wish to thoroughly explore the design space, they only need to extract the random-process model once. When they modify their designs and redo the dynamic-range estimation, their computation cost can be limited to the propagation of the previously obtained input models.

REFERENCES

[1] S. Mahlke, R. Ravindran, M. Schlansker, R. Schreiber, and T. Sherwood, "Bitwidth cognizant architecture synthesis of custom hardware accelerators," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 20, no. 11, pp. 1355–1371, Nov. 2001.

[2] A. Garcia-Ortiz, L. Kabulepa, T. Murgan, and M. Glesner, "Moment-based power estimation in very deep submicron technologies," in *Proc. Int. Conf. Computer Aided Design*, San Jose, CA, Nov. 9–12, 2003, pp. 107–112.

[3] C. F. Fang, R. A. Rutenbar, M. Puschel, and T. Chen, "Toward efficient static analysis of finite-precision effects in DSP applications via affine arithmetic modeling," in *Proc. Design Automation Conf.*, Anaheim, CA, Jun. 2–6, 2003, pp. 496–501.

[4] C. F. Fang, R. A. Rutenbar, and T. Chen, "Fast, accurate static analysis for fixed-point finite-precision effects in DSP designs," in *Proc. Int. Conf. Computer Aided Design*, San Jose, CA, Nov. 9–12, 2003, pp. 275–282.

[5] V. J. Mathews and G. L. Sicuranza, *Polynomial Signal Processing*. New York: Wiley, 2000.

[6] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 2nd ed. New York: McGraw-Hill, 1984.

[7] Y. Cao and H. Yasuura, "A system-level energy minimization approach using datapath width optimization," in *Proc. Int. Symp. Low Power Electronics and Design*, Huntington Beach, CA, Aug. 2001, pp. 895–903.

[8] K. Kum and W. Sung, "Combined word-length optimization and high-level synthesis of digital signal processing systems," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 20, no. 8, pp. 921–930, Aug. 2001.

[9] K. Kum, J. Kang, and W. Sung, "A floating-point to integer C converter with shift reduction for fixed-point digital signal processors," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, Phoenix, AZ, Mar. 15–19, 1999, pp. 2163–2166.

[10] T. Aamodt and P. Chow, "Embedded ISA support for enhanced floating-point to fixed-point ANSI C compilation," in *Proc. Int. Conf. Compilers, Architectures Synthesis for Embedded Systems*, San Jose, CA, Nov. 17–19, 2000, pp. 128–137.

[11] L. B. Jackson, "On the interaction of roundoff noise and dynamic range in digital filters," *Bell Syst. Tech. J.*, vol. 49, no. 2, pp. 159–184, Feb. 1970.

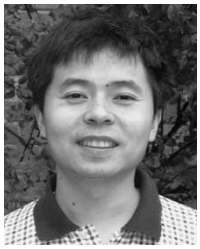
[12] J. Carletta, R. Veillette, F. Krach, and Z. Fang, "Determining appropriate precisions for signals in fixed-point IIR filters," in *Proc. Design Automation Conf.*, Anaheim, CA, Jun. 2–6, 2003, pp. 656–661.

[13] I. T. Jolliffe, *Principal Component Analysis*. New York: Springer-Verlag, 2002.

[14] J. E. Kolassa, *Series Approximation Methods in Statistics*. New York: Springer-Verlag, 1994.

[15] P. Hall, *The Bootstrap and Edgeworth Expansion*. New York: Springer-Verlag, 1992.

[16] A. Lakhang and H. Mausser, "Estimating the parameters of the generalized lambda distribution," *Algo Res. Q.*, vol. 3, no. 3, pp. 47–58, Dec. 2000.
[17] J. C. Park and M. Schlansker, "On predicated execution," HP Laboratories, Palo Alto, CA, Tech. Rep. HPL-91-58, May 1991.
[18] V. Chaiyakul, D. Gajski, and L. Ramachandran, "High-level transformations for minimizing syntactic variances," in *Proc. ACM/IEEE 30th DAC*, Dallas, TX, Jun. 1993, pp. 413–418.
[19] R. G. Ghanem and P. D. Spanos, *Stochastic Finite Elements: A Spectral Approach*. Mineola, NY: Dover, 2003. revised edition.
[20] O. P. Le Maitre, O. M. Knio, H. N. Najm, and R. G. Ghanem, "A stochastic projection method for fluid flow," *J. Comput. Phys.*, vol. 173, no. 2, pp. 481–511, Nov. 2001.
[21] H. Shumway and D. S. Stoffer, *Time Series Analysis and Its Applications*. New York: Springer-Verlag, 2000.
[22] A. Hyvarinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. New York: Wiley, 2001.
[23] M. C. Jeruchin, P. Balaban, and K. S. Shanmugan, *Simulation of Communication Systems*. New York: Plenum, 2000.



Bin Wu (M'99–S'03) received the B.S. and M.S. degrees in electrical engineering from Xian Jiaotong University, Xian, China, in 1995 and 1998, respectively. He is currently working toward the Ph.D. degree in the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada.

Jianwen Zhu (S'95–M'00) received the B.S. degree in electrical engineering from Tsinghua University, Beijing, China, in 1993, and the M.S. and Ph.D. degrees in computer science from the University of California, Irvine, in 1996, and 1999, respectively.

He is currently an Associate Professor in the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada. He is the coauthor of the book *SpecC: Specification Language and Methodology* (Boston, MA: Kluwer-Academic, 2000). His research interests include hardware/software codesign, high-level and logic synthesis, and programming language design and implementation.

Farid N. Najm (S'85–M'89–SM'96–F'03) received the Ph.D. degree in electrical and computer engineering (ECE) from the University of Illinois, Urbana-Champaign (UIUC), in 1989.

He worked at Texas Instruments until 1992, then was a Professor at UIUC until 1999, when he joined the ECE Department at the University of Toronto, Toronto, ON, Canada, where he is now a Professor and Vice-Chair of the ECE Department. His research interests include computer-aided design (CAD) for integrated circuits, with an emphasis on circuit-level issues related to power dissipation, timing, and reliability.

Dr. Najm received the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN Best Paper Award, in 1992, the National Science Foundation (NSF) Research Initiation Award, in 1993, and the NSF CAREER Award, in 1996. He is an Associate Editor for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN and was an Associate Editor for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) from 1997 to 2002. He has served as General Chairman for the ISLPED-99 conference and as Technical Program Co-Chairman for ISLPED-98. He has also served on the technical committees of ICCAD, DAC, CICC, ISLPED, and ISQED.