# Word-length Optimization and Error Analysis of a Multivariate Gaussian Random Number Generator

Chalermpol Saiprasert, Christos-Savvas Bouganis and George A. Constantinides

Department of Electrical & Electronic Engineering, Imperial College London
Exhibition Road, London SW7 2BT, United Kingdom.
{cs405,christos-savvas.bouganis,g.constantinides}@imperial.ac.uk

**Abstract.** 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 paper presents a novel approach for mapping a multivariate Gaussian random number generator onto an FPGA by automatically optimizing the computational path with respect to the resource usage. The proposed approach is based on the Eigenvalue decomposition algorithm which decomposes the design into computational paths with different precision requirements. Moreover, an error analysis on the impact of the error due to truncation is performed in order to provide upper bounds of the error inserted into the system. The proposed methodology optimises the usage of the available FPGA resources leading to area efficient designs without any significant penalty on the overall performance. Experimental results reveal that the hardware resource usage on an FPGA is reduced by a factor of two in comparison to current methods.

**Key words:** Multivariate Gaussian Distribution; Word-length optimization; FPGA

## 1 Introduction

Many stochastic processes in scientific experiments and mathematical analysis are modeled using Stochastic Differential Equations (SDE). In order to determine the solution to these stochastic processes, time consuming and computationally expensive simulations are employed. Financial mathematics is one of many fields that heavily relies on the simulation of SDE in which many financial instruments can be modeled. Examples can be found in value-at-risk [1] and credit-risk calculation [2]. The Monte Carlo method is one of the most widely used techniques to carry out these simulations [3]. At the heart of 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 produced from a variety of distributions, and one of the most commonly in use is the multivariate Gaussian distribution.

In recent years there is an increasingly high demand for computationally intensive calculations in finance due to the ever increasing 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) and Field Programmable Gate Array (FPGA) 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 literature concerning hardware acceleration of financial applications include the calculation of Monte Carlo based credit derivative pricing [4] and interest rates simulation [5]. In order to maximize the performance of the system, many researches have focused on the mimization of the hardware resources occupied by the random number generator. In the case of FPGA based Multivariate Gaussian random number generator, two pieces of work have been published so far [6], [7]. The approach in [6] is capable of producing samples at a high throughput but it lacks the flexibility to meet any given resource constraints. This has been addressed by the technique in [7] which offers the flexibility to accommodate a range of resource constraints when certain conditions are met.

The presented work differs from the previous published works as follows:

– It proposes a novel methodology that minimizes the resource usage of a multivariate Gaussian random number generator based on the Eigenvalue decomposition algorithm which decomposes the design into computational paths with different precision requirements. As a result, the number of hardware resource usage is reduced by up to a factor of two in comparison with existing techniques.
– An error analysis on the impact of the error due to truncation in the datapath is performed providing upper bounds of the error inserted into the system. To the best of the authors knowledge, this error analysis has not been previously addressed in the literature in the case of multivariate Gaussian random number generator.

## 2 Background

A multivariate Gaussian random distribution is defined by two parameters, its mean denoted by $\mathbf{m}$ and its covariance matrix $\boldsymbol{\Sigma}$. 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 Eigenvalue decomposition technique.

Under the Cholesky Factorization technique, $\boldsymbol{\Sigma}$ is decomposed using Cholesky decomposition into a product of a lower triangular matrix $\mathbf{A}$ and its transpose [3]. Then, the required samples are generated through a linear combination of univariate Gaussian samples $\mathbf{z}$ that follow a standard Gaussian distribution $N(0, 1)$.

Due to the lower triangular property of matrix $\mathbf{A}$, the number of computations are reduced by half 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 [6].

An alternative method is to decompose $\boldsymbol{\Sigma}$ by Eigenvalue decomposition [8] and using the Singular Value Decomposition algorithm (SVD). Since an $N \times N$ covariance matrix $\boldsymbol{\Sigma}$ is symmetric and semi-positive definite, all of its eigenvalues $\lambda$ are non-negative and its eigenvectors $\mathbf{u}$ are orthogonal. SVD expresses a matrix as a linear combination of three separable matrices [9]. Hence using SVD, $\boldsymbol{\Sigma}$ can be expressed as $\boldsymbol{\Sigma} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$ where $\mathbf{U}$ is an orthogonal matrix ($\mathbf{U}\mathbf{U}^T = \mathbf{I}$) containing columns $\mathbf{u}_1, ..., \mathbf{u}_N$ while $\boldsymbol{\Lambda}$ is a diagonal matrix with diagonal elements being $\lambda_1, ..., \lambda_N$. By letting $\mathbf{A} = \mathbf{U}\boldsymbol{\Lambda}^{1/2}$, multivariate Gaussian random samples that follow $N(\mathbf{m}, \boldsymbol{\Sigma})$ can be generated as in (1), where $\mathbf{z} \sim N(0, \mathbf{I})$.

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

As it will be demonstrated later on in this paper, this decomposition enables us to exploit different precision requirements of each decomposition level leading to an optimized hardware design. To the best of 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 [6]. Their approach is based on factorizing a 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 numbers every $N$ clock cycles where $N$ denotes the dimensionality of the distribution, and hence the size of the generated random vectors. In addition, the multiply-add operation in the algorithm is mapped 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 usage where the dimension of the distribution under consideration dictates the number of required DSP48 blocks on an FPGA.

An alternative approach has been proposed to solve the problem encountered in [6] 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 $\boldsymbol{\Sigma}$, by trading off "accuracy" for an improved resource usage [7]. In [7], "accuracy" is defined as the mean square error in the approximation of covariance matrix $\boldsymbol{\Sigma}$ by the SVD algorithm. With reference to resource usage, the approach in [7] requires $2K$ DSP48 blocks to produce a vector of size $N$, where $K$ denotes the number of decomposition levels required by using SVD to approximate the lower triangular matrix $\mathbf{A}$ within a certain accuracy. The resource usage can be reduced in comparison to [6] if $K$ is less than $N/2$ while achieving the same throughput performance. As well as an improved resource usage, 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 $\boldsymbol{\Sigma}$.

## 4 Proposed Methodology

The objective of the proposed system 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 that it exploits different precision requirements of different parts of the system in order to achieve a design that has the least error in the approximation of the input covariance matrix $\boldsymbol{\Sigma}$ and at the same time the least area requirements. The essential properties of the proposed algorithm are:

- It is based on the Eigenvalue decomposition 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.
- It produces ordered decompositions according to the impact on the error of the approximation of the covariance matrix. Hence, decompositions with less impact on the approximation error can be discarded resulting in the flexibility to exploit designs with specified resource constraint.

The full exploration of all possible precision requirements for each computational path is time consuming and computationally expensive. Hence, the proposed approach exploits a subset of the possible designs by introducing a library containing pre-defined hardware 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 and the optimum solution.

### 4.1 Overview of Algorithm and Architecture

Consider a covariance matrix $\boldsymbol{\Sigma}$, the algorithm applies the Eigenvalue decomposition algorithm to express $\boldsymbol{\Sigma} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$. According to (1), the random samples can be generated as in (2)

$$\mathbf{x} = \sum_{i=1}^{N} \sqrt{\lambda_i}\mathbf{u}_i z_i \tag{2}$$

Since the multivariate Gaussian samples are generated from a sum of the products of $\mathbf{u}$ and $z$, different levels of computation $i$ of (2) are broken down into decompositions which are mapped onto computational blocks (CB) designed for an FPGA. The proposed algorithm maps computational paths over this set of CBs targeting to reduce the resource usage and the approximation error of the covariance matrix. The objective of the exploration is to determine the best combination of CBs in order to construct a design which has the least approximation error as well as the least resource usage. The architecture is mapped to only logic elements on an FPGA so that the precision of the datapath can be varied as opposed to [6] and [7] where the precision of the architecture is fixed to 18bits as DSP48 blocks were utilized. Moreover, it has been shown that it is not a necessity to dedicate as much as 18 bits, which is the word-length supported by a DSP48. Quantizing the coefficients of the vectors to lower bit widths could be sufficient depending on the structure of the matrix under consideration [7]. The addition of mapping to DSP48 blocks can easily be included to the library whenever this is required.

The overall architecture of a multivariate Gaussian random number generator can be constructed from any combination of CBs. An instance is shown in Figure 2 where five blocks are selected out of a set of three types of CBs with different precisions namely $p_1, p_2$ and $p_3$. In the figure, $\mathbf{u}$ denotes the vector coefficient which is stored in on chip RAM blocks 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. In order for an improved throughput performance to be achieved, the operation is pipelined so that all of 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 which consume fewer resources and operate at a higher frequency in comparison to floating point arithmetic. The Eigenvalue decomposition algorithm also provides an order in the range of the coefficients. This has been exploited by the proposed algorithm in order for the resulting architecture to perform computations between intermediate results of similar dynamic range. A significant advantage for this is that the error due to truncation is minimized.

### 4.2 Proposed algorithm

Figure 1 shows an overview of the algorithm. The inputs to the proposed algorithm are a correlation matrix $\boldsymbol{\Sigma}$, 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 mean square error (MSE) between the original input matrix and the approximated one. Moreover, the proposed algorithm is fully parameterized such that it is able to support

**Algorithm**: Decompose a correlation matrix $\boldsymbol{\Sigma}$ given a target *MSE* and select the appropriate pre-defined hardware blocks to be mapped onto FPGA

```
$MSE_{min}$ = 1
WHILE $MSE_{min} > MSE_{target}$
    STAGE 1: SVD and Coefficients Quantization
    FOR i = 1:Number of $\boldsymbol{\Sigma}$ matrices
        Calculate the first decomposition level of $\boldsymbol{\Sigma}$ using $[\mathbf{u}, \boldsymbol{\lambda}, \mathbf{u}^T] = \mathrm{SVD}(\boldsymbol{\Sigma})$
        FOR j = 1:Number of hardware blocks
                Quantize $\sqrt{\boldsymbol{\lambda}}\mathbf{u} : \bar{\mathbf{u}} \leftarrow \sqrt{\boldsymbol{\lambda}}\mathbf{u}$ with $p_j$ bits precision
                $\bar{\boldsymbol{\Sigma}} = \bar{\boldsymbol{\Sigma}} + (\bar{\mathbf{u}}\bar{\mathbf{u}}^T)$
        END LOOP
    END LOOP
    STAGE 2: Calculate MSE
    FOR a = 1:Number of all possible designs
        For each design calculate $MSE$
    END LOOP
    Sort designs in the order of ascending MSE
    STAGE 3: Remove bad designs
    FOR b = 1:Number of all possible designs
        IF $resource_{b+1}$ >= $resource_b$ THEN
            Remove design b
        END IF
        Store remaining designs in a pool
    END LOOP
    IF Number of design in the pool > pool_size
        Keep $design_1$ up to $design_{pool\_size}$
    END IF
    Calculate the remaining of $\boldsymbol{\Sigma}$ matrices which will become new $\boldsymbol{\Sigma}$
    matrices for next iteration
    $\boldsymbol{\Sigma} = \boldsymbol{\Sigma} - \bar{\boldsymbol{\Sigma}}$
    $MSE_{min} = \min(MSE_{all\ designs})$
END LOOP
RETURN Valid Designs
```

**Fig. 1.** Outline of the proposed algorithm.

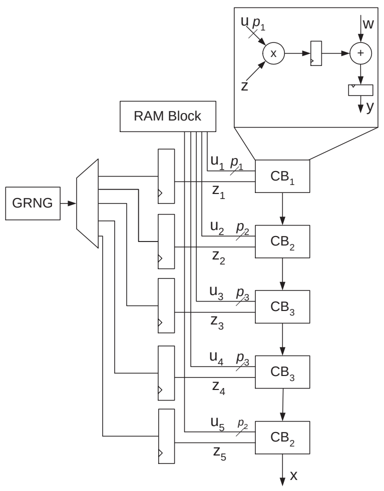

**Fig. 2.** Hardware architecture.

any number of pre-defined hardware blocks. The algorithm is divided into three stages. In the first stage, the correlation matrix $\boldsymbol{\Sigma}$ is decomposed into a vector $\mathbf{u}$ and a scalar $\lambda$. Then the vectors are converted into fixed point representation using one of the pre-defined available word-lengths. The effect of decomposing $\boldsymbol{\Sigma}$ and mapping it into different precision hardware blocks generates many possible designs. Note that the number of designs increases exponentially as the number of decomposition level increases. At a certain decomposition level $K$ the number of possible designs is expressed as $\sum_{i=1}^{K} B^i$, where $B$ denotes the number of hardware blocks.

The second stage of the algorithm calculates the MSE for all of the possible designs using

$$MSE_K = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left( \boldsymbol{\Sigma}_{i,j}^{K} - \overline{\boldsymbol{\Sigma}}_{i,j}^{K} \right)^2, \tag{3}$$

where $\overline{\boldsymbol{\Sigma}}$ represents the approximated correlation matrix, $N$ corresponds to the dimensionality of the distribution, $K$ refers to the number of decomposition levels, while $i$ and $j$ are the row and column indices of the matrix respectively.

The third stage of the algorithm discards any inferior designs. An inferior design is a design which, in comparison to another design, uses more hardware resources but produces worse MSE. Thus, the proposed algorithm is optimizing for the $MSE - Area$ 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. These steps are repeated until the design with the minimum MSE meets the target value specified by the user.

### 4.3 Overall System

In order to visualize the full functionality of the proposed approach, the overall structure of this methodology is illustrated in Figure 3 in which there are two main processes. The first process is the proposed algorithm which decomposes any given correlation matrix for a specified MSE into distinct vectors. The hardware library contains relevant information regarding the pre-defined 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.

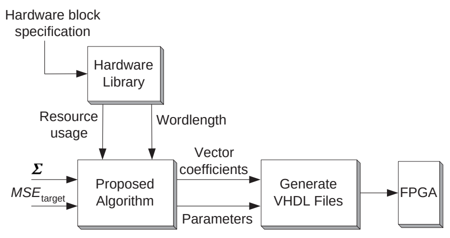

**Fig. 3.** Overall structure.

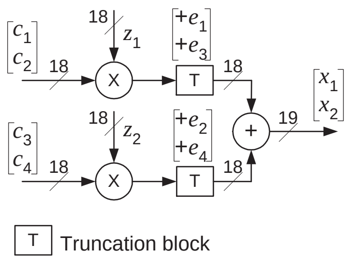

**Fig. 4.** Analysis of truncation error.

## 5 Error Analysis

The quantization of the coefficients and the truncation in the datapath of the proposed approach introduce two sources of errors in the system. The first source of error originates from the quantization of the coefficients of the correlation matrix. Assuming that no truncation takes place, the sole impact of this is that the sample correlation matrix of the generated samples will deviate from the original one. This error can be analytically calculated and the metric that is used to quantify this is the mean square error (MSE) in (3).

The second source of error is initiated by the truncation effect in the datapath where the multiplication and addition occur. Figure 4 illustrates a circuit diagram to generate a 2-dimensional vector $\mathbf{x}$. The numbers on the datapaths denote the word-length employed. $c_1$ and $c_2$ are the coefficients that correspond to the first decomposition level while $c_3$ and $c_4$ are the coefficients of the second decomposition level. Thus, the system implements the following expression:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} z_1 + \begin{bmatrix} c_3 \\ c_4 \end{bmatrix} z_2 \tag{4}$$

where $z_1, z_2 \sim N(0,1)$. The truncation of the datapath introduces an error $e$ to the system resulting in the final value of $\mathbf{x}$ expressed as in (5).

$$\begin{aligned} x_1 &= (c_1z_1 + e_1) + (c_3z_2 + e_2) \\ x_2 &= (c_2z_1 + e_3) + (c_4z_2 + e_4). \end{aligned} \tag{5}$$

The covariance between $x_1$ and $x_2$ which is given by the expectation $E\{x_1x_2\}$ is shown in (6).

$$\begin{aligned} E\{x_1x_2\} &= E\{(c_1z_1 + e_1 + c_3z_2 + e_2).(c_2z_1 + e_3 + c_4z_2 + e_4)\} \\ &= c_1c_2 + c_3c_4 + E\{e_1e_3\} + E\{e_1\}E\{e_4\} + E\{e_2\}E\{e_3\} + E\{e_2e_4\} \\ &= E\{x_1x_2\}_{original} + E\{e_1e_3\} + E\{e_1\}E\{e_4\} + E\{e_2\}E\{e_3\} + E\{e_2e_4\} \end{aligned} \tag{6}$$

where $E\{x_1x_2\}_{original}$ denotes the original targeted covariance between $x_1$ and $x_2$. It should be noted that $E\{e_1e_3\}$ and $E\{e_2e_4\}$ are non-zero since $e_1$ is correlated to $e_3$ and $e_2$ is correlated to $e_4$. Focusing on $e_1$ and $e_3$, the covariance between $e_1$ and $e_3$ can be expressed as $E\{e_1e_3\} = \rho_{e_1e_3}\sigma_{e_1}\sigma_{e_3}$, where $\rho_{e_1e_3}$ is the correlation between $e_1$ and $e_3$. The worst case error occurs when $\rho_{e_1e_3}$ is 1. Hence, $E\{e_1e_3\}$ is bounded by $E\{e_1e_3\} \leq \sigma^2_{e_1}$. 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 $\sigma^2_e = \frac{1}{12}2^{2d}(2^{2p_2} - 2^{2p_1})$, where $p$ denotes the word-length of the signal and $d$ refers to the scale. 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 $E\{x_1^2\} = \sigma^2_{e_1} + \sigma^2_{e_2}$. Thus, an upper bound of the error inserted to the system due to truncation of the datapath can be analytically calculated.

## 6 Results

The architecture has been implemented on a Stratix III EP3SE50F484C2 FPGA from Altera and Quartus II was utilized as a hardware synthesis tools. With regards 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.

### 6.1 Library Construction

The set of computational blocks 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 pre-defined in the hardware library for the construction of the multivariate Gaussian random number generator. The synthesis stage from Quartus II reports a resource usage of

115, 206, 300, and 332 LUTs for a CB with 4, 9, 15, and 18 precision bits respectively. These results are used by the proposed algorithm in order to optimize the architecture. In order to make a direct comparison between the the proposed approach and [6] and [7], DSP48 functionality has been mapped to logic on an FPGA and the reported resource usage are used accordingly.

## 6.2 Eigenvalue analysis

This section investigates the impact of the matrix structure on the proposed algorithm and the other two existing approaches [6],[7]. Four 5x5 correlation matrices with different eigenvalue profiles are randomly created, namely **A**,**B**,**C** and **D**. The three approaches under consideration are applied to the four generated matrices and the MSE calculated using (3) is plotted against the corresponding resource usage. It should be noted that this MSE does not take into account the truncation and the quantization effects. The MSE of approach [6] is used as a target MSE for the other two algorithms.

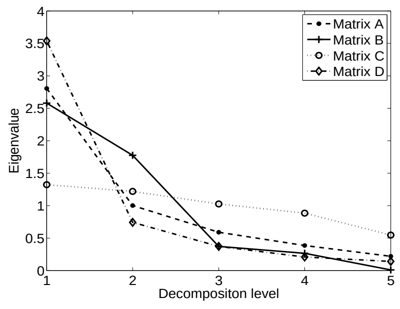

**Fig. 5.** Eigenvalue profiles of **A**,**B**,**C** and **D**

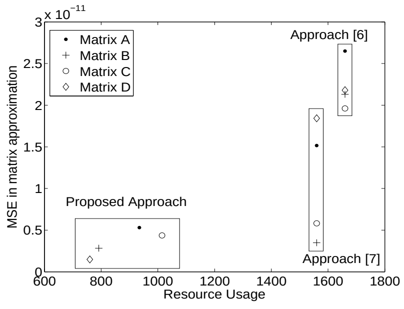

**Fig. 6.** MSE of **A**,**B**,**C** and **D** without truncation and quantization effects

Figure 5 illustrates a plot of the eigenvalues of the four matrices against the decomposition level. From the graph, four distinct lines can be observed indicating the difference in matrix structure where the eigenvalues of **A**, **B** and **D** drop rapidly in the first few levels of decomposition in comparison to **C** where the line is more flat. The MSE obtained from applying the proposed approach and approaches [6] and [7] is presented in Figure 6. The graph clearly demonstrates that the proposed approach provides the best designs for all four matrices in comparison to the other two approaches since a lower MSE is obtained using fewer resources. This is due to the use of hardware blocks with different precisions in order to map matrices with different precision requirements.

## 6.3 Evaluation of the proposed approach

The objective of this experiment is to assess the performance of the proposed algorithm in comparison to the approaches in [6] and [7]. In this experiment the three methods under consideration are applied to a 10x10 randomly generated correlation matrix **C**. The pool size of the proposed approach is set to 30 designs. The value of MSE in the approximation of **C** is plotted for all three approaches. In [6] and [7], the architecture is mapped onto DSP blocks as opposed to the proposed method where only logic elements are utilized. Note that the only error under consideration in this experiment is due to the quantization of the coefficients. Therefore, the MSE plotted on the graph indicates the lower bound of the approximation.

It can be observed from Figure 7 that the generated architectures from the proposed algorithm are the best designs in comparison with those from the other two approaches since the proposed algorithm obtains the same error using fewer resources. In particular, 10 DSP blocks which is equivalent to approximately 3,120 LUTs are required for approach [6] to achieve the same error as the proposed algorithm where only 1,250 LUTs are required. Hence, the resource usage is reduced by approximately a factor of 2.5. There is also an improvement in the resource usage in comparison to the approach in [7] where the resource requirement is reduced by about 2 to 2.5 times. With respect to [7], the reduction in resource usage is due to the use of Eigenvalue decomposition technique directly on the covariance matrix resulting in only one multiply-add operation as opposed to [7] where two operations are required for each decomposition level, and due to the word-length optimization of the datapath. The large drop in the MSE approximation observed is due to the fact that the whole matrix is approximated entirely by using specific number of decomposition levels. It should be noted that due to quantization error, the number of decomposition levels required for such a drop in the MSE may exceed the number of non-zero eigenvalues of the original matrix.

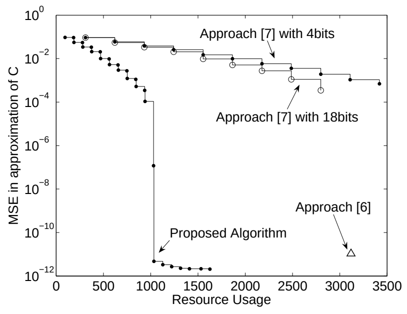

**Fig. 7.** Lower bound MSE

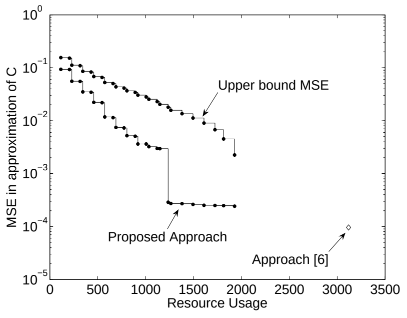

**Fig. 8.** Empirical MSE

In order to assess the quality of the random numbers produced by the proposed algorithm a large number of multivariate Gaussian samples are generated. They enable us to evaluate the sample correlation matrix which is then compared with the original correlation matrix in order to calculate the empirical MSE of the approximation. Thus, the error inserted due to truncation in the datapath as well as the quantization of the coefficients are taken into account as the correlation matrix is calculated from the empirical samples produced by the FPGA.

After applying the proposed algorithm to the matrix of interest and collecting all of the samples produced, the results are illustrated in Figure 8 where the empirical MSE of the proposed algorithm and the approach in [6] are plotted against the corresponding resource usage in which 100,000 vectors are generated. Note that the reported resource usage is obtained after synthesizing and performing place and route on all of the generated designs on the target device. The graph also illustrates the upper bound of the empirical MSE calculated using (6) from the error analysis model. As anticipated, the values of empirical MSE obtained in this experiment are much higher than the lower bound due to the error inserted from the truncation in the datapath. Figure 8 illustrates that the empirical MSE of the proposed approach converges to approximately $2.5 \times 10^{-4}$. This is slightly higher than the error obtained from [6] which is the reference design but much fewer resources are consumed by the designs produced by the proposed approach. In addition, it should be noted that the proposed approach offers more flexibility than [6] allowing the selection of designs for any given resource constraint.

### 6.4 Timing Analysis

Since the proposed approach targets applications where the architecture of multivariate Gaussian random number generator has to be mapped to various systems, the execution and synthesis and place and route time are of high importance. Figure 9 illustrates the execution time of the proposed algorithm for a range of matrices with different sizes. In addition, the pool size is varied between 30 to 120 for each matrix. The results show that the execution time increases linearly with the size of the matrix as well as the pool size. Figure 10 shows the required time for synthesizing and performing place and route for the generated designs with varying number of CB instantiations. It is apparent that as the number of CB instantiations increases the synthesis time also increases. The total time for a 50x50 matrix using 23 CBs is under 11 minutes where in the case of small matrix sizes the total time is much less where the entire process takes up to 3 minutes. It should also be noted that the average operating frequency of the designs is 411.32MHz bringing the actual throughput of the proposed design to the same level as in [6] and higher than [7].

## 7 Conclusion

This paper 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. In addition, an analysis of the error due to the truncation

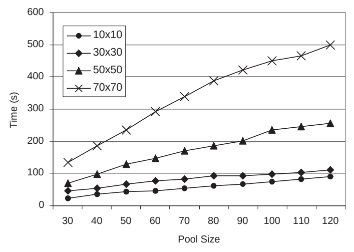

**Fig. 9.** Execution time of the algorithm.

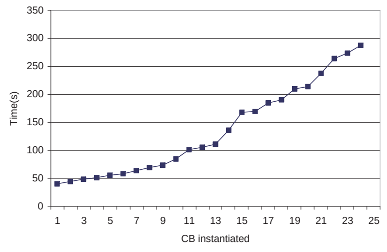

**Fig. 10.** Synthesis and Place and Route time.

operation along the datapath has been presented and an upper bound of this error is determined. Experimental results have demonstrated that by dedicating appropriate precisions to different computational paths, both the resource usage and the mean square error of the approximation of the targeted correlation matrix can be minimized. Future work includes the investigation of the relationship between the eigenvalue profile of the input correlation matrix and the type of the computational blocks that are required, targeting to optimize the performance and to reduce the run-time of the algorithm.

## References

1. P. Glasserman, P. Heidelberger, and P. Shahabuddin, "Variance reduction techniques for value-at-risk with heavy-tailed risk factors," in *Proceedings of the 32nd conference on Winter simulation*, 2000, pp. 604–609.
2. P. Glasserman and J. Li, "Importance sampling for portfolio credit risk," *Management Science*, vol. 51, pp. 1643–1656, 2003.
3. P. Glasserman, *Monte Carlo Methods in Financial Engineering.* Springer, 2004.
4. A. Kaganov, P. Chow, and A. Lakhany, "Fpga acceleration of monte-carlo based credit derivative pricing," in *Proceedings IEEE International Conference on Field Programmable Logic and Applications*, 2008, pp. 329–334.
5. D. B. Thomas, J. A. Bower, and W. Luk, "Automatic generation and optimization of reconfigurable financial monte-carlo simulations," in *Proceedings IEEE International Conference on Application-Specific Systems Architectures and Processors*, 2007, pp. 168–173.
6. D. B. Thomas and W. Luk, "Sampling from the multivariate gaussian distribution using reconfigurable hardware," in *Proceedings IEEE International Symposium on Field-Programmable Custom Computing Machines*, 2007, pp. 3–12.
7. C. Saiprasert, C.-S. Bouganis, and G. A. Constantinides, "Multivariate gaussian random number generator targeting specific resource utilization in an fpga," in *Proceedings IEEE Applied Reconfigurable Computing*, 2008, pp. 233–244.
8. N. H. Chan and H. Y. Wong, *Simulation Techniques in Financial Risk Management.* Wiley, 2006.
9. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C.* Cambridge University Press, 1992.