Design Exploration of an FPGA-Based Multivariate Gaussian Random Number Generator

Chalermpol Saiprasert

A thesis submitted for the degree of

Doctor of Philosophy in Electrical and Electronic Engineering of

Imperial College London

and the Diploma of Imperial College London

Department of Electrical and Electronic Engineering

Imperial College London

December 2010

Declaration

I herewith certify that the work presented in this thesis is my own work. All material in this thesis which is not my own work has been properly acknowledged.

Chalermpol Saiprasert

Abstract

Monte Carlo simulation is one of the most widely used techniques for computationally intensive simulations in a variety of applications including mathematical analysis and modeling and statistical physics. A multivariate Gaussian random number generator (MVGRNG) 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 due to their fine grain parallelism and reconfigurability properties and lower power consumption.

As well as the ability to achieve hardware designs with high throughput it is also desirable to produce designs with the flexibility to control the resource usage in order to meet given resource constraints. This work proposes a novel approach for mapping a MVGRNG 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. An analysis on the impact of the error due to truncation/rounding operation along the computa-

tional path is performed and an analytical expression of the error inserted into the system is presented.

Extra dimensionality is added to the feature of the proposed algorithm by introducing a novel methodology to map many multivariate Gaussian random number generators onto a single FPGA. The effective resource sharing techniques introduced in this thesis allows further reduction in hardware resource usage.

The use of MVGNRG can be found in a wide range of application, especially in financial applications which involve many correlated assets. In this work it is demonstrated that the choice of the objective function employed for the hardware optimization of the MVRNG core has a considerable impact on the final performance of the application of interest. Two of the most important financial applications, Value-at-Risk estimation and option pricing are considered in this work.

Acknowledgements

I would like to thank my supervisor, Dr. Christos Bouganis, and my co-supervisor, Dr. George A. Constantinides, for their infinite encouragement, support and understanding of my research.

Contents

Contents **5**

List of Figures **11**

List of Tables **15**

1 Introduction **19**

1.1 Motivation and Objectives 19

1.2 Thesis Outline 22

1.3 Contributions 23

1.4 Publications 26

2 Background **28**

2.1 Introduction 28

2.2 Monte Carlo Simulation 29

2.2.1 Definition 29

2.2.2 Methodology 30

2.2.3 Application Areas 31

2.3 Gaussian Distribution 32
2.3.1 Generation of Gaussian Random Samples 34
2.4 Multivariate Gaussian Distribution 36
2.4.1 Generating Multivariate Gaussian Random Samples . . . 39
2.5 Implementation of Random Number Generators 42
2.6 Field Programmable Gate Arrays 44
2.6.1 Basics 44
2.7 Related Work on Hardware based Random Number Generators 48
2.7.1 FPGA Based Gaussian Random Number Generators . . 48
2.7.2 FPGA-based Multivariate Gaussian Random Number Generators 51
2.8 Summary 56

3 Sampling from a Multivariate Gaussian Distribution with any given Hardware Resource Constraints 58

3.1 Introduction 58
3.2 Generating Multivariate Gaussian Samples 61
3.2.1 Approximating The Lower Triangular Matrix 61
3.3 Singular Value Decomposition: Definition 62
3.3.1 Applying the SVD Algorithm 63
3.3.2 Approximation Error Estimation 64
3.4 Proposed Methodology 65
3.4.1 High Level Overview of Architecture 65

3.4.2 Generation of $\mathbf{U}_{out}$ and $\mathbf{V}_{out}$ 67
3.5 Hardware Architecture 71
3.5.1 Hardware Implementation 73
3.6 Performance Evaluation 74
3.6.1 Impact of the Matrix Structure on the Proposed Algorithm 74
3.6.2 Evaluation of the Proposed Algorithm 76
3.6.3 Hardware Synthesis 81
3.7 Summary 82

4 An Optimized Hardware Multivariate Gaussian Random Number Generator 84

4.1 Introduction 84
4.2 Approximation of Correlation Matrix using Eigenvalue Decomposition Algorithm 87
4.3 Proposed Hardware Architecture 89
4.4 Analysis of Approximation Error 92
4.4.1 Impact of Correlation of Truncation Errors 97
4.4.2 Two Stage Error Model 99
4.4.3 Impact of the Correlation between Truncation Errors in the Addition Path 103
4.5 Proposed Methodology 105
4.5.1 Algorithm 1 107
4.5.2 Algorithm 2 108

4.5.3 Algorithm 3 . . . 108
4.5.4 Overview of Proposed Approach . . . 109
4.5.5 Overall Methodology . . . 111
4.6 Performance Evaluation . . . 114
4.6.1 Library Construction . . . 114
4.6.2 Evaluation of Proposed Methodology . . . 115
4.6.3 Error Distribution . . . 119
4.6.4 Effects of Coefficient Optimization . . . 120
4.6.5 Analysis of Empirical Approximation Error . . . 121
4.7 Summary . . . 124

5 Mapping Multiple Multivariate Gaussian Random Number Generators onto an FPGA 125

5.1 Introduction . . . 125
5.2 Proposed Algorithm . . . 127
5.3 FPGA Hardware Architecture . . . 131
5.4 Experimental Setup . . . 135
5.4.1 Extension of the Existing Approaches . . . 135
5.5 Performance Evaluation . . . 136
5.5.1 Analysis of Approximation Error and Resource Estimation Models . . . 137
5.5.2 Analysis of Empirical Data . . . 140
5.6 Summary . . . 145

6 Design of a Financial Application Driven Multivariate Gaussian Random Number Generator for an FPGA 146

6.1 Introduction . . . 146

6.2 Constructing Hardware Architecture . . . 148

6.3 Case Study: MVGRNG in Financial Applications . . . 151

6.3.1 Value-at-Risk of a Financial Portfolio . . . 151

6.3.2 Option Pricing . . . 153

6.3.3 Objective Functions . . . 153

6.3.4 Framework . . . 154

6.4 Experimental Setup . . . 156

6.4.1 Experiment I: Calculation of VaR . . . 158

6.4.2 Experiment II: Option Pricing . . . 159

6.5 Results: Evaluation of Hardware MVGRNG Blocks . . . 160

6.5.1 Experiment I: Calculation of VaR . . . 161

6.5.2 Experiment II: Option Pricing . . . 166

6.6 Summary . . . 170

7 Summary and Conclusions 171

7.1 Summary of Thesis Achievements . . . 171

7.2 Conclusions . . . 175

7.3 Future Work . . . 177

7.3.1 Short Term Goals . . . 177

7.3.2 Long Term Goals . . . 179

Bibliography **181**

List of Figures

1.1 Scope of This Work. 21

2.1 Overview of Monte Carlo Simulation. 30

2.2 Gaussian PDF with Different Variances. 33

2.3 Probability Density Function of a Multivariate Gaussian Distribution. 39

2.4 Architecture of a Typical FPGA. 45

2.5 A Typical Logic Block. 46

2.6 Switch Box. 46

2.7 An Overview of MVGRNG Block. 48

2.8 Hardware Implementation of [TL08]. 53

3.1 High level hardware implementation. 67

3.2 Outline of the algorithm. 69

3.3 Inner structure of the computational block. 72

3.4 Adder tree for the summing stage. 73

3.5 Eigenvalue profiles of matrices **A** and **B**. 75

3.6 MSE Comparison of matrices with different eigenvalue profile. . 76
3.7 Comparison of mean square error from two randomly generated matrices. 79
3.8 Error in the approximation of correlation matrices using empirical data. 80

4.1 Hardware architecture. 91
4.2 An architecture for a 2D MVGRNG. 93
4.3 Correlation between truncation errors. 99
4.4 Hardware Architecture for a 3D MVGRNG. 100
4.5 Correlation between Truncation Errors in the Addition Path. . . 104
4.6 Overview of the proposed approach. 111
4.7 Outline of the proposed algorithm. 112
4.8 Comparison of proposed algorithms with existing approaches. . 117
4.9 Error distribution. 120
4.10 Comparison of three algorithms. 121
4.11 Comparison between estimated and actual error for *Algorithm 1* 122
4.12 Comparison between estimated and actual error for *Algorithm 2* 123
4.13 Comparison between estimated and actual error for *Algorithm 3* 123

5.1 Proposed Algorithm. 132
5.2 Hardware Architecture. 133
5.3 Extension of Algorithm in Chapter 4 [SBCar]. 136
5.4 Estimate vs Empirical Approximation Error. 138

5.5 Estimate vs Empirical Resource Utilization. 139

5.6 Maximum Operating Frequency. 140

5.7 Comparison of Four 2x2 Correlation Matrices. 141

5.8 Comparison of Four 4x4 Correlation Matrices. 142

5.9 Comparison of Four 6x6 Correlation Matrices. 142

5.10 Comparison of Correlation Matrices with orders of 2 and 4. . . . 144

6.1 Proposed Methodology. 149

6.2 High Level Overview of Framework. 155

6.3 Experimental Setup Overview. 157

6.4 European-style VaR: Absolute Percentage Difference in the Mean Returns. 162

6.5 European-style VaR: Difference in the Upper and Lower Limits of the 95% Confidence Interval for Type I Matrices **A** and **B**. . 163

6.6 European-style VaR: Difference in the Upper and Lower Limits of the 95% Confidence Interval for Type II Matrices **C** and **D**. . 163

6.7 Asian-style VaR: Absolute Percentage Difference in the Mean Returns. 165

6.8 Asian-style VaR: Difference in the Upper and Lower Limits of the 95% Confidence Interval for Type I Matrices **A** and **B**. . . . 165

6.9 Asian-style VaR: Difference in the Upper and Lower Limits of the 95% Confidence Interval for Type II Matrices **C** and **D**. . . 166

6.10 Difference in Strike Price for Chooser Option 1 with **High** Correlated Assets. 167

6.11 Difference in Strike Price for Chooser Option 2 with **High** Correlated Assets. 167

6.12 Difference in Strike Price for Chooser Option 1 with **Low** Correlated Assets. 168

6.13 Difference in Strike Price for Chooser Option 2 with **Low** Correlated Assets. 169

7.1 Extension of Existing Algorithm in Chapter 5. 179

List of Tables

2.1 Comparison of Absolute Performance and Efficiency of RNGs (source: [THL09]). 43

2.2 Comparison of Different FPGA Based Gaussian Random Number Generators Implemented on a Xilinx Virtex-II XC2V4000-6 (source: [LVLL06]). 50

3.1 Resource usage in the Virtex-4 XC4VSX55 architecture using the proposed algorithm with 18 bit precision. 81

4.1 Summary of the three Algorithms. 106

4.2 Computational Block synthesis results. 115

6.1 Objective Functions. 154

6.2 Summary of Best Performing Metrics for Financial Applications of Interest. 160

List of Terms and Acronyms

ASIC Application Specific Integrated Circuit.

Block RAM On-chop block RAM memory.

CB Computational Block.

CDF Cumulative Distribution Function.

CLB Configurable Logic Block – Slices clustered together for the purposes of efficient routing.

Critical path The longest path of the circuit between storage elements, *e.g.* flip-flops.

Data path The path of the circuit through which data travel.

DSP Digital Signal Processor/Processing (context dependent).

FPGA Field Programmable Gate Array.

GPP General Purpose Processor.

GPU Graphics Processing Unit.

GRNG Gaussian Random Number Generator.

LUT Lookup Table.

MC simulation Monte Carlo simulation.

MSE Mean Square Error.

MVGRNG Multivariate Gaussian Random Number Generator.

Option In financial context, a contract between a buyer and seller.

PDF Probability Density Function.

Pipelining The process of inserting registers in the circuit to reduce clock period.

Portfolio A collection of financial investments.

RNG Random Number Generator.

Slice A fine-grain component on a Xilinx device. Unless explicitly stated, the component contains two 4-LUTs with optional output registers, and additional logic for efficient carry operations. Used as a unit for area measurement

SVD algorithm Singular Value Decomposition algorithm.

Synthesis The process of mapping an algorithm to hardware.

VaR Value-at-Risk, a measurement used to evaluate a risk of loss.

VHDL Very high speed integrated circuit Hardware Description Language.

Word-length The width of a stored value or a signal in bits.

Chapter 1

Introduction

1.1 Motivation and Objectives

Monte Carlo (MC) simulation is a well known method which is widely used in a variety of applications, especially in stochastic scientific processes and financial modeling [Gla04]. For instance, the computation of value-at-risk [GHS00a] and credit-risk calculation [GL03] 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 pre-requisites for almost all Monte Carlo simulations. These random numbers are drawn from a variety of distributions where one of the most commonly in use is the multivariate Gaussian distribution. Hence, a multivariate Gaussian random number generator (MVGRNG) is a key component in any Monte Carlo simulation.

In recent years, there is an increasingly high demand for computationally intensive calculations in these simulations as the number of variables associated with the target model increases. Traditionally, the random number generator and the MC simulation itself have been implemented on general purpose processors which often results in a very large processing time due to its computationally demanding nature [GHS00b].

Recently, alternative methods for the implementation of random number generators based on Graphics Processing Unit (GPU) [THL09] and Field Programmable Gate Array (FPGA) [WV08] have gained a great deal of attention due to their higher throughput performance and lower power consumption. Inspired by these factors, the work carried out in this thesis explores a wide range of design approaches for the implementation of a MVGRNG on an FPGA in order to achieve the most efficient and optimized hardware design in terms of hardware resource utilization on an FPGA. Moreover, an investigation into the impact of the design decisions taken towards the optimization and the construction of the MVGRNG based on the performance of some of the most widely used financial applications is performed.

One of the many optimization fields which will be the topic of investigation in this thesis is the hardware resource usage or the area constraint on an FPGA. As the number of resources on a single FPGA is limited, it is essential to keep the number of resources dedicated to the MVGRNG as low as possible without a significant penalty on the quality of the generated data. Hence, this could free up extra hardware resources to be utilized by other parts of the circuit if

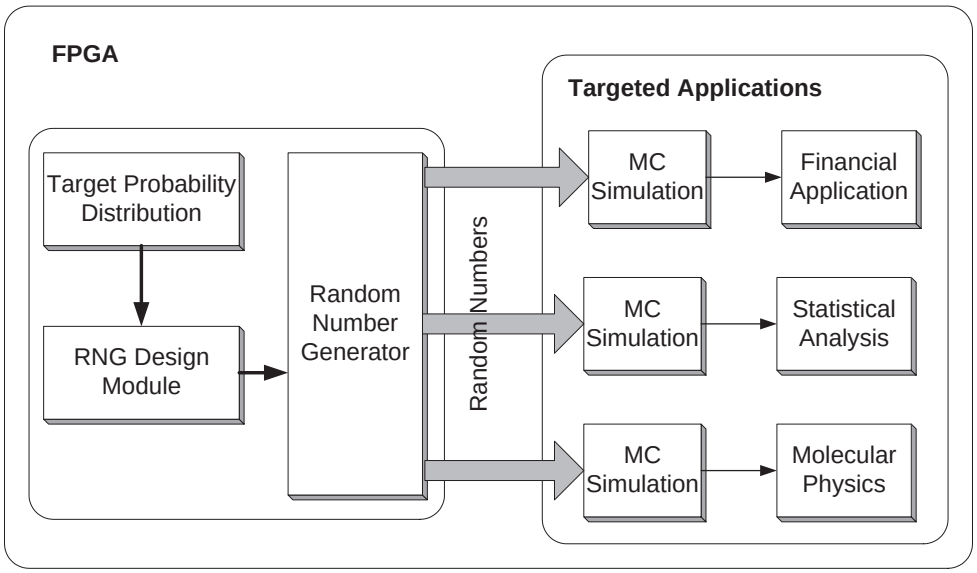


Figure 1.1: Scope of This Work.

required.

Figure 1.1 illustrates the scope of the work of this thesis. The core of this work lies in the Random Number Generator (RNG) Design Module where the algorithms proposed in the remaining chapters of this thesis are utilized in order to generate efficient hardware designs which are optimized for a multivariate Gaussian probability distribution to be mapped on an FPGA. The main function of the FPGA is to produce random numbers from a target distribution. These random numbers are then used to drive Monte Carlo simulation in a given application.

The main objective of this work is the exploration of hardware designs of multivariate Gaussian random number generator to be implemented on an FPGA and the associated design algorithms with different optimization functions to be utilized. The ultimate goal is to obtain the most efficient and optimized hardware for a given multivariate Gaussian distribution in terms of

hardware resource requirements.

1.2 Thesis Outline

The remainder of this thesis is composed of six further chapters and is organized as follows:

Chapter 2 begins with the introduction of Monte Carlo simulation and introduces the Gaussian and multivariate Gaussian probability distributions which are two of the most widely used distributions for the generation of random samples used to drive the simulation. A wide variety of methods to generate random numbers from these distributions are also presented. Moreover, an introduction to Field Programmable Gate Array is given together with concurrent and previous work with regards to hardware implementation of random number generators in the literature.

Chapter 3 presents the initial approach taken towards the mapping of a multivariate Gaussian random number generator on an FPGA with a view to explore the full design space. The methodology presented in this chapter demonstrates the concept of the initial work which will be fully explored in chapter 4.

Chapter 4 proposes a novel approach to produce a hardware design for MVGRNG along with a novel architecture to be implemented on an FPGA. Due to the quantization technique used in the algorithm in this approach, an error is inserted into the system. An in-depth error analysis is presented taking

into account all possible sources of error. As a result, an accurate error model is integrated into the proposed algorithm leading to a more optimized design.

In Chapter 5, an extra dimension is added to the approach proposed in Chapter 4 by considering the implementation of multiple multivariate Gaussian distributions using a single MVGRNG. The proposed approach exploits similarities in the precision requirements that exist between the different distributions under consideration. This subsequently creates an opportunity for an effective resource sharing by trading off the throughput of the random number generator for an improved resource usage which leads to a single generator that is optimized across all distributions of interest.

In Chapter 6, the investigation on the impact of the design decisions taken for the construction of the MVGRNG to the performance of various financial applications is presented. As a consequence, design criteria which are best suited for a specific financial application are established.

Finally, Chapter 7 summarizes the contributions of this thesis as well as providing the conclusions from the implementations proposed throughout the chapters. In addition, the main points which can be further investigated are given in order for further improvement.

1.3 Contributions

The main contributions of this thesis are as follows:

1. An initial approach towards the construction of a multivariate Gaussian

random number generator on an FPGA is presented. The proposed algorithm is based on the use of Singular Value Decomposition algorithm to approximate the lower triangular matrix which is a resultant matrix from applying the Cholesky decomposition to the target correlation matrix that characterizes the distribution of interest. As a consequence, the FPGA system designer has the flexibility to tune the hardware resource usage of the MVGRNG in accordance to the requirements. In addition to the proposed approach, a novel hardware architecture comprising of DSP blocks and adder tree structure to realize the functionality of the MVGRNG designed using the aforementioned algorithm is proposed (Chapter 3).

2. A novel approach to construct an MVGRNG for an FPGA implementation based on the Eigenvalue Decomposition algorithm is presented. The main difference between this contribution and the first contribution is that the matrix approximation is applied to the target correlation matrix rather than the lower triangular matrix as mentioned in the first contribution. This further minimizes the hardware resource utilization of the MVGRNG. Moreover, an in-depth analysis of all sources of errors inserted into the proposed system due to the approximation of the target correlation matrix is performed. Hence, a more complete and accurate error estimation model is provided. As a result, a novel optimization algorithm is presented taking into account the error estimation model to

produce a more optimized MVGRNG design (Chapter 4).

3. A novel FPGA architecture for an MVGRNG is proposed based on the approach in contribution 2. Unlike the architecture in contribution 1, the proposed architecture is comprised of multipliers and implemented only using LUTs. As a result, the precision of the computational path is no longer fixed which leads to a custom precision MVGRNG design (Chapter 4).

4. An innovative approach to implement a single FPGA based MVGRNG which is optimized for multiple multivariate Gaussian distributions. The proposed approach exploits similarities in the precision requirements that exist between the different distributions under consideration. This has proven to be extremely useful as a significant reduction in hardware resource usage is reported in comparison to other existing approaches (Chapter 5).

5. An investigation into the impact of the design decisions taken towards the optimization and the construction of the MVGRNG based on the performance of some of the most widely used financial applications is presented. Three design criteria are proposed for the construction of the MVGRNG using the financial application case studies which are the estimation of Value-at-Risk and the calculation of option pricing (Chapter 6).

1.4 Publications

The following publications have been published based on parts of the work which are presented in detail in this thesis.

Journal Publication:

- Chalermpol Saiprasert, Christos-Savvas Bouganis, George A. Constantinides, "An Optimized Hardware Architecture of a Multivariate Gaussian Random Number Generato", Accepted to appear in ACM Transaction on Reconfigurable Technology and Systems. (Chapter 4) [SBCar]

Conference Publications:

- Chalermpol Saiprasert, Christos-Savvas Bouganis, George A. Constantinides, "Multivariate Gaussian Random Number Generator Targeting Specific Resource Utilization in an FPG", Proceedings of Reconfigurable Computing: Architectures, Tools and Applications (ARC), pp. 233-244, March 2008. (Chapter 3) [SBC08]

- Chalermpol Saiprasert, Christos-Savvas Bouganis, George A. Constantinides, "Word-length Optimization and Error Analysis of a Multivariate Gaussian Random Number Generato", Proceedings of Reconfigurable Computing: Architectures, Tools and Applications (ARC), pp. 231-242, March 2009. (Chapter 4) [SBC09]

- Chalermpol Saiprasert, Christos-Savvas Bouganis, George A. Constantinides, "Design of a Financial Application Driven Multivariate Gaussian

Random Number Generator for an FPGA", Proceedings of Reconfigurable Computing: Architectures, Tools and Applications (ARC), pp. 182-193, March 2010. (Chapter 6) [SBC10a]

- Chalermpol Saiprasert, Christos-Savvas Bouganis, George A. Constantinides, "Mapping Multiple Multivariate Gaussian Random Number Generators on an FPGA", Proceedings of International Conference on Field Programmable Logic and Applications (FPL), pp. 89-94, August 2010. (Chapter 5) [SBC10b]

Chapter 2

Background

2.1 Introduction

In this chapter, the background information which is a pre-requisite for the work to be discussed throughout this thesis is presented in detail. Some of the background theories associated with the different approaches proposed in this thesis will be given at the beginning of each chapter. Moreover, some of the previous and concurrent work available in the literature focusing on the hardware implementation of random number generators will be thoroughly examined.

The chapter begins by providing the insights to a mathematical method called the Monte Carlo (MC) simulation. Its extensive usage can be found across a wide variety of applications such as financial engineering and statistical mathematics as one of the main features of the MC simulation is the computation of solutions to non-deterministic problem.

One of the most essential component in any Monte Carlo simulation is a random number generator which draws random samples from a given probability distribution. Some of the most widely used probability distributions, namely the Gaussian and the multivariate Gaussian distributions, are described in sections 2.3 and 2.4 respectively. The discussion of this chapter continues with different methods to generate random samples from these probability distributions which is traditionally implemented using computer software.

Section 2.5 discusses an alternative method to generate random samples from a given probability distribution through the use of hardware circuits. More importantly, a hardware device called Field Programmable Gate Array (FPGA) is introduced in section 2.6 as a target device for the implementation of the random number generator in this thesis.

Some of the related work in the literature regarding the FPGA implementation of random number generators are described in section 2.7. Finally, the approaches taken for the implementation of multivariate Gaussian random number generator (MVGRNG) on an FPGA are introduced in section 2.7.2.

2.2 Monte Carlo Simulation

2.2.1 Definition

Monte Carlo simulation is a computerized mathematical technique that relies on repeated random sampling and statistical analysis in order to compute

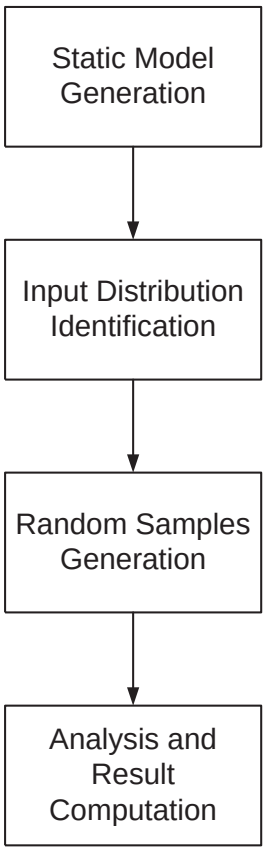


Figure 2.1: Overview of Monte Carlo Simulation.

the results. It is most suited to evaluating a non-deterministic process in which there are many possibilities that the outcome of this process might develop to over time even if the initial condition is known. This is due to extra uncertainties affecting the input parameters. Hence, the output of the non-deterministic process is often best described by a probability distribution.

2.2.2 Methodology

There are typically four main stages in a Monte Carlo simulation as illustrated in Figure 2.1. The initial stage of every Monte Carlo simulation is to develop a deterministic model which resembles closely to the actual scenario. In this

model, the most likely value of the input parameters are determined. As a consequence, the expected output is established by applying the mathematical function which describes the model to the previously determined input parameters.

In the second stage of the simulation the uncertainty elements, which originate from the stochastic nature of the input parameters, are inserted into the system. As a result, the underlying distribution which governs the input variables is determined through historical data.

The third stage is the core of a Monte Carlo simulation as this is where the element of randomness is introduced. After having identified the underlying distribution in the second stage, a set of random numbers are sampled from that distribution. Different sets of random samples are generated and used by the model in order to compute many possible paths that the model can take to arrive at different output values. Many distributions can be used to capture the underlying of the input variables and one of the most widely used distributions is the Gaussian distribution which will be discussed in more detail in the next section. Finally, statistical analysis is performed on the output values obtained from stage 3 in order to compute the final solution [Ray08].

2.2.3 Application Areas

Monte Carlo method is a very important method in a wide variety of application areas. It is especially useful in modeling the behaviour of stochastic processes with uncertainty in the input variables. Financial engineering is one

area where Monte Carlo simulation is heavily used. In [GHS00a], MC simulation is deployed for the calculation of value-at-risk. This allows an analyst to assess risk factors of certain financial assets with a certain confidence interval.

MC simulation is an essential component in another financial application known as portfolio optimization [Ver05]. In this application, financial investment is split into a collection of assets in order to diversify the chances of making profits. The expected return values of those assets can be modeled using MC simulation based on historical returns.

Another area which heavily relies on MC simulation is statistical physics. Molecular modeling is one of many subject areas in statistical physics which involve the use of MC simulation [BH02].

2.3 Gaussian Distribution

It has been established in the previous section that a random number generator plays an integral role in all Monte Carlo simulation. In this section, one of the most widely used distributions to model the randomness of the simulation is discussed.

The Gaussian probability distribution or the Normal distribution is one of the most widely used distributions in most application areas including mathematical modeling, statistics and financial engineering. It is a continuous distribution with a bell-shaped probability density function (PDF) that is symmetric about the mean. The probability density function of a Gaussian distribution

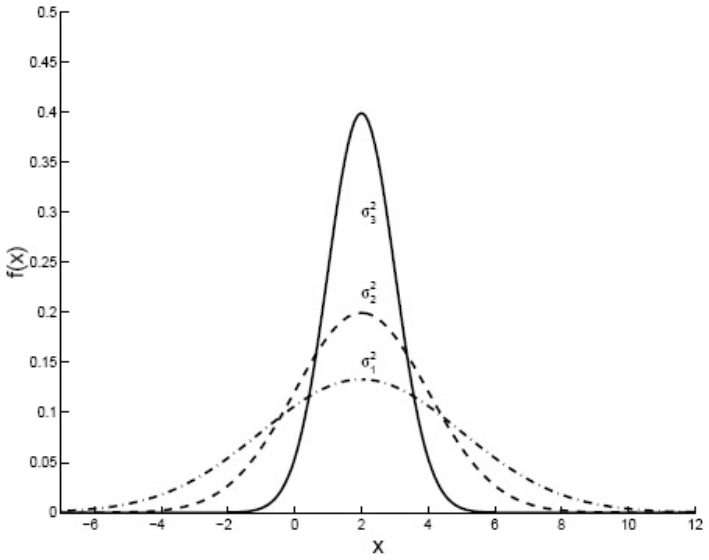


Figure 2.2: Gaussian PDF with Different Variances.

of random variable $\mathbf{x}$, $\mathbf{x} \sim N(\mu, \sigma^2)$ where μ is the mean and σ^2 is the variance, is given by (2.1).

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right). \tag{2.1}$$

Figure 2.2 illustrates the PDFs of three Gaussian distributions all with mean $\mu = 2$ while the variances σ_1^2, σ_2^2 and σ_3^2 are 9, 4 and 1 respectively. From the PDFs, the mean and the variance can be obtained from (2.2) and (2.3) respectively.

$$\mu = E\{\mathbf{x}\}. \tag{2.2}$$

$$\sigma^2 = E\{(\mathbf{x}-\mu)^2\}, \tag{2.3}$$

where the expression $E\{.\}$ denotes the expectation of a random variable. By definition, the mean μ is the expected value of the random variable and is the centroid of the PDF. The variance σ^2 is a measure of the dispersion of the random variable around the mean. A Gaussian distribution with zero mean and a variance of 1 is known as a standard Gaussian/Normal distribution. The Gaussian distribution is often used to model the behaviour of any variable which is likely to cluster around the mean, such as the birth weight of infants which is normally distributed with a mean of 3.27 kgs.

The integral of the PDF from $-\infty$ to x results in the cumulative distribution function (CDF) denoted as Φ and is expressed as in (2.4).

$$\Phi(x) = \int_{-\infty}^{x} P(x)\, dx = \frac{1}{2}\left[1 + erf(\frac{x}{\sqrt{2}})\right], \tag{2.4}$$

where erf denotes a special function called the error function. The CDF can be used to describe the probability for a random variable to fall in the interval $(-\infty, x]$.

2.3.1 Generation of Gaussian Random Samples

A wide range of algorithms exist in the literature for the generation of Gaussian random samples, most of which involve transformations of random variates produced from uniform distribution over the range (0,1). One of the most simplistic methods is called the CDF inversion method where it is based on the probability integral transform property which states that if U is distributed

uniformly on (0,1), then $\Phi^{-1}(U)$ will be Gaussian, where Φ is defined in (2.4). Thus, random Gaussian samples are generated by using Φ^{-1} to map the values of uniform samples u to Gaussian values.

The Box-Muller transform [BM58] is a well known method used for the generation of Gaussian random numbers. Under the Box-Muller transform two uniformly distributed independent random numbers u and v are used to produce a pair of Gaussian random samples x and y as in (2.5).

$$\begin{aligned} x &= \sqrt{-2\ln u}\cos(2\pi v) \\ y &= \sqrt{-2\ln u}\sin(2\pi v). \end{aligned} \tag{2.5}$$

Another widely used method to draw random samples from Gaussian distribution is called the Marsaglia polar method [Mar04]. It is an extension to the Box-Muller transform. The main difference is that the computation of the sin(.) and cos(.) functions is not required. Instead, u and v are sampled from uniform distribution over the range (-1,1) and a value s is computed using $s = u^2 + v^2$. The algorithm repeats until s is less than one, then the normally distributed samples x and y are returned and are expressed as follows:

$$\begin{aligned} x &= u\sqrt{\frac{-2\ln s}{s}} \\ y &= v\sqrt{\frac{-2\ln s}{s}}. \end{aligned} \tag{2.6}$$

A method which requires fewer computational power than the two methods mentioned earlier is known as the Piecewise linear approximation. Under this approximation the Gaussian distribution of interest is divided into a set of k triangular distributions. The width of each triangular distribution is $2w$ while its centre c_i is defined as $w((k+1)/2 - i)$. As the width of each triangular distribution is fixed at $2w$, the triangle overlaps with its adjacent neighbours. It is important to note that a triangular PDF can be easily generated as the sum of two uniform PDF's. Hence, an overall PDF of the Gaussian distribution can be created through sum of the overlaps using piecewise linear approximation [Kab00]. As a result, only addition and multiplication operations are required in order to sample from the Gaussian distribution.

2.4 Multivariate Gaussian Distribution

The generalization of one-dimensional Gaussian distribution or the univariate Gaussian distribution to higher dimensions is achieved through the multivariate Gaussian distribution. A random vector $\mathbf{x}_N = [x_1, x_2, ..., x_n]^T$ is Gaussian if its probability density function is given by (2.7).

$$P(\mathbf{x}_N) = \frac{1}{(2\pi)^{N/2} \mid \mathbf{\Sigma} \mid^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}_N - \mathbf{m})^T \mathbf{\Sigma}^{-1}(\mathbf{x}_N - \mathbf{m})\right\}, \qquad (2.7)$$

where $\mathbf{\Sigma}$ denotes the covariance matrix and $\mathbf{m}$ denotes a vector containing the mean values of each random variable. The covariance matrix $\mathbf{\Sigma}$ captures the covariance between the random variables in the random vector $\mathbf{x}_N$ and is symmetric with its elements. In general, the covariance between two random variables A and B is defined as in (2.8).

$$Cov(A,B) = E\{(A - E\{A\})(B - E\{B\})\} = E\{AB\} - E\{A\}E\{B\}. \quad (2.8)$$

As far as multiple variables are concerned, the covariance matrix offers a concise way of representing the covariances of all pairs of variables. The elements in a general covariance matrix can be expressed as follows

$$\begin{bmatrix} E\{\{\mathbf{x}_1 - \mu_{\mathbf{x}_1}\}^2\} & E\{(\mathbf{x}_1 - \mu_{\mathbf{x}_1})(\mathbf{x}_2 - \mu_{\mathbf{x}_2})\} & \dots & E\{(\mathbf{x}_1 - \mu_{\mathbf{x}_1})(\mathbf{x}_n - \mu_{\mathbf{x}_n})\} \\ E\{(\mathbf{x}_2 - \mu_{\mathbf{x}_2})(\mathbf{x}_1 - \mu_{\mathbf{x}_1})\} & E\{\{\mathbf{x}_2 - \mu_{\mathbf{x}_2}\}^2\} & \dots & E\{(\mathbf{x}_2 - \mu_{\mathbf{x}_2})(\mathbf{x}_n - \mu_{\mathbf{x}_n})\} \\ \vdots & \vdots & \ddots & \vdots \\ E\{(\mathbf{x}_n - \mu_{\mathbf{x}_n})(\mathbf{x}_1 - \mu_{\mathbf{x}_1})\} & \dots & \dots & E\{\{\mathbf{x}_n - \mu_{\mathbf{x}_n}\}^2\} \end{bmatrix}$$

The diagonal elements $\mathbf{\Sigma}_{i,i}$ of the covariance matrix represent the variances of the elements in vector $\mathbf{x}$ while the off-diagonal elements $\mathbf{\Sigma}_{i,j}$ denote the covariance between the elements in vector $\mathbf{x}$. Positive values of $\mathbf{\Sigma}_{i,j}$ indicate that if element x_i increases then element x_j will also increase and vice versa while negative covariance suggests that the two elements x_i and x_j will move

in opposite directions.

In order to normalise the elements inside a covariance matrix, the covariance between any two random variables is expressed as its correlation which is given in 2.9.

$$\begin{aligned} Cor(A,B) &= \frac{Cov(A,B)}{\sigma_A \sigma_B} \\ &= \frac{E\{AB\} - E\{A\}E\{B\}}{\sigma_A \sigma_B}, \end{aligned} \tag{2.9}$$

where σ_A and σ_B are the standard deviation of A and B respectively. Note that the correlation of a variable with itself, such as $Cor(A, A)$, is always 1. Hence, a covariance matrix can be transformed to a correlation matrix by dividing its value by the corresponding standard deviation values.

Figure 2.3 shows a probability density function of a two-dimensional Gaussian distribution with mean $\mathbf{m}$, covariance $\mathbf{\Sigma}$ and correlation C where,

$$\mathbf{m} = \begin{bmatrix} 40 \\ 60 \end{bmatrix}, \mathbf{\Sigma} = \begin{bmatrix} 100 & 30 \\ 30 & 140 \end{bmatrix}, C = \begin{bmatrix} 1 & 0.2535 \\ 0.2535 & 1 \end{bmatrix}$$

.

The multivariate Gaussian distribution is widely used in many applications especially those where many correlated factors are involved, for instance, statistical physics [BH02] and financial modeling [GHS00a]. More importantly, the main feature of the multivariate Gaussian distribution is its ability to capture and model the element of randomness of the correlated factors of interest.

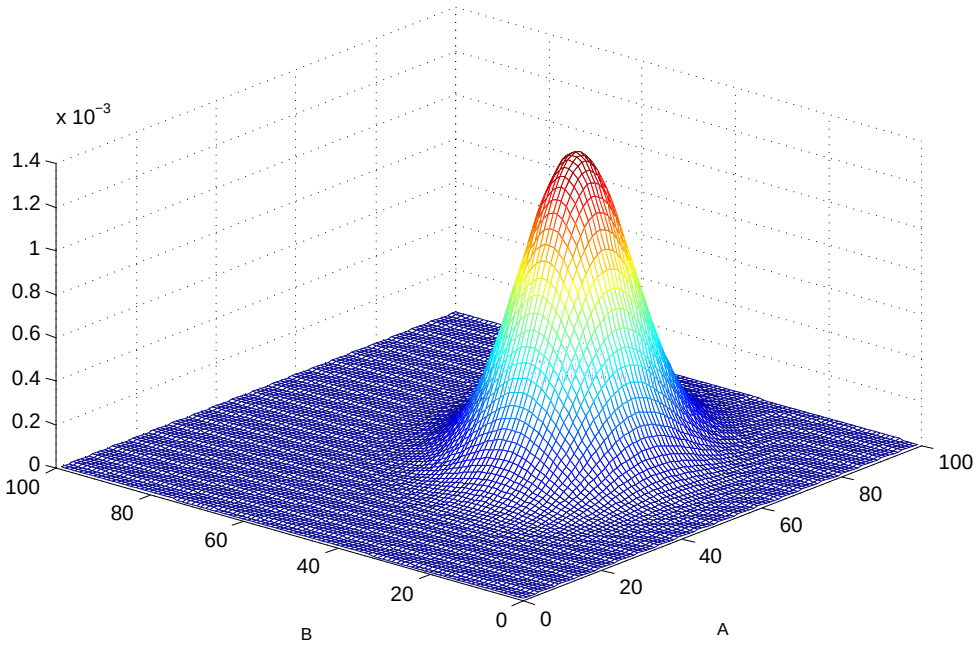


Figure 2.3: Probability Density Function of a Multivariate Gaussian Distribution.

The generated random samples are usually deployed to drive simulations such as Monte Carlo simulation. The multivariate Gaussian distribution will be the main focus for the remainder of this thesis due to its extensive usage across a wide range of applications.

2.4.1 Generating Multivariate Gaussian Random Samples

Many existing techniques are available in the literature for the generation of multivariate Gaussian random samples such as the Rotation method [Gra61], the Conditional method [SS62] and the Triangular Factorization method [Gra69]. Barr and Slezak [BS72] reviewed and implemented these three methods using

a general purpose processor in order to determine which method requires the least execution time. The evaluation of results have shown that the Triangular Factorization method is the most preferable method out of the three approaches as it requires less processing time and memory.

Under the Triangular Factorization method, multivariate Gaussian samples are generated based on the use of univariate Gaussian random samples followed by a transformation through the covariance matrix of interest resulting in multivariate Gaussian random vectors with their elements having the desired variance-covariance structure. This approach factorizes the covariance matrix $\mathbf{\Sigma}$ into a product of a lower triangular matrix $\mathbf{A}$ and its transpose as in (2.10).

$$\mathbf{\Sigma} = \mathbf{A}\mathbf{A}^T. \tag{2.10}$$

The required multivariate Gaussian random samples $\mathbf{x}$ are generated by multiplying a vector containing univariate Gaussian random numbers, $\mathbf{z} \sim N(O, \mathbf{I})$, with the lower triangular matrix $\mathbf{A}$ as shown in (2.11) to achieve the desired correlation structure. A vector $\mathbf{m}$ which contains the mean values is added to the product $\mathbf{Az}$ in order to adjust the means of the components.

$$\mathbf{x} = \mathbf{Az} + \mathbf{m}. \tag{2.11}$$

Equation (2.11) can be expanded to illustrate the full extent of the operation (2.12) and (2.13). x represents each element in the vector $\mathbf{x}$ which contains the multivariate Gaussian random samples, a denotes the coefficients

of the lower triangular matrix $\mathbf{A}$ and z denotes each element of the univariate Gaussian samples.

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} a_{1,1} & 0 & \dots & 0 \\ a_{2,1} & a_{2,2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{N,2} & \dots & a_{N,N} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{bmatrix} + \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_N \end{bmatrix} \tag{2.12}$$

$$\begin{aligned} x_1 &= a_{1,1}z_1 + m_1 \\ x_2 &= a_{2,1}z_1 + a_{2,2}z_2 + m_2 \\ x_3 &= a_{3,1}z_1 + a_{3,2}z_2 + a_{3,3}z_3 + m_3 \\ x_N &= a_{N,1}z_1 + a_{N,2}z_2 + a_{N,3}z_3 + ... + a_{N,N}z_N + m_N. \end{aligned} \tag{2.13}$$

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 [TL08].

2.5 Implementation of Random Number Generators

Random number generators have traditionally been implemented using a General Purpose Processor (GPP). Almost all computer programming languages offer functions or library routines to produce random numbers such as C, C++, Java and MATLAB. Some of the most widely used algorithms to generate samples from Univariate Gaussian distribution implemented on GPP are the Box-Muller and the Marsaglia polar method. As far as the Multivariate Gaussian distribution is concerned, the Triangular factorization is one of the most widely used algorithms for a GPP implementation taking advantage of the lower triangular matrix.

The random numbers are often used in many scientific applications such as to drive a wide range of simulations and modeling. These simulations often require a large amount of random numbers, thus, increasing the need for a fast random number generator. In the case of a software based multivariate Gaussian random number generator, the process of producing random samples involves matrix-vector multiplication which is a computationally expensive process. For instance, the generation of large amount of multivariate Gaussian samples, which is widely used in financial applications, is time consuming and large clusters of computer servers are often used to address this issue. The timing complexity and high power consumption due to the use of computer servers contribute to a major shortcoming of the software based

random number generators.

Recently, alternative methods for the implementation of Gaussian random number generators based on Graphics Processing Unit (GPU) and Field Programmable Gate Array (FPGA) have gained a great deal of attention. A comparison of absolute performance and efficiency of three different types of random number generators, namely uniform, Gaussian and Exponential, is shown in Table 2.1 using three different platforms. It can be seen that random number generators implemented on GPUs and FPGAs provide significantly better absolute performance in comparison to GPP implementation. A similar pattern is observed in terms of efficiency in power consumption where the FPGA is the most efficient followed by the GPU while the GPP consumes the most power.

Table 2.1: Comparison of Absolute Performance and Efficiency of RNGs (source: [THL09]).

	Performance (GSample/s)			Efficiency(MSample/joule)		
	GPP	GPU	FPGA	GPP	GPU	FPGA
Uniform	4.26	16.88	259.07	15.20	140.69	8635.73
Gaussian	0.89	12.90	12.10	3.17	107.52	403.20
Exponential	0.75	11.92	26.88	2.69	99.36	896.00

The goal of this work is the investigation of the design exploration of the implementation of multivariate Gaussian random number generator using an FPGA as a target device due to higher throughput performance and significantly lower power consumption in comparison to GPP implementation.

2.6 Field Programmable Gate Arrays

Field Programmable Gate Arrays (FPGAs) are flexible devices which are designed to be programmed and configured to implement almost any digital circuits after manufacturing. They contain up to hundreds of thousands of programmable logic elements and reconfigurable interconnects that allow the elements to be wired together. The major advantage of such devices is the flexibility and reconfigurability in the design and implementation of digital circuits. The same FPGA chip can be used for the implementation of different designs. Part of an incomplete design can be tested on the hardware while the remainder of the circuit can be added at any time once it is ready. Furthermore, the circuit currently loaded on the FPGA chip can simply be erased and another design can be easily programmed onto the board in seconds to allow the system designer to perform tests and reconfiguring if required. All of these aspects eliminate the time and cost factors associated with Application Specific Integrated Circuits (ASICs).

2.6.1 Basics

Field Programmable Gate Array was invented in 1984 by Ross Freeman, one of the co-founders of Xilinx Inc. [Xil04]. The first commercially viable FPGA was released in 1985. The FPGA, known as the XC2064, had 64 logic blocks and 1000 gates. In recent days, Xilinx Inc. are one of the two major players in the FPGA market with the other being Altera Corporation.

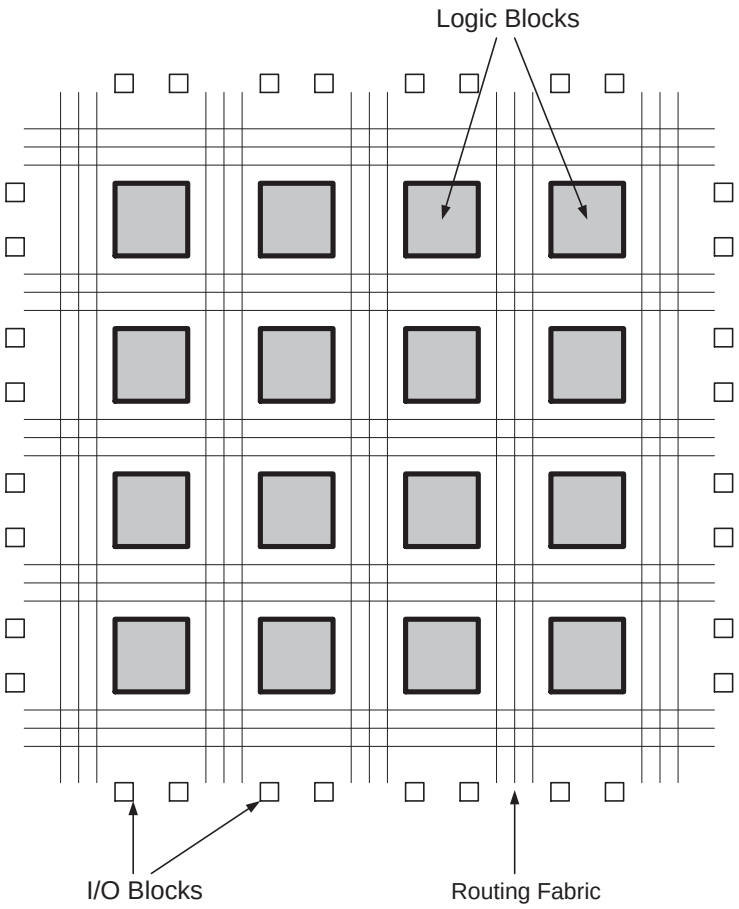


Figure 2.4: Architecture of a Typical FPGA.

In order to achieve the desired flexibility, two main components are required in an FPGA. Firstly, there should be simple and flexible circuit elements for the implementation of any arbitrary logic functions. These are known as logic blocks. Secondly, a programmable connection, known as the routing fabric, is required to link all the logic blocks together. A simplified architecture of a typical FPGA is shown in Figure 2.4.

A typical logic block usually consists of Look-Up Tables (LUTs) with circuit elements built around them. Any given logic function can be implemented onto these LUTs, each of which will have multiple inputs and a single output. Figure

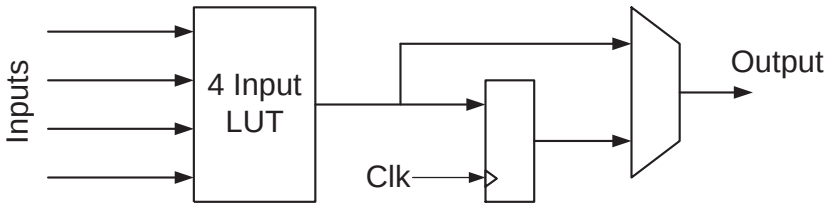


Figure 2.5: A Typical Logic Block.

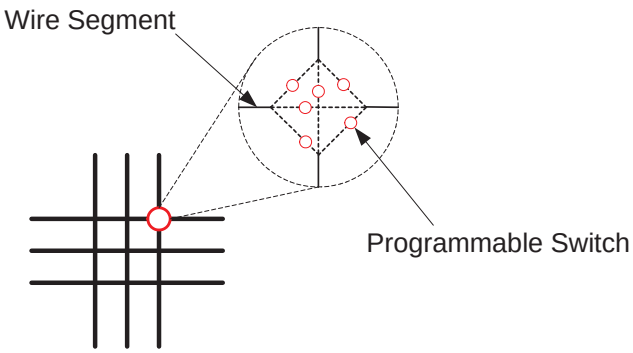


Figure 2.6: Switch Box.

2.5 illustrates a typical logic block with a 4 inputs LUT with its single output connected to a flip-flop.

The routing fabrics connect multiple arbitrary logic elements together via programmable switch-boxes. A switch-box exists at a point where a vertical and a horizontal channel intersect. The connection in a switch box is shown in Figure 2.6. In this switch-box topology many combinations of connections can be made from the six programmable switches to connect the channel segments depending on the FPGA system designer.

Embedded memories are another essential component in an FPGA as they allow temporary storage of data to be readily accessed on-chip. Significant advantages of these embedded memories are the increase in speed and eliminating the costs of extra I/O associated with off-chip RAM.

Modern FPGA families nowadays expand all of the basic capabilities of early models to include higher level functionality blocks fixed onto the chip when manufactured. One of these blocks is the embedded multiplier which has been introduced with the release of the Xilinx Virtex II. These embedded multipliers are extremely useful especially in digital signal processing applications generating a more efficient design as the extra routing between LUTs is avoided. Altera extended the idea of embedded multiplier by introducing a Digital Signal Processor (DSP) block in the Altera Stratix. The function of the DSP block is to multiply-accumulate as it consists of a multiplier and an adder.

In April 2010 Altera released the world's first FPGA with 28-nm technology called the Stratix V FPGA. The improvement of the Stratix V over its predecessors include higher bandwidth, higher levels of system integration, and flexibility. This device family offers more than 1 million logic elements and up to 53 Mb of embedded memory. The latest 28-nm FPGA device introduced by Xilinx in June 2010 is known as Virtex 7. The Virtex 7 family is reported to deliver a two-fold system performance improvement at 50 percent lower power in comparison to previous Xilinx devices.

2.7 Related Work on Hardware based Random Number Generators

2.7.1 FPGA Based Gaussian Random Number Generators

The importance of a univariate Gaussian random number generator should not be ignored since it forms the basis of the generation of multivariate Gaussian samples by providing the underlying randomness. Figure 2.7 shows an overview of a typical multivariate Gaussian random number generator block. The multivariate Gaussian random samples are produced by transforming the univariate Gaussian random samples using one of the algorithms discussed in section 2.4.

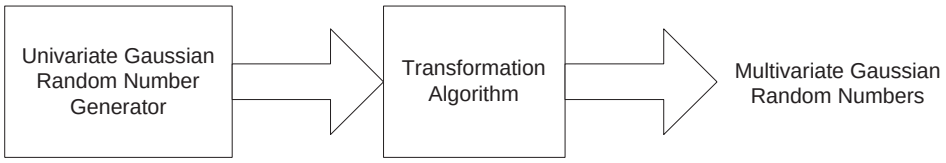


Figure 2.7: An Overview of MVGRNG Block.

Many architectures exist for the generation of univariate Gaussian random numbers on an FPGA platform. These include the Ziggurat method [ZLL+05], the Wallace method [LLV+05] and the Box-Muller method [LVLL06]. An extensive review of these techniques has been performed in [TLLV07] where it has been concluded that the Wallace method has the highest throughput while the Ziggurat method comes second.

In the Box-Muller method, independent uniform random variables are transformed into independent Gaussian variables. The Ziggurat method also employs the transformation of uniform samples into Gaussian samples. However, it is based on rejection method where the algorithm generates a set of candidate samples and then some of them are discarded depending on the specified conditions. Lastly, the Wallace method does not involve the transformation of uniform samples but it produces Gaussian samples directly.

Although the three methods discussed above are capable of producing Gaussian random samples using an FPGA platform, their hardware architectures involve the use of embedded multipliers. An alternative approach is established in [TL07] based on the piecewise-linear approximation targeting an architecture without multipliers. The piecewise-linear approximation technique enables the design to be pipelined easily in order to perform high speed operations. This is due to the fact that the design does not require any multipliers but only lookup tables, a subtractor and a comparator are utilized instead. Hence, the embedded multipliers can be allocated for other parts of the circuit if required.

Table 2.2: Comparison of Different FPGA Based Gaussian Random Number Generators Implemented on a Xilinx Virtex-II XC2V4000-6 (source: [LVLL06]).

	[ZLL+05]	[LLV+05]	[LVLL06]	[TL07]
Method	Ziggurat	Wallace	Box-Muller	Piecewise
Slices	891	770	1528	137
Block RAMs	4	6	3	1
18x18 Bits Multiplier	2	4	12	-
Operating Frequency (MHz)	170	155	233	249
Samples/Clock	0.993	1	2	1
Million Samples/sec	168	155	466	249

The piecewise-linear approximation algorithm is well suited for hardware implementation due to three major reasons. Firstly, the generator can be pipelined since there are no feedbacks. Secondly, the circuit requires only one memory access for each sample generated. The last reason is that the algorithm only performs the comparison and subtraction operations therefore only look up tables are required. Another advantage that the piecewise-linear approximation has over the other methods is that the target distribution can be altered at run-time by modifying the contents of RAM without the need for reconfiguration.

Table 2.2 shows a comparison between the four approaches mentioned earlier for the implementation of the same Gaussian random number generator. In terms of hardware performance with an FPGA as a target device, the approach in [TL07], where the piecewise-linear approximation algorithm is used, requires the least hardware resources to implement a Gaussian random number

generator while the Box-Muller GRNG [LVLL06] requires the most resources. Moreover, the approach in [TL07] has the highest clock speed while [LLV+05] is the slowest design.

2.7.2 FPGA-based Multivariate Gaussian Random Number Generators

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 using an FPGA and some of the most widely used are the Cholesky Factorization technique and the Singular Value Decomposition algorithm. These algorithms will be discussed in more depth in Chapter 3 and 4 of this thesis.

As far as the hardware based multivariate random number generator is concerned, designs can be classified into two major categories, serial and parallel. If a vector of N samples is to be generated then a serial generator outputs one element of the vector in every clock cycle requiring N clock cycles for creating a complete vector. On the other hand, a parallel generator produces a complete vector of size N every clock cycle.

In terms of performance, the parallel approach has a larger throughput than the serial approach requiring however more resources than the serial design. Thus, a trade off between throughput and resource usage exists between

the two approaches. The remainder of this thesis focuses on the class of serial designs since the target applications usually require multivariate Gaussian random samples of high dimensional vector such as the portfolio optimization of large number of assets in finance [LR01].

Thomas and Luk [TL08] developed a hardware model to generate multivariate Gaussian random numbers based on the Triangular Factorization method. This is one of the earliest approaches for generating multivariate Gaussian random samples implemented on an FPGA. According to the Triangular Factorization method, the multivariate Gaussian random samples $\mathbf{x}$ are generated by multiplying a lower triangular matrix $\mathbf{A}$ with the vector of independent univariate Gaussian random samples $\mathbf{z}$ as shown in (2.11).

In order to generate multivariate Gaussian samples of a vector of size N, parallel implementation is only practical for small N as both the number of registers required to store the coefficients of matrix $\mathbf{A}$ and the number of multiplications to be performed increases by $\frac{N^2}{2}$ as N increases. Therefore, the authors have chosen to generate the vectors serially such that it requires N clock cycles to completely generate all the elements inside each vector. The hardware architecture is illustrated in Figure 2.8 where a Multiply Accumulate unit is utilized to perform the multiply-add operation of the coefficients in matrix $\mathbf{A}$ and univariate Gaussian random vectors, denoted by $\mathbf{r}$ in the Figure. This functionality is realized through the DSP blocks on a target FPGA. In terms of resource usage the approach in [TL08] requires N DSP blocks in order to generate a vector of size N. It is worth mentioning that fixed point number

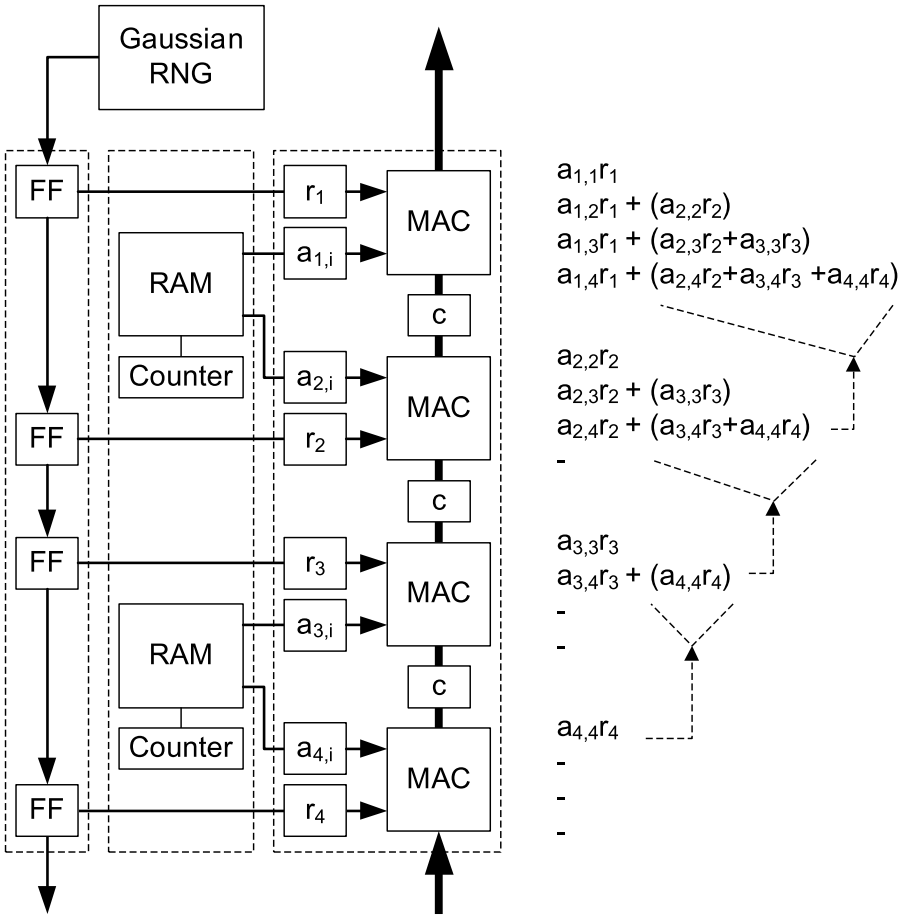


Figure 2.8: Hardware Implementation of [TL08].

representation is deployed throughout the hardware architecture of [TL08].

For the generation of univariate Gaussian random samples, Thomas and Luk employed just one univariate Gaussian generator for any size N of multivariate samples to be generated. This is achieved with the use of a long shift-register to distribute the produced vector while the next set of univariate samples is being generated in parallel.

If we consider a scenario where the number of available DSP blocks on an FPGA is fewer than the size of the output vector N then the number of available DSP blocks will not be adequate to implement the approach in [TL08] on to a single FPGA, with the throughput maintained at the same clock speed. In order to further illustrate this point, if a 1000-dimension vector is to be generated then a 1000x1000 covariance matrix is required resulting in 1000 DSP blocks to be implemented on an FPGA using the hardware architecture proposed in [TL08]. This amount exceeds the number of available DSP blocks available on a modern high-end FPGA.

For example, a high-end FPGA such as a Xilinx Virtex-4 offers up to 512 available DSP blocks [Xil07]. As a consequence, multiple FPGAs would be required to map this architecture in order to maintain the original target of producing N-dimension output vector per N clock cycles. Thus, a drawback of the approach in [TL08] is the lack of flexibility to accommodate designs where the size of the output vector is larger than the available DSP blocks on a single FPGA or the case where the system designer does not wish to allocate the required resources to this module.

In order to address this problem, the approach in Chapter 3 has been proposed as an alternative to produce multivariate Gaussian random samples by utilizing only a certain number of available DSP blocks specified by the system designer by allowing some error in the approximation of the correlation matrix while maintaining the same throughput as the approach in [TL08]. The novelty of the algorithm proposed in Chapter 3 is the use of Singular Value Decomposition algorithm to approximate the lower triangular matrix $\mathbf{A}$. As well as providing an improved resource usage, the approach in Chapter 3 offers the full 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 DSP blocks by trading off the discrepancy in the approximation of $\mathbf{\Sigma}$.

The precision of the computational path in the hardware architectures in both of the approaches previously discussed, [TL08] and the approach in Chapter 3, is fixed to the precision of the DSP block. It will be shown later in these chapters that allocating a fixed precision to all of the computation paths of the architecture does not lead to the optimum resource utilization.

In Chapter 4, the precision issue mentioned above has been exploited and word-length optimization techniques have been introduced to produce a hardware architecture with multiple precisions in its data path. The algorithm presented in Chapter 4 further reduces the required hardware resources in comparison to the existing approaches. In addition, an analysis of the correlation of errors due to truncation operations in the computation data path

has been presented and its effects have been modeled in the objective function targeting the optimum usage of the hardware resources.

2.8 Summary

This chapter has provided the basic pre-requisites required for the main topic of discussion in the remaining chapters of this thesis which is the design exploration of the implementation of multivariate Gaussian random number generator. In the first part of this chapter, an introduction to Monte Carlo simulation was given. This was followed by a description of two of the most widely used probability distributions for the generation of random numbers to be used in MC simulation namely, the Gaussian and multivariate Gaussian distributions.

As far as the implementation of these random number generators is concerned, the traditional method of using a general purpose processor is found to be computationally expensive and time consuming. Hence, a candidate device to accelerate this process known as Field Programmable Gate Array has been introduced. Moreover, the concurrent and previous work in the area of FPGA based random number generators have been discussed in the middle part of this chapter.

It can be concluded from the concurrent and existing work in the literature that the area of hardware implementation for a multivariate Gaussian random number generator is relatively new. Thus, many questions are still unanswered and a wide variety of design features are still left unexplored.

In the remaining chapters of this thesis, possible design considerations and different techniques for the implementation of MVGRNGs will be investigated in order to achieve a more efficient random number generator to be mapped on an FPGA.

Chapter 3

Sampling from a Multivariate Gaussian Distribution with any given Hardware Resource Constraints

3.1 Introduction

Monte Carlo methods are often the solution to non-deterministic processes which exist in a wide variety of application areas. From Chapter 2, it has been well established that these Monte Carlo simulations are very computationally expensive and time consuming when implemented on General Purpose Processor, often requiring a large cluster of PCs resulting in high power consumption. Field Programmable Gate Arrays (FPGAs) are often an alternative

target device to implement these simulations due to their fine grain parallelism, reconfigurability property and lower power consumption.

At the heart of any Monte Carlo simulation lies a random number generator block providing the element of randomness to drive the simulation. One of the most widely used random distributions is the multivariate Gaussian distribution. A random number generator block usually constitutes a part of a larger application such as Monte Carlo molecular modeling in statistical physics and portfolio optimization in financial mathematics. Hence, it is essential to efficiently dedicate the amount of hardware resources to such a generator as more resources can be allocated to the main application. One of the existing methods in the literature to generate multivariate Gaussian random numbers using an FPGA platform is by Thomas and Luk [TL08]. A drawback from [TL08] is that their algorithm does not have the flexibility to tune the number of resources required to implement the desired architecture in hardware to a specific resource requirement. Hence, only one fixed architecture is possible for a given multivariate Gaussian distribution.

Since the correlation matrix which characterizes the distribution of interest is constructed empirically from historical data it is expected that its elements deviate by certain amount from the underlying values. Therefore, a certain degree of freedom is allowed in the search for a hardware design by allowing some deviation in the approximation of the original correlation matrix. This chapter discusses an initial stage of producing designs that generate multivariate Gaussian random samples by permitting an approximation to the

original correlation matrix, while maintaining the same throughput as [TL08]. The quality of the approximation depends on the number of utilized resources when the design is mapped into hardware. In addition, a novel architecture for the implementation of a multivariate Gaussian random number generator in hardware, with the ability to tune the resource usage to the user's requirement, is presented.

The work presented in this chapter provides the main motivation of the thesis with regards to the design exploration of a multivariate Gaussian random number generator for an FPGA implementation. The main objective of this chapter is to demonstrate the concept of the initial methodology adopted in order to realize the required hardware implementation. The findings established in this chapter are preliminary and will be fully explored in Chapter 4.

The remainder of this chapter is organized as follows. Section 3.2 discusses how to generate random numbers from multivariate Gaussian distribution and introduces the matrix approximation technique used in the proposed algorithm. The proposed algorithm is described in Section 3.4 where the Singular Value Decomposition is applied to the proposed framework. In Section 3.5, the proposed hardware architecture utilized for the generation of multivariate Gaussian random samples to be implemented on an FPGA is presented. Experimental results regarding the performance evaluation of the proposed architecture are presented in Section 3.6 where a comparison to the existing method in the literature [TL08] is performed. Finally, Section 3.7 summarises

the work discussed in this chapter.

3.2 Generating Multivariate Gaussian Samples

Many existing techniques are available in the literature for the generation of multivariate Gaussian random samples such as the Rotation method [Gra61], the Conditional method [SS62] and the Triangular Factorization method [Gra69]. One of the hardware friendly techniques is the Triangular Factorization method where multivariate Gaussian samples are generated based on univariate Gaussian random samples. This approach factorizes the correlation matrix $\mathbf{\Sigma}$ into a product of a lower triangular matrix $\mathbf{A}$ and its transpose, $\mathbf{\Sigma} = \mathbf{A}\mathbf{A}^T$. Hence, only half the number of coefficients is required to be multiplied as the other half contains zeros. The required multivariate Gaussian random samples $\mathbf{x}$ are generated by multiplying a vector containing univariate Gaussian random numbers, $\mathbf{r} \sim N(\mathbf{o}, \mathbf{I})$ with the lower triangular matrix $\mathbf{A}$ to achieve the desired correlation structure while a vector $\mathbf{m}$ is added to adjust the mean values as shown in (3.1).

$$\mathbf{x} = \mathbf{A}\mathbf{z} + \mathbf{m}. \tag{3.1}$$

3.2.1 Approximating The Lower Triangular Matrix

In section 3.1 it can be deduced that the lower triangular matrix should be approximated in such a way that it takes advantage of the existing errors

in the empirically constructed correlation matrix due to the deviations of its underlying elements. The Singular Value Decomposition algorithm is chosen to approximate the lower triangular matrix **A** in the proposed algorithm as it will be demonstrated later in this section that the number of multiplication operations required to generate multivariate Gaussian random numbers can be reduced.

3.3 Singular Value Decomposition: Definition

The Singular Value Decomposition algorithm [PTVF92] or SVD is a technique where an $m \times n$ matrix **D** is decomposed into a product of an orthogonal matrix **U**, a diagonal matrix **S** and the transpose of another orthogonal matrix **V** expressed as follows:

$$\begin{aligned}
\mathbf{D} &= \mathbf{USV}^T \\
&= \begin{bmatrix} u_{1,1} & u_{1,2} & \dots & u_{1,m} \\ u_{2,1} & u_{2,2} & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ u_{m,1} & \dots & \dots & u_{m,m} \end{bmatrix} \begin{bmatrix} s_{1,1} & 0 & 0 & 0 \\ 0 & s_{2,2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & s_{m,n} \end{bmatrix} \begin{bmatrix} v_{1,1}^T & v_{1,2}^T & \dots & v_{1,n}^T \\ v_{2,1}^T & v_{2,2}^T & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ v_{n,1}^T & \dots & \dots & v_{n,n}^T \end{bmatrix}.
\end{aligned}$$

The diagonal elements in matrix **S** is known as the singular value. Essentially, the SVD algorithm expresses an initial matrix **D** as a linear combination of separable matrices using the least number of decomposition levels as possible.

3.3.1 Applying the SVD Algorithm

In essence, the Singular Value Decomposition expresses any given matrix into a sum of separable matrices allowing, in this case, the lower triangular matrix to be approximated according to a resource constraint on a given FPGA. Moreover, there is no restriction on the property of the matrix to be used for the generation of multivariate Gaussian samples.

For an $N \times N$ lower triangular matrix $\mathbf{A}$ the number of multiplications required to perform the computation in (3.1) is $N(N+1)/2$ which corresponds to the number of non-zero elements in matrix $\mathbf{A}$. In the proposed algorithm, the lower triangular matrix $\mathbf{A}$ is approximated by applying the Singular Value Decomposition algorithm.

The result of applying the SVD algorithm to the lower triangular matrix $\mathbf{A}$ is shown in (3.2) where the original matrix multiplication can be expressed as vector multiplication. $\mathbf{u}_i$ denotes the i^{th} column of matrix $\mathbf{U}$ while $\mathbf{v}_i$ denotes the i^{th} column of matrix $\mathbf{V}$ and K denotes the number of decomposition levels used by the algorithm. Under the SVD algorithm the number of general multiplications to achieve the same output is $2KN$. Therefore, the number of required multiplications can be reduced if K is less than $N/2$ in comparison to the approach in [TL08].

$$
\begin{aligned}
\mathbf{x} &= \mathbf{Az} \\
&= \mathbf{USV}^T r_i \\
&= \left(\sum_{i=1}^{K} \mathbf{u}_i s_i \mathbf{v}_i^T z_i \right) \\
&= \sum_{i=1}^{K} \mathbf{u}_i s_i \left(\mathbf{v}_i^T z_i \right)
\end{aligned}
\quad . \tag{3.2}
$$

SVD has proven to be extremely useful in other applications which involve large scale matrix multiplications such as in image processing. In [BCC05] SVD is applied to approximate a 2D filter used in the process of image convolution implemented on hardware. It was found that the approach produces the design with the optimum area usage as SVD reduces the number of high precision multipliers required for the implementation as well as distributing the DSP blocks efficiently in the design.

3.3.2 Approximation Error Estimation

Due to the fact that most of the large correlation matrices are constructed empirically, it is expected that they deviate from the true underlying matrix. The proposed methodology exploits this fact by approximating the lower triangular matrix up to the appropriate precision level defined by the user. The original lower triangular matrix **A** can be fully recovered when the number of

decomposition levels K is equal to the rank of the correlation matrix. Therefore, the correlation matrix of the original distribution can be approximated by taking into account K levels of decomposition, where $K \leq rank(\mathbf{A})$.

The metric which is used to assess the quality of the approximation is the mean square error (MSE) between the original correlation matrix and its approximation using (3.3).

$$Err_K = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left(\mathbf{\Sigma}_{i,j}^{K} - \overline{\mathbf{\Sigma}}_{i,j}^{K} \right)^2, \tag{3.3}$$

where $\mathbf{\Sigma}$ represents the original correlation matrix. $\overline{\mathbf{\Sigma}}$ corresponds to the approximated correlation matrix after approximating matrix $\mathbf{A}$ through the SVD algorithm, where $\mathbf{A} = \sum_{i=1}^{K} \mathbf{u}_i s_i \mathbf{v}_i^T$. N denotes the size of the output vector and K denotes the number of decomposition levels. i and j are the row and column indices of the matrix respectively.

3.4 Proposed Methodology

3.4.1 High Level Overview of Architecture

In section 3.3.1, the discussion regarding the approximation of the correlation matrix of interest is presented. The objective of this section is the mapping of the MVGRNG design into hardware. Fixed point number representation is deployed to store the matrix coefficients in this work since fixed point arithmetic leads to designs which are smaller and have higher operating frequency in an

FPGA compared to floating point arithmetic based designs. Hence, the matrix coefficients must be quantized to a specific fixed point precision before they can be used in the computation. As a result, quantization error is introduced when the vectors are mapped into hardware.

A high level overview of the proposed hardware architecture is required to be discussed at this stage in order to proceed to the main algorithm. From (3.2), the main operations to realize the functionality of the MVGRNG are the multiplication of the coefficients of the vectors and the univariate Gaussian samples and the summation of the resulting products (3.2). Figure 3.1 illustrates a high level hardware architecture of the multivariate Gaussian random number generator where the circuit comprises of K computational blocks representing K levels of decomposition. Thus, the number of computational blocks to be utilized in the architecture is relative to how well the lower triangular matrix is approximated.

As far as data storage is concerned, the vectors of coefficients $\mathbf{u}_q^i$ and $\mathbf{v}_q^i$ to be generated from the proposed algorithm can be stored either in the embedded block RAMs or in registers on the FPGA permitting parallel access to the data. In this approach, registers are deployed for small matrix order ($N < 20$) while block RAMs are used for larger matrix order.

It is important to note that the coefficients are quantized to the same precision for every computational block throughout the design. More in-depth discussion of the hardware architecture will be presented after the main algorithm in this section.

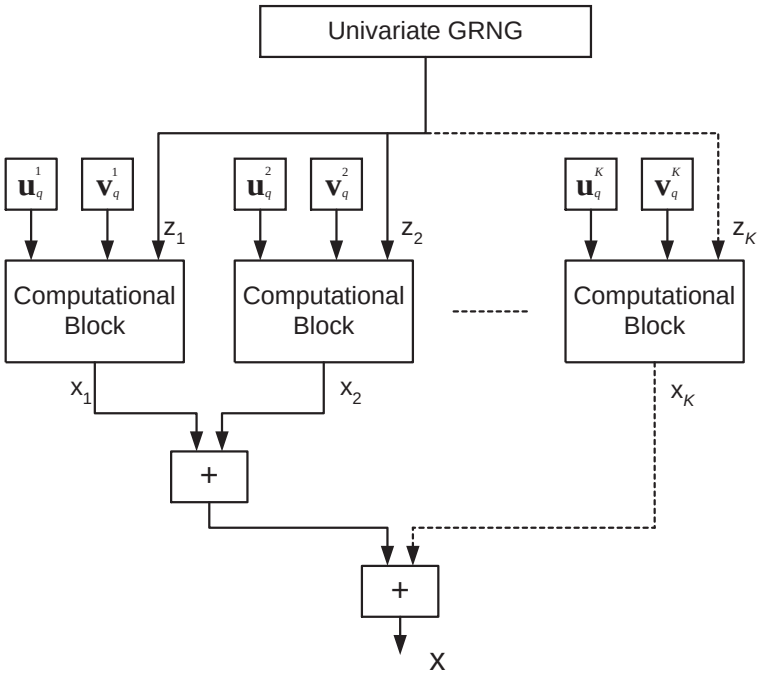


Figure 3.1: High level hardware implementation.

3.4.2 Generation of $\mathbf{U}_{out}$ and $\mathbf{V}_{out}$

This section describes the methodology of generating the relevant vectors of coefficients for the approximation of the lower triangular matrix $\mathbf{A}$ using the SVD algorithm. The proposed approach takes a correlation matrix $\mathbf{\Sigma}$ as an input and produces matrices $\mathbf{U}_{out}$ and $\mathbf{V}_{out}$ which contain the decomposition vectors $\mathbf{u}_q^i$ and $\mathbf{v}_q^i$ for each decomposition level i by applying the SVD algorithm to the lower triangular matrix $\mathbf{A}$. The subscripts q denotes the quantized version of the vectors as fixed point number representation is used. The decomposition can be driven either by the required error in the correlation matrix approximation or by the number of hardware resources to be allocated for the multivariate Gaussian random number generator.

An outline of the proposed algorithm to generate the coefficients of vectors

$\mathbf{U}$ and $\mathbf{V}$ is shown in Figure 3.2. The first step of the proposed algorithm is to obtain the first level of the decomposition which best approximates the lower triangular matrix $\mathbf{A}$ resulting in the generation of vectors $\mathbf{u}$ and $\mathbf{v}$ and a scalar s. It is important to note that the diagonal elements in matrix $\mathbf{S}$ are sorted in descending order. The columns of $\mathbf{U}$ and the rows of $\mathbf{V}$ are sorted in accordance with the singular values of $\mathbf{S}$. Thus, the best approximation of the remainder of $\mathbf{A}$, which is set to $\mathbf{A}$ at $i = 1$, at this decomposition level is from the product of the first column of $\mathbf{U}$, the first row of the first singular value and the first row of $\mathbf{V}$, which are denoted as $\mathbf{u}$, $\mathbf{v}$ and s respectively.

As the order in which the scalar s is multiplied by vectors $\mathbf{u}$ and $\mathbf{v}$ has no effect on the resulting product, the coefficient $\sqrt{s}$ is multiplied to both $\mathbf{u}$ and $\mathbf{v}$. As a consequence, only two elements are returned as outputs instead of three elements. The two vectors are then normalized in such a way that their coefficients lie in the range [-1,1) using a power of 2 scaling. This is to ensure that the range of numbers representable in hardware implementation is maximized. The vectors $\sqrt{s}\mathbf{u}$ and $\sqrt{s}\mathbf{v}$ are quantized to a user specified number of bits. Subsequently, the two quantized vectors are now represented by $\mathbf{u}_q$ and $\mathbf{v}_q$ respectively. At this stage of the algorithm an approximation of the lower triangular matrix $\mathbf{A}$, denoted by $\widehat{\mathbf{A}}$ is obtained by summing up the approximated matrices from each of the previous decomposition levels up to the current level.

As mentioned earlier in this section, an error due to the quantization of coefficients is inserted into the system due to the utilization of fixed point

Algorithm: Approximate a correlation matrix $\mathbf{\Sigma}$ given K DSP blocks and p bits precision

Calculate lower triangular matrix $\mathbf{A}$ using Cholesky Decomposition $\mathbf{A} = Chol(\mathbf{\Sigma})$
$\mathbf{A}_r = \mathbf{A}$
$N = \lceil \frac{K}{2} \rceil$

FOR $i = 1 : N$
- Calculate the first decomposition level of $\mathbf{A}$ using $[\mathbf{u}_i, s_i, \mathbf{v}_i^T] = \text{SVD}(\mathbf{A}_r, 1)$
- Transform $\sqrt{s}\mathbf{u}_i$ and $\sqrt{s}\mathbf{v}_i^T$ to be in the range [-1,1) using a power of 2 scaling
- Quantize $\sqrt{s}\mathbf{u}_i$: $\mathbf{u}_q^i \leftarrow \sqrt{s}\mathbf{u}_i$ with p bits precision
- Quantize $\sqrt{s}\mathbf{v}_i$: $\mathbf{v}_q^i \leftarrow \sqrt{s}\mathbf{v}_i$ with p bits precision
- $\widehat{\mathbf{A}} = \sum_{K=1}^{i} \mathbf{u}_q^K (\mathbf{v}_q^K)^T$
- $\mathbf{A}_r = \mathbf{A} - \widehat{\mathbf{A}}$
- Store $\mathbf{u}_q^i$ in a 2-dimensional array $\mathbf{U}_{out}(:,i)$
- Store $\mathbf{v}_q^i$ in a 2-dimensional array $\mathbf{V}_{out}(:,i)$

END LOOP

RETURN $\mathbf{U}_{out}$ and $\mathbf{V}_{out}$

Figure 3.2: Outline of the algorithm.

number representation. In order to take this quantization error into account, the algorithm updates the matrix $\mathbf{A}_r$ which is the remainder of lower triangular matrix $\mathbf{A}$ after every decomposition level by subtracting the original matrix $\mathbf{A}$ with the approximated matrix $\widehat{\mathbf{A}}$ as shown in (3.4).

$$\begin{aligned} \mathbf{A}_r &= \mathbf{A} - \widehat{\mathbf{A}} \\ &= \mathbf{A} - \sum_{K=1}^{i} \mathbf{u}_q^K (\mathbf{v}_q^K)^T, \end{aligned} \tag{3.4}$$

where $\mathbf{u}_q^K$ and $\mathbf{v}_q^K$ represent the quantized vectors with coefficients $\sqrt{s}\mathbf{u}$ and $\sqrt{s}\mathbf{v}$ respectively. The elements inside the approximated $\widehat{\mathbf{A}}$ get closer to the original matrix $\mathbf{A}$ as the decomposition level i increases while the approximation error decreases. This demonstrates that the proposed algorithm minimizes the inserted error in the system due to quantization effect by propagating it to the next level of decomposition [BCC05]. Hence, the added error is taken into account in the remaining decomposition stages.

After calculating the remainder matrix $\mathbf{A}_r$, the two vectors of coefficients, $\mathbf{u}_q$ and $\mathbf{v}_q$, are stored in matrices $\mathbf{U}_{out}$ and $\mathbf{V}_{out}$ respectively. The entire process is repeated, having the remaining matrix $\mathbf{A}_r$ as a starting point for the next decomposition level, until the termination condition is met.

It is important to note that the algorithm discussed in this section is carried out off-chip using a CPU in order to produce the required the two vectors of coefficients using fixed point number representation. In the next stage of the

methodology, these vectors of coefficients will be downloaded onto an FPGA for the computation of multivariate Gaussian random numbers.

3.5 Hardware Architecture

In the previous section, a methodology which approximate the lower triangular matrix $\mathbf{A}$ given a resource constraint and a certain precision on an FPGA has been introduced. The proposed algorithm described in the previous section produces vectors of coefficients for the generation of multivariate Gaussian random numbers. In this section, the relevant hardware architecture used to generate such random numbers as described in section 3.3.1 is discussed.

The inner structure of each computational block is illustrated in Figure 3.3. Both the multiply accumulation unit and the multiplier are mapped onto one of the available DSP blocks on the FPGA. In order to achieve the best possible throughput performance, the overall operation is pipelined into two main sections.

The first part performs the multiply-accumulate of the vector with univariate Gaussian samples $\mathbf{z}$ and a vector $\mathbf{v}_q$ to produce a scalar quantity zv which is stored in a register where it is fed back to be added with the next set of zv products in the next clock cycle. It is important to note that the entire process of multiply-add requires one clock cycle as this function is realized using the DSP block. In order to generate a vector of size N, it requires N cycles for the output in the register to be valid. The second part of the circuit performs

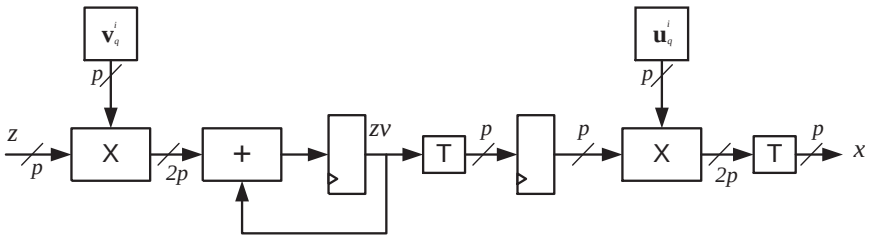


Figure 3.3: Inner structure of the computational block.

the multiplication of zv with the vector $\mathbf{u}_q$ requiring N clock cycles in order to generate a complete output vector of size N. The output of the second part of the circuit is fed in the adder tree.

As both stages require N cycles to produce a valid output, the design is optimized by running the two modules in parallel. The proposed approach offers the flexibility for the quantization precision of the data path to be user specific, denoted by p in Figure 3.3. The truncation block, denoted by T, is introduced into the framework where its function is to truncate the incoming data into the specified precision.

The final stage of the implementation is the summing stage where an adder tree structure is utilized for the addition of every decomposition of x from K computational blocks in order to obtain a valid element in the output vector. Figure 3.4 depicts the adder tree structure where there are K decomposition levels in this circuit. In order to achieve the optimum performance the tree is fully pipelined.

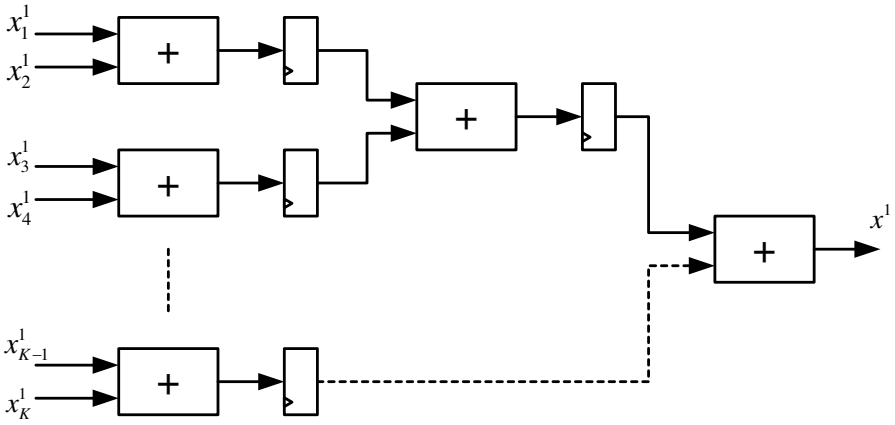


Figure 3.4: Adder tree for the summing stage.

3.5.1 Hardware Implementation

The multivariate Gaussian random number generator designed using the proposed approach is implemented onto a Virtex-4 XC4VSX55 FPGA from Xilinx. A benefit for selecting the Virtex-4 model is that it provides DSP blocks which have the capability to perform a multiply-accumulate operation in one clock cycle [Xil07]. In order to transform the architecture into a format which is compatible with the target FPGA, Handel-C compiler is used in this project. In addition, Xilinx ISE tools is utilized for hardware synthesis.

In this work the method in [TL07] for the implementation of the univariate random number generator is adopted as it is the most efficient in terms of resources requirement in comparison to other techniques in the literature. This is because the hardware architecture of a univariate random number generator based on the piecewise-linear approximation technique can be simply realized by the use of lookup tables, comparator and subtractor as mentioned in Chapter 2.

3.6 Performance Evaluation

This section presents experimental results for the multivariate Gaussian random number generator. The first set of experiments are to investigate the effect of the structure of the matrix under consideration on the overall performance of the proposed algorithm. The second set of experiments evaluate the proposed algorithm by analyzing the empirical samples produced from the hardware implemented model. Finally, the last section of this section presents the results obtained from the hardware synthesis of the proposed architecture.

3.6.1 Impact of the Matrix Structure on the Proposed Algorithm

The eigenvalues of a matrix can provide a lower bound on the error of the approximation for a given number of decomposition levels. In order to demonstrate that two lower triangular 30×30 square matrices $\mathbf{A}$ and $\mathbf{B}$ with different eigenvalue profiles have been selected, without the loss of generality, to be applied with the proposed algorithm. Matrices are said to have different eigenvalue profiles when the eigenvalues of one matrix decrease at a different rate than another matrix as the decomposition level increases.

The first ten eigenvalues of both matrices are plotted versus the level of decomposition in Figure 3.5. From the graph, it can be seen that matrix $\mathbf{A}$ is more separable than matrix $\mathbf{B}$ since fewer decomposition levels are required before the eigenvalue drops to small values. This is ideal in terms of hardware

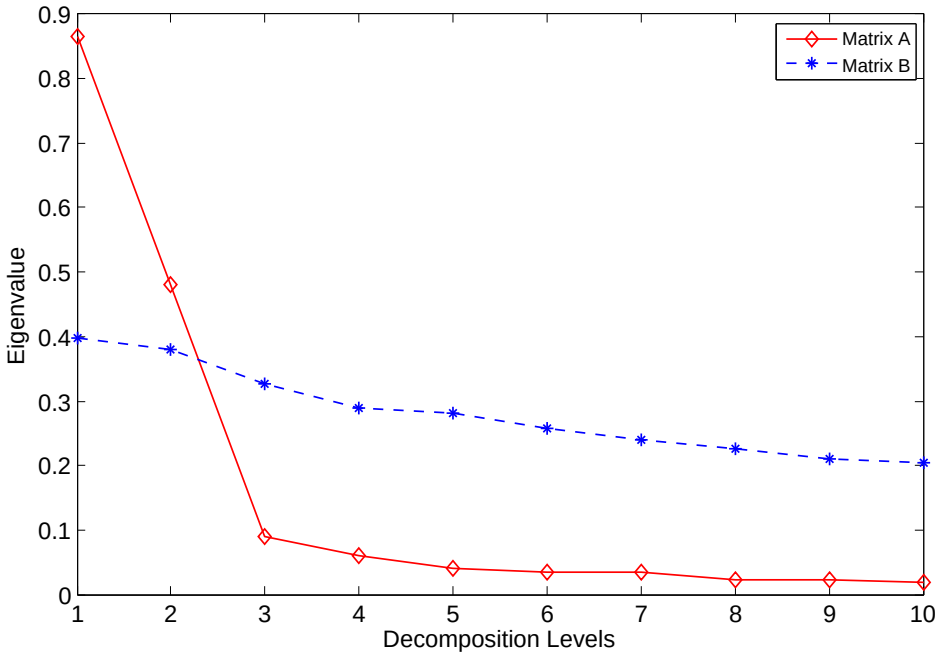


Figure 3.5: Eigenvalue profiles of matrices $\mathbf{A}$ and $\mathbf{B}$.

implementation since a large proportion of the original matrix $\mathbf{A}$ can be recovered using a few decomposition levels requiring fewer hardware resources in comparison to $\mathbf{B}$.

Figure 3.6 illustrates the mean square error (MSE) in the approximation of the lower triangular matrices $\mathbf{C}_A = \mathbf{A}\mathbf{A}^T$ and $\mathbf{C}_B = \mathbf{B}\mathbf{B}^T$ for different decomposition levels. From the figure it is apparent that matrix $\mathbf{C}_A$ requires approximately 17 levels of decomposition to achieve an MSE of around 10^{-30} while 30 levels of decomposition are required for matrix $\mathbf{C}_B$ to approximately obtain the same MSE. Thus, it can be concluded that matrices with different separability properties require considerably different numbers of decomposition levels to achieve a certain error in the approximation.

Since the values of the approximation error in Figure 3.6 are estimated

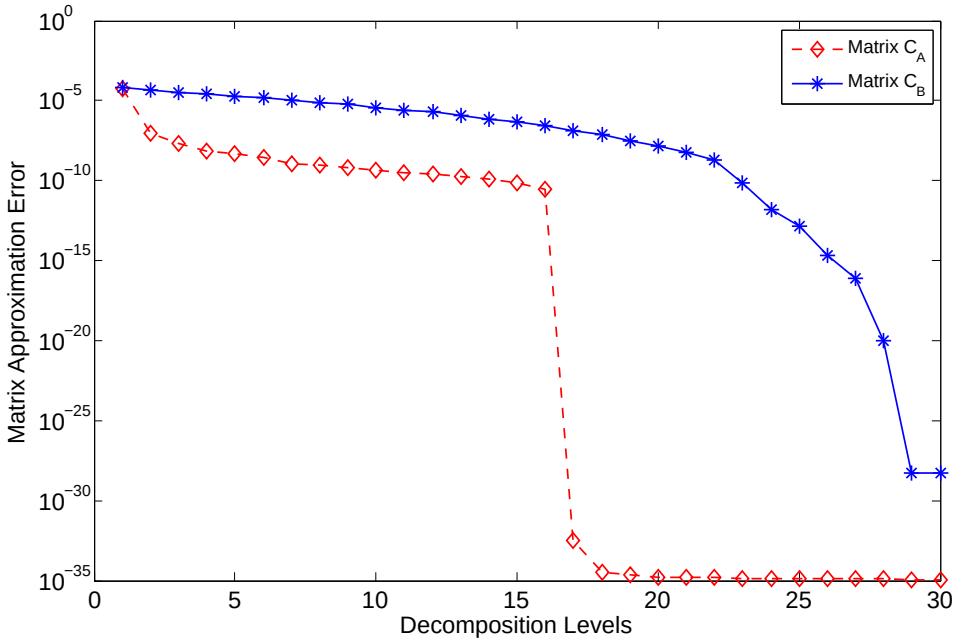


Figure 3.6: MSE Comparison of matrices with different eigenvalue profile.

without taking into account the quantization effect and truncation effect in the data path, the plot provides a lower bound for the achieved error for a given decomposition level in the proposed algorithm in Figure 3.2. In other words, this is the best possible approximation error that can be achieved since floating point number representation is used during the execution of the algorithm. In summary, the results obtained demonstrate that the performance of the proposed algorithm may depend on the separability property of the matrix of interest.

3.6.2 Evaluation of the Proposed Algorithm

The main concept behind this work is that by permitting a certain error in the approximation of the correlation matrix a reduction in the number of required

DSP blocks on the FPGA can be achieved.

Approximation Error with Quantization Only

As illustrated in section 3.5, the proposed approach employs two DSP blocks for every level of decomposition for the approximation of the lower triangular matrix, one for the multiply accumulation unit and one for the multiplier. An experiment is carried out where matrices **A** and **B** from the previous section are used to determine the achievable approximation error given a range of resources. In this experiment, an approximation error of the correlation matrix is the mean square deviation between the original correlation matrix and the approximated correlation matrix.

The proposed algorithm is applied to matrices $\mathbf{C}_A$ and $\mathbf{C}_B$ with no quantization, 4 bits quantization and 18 bits quantization to the vectors of coefficients obtained at the output stage of the algorithm. In order to compare the performance of the proposed algorithm with the technique in [TL08], the approximation error is also obtained for [TL08] where the algorithm is applied to the same set of matrices.

Figure 3.7 illustrates the obtained results where the approximation error for each of the multivariate Gaussian random number generator is plotted against the number of utilized DSP blocks. As expected, the approximation error decreases as the resource usage increases for both matrices. The results show that for matrix **B** there is only a considerable difference in the obtained approximation error between MVGRNGs using 4 bits quantization and 18

bits quantization when 10 or more DSP blocks are utilized using the proposed algorithm. However, the difference in approximation error between MVGRNGs using 4 bit and 18 bit precisions can clearly be seen for matrix **A** as shown in the graph.

The difference in the patterns of the approximation error plots between the two matrices is due to the difference in eigenvalue profiles. Moreover, for both matrices **A** and **B**, the approximation error of the correlation matrix obtained using the proposed algorithm with 18 bits word-length is slightly larger than the optimum approximation error obtained using the proposed algorithm without any quantization.

It is important to note that the approach in [TL08] is only able to generate samples using a fixed number of resources as can be seen in Figure 3.7. In addition, the proposed approach provides a trade-off opportunity between approximation error and the reduction in resource usage. Three different approximation error values are plotted which correspond to different levels of precision, all having utilized 30 DSP blocks for both **A** and **B** due to their rank being 30. More importantly, the results from the figure demonstrate that the proposed approach has the ability to produce designs across the available design space for a given correlation matrix while the approach in [TL08] does not have this flexibility, with both approaches maintaining the same throughput.

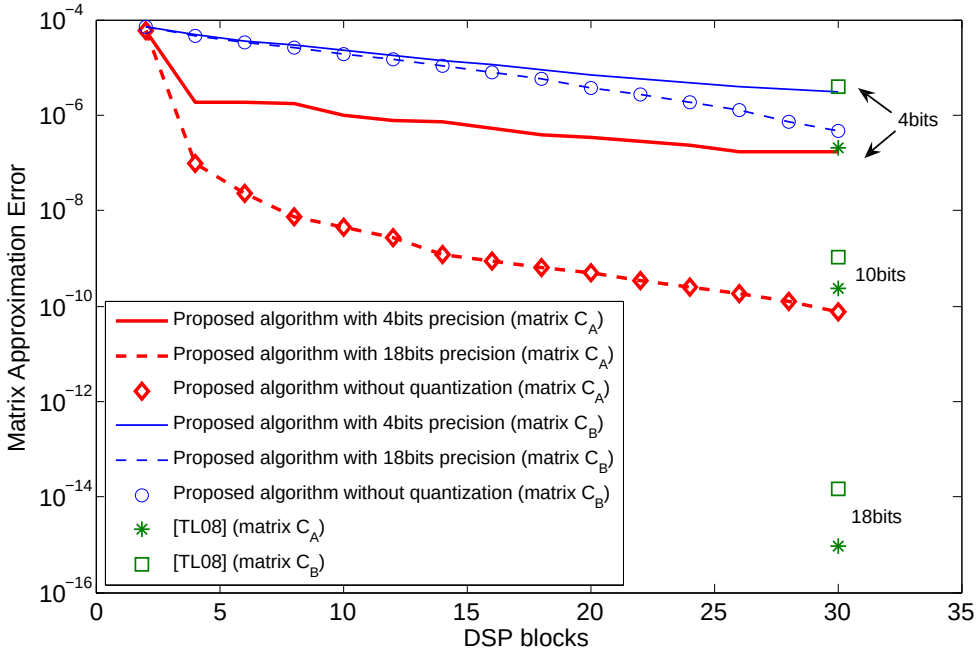


Figure 3.7: Comparison of mean square error from two randomly generated matrices.

Empirical Approximation Error

In order to fully assess the functionality of the proposed algorithm, it is necessary to investigate the correlation matrix taking into account the quantization effects of the data path. An experiment is performed to calculate the approximation error between the original correlation matrix and the empirical correlation matrix. To construct an empirical correlation matrix, 100,000 vectors of multivariate Gaussian samples are generated for both matrices under consideration. Correlation between these generated samples are estimated and hence, the empirical correlation matrix is obtained.

The number of DSP blocks utilized is set to half of the size of the output vector N, thus, in all cases, the decomposition level K is equal to $N/2$. Figure 3.8 illustrates the result of this experiment. As expected, it is apparent that

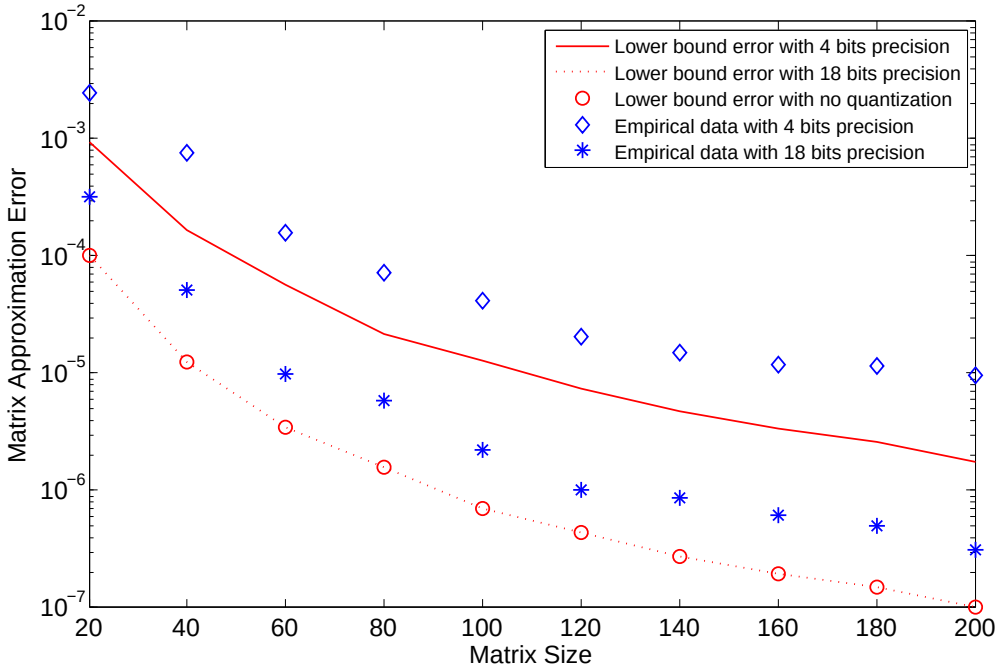


Figure 3.8: Error in the approximation of correlation matrices using empirical data.

the approximation error obtained for MVGRNGs using 18 bits precision is lower than that of MVGRNGs using 4 bits precision. The graph also depicts the lower bound of the approximation error for MVGRNGs using 4 and 18 bits precision and the best case scenario where no quantization to the coefficients is performed. Moreover, an apparent trend can be observed in the plot where the approximation error decreases as the size of the matrix increases. This behaviour depends on the structure of the correlation matrix and it should not be generalized since matrices with different eigenvalue profiles are expected to have different levels of approximation.

3.6.3 Hardware Synthesis

This section presents hardware synthesis results obtained from Xilinx Tools after implementing the MVGRNGs designed using the proposed approach onto a Virtex-4 FPGA. Table 3.1 illustrates the resource utilization when the MVGRNGs are mapped onto a Virtex-4 FPGA. Note that in this work, the designs are synthesized into hardware using Handel-C. N denotes the size of the output vector while K represents the decomposition level.

Table 3.1: Resource usage in the Virtex-4 XC4VSX55 architecture using the proposed algorithm with 18 bit precision.

Configuration	Block RAM	DSP Block	Slices	DSP Blocks Usage
N=20, K=5	10	10	185	1.95%
N=50, K=10	20	20	402	3.91%
N=100, K=30	60	60	913	11.72%
N=150, K=50	100	100	1841	19.53%
N=300, K=100	200	200	3678	39.06%

It is apparent from the table that the number of slices utilized scales linearly with the levels of decomposition. Moreover, the percentage of DSP block usage is calculated based on the fact that the Xilinx Virtex-4 XC4VSX55 contains 512 available DSP blocks [Xil07].

The table clearly illustrates the linear dependency between the decomposition levels and the required Block RAMs, DSP blocks and slice usage. On the other hand, the same design mapped onto an FPGA using the approach in [TL08] requires N number of DSP blocks where N denotes the size of the

output vector. The usage of DSP blocks on the Xilinx Virtex-4 XC4VSX55 range from 1.95% for a 20×20 matrix to 39.06% for a 300× 300 matrix.

Synthesis results from Xilinx tools show that the MVGRNGs designed using the proposed algorithm are, on average, able to generate 1 sample every 9.88ns. Thus, the throughput of the proposed architecture is $1.01 \times 10^8/N$ vectors per second where N denotes the size of the output vector. In addition, the average operating frequency of the MVGRNGs generated in this experiment is 101.21MHz.

3.7 Summary

In this chapter, an initial concept to serially generate multivariate Gaussian random samples with the ability to control the resource usage in a modern FPGA is presented. The key idea is the approximation of the lower triangular matrix, which is a product of applying Cholesky factorization to the input correlation matrix, using the Singular Value Decomposition algorithm for the generation of multivariate Gaussian random samples. The motivation behind this work is the need to produce a design that generates multivariate Gaussian random samples given any constraints of available resources and the fact that a certain error in the approximation of the correlation matrix can be tolerated.

The approximation in the lower triangular matrix enables the proposed algorithm to produce many designs with any given hardware resource constraint on an FPGA which is a major improvement over [TL08] as only one fixed de-

sign can be achieved for a given target multivariate Gaussian distribution. For large correlation matrices, where the size of the output vector is larger than the available DSP blocks in a single FPGA, the proposed algorithm offers the flexibility to implement the design in a single FPGA which is not possible using the currently available approach in [TL08].

Experimental results have shown that, depending on the structure of the correlation matrix of interest, it is not necessary to allocate the full precision of the DSP block to quantize the matrix coefficients as lower bit widths could be sufficient to achieve the required level of approximation error in the lower triangular matrix. On the performance point of view, although the approach in [TL08] outperforms the proposed approach, it does not offer the FPGA system designer to fully explore the design space, a feature which is provided by the proposed approach.

As the work carried out in this chapter contributes to the initial stages of the overall investigation, the preliminary results and findings obtained in this chapter will be further explored throughout the remainder of this thesis, especially in Chapter 4.

Chapter 4

An Optimized Hardware Multivariate Gaussian Random Number Generator

4.1 Introduction

A methodology to generate random samples from any given multivariate Gaussian distribution along with the relevant hardware architecture to be implemented on an FPGA has been presented in Chapter 3. The main feature for the approach proposed in Chapter 3 is the flexibility to produce multivariate Gaussian random number generators for any given resource constraints, a missing ingredient in the approach in [TL08]. However, the approach in Chapter 3 does not exploit the full flexibility of the design space since the precision of the data path is kept fixed throughout the design as the architecture is con-

structed from DSP blocks. An important conclusion that can be drawn from the experimental results in Chapter 3 is that different precision levels can be allocated to the different computation paths in the hardware architecture of a MVGRNG given target correlation matrices with different eigenvalue profiles.

The work presented in this chapter addresses this precision issue by utilizing word-length optimization techniques in order to produce an architecture which consists of computational paths with different precision requirements. The proposed algorithm in the chapter approaches the optimization problem from a different perspective to the approach in Chapter 3 as this work considers the approximation of the correlation matrix itself rather than the lower triangular matrix $\mathbf{A}$. Moreover, the use of Eigenvalue decomposition algorithm is introduced to the framework in order to approximate the correlation matrix of interest leading to improved hardware resource utilization.

Additionally, the algorithm for the correlation matrix approximation presented in Chapter 3 does not take into account the error due to truncation/rounding operations in the data path. A more accurate error calculation model is described in this chapter where an analysis on the impact of the error due to truncation/rounding operations in the data path is performed. Based on the findings from this analysis, a more accurate error estimation model is integrated into the algorithm in order to provide a more optimized hardware design for an MVGRNG.

The major contributions of this chapter is summarized as follows:

- The use of Eigenvalue decomposition algorithm to minimize the hardware

resource utilization of a multivariate Gaussian random number generator leading 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 data path is performed providing an analytical expression of such injected errors into the system.

- A novel methodology to combat the impact of the truncation/rounding error is introduced, targeting the production of random samples whose distribution is as close as possible to the target distribution of interest.

The organization of this chapter is as follows. An alternative method to the approach proposed in Chapter 3 for the approximation of the correlation matrix of interest is introduced in Section 4.2. In Section 4.3, the proposed hardware architecture for the implementation of an MVGRNG on an FPGA is presented. An in-depth analysis of the error in the approximation of the input correlation matrix is described in Section 4.4. As a consequence, the more accurate error model is integrated into the proposed methodology which is presented in Section 4.5 where various optimization algorithms are also introduced. The evaluation of the proposed approach is presented in Section 4.6.2 where its performance is compared with [TL08] and the approach proposed in Chapter 3. Section 4.7 finally concludes this chapter.

4.2 Approximation of Correlation Matrix using Eigenvalue Decomposition Algorithm

Many methods exist in the generation of multivariate Gaussian random numbers. The Triangular Factorization method is used in Chapter 3 as well as the approach in [TL08]. The approach presented in this chapter utilizes an alternative method which decomposes the correlation matrix $\mathbf{\Sigma}$ by Eigenvalue decomposition [CW06]. It is essential to distinguish the main difference between the approach in this chapter and the one in Chapter 3 where the approximation is performed on the lower triangular matrix, a resultant matrix from applying Cholesky factorization to the correlation matrix.

Using SVD algorithm, $\mathbf{\Sigma}$ can be expressed as $\mathbf{\Sigma} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ since a correlation matrix is symmetric, thus $\mathbf{\Lambda}$ is also symmetric. $\mathbf{U}$ 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 $\lambda_1, ..., \lambda_N$, where N is the matrix order. 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 (4.1), where $\mathbf{z} \sim N(\mathbf{0}, \mathbf{I})$.

$$
\begin{aligned}
\mathbf{x} &= \mathbf{A}\mathbf{z} + \mathbf{m} \\
&= \mathbf{U}\mathbf{\Lambda}^{1/2}\mathbf{z} + \mathbf{m} \\
&= (\sqrt{\lambda}_1\mathbf{u}_1 z_1 + \sqrt{\lambda}_2\mathbf{u}_2 z_2 + ... + \sqrt{\lambda}_K\mathbf{u}_K z_K) + \mathbf{m} \\
&= \sum_{k=1}^{K}(\sqrt{\lambda}_k\mathbf{u}_k z_k) + \mathbf{m}. \qquad (4.1)
\end{aligned}
$$

The value of K, which is the number of decomposition levels, should equal to the rank of the correlation matrix $\mathbf{\Sigma}$ for full representation. Thus, the correlation 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 correlation matrix $\mathbf{\Sigma}$ using K levels of decomposition, an error is introduced to the system due to the deviation of the approximated correlation matrix from the original one. In this chapter, the metric that is used to quantify this error is the mean square error which is given in (4.2).

$$
F(\mathbf{\Sigma}, \overline{\mathbf{\Sigma}}) := \frac{1}{N^2}\|\mathbf{\Sigma} - \overline{\mathbf{\Sigma}}\|^2, \qquad (4.2)
$$

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

As will be demonstrated later on in this chapter, 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 author's knowledge, this technique has not previously been applied to the hardware implementation of a multivariate Gaussian random number generator.

4.3 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 chapter we will focus on the generation of random samples from a centralized Gaussian distribution, that is a distribution with zero mean. Any other non-centralized distribution with the same correlation can be produced by a simple offset of the generated vectors. Consider a correlation matrix $\mathbf{\Sigma}$, the proposed algorithm applies the Eigenvalue decomposition algorithm to express $\mathbf{\Sigma} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$. According to (4.1), the random samples can be generated as in (4.3)

$$\mathbf{x} = \sum_{k=1}^{K} \sqrt{\lambda_k}\mathbf{u}_k z_k \simeq \sum_{k=1}^{K} \mathbf{c}^k z_k, \tag{4.3}$$

where $\mathbf{c}^k$ denotes a product of $\sqrt{\lambda_k} \cdot \mathbf{u}_k$ after the quantization stage with the desired word-length. In this approach, the vectors $\mathbf{c}^k$ are calculated by feeding back the error due to quantization of the previous decomposition levels, a method that was introduced in [BCC05].

The multivariate Gaussian samples are generated as the sum of products of $\mathbf{c}$ and z and, thus, the various decomposition levels k 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 data path can be varied as opposed to [TL08] and the approach in Chapter 3 where the precision of the architecture is fixed to 18 bits as only DSP48 blocks are utilized. As will be demonstrated later, quantizing the coefficients of the vectors $\mathbf{c}^k$ to lower bit widths could be sufficient to achieve the target approximation error 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 4.1 where five blocks are used with different precisions namely p_1, p_2, p_3, p_4 and p_5. In the figure, $\mathbf{c}^k$ denotes the vector coefficient at the k^{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 approach, registers are deployed for small matrix order ($N < 20$) while block

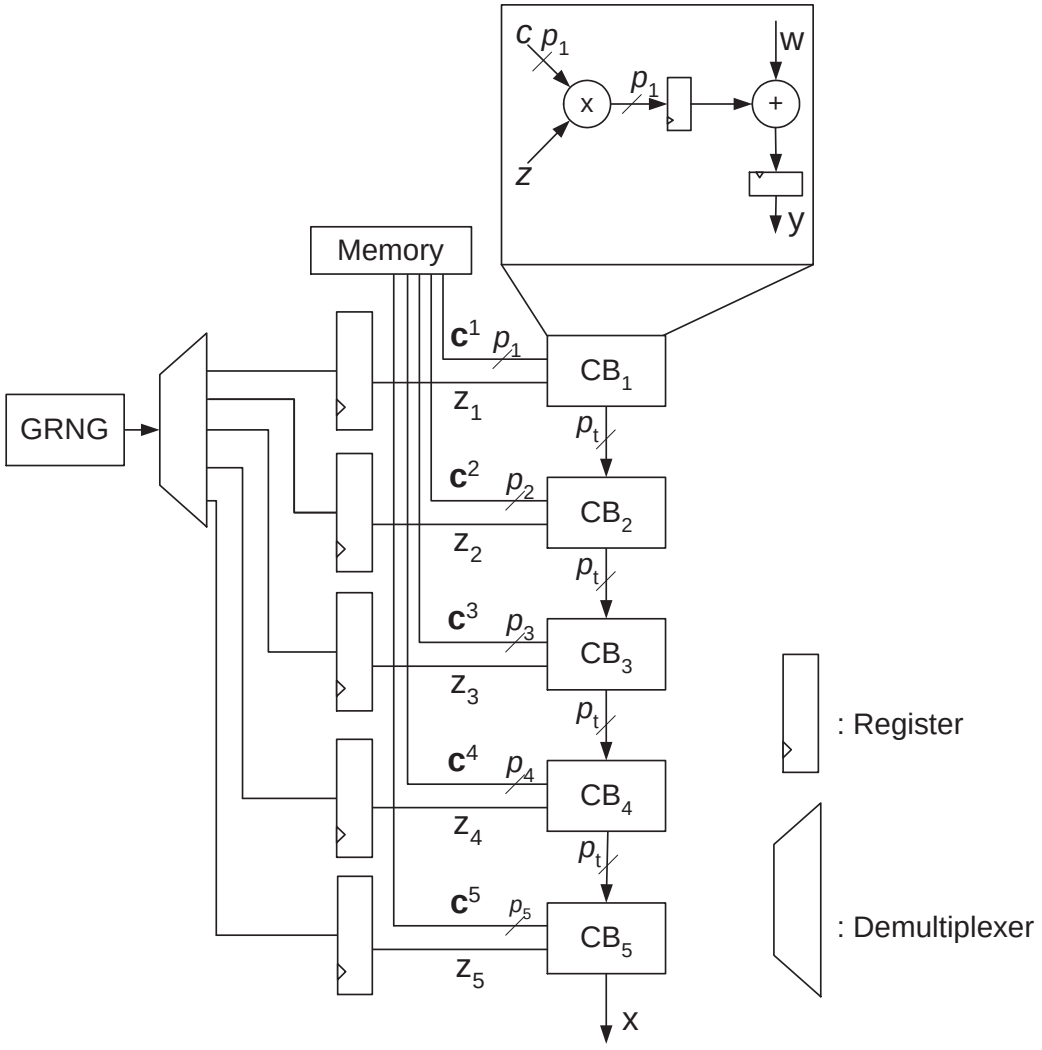


Figure 4.1: Hardware architecture.

RAMs are used for larger matrix order.

The main functionality of each CB is to multiply the coefficients c with a univariate Gaussian sample z. In this design, a single GRNG is required to produce the univariate Gaussian random numbers.

The product cz is added to the same product from a previous CB. The same operation is performed at every level of decomposition where each level is represented by a CB. The value of x is valid at the output after the coefficients

of the final CB have been multiplied and added with the product from previous decomposition levels. A complete vector of random numbers $\mathbf{x}$ will be ready after N clock cycles where N is the 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 which 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 will be demonstrated later in this chapter.

4.4 Analysis of Approximation Error

Let us consider a simple scenario of 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 4.2 where two multipliers and an adder are utilized to map the expression in (4.1) to generate random samples. Due to the use of fixed point number representation in this work, the coefficients of the correlation matrix are quantized to a certain precision. The following notation

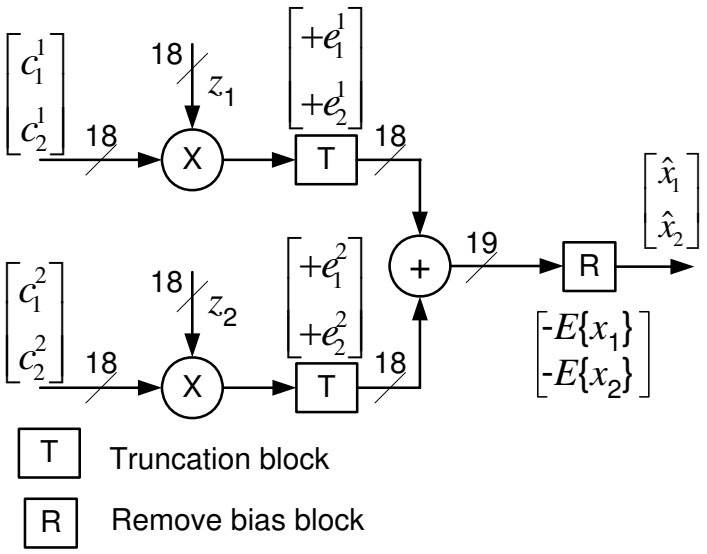


Figure 4.2: An architecture for a 2D MVGRNG.

is used in this chapter. 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 data paths denote the word-length employed. "T" represents a truncation block, where the word-length of the signal is truncated to the user-specified precision, while "R" is a block which subtracts a bias from the generated samples. The details regarding this block will be discussed later in this section.

The proposed system generates a random vector $\mathbf{x}$ from univariate random samples $z_1, z_2 \sim N(0,1)$ as described in (4.4). It is important to note that the truncation of the data path introduces an error e to the system. At this stage of the analysis, only the error due to truncation after the multiplier is considered,

any error due to truncation operations after the adder will be considered later in this chapter.

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \left(\begin{bmatrix} c_1^1 \\ c_2^1 \end{bmatrix} z_1 + \begin{bmatrix} e_1^1 \\ e_2^1 \end{bmatrix} \right) + \left(\begin{bmatrix} c_1^2 \\ c_2^2 \end{bmatrix} z_2 + \begin{bmatrix} e_1^2 \\ e_2^2 \end{bmatrix} \right). \tag{4.4}$$

The expected value of the random vectors $\mathbf{x}$ in (4.4) is:

$$\begin{aligned} E\left\{ \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right\} &= \begin{bmatrix} E\{c_1^1 z_1 + e_1^1 + c_1^2 z_2 + e_1^2\} \\ E\{c_2^1 z_1 + e_2^1 + c_2^2 z_2 + e_2^2\} \end{bmatrix} \\ &= \begin{bmatrix} E\{e_1^1\} + E\{e_1^2\} \\ E\{e_2^1\} + E\{e_2^2\} \end{bmatrix}, \end{aligned} \tag{4.5}$$

since the expectation of Gaussian random samples, $E\{z\}$, is zero. The expression in (4.5) 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:

$$\begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} E\{e_1^1\} + E\{e_1^2\} \\ E\{e_2^1\} + E\{e_2^2\} \end{bmatrix}. \tag{4.6}$$

This operation is realized through block "R" in Figure 4.2.

The correlation matrix $\overline{\mathbf{\Sigma}}$ of the generated samples $\hat{\mathbf{x}}$ is given in (4.7).

$$
\begin{aligned}
\overline{\mathbf{\Sigma}} &= E\{\hat{\mathbf{x}}\hat{\mathbf{x}}^T\} - E\{\hat{\mathbf{x}}\}E\{\hat{\mathbf{x}}\}^T \\
&= E\left\{ \begin{bmatrix} \hat{x}_1 \\ \\ \hat{x}_2 \end{bmatrix} \begin{bmatrix} \hat{x}_1 \\ \\ \hat{x}_2 \end{bmatrix}^T \right\} \\
&= \begin{bmatrix} E\{\hat{x}_1^2\} & E\{\hat{x}_1\hat{x}_2\} \\ \\ E\{\hat{x}_2\hat{x}_1\} & E\{\hat{x}_2^2\} \end{bmatrix}. \qquad (4.7)
\end{aligned}
$$

The non-diagonal elements in the correlation matrix are given by the expectation $E\{\hat{x}_1\hat{x}_2\}$ as in (4.8).

$$
\begin{aligned}
E\{\hat{x}_1\hat{x}_2\} &= c_1^1c_2^1 + c_1^2c_2^2 + E\{e_1^1e_2^1\}E\{e_1^2e_2^2\} - E\{e_1^1\}E\{e_2^1\} - E\{e_1^2\}E\{e_2^2\} \\
&= E\{x_1x_2\}_{input} + \rho_{e_1^1e_2^1}\sigma_{e_1^1}\sigma_{e_2^1} + \rho_{e_1^2e_2^2}\sigma_{e_1^2}\sigma_{e_2^2}, \qquad (4.8)
\end{aligned}
$$

where $E\{x_1x_2\}_{input}$ denotes the input targeted correlation 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 (4.9).

$$
\sigma_e^2 = \frac{1}{12}2^{2d}(2^{-2p_2} - 2^{-2p_1}), \qquad (4.9)
$$

where p denotes the word-length of the signal and d refers to its scale [CCL04]. In addition, $E\{e\}$ is given by $-2^{d-1}(2^{-p_2} - 2^{-p_1})$. Similarly, the diagonal

elements of the sample correlation matrix can be expressed as in (4.10):

$$E\{\hat{x}_1^2\} = E\{x_1^2\}_{input} + \sigma^2_{e_1^1} + \sigma^2_{e_2^1}$$
$$E\{\hat{x}_2^2\} = E\{x_2^2\}_{input} + \sigma^2_{e_1^2} + \sigma^2_{e_2^2}. \tag{4.10}$$

Note that the above analysis does not take into account the truncation operations in the adder path of the architecture. Thus the correlation matrix of the generated samples is given in (4.11).

$$\overline{\mathbf{\Sigma}} = \mathbf{C}_K + \mathbf{W}, \tag{4.11}$$

where $\mathbf{C}_K$ is a matrix that approximates the input correlation 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 correlation 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 (4.12). Note that $w_{i,j}$ is an error due to the correlation of truncation errors from samples x_i and x_j.

$$w_{i,j} = \sum_{k=1}^{K} \rho_{e_i^k e_j^k} \sigma_{e_i^k} \sigma_{e_j^k}. \tag{4.12}$$

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 (4.12) simplifies to $\sum_{k=1}^{K} \sigma^2_{e_i^k}$. The expression in (4.12) illustrates

that the correlation of the truncation errors of each computational block contributes to the final approximation error of the correlation matrix.

Throughout this section, we have focused on the discussion of the error in the approximation of the correlation matrix due to truncation operations in the data path. 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.

4.4.1 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 (4.13)

$$\begin{aligned} e_1 &= c_1 z - [c_1 z]_{tr} \\ e_2 &= c_2 z - [c_2 z]_{tr}, \end{aligned} \tag{4.13}$$

where $[.]_{tr}$ denotes the truncated version of the expression. The correlation ρ is independent of any scaling of the random sequence z. By multiplying the

sequence of random samples z by α, the expressions of the truncation errors are given in (4.14)

$$\begin{aligned} e_1' &= c_1(\alpha z) - [c_1(\alpha z)]_{tr} = (\alpha c_1)z - [(\alpha c_1)z]_{tr} \\ e_2' &= c_2(\alpha z) - [c_2(\alpha z)]_{tr} = (\alpha c_2)z - [(\alpha c_2)z]_{tr}. \end{aligned} \tag{4.14}$$

It is essential to note that the two scaled truncation errors have the same correlation as the original truncation errors. Thus,

$$\rho(\alpha c_1, \alpha c_2) = \rho(c_1, c_2). \tag{4.15}$$

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

Figure 4.3 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. In statistics, a p-value is defined as a measurement to estimate how much evidence there is against the null hypothesis. The null hypothesis is known as the default position where it represents the hypothesis of no change or no effect. The smaller the p-value,

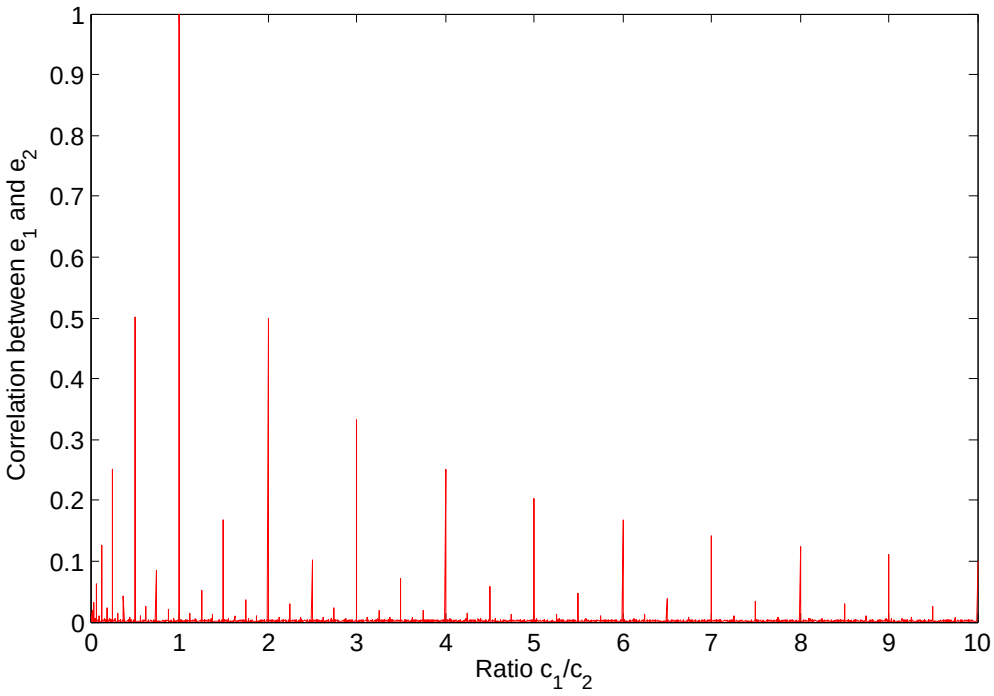


Figure 4.3: Correlation between truncation errors.

the better evidence we have against the null hypothesis.

4.4.2 Two Stage Error Model

The error analysis up until this point focuses on the impact of the truncation error which occurs after the multiplication stage only, while it does not consider the second truncation stage which can take place in the addition path of the architecture. The analysis in this section provides a more generalized description of the error model and demonstrates that the impact of the second truncation stage can be safely ignored except for one special case.

Figure 4.4 depicts an instance of a hardware architecture which is used to generate a 3-dimensional vector $\mathbf{x}$. In this instance, the architecture comprises of three sets of multipliers and adders indicating the use of three levels of

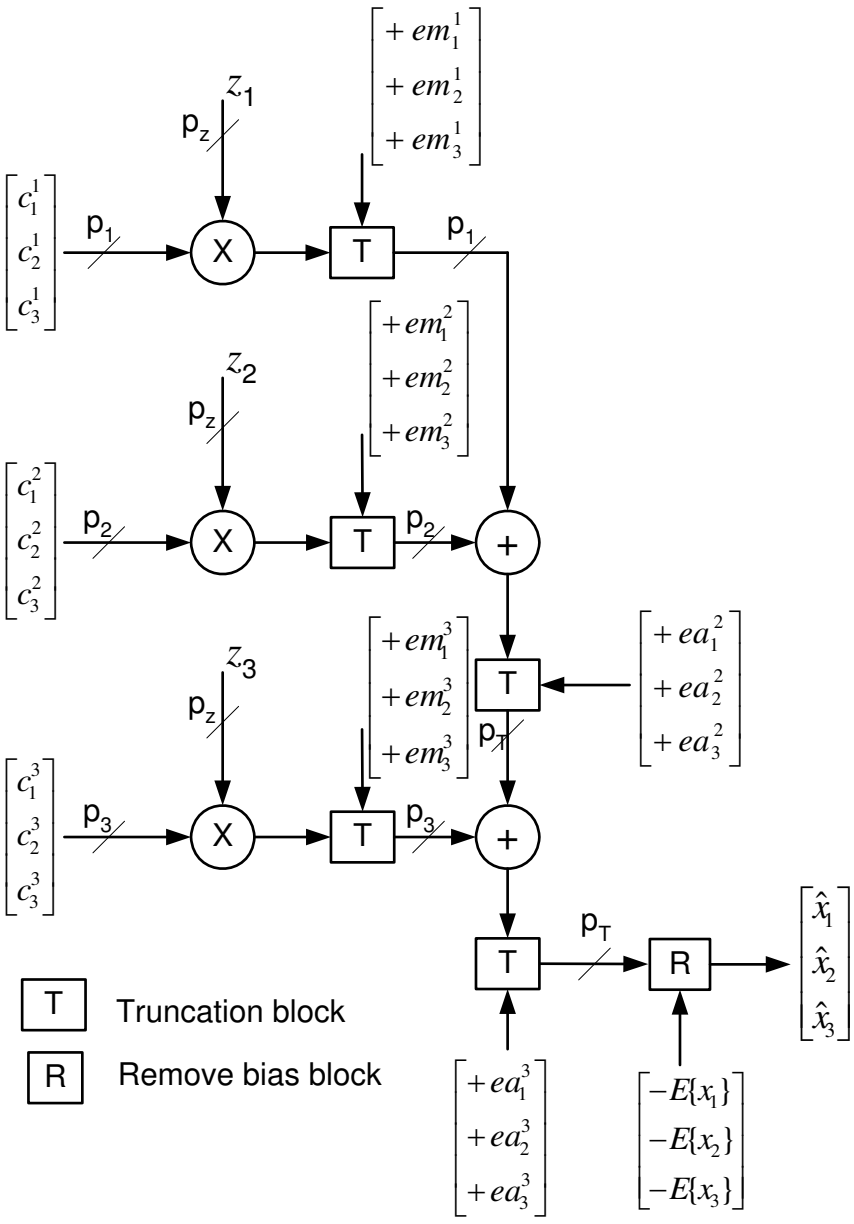


Figure 4.4: Hardware Architecture for a 3D MVGRNG.

decomposition. The coefficients of the correlation matrix are quantized to a certain precision since fixed point number representation is employed in this work.

The system in Figure 4.4 generates a random vector $\mathbf{x}$ where the expressions of its elements are given in (4.16)

$$\begin{aligned} x_1 &= c_1^1 z_1 + em_1^1 + c_1^2 z_2 + em_1^2 + ea_1^2 + c_1^3 z_3 + em_1^3 + ea_1^3 \\ x_2 &= c_2^1 z_1 + em_2^1 + c_2^2 z_2 + em_2^2 + ea_2^2 + c_2^3 z_3 + em_2^3 + ea_2^3 \\ x_3 &= c_3^1 z_1 + em_3^1 + c_3^2 z_2 + em_3^2 + ea_3^2 + c_3^3 z_3 + em_3^3 + ea_3^3, \end{aligned} \tag{4.16}$$

where z_i denotes univariate Gaussian random samples, em represents the error introduced to the system due to the first stage truncation operations after the multiplication, while ea denotes the error due to second stage truncation in the addition path.

The correlation matrix $\overline{\mathbf{\Sigma}}$ of the generated samples $\hat{\mathbf{x}}$ is given in (4.17).

$$
\begin{aligned}
\overline{\mathbf{\Sigma}} &= E\{\hat{\mathbf{x}}\hat{\mathbf{x}}^T\} - E\{\hat{\mathbf{x}}\}E\{\hat{\mathbf{x}}\}^T \\
&= E\left\{ \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \end{bmatrix} \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \end{bmatrix}^T \right\} \\
&= \begin{bmatrix} E\{\hat{x}_1^2\} & E\{\hat{x}_1\hat{x}_2\} & E\{\hat{x}_1\hat{x}_3\} \\ E\{\hat{x}_2\hat{x}_1\} & E\{\hat{x}_2^2\} & E\{\hat{x}_2\hat{x}_3\} \\ E\{\hat{x}_3\hat{x}_1\} & E\{\hat{x}_3\hat{x}_2\} & E\{\hat{x}_3^2\} \end{bmatrix},
\end{aligned} \tag{4.17}
$$

as the expectation $E\{\hat{\mathbf{x}}\}$ is zero. Let us consider one of the non-diagonal elements in the correlation matrix, which is given by the expectation $E\{\hat{x}_1\hat{x}_2\}$. This can be expressed as in (4.18).

$$
\begin{aligned}
E\{\hat{x}_1\hat{x}_2\} = {} & c_1^1 c_2^1 + c_1^2 c_2^2 + c_1^3 c_2^3 + \rho_{em_1^1 em_2^1}\sigma_{em_1^1}\sigma_{em_2^1} \\
& + \rho_{em_1^2 em_2^2}\sigma_{em_1^2}\sigma_{em_2^2} + \rho_{em_1^3 em_2^3}\sigma_{em_1^3}\sigma_{em_2^3} \\
& + \rho_{ea_1^2 ea_2^2}\sigma_{ea_1^2}\sigma_{ea_2^2} + \rho_{ea_1^3 ea_2^3}\sigma_{ea_1^3}\sigma_{ea_2^3},
\end{aligned} \tag{4.18}
$$

since $E\{z_i^2\} = 1$ and $E\{z_i z_j\} = 0$ for $i \neq j$ by definition. $\rho_{e_b^a e_d^c}$ is the correlation coefficient between the truncation errors e_b^a and e_d^c. σ denotes the variance of the error inserted to the system due to truncation of any given signal (p_1, d) to a signal (p_2, d) and is given by (4.9).

From the general expression of the approximated correlation matrix $\overline{\mathbf{\Sigma}}$ in (4.11), the general expression of the non-diagonal elements $w_{i,j}$ of $\mathbf{W}$ is given

in (4.19).

$$w_{i,j} = \sum_{k=1}^{K} \left\{ \rho_{em_i^k em_j^k} \sigma_{em_i^k} \sigma_{em_j^k} + \rho_{ea_i^k ea_j^k} \sigma_{ea_i^k} \sigma_{ea_j^k} \right\}. \tag{4.19}$$

The expression for the diagonal elements $w_{i,i}$ is given in (4.20)

$$w_{i,i} = \sum_{k=1}^{K} \left\{ \sigma^2_{em_i^k} + \sigma^2_{ea_i^k} \right\}, \tag{4.20}$$

since the correlations $\rho_{em_i^k em_i^k}$ and $\rho_{ea_i^k ea_i^k}$ are one.

4.4.3 Impact of the Correlation between Truncation Errors in the Addition Path

In this section, the behaviour of the correlation between the truncation errors in the addition path is investigated. In order to see its impact, a pair of adjacent multipliers is considered as their outputs represent two signals from consecutive decomposition levels to be added together. Let us express two pairs of coefficients from Figure 4.4 as the following ratios, $\frac{c_2^1}{c_1^1}$ and $\frac{c_2^2}{c_1^2}$.

Computer simulations are performed on the error model given in Figure 4.4 for different values of the ratios $\frac{c_2^1}{c_1^1}$ and $\frac{c_2^2}{c_1^2}$ from 0 to 1 using 500,000 univariate Gaussian random samples z for each instance. The relationship between the coefficients is expressed as a ratio since the correlation between the resulting truncation errors is a function of the ratio of the coefficients, and not of their individual values as has been demonstrated previously.

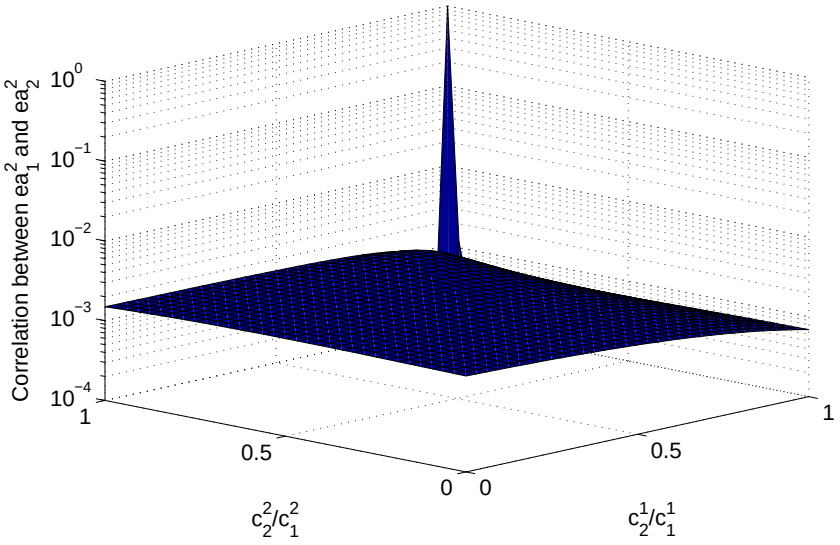


Figure 4.5: Correlation between Truncation Errors in the Addition Path.

Figure 4.5 illustrates a plot of the correlation between the truncation errors ea_1^2 and ea_2^2 which correspond to the truncation of the addition of the products of $c_1^1 z_1$ with $c_1^2 z_2$ and $c_2^1 z_1$ with $c_2^2 z_2$ respectively, over a range of the specified ratios. It can clearly be seen from the graph that the correlation between the two truncation errors is very low for all values of ratios except when both ratios are equal to 1. Hence, the correlation between truncation error due to the addition operation is insignificant except for the case where the individual pair of coefficients have the same ratio. Thus, the expressions in (4.19) and (4.20) are given in (4.21) and (4.22) respectively.

$$w_{i,j} = \sum_{k=1}^{K} \left\{ \rho_{em_i^k em_j^k} \sigma_{em_i^k} \sigma_{em_j^k} + \beta \left(\frac{c_i^k}{c_j^k} \right) \rho_{em_i^n em_j^k} \sigma_{em_i^k} \sigma_{em_j^k} \right\}. \tag{4.21}$$

$$w_{i,i} = \sum_{k=1}^{K} \left\{ \sigma^2_{em_i^k} + \beta \left(\frac{c_i^k}{c_j^k} \right) \sigma^2_{ea_i^k} \right\}, \tag{4.22}$$

where $\beta \left(\frac{c_i^k}{c_j^k} \right)$ is equal to one only when the ratios of the coefficients in the two consecutive decomposition level are equal. This can be expressed as, $\left(\frac{c_i^k}{c_j^k} \right) = \left(\frac{c_i^{k+1}}{c_j^{k+1}} \right)$.

4.5 Proposed Methodology

In section 4.3, a hardware architecture based on the Eigenvalue decomposition algorithm to generate random samples from a multivariate Gaussian distribution with a given correlation matrix has been introduced, and an error analysis has been performed in section 4.4. In this section, the methodology of mapping such architecture to reconfigurable hardware is described.

The use of mean square error of the approximation error between the input correlation matrix $\mathbf{\Sigma}$ and the sample correlation matrix has so far been the focus of this work. However, the proposed approach can be generalized to optimize other metrics as well. Considering the mean square error as the target metric, three different approaches for the selection of appropriate coefficients for each decomposition level are proposed. 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. *Algorithm 3* inherits the property of *Algorithm 2* with an

Table 4.1: Summary of the three Algorithms.

	min(Quantization Error)	min(Truncation Error)	min(Correlation between Truncation Error)
Algorithm 1	Yes		
Algorithm 2	Yes	Yes	
Algorithm 3	Yes	Yes	Yes

addition of an optimization technique to minimize 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.

A summary of all three algorithms proposed in this chapter is presented in Table 4.1.

4.5.1 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 (4.23).

$$f_1(\mathbf{c}_K) = \frac{1}{N^2}\|\mathbf{R}_{K-1} - (\mathbf{c}_K\mathbf{c}_K^T)\|^2, \tag{4.23}$$

where the $\mathbf{R}_{K-1}$ denotes the remaining of the original correlation matrix after $K-1$ levels of decomposition, where $\mathbf{R}_0$ is defined as the original correlation 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 correlation matrix and the estimated correlation matrix using 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.

4.5.2 Algorithm 2

In *Algorithm 2*, the overall approximation error of the correlation 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 (4.24).

$$f_2(\mathbf{c}_K) = \frac{1}{N^2}\|\mathbf{R}_{K-1} - (\mathbf{c}_K\mathbf{c}_K^T + \mathbf{W})\|^2, \tag{4.24}$$

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

4.5.3 Algorithm 3

In order to combat the correlation effects of error due to the truncation in the data path, 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 correlation matrix by minimizing the objective function in (4.25).

$$f_3(\mathbf{c}_K) = \frac{1}{N^2}\|\mathbf{R}_{K-1} - (\mathbf{c}_K\mathbf{c}_K^T + \mathbf{W}(\rho(\mathbf{c}_K)))\|^2, \tag{4.25}$$

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 4.3 in order to find the coefficients that minimize (4.25). 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.

4.5.4 Overview of Proposed Approach

Similar to the existing approaches, the elements of each random vector are serially generated resulting in 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

correlation 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 pre-defined hardware computational blocks with different user-specified precisions for the implementation of each computational path. The library provides a data set associated with each computational block as inputs to the proposed algorithm. These are the hardware resource usage and the precision requirements for each block. The addition of more computational blocks is straightforward and will not be discussed further in the remainder of this thesis.

The data set in the library will be used by the proposed algorithm to determine a set of best possible designs for a given multivariate Gaussian distribution. 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.

In order to visualize the full functionality of the proposed approach, the overall structure of this methodology is illustrated in Figure 4.6, in which there are two main processes. The first process is the proposed algorithm which decomposes any given correlation matrix for a specified approximation error. The hardware library contains information regarding the pre-defined hardware blocks including the word-length and the corresponding resource usage of each

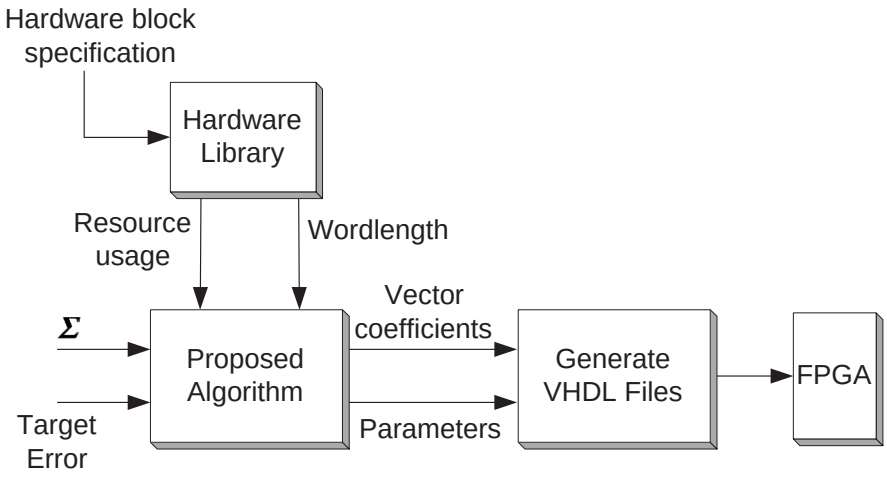


Figure 4.6: Overview of the proposed approach.

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.

4.5.5 Overall Methodology

In this section, an overall description of the proposed methodology is given, which is depicted in Figure 4.7. The inputs to the system are a correlation 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 wordlength of its data path, and finally the target approximation error between the original input matrix and the expected correlation matrix. Note that the proposed methodology is fully parameterized such that it is able to support any number of pre-defined hardware blocks.

The proposed methodology is divided into three stages. In the first stage,

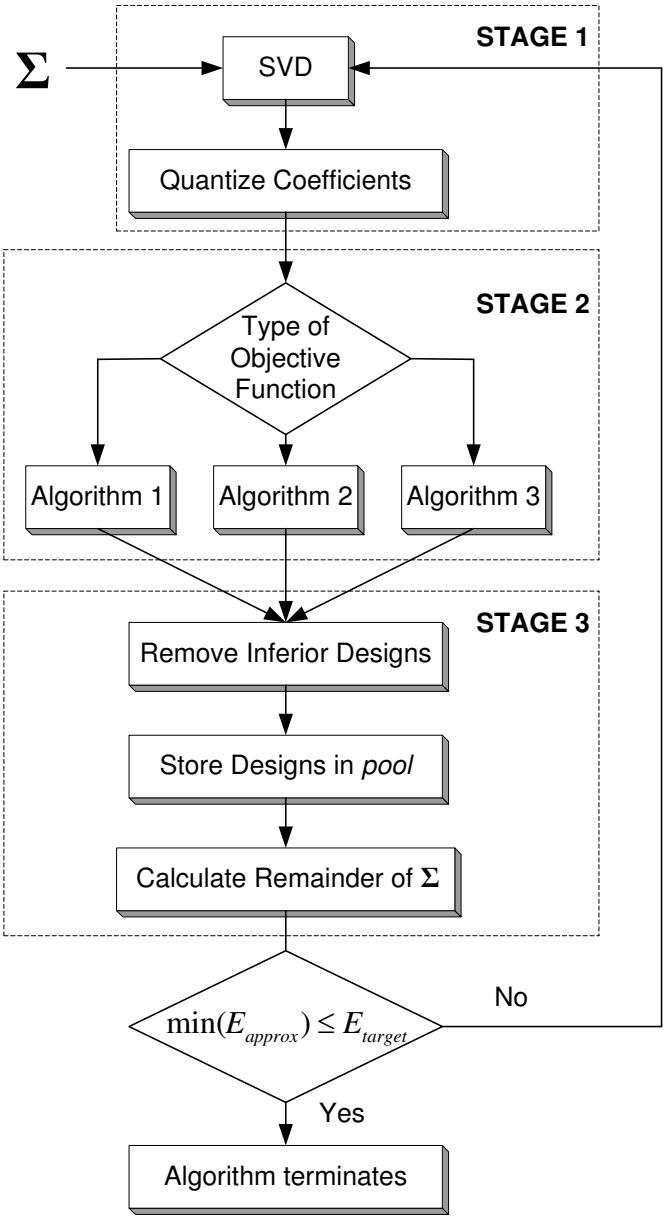


Figure 4.7: Outline of the proposed algorithm.

the correlation matrix $\mathbf{\Sigma}$ is decomposed into vector $\mathbf{c}$ and the 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 proposed algorithms, *Algorithms 1*, *2* and *3*. 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 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 a specified termination condition is met. In the proposed algorithm, a termination condition can be a given hardware resource constraint or the required approximation error, for instance the algorithm terminates when a design with the minimum approximation error meets the target value

specified by the user.

4.6 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 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.

4.6.1 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 pre-defined in the hardware library for the construction of the multivariate Gaussian random number generator. Table 6.2 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 as the precision increases. In order to make a direct comparison between the proposed approach and [TL08] and the approach in Chapter 3, 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.

Table 4.2: Computational Block synthesis results.

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

4.6.2 Evaluation of Proposed Methodology

In this section the performance of the proposed methodology in comparison to the existing approaches from the literature is examined. The performance of the three proposed algorithms is assessed and it is compared against the state-of-the-art approaches in [TL08] and the approach in Chapter 3.

The five algorithms are applied to a randomly generated 5x5 correlation matrix to produce many instances of MVGRNGs. 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 correlation matrices. A sample correlation matrix is constructed from working out the correlation between actual random numbers produced for each of the variables.

The plot in Figure 4.8 illustrates the mean square error in the approxi-

mation of the original correlation matrix and the sample correlation matrices for a range of MVGRNGs generated by the five algorithms. From the plot it is important to note that the three proposed algorithms and the approach in Chapter 3 have the flexibility to produce designs across all of the available design space. In contrast, the approach in [TL08] can only produce one fixed design for a given correlation matrix. This is due to the decomposition technique utilized in the proposed approach and the approach in Chapter 3.

Let us consider the relative performance between the three proposed algorithms and the approach in [TL08]. The plots show that, for similar resource utilization, the designs generated from *Algorithms* 1, 2 and 3 produce a better approximation error than [TL08], achieving a reduction in the approximation error by 91%, 94% and 97% respectively. Moreover, for similar values of approximation error in the correlation matrix, there is a 60% reduction in hardware resource utilization relative to [TL08] for all three algorithms.

Moreover, the plot demonstrates that the hardware resource utilization of the three proposed algorithms is significantly reduced in comparison to the approach in Chapter 3. This is mainly due to the use of the Eigenvalue decomposition algorithm to decompose the input correlation matrix instead of decomposing the resulting matrix from the Cholesky decomposition, which reduces the number of multiplications 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 correlation matrix. The er-

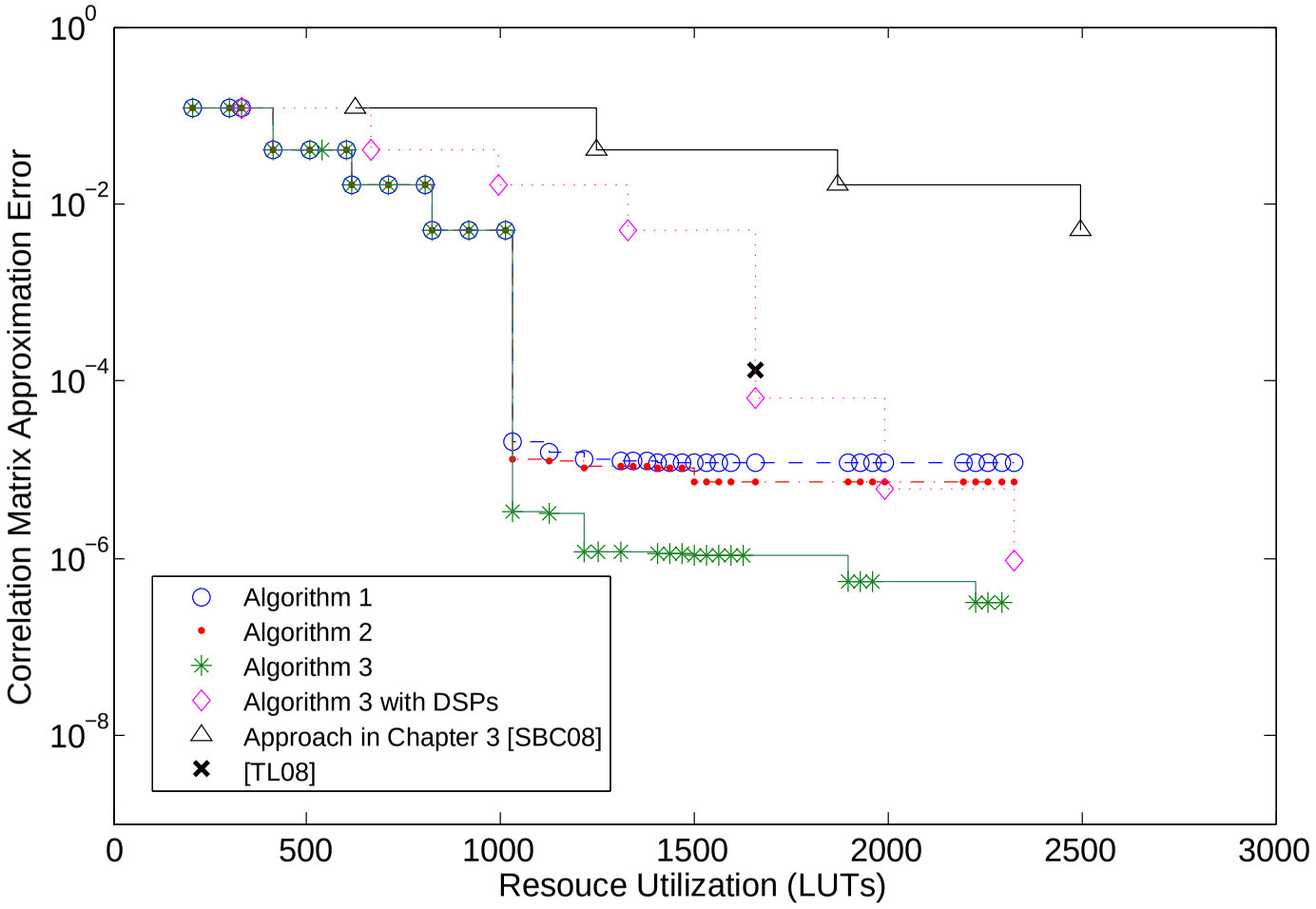


Figure 4.8: Comparison of proposed algorithms with existing approaches.

ror 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 the fact 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 data path. 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 [TL08], *Algorithm 3* has been applied to 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 [TL08] by approximately 53%. This is due to the fact that the correlation between the truncation errors of each DSP is taken into account during the selection of the coefficients.

4.6.3 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 correlation matrix from section 4.6.2.

Figure 4.9 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 levels increases is steeper when higher precision is allocated in the data paths.

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 data path.

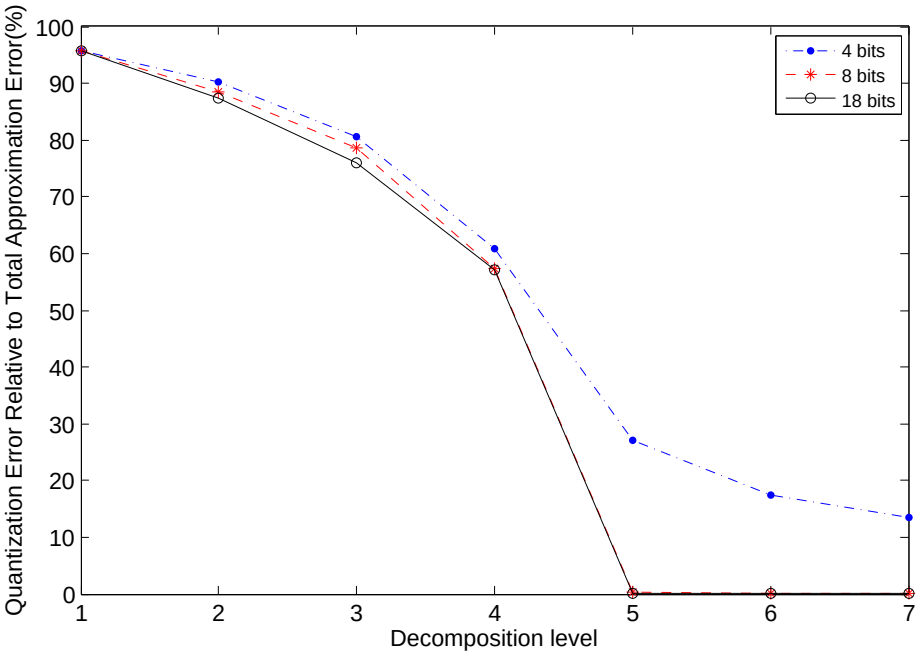


Figure 4.9: Error distribution.

4.6.4 Effects of Coefficient Optimization

Figure 4.10 illustrates a plot of the quantization error of the same 5x5 matrix as in section 4.6.2 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 more computational blocks are added 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 correlation 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 correlation matrix. *Algorithm 2* may exhibit the same behaviour as *Algorithm 3*.

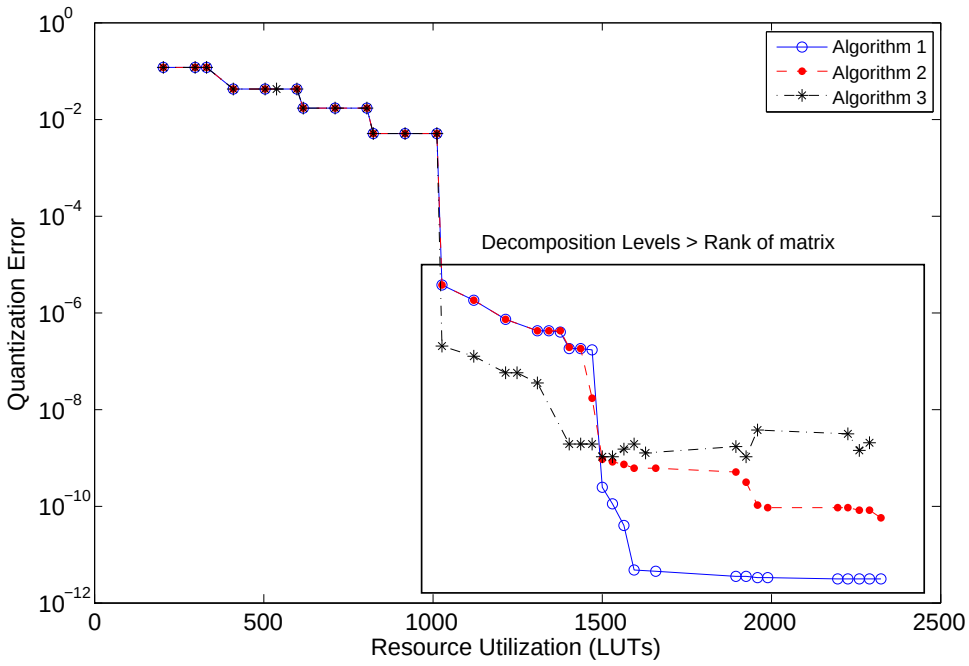


Figure 4.10: Comparison of three algorithms.

4.6.5 Analysis of Empirical 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 correlation matrix using 100,000 vectors. Similar to previous experiments, this is the result of the same 5x5 correlation matrix as in section 4.6.2.

The plots in Figures 4.11, 4.12 and 4.13 show the comparison between the estimated and the actual approximation errors from the three proposed algorithms respectively. 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*

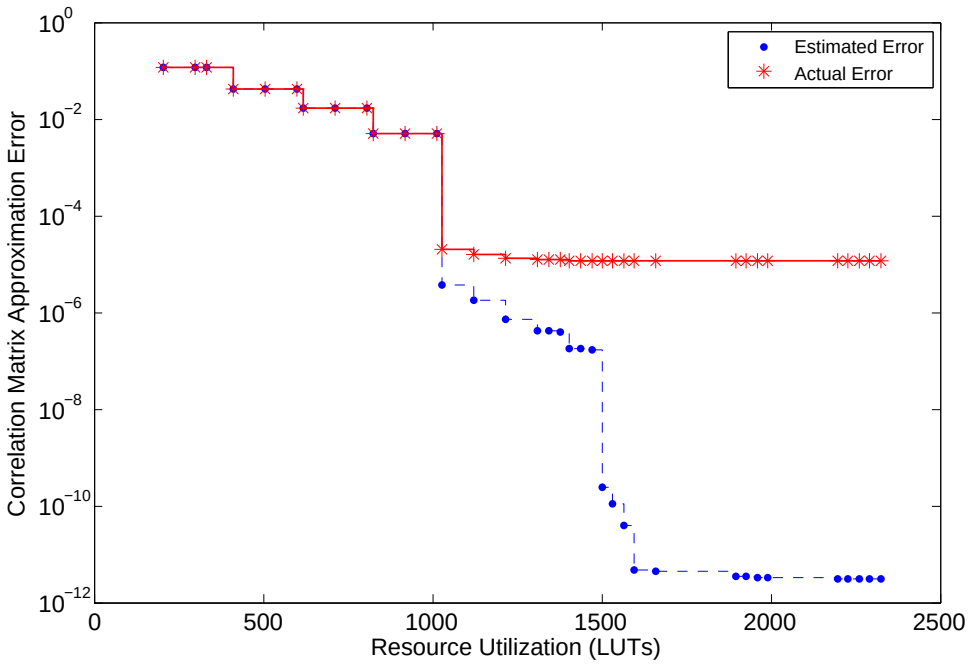


Figure 4.11: Comparison between estimated and actual error for *Algorithm 1*

is similar since both algorithms model the impact of the correlation due to the truncation in the sample correlation 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.

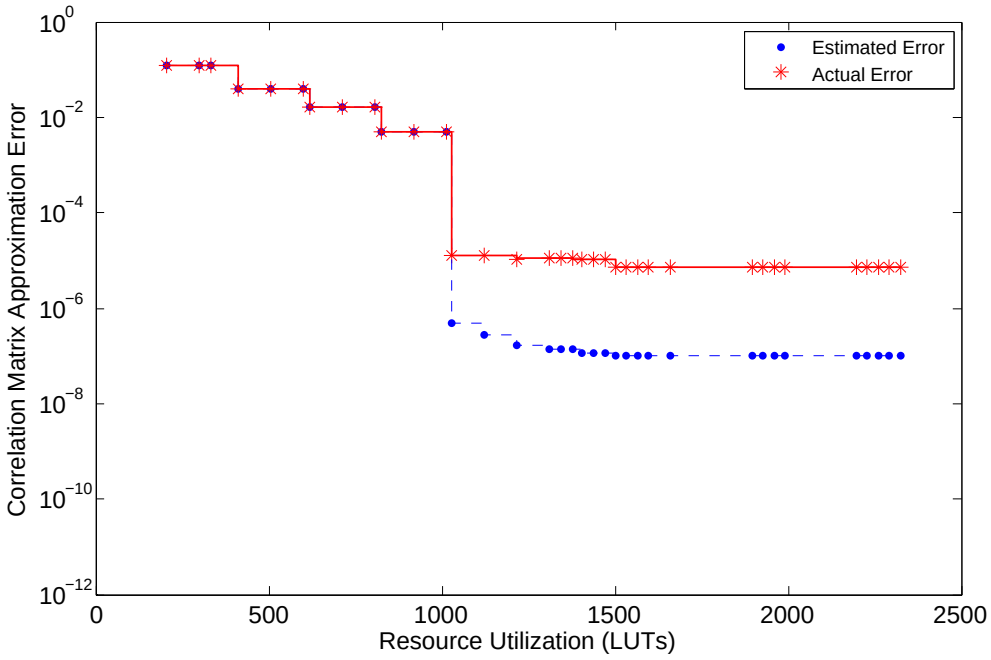


Figure 4.12: Comparison between estimated and actual error for *Algorithm 2*

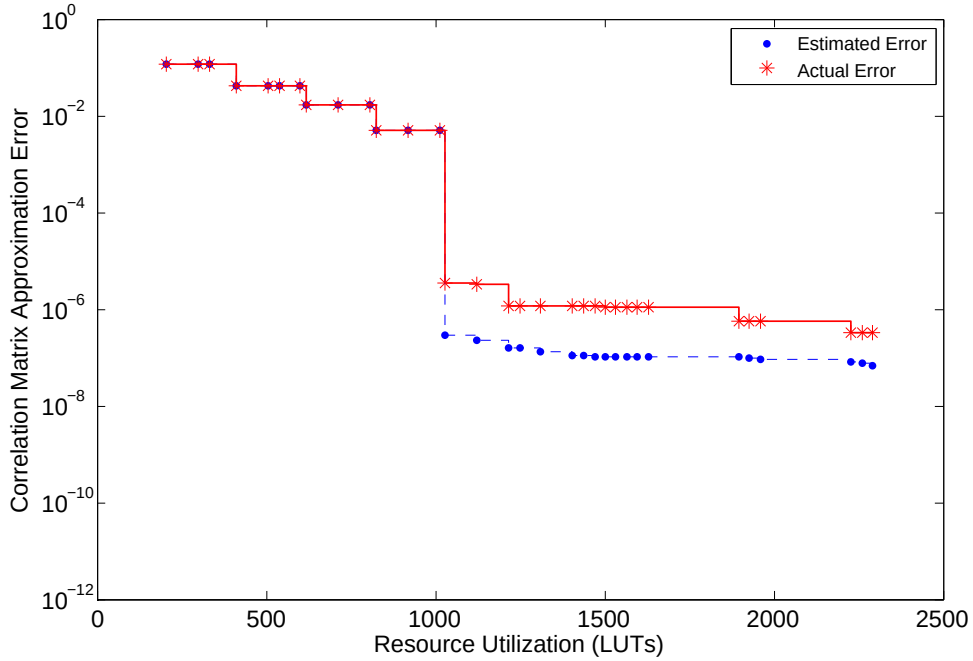


Figure 4.13: Comparison between estimated and actual error for *Algorithm 3*

4.7 Summary

This chapter 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 data path 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, the work in this chapter demonstrates the impact of the correlation of the truncation/rounding operations to the output of the system and provides a methodology to minimize it.

Chapter 5

Mapping Multiple Multivariate Gaussian Random Number Generators onto an FPGA

5.1 Introduction

The focus of this thesis so far has been on the generation of random numbers from a given multivariate Gaussian distribution implemented on an FPGA. The hardware architecture of the MVGRNG in Chapter 3 is based on the use of the SVD algorithm to approximate the lower triangular matrix obtained from applying Cholesky decomposition to the correlation matrix of interest. This enables for the flexibility to produce MVGRNGs according to any given hardware resource constraint on the FPGA. The framework proposed in Chapter 4 introduces extra dimensionality in the design optimization algorithm with the

use of word-length optimization techniques which allows custom precision requirements in the hardware design together with a more accurate error model.

Up to this point, existing approaches including [TL08] and the two approaches proposed in Chapters 3 and 4 for the implementation of the FPGA-based MVGRNGs have only concentrated on the design and optimization of a single multivariate Gaussian random number generator. However, since the MVGRNG block is usually a part of a larger application which is mapped onto an FPGA, its area requirement can be crucial as more of the hardware resource on the FPGA can be allocated for the main application. The proposed approach in this chapter adds an extra dimension to the above optimization problem by considering the mapping of multiple multivariate Gaussian distributions with the aim of reducing the required resources by exploring any redundancy that exists between the distributions.

Many applications which are driven by the MVGRNG do not require very high throughput of random numbers and so the throughput of the design can be traded off for an improved hardware resource usage. Hence, random numbers from many multivariate Gaussian distributions can be sampled from a single hardware block which is optimized for all distributions of interest. In more detail, the approach presented in this chapter exploits similarities in the precision requirements that exist between different distributions under consideration by exploring the similarities in the correlation matrices which characterize the target distributions. This subsequently creates an opportunity for effective resource sharing by trading off the throughput of the random number

generator for an improved resource usage which leads to a single generator that is optimized across all distributions of interest.

The organization of this chapter is as follows. The remaining sections of this chapter begins with Section 5.2 where the proposed approach to implement a single MVGRNG which is optimized for all input distributions. This is followed by the novel hardware architecture to be implemented on an FPGA which is presented in Section 5.3. Section 5.4 discusses the set up of the experiments to be carried out for the work in this chapter. The results and the relevant analysis are presented in Section 5.5. Finally, Section 5.6 concludes the chapter.

5.2 Proposed Algorithm

The proposed algorithm is based on the MVGRNG architecture proposed in Chapter 4, where word-length optimization techniques to produce generators with mixed precisions in the architecture are utilized. The hardware architecture in Chapter 4 is based on the Eigenvalue decomposition algorithm, which decomposes an input correlation matrix into a set of vectors. Then, an optimization process that selects the appropriate precision for each decomposition level is performed in order to achieve a good approximation to the input correlation matrix minimizing at the same time the resource utilization.

The main idea behind the work in this chapter is to reuse each decomposition block calculation, which is the most resource demanding part of the design, across many MVGRNGs. In order for this to be achievable, an opti-

mum selection of the precision to be used in the blocks should be performed such that the chosen precision should minimize the approximation error across all correlation matrices under consideration leading to a more efficient allocation of hardware resources for all distributions of interest. It is important to note that the vector coefficients for each of the correlation matrices are specific to each distribution as only the precision requirement for each decomposition is optimized to be the same for all correlation matrices of interest. Thus, no unwanted correlation is introduced between the different distributions under consideration.

Figure 5.1 gives an overview of the proposed algorithm. The main input to the proposed algorithm are the correlation matrices which characterize the target distributions, hardware blocks specification, choice of objective function and the termination condition. Firstly, all M correlation matrices are individually decomposed using the eigenvalue decomposition algorithm resulting in vectors of coefficients $\mathbf{c}$. These coefficients are quantized into fixed point representation using the user-specified word-lengths from a hardware library. As a result, many possible candidate designs are generated for each input correlation matrix exploring the approximation error-hardware resource usage design space.

In order to assess how good a design is, an error between the target correlation matrix and the approximated matrix obtained after the decomposition and quantization step is calculated. Note that the flexibility of the proposed algorithm allows for any error metric to be used as an objective function for

the optimization. In this work, the mean square error is used as the objective function (5.1).

$$MSE_K^i = \frac{1}{N_i^2} \left\| \mathbf{R}_{K-1}^i - (\mathbf{C}_K^i + \mathbf{W}^i) \right\|_F^2, \tag{5.1}$$

where K denotes the decomposition level and i denotes the i^{th} distribution of interest, $i \in [1, ..., M]$. $\mathbf{R}_{K-1}^i$ denotes the remainder of the original correlation matrix after $K-1$ levels of decomposition with $\mathbf{R}_0^i$ defined as the original correlation matrix at the start of the algorithm. The $\|.\|_F$ operator denotes the Frobenius norm while N is the order of the matrix. The approximated correlation matrix for each candidate design $\overline{\mathbf{\Sigma}}^i$ can be expressed as in (5.2)

$$\overline{\mathbf{\Sigma}}^i = \mathbf{C}_K^i + \mathbf{W}^i, \tag{5.2}$$

where $\mathbf{C}_K^i$, approximates the input correlation using K levels of decomposition taking into consideration the quantization effects and $\mathbf{W}^i$ is an expected truncation error matrix, which is a function of the correlation of the introduced errors due to truncation operations.

After obtaining individual approximation errors, the algorithm estimates the overall approximation error of all M input matrices by adding all individual errors together. In the next stage of the algorithm, inferior designs are discarded. An inferior design is defined as one that uses more hardware resources but produces worse approximation error in comparison to another design. At this point, the algorithm only stores the designs which are optimum

for all of the input distributions by exploiting any redundancy that exists in the precision requirement between the M correlation matrices. It is important to note that all input matrices share the same precision requirements for the data path of the computational block at each level of decomposition while having distinct coefficients corresponding to their own distribution. The last step of this stage calculates the remainder of the original matrices which will be used as the starting point for the next iteration.

The algorithm repeats until the termination condition is met. Examples of the termination condition are a specified approximation error, a given resource constraint or the number of decomposition levels to be used. The output of the proposed algorithm is a set of vectors of coefficients which best approximate the input M correlation matrices under a given objective function. Therefore, the proposed algorithm constructs a custom MVGRNG from computational blocks with different precision requirements targeting the minimization of approximation error and resource usage of each of the input correlation matrices, resulting in optimized designs for all target distributions. The expressions in (5.3) and (5.4) illustrate how a vector of three multivariate Gaussian samples is calculated for correlation matrices 1 and 2 from the set M.

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_1 = \begin{bmatrix} c_1^1 \\ c_2^1 \\ c_3^1 \end{bmatrix}_1 z_1^1 + \begin{bmatrix} c_1^2 \\ c_2^2 \\ c_3^2 \end{bmatrix}_1 z_2^1 + ... + \begin{bmatrix} c_1^K \\ c_2^K \\ c_3^K \end{bmatrix}_1 z_K^1. \tag{5.3}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}_2 = \begin{bmatrix} c_1^1 \\ c_2^1 \\ c_3^1 \end{bmatrix}_2 z_1^2 + \begin{bmatrix} c_1^2 \\ c_2^2 \\ c_3^2 \end{bmatrix}_2 z_2^2 + ... + \begin{bmatrix} c_1^K \\ c_2^K \\ c_3^K \end{bmatrix}_2 z_K^2, \tag{5.4}$$

where c_1^1 refers to the first coefficient in vector $\mathbf{c}^i$ in the first decomposition level. Note that different sets of univariate Gaussian random samples are used to drive each generator, in order to avoid introducing any unwanted correlation between the distributions of interest. This is denoted by $\mathbf{z}^i$, where i and the subscript of the square brackets denote the number of matrix from a set M.

In summary, a methodology to construct a hardware architecture for multiple multivariate Gaussian distributions under a shared generator has been described. The proposed algorithm targets any redundancy between the input distributions in order to select the most effective coefficients and the corresponding precisions for each computational block under a given objective function to minimize the approximation error across all target distributions. It is important to note that no correlation between the input correlation matrices is introduced.

5.3 FPGA Hardware Architecture

In this section, the general hardware architecture for a multiple multivariate Gaussian random number generator is described. Fixed point number representation is deployed throughout the entire design as it produces designs which

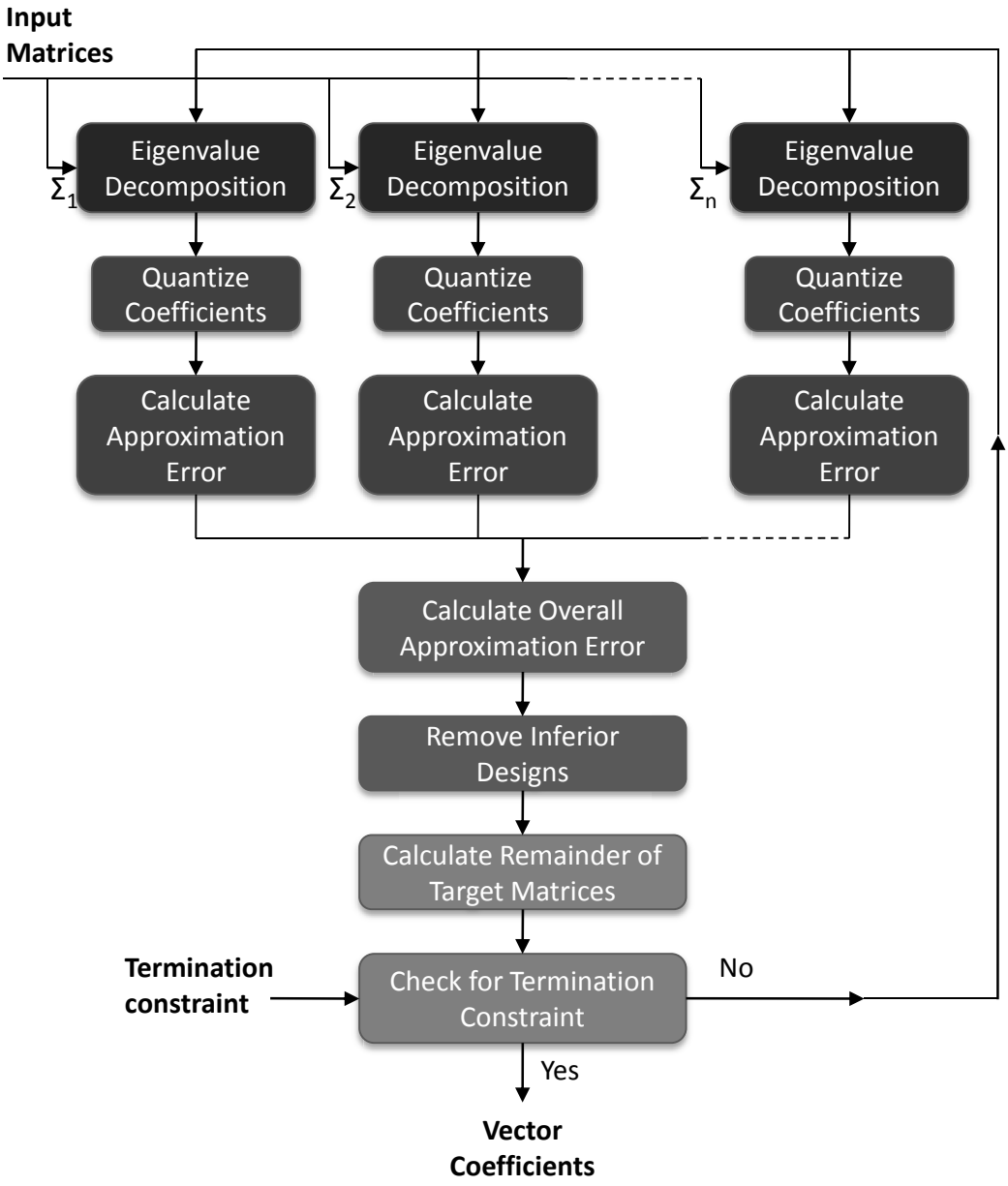


Figure 5.1: Proposed Algorithm.

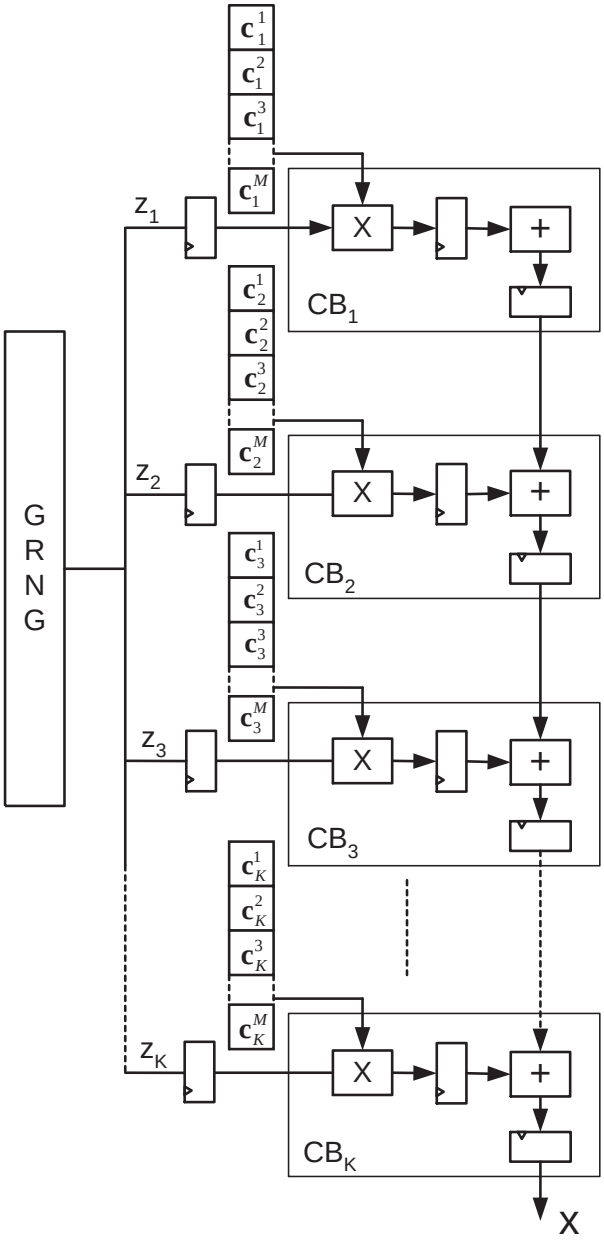


Figure 5.2: Hardware Architecture.

consume fewer resources and operate at a higher frequency in comparison to designs that use floating point arithmetic. The proposed hardware architecture is shown in Figure 5.2. The generator consists of K building blocks called computational block (CB) where K denotes the number of decomposition levels used in the approximation of the correlation matrices under consideration. Each block contains an adder and a multiplier with its own precision requirements for its data path. The architecture is mapped to logic elements only, so that the precision of the data path can be finely tuned. The architecture can be constructed using any number of CBs accommodating any given resource constraints. The precision in the data path of each CB is optimized for all vectors of coefficients $\mathbf{c}_j^i$, where j denotes the decomposition level and i denotes the number of correlation matrix. The precision in the adder path is fixed to the maximum of the precisions of all CBs. independent and identically distributed (i.i.d) univariate Gaussian samples z are produced from the GRNG block, hence any unwanted correlation between the resulting vectors is avoided.

In order for a high throughput to be achieved, the multiply-add operation is pipelined and all the computational blocks operate in parallel. The elements of each multivariate Gaussian random vector are serially generated so that one complete vector is produced per $\sum_{i=1}^{M} N_i$ clock cycles, where N_i denotes the dimensionality of the i^{th} multivariate Gaussian distributions and M is the number of distributions of interest.

5.4 Experimental Setup

In order for a fair comparison to be achieved between the proposed approach and the existing approaches, the throughput of all the generators under consideration must be adjusted to be at the same level. In this work the throughput of all generators under investigation should match the value of the proposed approach which is one complete vector per $\sum_{i=1}^{M} N_i$ clock cycles.

5.4.1 Extension of the Existing Approaches

The architecture of the approach in [TL08] does not require modification to match the desired throughput as no optimization across all input correlation matrices can be performed due to the fixed precision in the data path of the DSP blocks. Thus, a single generator consisting of N DSP blocks will be used for the generation of samples from M multivariate Gaussian distributions, where N denotes the matrix order. It is important to note that no effective resource sharing is achieved as no optimization is performed on the design of the MVGRNG across all distributions.

Since the algorithm in Chapter 4 only optimizes for one correlation matrix at a time, M generators are required to sample from M distributions. However, all generators constructed using the algorithm in Chapter 4 [SBCar] will have a throughput of $\frac{1}{N}$. Hence, a modification is carried out in order to match the desired throughput by reducing the hardware resources. This is reflected in the selection of the computational blocks where the vector of coefficients of M

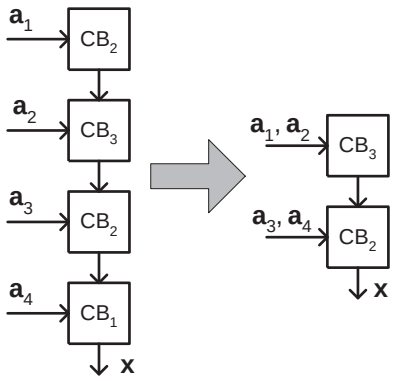


Figure 5.3: Extension of Algorithm in Chapter 4 [SBCar].

consecutive decomposition levels are forced to share the same computational block. However, the optimization of each CB takes into account M consecutive decomposition levels. Figure 5.3 illustrates an example of such modification where a 2x2 matrix $\mathbf{A}$ is considered. The first two consecutive decomposition levels in the approximation of $\mathbf{A}$ are shared by using one CB, in which the optimization is performed within the matrix between the first two levels. The same procedure applies to the next two consecutive decomposition levels. Note that the same number of GRNG as the proposed approach in this chapter is used in this modification to the algorithm in Chapter 4. In summary, all of the generators under consideration have been modified in order to have the same throughput using resource sharing techniques.

5.5 Performance Evaluation

In this section, the performance of the proposed approach is assessed in comparison to the existing approaches. The designs generated from the proposed

methodology, as well as the extension of the approach in [SBCar], have been implemented on a Stratix III EP3SE50F484C2 FPGA from Altera and Quartus II was utilized as a hardware synthesis tool. The word-length of the univariate Gaussian samples $\mathbf{z}$ is fixed to 18 bits precision throughout the entire design, where 3 bits are allocated for the integer part and 14 bits for the fractional part with one bit allocated as the sign bit.

In this experiment, four sets of input correlation matrices are considered. Each of the first three sets contains four matrices with identical matrix order which are 2x2, 4x4 and 6x6 matrices respectively. The last set of input correlation matrices is a mixture of matrices with different orders, namely two 2x2 and two 4x4 matrices. The empirical approximation error in the sample correlation matrix using 100,000 vectors of multivariate Gaussian random samples is obtained for each of the four sets.

5.5.1 Analysis of Approximation Error and Resource Estimation Models

This section focuses on the comparison between the estimated approximation error of each of the generated MVGRNG from the proposed approach in this chapter with the empirical approximation error from the sample correlation matrix. This will enable us to assess how well the error estimation model in the proposed algorithm performs. Figure 5.4 shows a plot of estimate versus empirical approximation error of the MVGRNGs designed using the proposed

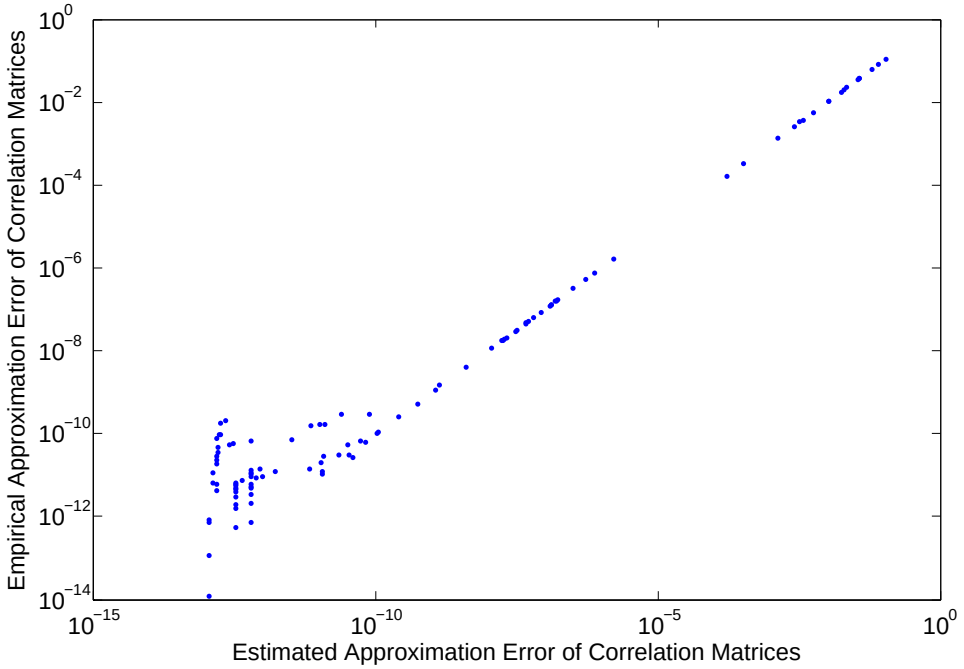


Figure 5.4: Estimate vs Empirical Approximation Error.

approach. It can be clearly seen that the plot is a straight line with a gradient of 1. The more scattered points in the lower left hand corner are considered insignificant as these errors are in the range of 10^{-10} to 10^{-15}. Thus, the error calculation in the proposed approach is accurately modeled as it provides almost a perfect match between estimate and empirical approximation errors. This signifies that the error model used to drive the optimization in the proposed approach is valid.

Since the MVGRNGs are constructed from a combination of CBs, the proposed approach estimates the total resource usage of the design based on the pre-calculated usage of each CB. The plot in Figure 5.5 shows that the estimate resource usage increases linearly with the empirical resource usage. The plot is fitted with a straight line given in (5.5).

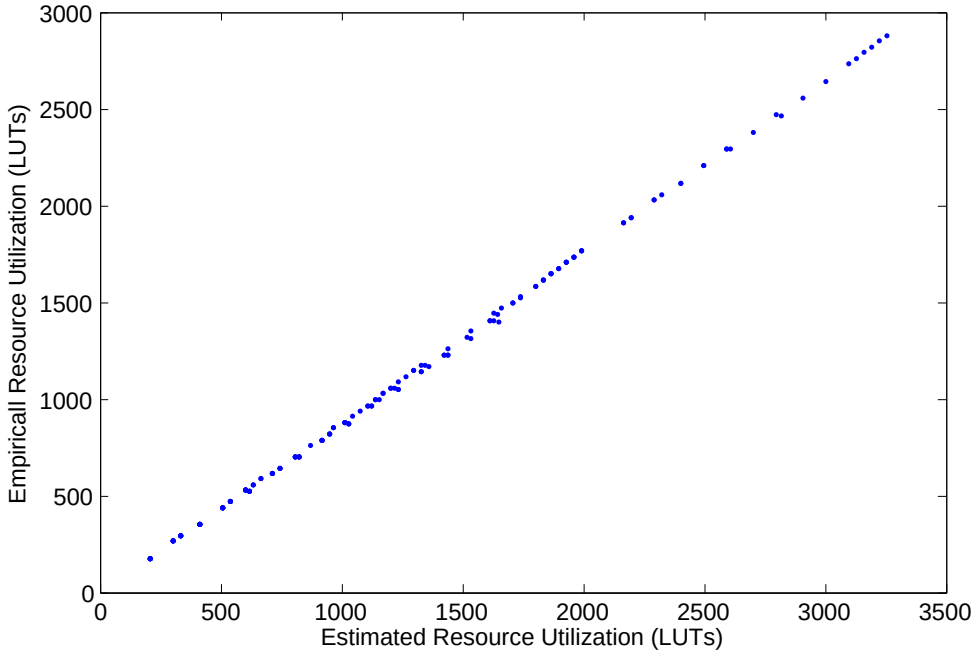


Figure 5.5: Estimate vs Empirical Resource Utilization.

$$y = 1.13x + 16, \tag{5.5}$$

where x and y denote the empirical and estimate resource usage respectively. Hence, a more accurate estimation of resource usage is fitted back to the proposed algorithm using (5.5). The reason for this mismatch is due to the extra optimization steps performed during the technology mapping stage which are not modeled in the proposed approach. However, a method to provide feedback to the proposed algorithm on these optimization is provided.

Figure 5.6 illustrates a histogram of the maximum operating frequencies of the MVGRNGs produced by the proposed approach. The maximum operating frequencies range from 395 MHz to 512 MHz with a mean of 457 MHz.

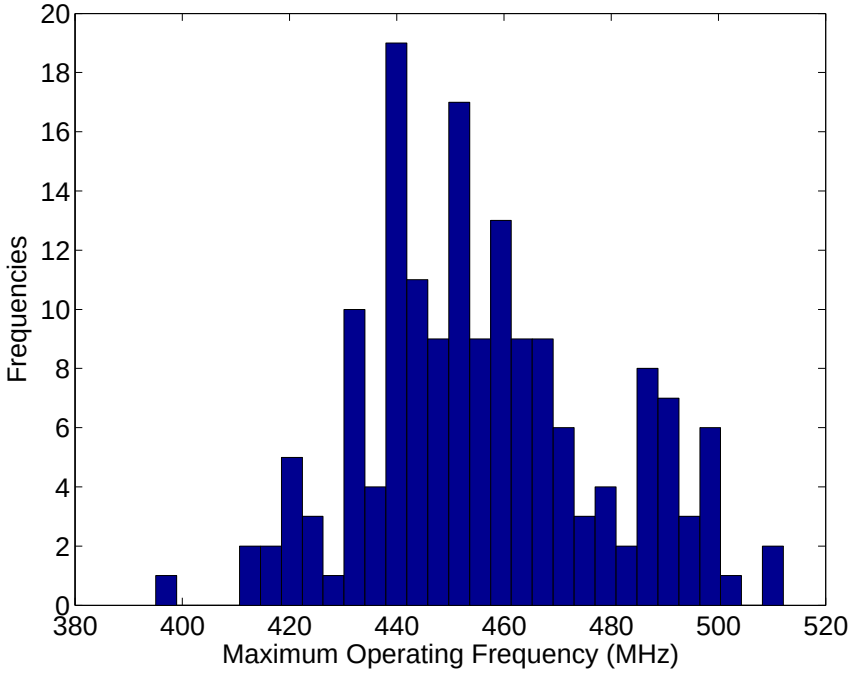


Figure 5.6: Maximum Operating Frequency.

5.5.2 Analysis of Empirical Data

In this section, the performance of the MVGRNGs designed from the three approaches under investigation, which are the proposed approach, the extension of the approach in Chapter 4 and the approach in [TL08], are examined. The graphs in Figures 5.7, 5.8 and 5.9 illustrate the empirical approximation error of the target correlation matrices using the first three sets of input correlation matrices. These are matrices with orders of 2, 4 and 6 respectively. Each point in the plots represents a design produced from a corresponding approach using a certain number of LUTs. It is important to note that the proposed approach and the approach in Chapter 4 have the capability to produce designs across a range of resource constraints for a given set of correlation matrices while only one fixed design is possible using [TL08].

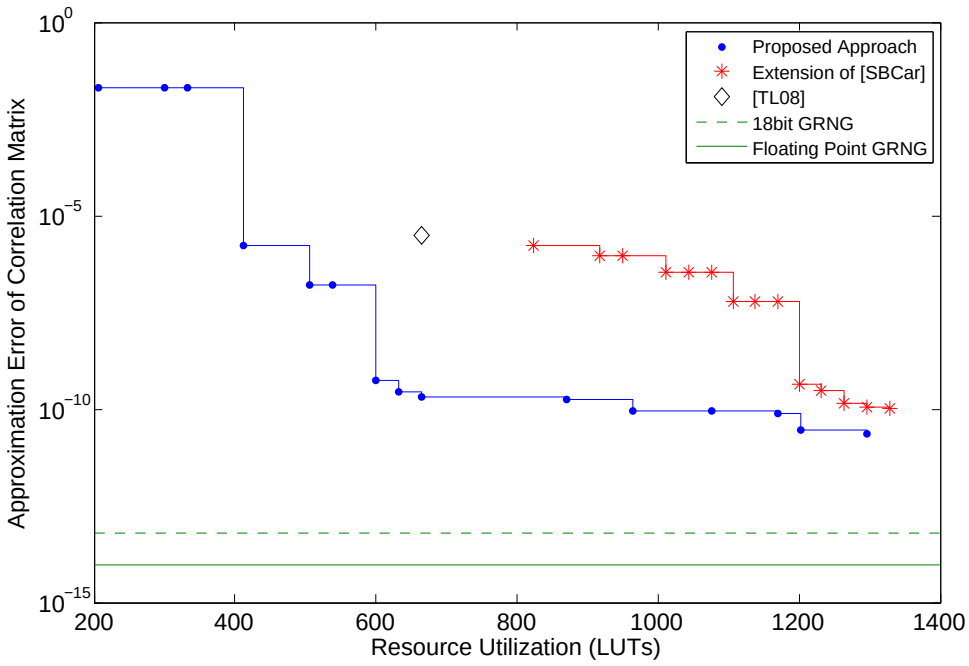


Figure 5.7: Comparison of Four 2x2 Correlation Matrices.

The largest reduction in resource usage is observed in Figure 5.9 where for the same approximation error of approximately 1.2×10^{-7} the proposed approach provides a 50% improvement on resource usage over the extension of the approach in Chapter 4. The same comparison is made between the proposed approach and [TL08] where it was found that the proposed approach offers up to 38% improvement on resource usage.

The largest reduction in correlation matrix approximation error based on empirical data is observed in Figure 5.8. For similar resource utilization of approximately 1250 LUTs, the proposed approach offers 7 orders of magnitude of improvement on the approximation error over the extension of the approach in Chapter 4 while 5 orders of magnitude of improvement is achieved over [TL08].

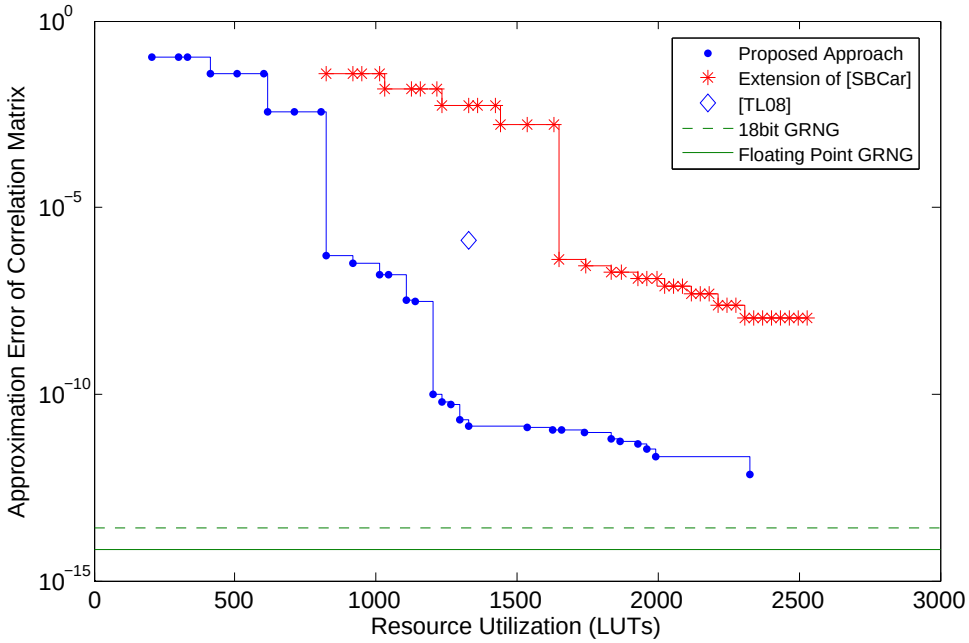


Figure 5.8: Comparison of Four 4x4 Correlation Matrices.

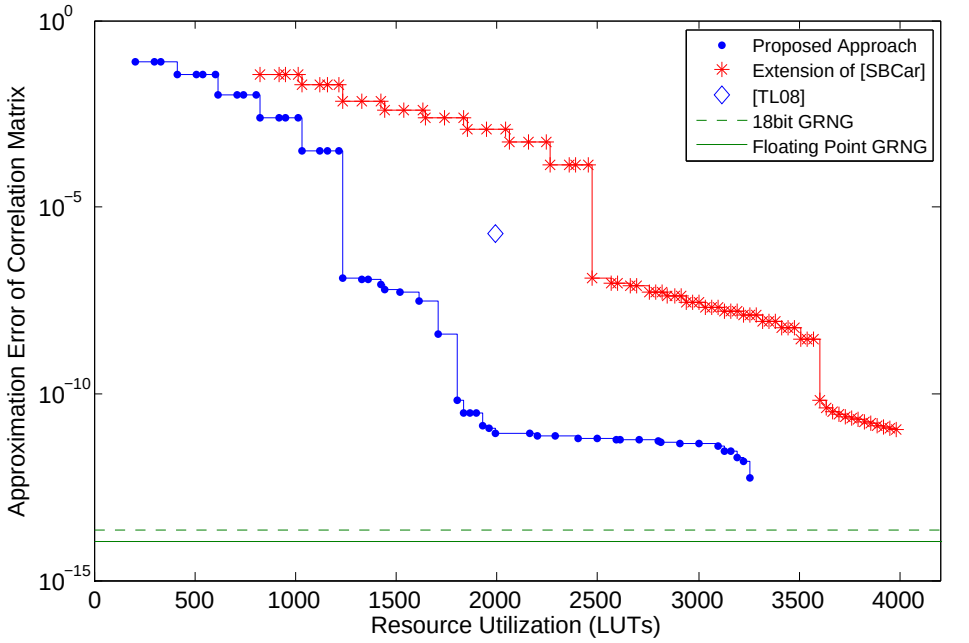


Figure 5.9: Comparison of Four 6x6 Correlation Matrices.

It can be concluded from the plots that for all matrix orders, the proposed approach provides the best MVGRNGs in comparison to [TL08] and the extension of the approach in Chapter 4 as these generators achieve lower approximation error while utilizing the same hardware resources and having the same throughput. This was achieved by taking into account precision similarities that exist between the input correlation matrices in a given set and effectively map them in the proposed algorithm. Thus, the overall performance of the MVGRNG is further improved in comparison to existing approaches.

A similar trend is observed for a set of input correlation matrices with different matrix orders. The results of this test are depicted in Figure 5.10. It can be seen that the proposed approach produces designs that achieve better approximation error for the same resource usage in comparison to existing approaches.

In order to assess the impact of the finite precision of univariate Gaussian samples used to drive the described generators, to their performance, two "virtual" designs are considered. These designs are described as "virtual", as only their empirical approximation error is considered. The first "virtual" design uses the same quantized univariate Gaussian samples as the mentioned designs, but their mapping to sample from multivariate Gaussian distributions is performed by employing the lower triangular matrix using Cholesky decomposition and double precision floating point arithmetic (MATLAB). The achieved approximation error is denoted by the dotted lines on the plots in Figures 5.7 ,5.8, 5.9 and 5.10. The second "virtual" design, is similar to the first one, but

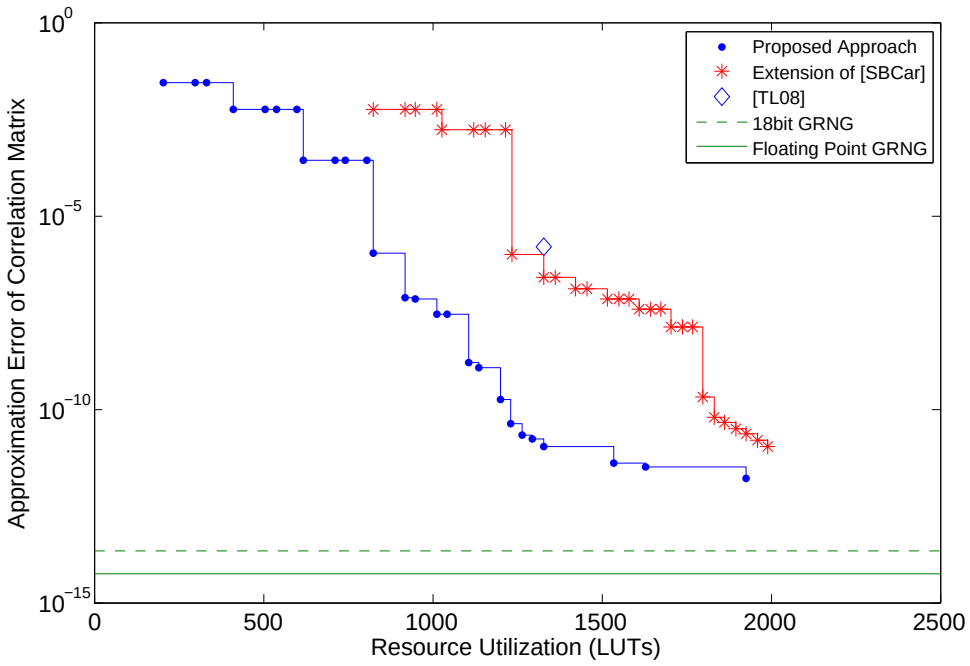


Figure 5.10: Comparison of Correlation Matrices with orders of 2 and 4.

no quantization is performed to the set of univariate Gaussian samples. This case models the "best" possible performance using the specific set of univariate samples, and it is denoted by the solid lines. The results demonstrate that the approximation error achieved by the generators from the proposed approach is very close to the virtual designs.

5.6 Summary

In this chapter, a novel approach is presented to design and implement a hardware architecture that effectively maps multiple multivariate Gaussian random number generators on an FPGA. To the best of the author's knowledge none of the existing work in the literature is aimed for a single multivariate Gaussian Random number generator which is specifically designed for all target distributions.

The proposed approach exploits redundancy that exists between the precision requirements of the target multivariate Gaussian distributions leading to designs which are optimized for all target distributions while minimizing the hardware resource utilization. The framework is based on the algorithm derived in Chapter 4 where word length optimization techniques are employed to enable multiple precisions in the design. Experimental results have demonstrated that the proposed approach outperforms the existing approaches achieving up to 50% reduction in hardware resource utilization without any compromise in the approximation error of the target correlation matrices and the achieved throughput performance.

Chapter 6

Design of a Financial Application Driven Multivariate Gaussian Random Number Generator for an FPGA

6.1 Introduction

The algorithms proposed in Chapters 3, 4 and 5 for the hardware implementation of multivariate Gaussian random number generators have one common theme in that the optimization is performed over the approximation of the input matrices. This is also the case for the work proposed in [TL08].

To the best of the author's knowledge, no published work in the literature has addressed the question regarding the relative performance of the various

optimization objective functions in the design of a random number generator block focusing on the impact that this has on the performance of a financial application. The authors in [TL08] carried out an evaluation of the performance of their approach for a financial application, the Delta-Gamma asset simulator, but the optimization of the hardware architecture of a MVGRNG over different objective functions has not been considered.

In this chapter, the investigation on the impact of the design decisions taken for the optimization of the MVGRNG to the performance of a financial application is performed. The applications under investigation are the estimation of the Value-at-Risk (VaR) of a financial portfolio and option pricing, two of the most widely used applications in the financial industry. The work presented in this chapter is based on the framework proposed in Chapter 4, as it can accommodate any objective function for the hardware optimization of a MVGRNG design. Three design criteria are proposed for the design of a MVGRNG targeting an FPGA device, and their impact to the estimation of VaR calculation and option pricing are investigated for a set of hardware resources.

The organization of this chapter is as follows. The main algorithm to construct a multivariate Gaussian random number generator is discussed in Section 6.2. Moreover, a novel framework which has the ability to accommodate any objective functions for the optimization of the MVGRNG design is presented. Section 6.3 introduces the two financial applications which are used as the case studies in this work. The overall framework is shown in Section

6.3.4 in order to visualize the scope of this work while Section 6.4 describes the set up of the experiments carried out in this work. The results and the relevant analysis are presented in Section 6.5. Finally, Section 6.6 concludes the chapter.

6.2 Constructing Hardware Architecture

In this section, the methodology of mapping the hardware architecture to an FPGA is discussed. The hardware architecture of the MVGRNG to be used in this chapter is based on the architecture in Chapter 4 where the MVGNRG is constructed from a combination of computational blocks with custom precisions. The main difference in the design methodology between this chapter and Chapter 4 is that the algorithm in this chapter is aimed towards the optimization of the error metric based on the application to be driven by the random numbers generated whereas the optimization in the approach in Chapter 4 is based solely on the approximation error of the input correlation matrix.

The high level overview of the proposed methodology is depicted in Figure 6.1. There are three main stages in the proposed methodology where, in the first stage, the correlation matrix $\mathbf{\Sigma}$ is decomposed into a vector $\mathbf{c}$ and its transpose using the Eigenvalue decomposition algorithm, see section 4.2. The resulting vectors are converted into fixed point representation using one of the user-specified word-lengths from the hardware library. In the second stage, the appropriate coefficients are selected in order to minimize the error metric

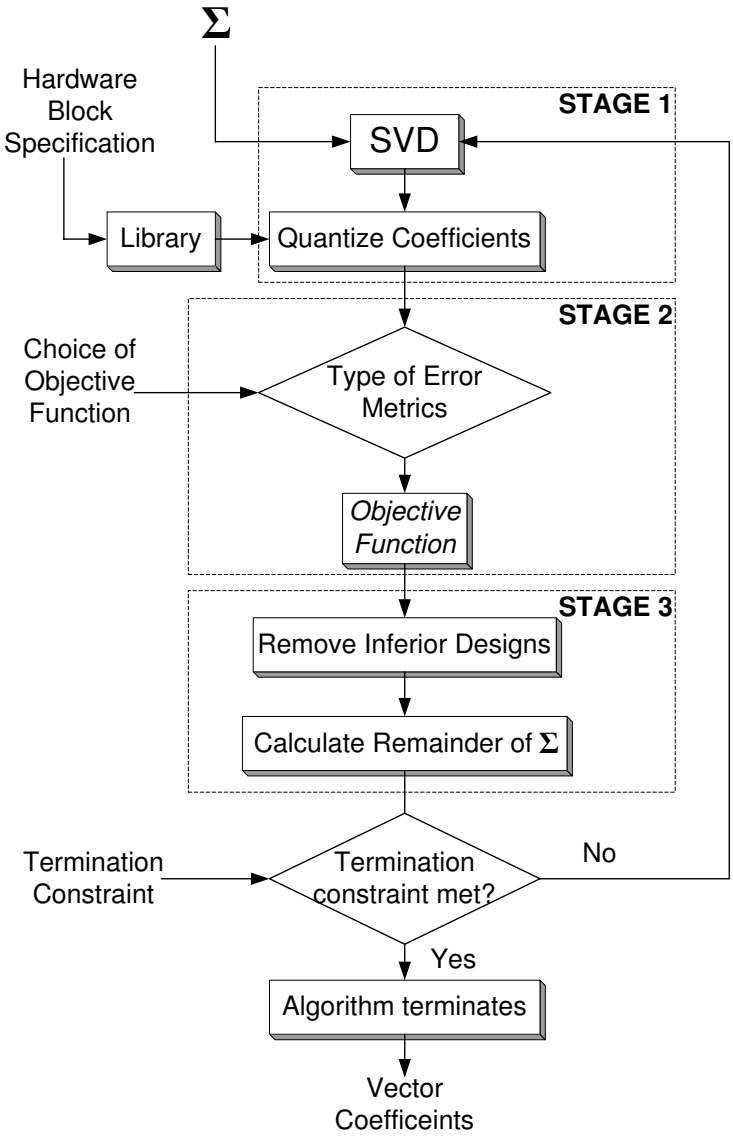


Figure 6.1: Proposed Methodology.

according to the selected objective function which will be described in the next section. The third stage of the proposed approach removes any inferior designs. The last step of this stage calculates the remainder of the original matrix $\mathbf{\Sigma}$ which is the starting point for the next iteration. These steps are repeated until the termination condition is met. The output of the proposed approach is a set of vectors of coefficients $\mathbf{c}_i$ which best approximate the input correlation matrix under a given norm.

The approximated correlation matrix $\overline{\mathbf{\Sigma}}$ can be decomposed as $\overline{\mathbf{\Sigma}} = \mathbf{C}_K + \mathbf{W}$. $\mathbf{C}_K$ approximates the input correlation matric using K levels of decomposition taking into consideration the quantization effects. The matrix $\mathbf{W}$ is an expected truncation error matrix, which is a function of the correlation between the error due to truncation operations.

In summary, a framework to design a hardware architecture of a MV-GRNG with the ability to accommodate different objective functions has been discussed. The coefficients of the correlation matrix of the distribution of interest are chosen based on the objective function of interest. The proposed approach provides an expectation of the generated correlation matrix given the coefficients in the various levels of the architecture taking into consideration all sources of errors injected into the system. It should be noted that the proposed system takes into account the correlation effect due to truncation error in the approximation of the correlation matrix. In the next section, the quality of the random samples produced based on the proposed methodology are used for the estimation of the Value-at-Risk of a financial portfolio and the

pricing of options so the impact of the different objective functions on the two financial applications can be investigated.

6.3 Case Study: MVGRNG in Financial Applications

This section focuses on the two financial applications of interest namely, the estimation of Value-at-Risk of a portfolio and the pricing of an option. The first part of this section provides a background description of the two applications. The discussion is then shifted towards correlated assets where a multivariate Gaussian random number generator is deployed to model the correlation between those underlying assets. Three different objective functions are introduced for the hardware design of the MVGRNG, given a certain constraint on the available resources, and their impact on the performance of the two financial applications is investigated.

6.3.1 Value-at-Risk of a Financial Portfolio

A portfolio is a collection of financial investments such as stocks, bonds and options. Essentially, the purpose of holding a portfolio is to limit the risk while increasing the possibility of making profit, where the investment is spread over a variety of assets with different degrees of risk factors. In order to quantify the expected loss of a portfolio with many correlated assets, the progression of the risk factors affecting this portfolio must be taken into account. Monte

Carlo simulation is used to simulate the time evolution of the risk factors due to their stochastic nature. Let us consider a portfolio containing N assets.

The market price of each asset is denoted by S_i having drift μ_i and volatility σ_i, where $i = 1 \ldots N$. The correlation between all of the assets in the portfolio under consideration is captured by a correlation matrix $\mathbf{\Sigma}$. The dynamics of the price of each asset are modeled as in (6.1).

$$\ln(S_i(t_E)) = \ln(S_i(t_0)) + \sum_{j=1}^{E}(\mu_i \delta t + x_j), \tag{6.1}$$

where x_j denotes a multivariate Gaussian random sample that follows $N(0, \mathbf{\Sigma})$. The time interval from t_0 to t_E is divided into intervals of length δt [Gla04].

The path taken by the random walk algorithm during the simulation is of interest only for *path-dependent* derivatives such as an Asian-style derivative where the price at maturity is the average price of the path taken over a specified period [Gla04]. A European-style derivative, on the other hand, does not take this into account and the price at maturity is taken at the end of the specified period [Gla04]. Value-at-Risk (VaR) is a measurement used to evaluate a risk of loss of a specific portfolio and describes probabilistically the market risk of a trading portfolio by measuring the worst expected loss over a specific time interval at a given level of confidence.

6.3.2 Option Pricing

An option is a contract between a buyer and seller which permits a buyer, depending the type of option held, the right to purchase or sell a particular asset at an agreed price on or before the option's expiration time. In this work, the option called "Chooser option" is considered.

The strike price, which is a price when the option is being exercised, of a Chooser option with, for instance, three underlying assets can be determined by the maximum of the sum of the two highest priced assets at the end of a specified period of time. Monte Carlo methods are used to calculate the value of these options where multiple underlying assets are involved. In particular, the MC simulation models the price of each of the asset over the lifetime of the option. Any sources of uncertainty are captured and taken into account towards the simulation by the use of drawing random samples from a multivariate Gaussian distribution where the correlation between the assets of interest can be modeled using the corresponding correlation matrix.

6.3.3 Objective Functions

In this work, three objective functions for the selection of coefficients and the precision of the computational paths for each decomposition level in the design of the MVGRNG core are proposed. The following notation is used for all three objective functions. $\mathbf{R}_{K-1}$ denotes the remainder of the original correlation matrix after $K-1$ levels of decomposition with $\mathbf{R}_0$ defined as the

Table 6.1: Objective Functions.

Error to be Minimized	Objective Functions
Maximum Absolute Error	$\max \left\lvert \mathbf{R}_{K-1} - (\mathbf{c}_K \mathbf{c}_K^T + \mathbf{W}) \right\rvert$
Maximum Relative Error	$\max \left\lvert \frac{\mathbf{R}_{K-1} - (\mathbf{c}_K \mathbf{c}_K^T + \mathbf{W})}{\mathbf{R}_{K-1}} \right\rvert$
Mean Square Error	$\frac{1}{N^2} \left\lVert \mathbf{R}_{K-1} - (\mathbf{c}_K \mathbf{c}_K^T + \mathbf{W}) \right\rVert^2$

original correlation matrix at the start of the algorithm. The max(.) operator is an element-wise operation for a matrix of interest. Table 6.1 illustrates the three objective functions to be used in this work. The first objective function selects the coefficients based on the minimization of the maximum absolute error in the approximation of the correlation matrix while the second and third objective functions minimize the relative and mean square error in the correlation matrix approximation respectively. All three objective functions take into account the correlation effect between the truncation errors to the final correlation matrix.

6.3.4 Framework

Figure 6.2 shows the framework deployed in this work. There are two main components in the system. The first part generates random samples from a given multivariate Gaussian distribution which captures the correlation between the assets of interest. These random numbers are produced by the proposed system where the generator is mapped onto an FPGA using mixed

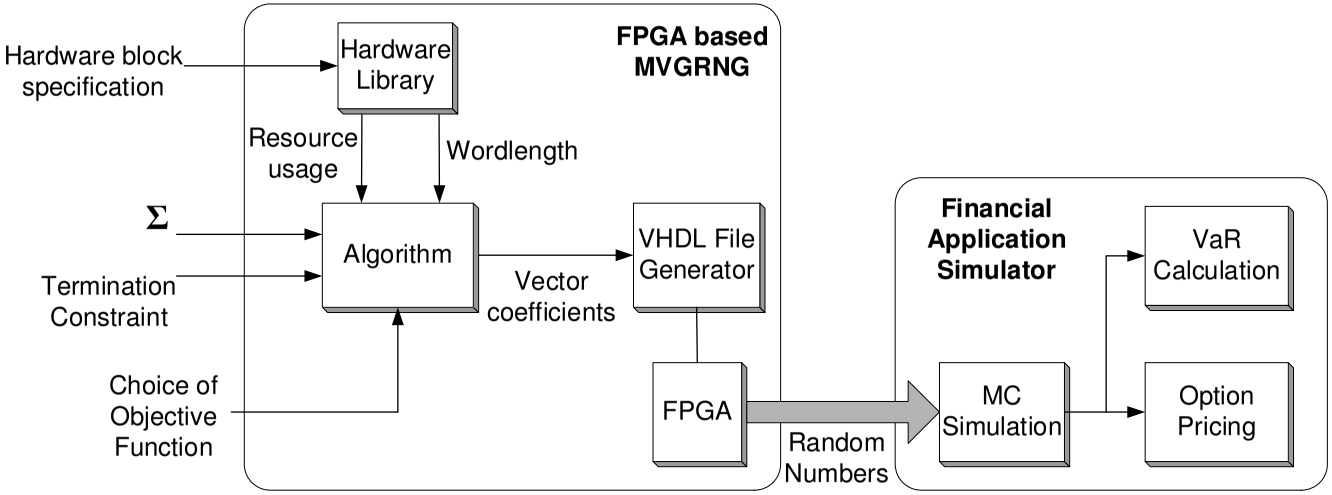


Figure 6.2: High Level Overview of Framework.

precision in the computational paths. The second part of the framework implements the two financial applications under consideration using a CPU which is fed by the random numbers generated by the proposed hardware block.

6.4 Experimental Setup

The purpose of this investigation is to assess the performance of the random number generator core when the already mentioned objective functions are used for its optimization, for a given resource constraint, using the calculation of VaR and option pricing application as testbenches. Since we are interested in the impact of the objective function for the design of the MVGRNG hardware block on the performance of the applications under investigation, it is adequate for this work for the two applications to be implemented on a CPU. The target device for the MVGRNG is a Stratix III EP3SE50F484C2 FPGA from Altera and Quartus II is utilized as a hardware synthesis tool.

The precision of the univariate Gaussian random samples z is kept constant throughout the design at 18 bits with 3 bits dedicated for the integer part and 14 bits dedicated for the fractional part. The set of computational blocks (CBs) that is used in the proposed framework is chosen in order to cover the range between 10 to 18 bits precision in an almost uniform way where three CBs with 10, 14 and 18 bits precision have been pre-defined in the hardware library. After the synthesis stage from Quartus II, the resource utilization of 212, 296 and 332 LUTs is reported for CBs using 10, 14 and 18 bits precision

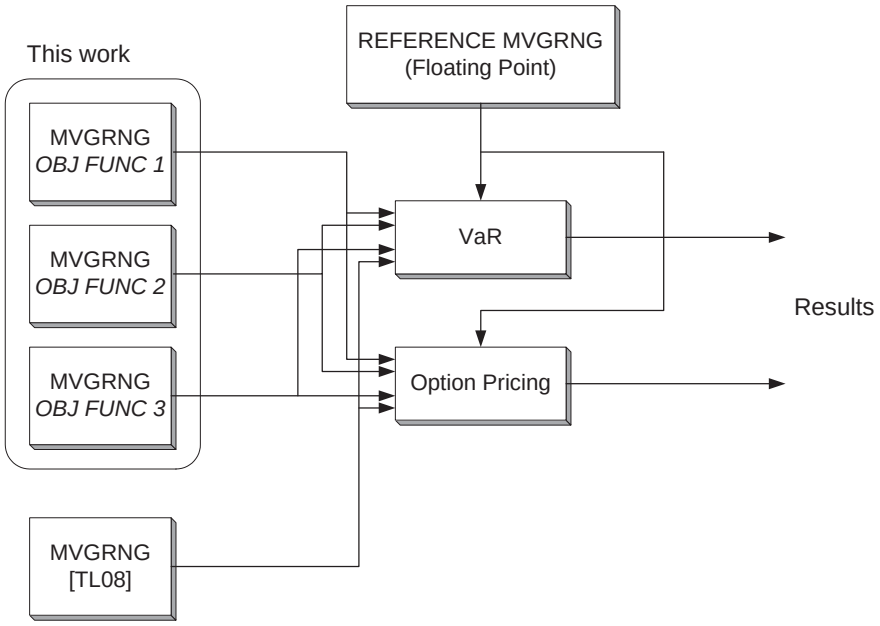


Figure 6.3: Experimental Setup Overview.

respectively. These results are used by the three objective functions under consideration in order to optimize for the desired architecture.

In this work two types of input correlation matrices are considered. Type I denotes matrices with high cross-correlation between the underlying assets whereas Type II denotes correlation matrices with very low cross correlation.

Figure 6.3 illustrates the overview of the experimental setup. In total, four types of MVGRNG hardware blocks are investigated and compared. The first three generators are implemented on FPGA using the proposed methodology optimizing for three different error metrics as seen in objective functions 1, 2 and 3 respectively, see Table 6.1. The fourth MVGRNG is generated using [TL08]. The generated random numbers are then used for the calculation of VaR of both European-style and Asian-style derivatives and produce the risk assessment for the portfolios of interest as well as driving the calculation of

option pricing. Each of the four generators is fed with univariate Gaussian random samples with the same seed in order to have the same sequence of random numbers for a fair comparison.

The results from the four generators are compared to a reference design implemented on general purpose processor using double precision floating point number representation for the generation of random numbers. Note that the original input correlation matrices are used for the reference design as no quantization is performed. As a consequence, the outcome of the two applications driven by the random numbers generated using the reference design are considered to be the actual results.

6.4.1 Experiment I: Calculation of VaR

In the first experiment, four portfolios each with five correlated assets are considered. The underlying correlations between the five assets for each portfolio are modeled by the four correlation matrices, namely **A**, **B**, **C** and **D**. **A** and **B** are of Type I while **C** and **D** are of Type II. Cross correlation values for Type I matrices are above 0.9 while that of Type II matrices are below 0.01.

A fixed resource constraint is used as a terminating condition for the three objective functions of interest where the constraint is set to 1660 LUTs since 1660 is equivalent to five DSP blocks which is the amount of hardware resources required by [TL08] for a 5x5 correlation matrix. Note that the approach in [TL08] is based solely on DSP blocks for producing multivariate Gaussian random numbers on an FPGA platform.

In this experiment, 100,000 vectors of multivariate Gaussian random samples are generated from each of the four generator and the reference design. These random numbers are employed to drive the Monte Carlo simulations which are carried out over a period of one year with a timestep of one day. At the completion of the simulations, the price at maturity of each portfolio is obtained which is the sum of all five assets.

6.4.2 Experiment II: Option Pricing

In this experiment two types of Chooser option with 3 assets are considered. The strike price of Chooser option 1 is defined as the sum of all of its assets while that of Chooser option 2 is defined as the maximum of the sum of the two highest priced assets at the end of a specified period of time as expressed in (6.2).

$$\begin{aligned} S1_{strike} &= S1_1 + S1_2 + S1_3 \\ S2_{strike} &= max\left\{(S2_1 + S2_2), (S2_1 + S2_3), (S2_2 + S2_3)\right\}, \end{aligned} \tag{6.2}$$

where $S1_i$ denotes the price of the i^{th} asset for Chooser option 1 while $S1_{strike}$ denotes the strike price of Chooser option 1. Similar notations are used for Chooser option 2.

Taking into account the two different correlation matrix types I and II we end up with four possible cases of Chooser options, Chooser option 1 and 2

both with high and low cross correlations between their assets. The same procedure as in Experiment I is taken in order to generate the MVGRNG hardware blocks with the exception of the termination condition being set to 996 LUTs for resource constraint which is equivalent to three DSP blocks, the amount of hardware required by [TL08].

6.5 Results: Evaluation of Hardware MVGRNG Blocks

In order to establish the overall picture of the two experiments a summary of the final results of both experiments are presented first in this section. Table 6.2 summarizes the best generator blocks, which produce the closest results for the generation of random numbers from a MVGRNG to the actual values, for each financial application.

Table 6.2: Summary of Best Performing Metrics for Financial Applications of Interest.

Experiment	Financial Applications	Best Metric	
		Type I	Type II
I	European-style VaR	MSE	REL
	Asian-style VaR	MSE	REL
II	Pricing of Chooser Option 1	MSE	REL
	Pricing of Chooser Option 2	MSE	[TL08]

For both applications where the underlying assets are highly correlated,

the best results are obtained by employing the objective function based on the mean square error. On the other hand, with low cross correlation, the objective function which is based on the relative error is the best metric for three out of four cases with an exception of the pricing of Chooser Option 2 where [TL08] provides the best results. More detailed discussion for each experiment will be given in the next section.

The reason for this apparent pattern is that the relative error tries to approximate the target correlation matrix based on the relative approximation errors for each element in the matrix. Therefore, the low values of cross correlation between the assets are taken into account during the approximation using the relative error metric which is not the case for other metrics.

In terms of the hardware performance, the operating frequency of all designs is in the range of 380 to 420 MHz, where the mean is 401.22MHz. The following sections present a more in-depth analysis of the results presented in 6.2, detailing exactly how the conclusions of the best performing metrics have been reached.

6.5.1 Experiment I: Calculation of VaR

In this experiment, the returns of each of the four portfolios are obtained for many MC simulation paths. In order to evaluate the results, the mean returns of each portfolio is analyzed. Figure 6.4 illustrates a plot of the difference in mean return of each portfolio with respect to the mean return of the reference design for a European-style VaR. The height of the bar represents the deviation

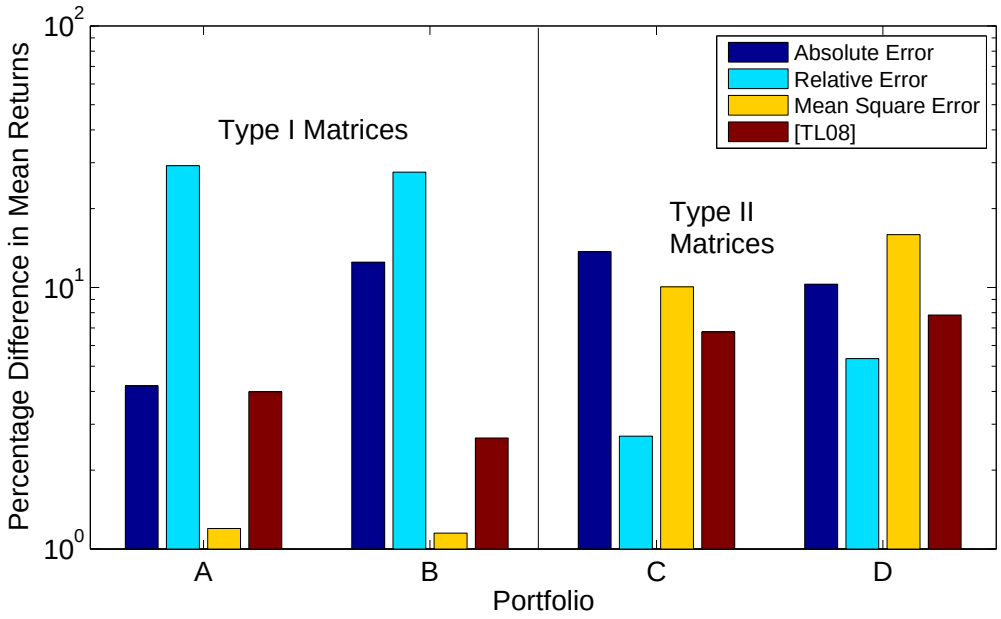


Figure 6.4: European-style VaR: Absolute Percentage Difference in the Mean Returns.

of the portfolio from the value of the mean return of the reference design.

It can be seen that the mean square error metric gives the nearest approximations to the actual values for Type I matrices while the relative error is the best metric for Type II matrices. In terms of the accuracy of the prediction of the mean returns, it can be deduced from the plots that an improvement in performance within the range of 5% to 96% is reported for Type I matrices while 24% to 80% improvement is observed for Type II matrices using the four different generators for the same hardware resource utilization for the MVGRNG block.

The upper and lower limits of the value of the returns of each portfolio given a 95% level of confidence are investigated in this part of the experiment. The upper limit is defined as the maximum expected return while the lower limit

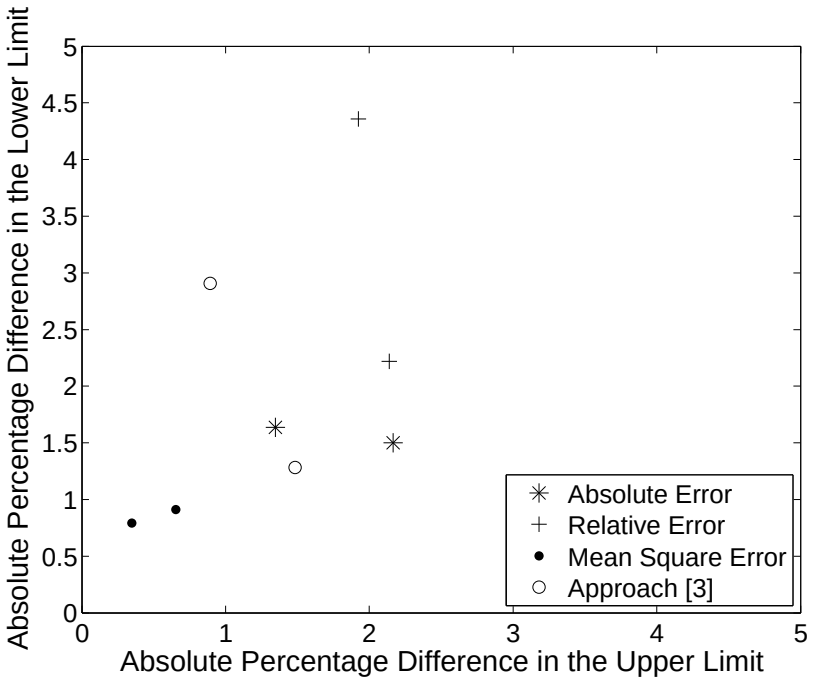


Figure 6.5: European-style VaR: Difference in the Upper and Lower Limits of the 95% Confidence Interval for Type I Matrices **A** and **B**.

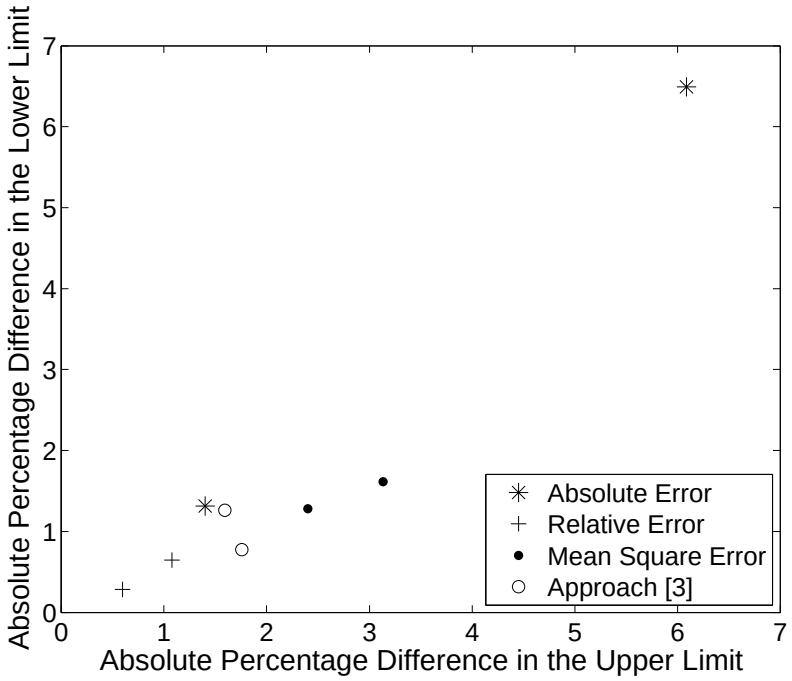


Figure 6.6: European-style VaR: Difference in the Upper and Lower Limits of the 95% Confidence Interval for Type II Matrices **C** and **D**.

is the minimum expected return of each portfolio over a specified confidence interval. The deviation of the upper and lower limits of the four generators from the reference design for Type I and Type II matrices are plotted in Figures 6.5 and 6.6.

It can be seen from Figures 6.5 that the generators utilizing the mean square error metric provide the least deviation from the reference design while the generators using the relative error metric provide the largest deviation for portfolios with highly correlated assets. On the other hand, the generators with relative error metric provide the nearest results to the reference design for portfolios with low cross correlation between the assets. The plots reinforce the point previously observed for the best metric for Type I and II matrices. Note that these deviations are arbitrary for all generators, therefore it cannot be predicted if the upper and lower limits have been overestimated or underestimated.

Similar conclusion can be drawn for the Asian-style Value-at-Risk as can be seen in Figures 6.7, 6.8 and 6.9. Figure 6.7 illustrates a plot of the difference in mean return of each portfolio with respect to the mean return of the reference architecture for an Asian-style VaR. As previously established for the European-style VaR, it can be seen that the mean square error metric provides the best approximations to the actual values for Type I matrices where the correlation between the assets are high. On the other hand, the relative error is the best metric for Type II matrices where the corresponding portfolios contain assets with low cross correlation.

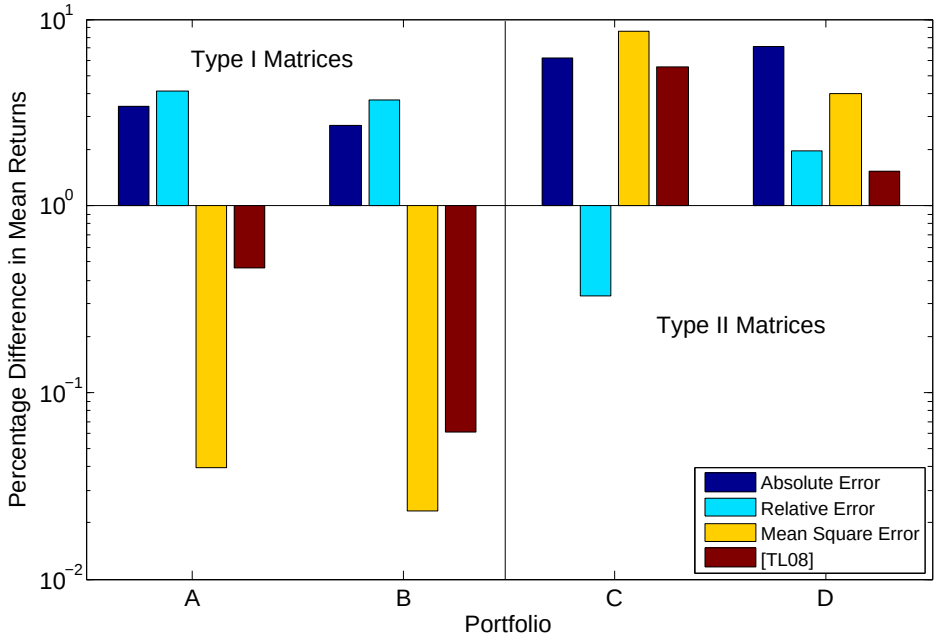


Figure 6.7: Asian-style VaR: Absolute Percentage Difference in the Mean Returns.

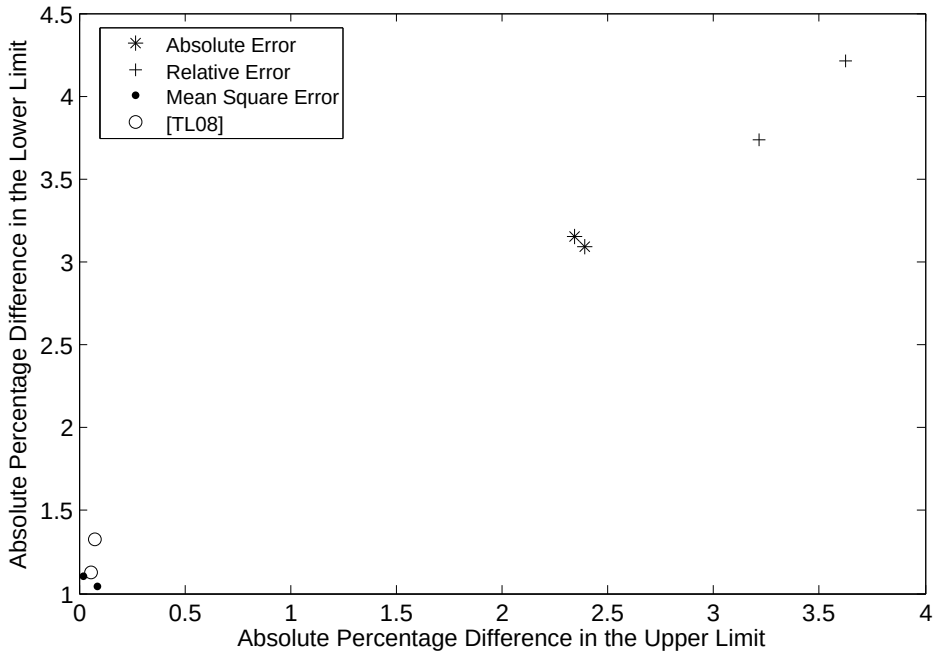


Figure 6.8: Asian-style VaR: Difference in the Upper and Lower Limits of the 95% Confidence Interval for Type I Matrices **A** and **B**.

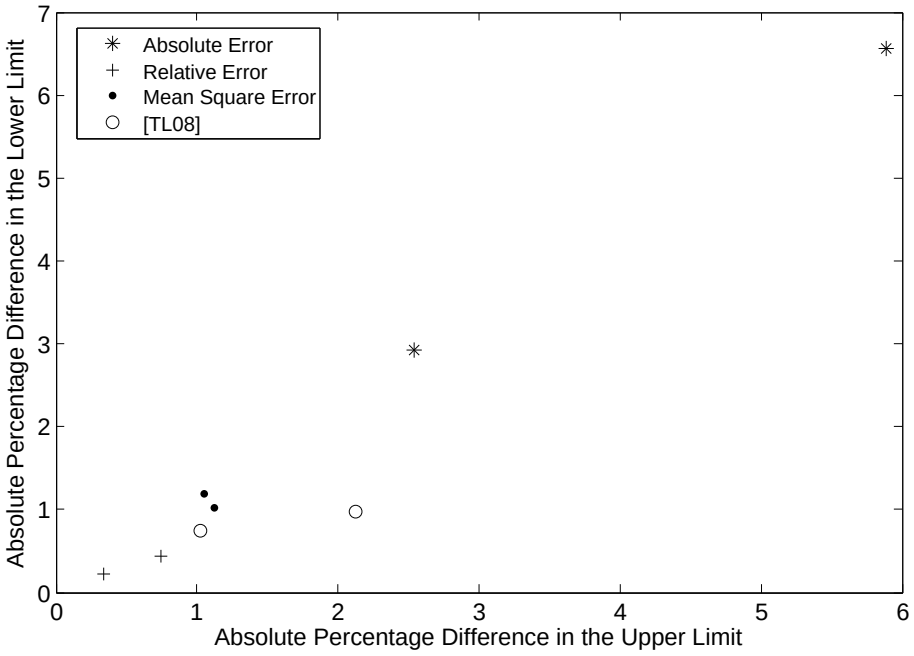


Figure 6.9: Asian-style VaR: Difference in the Upper and Lower Limits of the 95% Confidence Interval for Type II Matrices **C** and **D**.

6.5.2 Experiment II: Option Pricing

Figures 6.10 6.11 6.12 and 6.13 illustrate plots of the deviation of the strike price from the actual values for Chooser options 1 and 2 over a range of resources. All graphs show that the deviation from the actual strike price decreases as the resource utilization of the MVGRNG hardware block increases. Similar to the previous experiment, these deviations are arbitrary for all generators, therefore it cannot be predicted if the strike price has been overestimated or underestimated.

One important point from these plots is the ability for the proposed approach to produce designs across all design space whereas [TL08] will only offer one design for a given correlation matrix. The plots indicate that for Type I

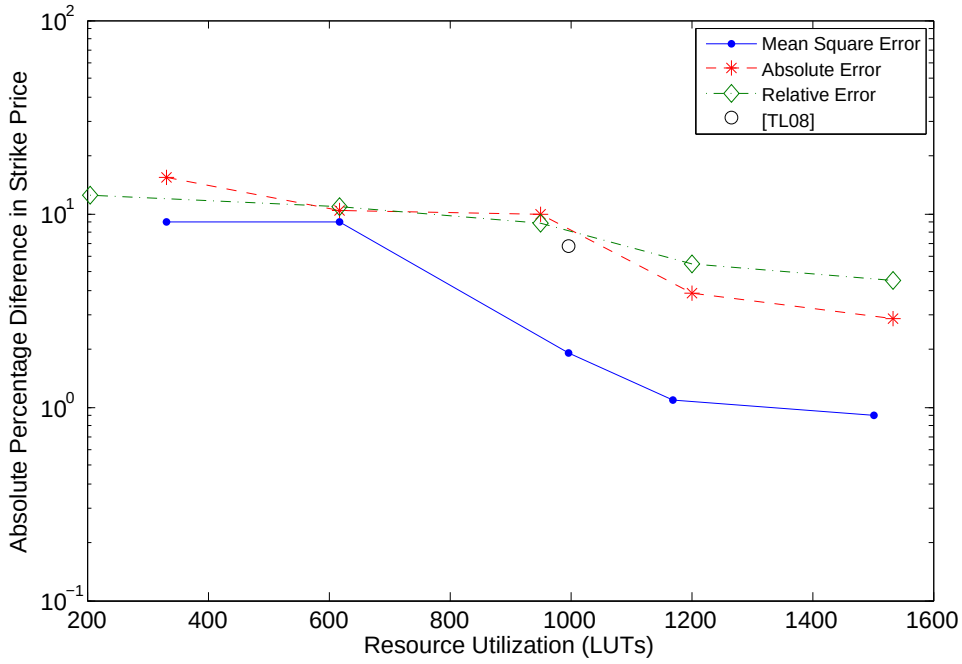


Figure 6.10: Difference in Strike Price for Chooser Option 1 with **High** Correlated Assets.

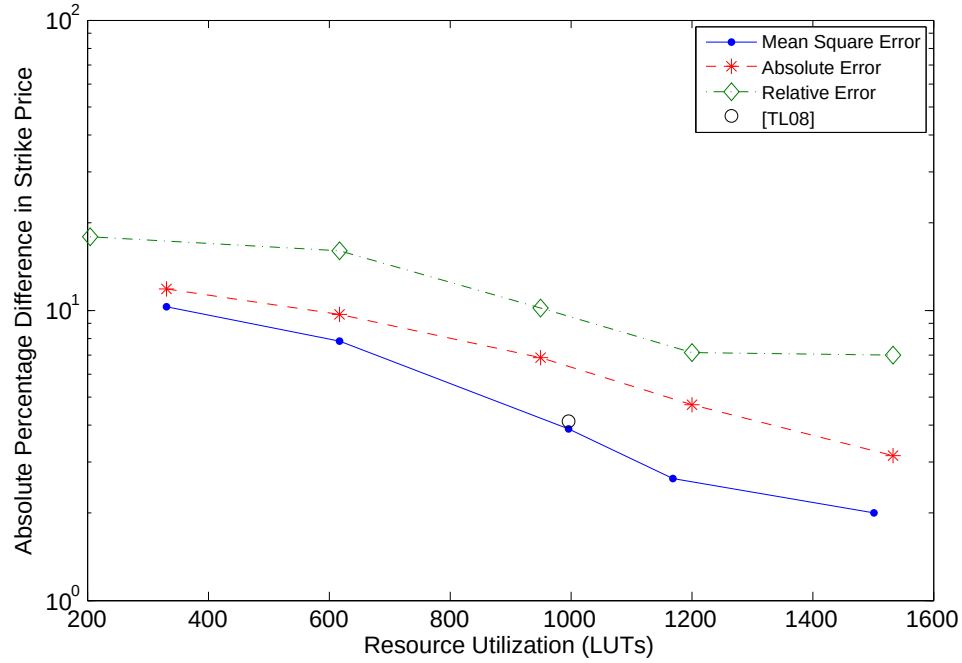


Figure 6.11: Difference in Strike Price for Chooser Option 2 with **High** Correlated Assets.

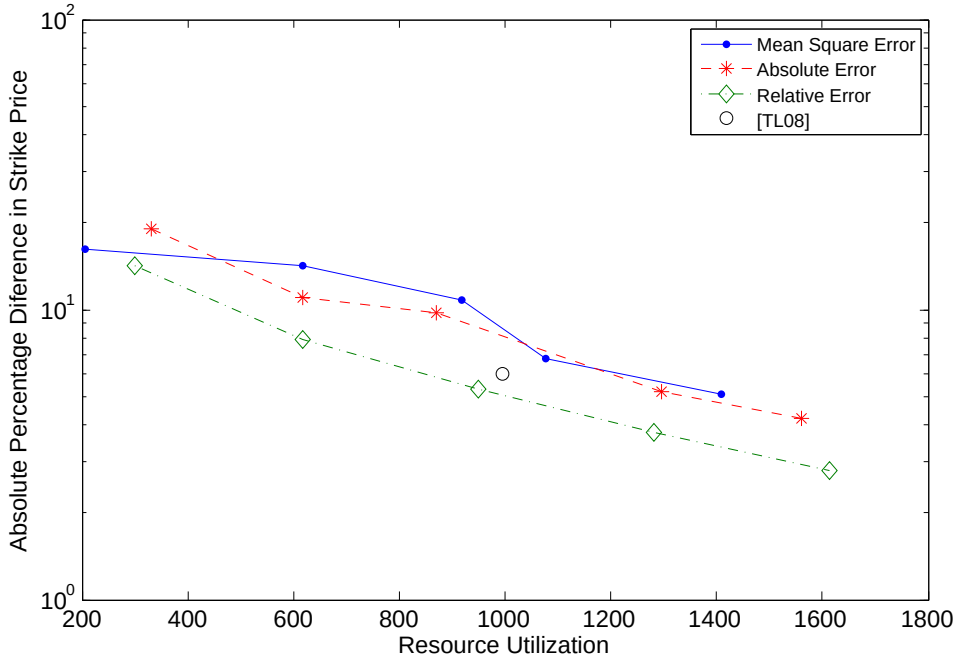


Figure 6.12: Difference in Strike Price for Chooser Option 1 with **Low** Correlated Assets.

matrix, an improvement in performance within the range of 17% to 81% and 11% to 71% is reported for Chooser options 1 and 2 respectively using the four different generators. For Type II matrix, an improvement in performance within the range of 12% to 51% is reported for Chooser option 1 and 8% to 39% for Chooser option 2 using the three four objective functions.

In terms of the best performing matric for Chooser options with high correlated assets, the plots in Figures 6.10 and 6.11 demonstrate that the mean square error metric (blue line) provides the nearest estimation to the reference design as the least deviation is observed in the percentage difference in strike price.

The plots from Figure 6.12 show that the best metric for Chooser option 1 with low correlated assets is the relative error metric as it produces the least

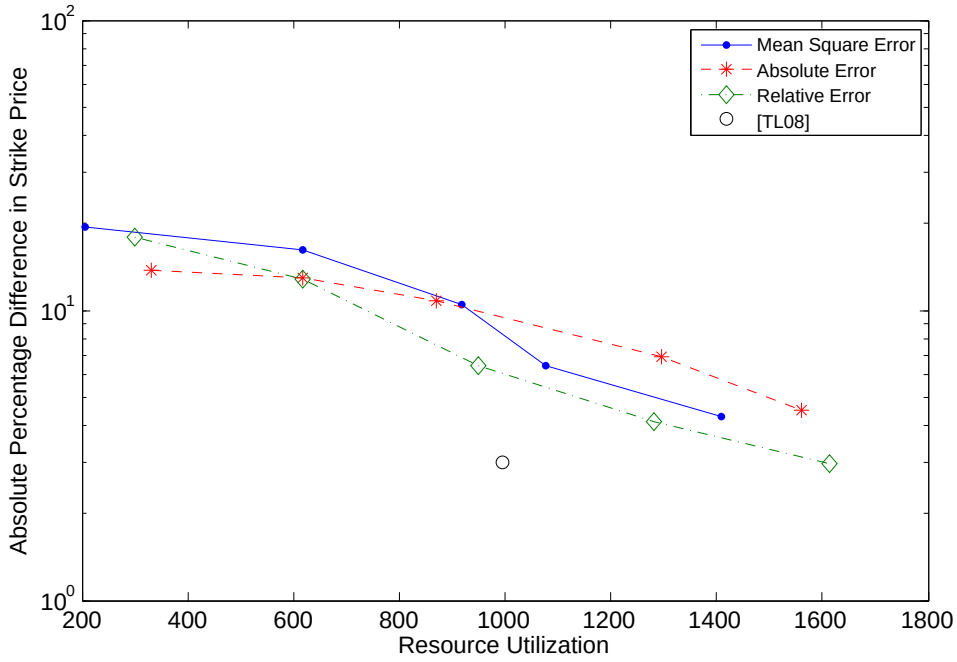


Figure 6.13: Difference in Strike Price for Chooser Option 2 with **Low** Correlated Assets.

deviation in strike price from the reference design. Finally, the plots in Figure 6.13 demonstrate that the generator from the approach in [TL08] provides the nearest result to the reference design.

6.6 Summary

This chapter investigates the impact of the design decisions taken in the production of a MVGRNG targeting two financial applications using the methodology presented in Chapter 4. Three design criteria for the optimization of the MVGRNG are proposed and their impact on the estimation of VaR financial problem and option pricing are investigated.

Simulation results have shown that, based on the two case studies, for an application with highly correlated assets, the objective function which optimizes for the mean square error in the approximation of the target correlation matrix of the multivariate Gaussian distribution provides the best performance for the same hardware resource utilization.

On the other hand, for an application with low cross correlation between its assets the objective function based on the relative error provides the best results. The only exception being in the experiment for Chooser Option calculation for low correlated assets where the approach in [TL08] provides the best performance. In terms of the accuracy of the prediction of the mean returns, an improvement in performance of up to 96% is reported for VaR calculation while up to 81% improvement is observed for option pricing using the four different objective functions.

Chapter 7

Summary and Conclusions

7.1 Summary of Thesis Achievements

The generation of random numbers from multivariate Gaussian distribution using hardware circuits targeting an FPGA is a relatively recent approach. It has clearly been demonstrated in the literature that the hardware implementation has one crucial advantage over the software counterpart using general purpose processor which is the speed at which these random numbers are produced. The reason for this to be achievable is due to the fact that the FPGA devices provide fine-grain parallelism and reconfigurability property for the implementation of digital circuits.

In comparison to another hardware approach in the form of ASICs implementation, the FPGA implementation offers the possibility of mapping different distributions for the generation of random numbers at run-time. This is not possible with ASIC technology as the design is fixed at manufacturing

stage and thus cannot be modified.

The hardware resource utilization on an FPGA is one of many important design decisions when optimizing for a desired hardware architecture. This is due to the fact that a random number generator is usually a block in a larger application, such as Monte Carlo simulation. Hence, it is essential that the area requirement of a random number generator is kept as low as possible.

As far as the existing approaches for the implementation of a multivariate Gaussian random number generator on an FPGA are concerned, none of the approaches in the literature currently explores the hardware resource usage design space. In other words, only one fixed hardware architecture is available for a given multivariate Gaussian probability distribution.

The discussion of the work presented in this thesis began with a novel approach to draw random samples from multivariate Gaussian distribution using a digital circuit implemented on an FPGA device as proposed in Chapter 3. The main feature for the algorithm presented in Chapter 3 is that it offers the flexibility to produce MVGRNGs for any given hardware resource constraints for a target multivariate Gaussian distribution. This feature is achieved by approximating the target correlation matrix which characterises the distribution of interest. The matrix approximation leads to the ability to decompose the computational structure of the required hardware design. Hence, the algorithm is the first innovative method to fully explore the resource usage design space. As well as offering the flexibility to produce designs across the design space, the hardware resource requirement of the MVGRNG is reduced in comparison

to other approaches in the literature.

The hardware architecture associated with the approach proposed in Chapter 3 comprises of DSP blocks, on chip block RAMs and an adder tree structure. The use of DSP block to map the functionality of the multiply accumulator means that the precision requirement of the input matrix coefficients is fixed to the precision of the DSP block. However, experimental results in Chapter 3 have demonstrated that lower precision can be allocated to the computational path depending on the structure of the input correlation matrix in order to achieve a more optimized design. This fixed precision issue has been addressed in Chapter 4 in which a novel approach has been proposed to implement a MV-GRNG that can be constructed from blocks of predefined hardware using any given precision requirement. This has been made possible due to the use of LUT based multiplier in the hardware architecture as opposed to the DSP block used in the approach in Chapter 3 and the existing method in the literature [TL08]. The other main difference is that the approach in Chapter 4 approximates the input correlation matrix instead of the lower triangular matrix. These two main features enable the hardware resource requirement of the designs produced using the proposed algorithm to be further minimized.

Due to the nature of the proposed algorithms in this thesis, an error is inserted into the system when the input correlation matrix is approximated. An in-depth error analysis has been performed in Chapter 4 taking into account all possible sources of approximation errors. The error model has been integrated into the proposed algorithm and experimental results have shown that

the MVGRNGs produced from the approach in Chapter 4 generates random numbers with higher accuracy than MVGRNGs from Chapter 3 and from the approach in [TL08] as well as producing designs which require fewer resource usage.

In Chapter 5, the optimization in Chapter 4 has been further extended by considering the implementation of a single generator which is optimized for many multivariate Gaussian distributions. The novel approach proposed in Chapter 5 makes use of any similarities that exist in the precision requirements of the different input multivariate Gaussian distributions. As a consequence, an innovative resource sharing algorithm has been proposed to produce a single MVGRNG best suited for all distributions of interest. This reduces the hardware resource requirement in comparison to individually implementing the MVGRNG.

Chapter 6 of this thesis focuses on the application domain of random number generation. An exploration into the impact of design decisions to be taken for the construction of an MVGRNG with regards to the performance of two of the most widely used financial applications namely, the estimation of Value-at-Risk and the calculation of option pricing. The MVRGNGs generated from the proposed approach are used to produce random numbers to drive the Monte Carlo simulations for each of the two applications. Three design criteria have been proposed in the investigation and experiments have been conducted in order to establish the best design criteria for the two applications of interest.

7.2 Conclusions

In conclusion, many techniques are available in the literature to design a hardware architecture for the implementation of a multivariate Gaussian random number generator on an FPGA. The work presented in this thesis proposes innovative approaches in order to implement such a block exploring a wide range of techniques which are used to optimize the generator design targeting the minimization of hardware resource requirement as well as the error in the approximation of the input correlation matrix.

More specific conclusions from individual chapters in this thesis are given below.

- Chapter 3 provides the fundamental platform of the entire thesis by introducing an algorithm to approximate the target correlation matrix. This approximation enables the exploration of the design space by having many different hardware designs for a given target correlation matrix. The experimental section reveals that the precision requirement in the computational path of the MVGRNG design depends on the structure of the target correlation matrix.

- The approach proposed in Chapter 4 approximates the target correlation matrix itself instead of the lower triangular matrix. Hence, the algorithm is not restricted to only matrices with lower triangular structure. In addition, the proposed algorithm estimates the appropriate precision requirement to be used in the computational paths based on the mini-

mization of approximation error of the target correlation matrix. As a result of these two features, more reduction in resource usage is reported in comparison to existing approaches. Moreover, the analysis of the error inserted into the design demonstrates that there is a correlation between errors created from the truncation in the data-path. It has been found that only the correlation of truncation errors in the multiplication stage adds any significant effect to the overall approximation error as the effect of correlation in truncation errors in the adder path is found to be minimal.

- An approach to generate random numbers from many multivariate Gaussian distributions using one single generator is explored in Chapter 5. It has been found that by sharing the precision requirements of the decomposition across all target correlation matrices, up to 50% reduction in resource usage is achieved in comparison to other approaches.

- Chapter 6 investigates the design decisions taken for the production of a MVGRNG targeting two financial applications. Experimental results demonstrate that the mean square error metric in the approximation of the target correlation matrix is best for applications that involve highly correlated assets. On the other hand, the relative error metric is well suited for applications with low correlation between the assets.

7.3 Future Work

Although this thesis has investigated and tackled some of the important issues in the hardware design and implementation of a multivariate Gaussian random number generator, a number of interesting points still remain to be explored as part of the future research. Some of the possible future research directions are introduced in this section.

7.3.1 Short Term Goals

Floating Point Implementation

All of the designs exploration in this thesis have been implemented solely on fixed point number representation in order to achieve high throughput performance and low resource utilization. An alternative design consideration is the hardware implementation of the MVGRNG using floating point number representation. The multiplication in the computational block used throughout this thesis can be performed using custom precision floating point. This leads to a better dynamic range for the matrix coefficients and subsequently a better approximation to the correlation matrix. However, this comes with a cost of extra hardware resource usage required on an FPGA. Further constraints can be inserted to the existing algorithms in chapters 4 and 5 in order to evaluate which number representation to implement based on the required applications.

Multiple Constant Multiplier

One possible design consideration which can be incorporated to the algorithms proposed in chapters 4, 5 and 6 is the utilization of a multiple constant multiplier (MCM) instead of a traditional multiplier. The multiple constant multiplier performs multiplication by replacing a constant multiplication by additions and shifts which is very hardware friendly. In this way a hardware MVGRNG which is optimized for one distribution can be used to generate multivariate Gaussian random samples for another distribution which it is not optimized for. Further optimization in the use of MCM can be considered.

Further Design Optimization for Multivariate Distribution

A design methodology to generate a MVGRNG which is optimized for a target multivariate Gaussian distributions has been introduced in Chapter 4. Under this method, the algorithm and place and route is executed once for a given target correlation matrix. The entire process is required to be performed again if one or more distributions are to share the same MVGRNG.

In order to save on additional time to re-execute the algorithm and place and route, the current algorithm in Chapter 4 can be further extended as shown in Figure 7.1. The first addition to the algorithm is the Eigenvalue profile constructor block which estimates the Eigenvalue profile of the existing correlation matrix. This will be used by the second part of the algorithm where the existing profile will be compared with the profile of the new correlation matrix. If a match is detected then the existing MVGRNG can be used for

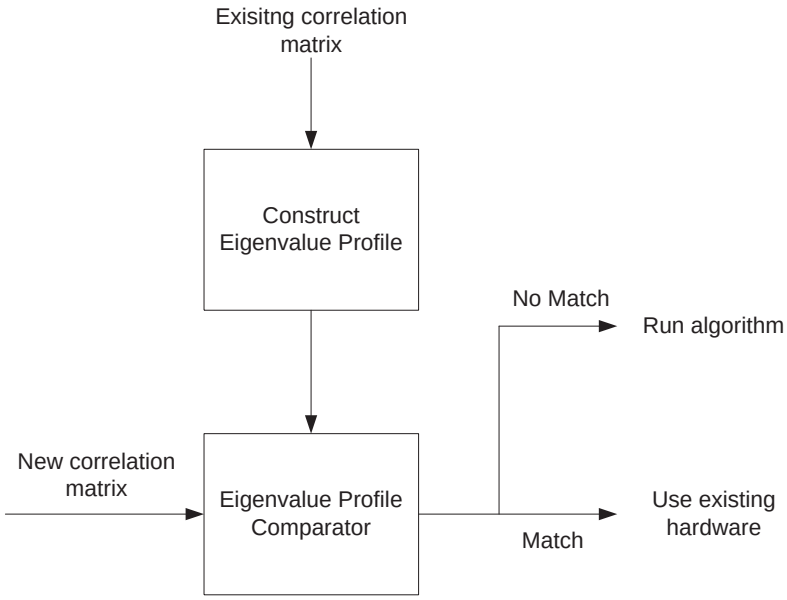


Figure 7.1: Extension of Existing Algorithm in Chapter 5.

the new correlation matrix else the algorithm is re-executed to generate a new design.

This approach can also be extended to cater for the algorithm in Chapter 5 with multiple distributions. The only difference is that the Eigenvalue profile constructor block will require to estimate the combined Eigenvalue profile for a set of target correlation matrices instead of just one.

7.3.2 Long Term Goals

- In Chapter 4, a method to estimate the correlation of the coefficients has been introduced. This is currently achieved by simulation. A more accurate and less time consuming estimate can be obtained through an approach which analytically establishes the correlation values given the coefficients.

- The methodologies proposed in Chapter 4 and 5 can be applied for the mapping of other probability distributions of interest. Word-length optimization technique can be applied to a target distribution and the precision requirements of the computational path can be investigated.

- The implementation of the Monte Carlo simulation with the target application can be implemented and optimized on an FPGA. In the case of an application which requires custom precision, an investigation can be performed in order to estimate the required precisions of the output of the random number generator block.

Bibliography

[BCC05] Christos-Savvas Bouganis, George A. Constantinides, and Peter Y. K. Cheung. A novel 2d filter design methodology for heterogeneous devices. In *FCCM '05: Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'05)*, pages 13–22, Washington, DC, USA, 2005. IEEE Computer Society.

[BH02] K. Binder and D.W. Heermann. *Monte Carlo Simulation in Statistical Physics. An Introduction (4th edition)*. Springer, 2002.

[BM58] George Edward Pelham Box and Mervin Edgar Muller. A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, 29(2):610–611, 1958.

[BS72] Donald R. Barr and Norman L. Slezak. A comparison of multivariate normal generators. *Commun. ACM*, 15(12):1048–1049, 1972.

[CCL04] George A. Constantinides, Peter Y. K. Cheung, and Wayne Luk. *Synthesis And Optimization Of DSP Algorithms*. Kluwer Academic

Publishers, Norwell, MA, USA, 2004.

[CW06] Ngai Hang Chan and Hoi Ying Wong. *Simulation Techniques in Financial Risk Management*. Wiley, 2006.

[GHS00a] Paul Glasserman, Philip Heidelberger, and Perwez Shahabuddin. Variance reduction techniques for value-at-risk with heavy-tailed risk factors. In *Proceedings of the 32nd conference on Winter simulation*, pages 604–609, 2000.

[GHS00b] Paul Glasserman, Philip Heidelberger, and Perwez Shahabuddin. Variance reduction techniques for value-at-risk with heavy-tailed risk factors. In *WSC '00: Proceedings of the 32nd conference on Winter simulation*, pages 604–609, San Diego, CA, USA, 2000. Society for Computer Simulation International.

[GL03] Paul Glasserman and Jingyi Li. Importance sampling for portfolio credit risk. *Management Science*, 51:1643–1656, 2003.

[Gla04] Paul Glasserman. *Monte Carlo Methods in Financial Engineering*. Springer, 2004.

[Gra61] Franklin A. Graybill. *An Introduction to Linear Statistical Models.* McGraw Hill, New York, 1961.

[Gra69] Franklin A. Graybill. *Introduction to Matrices with Applications in Statistics*. Wadsworth, Belmont, CA, 1969.

[Kab00] Peter Kabal. Generating gaussian pseudo-random deviates. Technical report, Department of Electrical and Computer Engineering, McGill University, 2000.

[LLV+05] Dong-U Lee, Wayne Luk, John D. Villasenor, Guanglie Zhang, and Phillip H.W. Leong. A hardware gaussian noise generator using the wallace method. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13:911–920, 2005.

[LR01] Moshe Levy and Yaacov Ritov. Portfolio optimization with many assets: The importance of short-selling. University of california at los angeles, anderson graduate school of management, Anderson Graduate School of Management, UCLA, May 2001.

[LVLL06] Dong-U Lee, John D. Villasenor, Wayne Luk, and Philip H. W. Leong. A hardware gaussian noise generator using the box-muller method and its error analysis. *IEEE Transactions On Computers.*, 55(6):659–671, 2006.

[Mar04] George Marsaglia. Evaluating the normal distribution. *Journal of Statistical Software*, 11(5):1–11, 7 2004.

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C.* Cambridge University Press, 1992.

[Ray08] Samik Raychaudhuri. Introduction to monte carlo simulation. In *WSC '08: Proceedings of the 40th Conference on Winter Simulation*, pages 91–100. Winter Simulation Conference, 2008.

[SBC08] Chalermpol Saiprasert, Christos-Savvas Bouganis, and George A. Constantinides. Multivariate gaussian random number generator targeting specific resource utilization in an FPGA. In *Reconfigurable Computing: Architectures, Tools and Applications, 6th International Symposium, ARC 2008*, pages 233–244, Berlin, Heidelberg, 2008. Springer-Verlag.

[SBC09] Chalermpol Saiprasert, Christos-Savvas Bouganis, and George A. Constantinides. Word-length optimization and error analysis of a multivariate gaussian random number generator. In *Reconfigurable Computing: Architectures, Tools and Applications, 6th International Symposium, ARC 2009*, pages 231–242, Berlin, Heidelberg, 2009. Springer-Verlag.

[SBC10a] Chalermpol Saiprasert, Christos-Savvas Bouganis, and George A. Constantinides. Design of a financial application driven multivariate gaussian random number generator for an fpga. In *Reconfigurable Computing: Architectures, Tools and Applications, 6th International Symposium, ARC 2010*, pages 182–193, Berlin, Heidelberg, 2010. Springer-Verlag.

[SBC10b] Chalermpol Saiprasert, Christos-Savvas Bouganis, and George A. Constantinides. Mapping multiple multivariate gaussian random number generators on an fpga. In *International Conference on Field Programmable Logic and Applications, FPL 2010*, pages 89–94, Berlin, Heidelberg, 2010. Springer-Verlag.

[SBCar] Chalermpol Saiprasert, Christos-Savvas Bouganis, and George A. Constantinides. An optimized hardware architecture of a multivariate gaussian random number generator. *ACM Transactions on Reconfigurable Technology and Systems*, 2009 (to appear).

[SS62] Ernest M. Scheuer and David S. Stoller. On the generation of normal random vectors. *Technometrics*, 4(2):278–281, May 1962.

[THL09] David B. Thomas, Lee Howes, and Wayne Luk. A comparison of CPUs, GPUs, FPGAs, and massively parallel processor arrays for random number generation. In *FPGA '09: Proceeding of the ACM/SIGDA international symposium on Field programmable gate arrays*, pages 63–72, New York, NY, USA, 2009. ACM.

[TL07] David B. Thomas and Wayne Luk. Non-uniform random number generation through piecewise linear approximations. *IET Computers & Digital Techniques*, 1:312–321, 2007.

[TL08] David B. Thomas and Wayne Luk. Multivariate gaussian random number generation targeting reconfigurable hardware. *ACM Trans-*

actions on Reconfigurable Technology and Systems, 1(2):1–29, 2008.

[TLLV07] David B. Thomas, Wayne Luk, Philip H.W. Leong, and John D. Villasenor. Gaussian random number generators. *ACM Comput. Surv.*, 39(4):11, 2007.

[Ver05] Michael Verhofen. Markov chain monte carlo methods in financial econometrics. *Financial Markets and Portfolio Management*, 19:397–405, 2005.

[WV08] Nathan A. Woods and Tom VanCourt. FPGA acceleration of quasi-monte carlo in finance. In *Proceedings IEEE International Conference on Field Programmable Logic and Applications*, pages 335–340, 2008.

[Xil04] Xilinx. Celebrating 20 years of innovation. *Xilinx Xcell Journal*, 48:14–16, 2004.

[Xil07] Xilinx. Virtex-4 family overview, 2007.

[ZLL+05] Guanglie Zhang, Phillip H.W. Leong, Dong-U Lee, John D. Villasenor, Ray C.C. Cheung, and Wayne Luk. Ziggurat-based hardware gaussian random number generator. In *Proceedings IEEE International Symposium on Field Programmable Logic and Applications*, pages 275–280, 2005.