# An Optimized Hardware Architecture of a Multivariate Gaussian Random Number Generator

CHALERMPOL SAIPRASERT, CHRISTOS-S. BOUGANIS,
and GEORGE A. CONSTANTINIDES
Imperial College London

Monte Carlo simulation is one of the most widely used techniques for computationally intensive simulations in mathematical analysis and modeling. A multivariate Gaussian random number generator is one of the main building blocks of such a system. Field Programmable Gate Arrays (FPGAs) are gaining increased popularity as an alternative means to the traditional general purpose processors targeting the acceleration of the computationally expensive random number generator block. This article presents a novel approach for mapping a multivariate Gaussian random number generator onto an FPGA by optimizing the computational path in terms of hardware resource usage subject to an acceptable error in the approximation of the distribution of interest. The proposed approach is based on the eigenvalue decomposition algorithm which leads to a design with different precision requirements in the computational paths. An analysis on the impact of the error due to truncation/rounding operation along the computational path is performed and an analytical expression of the error inserted into the system is presented. Based on the error analysis, three algorithms that optimize the resource utilization and at the same time minimize the error in the output of the system are presented and compared. Experimental results reveal that the hardware resource usage on an FPGA as well as the error in the approximation of the distribution of interest are significantly reduced by the use of the optimization techniques introduced in the proposed approach.

Categories and Subject Descriptors: B.2.1 [**Arithmetic and Logic Structures**]: Design Styles

General Terms: Design, Performance, Algorithms

Additional Key Words and Phrases: Multivariate Gaussian distribution, word-length optimization, FPGA, hardware architecture optimization

The authors gratefully acknowledge the support of the UK EPSRC (Grant reference EP/C549481/1).
Authors' address: C. Saiprasert, C.-S. Bouganis, and G. A. Constantinides, Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, United Kingdom; email: cs405@imperial.ac.uk.

**ACM Reference Format:**
Saiprasert, C., Bouganis, C.-S., and Constantinides, G. A. 2010. An optimized hardware architecture of a multivariate Gaussian random number generator. ACM Trans. Reconfig. Techn. Syst. 4, 1, Article 2 (December 2010), 21 pages. DOI: 10.1145/1857927.1857929.
http://doi.acm.org/10.1145/1857927.1857929.

## 1. INTRODUCTION

Monte Carlo (MC) simulation is a well known technique which is widely used in a variety of applications, especially in stochastic scientific processes and financial modeling [Glasserman 2004]. The computation of value-at-risk [Glasserman et al. 2000] and credit-risk calculation [Glasserman and Li 2003] are some of the examples of financial applications which heavily rely on these simulations under which many financial instruments can be modeled. At the heart of a Monte Carlo simulation lies a sequence of randomly generated numbers, which is one of the essential prerequisites for almost all Monte Carlo simulations. These random numbers are drawn from a variety of distributions where the most commonly in use is the multivariate Gaussian distribution. Hence, a key component in any Monte Carlo simulation is a multivariate Gaussian random number generator (MVGRNG).

In recent years, there is an increasingly high demand for computationally intensive calculations in finance due to the ever increasing number of assets and portfolios size. Traditionally, the random number generator and the financial application itself have been implemented on general purpose processors. Recently, alternative methods based on Graphics Processing Unit (GPU) [Thomas et al. 2009] and Field Programmable Gate Array (FPGA) [Woods and VanCourt 2008] Thomas and Luk [2008a] have gained a great deal of attention due to their higher throughput performance and lower power consumption. In this work we target an FPGA device due to its capability to provide fine-grain parallelism and reconfigurability. Existing works in the literature concerning hardware acceleration of financial applications include the calculation of Monte Carlo–based credit derivative pricing [Kaganov et al. 2008] and interest rates simulation [Thomas et al. 2007]. In order to maximize the performance of the system, many researches have focused on the minimization of the hardware resources occupied by the random number generator. In the case where a mapping of a multivariate Gaussian random number generator to an FPGA platform is targeted, three pieces of work have been published so far in the literature [Saiprasert et al. 2008, 2009; Thomas and Luk 2008b]. The approach in Thomas and Luk [2008b] is based on Cholesky decomposition and is capable of producing samples at a high throughput, but it lacks the flexibility to produce an architecture for any given resource constraints. This has been addressed by the technique proposed in Saiprasert et al. [2008] which offers the flexibility to accommodate a range of resource constraints when certain conditions are met. However, the approach in Saiprasert et al. [2008] does not exploit the full flexibility of the design space since the precision of the datapath is kept fixed throughout the design. This precision issue has been taken into account in Saiprasert et al. [2009] where word-length optimization techniques are utilized in order to produce an architecture which consists of multiple precisions in its datapath.

This article presents in more depth the main ideas of our previous work in Saiprasert et al. [2009], and introduces two new algorithms for the design optimization of the multivariate Gaussian random number generator block. The major contributions of this article are as follows:

—The use of eigenvalue decomposition algorithm to minimize the hardware resource utilization of a multivariate Gaussian random number generator. The use of the proposed approach leads to a design that consists of computational paths with different precision requirements. This methodology produces designs with reduced resource usage in comparison to existing approaches without degrading the quality of the system's output.
—An in-depth error analysis on the impact of the error due to truncation/rounding operations in the datapath is performed providing an analytical expression of such injected error into the system.
—Two novel methodologies to combat the impact of the truncation/rounding error are introduced, targeting the production of random samples whose distribution is as close as possible to the target distribution of interest.

## 2. GENERATION OF MULTIVARIATE GAUSSIAN RANDOM NUMBERS

An important prerequisite to the generation of multivariate Gaussian samples is a set of independent univariate Gaussian samples. Many methodologies exist in the literature for the generation of univariate Gaussian random numbers. These include the Ziggurat method [Marsaglia and Tsang 2000], the Wallace method [Wallace 1996] and the Box-Muller method [Box and Muller 1958]. All of these methods have been implemented on an FPGA [Lee et al. 2005, 2006; Zhang et al. 2005]. An extensive review of these techniques has been performed in Thomas et al. [2007] where it has been concluded that the Wallace method has the highest throughput while the Ziggurat method comes second.

This work focuses on the multivariate Gaussian random distribution which is defined by two parameters, its mean denoted by $\mathbf{m}$ and its covariance matrix $\mathbf{\Sigma}$. Many techniques have been deployed to generate random samples from this distribution and some of the most widely used are the Cholesky Factorization technique and the Singular Value Decomposition algorithm.

### 2.1 Cholesky Factorization

Given a target multivariate Gaussian random distribution with mean $\mathbf{m}$ and covariance $\mathbf{\Sigma}$, random samples from such a distribution using the Cholesky factorization technique are generated as follows. Initially, the covariance matrix $\mathbf{\Sigma}$ is decomposed using Cholesky decomposition into a product of a lower triangular matrix $\mathbf{A}$ and its transpose, $\mathbf{\Sigma} = \mathbf{A}\mathbf{A}^T$ [Glasserman 2004]. The required samples $\mathbf{x}$ are generated through a linear combination of univariate Gaussian samples $\mathbf{z}$ that follow a standard Gaussian distribution $N(\mathbf{0}, \mathbf{I})$ as in (1).

$$\mathbf{x} = \mathbf{m} + \mathbf{A}\mathbf{z}. \tag{1}$$

Due to the lower triangular property of matrix $\mathbf{A}$, the number of computations are reduced by almost a factor of two in comparison to full matrix-vector multiplication. Hence, this method is widely used in software-based applications as well as in some hardware approaches as in [Thomas and Luk 2008b]. It should be noted that this method is only applicable when the covariance matrix $\mathbf{\Sigma}$ is positive-definite.

## 2.2 Approximation of Covariance Matrix Using Eigenvalue Decomposition Algorithm

An alternative method is to decompose $\mathbf{\Sigma}$ by eigenvalue decomposition [Chan and Wong 2006]. A technique known as Singular Value Decomposition (SVD) algorithm which expresses a matrix as a linear combination of three separable matrices [Press et al. 1992] is used in this approach. Using SVD, $\mathbf{\Sigma}$ can be expressed as $\mathbf{\Sigma} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ where $\mathbf{U}$ is an orthogonal matrix ($\mathbf{U}\mathbf{U}^T = \mathbf{I}$) containing eigenvectors $\mathbf{u}_1, ..., \mathbf{u}_N$ while $\mathbf{\Lambda}$ is a diagonal matrix with diagonal elements being eigenvalues $\lambda_1, ..., \lambda_N$. By letting $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}^{1/2}$, multivariate Gaussian random samples that follow $N(\mathbf{m}, \mathbf{\Sigma})$ can be generated as in (2), where $\mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$.

$$\begin{aligned}
\mathbf{x} &= \mathbf{A}\mathbf{z} + \mathbf{m} \\
&= \mathbf{U}\mathbf{\Lambda}^{1/2}\mathbf{z} + \mathbf{m} \\
&= (\sqrt{\lambda_1}\mathbf{u}_1 z_1 + \sqrt{\lambda_2}\mathbf{u}_2 z_2 + \ldots + \sqrt{\lambda_N}\mathbf{u}_N z_N) + \mathbf{m} \\
&= \sum_{i=1}^{K}(\sqrt{\lambda_i}\mathbf{u}_i z_i) + \mathbf{m}.
\end{aligned} \tag{2}$$

The value of $K$, which is the number of decomposition levels, should equal to the rank of the covariance matrix $\mathbf{\Sigma}$ for full representation. Thus, the covariance matrix of the original distribution $\mathbf{\Sigma}$ can be approximated by taking into account $K$ levels of decomposition where $K \leq rank(\mathbf{\Sigma})$.

By approximating the covariance matrix $\mathbf{\Sigma}$ using $K$ levels of decomposition, an error is introduced to the system due to the deviation of the approximated covariance matrix from the original one. In this work, the metric that is used to quantify this error is the mean square error which is given in (3).

$$F(\mathbf{\Sigma}, \overline{\mathbf{\Sigma}}) := \frac{1}{N^2}\|\mathbf{\Sigma} - \overline{\mathbf{\Sigma}}\|^2, \tag{3}$$

where $\overline{\mathbf{\Sigma}}$ denotes the approximated covariance matrix and $\|.\|$ denotes the Frobenius norm.

As will be demonstrated later in this article, the decomposition introduced in this section enables us to exploit the different precision requirements of each decomposition level leading to an optimized hardware design. To the best of the authors' knowledge, this technique has not previously been applied to the hardware implementation of a multivariate Gaussian random number generator.

## 3. RELATED WORK

The first FPGA-based multivariate Gaussian random number generator has been published by Thomas and Luk [2008b]. Their approach is based on factorizing the input covariance matrix using Cholesky decomposition in order to take advantage of the lower triangular property of the resulting matrix. The design has the capability to serially generate a vector of multivariate Gaussian random numbers every $N$ clock cycles where $N$ denotes the dimensionality of the distribution. The authors map the multiply-add operation in the algorithm onto DSP48 blocks on an FPGA, requiring $N$ blocks for an $N$-dimensional Gaussian distribution. One apparent shortcoming of their approach is the restriction in resource allocation since the dimension of the distribution under consideration dictates the number of required DSP48 blocks on the FPGA.

An alternative approach has been proposed in Saiprasert et al. [2008] to solve the problem encountered in Thomas and Luk [2008b], where an algorithm based on the use of Singular Value Decomposition algorithm was introduced to approximate the lower triangular matrix $\mathbf{A}$, the result of applying Cholesky decomposition on the covariance matrix $\mathbf{\Sigma}$, by trading off "accuracy" for an improved resource usage. In Saiprasert et al. [2008], "accuracy" is defined as the mean square error in the approximation of the input covariance matrix $\mathbf{\Sigma}$. The approach in Saiprasert et al. [2008] requires $2K$ DSP48 blocks to produce a vector of size $N$, where $K$ denotes the number of decomposition levels required to approximate the lower triangular matrix $\mathbf{A}$ within a certain accuracy using the SVD algorithm. The resource usage can be reduced in comparison to Thomas and Luk [2008b] if $K$ is less than $N/2$ while the generated architecture achieves the same throughput performance. As well as an improved resource usage, which is data dependent, this approach offers the flexibility to produce a hardware system that meets any given resource constraint. That is the dimensionality of the Gaussian distribution no longer dictates the number of required DSP48 blocks by trading off the "accuracy" in the approximation of $\mathbf{\Sigma}$. However, it has been shown in [Saiprasert et al. 2008] that allocation of a fixed precision to all of the computation paths of the design does not lead to the optimum resource utilization.

## 4. PROPOSED HARDWARE ARCHITECTURE

This section describes the hardware architecture for an FPGA implementation of a multivariate Gaussian random number generator using the proposed approach, which is based on the Eigenvalue decomposition algorithm. For the remainder of this paper we will focus on the generation of random samples from a centralized Gaussian distribution, that is a distribution with zero mean. Any other noncentralized distribution with the same covariance can be produced by a simple offset of the generated vectors. Consider a covariance matrix $\mathbf{\Sigma}$, the proposed algorithm applies the Eigenvalue decomposition algorithm to express $\mathbf{\Sigma} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$. According to (2), the random samples can be generated as in (4)

$$\mathbf{x} = \sum_{i=1}^{N} \sqrt{\lambda_i}\mathbf{u}_i z_i \simeq \sum_{i=1}^{N} \mathbf{c}^i z_i, \tag{4}$$

where $\mathbf{c}^i$ denotes a product of $\sqrt{\lambda_i} \cdot \mathbf{u}_i$ after quantization with the desired word-length for the remainder of this paper. In this work, the vectors $\mathbf{c}^i$ are calculated by feeding back the error due to quantization of the previous decomposition levels, a method that was introduced in Bouganis et al. [2009].

The multivariate Gaussian samples are generated as the sum of products of $\mathbf{c}$ and $z$ and, thus, the various decomposition levels $i$ are mapped onto computational blocks (CB) designed for an FPGA. Each CB contains the hardware which performs the multiply-add operation. The architecture is mapped to logic elements only, so that the precision of the datapath can be varied as opposed to Thomas and Luk [2008b] and Saiprasert et al. [2008], where the precision of the architecture is fixed to 18bits as only DSP48 blocks are utilized. As it will be demonstrated later, quantizing the coefficients of the vectors $\mathbf{c}^i$ to lower bit widths could be sufficient depending on the structure of the matrix under consideration.

The overall architecture of a multivariate Gaussian random number generator is constructed from a combination of CBs. An instance is shown in Figure 1, where five blocks are used with different precisions namely $p_1$, $p_2$, $p_3$, $p_4$, and $p_5$. In the figure, $\mathbf{c}^i$ denotes the vector coefficient at $i^{th}$ decomposition level while $z$ denotes the univariate Gaussian random numbers. The above precisions refer to the word-length of the coefficients. The word-length of $z$ is fixed to a user-specified precision. The precision in the adder path, which runs through all of the computation blocks is fixed to $p_t$, the maximum precision out of the precisions of all CBs used. The GRNG block denotes the univariate Gaussian random number generator which produces $z$, the univariate Gaussian random samples. The memory block is used to store the coefficients. The memory block can be instantiated as either block RAMs or registers. In this work, registers are deployed for small matrix order ($N < 20$) while block RAMs are used for larger matrix order.

In order for an improved throughput performance to be achieved, the operation is pipelined so that all the computational blocks operate in parallel. As far as the number representation is concerned, fixed-point precision is used throughout the entire design. Fixed-point arithmetic produces designs that consume fewer resources and operate at a higher frequency in comparison to floating-point arithmetic. The eigenvalue decomposition algorithm also provides an ordering of the coefficients according to their dynamic range. This has been exploited by the proposed algorithm in order for the resulting design to perform computations between intermediate results of similar dynamic range. A significant advantage of this is that the error due to truncation operations is minimized, as it will be demonstrated later in this article.

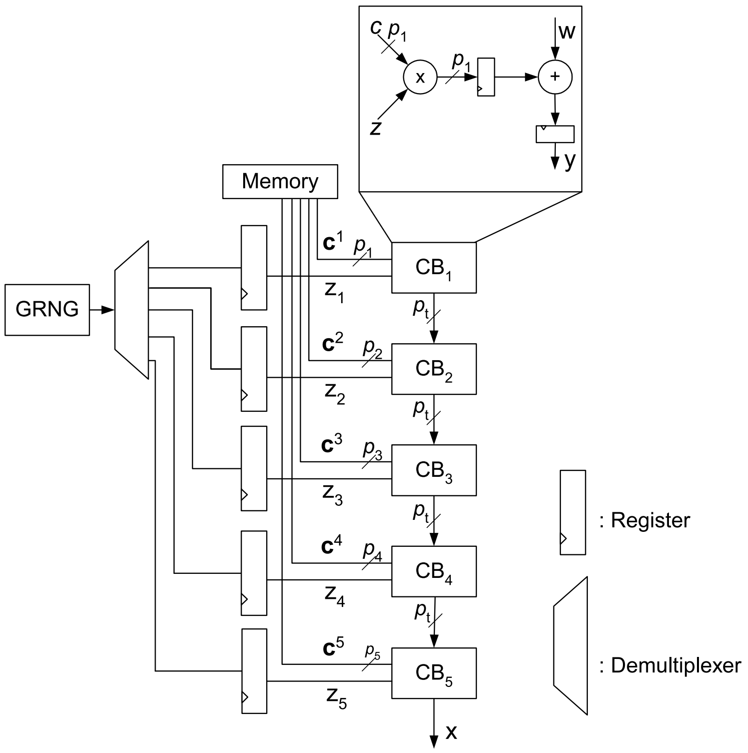


Fig. 1. Hardware architecture.

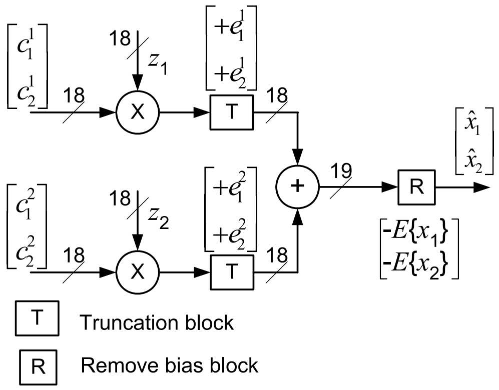


Fig. 2. An architecture for a 2D MVGRNG.

## 5. ANALYSIS OF APPROXIMATION ERROR

Let us consider a 2D Gaussian random number generator. An instance of the hardware architecture required to generate a 2-dimensional vector $\mathbf{x}$ is illustrated in Figure 2 where two multipliers and an adder are utilized to map the expression in (2) to generate random samples. Due to the use of fixed point number representation in this work, the coefficients of the covariance matrix are quantized to a certain precision. In this article, the following notation is used. $c_j^k$ denotes a coefficient in a vector $\mathbf{c}$, where $j$ denotes the position of that coefficient in the vector and $k$ denotes the decomposition level. Hence, $c_1^1$ and $c_2^1$ are the quantized coefficients that correspond to the first decomposition level while $c_1^2$ and $c_2^2$ are the quantized coefficients of the second decomposition level. The numbers on the datapaths denote the word-length employed. “T” represents a truncation block, where the word-length of the signal is truncated to the specified precision, while “R” is a block which subtracts a bias from the generated samples. The details regarding this block will be discussed later.

The proposed system generates a random vector $\mathbf{x}$ from univariate random samples $z_1, z_2 \sim N(0,1)$ as described in (5). Note that the truncation of the datapath introduces an error $e$ to the system.

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \left( \begin{bmatrix} c_1^1 \\ c_2^1 \end{bmatrix} z_1 + \begin{bmatrix} e_1^1 \\ e_2^1 \end{bmatrix} \right) + \left( \begin{bmatrix} c_1^2 \\ c_2^2 \end{bmatrix} z_2 + \begin{bmatrix} e_1^2 \\ e_2^2 \end{bmatrix} \right). \tag{5}$$

The expected value of the random vectors $\mathbf{x}$ in (5) is:

$$\begin{aligned} E\left\{ \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right\} &= \begin{bmatrix} E\{c_1^1 z_1 + e_1^1 + c_1^2 z_2 + e_1^2\} \\ E\{c_2^1 z_1 + e_2^1 + c_2^2 z_2 + e_2^2\} \end{bmatrix} \\ &= \begin{bmatrix} E\{e_1^1\} + E\{e_1^2\} \\ E\{e_2^1\} + E\{e_2^2\} \end{bmatrix}, \end{aligned} \tag{6}$$

since $E\{z\}$ is zero. The expression in (6) implies that there is a bias in the generated samples due to the truncation operation. Thus, the mean of the generated samples is no longer zero. This bias must be removed before outputting the final samples. We denote the samples with zero mean as $\hat{\mathbf{x}}$ which are expressed as:

$$\begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} E\{e_1^1\} + E\{e_1^2\} \\ E\{e_2^1\} + E\{e_2^2\} \end{bmatrix}. \tag{7}$$

This operation is realized through block “R” in Figure 2. The covariance matrix $\overline{\boldsymbol{\Sigma}}$ of the generated samples $\hat{\mathbf{x}}$ is given in (8).

$$\begin{aligned} \overline{\boldsymbol{\Sigma}} &= E\{\hat{\mathbf{x}}\hat{\mathbf{x}}^T\} - E\{\hat{\mathbf{x}}\}E\{\hat{\mathbf{x}}\}^T \\ &= E\left\{ \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix}^T \right\} \\ &= \begin{bmatrix} E\{\hat{x}_1^2\} & E\{\hat{x}_1\hat{x}_2\} \\ E\{\hat{x}_2\hat{x}_1\} & E\{\hat{x}_2^2\} \end{bmatrix}. \end{aligned} \tag{8}$$

The nondiagonal elements in the covariance matrix is given by the expectation $E\{\hat{x}_1\hat{x}_2\}$ as in (9).

$$\begin{aligned} E\{\hat{x}_1\hat{x}_2\} &= c_1^1 c_2^1 + c_1^2 c_2^2 + E\{e_1^1 e_2^1\}E\{e_1^2 e_2^2\} - E\{e_1^1\}E\{e_2^1\} - E\{e_1^2\}E\{e_2^2\} \\ &= E\{x_1 x_2\}_{input} + \rho_{e_1^1 e_2^1}\sigma_{e_1^1}\sigma_{e_2^1} + \rho_{e_1^2 e_2^2}\sigma_{e_1^2}\sigma_{e_2^2}, \end{aligned} \tag{9}$$

where $E\{x_1x_2\}_{input}$ denotes the input targeted covariance between $\hat{x}_1$ and $\hat{x}_2$ and $\rho_{e_1^1e_2^1}$ and $\rho_{e_1^2e_2^2}$ are the correlation between $e_1^1$ and $e_2^1$, and between $e_1^2$ and $e_2^2$ respectively. The variance of the error inserted to the system due to truncation of a signal $(p_1, d)$ to a signal $(p_2, d)$ is given by (10).

$$\sigma_e^2 = \frac{1}{12}2^{2d}(2^{-2p_2} - 2^{-2p_1}), \tag{10}$$

where $p$ denotes the word-length of the signal and $d$ refers to its scale [Constantinides et al. 2004]. In addition, $E\{e\}$ is given by $-2^{d-1}(2^{-p_2} - 2^{-p_1})$. Similarly, the diagonal elements of the sample covariance matrix can be expressed as $E\{\hat{x}_1^2\} = E\{x_1^2\}_{input} + \sigma^2_{e_1^1} + \sigma^2_{e_2^1}$ and $E\{\hat{x}_2^2\} = E\{x_2^2\}_{input} + \sigma^2_{e_1^2} + \sigma^2_{e_2^2}$. Note that the above analysis does not take into account the truncation operations in the adder path of the architecture. Thus the covariance matrix of the generated samples is given in (11).

$$\overline{\mathbf{\Sigma}} = \mathbf{C}_K + \mathbf{W}, \tag{11}$$

where $\mathbf{C}_K$ is a matrix that approximates the input covariance matrix $\mathbf{\Sigma}$ using $K$ levels of decomposition taking into consideration the quantization effects and $\mathbf{W}$ is an expected truncation error matrix. In general, for any given covariance matrix of order $N$, an approximation using $K$ decomposition levels leads to an expected truncation error matrix $\mathbf{W}$ where its element $w_{i,j}$ is given in (12). Note that $w_{i,j}$ is an error due to the correlation of truncation errors from samples $x_i$ and $x_j$.

$$w_{i,j} = \sum_{k=1}^{K} \rho_{e_i^k e_j^k}\sigma_{e_i^k}\sigma_{e_j^k} \tag{12}$$

For the diagonal elements of $\mathbf{W}$, $w_{i,i}$, the correlation $\rho_{e_i^k e_i^k}$ is one. Hence, the expression in (12) is simplified to $\sum_{k=1}^{K}\sigma^2_{e_i^k}$. The expression in (12) illustrates that the correlation of the truncation errors of each computational block contributes to the final approximation error of the covariance matrix.

Throughout this section, we have focused on the discussion of the error in the approximation of the covariance matrix due to truncation operations in the datapath. An alternative to a truncation operation is rounding. In this case, the same error analysis applies but with the exception of the bias being zero.

### 5.1 Impact of Correlation of Truncation Errors

Let us consider two coefficients $c_1$ and $c_2$ that belong to the same vector of the same decomposition level. In the proposed architecture, the coefficients $c_1$ and $c_2$ are multiplied with $z$ in order to generate the samples $x_1$ and $x_2$. Since the coefficients are fixed for the given distribution of interest, a correlation is introduced between the truncation errors from the two computational paths, which is a function of the coefficients, $\rho(c_1, c_2)$. The two truncation errors are described in (13)

$$\begin{aligned} e_1 &= c_1z - [c_1z]_{tr} \\ e_2 &= c_2z - [c_2z]_{tr}, \end{aligned} \tag{13}$$

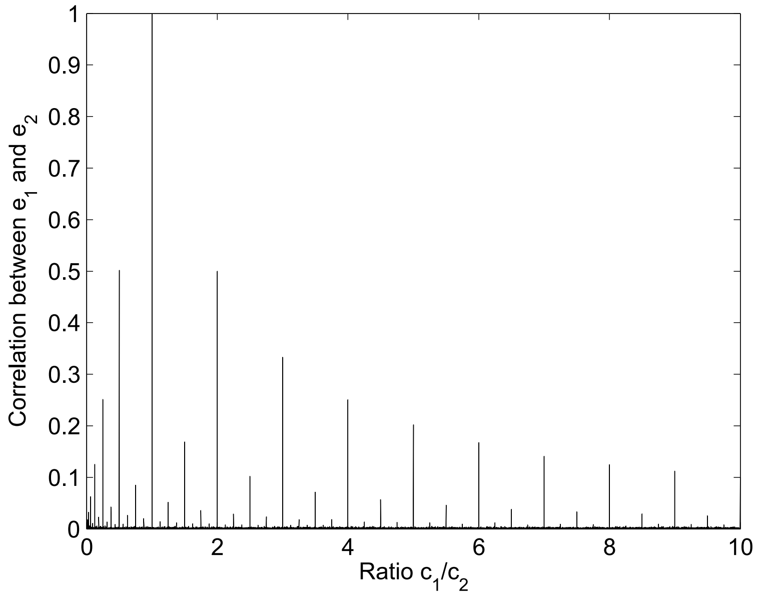


Fig. 3. Correlation between truncation errors.

where $[.]_{tr}$ denotes the truncated version of the expression. The correlation $\rho$ is independent of any scaling of the random sequence $z$. By multiplying the sequence of random samples $z$ by $\alpha$, the expressions of the truncation errors are given in (14)

$$\begin{aligned} e'_1 &= c_1(\alpha z) - [c_1(\alpha z)]_{tr} = (\alpha c_1)z - [(\alpha c_1)z]_{tr} \\ e'_2 &= c_2(\alpha z) - [c_2(\alpha z)]_{tr} = (\alpha c_2)z - [(\alpha c_2)z]_{tr}, \end{aligned} \tag{14}$$

having the same correlation as before. Thus,

$$\rho(\alpha c_1, \alpha c_2) = \rho(c_1, c_2). \tag{15}$$

We can set $\alpha$ to be $\frac{1}{c_2}$, resulting to $\rho(c_1, c_2) = \rho(\frac{c_1}{c_2}, 1) = \rho(\frac{c_1}{c_2})$, which implies that the correlation is a function of the ratio of the coefficients, and not of their individual values.

Figure 3 illustrates the correlation between the truncation errors $e_1$ and $e_2$ for different values of the ratio $\frac{c_1}{c_2}$. The data has been collected through computer simulations. In this graph, the relationship between the coefficients is expressed as a ratio of the two coefficients under consideration in order to construct a generic reference graph for all possible value of coefficients. The corresponding p-values are under the 0.05 threshold.

## 6. PROPOSED METHODOLOGY

In Section 4, a hardware architecture based on the Eigenvalue decomposition algorithm of the covariance matrix has been introduced, and an error analysis has been performed in Section 5. In this section, the methodology of mapping such architecture to reconfigurable hardware is described.

In this work we have focused on the use of mean square error of the approximation error between the input covariance matrix $\mathbf{\Sigma}$ and the sample covariance matrix. However, the proposed approach can be generalized to optimize other metrics as well. Considering the mean square error as our target metric, we propose three different approaches for the selection of appropriate coefficients for each decomposition level. The selection criteria in Algorithm 1 is based on the minimization of the quantization error only. In Algorithm 2, the coefficients are selected with respect to the minimization of the overall approximation error which takes into account the quantization and the truncation error. In Algorithm 3 an optimization technique is introduced, which minimizes the correlation between the truncation errors leading to an optimized total approximation error. All the proposed algorithms are iterative. In each iteration, a new decomposition level in the matrix approximation is introduced, and its coefficients are estimated.

## 6.1 Algorithm 1

Algorithm 1 selects the appropriate coefficients based on the minimization of the quantization error only. The objective function in each decomposition level is given in (16).

$$f_1(\mathbf{c}_K) = \frac{1}{N^2}\|\mathbf{R}_{K-1} - (\mathbf{c}_K\mathbf{c}_K^T)\|^2, \tag{16}$$

where the $\mathbf{R}_{K-1}$ denotes the remaining of the original covariance matrix after $K-1$ levels of decomposition, where $\mathbf{R}_0$ is defined as the original covariance matrix at the start of the algorithm. Hence, the objective function in Algorithm 1 optimizes for the mean square error of the difference between the remaining portion of the original covariance matrix and the estimated covariance matrix including $K$ decomposition levels. It is important to note that the objective function of the optimization in this algorithm does not take into account the error due to the truncation operations. The approximation error is due to the quantization of the coefficients only.

## 6.2 Algorithm 2

In Algorithm 2, the overall approximation error of the covariance matrix, which consists of quantization and truncation errors is taken into account. The objective function for the minimization of the approximation error is expressed as in (17).

$$f_2(\mathbf{c}_K) = \frac{1}{N^2}\|\mathbf{R}_{K-1} - (\mathbf{c}_K\mathbf{c}_K^T + \mathbf{W})\|^2, \tag{17}$$

where $\mathbf{W}$ denotes the expected error due to truncation/rounding operations in the datapath. Similar to Algorithm 1, $\mathbf{R}_{K-1}$ denotes the remaining of the original covariance matrix after $K-1$ levels of decomposition. In each iteration the algorithm selects the coefficients that minimize the approximation error of the original covariance matrix with respect to the estimated sample covariance matrix, taking into account the correlation due to truncation/rounding operations.

## 6.3 Algorithm 3

In order to combat the effects of error due to the truncation in datapath, an optimization technique is introduced in Algorithm 3. Similar to Algorithm 2, the main objective of this algorithm is to select the coefficients based on the minimization of the quantization error and the expected truncation error. The added feature is that at each decomposition level the algorithm searches for the coefficients which minimize the overall quantization and the expected truncation errors. The starting point of the search is given by the coefficients obtained after the eigenvalue decomposition algorithm which only minimize the quantization error. Then, the algorithm optimizes the error in the approximation of the original covariance matrix by minimizing the objective function in (18).

$$f_3(\mathbf{c}_K) = \frac{1}{N^2}\|\mathbf{R}_{K-1} - (\mathbf{c}_K\mathbf{c}_K^T + \mathbf{W}(\rho(\mathbf{c}_K)))\|^2, \tag{18}$$

where the expected truncation error $\mathbf{W}$ is a function of the correlation between the truncation errors of any pair of coefficients in the vector $\mathbf{c}_K$. The optimization takes into account the information depicted in Figure 3 in order to find the coefficients that minimize (18). In summary, the algorithm tries to steer the coefficients away from the peaks, which represent high correlations, minimizing at the same time the overall matrix approximation under the mean square error metric.

## 6.4 Overview of Proposed Approach

The objective of this work is to generate random samples from a multivariate Gaussian distribution using a hardware architecture implemented on an FPGA. Similar to the existing approaches, the elements of each random vector are serially generated resulting to a throughput of one vector per $N$ clock cycles for an $N$ dimensional Gaussian distribution.

The main feature of the proposed methodology is its ability to exploit different precision requirements for various parts of the system in order to achieve a design that has the least error in the approximation of the input covariance matrix $\mathbf{\Sigma}$ under the mean square error metric and at the same time the least area requirements.

Since the full exploration of all possible precision requirements for each computational path is computationally expensive, the proposed approach exploits a subset of the possible designs by introducing a library containing predefined hardware computational blocks with different user-specified precisions for the implementation of each computational path. Therefore, there is a trade-off between the complexity of the search stage and the optimality of the obtained result. The objective of the exploration is to determine the best combination of CBs in order to construct a design which has the minimum approximation error with the minimum resource usage.

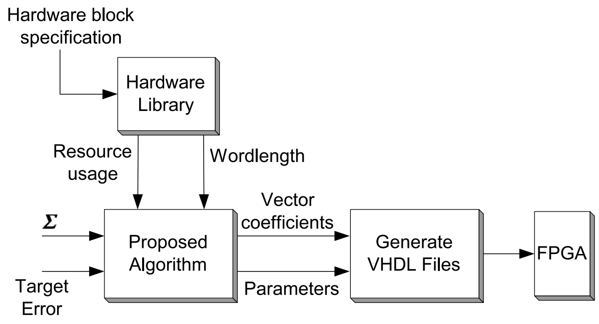


Fig. 4. Overview of the proposed approach.

In order to visualize the full functionality of the proposed approach, the overall structure of this methodology is illustrated in Figure 4, in which there are two main processes. The first process is the proposed algorithm which decomposes any given covariance matrix for a specified approximation error. The hardware library contains information regarding the predefined hardware blocks including the word-length and the corresponding resource usage of each block. Finally, the second process generates VHDL files based on the information acquired from the first process together with the specification from the hardware library. The entire process is fully automated and parameterizable to accommodate any type of resources and any precision requirements.

### 6.5 Overall Methodology

In this section, an overall description of the proposed methodology is given, which is depicted in Figure 5. The inputs to the system are a covariance matrix $\mathbf{\Sigma}$, the allowed resource usage (in LUTs), a set of types of CBs that are available along with resource usage information for each CB and the word-length of its datapath, and finally the target approximation error between the original input matrix and the expected covariance matrix. Note that the proposed methodology is fully parameterized such that it is able to support any number of predefined hardware blocks.

The proposed methodology is divided into three stages. In the first stage, the covariance matrix $\mathbf{\Sigma}$ is decomposed into a vector $\mathbf{c}$ and its transpose. Then, the vectors are converted into fixed point representation using one of the predefined available word-lengths. The effect of decomposing $\mathbf{\Sigma}$ and mapping it into different precision hardware blocks gives rise to a number of designs for exploration.

The second stage of the algorithm estimates the approximation error for all of the possible designs using the three proposed algorithms. Note that the possible values of the coefficients are determined by the precisions of the CBs during the optimization of the objective functions in all algorithms. In the third stage of the algorithm, any inferior designs are discarded. An inferior design is a design which, in comparison to another design, uses more hardware resources but produces worse approximation error. Thus, the proposed algorithm

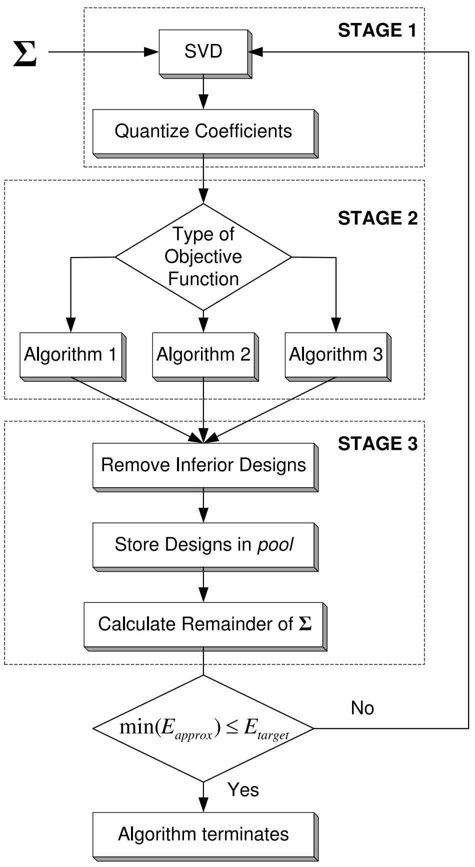


Fig. 5. Outline of the proposed algorithm.

is optimizing for the $ApproximationError - Area$ design space. In order to further tune the required search time, a parameter called "pool size" is introduced. A pool size indicates the maximum number of designs to be kept from one iteration to the next during the execution of the algorithm. The last step of this stage is to calculate the remainder of the original matrix $\mathbf{\Sigma}$ which will be used as a starting matrix in the next iteration. These steps are repeated until the design with the minimum approximation error meets the target value specified by the user.

## 7. PERFORMANCE EVALUATION

The designs generated from the proposed methodology have been implemented on a Stratix III EP3SE50F484C2 FPGA from Altera and Quartus II was utilized as a hardware synthesis tool. With regard to the word-length of the univariate samples $z$, a fixed 18 bits precision is allocated throughout the entire design. $z$ is an 18-bit signed number with 3 bits dedicated for the integer part and 14 bits dedicated for the decimal part. Hence, approximately 99.7% of the dynamic range of a standard normal distribution can be represented.

Table I. Computational Block Synthesis Results

| Word-length | Resource Usage (LUTs) |
|---|---|
| 4 | 115 |
| 9 | 206 |
| 15 | 300 |
| 18 | 332 |

## 7.1 Library Construction

The set of computational blocks (CBs) that is used in the proposed framework is chosen in order to cover the range between 4 to 18 bits precision in an almost uniform way. Four CBs with 4, 9, 15, and 18 bits precision have been predefined in the hardware library for the construction of the multivariate Gaussian random number generator. Table I shows the resource utilization obtained from the synthesis stage from Quartus II. These results are used by the proposed algorithm in order to optimize the architecture. As expected, the number of LUTs increases linearly as the precision increases. In order to make a direct comparison between the proposed approach and Thomas and Luk [2008b] and Saiprasert et al. [2008], DSP48 functionality has been mapped to logic on the same FPGA device. Note that all the generated designs have been successfully synthesized to a frequency in the range of 380 to 420 MHz.

## 7.2 Evaluation of Proposed Methodology

In this section we evaluate the performance of the proposed methodology in comparison to the existing approaches from the literature. The performance of the three proposed algorithms is assessed, and it is compared against the state-of-the-art approaches in Thomas and Luk [2008b] and Saiprasert et al. [2008].

The five algorithms are applied to a randomly generated 5x5 covariance matrix. The generated architectures are synthesized and 100,000 vectors of multivariate Gaussian numbers are produced from each of these designs in order to construct the corresponding sample covariance matrices.

The plot in Figure 6 illustrates the mean square error in the approximation of the original covariance matrix and the sample covariance matrices for a range of designs generated by the five algorithms. From the plot it is important to note that the three proposed algorithms and the approach in Saiprasert et al. [2008] have the flexibility to produce designs across all of the available design space. In contrast, the approach in Thomas and Luk [2008b] can only produce one fixed design for a given covariance matrix. This is due to the decomposition technique utilized in the proposed approach and the approach in Saiprasert et al. [2008].

Let us consider the relative performance between the three proposed algorithms and the approach in Thomas and Luk [2008b]. The plots show that, for similar resource utilization, the designs generated from Algorithms 1, 2, and 3 produce a better approximation error than Thomas and Luk [2008b], achieving a reduction in the approximation error by 91%, 94%, and 97% respectively. Moreover, for similar values of approximation error in the covariance matrix,

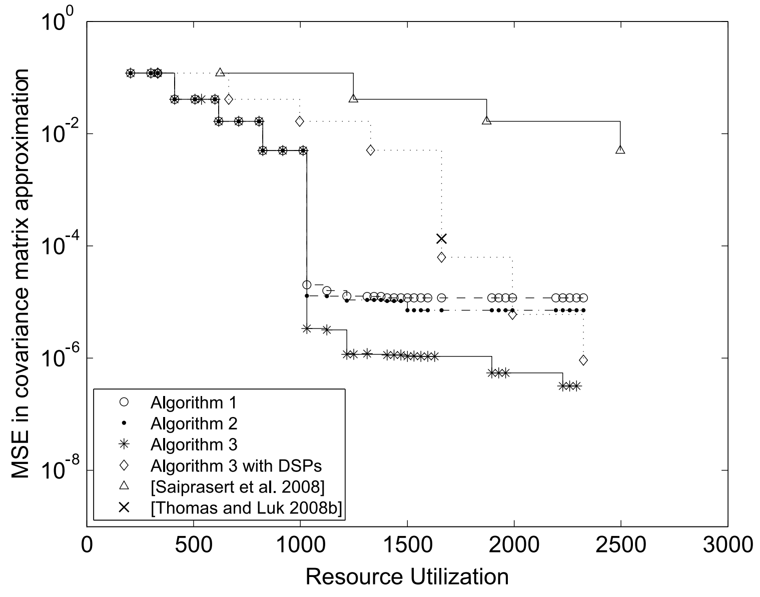


Fig. 6. Comparison of proposed algorithms with existing approaches.

there is a 60% reduction in hardware resource utilization relative to Thomas and Luk [2008b] for all three algorithms.

Moreover, the plot demonstrates that the hardware resource utilization of the three proposed algorithms are significantly reduced in comparison to the approach in Saiprasert et al. [2008]. This is mainly due to the use of the eigenvalue decomposition algorithm to decompose the input covariance matrix instead of decomposing the resulting matrix from the Cholesky decomposition, which reduces the number of multiplication by half and also the application of word-length optimization techniques in the design of the architectures.

As expected, Algorithm 3 performs better than Algorithm 1 and Algorithm 2 with respect to the approximation error of the covariance matrix. The error plots of the three algorithms remain approximately the same for designs smaller than 1000 LUTs. However, after this point Algorithm 3 generates the best designs which produce significantly less approximation error. This demonstrates that the technique of selecting the coefficients by minimizing the objective function in Algorithm 3, which encapsulates the correlation between the truncation errors, minimizes the overall error of the system leading to an improved design performance. The reason for a significant improvement after 1000 LUTs is that at this point the number of decomposition level reaches the rank of the matrix under consideration. Further decomposition levels are required due to the effects of coefficient quantization and correlation of the truncation operations in the datapath. This will be discussed in more detail in the next section.

In order to make a direct comparison between the proposed approach and the approach in [Thomas and Luk 2008b], Algorithm 3 has been applied to

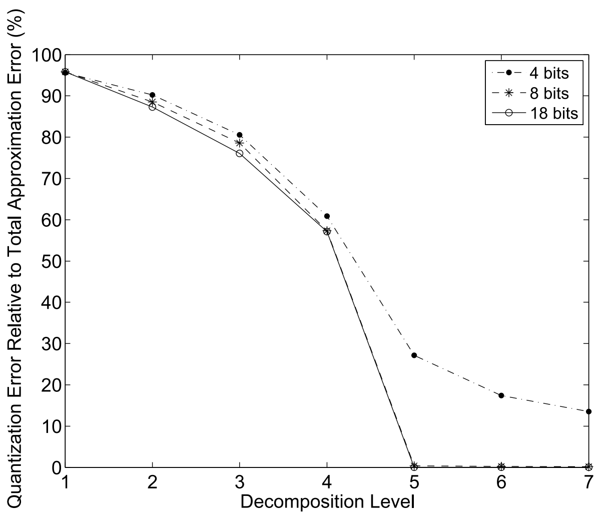


Fig. 7. Error distribution.

the same 5x5 matrix but utilizing only DSP blocks to map the computation paths. The result is another set of architectures plotted on the graph where it can be observed that for the same number of utilized DSP blocks, Algorithm 3 obtains better approximation error than the approach in Thomas and Luk [2008b] by approximately 53%. This is because the correlation between the truncation errors of each DSP is taken into account during the selection of the coefficients.

### 7.3 Error Distribution

The purpose of this experiment is to investigate the distribution of the two error domains, quantization and correlation of the truncation errors, over a range of designs for the 5x5 covariance matrix from Section 7.2. Figure 7 shows a plot of the distribution of the quantization error relative to the total approximation error for designs generated from Algorithm 3 over 7 decomposition levels using fixed precision computational blocks with 4, 8, and 18 bits respectively. It can be seen that the quantization error drops as the number of decomposition level increases. The quantization error is dominant (more than 50%) for the first 4 decomposition levels while the truncation error becomes dominant after the fifth level of decomposition. This is due to the fact that the decomposition level reaches the rank of the matrix. In addition, the drop in quantization error as the number of decomposition level increases is steeper when higher precision is allocated in the datapaths. It is important to note that since fixed-point precision is deployed in this work, the quantization error does not reach zero after reaching a decomposition level equal to the rank of the matrix. Thus, after having reached this decomposition level in the approximation, the overall approximation error can be minimized by targeting the error due correlation of the truncation errors in the datapath.

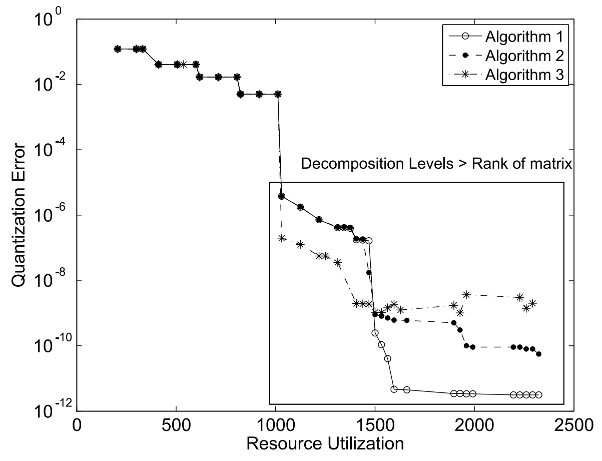


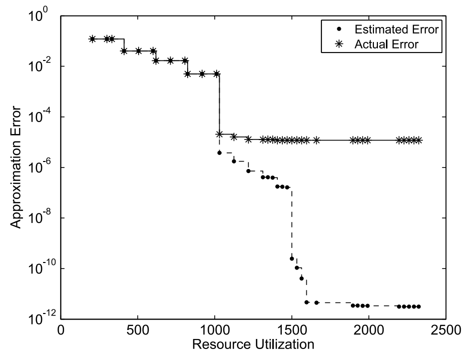
Fig. 8. Comparison of three algorithms.

## 7.4 Effects of Coefficient Optimization

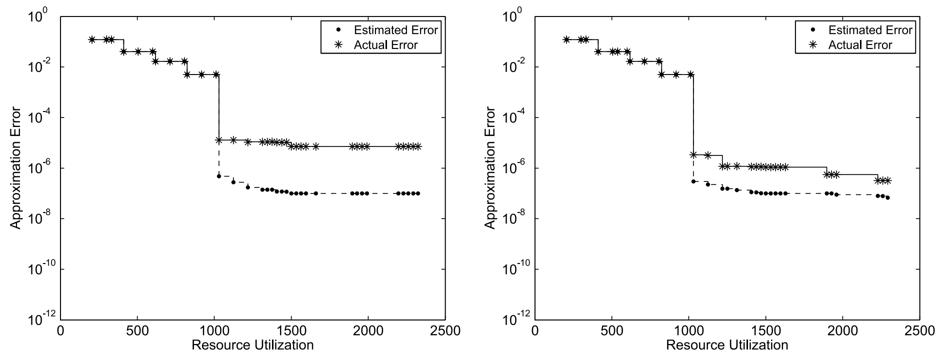
Figure 8 illustrates a plot of the quantization error of the same 5x5 matrix as in Section 7.2 for a set of designs generated using the three algorithms in the proposed methodology. The plot of Algorithm 1 behaves as expected since the approximation error decreases as we add more computational blocks to approximate the original matrix. However, a different trend is observed for Algorithm 3 after the number of decomposition levels has reached the rank of the input covariance matrix. In this case, the approximation error does not decrease as anticipated. This is due to the fact that the coefficients are selected in a way to minimize the impact of the correlation of the truncation errors in the sample covariance matrix. Algorithm 2 may exhibit the same behaviour as Algorithm 3.

## 7.5 Analysis of Actual Approximation Error

This section focuses on the comparison between the estimated error of each of the generated architectures from the three proposed algorithms with the empirical error obtained from the sample covariance matrix using 100,000 vectors. Similar to previous experiments, this is the result of the same 5x5 covariance matrix as in Section 7.2. The plots in Figure 9 show the comparison between the estimated and the actual approximation errors from the three proposed algorithms. As expected for Algorithm 1, the gap between the predicted error by the algorithm and the actual error is very large as the objective function does not model the error due to truncation operations. The estimation of approximation error in Algorithm 2 and Algorithm 3 is similar since both algorithms model the impact of the correlation due to the truncation in the sample covariance matrix providing a better estimate of the final error. Algorithm 3 provides the best estimation out of the three algorithms due to the use of the optimization techniques in order to select the coefficients taking into account the correlation between errors due to truncation operations. It can be observed from



(a) Algorithm 1



(b) Algorithm 2 (c) Algorithm 3

Fig. 9. Comparison between estimated and actual error.

Figure 9(b) and 9(c) that there are still some deviation between the estimated and the actual error despite the inclusion of truncation error in the optimization criteria. The gap in these plots suggests that the remaining error is due to other factors which the proposed approach has not yet modeled, which will be part of the future work.

## 8. CONCLUSION

This article has presented a novel approach to construct hardware architectures to generate random samples from a multivariate Gaussian distribution. The proposed approach is based on the eigenvalue decomposition technique where the computation path is decomposed into paths with different precision requirements in order to minimize the resource usage with minimal impact to the quality of the random samples. An analysis of the error due to the truncation/rounding operations along the datapath has been presented, and an expression that models the overall error inserted into the system is derived. As a result, three algorithms are proposed to minimize the error inserted into the system. Experimental results have demonstrated that by dedicating appropriate precisions to different computational paths, the error at the output of the system and the resource usage can be minimized. Moreover, this work demonstrates the impact of the correlation of the truncation/rounding operations to the output of the system and provides a methodology to minimize it.

For future work we plan to investigate alternative metrics as the objective function in order to select appropriate coefficients for the hardware architecture, and to model the remaining sources of error in the system.

## REFERENCES

BOUGANIS, C.-S., PARK, S.-B., CONSTANTINIDES, G. A., AND CHEUNG, P. Y. K. 2009. Synthesis and optimization of 2d filter designs for heterogeneous FPGAs. *ACM Trans. Reconfig. Technol. Syst. 1*, 4, 1–28.

BOX, G. E. P. AND MULLER, M. E. 1958. A note on the generation of random normal deviates. *Annals Math. Stat. 29*, 2, 610–611.

CHAN, N. H. AND WONG, H. Y. 2006. *Simulation Techniques in Financial Risk Management*. Wiley.

CONSTANTINIDES, G. A., CHEUNG, P. Y. K., AND LUK, W. 2004. *Synthesis and Optimization of DSP Algorithms*. Kluwer Academic Publishers, Norwell, MA.

GLASSERMAN, P. 2004. *Monte Carlo Methods in Financial Engineering*. Springer.

GLASSERMAN, P. AND LI, J. 2003. Importance sampling for portfolio credit risk. *Manage. Sci. 51*, 1643–1656.

GLASSERMAN, P., HEIDELBERGER, P., AND SHAHABUDDIN, P. 2000. Variance reduction techniques for value-at-risk with heavy-tailed risk factors. In *Proceedings of the 32nd Conference on Winter Simulation*. 604–609.

KAGANOV, A., CHOW, P., AND LAKHANY, A. 2008. FPGA acceleration of Monte-Carlo based credit derivative pricing. In *Proceedings of the IEEE International Conference on Field Programmable Logic and Applications*. 329–334.

LEE, D.-U., LUK, W., VILLASENOR, J. D., ZHANG, G., AND LEONG, P. H. 2005. A hardware Gaussian noise generator using the Wallace method. *IEEE Trans. VLSI Syst. 13*, 911–920.

LEE, D.-U., VILLASENOR, J. D., LUK, W., AND LEONG, P. H. W. 2006. A hardware Gaussian noise generator using the Box-Muller method and its error analysis. *IEEE Trans. Comput. 55,* 6, 659–671.

MARSAGLIA, G. AND TSANG, W. W. 2000. The ziggurat method for generating random variables. *J. Statist. Softw. 5*, 8, 1–7.

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. 1992. *Numerical Recipes in C*. Cambridge University Press.

SAIPRASERT, C., BOUGANIS, C.-S., AND CONSTANTINIDES, G. A. 2008. Multivariate Gaussian random number generator targeting specific resource utilization in an FPGA. In *Proceedings of the 4th International Workshop on Reconfigurable Computing (ARC'08)*. Springer-Verlag, 233–244.

SAIPRASERT, C., BOUGANIS, C.-S., AND CONSTANTINIDES, G. A. 2009. Word-length optimization and error analysis of a multivariate Gaussian random number generator. In *Proceedings of the 5th International Workshop on Reconfigurable Computing (ARC'09)*. Springer-Verlag, 231–242.

THOMAS, D. B. AND LUK, W. 2008a. Credit risk modelling using hardware accelerated Monte-Carlo simulation. In *Proceedings of the IEEE International Symposium on Field-Programmable Custom Computing Machines*. 229–238.

THOMAS, D. B. AND LUK, W. 2008b. Multivariate Gaussian random number generation targeting reconfigurable hardware. *ACM Trans. Reconfig. Technol. Syst. 1*, 2, 1–29.

THOMAS, D. B., BOWER, J. A., AND LUK, W. 2007. Automatic generation and optimization of reconfigurable financial Monte-Carlo simulations. In *Proceedings of the IEEE International Conference on Application-Specific Systems Architectures and Processors*. 168–173.

THOMAS, D. B., LUK, W., LEONG, P. H., AND VILLASENOR, J. D. 2007. Gaussian random number generators. *ACM Comput. Surv. 39*, 4, 11.

THOMAS, D. B., HOWES, L., AND LUK, W. 2009. A comparison of CPUs, GPUs, FPGAs, and massively parallel processor arrays for random number generation. In *Proceeding of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA'09)*. ACM, New York, 63–72.

WALLACE, C. S. 1996. Fast pseudorandom generators for normal and exponential variates. *ACM Trans. Math. Soft. 22*, 1, 119–127.

WOODS, N. A. AND VANCOURT, T. 2008. FPGA acceleration of quasi-Monte Carlo in finance. In *Proceedings of the IEEE International Conference on Field Programmable Logic and Applications*. 335–340.

ZHANG, G., LEONG, P. H., LEE, D.-U., VILLASENOR, J. D., CHEUNG, R. C., AND LUK, W. 2005. Ziggurat-based hardware Gaussian random number generator. In *Proceedings of the IEEE International Symposium on Field Programmable Logic and Applications*. 275–280.

Received April 2009; accepted June 2009