



Cairo University

A NOVEL MODELING FOR INEXACT GATES AND ANALYSIS OF QUASI-TRNG APPLICATION

By

Mohamed Alaaeldin Khalil Abuelala

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

FACULTY OF ENGINEERING , CAIRO UNIVERSITY
GIZA, EGYPT

2019

A NOVEL MODELING FOR INEXACT GATES AND ANALYSIS OF QUASI-TRNG APPLICATION

By
Mohamed Alaaeldin Khalil Abuelala

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

Under the Supervision of

Dr. Ahmed K. F. Khattab **Prof. Dr. Hossam A. H. Fahmy**

.....

Associate Professor

Electronics and Communications Department
Faculty of Engineering , Cairo University

.....

Professor

Electronics and Communications Department
Faculty of Engineering , Cairo University

Prof. Dr. Amr G. A. Wassal

.....

Professor

Computer Department
Faculty of Engineering , Cairo University

FACULTY OF ENGINEERING , CAIRO UNIVERSITY
GIZA, EGYPT
2019

A NOVEL MODELING FOR INEXACT GATES AND ANALYSIS OF QUASI-TRNG APPLICATION

By
Mohamed Alaaeldin Khalil Abuelala

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

Approved by the Examining Committee:

Dr. Ahmed Khattab Fathy Khattab,	Thesis Main Advisor
---	---------------------

Prof. Dr. Hossam Aly Hassan Fahmy,	Advisor
---	---------

Prof. Dr. Amr Galaleldin Ahmed Wassal,	Advisor
---	---------

Prof. Dr. Ahmed Hussein Mohamed Khalil,	Internal Examiner
--	-------------------

Prof. Dr. Mohamed Watheq El-Kharashi,	External Examiner
Professor at Faculty of Engineering, Ain Shams University	

FACULTY OF ENGINEERING , CAIRO UNIVERSITY
GIZA, EGYPT
2019

Engineer's Name: Mohamed Alaaeldin Khalil Abuelala
Date of Birth: 24/10/1992
Nationality: Egyptian
E-mail: mohamed.abuelala@ieee.org
Phone: (+20) 100-544-7786
Address: Corniche El Nil st., Maadi, Cairo, 11728
Registration Date: 01/10/2014
Awarding Date: 2019
Degree: Master of Science
Department: Electronics and Communications Engineering



Supervisors:

Dr. Ahmed K. F. Khattab
Prof. Dr. Hossam A. H. Fahmy
Prof. Dr. Amr G. A. Wassal

Examiners:

Dr. Ahmed Khattab Fathy Khattab (Thesis main advisor)
Prof. Dr. Hossam Aly Hassan Fahmy (Advisor)
Prof. Dr. Amr Galaleldin Ahmed Wassal (Advisor)
Prof. Dr. Ahmed Hussein Mohamed Khalil (Internal examiner)
Prof. Dr. Mohamed Watheq El-Kharashi (External examiner)
(Professor at Faculty of Engineering, Ain Shams University)

Title of Thesis:

A Novel Modeling for Inexact Gates and Analysis of Quasi-TRNG
Application

Key Words:

Modeling; Algorithm; Logic families; Arithmetic Circuits; Random Number Generators

Summary:

This work aims to propose a new model for inexact logic gates. The use of these logic gates as simple cells in complex circuits is investigated as well as checking the performance of different applications such as adders and random number generators. The work helps to abstract the error in logic gates by modeling it to a probability density function, and injecting this error in the output of the gate. It also gives a guide to solve the non-uniformity in random number generator for future work.

Disclaimer

I hereby declare that this thesis is my own original work and that no part of it has been submitted for a degree qualification at any other university or institute.

I further declare that I have appropriately acknowledged all sources used and have cited them in the references section.

Name: Mohamed Alaaeldin Khalil Abuelala

Date: / /

Signature:

Dedication

To my family and to the memory of my uncle who encouraged me to finish it.

Acknowledgements

All praise is to almighty **ALLAH** who guided me, aided me and gave me strength to complete this thesis. Many people have helped me throughout my Master's work and I would like to express my sincerest thanks to all of them. I am grateful to my advisors, **Dr. Ahmed Khattab** and **Prof. Dr. Amr Wassal** for their guidance, kind patience and continuous support during my postgraduate years of study. I dedicate a special feeling of devotion to **Prof. Dr. Hossam Aly Hassan Fahmy** for his kind guidance inspiration and encouragement. I would also like to thank my friends and colleagues in **Si-Ware Systems** for their true help and support throughout my postgraduate study. Finally, I would like to thank my family for helping me survive all the stress and pressure during my study and not letting me give up.

Table of Contents

Disclaimer	i
Dedication	ii
Acknowledgements	iii
Table of Contents	iv
List of Tables	vii
List of Figures	ix
List of Symbols and Abbreviations	xii
List of Publications	xiii
Abstract	xiv
1 INTRODUCTION	1
1.1 Inexact Computing	1
1.1.1 Approximate Computing	1
1.1.2 Probabilistic (Stochastic) Computing	2
1.1.3 Comparison Overview	2
1.2 Thesis Contributions	4
1.3 Thesis Outline	4
2 BACKGROUND AND LITERATURE REVIEW	5
2.1 Stochastic Computing	5
2.1.1 Stochastic Number (SN)	5
2.2 Probabilistic Computing	7
2.2.1 Probabilistic CMOS	7
2.2.2 Memristor	8
2.2.2.1 Sequential Memristor Logic	9
2.2.2.2 Deterministic AND Logic Gate	10
2.2.2.3 Probabilistic Sequential Logic	11
2.2.2.4 Probabilistic AND Logic	12
2.3 Applications Overview	14

2.3.1	Adder	14
2.3.1.1	Lower-bit OR Adder (LOA)	15
2.3.1.2	Probabilistic Full Adder (PFA)	15
2.3.2	Random Number Generators	16
2.3.2.1	True Random Number Generators	16
2.3.2.2	Pseudo Random Number Generators	17
2.3.2.3	Comparison between TRNG and PRNG	17
3	INEXACT GATES MODELING	19
3.1	Approximate Gate	20
3.2	Probabilistic Gate	22
3.3	Simple and Complex Gates	24
3.4	Results	25
3.4.1	Approximate Gates	25
3.4.1.1	Simple Approximate Gates	25
3.4.1.2	Complex Approximate Gates	25
3.4.2	Probabilistic Gates	27
3.4.2.1	Simple Probabilistic Gates	27
3.4.2.2	Complex Probabilistic Gates	30
4	ANALYSIS OF CHAINS AND ADDER MODELS	36
4.1	Chain of Gates	37
4.1.1	Inverter Chain	37
4.1.2	XOR chain	39
4.1.2.1	Chain of simple XOR	40
4.1.2.2	Chain of topology-I XOR	41
4.1.2.3	Chain of topology-II XOR	41
4.1.2.4	Chain of topology-III XOR	42
4.1.2.5	Chain of topology-IV XOR	43
4.1.2.6	Comparison between different topologies of XOR	44
4.2	Analysis of ripple carry adder	46
4.2.1	Output percentage error versus probability of error	46
4.2.2	Mean error distance versus imprecise bits	49
4.3	Conclusion	50
5	QUASI-TRNG IMPLEMENTATION	51
5.1	Linear Feedback Shift Register	51
5.2	Statistical Tests	53

5.3	Results	53
5.3.1	Different Topologies	54
5.3.1.1	Histograms of Order-8 LFSR	54
5.3.1.2	NIST tests results	54
5.3.1.3	Probabilities of Order-4 LFSR	56
5.3.2	Different Orders	59
5.3.3	Different Polynomials	62
5.4	Conclusion	63
6	CONCLUSION AND FUTURE WORK	64
6.1	Summary of The Work	65
6.2	Impact on The Digital Flow	66
6.3	Future Work	66
	References	67
	Appendix A MEMRISTIVE LOGIC GATES	71
A.1	Deterministic Sequential logic	73
A.2	Probabilistic Sequential logic	74

List of Tables

1.1	The relation between accuracy and precision	3
1.2	Comparison between approximate and probabilistic computing	3
2.1	Applied voltages across terminals through the cycles for sequential AND, OR, NAND, and NOR logic gates	10
2.2	The truth table for sequential AND logic gate	11
2.3	$P(0)$, $P(1)$, and accuracy of sequential AND, OR, NAND, and NOR logic gates based on stochastic memristors	13
2.4	Comparison between LOA and PFA	16
2.5	Comparison between true and pseudo random number generators	18
3.1	Truth table for conventional OR gate and its approximate implementations with different errors	22
3.2	Truth table for AND, OR, XOR logic gates with their $P_e = 0$ and 1	24
3.3	Characteristics of XOR topologies	25
3.4	OPE in simple logic gates with error as uniform distribution	29
3.5	OPE in simple logic gates with error as normal distribution	29
3.6	OPE in different complex XOR logic gate with error as uniform distribution	31
3.7	OPE in different complex XOR logic gate with error as normal distribution	31
4.1	OPE for error uniform distribution in chain of inverters	38
4.2	OPE(%) for Normal Distribution in Chain of Inverters	39
4.3	Truth Table of SUM in FA for different input combinations	40
4.4	OPE(%) for uniform and normal distributions in the chain of simple XOR	40
4.5	OPE(%) for uniform and normal distributions in the chain of topology-i XOR	41
4.6	OPE(%) for uniform and normal distributions in the chain of topology-ii XOR	42
4.7	OPE(%) for uniform and normal distributions in the chain of topology-iii XOR	43
4.8	OPE(%) for uniform and normal distributions in the chain of topology-iv XOR	43
4.9	OPE(%) for uniform distribution in the second stage for XOR chain	45
4.10	OPE(%) for normal distribution in the second stage for XOR chain	45
5.1	10-bit Programmable LFSR Polynomials	52

5.2	The proportion value (PP), and the availability of p-values (PV) of the NIST tests showing results for probabilistic LFSR model with accurate AND gate in the feedback	56
5.3	The proportion value (PP), and the availability of p-values (PV) of the NIST tests showing results for probabilistic LFSR model with probabilistic AND gate in the feedback	56
5.4	Number probabilities for different topologies of order-4 programmable LFSR with important statistical values	58
5.5	The proportion value (PP), and the availability of p-values (PV) of the NIST tests showing results for probabilistic LFSR model for different orders	61
A.1	The truth table for sequential OR logic gate	73
A.2	The truth table for sequential NAND logic gate	73
A.3	The truth table for sequential NOR logic gate	74

List of Figures

1.1	Different imprecise computing techniques	1
1.2	The relation between probability density of generated imprecise results and target value	2
2.1	Conversion circuit (a) binary to stochastic, and (b) stochastic to binary	6
2.2	Stochastic multiplier using AND gate : (a) model, (b) exact, and (c,d) approximate	7
2.3	(a) PCMOS switch (b) The digital representation for '0' and '1' with the switching probability of error for a PCMOS	8
2.4	The relation between four fundamental elements	9
2.5	Diagram for the applied inputs through the cycles of sequential AND logic gate	11
2.6	Analysis diagram for probabilistic sequential AND logic gate	12
2.7	Block diagram for simple ripple carry adder	14
2.8	Block Diagram of LOA	15
2.9	Block Diagram of PFA	15
2.10	Block Diagram of TRNG	17
3.1	Classification and paradigm for inexact computing techniques	19
3.2	Output percentage error model	20
3.3	The model for approximate logic gate	20
3.4	The flowchart for approximate logic gate	21
3.5	The model for probabilistic logic gate	22
3.6	The flowchart for probabilistic logic gate	23
3.7	Different Topologies for XOR gate	24
3.8	OPE for AND, NAND, OR, NOR, and XOR with error in input combination "00"	26
3.9	OPE for different XOR topologies with error in gates input combinations: (a) "00", (b) "01", (c) "10", (d) "11", (e) fixed for gates of same type , and (f) random for different gates	26
3.10	OPE for simple gates, and CDF of the injected noise (on the right), and histogram of the injected noise with the fitted PDF of it (on the left) are plotted for: (a) uniform, and (b) normal distributions	27
3.11	$P(0)$, $P(1)$, and OPE for each simple gate for: (a) uniform, and (b) normal distributions	28
3.12	OPE for different XOR topologies, and CDF of the injected noise (on the right), and histogram of the injected noise with the fitted PDF of it (on the left) are plotted for: (a) uniform, and (b) normal distributions	30

3.13	$P(0)$, $P(1)$, and OPE for each gate in XOR topology-i for error as distribution (a) Uniform, and (b) Normal	32
3.14	$P(0)$, $P(1)$, and OPE for each gate in XOR topology-ii for error distribution (a) Uniform, and (b) Normal	33
3.15	$P(0)$, $P(1)$, and OPE for each gate in XOR topology-iii for error distribution (a) Uniform, and (b) Normal	34
3.16	$P(0)$, $P(1)$, and OPE for each gate in XOR topology-iv for error distribution (a) Uniform, and (b) Normal	35
4.1	5-stages chain of inverters	37
4.2	$OPE(\%)$ for each stage in the inverter chain (on the right) and Histogram of the injected noise with the fitted PDF of it (on the left) are plotted for, (a) Uniform Distribution, and (b) Normal Distribution	38
4.3	(a) 2-stages XOR chain, and (b) Sum function in Full adder	39
4.4	$OPE(\%)$ for each stage in the chain of simple XOR gate (on the right) and Histogram of the injected noise with the fitted PDF of it (on the left) are plotted for, (a) Uniform Distribution, and (b) Normal Distribution	40
4.5	$OPE(\%)$ for each stage in the chain of topology-i XOR gate (on the right) and Histogram of the injected noise with the fitted PDF of it (on the left) are plotted for, (a) Uniform Distribution, and (b) Normal Distribution	41
4.6	$OPE(\%)$ for each stage in the chain of topology-ii XOR gate (on the right) and Histogram of the injected noise with the fitted PDF of it (on the left) are plotted for, (a) Uniform Distribution, and (b) Normal Distribution	42
4.7	$OPE(\%)$ for each stage in the chain of topology-iii XOR gate (on the right) and Histogram of the injected noise with the fitted PDF of it (on the left) are plotted for, (a) Uniform Distribution, and (b) Normal Distribution	43
4.8	$OPE(\%)$ for each stage in the chain of topology-iv XOR gate (on the right) and Histogram of the injected noise with the fitted PDF of it (on the left) are plotted for, (a) Uniform Distribution, and (b) Normal Distribution	44
4.9	$OPE(\%)$ for 2^{nd} stage in different XOR chains, and CDF of the injected noise in this stage (on the right) and Histogram of the injected noise with the fitted PDF of it (on the left) are plotted for, (a) Uniform Distribution, and (b) Normal Distribution	44
4.10	Full adder model and its logic gates combinations	46
4.11	OPE for sum and C_{out} in RCA with different imprecise bits using different XOR topologies for error distribution as: uniform (on the left), and normal (on the right)	48
4.12	MED vs the number of imprecise lower bits in different adders with $P_e = 0.1$ (on the left), and $P_e = 0.2$) on the right	49
5.1	Block Diagram of 10-bit generic LFSR	52

5.2	Histogram of the symbols is plotted for different XOR topologies with, (a) Accurate, (b) Probabilistic Simple, (c) Probabilistic Topology-I, (b) Probabilistic Topolgy-II, (c) Probabilistic Topology-III, and (d) (b) Probabilistic Topology-IV	55
5.3	Diagrams for number probabilities of order-4 programmable LFSR	57
5.4	Histogram of the symbols is plotted for different PRNG Orders, (a) Order-4, (b) Order-6, (c) Order-8, and (d) Order-10	61
5.5	Histogram of the symbols is plotted for different polynomials, (a) 1000, (b) 1001, (c) 1010, (d) 1011, (e) 1100, (f) 1101, (g) 1110, and (h) 1111	62
A.1	Diagram for the applied inputs through the cycles of sequential OR logic gate	74
A.2	Diagram for the applied inputs through the cycles of sequential NAND logic gate	75
A.3	Diagram for the applied inputs through the cycles of sequential NOR logic gate	75

List of Symbols and Abbreviations

Symbol	Description
ϵ	A metric for imprecise gate depends on source of error
μ	The mean or expectation of the normal distribution
σ	The standard deviation of the normal distribution

Abbreviation	Description
TPU	Tensor Processing Unit
SN	Stochastic Number
RNG	Random Number Generator
TRNG	True Random Number Generator
PRNG	Pseudo Random Number Generator
OPE	Output Percentage Error
P_e	Probability of Error
ED	Error Distance
MED	Mean Error Distance
LFSR	Linear feedback Shift Register

List of Publications

Accepted:

1. Mohamed A. K. Abuelala, Amr Wassal, Ahmed Khattab, Hossam A. H. Fahmy. "A Novel Model for Injecting Error in Probabilistic Gates." In The 31st International Conference on Microelectronics (IEEE-ICM), Cairo, Egypt, 2019

Abstract

Computing systems become progressively substantial with media and data handling. An accurate result is not necessary in media processing such as in audio, image, video, graphics, pattern recognition, and data mining. In these applications, it is adequate to obtain approximate or less-than-optimal result. For example, Human recognition is not sensitive to the high frequency variations in signal processing applications. Also, there is no need to quantize the signal in higher bits once the noise floor is acceptable.

There are many sources to tolerate the imprecise results as the ability of human brain to detect the missing information, and the redundancy of input data which allows the algorithm to be adequate despite being lossy. This imprecise-tolerance is proved to be helpful in energy reduction for these applications. Different computing systems are introduced for this purpose as approximate, probabilistic, and stochastic computing.

In this thesis, a literature review for different error-resilient paradigms in computing systems is presented. Also, an investigation for the effect of probabilistic behavior is introduced for some devices as memristor, and probabilistic CMOS. We propose an abstract model for inexact gates with new metric, output percentage error, that gives a percentage result for the total inexact outputs compared with the generated ones in accurate logic gates. This model can be useful for EDA tools in modeling the effect of these inexact logic gates on the performance of complex applications.

We discuss the proposed approximate and stochastic gate models, as well as the modeling of different simple gates, and four complex XOR topologies. Furthermore, a detailed analysis for different chains as inverter, and XOR gates connected serially to examine dependencies over the chain is provided. Two different applications are discussed; the realization of adder model, and the performance of programmable true random number generator using simple stochastic XOR gates. Finally, some potential areas are stated for future work.

Chapter 1

Introduction

1.1 Inexact Computing

There is a vital concern about the future computing system that can solve difficult problems in an energy-efficient manner. Many features need to be granted in the next computing system according to the target application. To get the better usage of the system, it is essential to understand the application's requirements as some applications can tolerate some error in their arithmetic units to get low-power consumption. For example, allowing about 5% loss in classification accuracy for k-means clustering algorithm can lead to 50x improvement in energy saving compared to the fully accurate classification [1]. In many fields as machine learning, pattern recognition and signal processing, accurate results are not necessary and they can accept some error in the results.

The error in logical gate may be deterministic or non-deterministic depending on its source. In fact, this is the fundamental concept of imprecise or inexact computing which indicates that the system either hardware or software does not produce the exact result at each run, and the level of application will tolerate this error. In contrast, the exact computing techniques will guarantee to deliver correct data to the application layer, such as using correction mechanisms to identify and correct any faulty bits in communication data streams. Several recent researches have explored imprecise computing in both hardware and software, figure 1.1 shows the classification of different methods for imprecise computing [2].

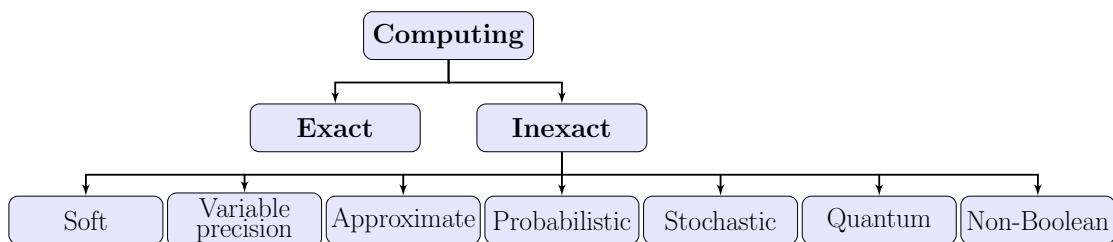


Figure 1.1: Different imprecise computing techniques

1.1.1 Approximate Computing

Approximate computing is considered the generic technique for all imprecise computing techniques, and a very promising approach to energy-efficient design. It utilizes deterministic hardware designs that produce imprecise results with little accuracy loss, and allows improvement in energy and performance efficiency by involving the statistical properties of the data or algorithms. The same answer is always can be detected after the

same number of iterations.

Approximate computing has been used in a variety of domains where the applications are error-tolerant, as: noisy inputs in sensors, probabilistic outputs in machine learning, and face-detection applications. Recently, Google uses approximate computing in their custom ASIC Tensor processing units (TPU) which operates as an accelerator for machine learning on Google cloud [3].

For approximation techniques, many researches concentrate on the characterization concepts for these techniques as correctness, controllability, and reproducibility [4]. Several techniques can be used for manipulating approximate computing on different levels, as: Loop perforation [5, 6] in **Software** layer which is done by skipping some iterations of the loop, or function substitution to increase the performance of target application. For **system architecture**, many applications do not need all bits for data of floating point arithmetic, so accuracy reduction is done by removing lower bits in architecture layer [7, 8]. Also, it is achieved on **device level** by using new memory technology to store data approximately, as memristor.

1.1.2 Probabilistic (Stochastic) Computing

Stochastic computing is the general methodology of the probabilistic computing. The initial concepts of this technique were proposed in the 1950s as an alternative to conventional computing. It is a non-deterministic methodology to computation, which can be implemented in hardware or software. The specific probabilistic computing leverages the physical limits relating to the scaling down of feature size which cause intrinsic probabilistic behavior of the underlying circuit fabric, or the stochastic nature of a binary switch under the effect of thermal noise [9].

Stochastic computing uses random binary bit streams that are executed in time and in series. The computation is performed by applying these streams on the gate and measuring its statistic. Stochastic computing gets different answers every time which mean it has a loss of precision.

1.1.3 Comparison Overview

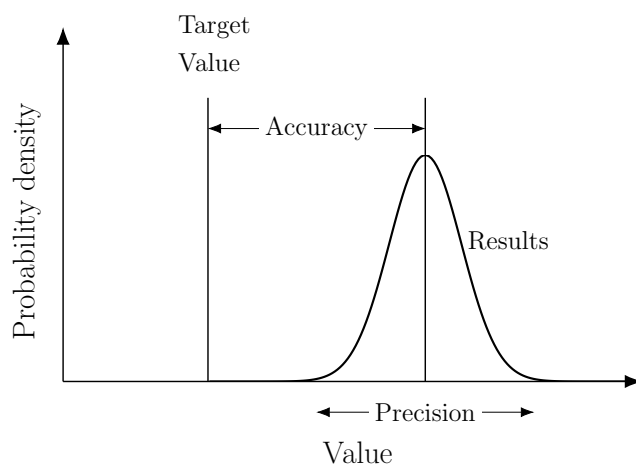


Figure 1.2: The relation between probability density of generated imprecise results and target value

Accuracy and precision [10] are the most important metrics that show the difference between approximate and stochastic computing. For different results of measurements with errors related to each result, precision is used to describe the dissemination of these errors corresponding to each other. Precision depends on the number of digits to present the result as increasing these numbers will make a better precision for the result. On the other hand, accuracy is an indicator for the correctness of a measurement, simply it describes how close the average approximate results to the accurate result. The relation between the generated results and the target (reference) value is shown in figure 1.2.

Approximate computing points to how close results are to each other every run, while stochastic computing calculates close results to the true value. Table 1.1 shows the relation between precision and accuracy as independent quantities which means the results can be precise but not accurate as the case in approximate computing, or using hardware which provides almost accurate but imprecise results as in stochastic computing. Table 1.2 presents the comparison between approximate and stochastic computing from different aspects as error type, accuracy, and precision [11].

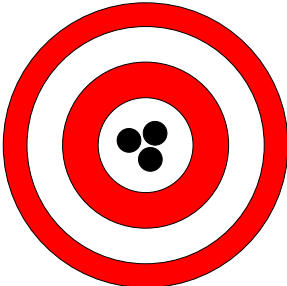
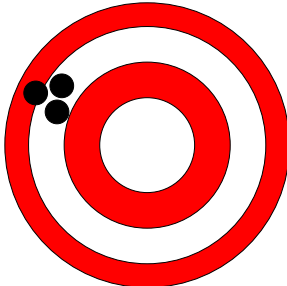
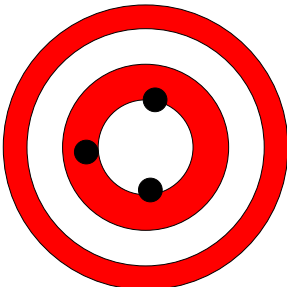
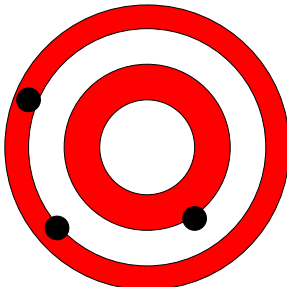
	Accurate	Not Accurate
Precise		
Not Precise		

Table 1.1: The relation between accuracy and precision

Approximate Computing	Probabilistic Computing
Deterministic Error	Not-deterministic Error
Precise but not accurate	Accurate but not precise
The same answer after the same number of iterations	Different answer every time
Can give low power consumptions, better resilience	

Table 1.2: Comparison between approximate and probabilistic computing

1.2 Thesis Contributions

The contributions of the thesis can be summarized as follows:

1. An abstract model for imprecise computing (approximate and probabilistic) which can be useful in modeling their effects in EDA tools.
2. Investigation for different circuits as: chain of gates, adders, and random number generator to check the injected stochastic behavior error effect on the functionality of these circuits.
3. Analysis of proposed random number generator which is implemented using generic pseudo implementation to generate fully digital true random sequences.

1.3 Thesis Outline

This thesis is organized as follows:

- *Chapter 2*: provides a literature review for stochastic and approximate computing. Also, investigating the effect of probabilistic behavior for some devices as memristor. Furthermore, an overview for different imprecise applications is introduced.
- *Chapter 3*: discusses the proposed approximate and stochastic gate models, as well as the modeling of different simple gates, and four complex XOR topologies. Also, simulation results are presented with a comparison for the provided XOR topologies.
- *Chapter 4*: provides a detailed analysis for different chains as inverter, and XOR gates connected serially to examine dependencies over the chain. Also, The results for adder models are investigated.
- *Chapter 5*: shows the realization of programmable true random number generator using stochastic XOR gates, and investigates different histograms of the generated random sequences. Also, it presents the test results of output analysis using the NIST Sp. 800-22 statistical tests.
- *Chapter 6*: concludes the thesis and shows potential areas for future work.

Chapter 2

Background and Literature Review

This chapter reviews important concepts of stochastic, probabilistic, and approximate computing. It surveys the features of stochastic computing, as: the error flexibility, small size, and probabilistic forms which make stochastic computing able to replace conventional techniques in certain applications. Then, the probabilistic behavior of some promising devices for the probabilistic computing is reviewed. Also, an application literature is presented for inexact adder techniques beside the random number generator.

2.1 Stochastic Computing

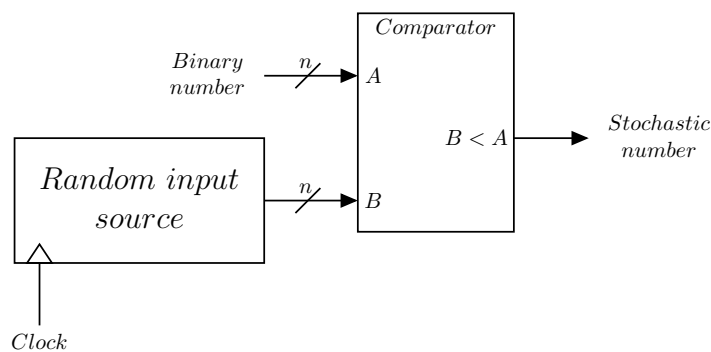
The paper of Von Neumann [12] introduced basic concepts about stochastic computing but the theory was fully improved in the 1960s [13, 14]. SC was proposed as an alternative low-cost computing system rather than traditional binary system using logic elements. It represents and forms data in the scheme of digitized probabilities. SC utilizes high-ease arithmetic units which were the main design concern in the past. Regardless of this advantage, SC was considered as unfeasible cause of low precision and long calculation times. However, modern computing systems are subjected to two factors which limit the system implementation: (1) Application specifications as high reliability, and low power consumption. (2) Physical phenomena like manufacturing defects, process variations, and soft errors. These aspects make an advantage for error resilience which is a behavior of probabilistic computing technique. Many surveys and approaches were presented to discuss benefits and drawbacks of stochastic computing and its history [15–17].

2.1.1 Stochastic Number (SN)

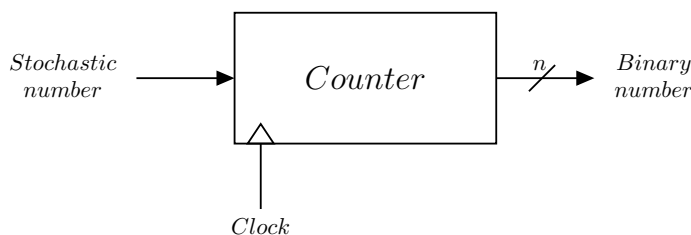
Numbers in SC are described as probabilities instead of arithmetic under normal and faulty condition that can be processed by simple circuits. This probability can be denoted as a number p which shows the probability of detecting 1's in a stream of bits with length N , it depends only on this ratio $\frac{\text{Numb.of 1's}}{N}$ not on their positions in the stream which means they have equal weights in the representation. For example, 12-bits stream (0001 0000 0110) has $p = 0.25$. It is important to note that there are no constraints on the length or bit's weight of bit-stream which means 0.25 can have many representations as (0100), (0001), and (0001 0100). Despite this feature makes the representation of stochastic number flexible and error tolerant, it forces a limitation for the accuracy of SC. Increasing the length of bit stream can be a good solution to get a higher accuracy for SC but it will affect the runtime.

Conversion between binary numbers and stochastic numbers, and vice versa, are important factors in SC. The simple model which widely used for conversion between binary to stochastic is shown in figure 2.1i. SC operates on n -bit long sequences generated

from random source as pseudo random number generator. These sequences are compared with n-bit binary input to produce sequence denoted by S of stochastic number. The mechanism of comparison is if the binary number is less than the generated random number, the result will be 0. and 1 otherwise. It is simple to generate binary number from stochastic number. As the representation of probability p depends on number of 1s in the stochastic sequence, so only a counter can be used to calculate number of these 1s and get p as shown in figure 2.1ii.



(i)



(ii)

Figure 2.1: Conversion circuit (a) binary to stochastic, and (b) stochastic to binary

Using simple logic gates to calculate complex computations was the main interest of SC. For example, suppose that there are two bit-streams S_1, S_2 which are random, independent streams with probabilities p_1, p_2 respectively, and a multiplication operation will be computed. SC circuit of simple logical AND gate can be used to perform the multiplication and the generated sequence will have output probability $p_1 * p_2$. Figure 2.2 shows the utilization of AND gate as a stochastic multiplier between different representations of $\frac{2}{8}$ and $\frac{4}{8}$ for exact and approximate computation, as follow:

- Figure 2.2ii represents the exact multiplication as the output probability is $\frac{1}{8}$.
- Two different representations for $\frac{2}{8}$ are used in figures 2.2iii, 2.2iv to show the effect of alternative stochastic number representation. The output probabilities are $\frac{0}{8}$ and $\frac{2}{8}$ can be explained as approximation to the exact result. However, this example show the accuracy problem in SC so generating good stochastic numbers is a fundamental feature to use SC in different applications.

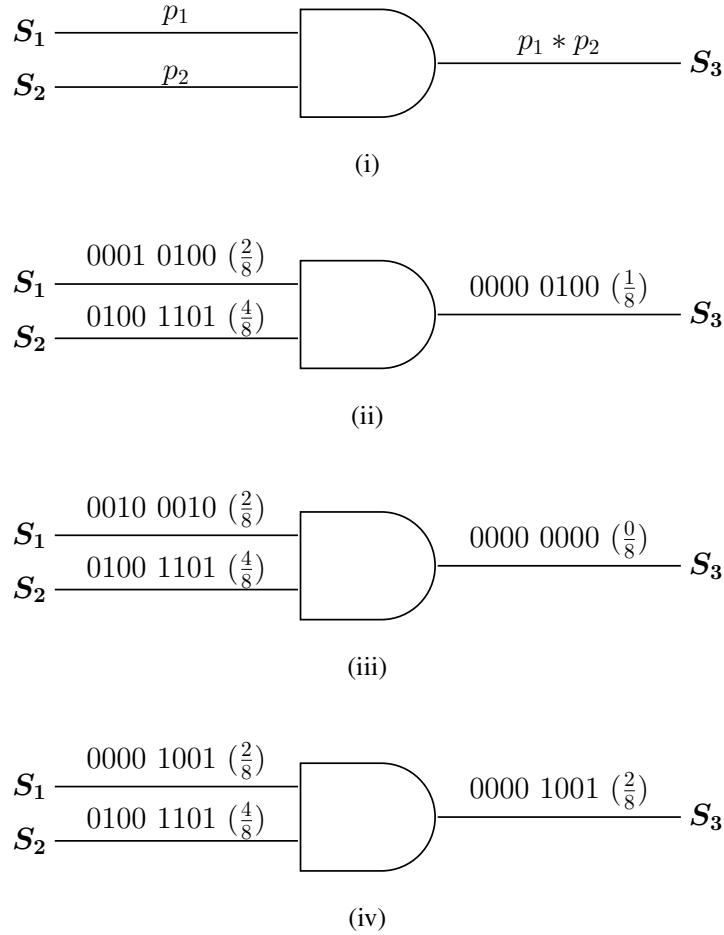


Figure 2.2: Stochastic multiplier using AND gate : (a) model, (b) exact, and (c,d) approximate

2.2 Probabilistic Computing

Development rate in semiconductor industry is depending on Moore's law which state that every 18 months the number of transistors will be doubled for the same area [18, 19]. Now, there is a need to keep the same rate of development so we have to use new computing methods and develop new devices due to the physical constraints. Probabilistic computing is one of these promising techniques which can provide acceptable incorrect results for applications that can tolerate some errors. Also, using probabilistic computing can improve other aspects of design such as power consumption. There are many methods introduced to improve probabilistic computing from the level of devices till block's structure. In this section, two promising devices are reviewed: Probabilistic CMOS (PCMOs), and Memristor.

2.2.1 Probabilistic CMOS

Term probabilistic CMOS (PCMOs) appeared as a result of scaling CMOS device into the nanoscale regime. This scaling which represents an end of Moore's law [20, 21], leads to probabilistic behavior in the device operation cause of process variations and noise fluctuations which show challenge in the traditional design methodologies. Therefore, researchers have worked on the characterization of these probabilistic CMOS devices to

overcome this challenge at different levels, as using these devices in conventional design circuits is unsuitable cause they depend on deterministic behavior for the used devices. This work of Borkar et al. [22] shows the impact of probabilistic behavior on design circuits.

In this section an overview for the modeling and probabilistic behavior of PCMOS is presented. The concern will be about the reduction of supply voltage V_{dd} in these CMOS devices which make the probabilistic behavior controlled by this reduction and the effect of thermal noise. An abstract clarification for probabilistic switches which can be useful in the implementation of imprecise logic is presented in [23]. Figure 2.3i shows a simple inverter CMOS whose output V_{out} probabilistic behavior is controlled by thermal noise only, this is the modeling of PCMOS switch.

The digital representation for logic '0' and '1' is presented in figure 2.3ii as logic '0' is represented by a Gaussian distribution whose mean value is 0 and variance is σ , in this case, the correct output will be 0. Same for logic '1', as it can be mapped to a normal distribution with a mean value V_{dd} and variance σ , the correct output will be 1 in this case. The dark gray in the graph shows probability of '1' to be treated as '0', and the light gray shows the probability of '0' to be treated as '1' which means the gray region represents the probability of error in case of switching for a PCMOS inverter. From this representation, probability of correctness indicated by p can be expressed as follow:

$$p = 1 - \frac{1}{2}erfc\left(\frac{V_{dd}}{2\sqrt{2}}\right)$$

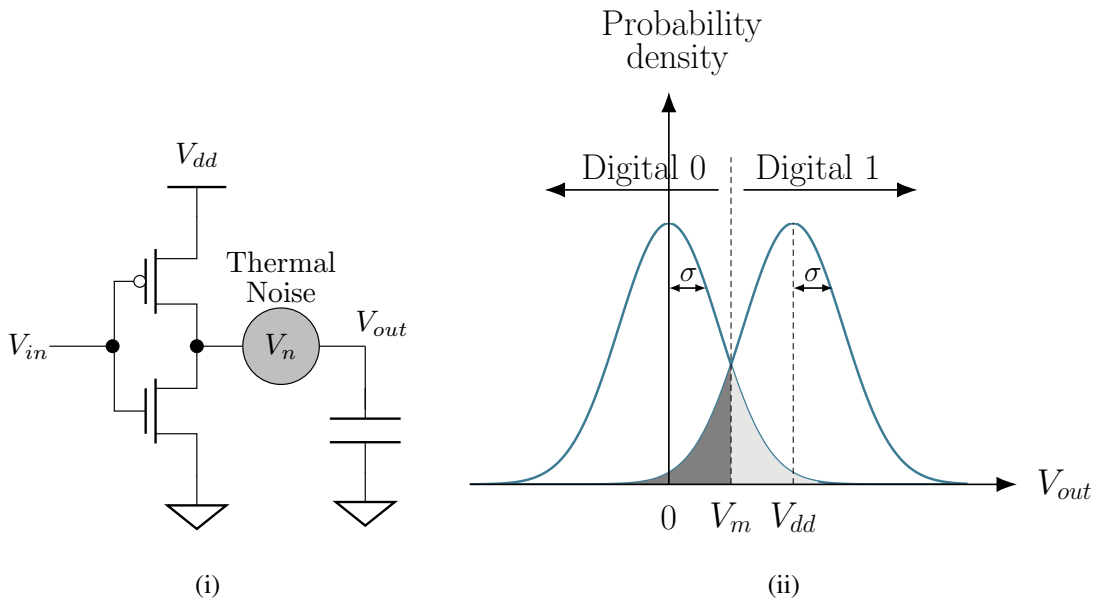


Figure 2.3: (a) PCMOS switch (b) The digital representation for '0' and '1' with the switching probability of error for a PCMOS

2.2.2 Memristor

Memristor is a new element in electrical circuits. Scientists consider it as the fourth element beside resistor, capacitor, and inductor which clarify the relation between flux and charge as shown in figure 2.4 which describes relationship between the four elements [24].

One of memristor's unique features is that computing inside the memory that decreases time to transfer data between arithmetic unit and memory.

Memristor changes its resistance according to the applied input between 2 states: low resistance R_{on} and high resistance R_{off} . The switching of resistance is another important feature in memristor as it is stochastic in terms of the time period and level of the input voltage. Different methods for using memristor in a logic domain are proposed in [25–27]. Memristor can be used in both approximate and stochastic computing [28, 29]. Sequential memristor logic method is reviewed in this section.

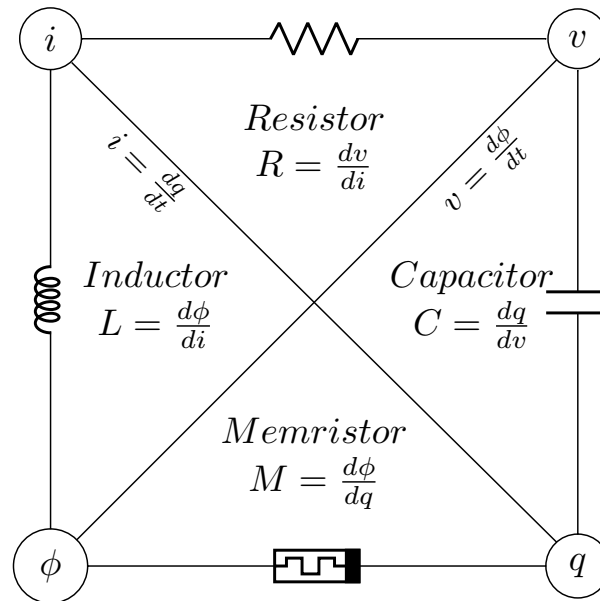


Figure 2.4: The relation between four fundamental elements

2.2.2.1 Sequential Memristor Logic

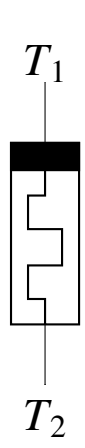
The memristor has two state of resistance switching R_{off} and R_{on} which are treated as logic '0' and '1' respectively in the digital domain [25]. In sequential memristor logic, the hardware of any logic gate is very simple as it uses only memristors. A sequential voltage levels are applied into the top and bottom electrodes of the memristor which are used as memristor's terminals depending on the logical inputs for the gate. One memristor can give 14 out of 16 logical operations [30]. The concept depends on the voltage difference between memristor's terminals, as follow:

- The terminals of memristor are indicated as T_1 and T_2 . The threshold voltage defines the polarity of applied input voltage, if it is above threshold, the input voltage will be positive and vice versa.
- If the applied input voltage is positive, the resistance state of memristor will be ON or logically '1'.
- If the applied input voltage is negative, the resistance state of memristor will be OFF or logically '0'.
- Zero-input voltage makes no change in the state of memristor.

The basic 2-inputs logical operations (AND, OR, NAND and NOR) which has inputs in_1 , and in_2 can be obtained by applying sequential voltage levels corresponding to these inputs into memristor's terminals through 3 cycles, and the logical output is determined by the final resistance state. The operations through three cycles are as follows:

1. The initialization cycle to confirm the resistance state is OFF.
2. The first logical input (in_1) as a voltage level is applied.
3. The second logical input (in_2) as a voltage level is applied, the output logic after the third cycle is based on the resistance state either ON or OFF.

T_2 has different voltage in each cycle to obtain the desired logical operation as shown in table 2.1 [31]. The cycles show DeMorgan's theorem rules of breaking NAND to negative OR, and vice versa. Another approach by initializing the first cycle as ON state and applying different values for T_2 is presented in [29].



Operation	Terminal	Cycle 1	Cycle 2	Cycle 3
AND	T_1	0	in_1	in_2
	T_2	1	0	1
OR	T_1	0	in_1	in_2
	T_2	1	0	0
NAND	T_1	0	1	1
	T_2	1	in_1	in_2
NOR	T_1	0	1	0
	T_2	1	in_1	in_2

Table 2.1: Applied voltages across terminals through the cycles for sequential AND, OR, NAND, and NOR logic gates

2.2.2.2 Deterministic AND Logic Gate

A detailed demonstration for AND logic gate is presented to clarify the operation of deterministic sequential logics. The diagram in figure 2.5 shows the used cycles to implement the sequential AND logic gate. Starting from the left symbol which describes the used model through the operation, it shows the memristor device as a square block and inside it the previous state of memristor. The logic gate inputs will be applied across the terminals T_1 and T_2 of the memristor. Cycle 1 doesn't care about the previous state of the memristor as a negative voltage is applied to switch the memristor's resistance into OFF state. In cycle 2, in_1 is applied to T_1 , and '0' is connected to T_2 which will make the resistance state ON in case of $in_1 = 1$ as the memristor will have a positive voltage. If $in_1 = 0$ is applied, zero potential difference across memristor's terminal is the result which makes no change in the state of memristor and keep the previous state OFF. Similarly in cycle 3, in_2 is applied to T_1 , and T_2 is driven by '1' which will make the state either maintain its state in case of $in_2 = 1$ or OFF for $in_2 = 0$. The last cycle, the resistance state can be read by applying a small signal S_r to T_1 . Table 2.2 shows the truth table for sequential AND gate with resistance states through the operational cycles for all the

combinations of inputs. Truth tables for sequential OR, NAND, NOR logic gates are in Appendix A.

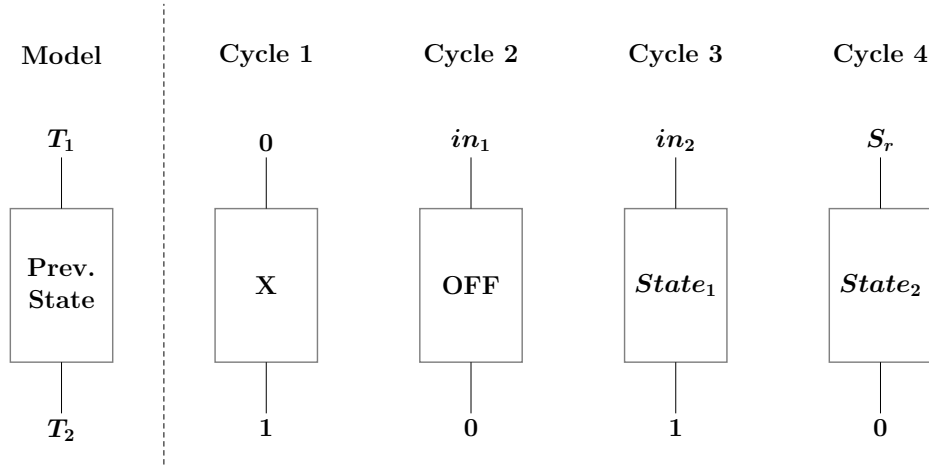


Figure 2.5: Diagram for the applied inputs through the cycles of sequential AND logic gate

Operation	Terminal	Cycle 1	Cycle 2	Cycle 3
AND	T_1	0	in_1	in_2
	T_2	1	0	1
in_2	in_1	$State_0$	$State_1$	$State_2$
0	0	OFF	No change	OFF
0	1	OFF	ON	OFF
1	0	OFF	No change	No change
1	1	OFF	ON	No change

Table 2.2: The truth table for sequential AND logic gate

2.2.2.3 Probabilistic Sequential Logic

Due to the stochastic behavior in the resistance switching of memristor, the sequential logic has been improved and combined with probability theory to deal with the probabilistic memristor. The differentiation between deterministic and probabilistic domains depends on the applied input voltage levels according to the threshold voltage. To guarantee switching of memristor to the correct states, the input voltage is applied above the threshold voltage in deterministic domain, as shown in [32], for a bipolar memristor which has a threshold voltage of 4.6V, the input voltage applied should be $\pm 5V$. On the other hand, the input voltage is applied below the threshold voltage in the probabilistic domain which causes uncertainty in switching. The switching probability $P_s(t)$ defines the probabilistic behavior of memristor. Integrating this behavior in the operation of sequential logic gates introduces errors into the output.

The proposed model for sequential logic in probabilistic domain is reviewed for this work [31]. There are some assumptions placed to ensure model analysis is defined. They are as follows: The starting cycle of the model is OFF state, the inputs in_1 and in_2 are deterministic, and only switching probability $P_s(t)$ that models the stochastic behavior. Thus, the accuracy of probabilistic sequential logic gate depends on the applied input

voltage through the cycles of the operation, and the switching probability of the memristor. The definition of accuracy is stated as getting the correct output for its corresponding input. For example, AND gate is precise when output is '1' for both inputs in_1 and in_2 are '1', and '0' otherwise. The measurement of any gate's accuracy which has total input combinations N , N_0 number of inputs to get a '0' output, and N_1 number of inputs to get a '1' output is determined using the following equation:

$$Accuracy(Gate) = \frac{N_0}{N} P_{out}(0) + \frac{N_1}{N} P_{out}(1) \quad (2.1)$$

2.2.2.4 Probabilistic AND Logic

The analysis diagram for probabilistic sequential AND gate is shown in figure 2.6. There are 3 events through the operation: (1) No change in case of applying zero-input voltage, (2) No switching which its applied input difference generates state same as the previous one, the probability of these events is 1, and (3) Switching which may occur with probability $P_s(t)$ or not occur with probability $1 - P_s(t)$. Similar to the characterization of AND gate, the analysis diagrams for probabilistic sequential OR, NAND, and NOR logic gates are presented in Appendix A.

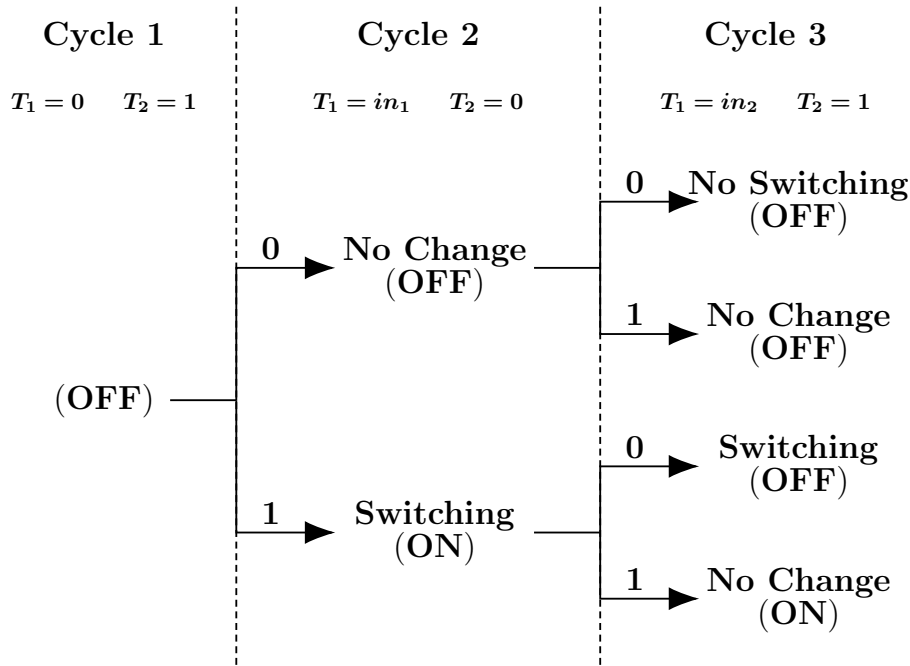


Figure 2.6: Analysis diagram for probabilistic sequential AND logic gate

The operation of probabilistic sequential AND logic gate is clarified by applying all inputs combination in the form (in_2, in_1) to the diagram in figure 2.5. Investigating the effect of all input combinations on the AND operation through different cycles as follows:

- The first output should be '0' for the input combination (0,0). Memristor is on the OFF state for the first cycle, the second cycle does not change the state of the memristor as in_1 is 0 which makes zero voltage difference across memristor's terminals, and similarly in the third cycle no switching will happen cause of in_2 is 0

so a negative voltage is applied. Thus, the probability to reach a correct '0' output is independent of switching:

$$P_{(0,0)}(0) = 1$$

- The second output for input combination (0,1) should be '0'. The first cycle initializes the OFF state, and since the difference-voltage inputs are zero on both the second and third cycles, there is no change for the resistance state. Thus, the probability to reach a correct '0' output for input combination (0,1) is as (0,0):

$$P_{(0,1)}(0) = 1$$

- The correct output for the third input combination (1,0) should be '0'. After the OFF state, there are two scenarios to get the correct output: (1) If the memristor switches in the last cycle $P_s(t)$, or (2) it does not switch in the second cycle $1 - P_s(t)$ and third cycle $1 - P_s(t)$. Hence the probability to get a correct '0':

$$P_{(1,0)}(0) = P_s(t) + (1 - P_s(t))(1 - P_s(t))$$

$$P_{(1,0)}(0) = 1 - P_s(t) + P_s^2(t).$$

- The last input combination (1,1) should switch the output to ON state and get a logic '1'. The switching will happen only in the second cycle so the probability to get a correct output '1' is:

$$P_{(1,1)}(1) = P_s(t)$$

- Adding all of these probabilities to get the final output probability for sequential AND gate, as: $P_{out}(0)$ which is the probability to get a correct '0' output, and $P_{out}(1)$ which is the probability to get a correct '1' output for all input combinations as follows:

$$P_{out}(0) = \frac{1}{3}(3 - P_s(t) + P_s^2(t))$$

$$P_{out}(1) = P_s(t).$$

- The measurement of AND gate's accuracy is determined using equation 2.1 as follow:

$$Accuracy(AND) = \frac{1}{4}(3 + P_s^2(t))$$

The final probabilistic behaviors of different sequential logic gates are presented in table 2.3 [31]. As shown, AND and NOR gates have the same probabilistic behaviors. Also, OR and NAND are typical in their behaviors. There is higher accuracy for AND and NOR which have '0' output in 3 out of 4 input combinations cause of the initial state is OFF which makes the memristor have higher probability to get '0' rather than '1'.

Gate	$P(0)$	$P(1)$	Accuracy
AND	$\frac{1}{3}(3 - P_s(t) + P_s^2(t))$	$P_s(t)$	$\frac{1}{4}(3 + P_s^2(t))$
OR	1	$\frac{1}{3}(4P_s(t) - P_s^2(t))$	$\frac{1}{4}(1 + 4P_s(t) - P_s^2(t))$
NAND	1	$\frac{1}{3}(4P_s(t) - P_s^2(t))$	$\frac{1}{4}(1 + 4P_s(t) - P_s^2(t))$
NOR	$\frac{1}{3}(3 - P_s(t) + P_s^2(t))$	$P_s(t)$	$\frac{1}{4}(3 + P_s^2(t))$

Table 2.3: $P(0)$, $P(1)$, and accuracy of sequential AND, OR, NAND, and NOR logic gates based on stochastic memristors

2.3 Applications Overview

This section investigates different applications can be utilized using imprecise computing: addition operation and generation of random numbers. Two implementations for imprecise adders are introduced. Also, an overview about random number generator types and implementation is presented later.

2.3.1 Adder

Adder is the most important arithmetic component in soft computing systems which is sometimes known as computational intelligence (CI). This computing systems depends on that many applications can accept degree of uncertainty, and tolerate loss of precision in their arithmetic outputs without effecting the system performance . These features are the main differences between imprecise and conventional computing.

Different techniques have been proposed to model approximate adders. For example, voltage over-scaling (VOS) for CMOS circuits [33–35], this technique reduces the supply voltage to achieve low-power consumption. Another method relies on redesigning the circuits by removing some of their components, as approximate mirror adder (AMA) which is based on removing some of its transistor as proposed in [33] for three different implementation. Also, paper [36] presents pass transistors utilized to implement three adders using the implementation of approximate XOR/XNOR gates with multiplexers. Moreover, transmission gates (TGs) are used as a component replacement for pass transistors to propose two multiplexer-based approximate adders in [37]. Figure 2.7 shows an exact ripple-carry adder (RCA) which is a common conventional full adder (CFA) which is implemented by cascaded full adder components.

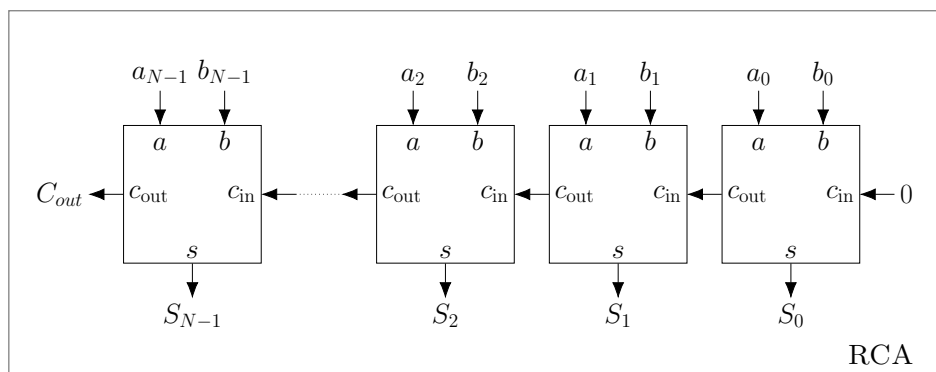


Figure 2.7: Block diagram for simple ripple carry adder

Imprecise adders can be categorized into run-time and design-time techniques which are useful for biomedical applications [38]. These techniques are so promising for low-power consumption applications. They are based on the operation of: (1) deterministic approximate logic, as replacing XOR gate with OR gate for the least significant bits (LSBs) of the adder which is known as bio-inspired Lower-part OR Adder (LOA), or (2) imprecise arithmetic, as Probabilistic Full Adder (PFA) which is implemented using nanometric devices (Probabilistic CMOS). A review for these papers [39, 40] about different conventional and soft adders is introduced in this section. Error Distance (ED) and Mean Error Distance (MED) are metrics used in these papers [36, 39] to assess

reliability of the proposed adders. Paper [39] compares between different approximate and probabilistic adder implementations as AMA, LOA and PFA which will be reviewed.

2.3.1.1 Lower-bit OR Adder (LOA)

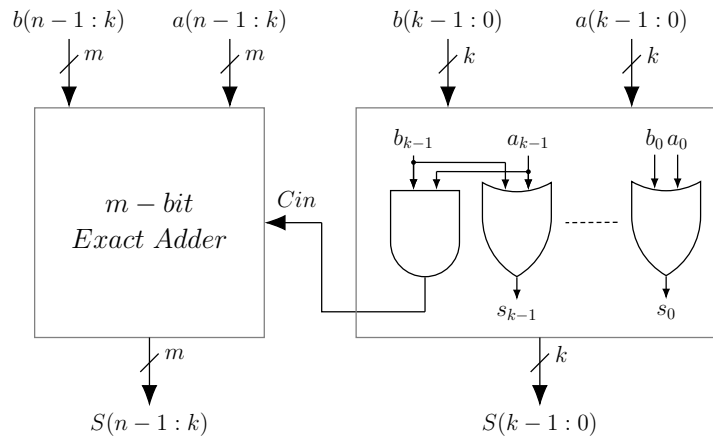


Figure 2.8: Block Diagram of LOA

The hardware structure of LOA is shown in Figure 2.8. LOA adder with size n-bits is divided into two smaller adders:

- Exact m-bits Adder for the most significant bits (MSBs) to compute the precise calculations for higher part.
- Approximate k-bits adder for the least significant bits (LSBs) using OR gates rather than XOR gates, with an additional AND gate for the MSB in the lower part to generate carry in signal (C_{in}) for the higher part.

2.3.1.2 Probabilistic Full Adder (PFA)

The hardware structure of PFA is same as LOA as shown in Figure 2.9. The difference is in the implementation of lower part, as PFA is used probabilistic CMOS (PCMOS) to implement it. PCMOS is a nanoscale device which can achieve low-power consumption. Table 2.4 shows a the elements, advantages and disadvantages between LOA and PFA.

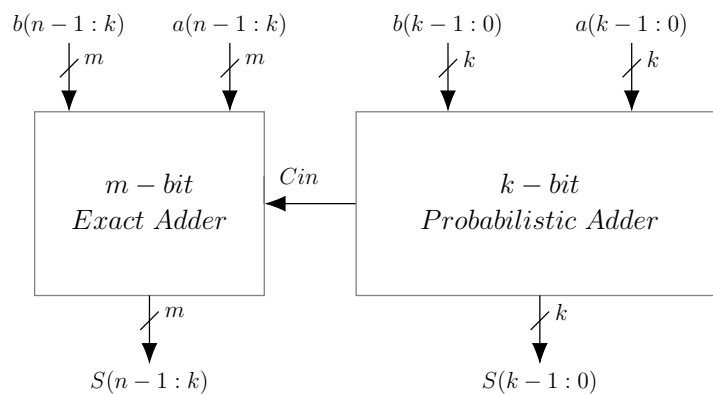


Figure 2.9: Block Diagram of PFA

	LOA	PFA
Element	Approximate Logic	Probabilistic CMOS
Advantages	Balance precision with other performance metrics A good trade-off between reliability, area, power	Balance performance at nanometric scales Better reliability at a small gate error rate
Drawbacks	Loss of precision cause of ignoring the normal carries in the LSBs	Larger overhead in area and static power consumption

Table 2.4: Comparison between LOA and PFA

2.3.2 Random Number Generators

Random numbers are critical elements for an entire scope of applications, including: cryptography, gaming, computer simulation and statistical sampling. The generators of these numbers are required to produce secure long sequences of random values. An investigation for different types of random number generators is presented. Also, a study for more randomness generator is clarified in chapter 5. There are basically two sorts of random number generators - true random number generators (TRNGs) and pseudo random number generators (PRNGs). The main contrast between TRNGs and PRNGs is that: TRNGs sample a source of entropy, in other words, they utilize unpredictable physical means to produce numbers. On the other side, PRNGs use deterministic mathematical algorithms which are totally computer-generated to create the numbers. A brief summary of TRNGs and PRNGs is explained below.

2.3.2.1 True Random Number Generators

A true random number generator needs a normally happening source of randomness, as: entropy, to produce arbitrary numbers. It tests this source of entropy and procedures it through a computer to create an arrangement of random numbers. The entropy source regularly comprises of some physical amount, for example, the noise in an electrical circuit (e.g., variance in fan noise), the quantum consequences in a semiconductor, or the timing of user processes (e.g., mouse movements). Different generators of these sources may be used. A TRNG generates randomness introduced in the fundamental physical source and the state of a TRNG is independent of the previous states which makes the prediction of generated random sequence impossible. Three elements are the main components of a basic TRNG circuits, the source of randomness, the extraction circuit (sampler) and the post-processing unit [41], as shown in figure 2.10.

1. **Random source:** is the main component as it decides the randomness of TRNG. Various sources are introduced, for example, clock jitter [42, 43], thermal noise [44, 45], and metastability of the circuit [42]. The random source is mainly an analog circuit that produces analog signal which has a random property inside as the white noise.
2. **The extraction circuit (Sampler):** As the random source generates an analog signal, there is a need for sampler to the generated signal into random number binary sequences. There are different sampler methods, as, using a D flipflop to sample

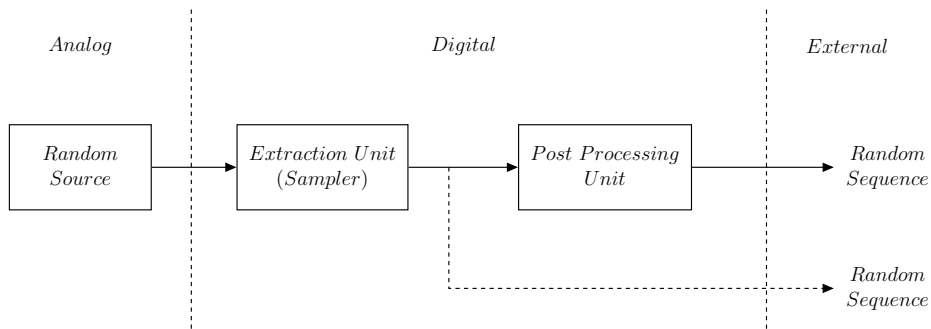


Figure 2.10: Block Diagram of TRNG

the signal generated from random source as clock jitter. Anyway, it is required to gather the oscillation time of the random source by a counter. The extraction circuit impacts the speed of TRNG, as generating higher bit rates will need to use higher sample frequency which can decrease the randomness of TRNG. This is the reason that TRNG is slower than PRNG.

3. **Post-processing unit:** It is a technique to pack the output sequence after sampler unit. It will increase the strength of TRNG but also decrease the output bitrate. This part is not essential for TRNG as sometimes the random source is powerful enough. However, the output of random source may have some unsatisfactory properties, as the proportion of zeros and ones in the sequence. Using post-processing unit can eliminate these bad properties and provide great randomness in the generated pattern. There are different techniques for post-processing unit such as XOR trees [46], and Von Neumann Extractor [47, 48]

2.3.2.2 Pseudo Random Number Generators

The second technique utilizes computational calculations that can create long successions of obviously arbitrary outcomes, which are in actuality totally controlled by an introductory value, known as a seed or initial key. Thus, the whole apparently random string can be duplicated if the seed is known. This kind of arbitrary number generator is frequently called a pseudo random number generator, also known as a deterministic random bit generator (DRBG). PRNG commonly does not depend on origin of normally happening entropy, however it might be occasionally seeded by characteristic sources.

The periodicity of generated sequence is another important feature in PRNG. As once the initial key (seed) is fixed, the output sequence will repeat after specific periods. For example, n-bits PRNG will generate its random pattern after period $2^n - 1$. Also, the seed selection is so important for the period's length, as it can be so short if the initial seed isn't suitable for the used implementation of PRNG (such seeds may be called weak). Linear congruential generator (LCG) [49], Lagged Fibonacci generator (LFB or LFib), and Linear feedback shift register (LFSR) are different techniques to implement PRNG.

2.3.2.3 Comparison between TRNG and PRNG

In looking at TRNGs and PRNGs, each have their benefits and drawbacks which are summarized in table 2.5.

Implementation	TRNGs	PRNGs
Sequence	Non-deterministic	Deterministic
Complexity	Complex	Simple
Efficiency	Low	Excellent
Speed	Slow	Fast
Cost	Expensive	Cheap

Table 2.5: Comparison between true and pseudo random number generators

There are many imprecise computing techniques which show a promising performance in trade off between accuracy and power consumption. The source of error can be modeled in different levels from devices till system architecture. A review for Stochastic computing features and model was clarified to show the source of error in this imprecise computing technique. Also, devices which have probabilistic error behavior in their states were introduced, as memristor and PCMOs which can be classified into probabilistic computing. Moreover, a study for different applications was presented as approximate and probabilistic adder techniques which will be analyzed in chapter 4.

Finally, a review on different random number generator was discussed as programmable LFSR technique will be introduced and analyzed later. Programmable LFSR is the main block to implement pseudo random number generator with different orders based on specific polynomials. Using probabilistic AND, XOR logic gate models which will be proposed in the next chapter to replace the conventional gates in this programmable LFSR is an interest of thesis work. The generated sequences from the LFSR model with these imprecise gates will have a characteristic of true random patterns which will make the modified LFSR worked as a Quasi-TRNG generator. More details about the analysis is provided in chapter 5.

Chapter 3

Inexact Gates Modeling

The models of approximate and probabilistic or Stochastic gates are introduced in this chapter to show the proposed method for error injection in the logic gate for both cases. Figure 3.1 shows these different inexact computing methodologies with paradigms for them to distinguish between the source of error in each computing technique. For instance, a lower-bit OR adder (LOA) represents the approximate computing on the level of system architecture for adders. The probabilistic switching behavior in PCMOs is an example for the devices level of probabilistic computing. Also, a stochastic multiplier with inputs and output mapped into probabilities is a model for stochastic computing.

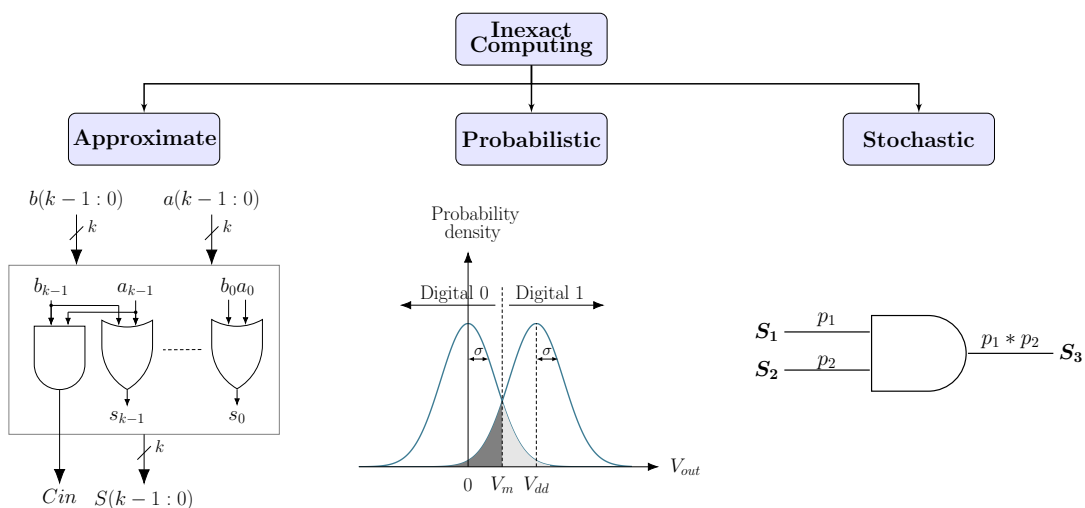


Figure 3.1: Classification and paradigm for inexact computing techniques

The proposed models are applied on various simple logic gates to prove the validity of error injection method by calculating a simple metric referred to as output percentage error (OPE) which is similar to error distance (ED) metric proposed in [50]. OPE relates outputs of N random inputs for the imprecise gate to the relative outputs from the accurate gate, then accumulates the number of differences between precise and imprecise outputs, and calculates the percentage error as clarified in figure 3.2. The main usage for OPE is being an indicator for the behavior of any gate's output in case of injecting it by a probability distribution function or deterministic error which models the injected error inside this gate for either probabilistic or approximate computing respectively. In general, OPE for any compared outputs is given by:

$$OPE = \left(\frac{\sum_{i=0}^N |out(i)_{acc} - out(i)_{imprecise}|}{N} \right) * 100\% \quad (3.1)$$

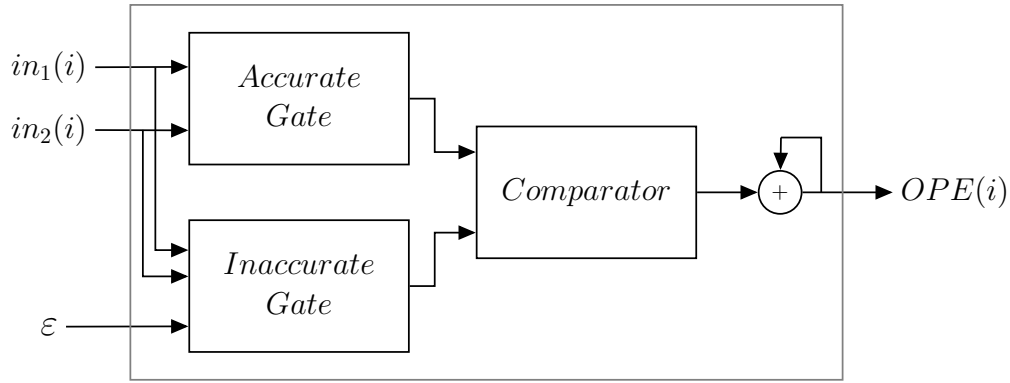


Figure 3.2: Output percentage error model

In figure 3.2, ϵ is the metric to model gate error which depends on the type of the injected data. It could be correlated or uncorrelated to the inputs of gate. Also, ϵ can be dependent or independent on the gates' type. For approximate mode, the injected error is dependent on inputs combinations, but it is independent on gate type and uncorrelated to input data in probabilistic model. The error in probabilistic gate is modeled as a probability function as in PCMOS, it is modeled as the intersection between two normal distribution which are the logical representation for logic '0', and '1'. The probability distribution function (PDF) shows the number of times a specific discrete probability value p_i of the random variable A occurs. On the other side, CDF is the cumulative distribution function $F(x)$ that shows the probability of random variable being less than a specified value x . From the definitions of PDF and CDF, the distribution of any device error can be represented as a PDF, and accordingly, the gate's OPE of gates using single device to operate is supposed to be same as the CDF.

3.1 Approximate Gate

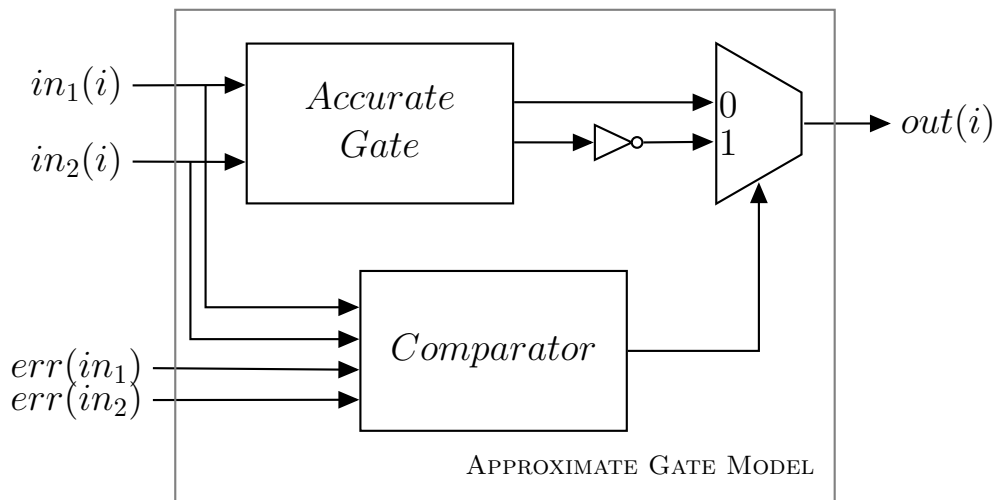


Figure 3.3: The model for approximate logic gate

The source of error in any approximate logic gate is deterministic which means for specific inputs the gate will get a wrong output. It can be as a result of the system or transistor level. For example, using OR gates rather than XOR gates in least significant bits (LSBs) of n-bit adder, known as Lower-Part-OR Adder (LOA) [8], is a common method for injecting a deterministic error to balance precision with other performance metrics such as power consumption as in biomedical applications. For one of these LSBs, If we apply two independent streams for the inputs in_1, in_2 with uniformly distributed combinations of 00, 01, 10, and 11, the output is supposed to have a fixed probability of error $Pe = 0.25$ for sufficient N samples. Also, logic reduction at transistor level to get a lower power consumptions, as in the approximate mirror adder (AMA) [51], is another source of error for the approximate gate. Fig. 3.3 shows a model for the approximate logic gate.

For 2-input logic gate, the error is injected by comparing the desired $err(in_1), err(in_2)$ with the time space gates' inputs $in_1(i), in_2(i)$ respectively. If both values are equal the gate will generate incorrect output for the inputs. Figure 3.4 presents the flow chart of error injection in approximate gate model.

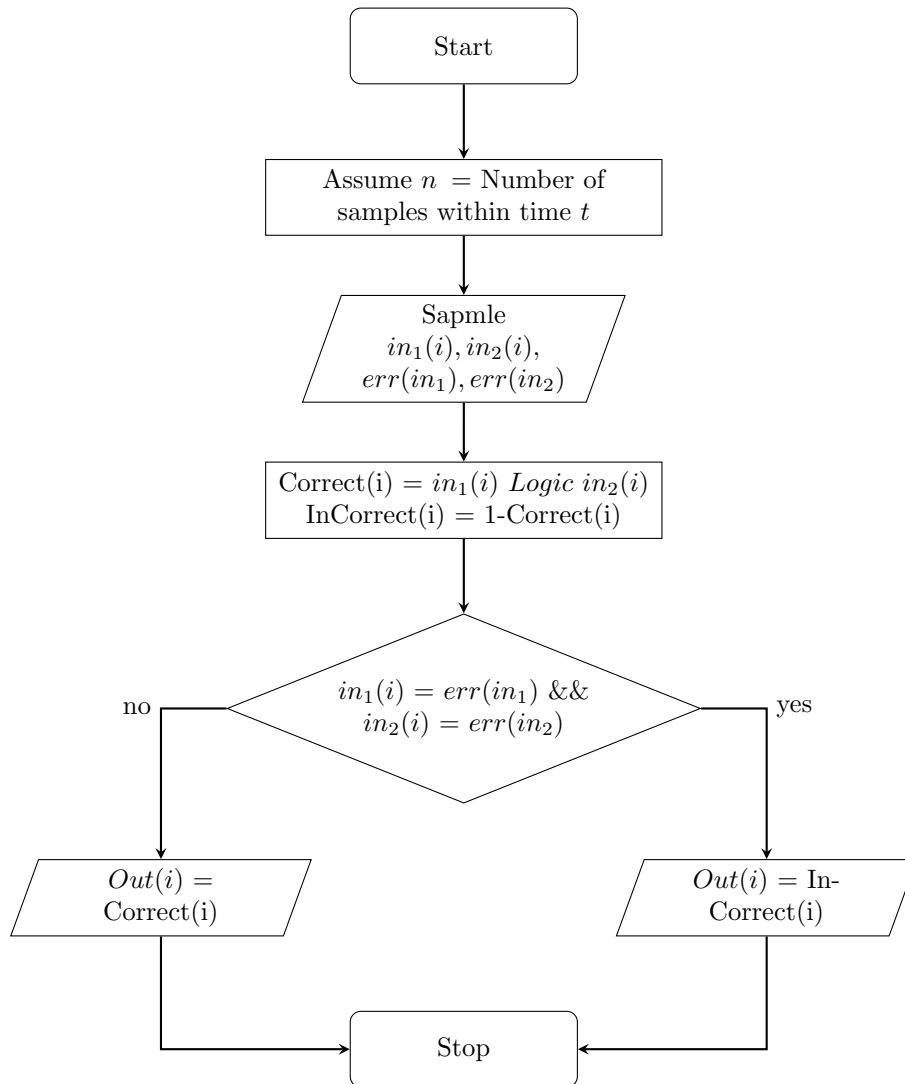


Figure 3.4: The flowchart for approximate logic gate

The injected error is function of the inputs combination; The location of error in logic gate is modeled by the terms $err(in_1)$, $err(in_2)$. For instance, The error can be located for $in_1 = 0$ and $in_2 = 0$ so the output will be 1 as shown in table 3.1 which presents the truth table for OR gate and its approximate with injected error for different inputs. The results can be clarified, as:

- Error in inputs 00 can be modeled as stuck at 1 for OR gate.
- Detection of odd numbers in case of approximate OR gate with error in 10.
- Faults case of 11 is so useful to generate the correct outputs of XOR gate with approximate OR gate implementation which leads to reduce power consumption.

in_2	in_1	Acc	Approximate			
			00	01	10	11
0	0	0	①	0	0	0
0	1	1	1	①	1	1
1	0	1	1	1	①	1
1	1	1	1	1	1	①

Table 3.1: Truth table for conventional OR gate and its approximate implementations with different errors

3.2 Probabilistic Gate

The proposed Probabilistic gate model in figure 3.5 is an abstract model and includes a novel contribution in the error injection which is non-deterministic. This error is as a result of the physical behavior of the implementing device, such as transistor in small scaling, or memristor. There is an assumption that the probabilistic behavior of the device can be modeled as a probability distribution function that is between $[0, 1]$. Actually the proposed model is analogous to Probabilistic computational models (SCMs) that is proposed in [52] with a difference of the error meaning in the model.

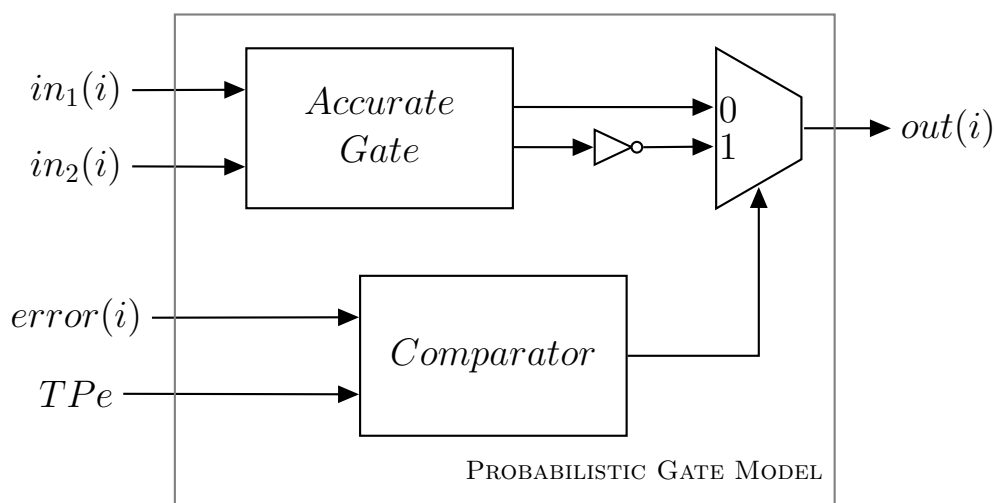


Figure 3.5: The model for probabilistic logic gate

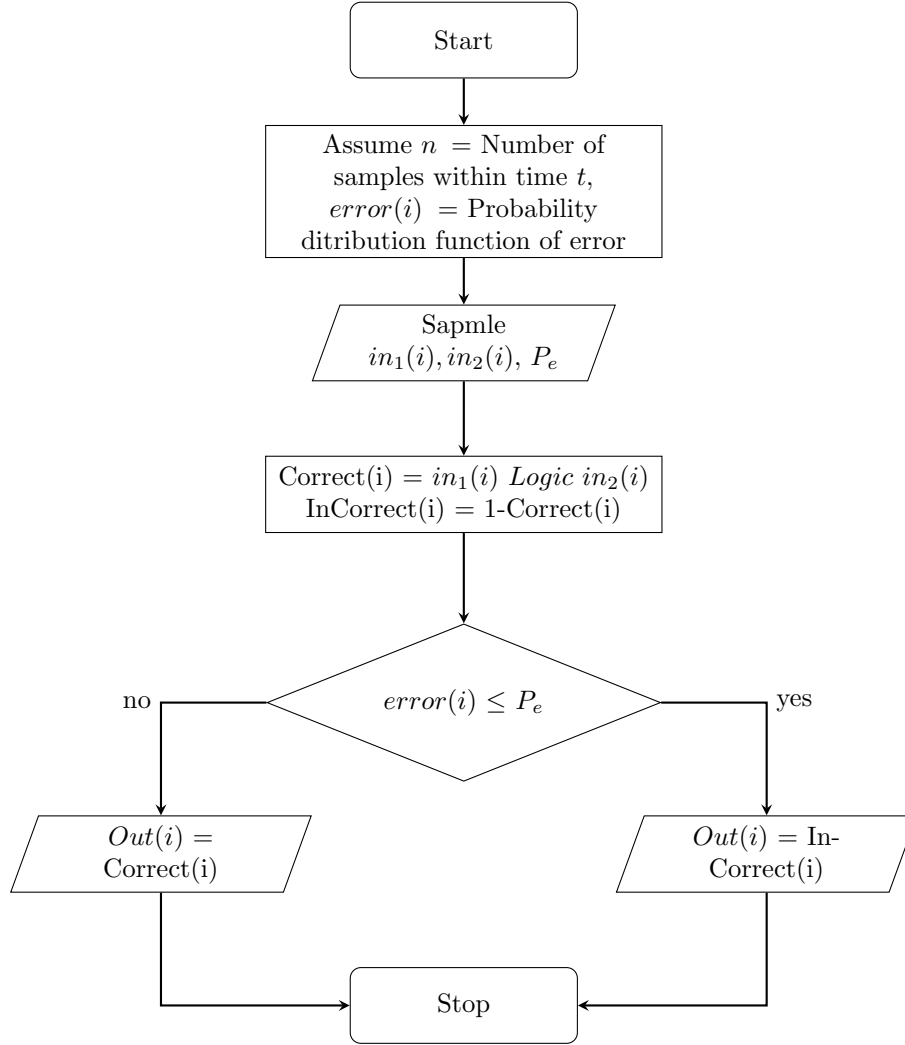


Figure 3.6: The flowchart for probabilistic logic gate

Within a specific time t , the logic gate has a finite number of inputs and outputs N . According to the error distribution and the probability of error, the model forces gate's output to be correct or incorrect. The model consists of three main components:

1. Accurate gate which defines the main required functionality of the model based on its inputs.
2. Comparator between the desired probability of error in the model and the estimated physical noise PDF. The result of comparator depends on the probability distribution function of the noise.
3. Multiplexer selects between result of the accurate gate or its inversion based on the output of the comparator.

Unlike the traditional stochastic approaches that use the numbers which are represented as a bit-streams and process these numbers as probabilities [53], the source of error in our probabilistic gate model (ϵ) is a function of *error* and *Target Probability Error (TPE)* which are characteristics of the physical error of the device. Term *error* declares the

probability distribution function of the device physical error, and TPE is related to the target number of outputs forced to be incorrect for a set of inputs. Therefore, OPE of logic circuit will depend on the injected distribution. Table 3.2 shows the effect of ϵ when it is 0 and 1 on different gates.

	Inputs			AND	OR	XOR
	ϵ	B	A			
Accurate	0	0	0	0	0	0
	0	0	1	0	1	1
	0	1	0	0	1	1
	0	1	1	1	1	0
Probabilistic	1	0	0	1	1	1
	1	0	1	1	0	0
	1	1	0	1	0	0
	1	1	1	0	0	1

Table 3.2: Truth table for AND, OR, XOR logic gates with their $P_e = 0$ and 1

It is clear that in case of P_e is '0' the gate will behave normally and its functionality will be correct. In case of P_e is '1' for all inputs combination, it will behave normally too but with inverted functionality. ϵ may be 1 or 0 through the time space which will affect different input combinations with various probability. Figure 3.6 clarifies the model.

3.3 Simple and Complex Gates

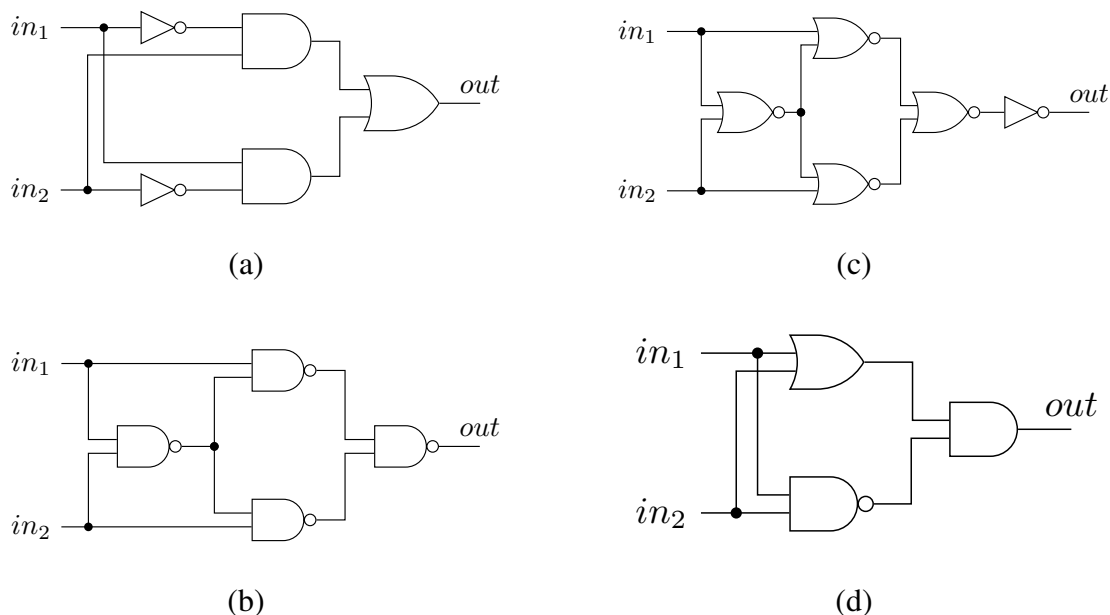


Figure 3.7: Different Topologies for XOR gate

In this section, there is an assumption that there is a device that can assemble all different simple logic gates, as: NOT, AND, OR, NAND, NOR, ... etc by arranging

components in different architecture. Then, applying the approximate and Probabilistic models on these gates to investigate the effect of incorrect outputs in the gate's functionality. Four different XOR topologies are investigated to check the effect of simple gates from aspects as: gates count, structure, and logic levels on complex XOR gate. Figure 3.7 and table 3.3 show these topologies with their characteristics.

Topology	Gate Counts	Structure	levels
a	5	Symmetric	3
b	4	Symmetric	3
c	5	Symmetric	4
d	3	Non-Symmetric	2

Table 3.3: Characteristics of XOR topologies

3.4 Results

In this section, the proposed models are assessed for approximate and probabilistic gates using Matlab. There are two sections of results: (1) Simple and Complex gates based on approximate model, and (2) Simple and Complex gates based on probabilistic model, Six simple logic gates are used in the evaluation of models (AND, NAND, OR, NOR, NOT, and XOR).

3.4.1 Approximate Gates

For approximate gates, two independent random inputs are applied to simple and complex gates. The error is injected as a combination of the inputs representations to prove the statistical properties of approximate gate, as: the result will be the same over time.

3.4.1.1 Simple Approximate Gates

Injecting error in simple approximate gates by applying $err(in_1)$ and $err(in_2)$ to 00, 01, 10, and 11 is the first technique to prove the validity of approximate gate model. Figure 3.8 shows that OPE of simple gates is dependent only on the source of error which represents one of inputs combination. In this case, input combination is selected. OPE is proved to be fixed over time so 25% of outputs will be incorrect as the error is in only one combination and independent on P_e . The error could be in any numbers of inputs representation which leads to change OPE.

3.4.1.2 Complex Approximate Gates

Investigating the results in figure 3.9 for complex approximate XOR gates shows that:

- Blackbox topology is a representation for XOR gate in case it can be implemented as a simple gate which is the reference to measure the complex approximate gate performance.
- OPE in gates with source of error in 00 for topologies 3 and 4 is 0% due to the effect of NAND gate as in topology 3 the error will propagate in each NAND gate, but on

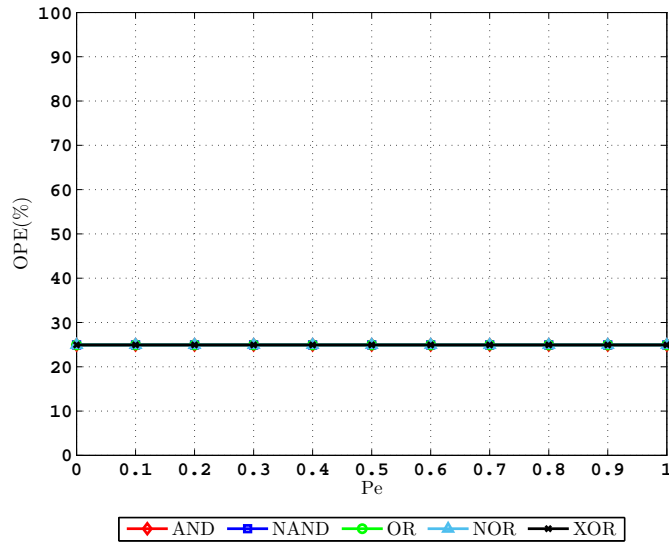


Figure 3.8: OPE for AND, NAND, OR, NOR, and XOR with error in input combination "00"

topology 4 OR gate will be always '1' which prevents the error occurrence in AND gate.

- For the gates with error in 10, 01, and 11, *OPE* is 50% which declares that half of the samples will be wrong.

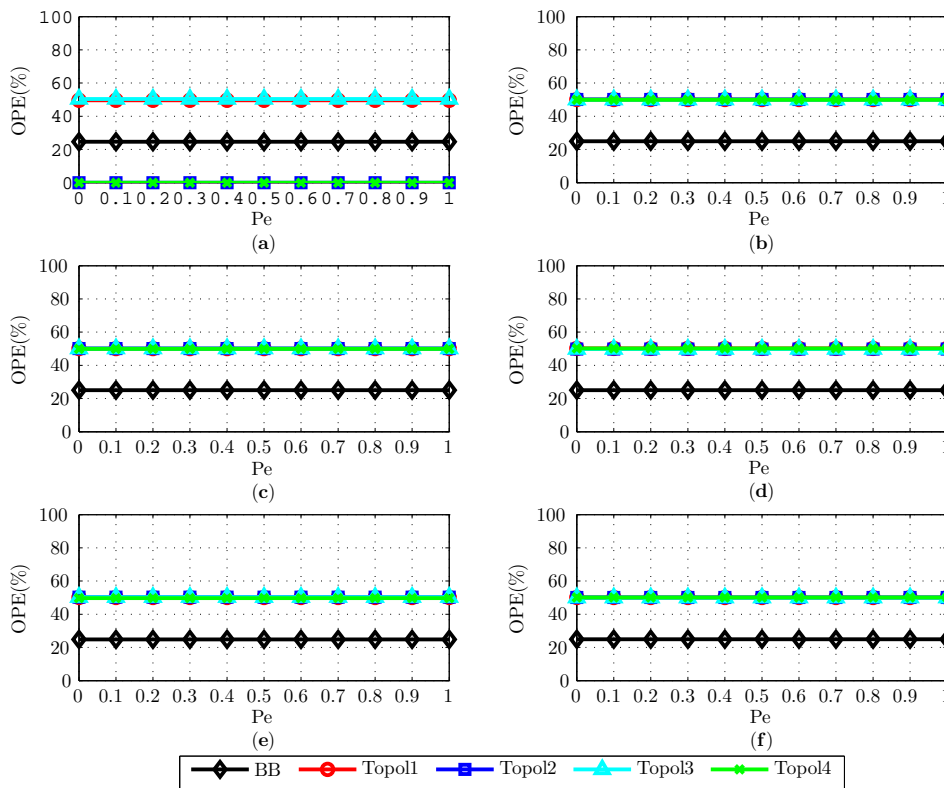


Figure 3.9: OPE for different XOR topologies with error in gates input combinations: (a) "00", (b) "01", (c) "10", (d) "11", (e) fixed for gates of same type, and (f) random for different gates

3.4.2 Probabilistic Gates

We apply uniform and normal distributions by generating sufficient number of random data and fit probability distribution function to these data. As there is no control on the physical error in the real device, we sweep P_e from [0 : 1] to get the corresponding OPE. The generated random data are fitted to its corresponding probability distribution function. Then, OPE is calculated over P_e range to check the effect of added noise on simple and complex gates. OPE is compared with the cumulative distribution function.

3.4.2.1 Simple Probabilistic Gates

For any Probabilistic gates, $P(0)$ is the probability of correct '0' output, and $P(1)$ is the probability of correct '1' output. The results of OPE are found to be identical with CDFs of fitted PDFs for the injected noise either as uniform or normal distribution as shown in figure 3.10. This proof is very useful to check the effect of different noise distributions in less reliable components in complex systems. OPE equation 3.1 can be re-written for N input samples as:

$$OPE = \left(\sum_{i \leq P_e} \frac{|error(i)|}{N} \right) * 100\% \quad (3.2)$$

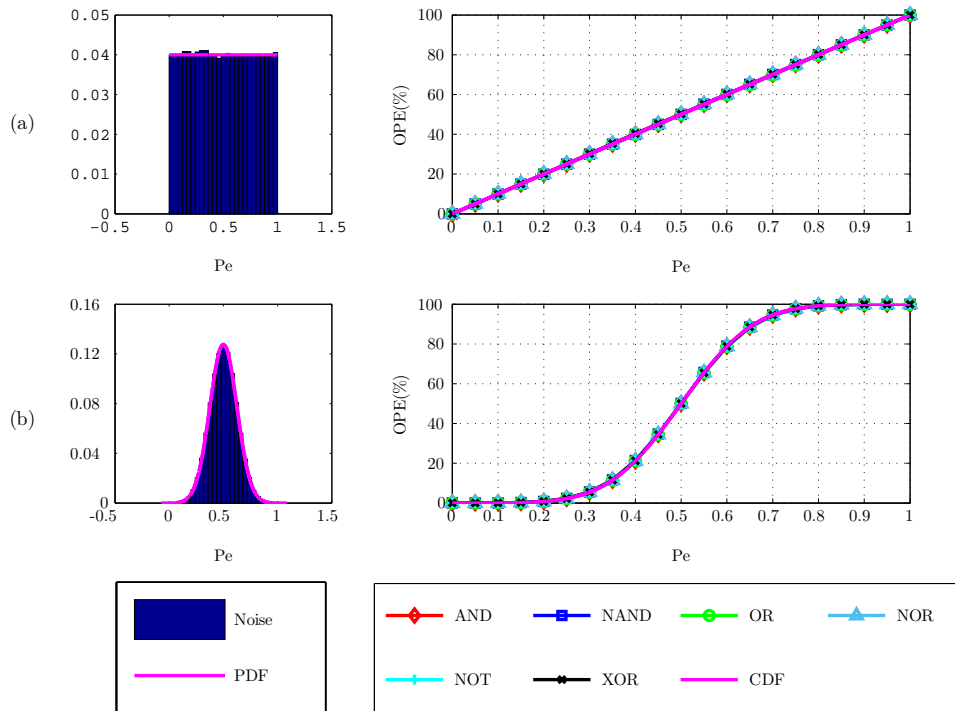


Figure 3.10: OPE for simple gates, and CDF of the injected noise (on the right), and histogram of the injected noise with the fitted PDF of it (on the left) are plotted for: (a) uniform, and (b) normal distributions

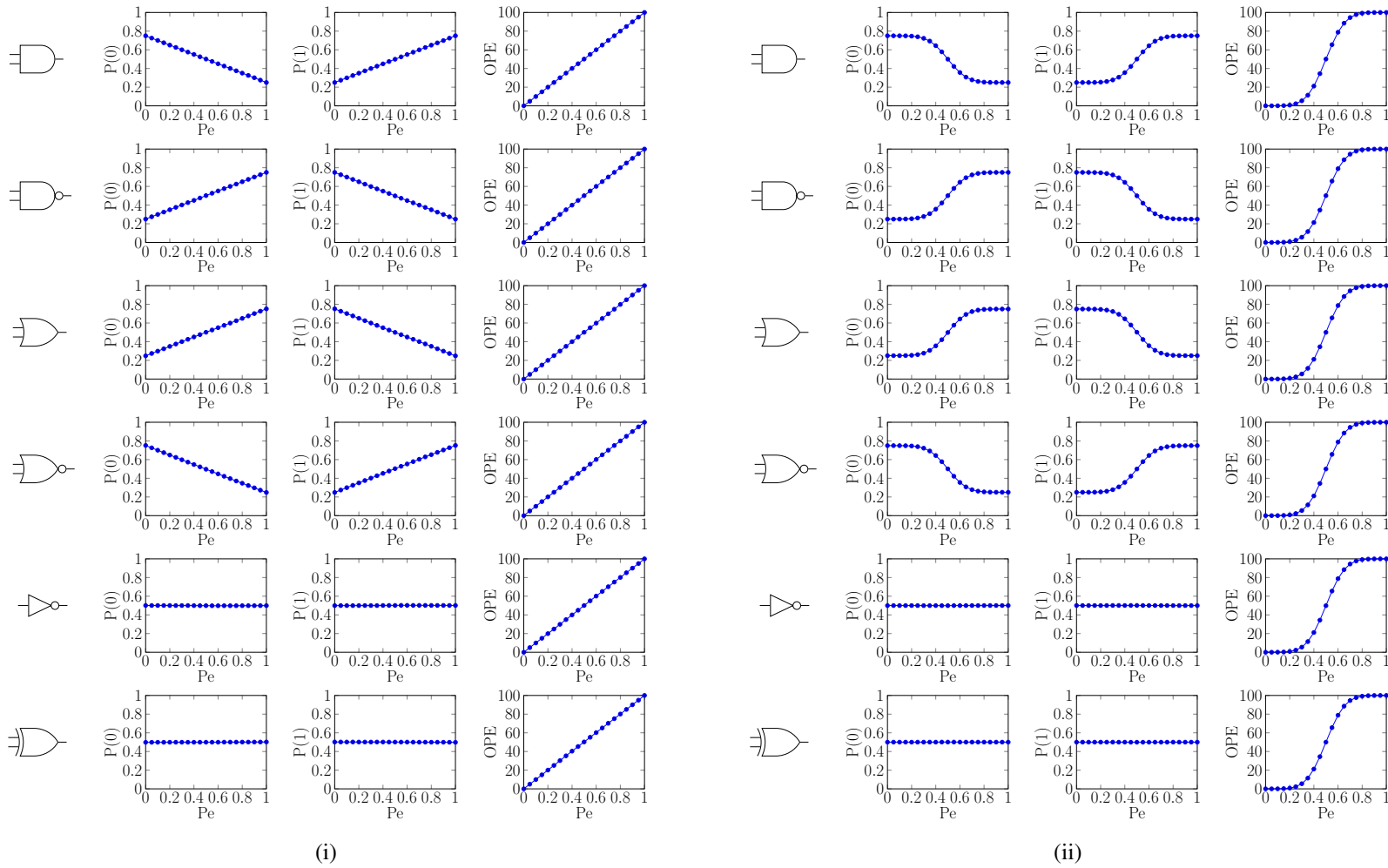


Figure 3.11: $P(0)$, $P(1)$, and OPE for each simple gate for: (a) uniform, and (b) normal distributions

For any simple gate, applying equally weighted input combinations on the input will make output $P(0)$ and $P(1)$ fixed as in the truth table of this gate. Figures 3.11i, and 3.11ii show $P(0)$, $P(1)$, and OPE for different simple gates for uniform and normal error distribution respectively, investigating the graphs shows that:

- $P(0) + P(1) = 1$ for any Pe .
- $P(0|Pe = 0) = P(1|Pe = 1)$, $P(0|Pe = 1) = P(1|Pe = 0)$, $OPE(Pe = 0) = 0\%$, and $OPE(Pe = 1) = 100\%$ for same input combinations which means that simple gates behave in normal mode when $Pe = 0$ and inverted when $Pe = 1$.
- OPE for simple gates is independent of the functionality of these gates, as it is only effected by the noise distribution in the gate
- De Morgan's theorems are valid on the simple gates as shown in (AND, NOR), (NAND, OR) $\overline{P(0) + P(1)} = \overline{P(0)}.\overline{P(1)}$, $\overline{P(0).P(1)} = \overline{P(0)} + \overline{P(1)}$
- Tables 3.4 and 3.5 show the numerical values for the output percentage error to each simple gate in the full range for probability of error.

Pe	AND	NAND	OR	NOR	NOT	XOR
0	0	0	0	0	0	0
0.1	9.96	9.96	9.98	10.00	9.94	9.97
0.2	20.03	19.98	20.01	20.09	19.92	20.09
0.3	30.07	30.03	30.00	30.09	29.89	30.25
0.4	40.06	39.94	39.99	40.06	39.93	40.38
0.5	50.07	50.07	50.07	50.09	50.05	50.21
0.6	60.07	60.06	59.90	60.13	60.13	60.22
0.7	70.08	70.06	69.84	70.08	70.14	70.20
0.8	80.01	80.02	79.87	80.01	80.14	80.10
0.9	89.94	90.01	89.96	90.02	90.04	90.01
1	100	100	100	100	100	100

Table 3.4: OPE in simple logic gates with error as uniform distribution

Pe	AND	NAND	OR	NOR	NOT	XOR
0	0	0	0	0	0	0
0.1	0.08	0.07	0.07	0.06	0.07	0.06
0.2	0.78	0.82	0.79	0.80	0.82	0.80
0.3	5.42	5.58	5.43	5.48	5.45	5.50
0.4	21.14	21.35	21.18	21.11	21.15	21.21
0.5	50.06	50.11	50.02	49.98	49.86	50.00
0.6	78.81	78.96	78.70	78.79	78.81	78.82
0.7	94.53	94.55	94.44	94.51	94.46	94.55
0.8	99.20	99.18	99.17	99.19	99.19	99.18
0.9	99.93	99.93	99.93	99.93	99.94	99.93
1	100	100	100	100	100	100

Table 3.5: OPE in simple logic gates with error as normal distribution

3.4.2.2 Complex Probabilistic Gates

The uniform and normal distributions of the injected noise are applied on the four XOR topologies to check the effect of simple unreliable gates on complex ones. The last gate relates its output probability to the input which is affected by the first gates and error distribution. Figure 3.12 shows different OPE for the XOR topologies, as follow:

- The results for complex Probabilistic xor gates show great improvement in OPE for Pe higher than 0.5 and little degradation in the performance for Pe less than 0.5.
- Third topology which has the largest level of gates and the highest count has the best OPE among other topologies cause of the last NOT gate in the constellation which has the great impact on OPE improvement as presented in figures 3.15i, 3.15ii.
- First and second topology has the same OPE despite the difference constellation as the identical OPE between each level in first topology with its analogous level in second topology as shown in figures 3.13i, 3.14i for uniform distribution, and 3.13ii, 3.14ii for normal distribution.
- All levels of gates in the XOR topologies have the same OPE . Topologies with even levels (3, 4) show better OPE rather than that with odd levels (1, 2).

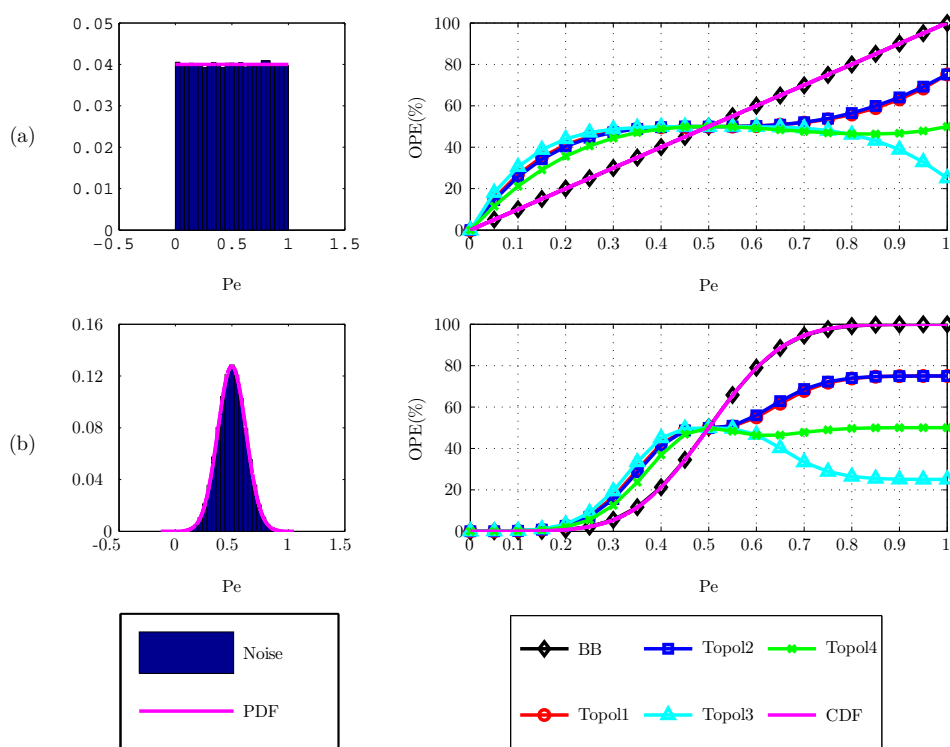


Figure 3.12: OPE for different XOR topologies, and CDF of the injected noise (on the right), and histogram of the injected noise with the fitted PDF of it (on the left) are plotted for: (a) uniform, and (b) normal distributions

<i>PE</i>	<i>XOR_{BB}</i>	<i>XOR_{Imp1}</i>	<i>XOR_{Imp2}</i>	<i>XOR_{Imp3}</i>	<i>XOR_{Imp4}</i>
0	0	0	0	0	0
0.1	10.03	26.92	25.76	30.52	21.22
0.2	20.06	41.18	40.39	43.99	35.68
0.3	29.95	47.53	47.20	48.69	44.45
0.4	40.05	49.61	49.67	49.88	48.86
0.5	50.01	49.92	50.00	49.93	50.08
0.6	60.03	50.21	50.26	49.82	49.22
0.7	69.91	51.88	52.06	49.23	47.65
0.8	80.01	55.77	56.47	46.12	46.47
0.9	90.06	63.07	64.09	38.90	46.76
1	100	75.13	75.13	25.06	49.93

Table 3.6: OPE in different complex XOR logic gate with error as uniform distribution

<i>PE</i>	<i>XOR_{BB}</i>	<i>XOR_{Imp1}</i>	<i>XOR_{Imp2}</i>	<i>XOR_{Imp3}</i>	<i>XOR_{Imp4}</i>
0	0	0	0	0	0
0.1	0.07	0.24	0.24	0.29	0.17
0.2	0.82	2.78	2.60	3.40	2.04
0.3	5.57	16.53	15.60	19.39	12.50
0.4	21.30	42.19	41.37	45.05	36.94
0.5	50.21	49.94	49.90	49.90	49.90
0.6	78.99	54.99	55.98	46.67	46.37
0.7	94.48	67.77	68.71	33.45	47.80
0.8	99.16	73.81	74.01	26.45	49.68
0.9	99.93	74.94	74.95	25.17	50.01
1	100	75.02	75.02	25.07	50.03

Table 3.7: OPE in different complex XOR logic gate with error as normal distribution

The simulation results in this chapter show the correctness of proposed imprecise models for either approximate or probabilistic gates as OPEs for the injected PDFs are shown to be as CDFs of the distributions for simple logic gates. A study on different complex XOR logic gates was provided to check the effect of error distribution in these topologies. Next chapters will discuss the development of using the simple and complex models in different applications, as: chapter 4 will present the reaction of these imprecise gates in chain of independent gates, and analyze the simulation results of imprecise adders. Also, these XOR models will be involved to generate random number sequence which have properties of true randomness in chapter 5.

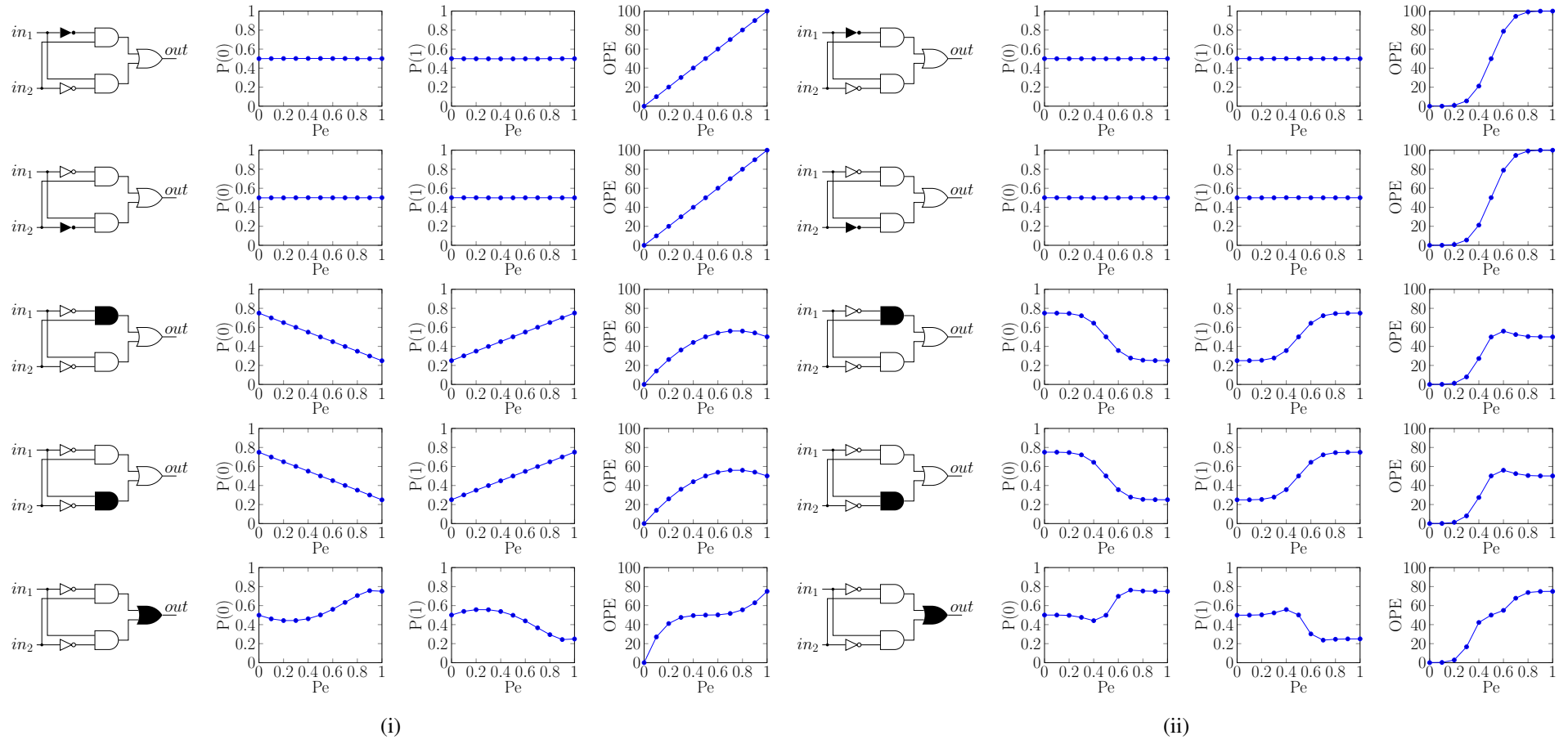


Figure 3.13: $P(0)$, $P(1)$, and OPE for each gate in XOR topology-i for error as distribution (a) Uniform, and (b) Normal

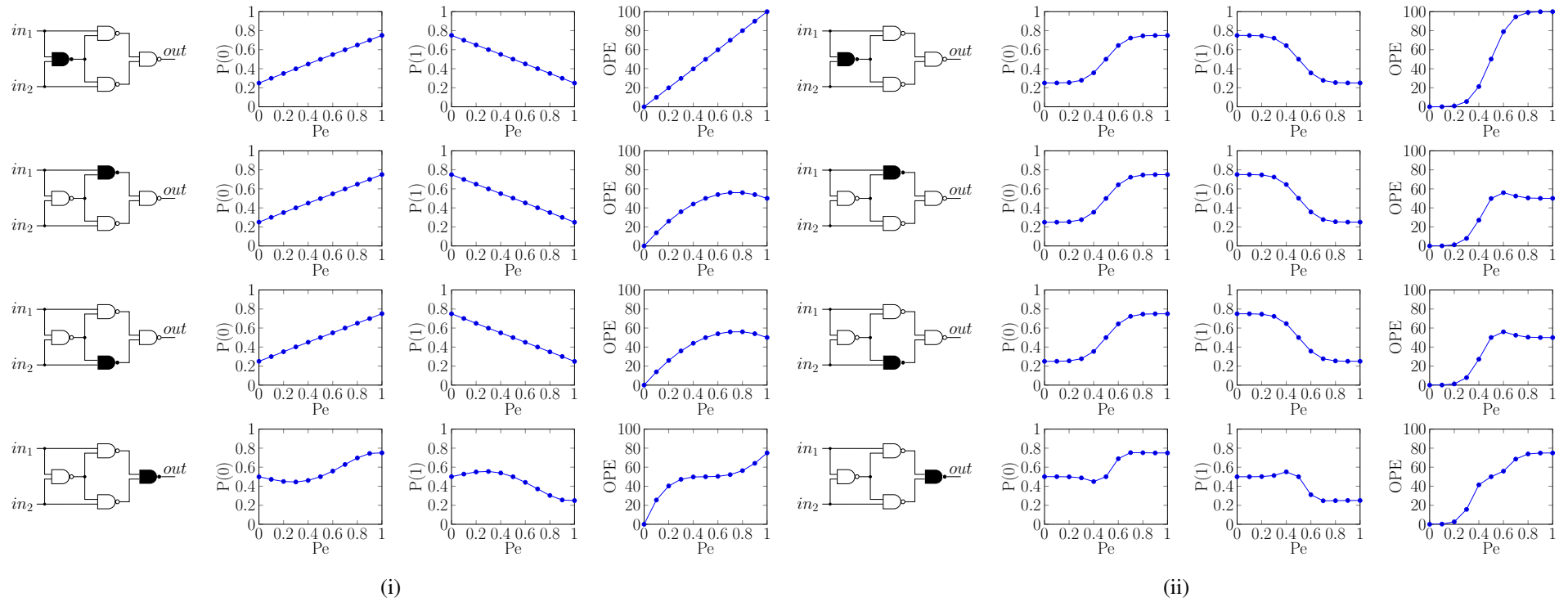


Figure 3.14: $P(0)$, $P(1)$, and OPE for each gate in XOR topology-ii for error distribution (a) Uniform, and (b) Normal

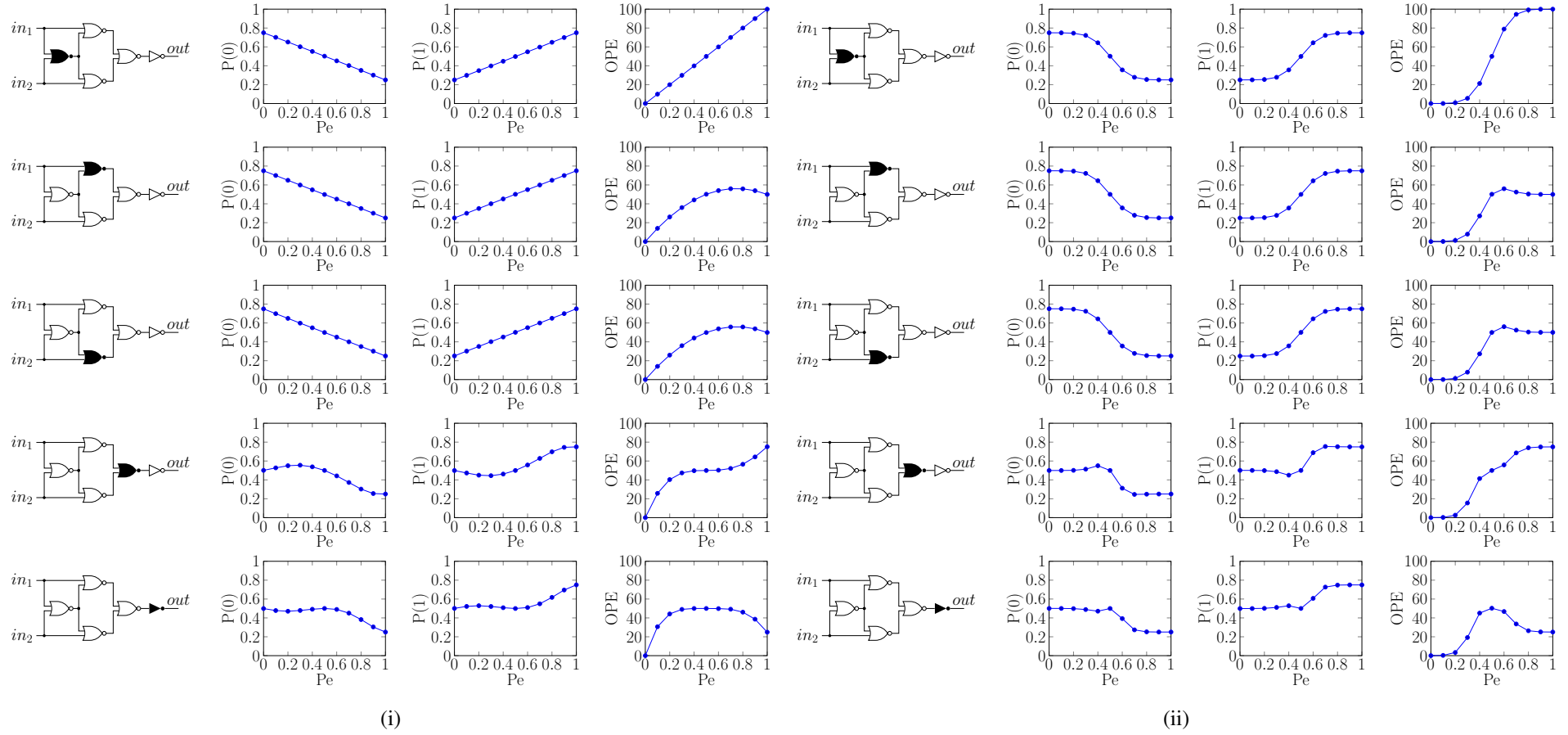


Figure 3.15: $P(0)$, $P(1)$, and OPE for each gate in XOR topology-iii for error distribution (a) Uniform, and (b) Normal

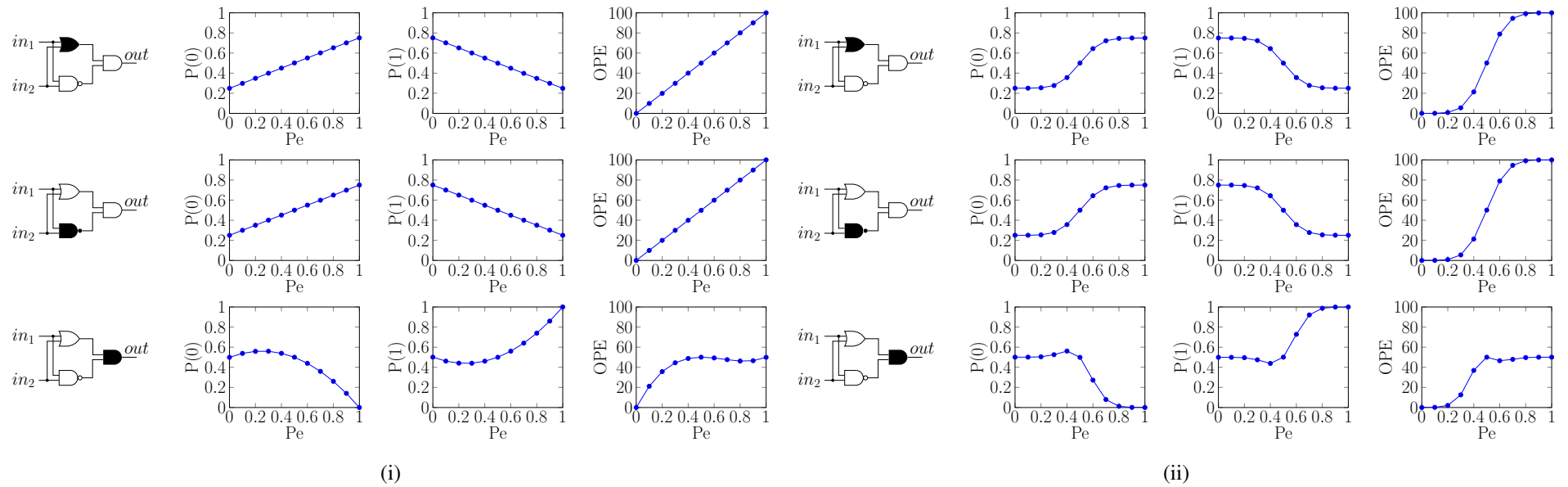


Figure 3.16: $P(0)$, $P(1)$, and OPE for each gate in XOR topology-iv for error distribution (a) Uniform, and (b) Normal

Chapter 4

Analysis of Chains and Adder Models

This chapter investigates the effect of connecting couple gates serially with inputs applied to the first gate to examine dependencies between the gates. This study is applied on a chain of NOT gates, and different architectures of XOR gates introduced in chapter 3 to mimic sum of full adder (FA) model. Moreover, an analysis for approximate and various probabilistic 8-bit adders is introduced with the results of applying uniform and normal error distributions on these probabilistic and approximate adder models.

4.1 Chain of Gates

Two chain of gates are investigated in this section to check the effect of error in the first gate on the rest gates in the chain. 5 stages of inverters connected in series, and 2-XOR chain are presented and clarified. The chain of XOR is examined for the different topologies presented in chapter 3. These chains are tested by applying uniformly distributed 0, 1 stream for the chain inputs.

4.1.1 Inverter Chain

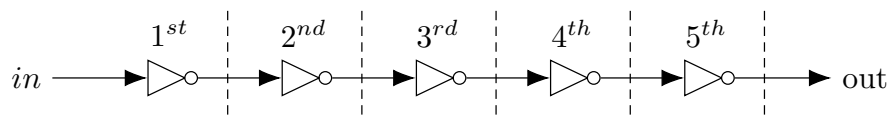


Figure 4.1: 5-stages chain of inverters

Chain of 5 inverters connected serially is shown in figure 4.1. Uniform and normal distributions are applied to model error on each stage of these stages. Independent input of the generated distribution is applied on the input port. Figure 4.2 presents the output of each stage is observed to show the effect of its error and previous stage's output. The results from the plotted graphs can show:

- The effect of injected error in each stage has the same behavior for either normal or uniform distribution.
- When probability of error is in the range $0 : 0.5$, the output percentage error in each stage is increased for the same P_e which means there is a small degradation in the performance of gates.

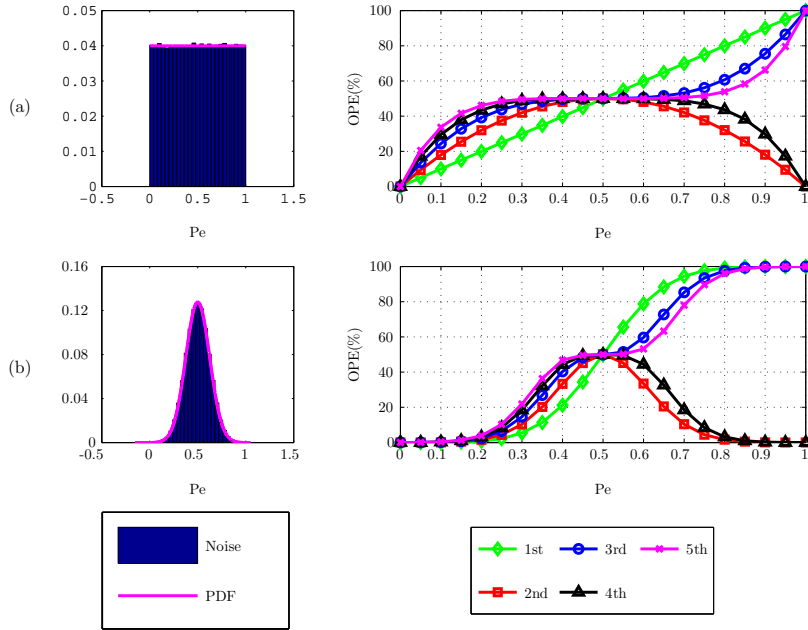


Figure 4.2: OPE(%) for each stage in the inverter chain (on the right) and Histogram of the injected noise with the fitted PDF of it (on the left) are plotted for, (a) Uniform Distribution, and (b) Normal Distribution

- When probability of error is in the range 0.5 : 1, the output percentage error in each stage is decreased for the same P_e in even and odd stages which means there is an improvement in the performance of gates.
- For normal distribution, OPEs for all the stages are almost the same in case of P_e is in the range 0 : 0.2, and for the even and odd stages in the range 0.8 : 1.
- The probability of error = 0.5 can be considered as an equilibrium point for all the stages. As whether the stage is odd or even, and the error distribution is uniform or normal, the output percentage error of the gates will be equal 50%.
- Tables 4.1 and 4.2 show the numerical values for the output percentage error to each stage in the full range for probability of error.

PE	1 st stage	2 nd stage	3 rd stage	4 th stage	5 th stage
0	0	0	0	0	0
0.1	9.989	17.95	24.40	29.50	33.62
0.2	19.90	31.93	39.19	43.41	46.08
0.3	29.82	41.92	46.77	48.86	49.59
0.4	39.78	48.02	49.54	49.85	49.95
0.5	49.8	50.02	49.99	50.03	49.98
0.6	59.805	48.08	50.44	49.89	49.93
0.7	69.84	42.086	53.08	48.72	50.53
0.8	79.96	32.06	60.69	43.62	53.86
0.9	89.97	18.06	75.45	29.69	66.21
1	100	0	100	0	100

Table 4.1: OPE for error uniform distribution in chain of inverters

<i>PE</i>	<i>1ststage</i>	<i>2ndstage</i>	<i>3rdstage</i>	<i>4thstage</i>	<i>5thstage</i>
0	0	0	0	0	0
0.1	0.06	0.14	0.213	0.29	0.359
0.2	0.782	1.587	2.37	3.14	3.89
0.3	5.452	10.3	14.60	18.48	21.94
0.4	21.20	33.32	40.38	44.38	46.80
0.5	49.94	50.17	49.86	49.96	49.97
0.6	78.78	33.51	59.65	44.44	53.16
0.7	94.48	10.38	85.31	18.55	78.03
0.8	99.15	1.649	97.57	3.18	96.05
0.9	99.93	0.136	99.79	0.27	99.65
1	100	0	100	0	100

Table 4.2: OPE(%) for Normal Distribution in Chain of Inverters

4.1.2 XOR chain

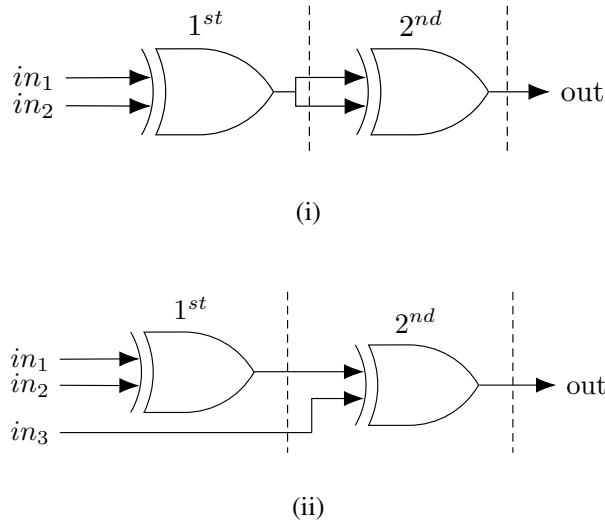


Figure 4.3: (a) 2-stages XOR chain, and (b) Sum function in Full adder

Figure 4.3i presents 2 XOR gates connected serially by applying the output of first stage into both inputs of the second stage. This is the basic method to implement sum function in full adder which is shown in figure 4.3ii and its truth table is presented in table 4.3. in_1 and in_2 for the first stage represent the weighted bits of the numbers to be add, and in_3 is driven to add C_{out} from the previous full adder. For example, adding A, B which are n bits is done by applying $A(i)$ in in_1 , $B(i)$ in in_2 , $C_{out}(i-1)$ in in_3 where i is the location of bit in the stream.

For simplicity, the output of first stage is connected to both inputs of the second stage which will add some dependency in the this stage. Two uniform random input streams are applied for in_1 and in_2 for independent uniform and normal error distributions in each stage. The chain is tested for the simplest XOR gate and the four proposed topologies of XOR gate in chapter 3 to study the effect of probabilistic error in XOR chain.

in_1	1	0	1	0	1	0	1	0
in_2	1	1	0	0	1	1	0	0
in_3	1	1	1	1	0	0	0	0
Sum	1	0	0	1	0	1	1	0

Table 4.3: Truth Table of SUM in FA for different input combinations

4.1.2.1 Chain of simple XOR

Investigating the effect of Uniform and Normal error distributions in the chain of simple XOR gates is shown in figure 4.4. Also, table 4.4 presents numerical values for output percentage error in first and second stages. For both distributions, OPE is symmetrical in the second stage. As in the chain of simple inverters, the lower P_e has less degradation in OPE. On the other side, there is a great improvement in OPE for higher P_e .

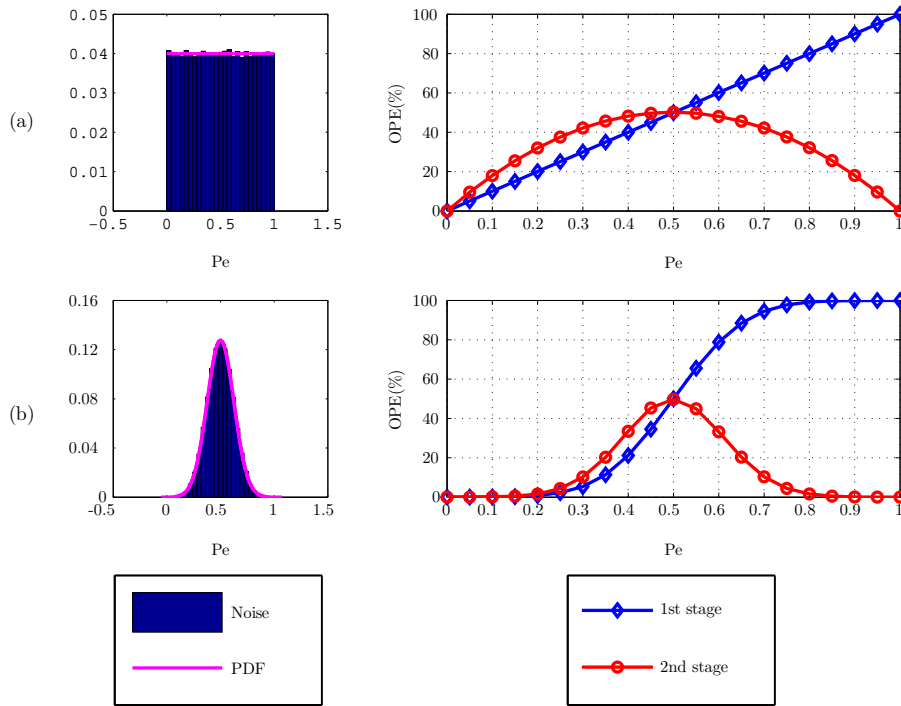


Figure 4.4: OPE(%) for each stage in the chain of simple XOR gate (on the right) and Histogram of the injected noise with the fitted PDF of it (on the left) are plotted for, (a) Uniform Distribution, and (b) Normal Distribution

	PE	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Unif	1^{st}	0	10.04	20.09	29.99	40.01	49.98	60.17	70.09	80.01	89.99	100
	2^{nd}	0	17.99	32.04	42.2	48.21	50.	48.1	42.2	32.15	18.07	0
Norm	1^{st}	0.01	0.07	0.77	5.38	21.19	50.04	78.78	94.52	99.17	99.93	99.99
	2^{nd}	0.01	0.15	1.59	10.26	33.47	49.73	33.20	10.33	1.63	0.13	0.01

Table 4.4: OPE(%) for uniform and normal distributions in the chain of simple XOR

4.1.2.2 Chain of topology-I XOR

For a chain of XOR from topology-i, the results shown in figure 4.5 can be concluded as follow: For uniform distribution error, OPE in the second stage is almost the same as first stage for P_e in the range of [0.3 : 0.7] which equals 50% from the total n input combinations. Error of normal distribution shows better performance than uniform distribution for OPE in the first and second stage of the chain in lower P_e . On the other hand, for higher P_e the OPE shows better performance in the second stage only. Table 4.5 shows the numerical values for OPE in different stages of the chain for both distributions.

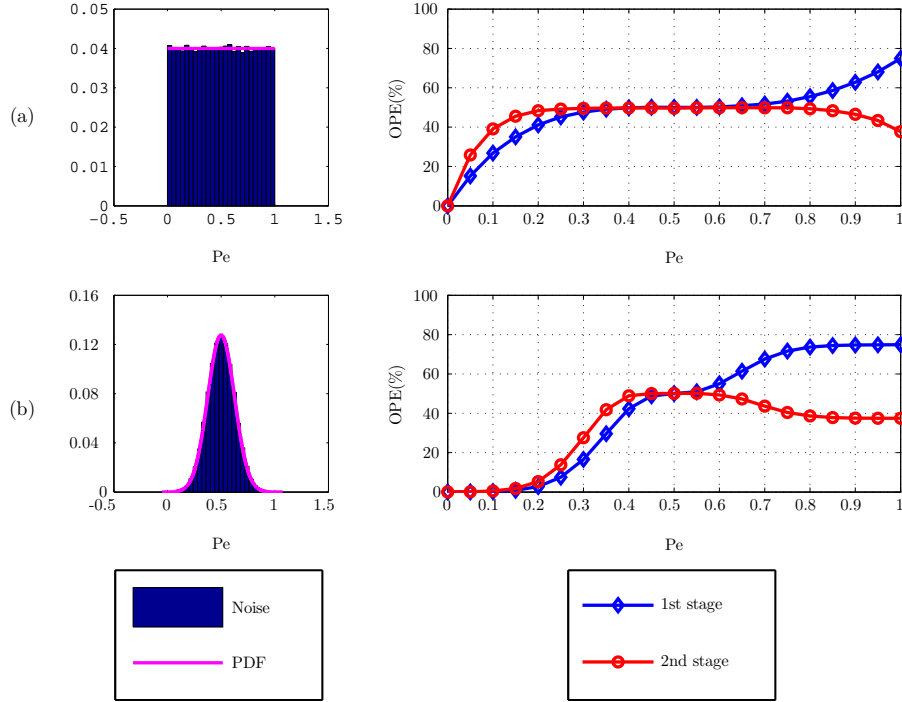


Figure 4.5: OPE(%) for each stage in the chain of topology-i XOR gate (on the right) and Histogram of the injected noise with the fitted PDF of it (on the left) are plotted for, (a) Uniform Distribution, and (b) Normal Distribution

	PE	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Unif	1 st	0	26.82	41.12	47.66	49.82	50.01	50.	51.7	55.52	62.77	74.92
	2 nd	0	39.08	48.35	49.55	49.75	49.63	49.79	49.82	49.32	46.54	37.67
Norm	1 st	0.02	0.	2.76	16.55	42.33	50.03	54.99	67.54	73.68	74.74	74.83
	2 nd	0.03	0.50	5.32	27.63	48.91	50.09	49.32	43.67	38.66	37.58	37.49

Table 4.5: OPE(%) for uniform and normal distributions in the chain of topology-i XOR

4.1.2.3 Chain of topology-II XOR

Figure 4.6 shows OPE for the first and second stage in the chain of topology-ii XOR for uniform and normal error distributions. Also, the numerical values are presented in table 4.6. For the applied error with uniform distribution, OPE in both stages are the for P_e range [0.35 : 0.65]. For P_e less than 0.35, OPE has a little degradation in the second stage, and better performance for P_e higher than 0.65. OPE in the second stage for normal distribution error show better performance in full range of P_e rather than the uniform

distribution. Also, for both distributions the higher P_e , the better OPE in the second stage of the chain.

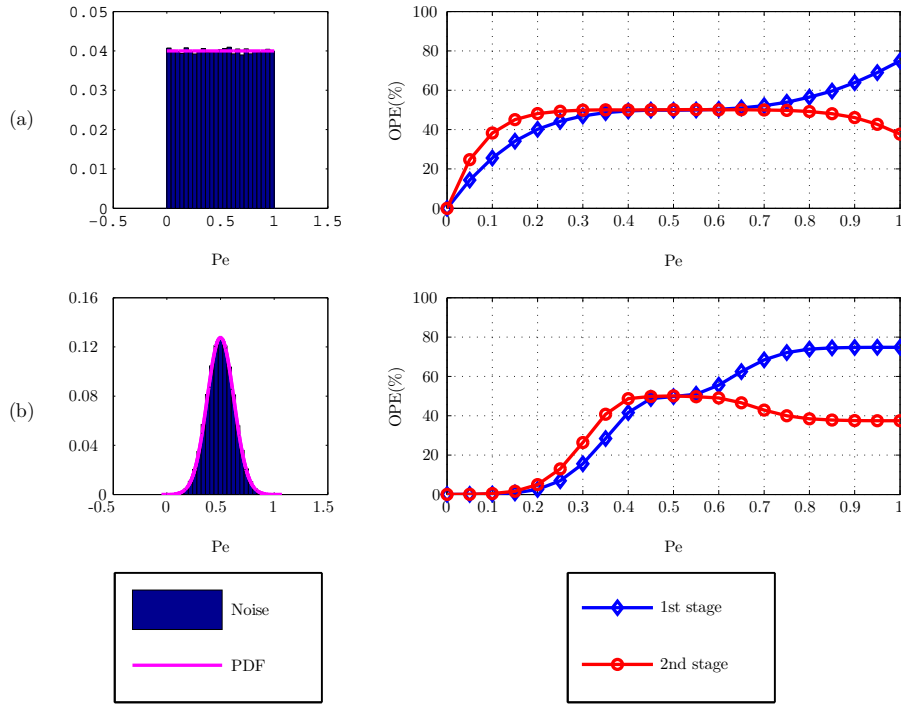


Figure 4.6: OPE(%) for each stage in the chain of topology-ii XOR gate (on the right) and Histogram of the injected noise with the fitted PDF of it (on the left) are plotted for, (a) Uniform Distribution, and (b) Normal Distribution

	PE	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Unif	1 st	0	.59	40.18	46.95	49.5	49.86	50.29	52.16	56.46	63.86	74.92
	2 nd	0	38.27	48.16	49.87	49.99	50.12	50.04	49.98	49.18	46.08	37.67
Norm	1 st	0.02	0.24	2.58	15.57	41.65	49.78	55.54	68.44	73.86	74.75	74.82
	2 nd	0.03	0.46	5.01	26.37	48.68	50.14	49.06	42.88	38.47	37.57	37.50

Table 4.6: OPE(%) for uniform and normal distributions in the chain of topology-ii XOR

4.1.2.4 Chain of topology-III XOR

Chain of XORs from topology-III shows better performance in both stages as the max OPE is about 50% for the total number of tested inputs when $P_e = 0$. Also, the first and second stages are almost following each other with lower performance at higher P_e in the second XOR gate. Therefore, the OPE performance is expected for longer chain with this type of XOR logic gate to follow the previous gates as concluded from the OPE for uniform and normal distributions in figure 4.7. The numerical results of OPE for the full range of P_e is presented in table 4.7 for normal and uniform distributions.

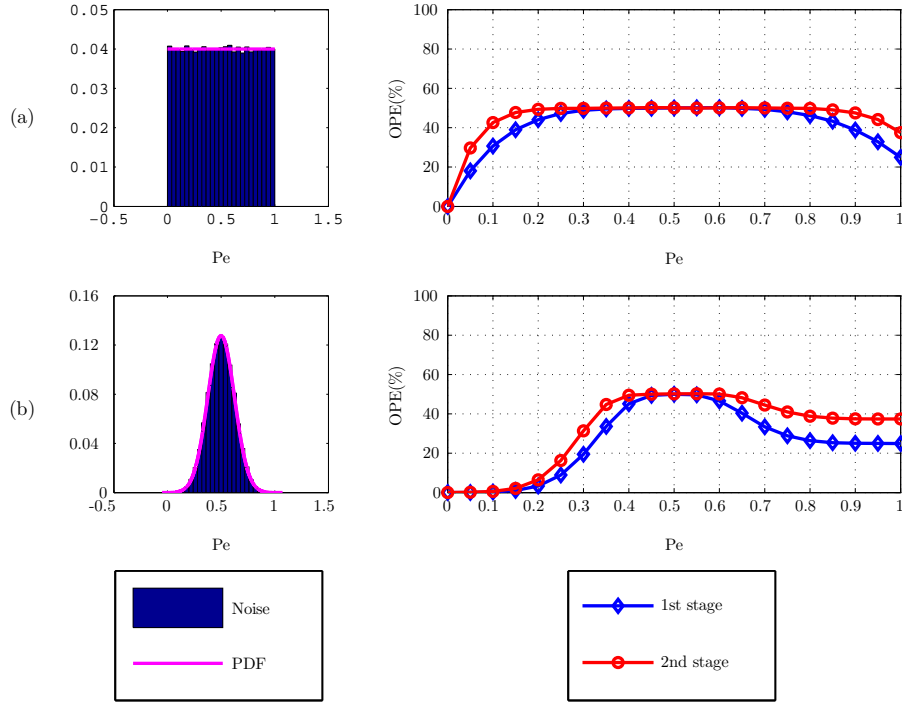


Figure 4.7: OPE(%) for each stage in the chain of topology-iii XOR gate (on the right) and Histogram of the injected noise with the fitted PDF of it (on the left) are plotted for, (a) Uniform Distribution, and (b) Normal Distribution

	PE	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Unif	1 st	0	30.68	44.05	48.85	49.84	50.02	50.16	49.35	46.18	38.85	24.91
	2 nd	0	42.57	49.35	49.83	50.05	50.08	50.09	50.05	49.8	47.49	37.45
Norm	1 st	0.02	0.30	3.35	19.45	45.10	50.00	46.59	33.45	26.41	.06	24.94
	2 nd	0.03	0.58	6.48	31.38	49.45	50.17	50.08	44.49	38.82	37.51	37.41

Table 4.7: OPE(%) for uniform and normal distributions in the chain of topology-iii XOR

4.1.2.5 Chain of topology-IV XOR

	PE	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Unif	1 st	0	21.	35.59	44.2	48.7	49.92	49.1	47.59	46.42	46.83	49.99
	2 nd	0	33.49	45.9	49.55	50.02	50.01	50.06	50.02	49.71	49.91	49.94
Norm	1 st	0.01	0.17	1.97	12.41	37.09	50.05	46.39	47.95	49.72	50.07	50.10
	2 nd	0.02	0.33	3.87	21.87	46.71	49.87	49.76	49.94	50.07	50.05	50.06

Table 4.8: OPE(%) for uniform and normal distributions in the chain of topology-iv XOR

For chain of XOR from topology-IV, figure 4.8 presents output percentage error vs probability of error for first and second stages of the chain for error as uniform and normal distribution. It is clear the maximum OPE is 50% of total number for N bit stream. Also, the OPE for both stages are almost identical for higher P_e once it settled on P_e of 0.5 for any of the applied distributions. There is a little degradation for lower P_e in both stages. For normal distribution, this degradation is in the region of $[0.2 : 0.5]$ of P_e range. The OPEs for full range of P_e is presented in table 4.8 for both distributions.

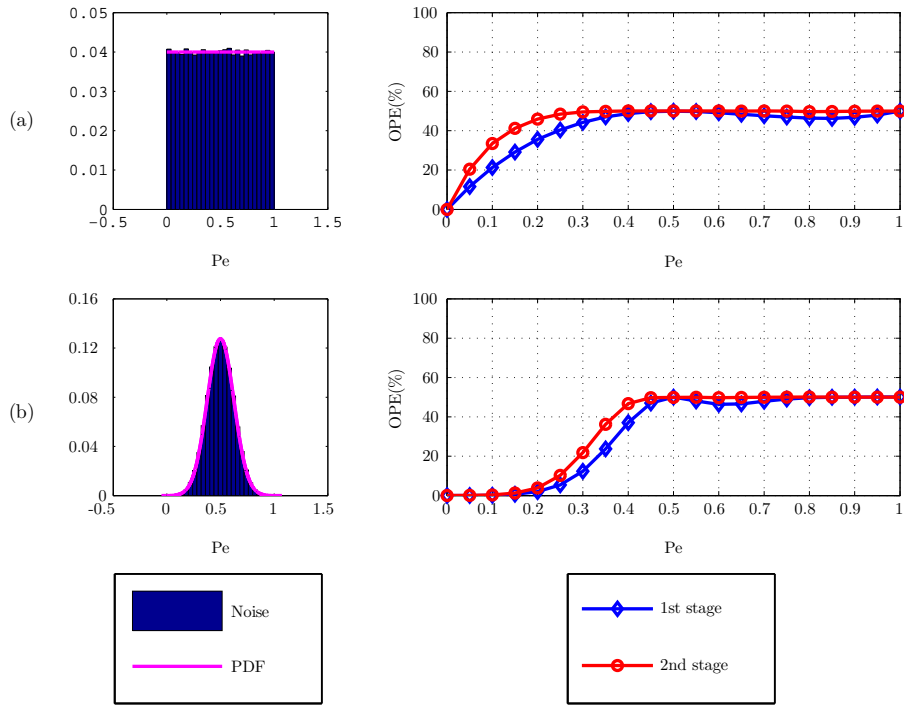


Figure 4.8: OPE(%) for each stage in the chain of topology-iv XOR gate (on the right) and Histogram of the injected noise with the fitted PDF of it (on the left) are plotted for, (a) Uniform Distribution, and (b) Normal Distribution

4.1.2.6 Comparison between different topologies of XOR

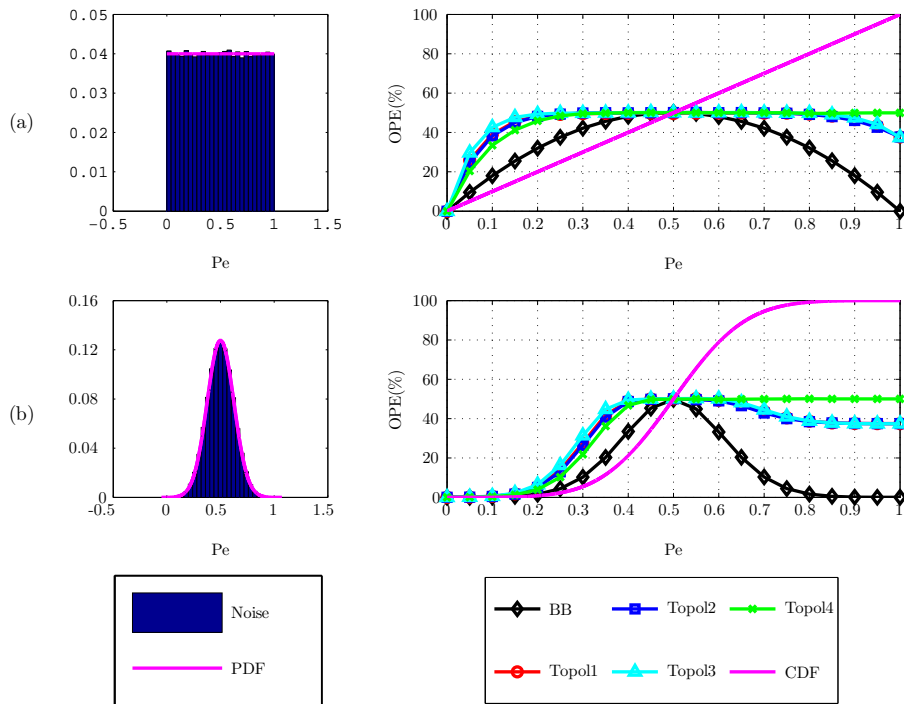


Figure 4.9: OPE(%) for 2^{nd} stage in different XOR chains, and CDF of the injected noise in this stage (on the right) and Histogram of the injected noise with the fitted PDF of it (on the left) are plotted for, (a) Uniform Distribution, and (b) Normal Distribution

A detailed analysis for the different XOR chains is clarified based on figure 4.9 that shows the output percentage error for each XOR topology in the second stage of the chain for both uniform and normal distributions beside the CDF of a single gate error. Moreover, numerical values for output percentage error in the second stage for these chains are presented in table 4.9 for uniform and table 4.10 for normal distributions. The results can be concluded as follow:

<i>PE</i>	<i>XOR_{BB}</i>	<i>XOR_{Imp1}</i>	<i>XOR_{Imp2}</i>	<i>XOR_{Imp3}</i>	<i>XOR_{Imp4}</i>
0	0	0	0	0	0
0.1	17.99	39.08	38.27	42.57	33.48
0.2	32.04	48.35	48.16	49.345	45.89
0.3	42.19	49.55	49.87	49.83	49.54
0.4	48.21	49.74	49.98	50.04	50.02
0.5	50.	49.62	50.12	50.075	50.00
0.6	48.10	49.79	50.03	50.08	50.05
0.7	42.20	49.82	49.975	50.05	50.01
0.8	32.15	49.32	49.18	49.8	49.71
0.9	18.07	46.53	46.07	47.49	49.91
1	0	37.67	37.67	37.45	49.935

Table 4.9: OPE(%) for uniform distribution in the second stage for XOR chain

<i>PE</i>	<i>XOR_{BB}</i>	<i>XOR_{Imp1}</i>	<i>XOR_{Imp2}</i>	<i>XOR_{Imp3}</i>	<i>XOR_{Imp4}</i>
0	0.006	0.023	0.0275	0.03	0.016
0.1	0.1475	0.493	0.463	0.58	0.334
0.2	1.586	5.32	5.013	6.483	3.868
0.3	10.26	27.62	26.37	31.38	21.865
0.4	33.46	48.90	48.68	49.45	46.71
0.5	49.73	50.09	50.1	50.17	49.86
0.6	33.19	49.32	49.05	50.0775	49.76
0.7	10.3275	43.66	42.87	44.48	49.94
0.8	1.633	38.65	38.47	38.815	50.07
0.9	0.13	37.575	37.57	37.51	50.045
1	0.008	37.48	37.49	37.41	50.05

Table 4.10: OPE(%) for normal distribution in the second stage for XOR chain

- The best OPE performance is in simple XOR logic gate either in the first stage or in the second stage, cause the error distribution is applied only for single logic gate.
- Chains with simple XOR logic gates show a symmetric OPE which mean the second stage is working as an inverter for the first stage which leads for example to have 0% OPE when P_e is '1'.
- The first and second topologies of XOR logic gate show same performance in both stages of the chain for uniform and normal distributions which is as a result of the gates structure in these topologies.

- for lower range of P_e , the third topology achieves best performance for OPE, but for higher range it is almost settled on 50% which mean half of the generated outputs will be wrong.
- For Normal distribution, the applied data has $\mu = 0.5$ and $\sigma = 0.125$ which shows the reason that all generated CDFs are centered around $P_e = 0.5$.

4.2 Analysis of ripple carry adder

In this section, an analysis for 8-bit ripple carry adder (RCA) is introduced using model from approximate and probabilistic gates. As discussed previously, full adder (FA) is the main component to build RCA, so 8-bit ripple carry adder consists of 8 full adder blocks. The model and logic gate combinations for FA is presented in figure 4.10. Adder models are tested by generating around 12000 numbers for the inputs to calculate a reasonable OPE and MED as a, b inputs are in the range $[0, 5]$, and C_{in} is 0, 1. Two simulation results proposed to assess the model performance for adder are:

1. OPE vs P_e for different RCA models using simple and the four complex topologies for XOR logic gate in the full adder while varying the number of incorrect bits for the lower part till all adder bits be covered.
2. MED vs number of incorrect lower bits in the adder for probabilistic adder model with probability of error 0.1, 0.2 for uniform and normal distributions while using approximate adder model as a reference.

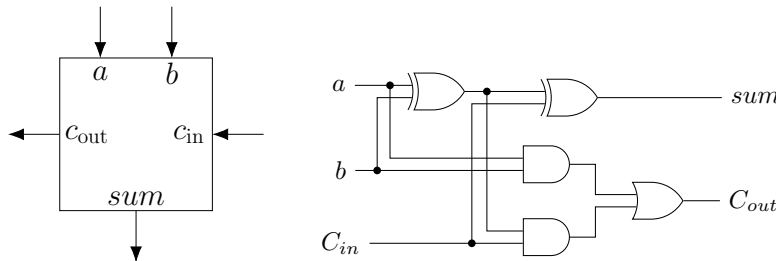


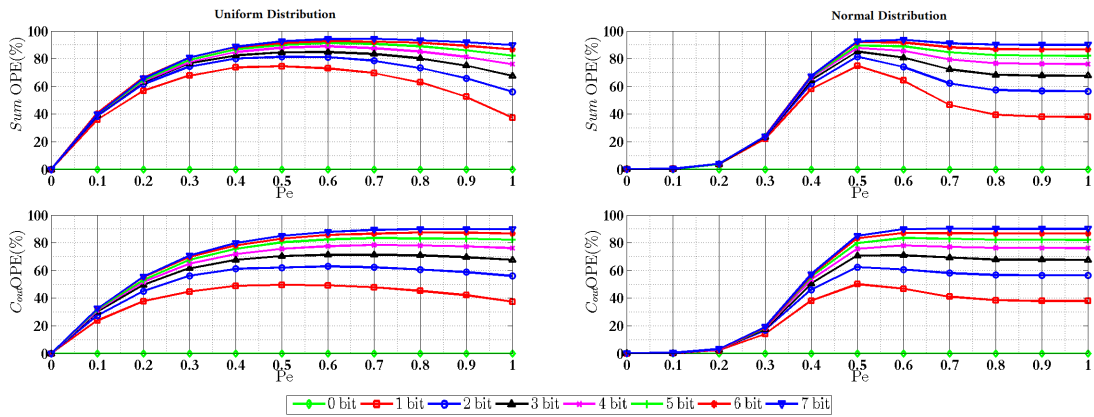
Figure 4.10: Full adder model and its logic gates combinations

4.2.1 Output percentage error versus probability of error

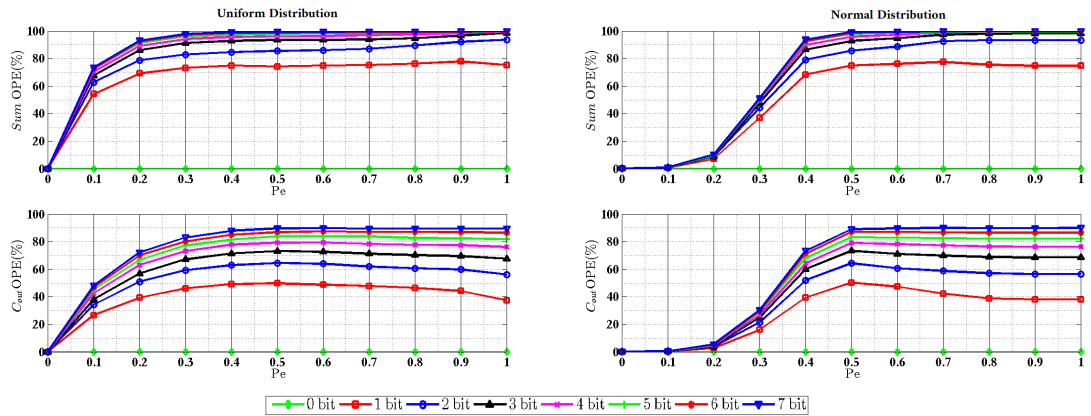
Figure 4.11 shows the simulation results for OPE vs P_e for different 8-bits RCA using various XOR topologies. Also, the number of inexact bits in the adder is covered for all the cases from no faulty bits to all wrong bits in the adder. These adder models are applied to uniform and normal distributions as error function across their bits. The simulation results are analyzed and concluded as follow:

- Increasing number of imprecise bits in the lower part leads to higher OPE which means more wrong calculations for both sum and C_{out} .
- For both distributions, RCAs with XOR topologies I, II still have the same OPE across the full range of P_e and for any number of incorrect bits.

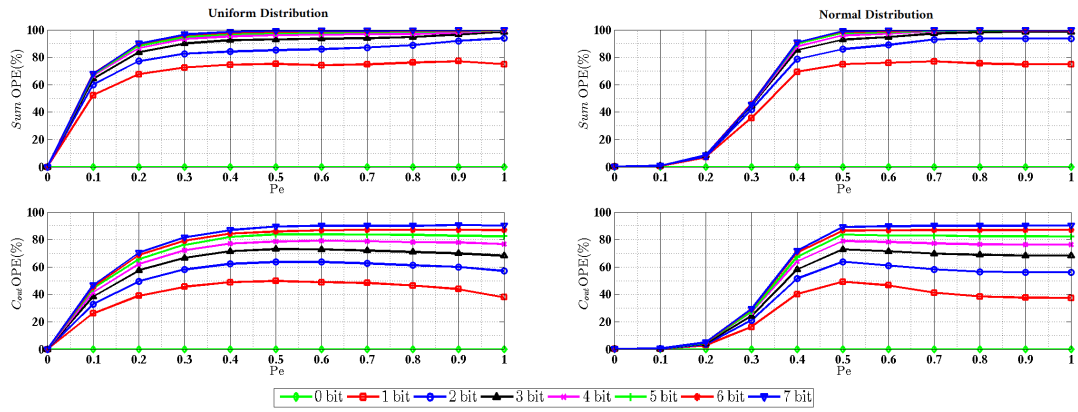
- simple XOR and topologies II, IV show almost same OPE performance which always have the worst OPE for error with normal distribution is when $P_e = 0.5$ as this 0.5 is the mean of this normal distribution.
- Another proof for the validity of proposed model is that the models with errors as uniform distributions across the gates tend to have OPE almost uniform, the same corresponding to errors with normal distributions. This is a main theory in statistics named mixture distribution [54, 55].
- All topologies have almost the same OPE for C_{out} as there are 4 different sources to calculate it: the external inputs (a , b , and C_{in}) beside the first XOR logic gate output. Also, the logic gates (AND, OR) used to generate C_{out} are simple.



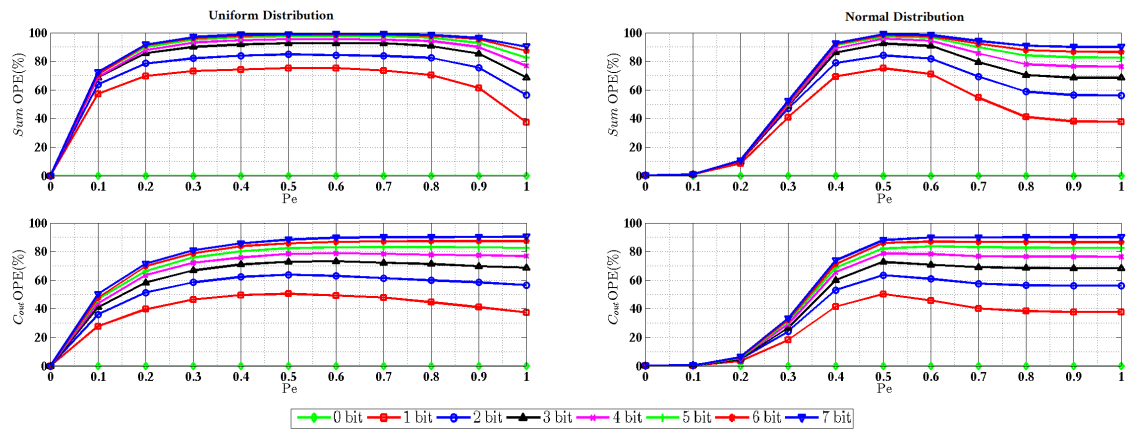
(i) RCA using probabilistic simple XOR logic gate



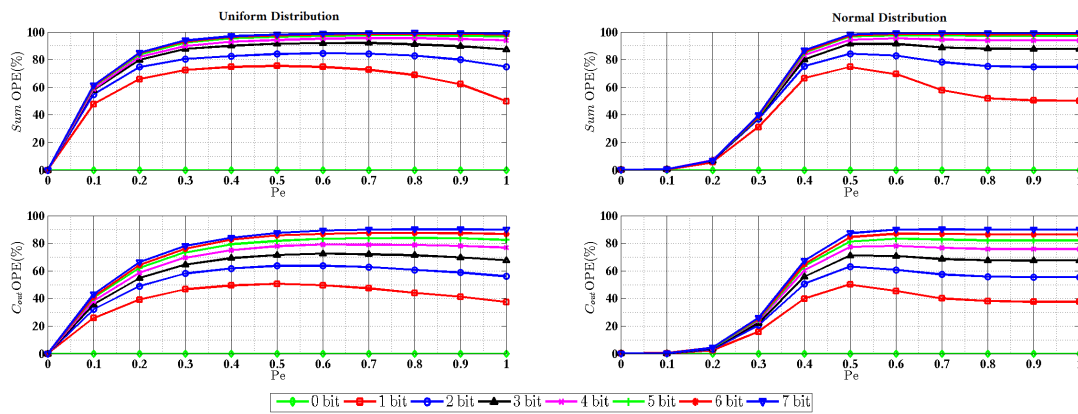
(ii) RCA using probabilistic topology-I XOR logic gate



(iii) RCA using probabilistic topology-II XOR logic gate



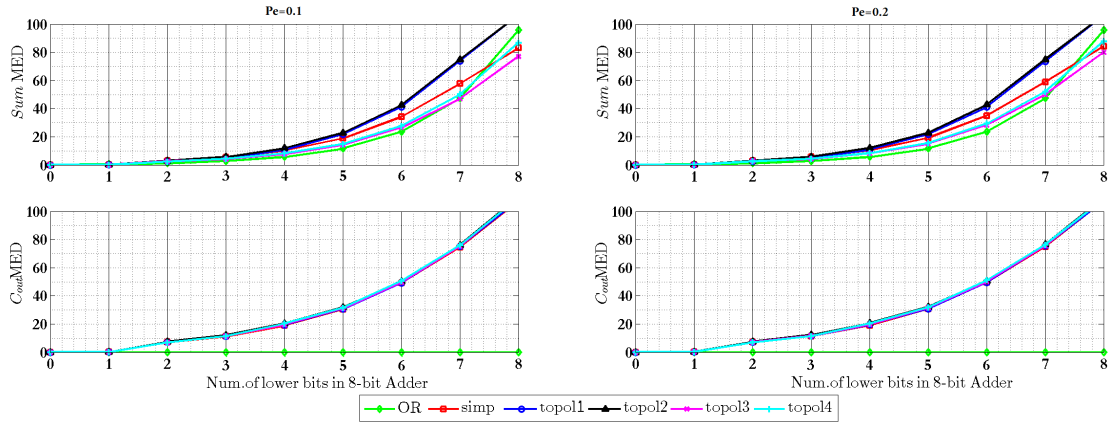
(iv) RCA using probabilistic topology-III XOR logic gate



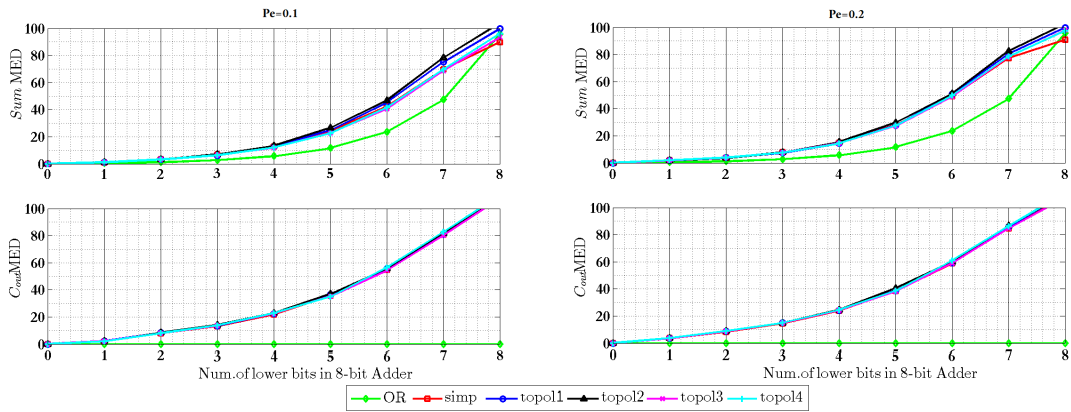
(v) RCA using probabilistic topology-IV XOR logic gate

Figure 4.11: OPE for sum and C_{out} in RCA with different imprecise bits using different XOR topologies for error distribution as: uniform (on the left), and normal (on the right)

4.2.2 Mean error distance versus imprecise bits



(i) Normal error distribution



(ii) Uniform error distribution

Figure 4.12: MED vs the number of imprecise lower bits in different adders with $P_e = 0.1$ (on the left), and $P_e = 0.2$ on the right

The simulation results in figure 4.12 shows the mean error distance while varying the number of imprecise bits from the lower part in 8-bit RCA. The purpose of these results is to determine efficiency of the model by comparing the generating MED results with these in Liang et al. [40]. Probabilistic adder model using error distribution as normal can approximately be an example for probabilistic full adder (PFA) where the error probability is the intersection of two normal distributions. Also, an approximate adder model is used to mimic lower OR bit adder (LOA) by injecting fixed error for the inputs combination $a = 1, b = 1$. Moreover, this model will be used as a reference between probabilistic adder models with uniform and normal distributions. These results prove that the proposed model for approximate and probabilistic gates are efficient as:

- For both adder models, MED is increasing exponentially by forcing more number of bits in the lower part to be incorrect.
- Comparing MED generated from the suggested models with those from Liang work [40] shows almost the same MED performance for LOA model, and deviation

in the results of PFA as a result of using normal distribution not the intersection between two normal distributions.

- The first and second XOR topologies shows the worst MED measurements while the third and fourth show better performance with advantage for the third topology in increasing the number of imprecise bits.
- MEDs of all topologies for uniform distribution error are almost the same, cause the data is equally weighted in the distribution.
- C_{out} for LOA is isolated as the injected error is in input combination $a = 1, b = 1$ which makes C_{out} in this case always '1' regardless the wrong output generated from the first XOR logic gate.

4.3 Conclusion

In this chapter, simulation results for the stages in XOR and inverter chains were presented to clarify the result of connecting gates from same type in series, and study the dependencies between these gates. The chain of simple logic gate confirmed the exactness of suggested models as measured OPEs tend to follow the injected error for uniform and normal distributions. Also, a ripple carry adder were investigated using approximate and probabilistic models. The effectiveness of these imprecise adder models were confirmed by comparing the generated MEDs with another work.

Chapter 5

Quasi-TRNG implementation

This chapter discusses a combination of the proposed probabilistic XOR gate models in one of random number generator's implementation to check the effect of these imprecise gates on system's properties. The histograms and outputs probability are investigated for both accurate and probabilistic models. Also, NIST statistical tests are applied on the generated sequences from the implementation to check its performance.

5.1 Linear Feedback Shift Register

A Linear Feedback Shift Register (LFSR) is a common technique to implement pseudo random number generators (PRNGs). It can be implemented in both hardware and software [56]. A LFSR is a sort of shift register which has a linear function of its previous states using a feedback. Inserting XOR gates in the feedback is the most common method to represent the desirable linear function. The initial value for the shift register is known as seed. For any n -order LFSR, some bits are engaged in the feedback polynomial to determine the linear function. Choosing the locations of these bits (Called Taps) are so important to achieve the maximal sequence length $2^n - 1$.

There are two different kinds of LFSR structure: Fibonacci LFSR and Galois LFSR. The main difference between both structures is based on the location of taps, for Fibonacci configuration, the taps are located externally which mean each tap's output is connected to the next tap. In other words, the XOR taps are cascaded so the generation of data in the sequence depends on the taps connections. In Galois configuration, there are internal taps between the registers which means no XOR gates run in serial, therefore the delay time is reduced compared with Fibonacci configuration, thus Galois is more efficient in random number generation. Applying the same feedback polynomial for both configuration will produce the same length of state period. 10-bit programmable Galois LFSR is introduced in figure 5.1. This implementation can achieve PRNG with orders from [2 : 10]. Table 5.1 defines the best polynomials to get maximal sequence length [57].

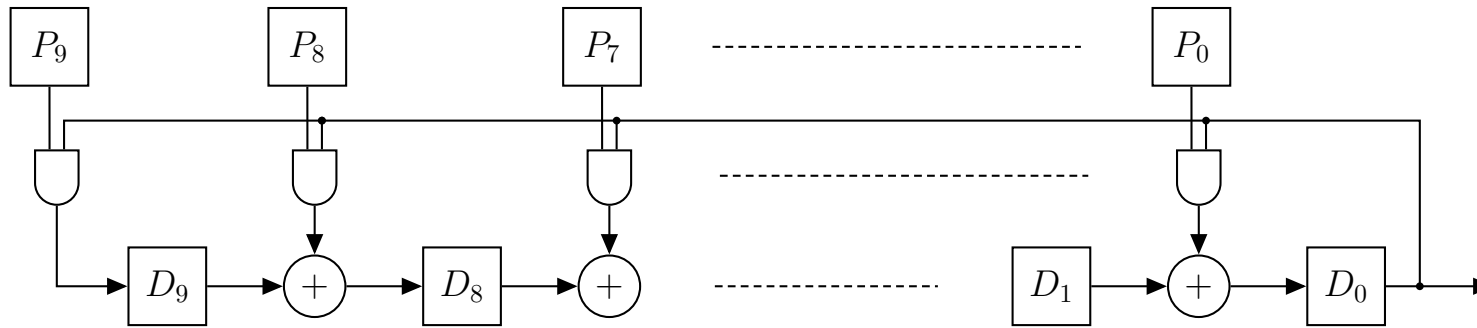


Figure 5.1: Block Diagram of 10-bit generic LFSR

Resolution	Sequence Length	Feedback Taps	10-bit Polynomial
2	3	1, 2	10'b00 0000 0011
3	7	2, 3	10'b00 0000 0110
4	15	3, 4	10'b00 0000 1100
5	31	2, 3, 4, 5	10'b00 0001 1110
6	63	2, 3, 5, 6	10'b00 0011 0110
7	127	4, 5, 6, 7	10'b00 0111 1000
8	255	4, 5, 6, 8	10'b00 1011 1000
9	511	5, 6, 8, 9	10'b01 1011 0000
10	1023	6, 7, 9, 10	10'b11 0110 0000

Table 5.1: 10-bit Programmable LFSR Polynomials

5.2 Statistical Tests

Different statistical testing environments can be applied to the generated sequences of random generator to detect the specific characteristic expected of truly random sequences, and ensure that the source of randomness is operating in sufficient way. For example, it is important to check the randomness of random generator in computer security which required convenient metrics to test the used generator. There are many statistical tests, each of them depends on some function of the generated sequences. A literature survey for different randomness tests is presented in [58]. It can not be considered that there is a complete statistical test of randomness. These statistical tests - like Diehard [59], NIST [60], and FIPS [61] - check the validity of randomness properties related to the sequence. At the end of chapter, a detailed results of the statistical testing on generated bit sequences, NIST Sp. 800-22 statistical tests, are presented.

5.3 Results

This section introduces the results of three different histograms for tests applied on the programmable LFSR. These tests are generated to check the simulation results as follow:

1. Checking the effect of different probabilistic XOR topologies on the randomness of LFSR compared with the accurate XOR logic gate, Histograms for programmable LFSR with order-8 is presented and tested using NIST tests. Also, output probability for the generated numbers of order-4 with different statistical quantities as mean, deviation, and median are introduced.
2. Applying the probabilistic LFSR with different XOR models on various orders as (4, 6, 8, and 10) to confirm the distribution of generated sequences have specific manner regardless the selected order.
3. Choosing XOR topology from previous tests to check the randomness of generated sequences using different polynomials for order-4 LFSR rather than the best polynomial given in tab 5.1.

There are three different sources of error in the programmable PRNG which can be investigated through the implied tests: probabilistic logic gates (XOR, AND), and registers. The simulation results concern about the probabilistic logic gates in the model with accurate registers. Also, the uniform distribution is chosen to be applied on these probabilistic gates. Another important metric is P_e which is fixed across all the probabilistic gate through the simulations by a value 0.1. The generated numbers are in the range of $10^5 - 10^6$ samples to evaluate the probabilistic model.

For programmable LFSR, the selected n-bits polynomial for any order-n will not isolate the higher probabilistic bits from propagating to the target bits in case of using probabilistic AND logic gate in the feedback. As in the accurate programmable LFSR technique, the AND gates between the polynomial and feedback from the output force propagation of 0's in the XOR gates for higher bits based on the target order.

5.3.1 Different Topologies

5.3.1.1 Histograms of Order-8 LFSR

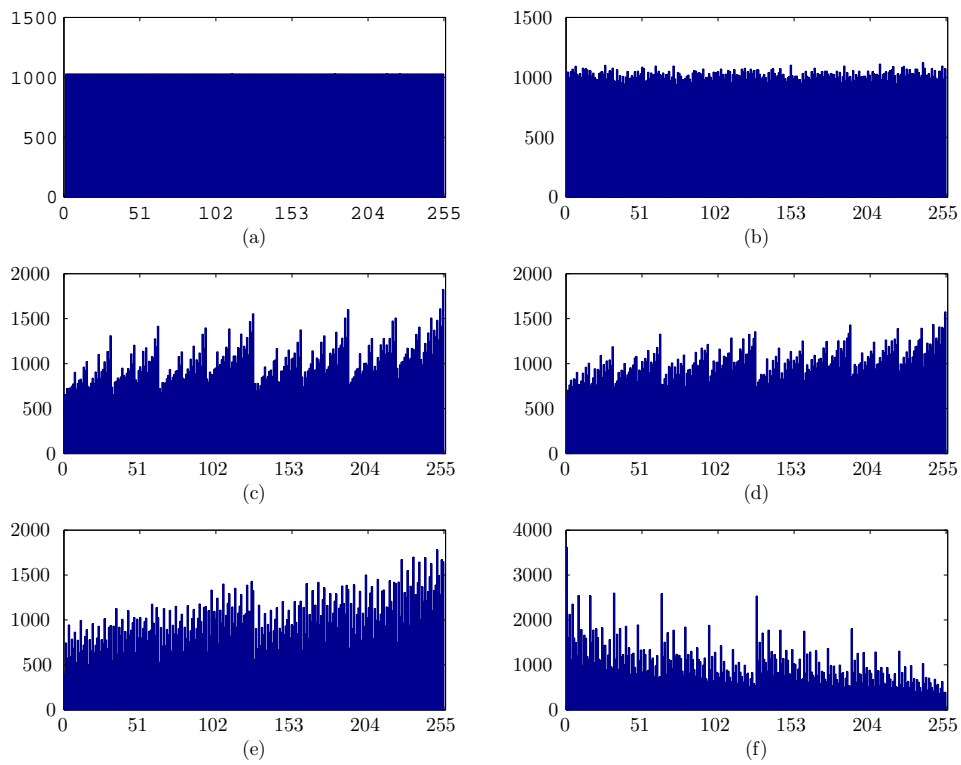
The histograms for order-8 programmable LFSR with different topologies of probabilistic XOR logic gates as a taps between registers are presented for two cases: (1) accurate AND gates, and (2) probabilistic AND gates in the feedback as shown in figures 5.2i, 5.2ii respectively. From these histograms, it is clear that probabilistic XOR is the dominant logic gate in the programmable LFSR model, as the generated numbers distribution in both graphs is almost the same. The obtained histograms for order-8 probabilistic LFSR model can be concluded as follow:

- The sequence distribution of the LFSR model with $P_e = 0$ in the probabilistic logic gates is uniform which confirms the exactness of proposed models to be free from error and mimic accurate model in case of P_e is '0'.
- The effect of probabilistic AND logic gate in the accurate model is shown in figure 5.2ii(a). The histogram shows that the probability of getting number '0' is higher than $2x$ the other numbers' probability. This is one of the drawbacks for accurate LFSR which it can not cover from state '0' or the maximum value '255' for order-8 in case of using XOR or XNOR gates respectively as taps. This is another proof for the dominance of XOR gate in the model.
- For probabilistic XOR with $P_e = 0.1$, The histogram for simple model shows a little deviation in the distributed probability for generated numbers. On the other hand, complex models show almost an analogous histograms as there are about 8 regions in the histograms of topologies I, II, and IV which indicate higher probability for some numbers rather than the others within these regions. Topology III shows almost the better histogram distribution among the other topologies.

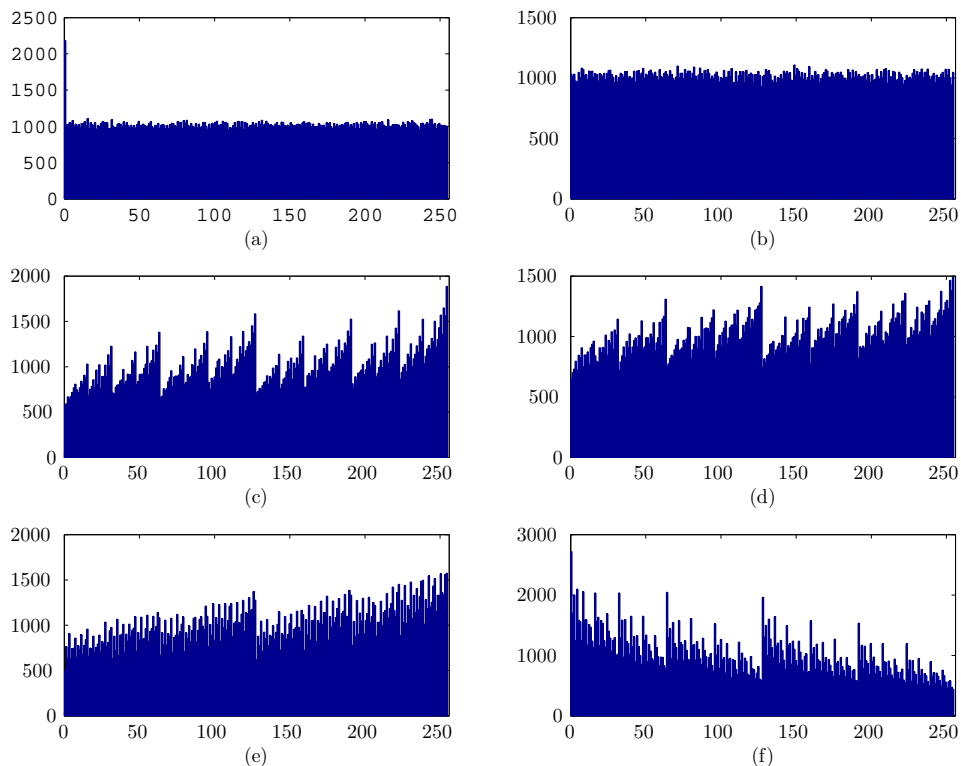
5.3.1.2 NIST tests results

The LFSR models with these probabilistic XOR logic gates are applied for NIST tests to determine the randomness efficiency of the generated sequences for accurate and all probabilistic XOR topologies. Tables 5.2, present the results of NIST tests in two ways: proportion value (PP) which is normalized to one, and the availability of P-values (PV) distribution which is acceptable for values higher than 0.0001 as clarified in [60]. For NIST sp800-22 platform, 10 bit streams are tested which limit pass rate for statistical test to 8 streams. Each stream has 10000 random bits obtained from the probabilistic LFSR models Random executions, and random executions variant tests are not involved in the results. α is set to 0.01 through the statistical tests.

Table 5.2 shows an improvement in the PV-values in non-overlapping template test for probabilistic LFSR over the accurate model. Also, it is clear that linear complexity -an important test for many key generators- is passed for both PV-values and PP which confirms the complexity ins generated sequences. Some tests related to 1's and 0's distributions are failed for complex XOR topologies, post processing techniques [47,48] can be used to improve the number's distribution and these tests. According to table 5.3 Topology III shows the better performance among the other probabilistic topologies. Therefore, it will be used in testing the probabilistic LFSR for different orders and polynomials.



(i) Accurate AND gates in the feedback



(ii) Probabilistic AND gates in the feedback

Figure 5.2: Histogram of the symbols is plotted for different XOR topologies with, (a) Accurate, (b) Probabilistic Simple, (c) Probabilistic Topology-I, (d) Probabilistic Topology-II, (e) Probabilistic Topology-III, and (f) Probabilistic Topology-IV

NIST tests	<i>Accurate</i>		<i>Simple</i>		<i>Topol₁</i>		<i>Topol₂</i>		<i>Topol₃</i>		<i>Topol₄</i>	
	<i>PV</i>	<i>PP</i>	<i>PV</i>	<i>PP</i>	<i>PV</i>	<i>PP</i>	<i>PV</i>	<i>PP</i>	<i>PV</i>	<i>PP</i>	<i>PV</i>	<i>PP</i>
Frequency	✓	1	✓	0.8	✗	0	✗	0	✗	0	✗	0
B. Frequency	✗	0	✗	0	✗	0	✗	0	✗	0	✗	0
C. Sums	✓	1	✗	0.6	✗	0	✗	0	✗	0	✗	0
Runs	✓	1	✓	1	✗	0.1	✗	0	✗	0	✗	0
Longest Run	✗	0	✗	0	✗	0.3	✗	0.4	✓	0.8	✗	0
Rank	✗	0	✗	0	✗	0	✗	0	✗	0	✗	0
FFT	✗	0	✗	0	✗	0	✗	0	✗	0	✗	0
N. O. Temp.	✗	1	✓	1	✓	1	✓	1	✓	1	✓	1
O. Temp.	✗	0.3	✓	1	✗	0.4	✗	0.2	✓	0.8	✓	1
Universal	✗	0	✗	1	✗	1	✗	1	✗	0	✗	1
Serial	✗	0	✗	0	✗	0	✗	0	✗	0.2	✗	0
L. Complex.	✗	0	✓	1	✓	1	✓	1	✓	1	✓	1

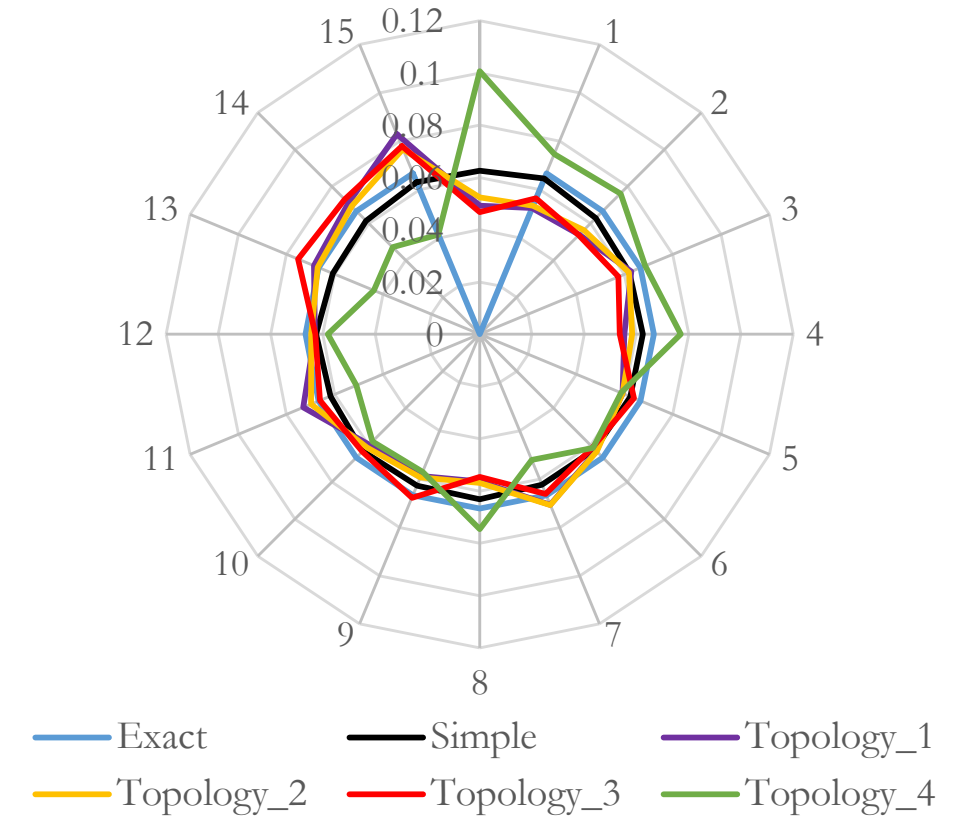
Table 5.2: The proportion value (PP), and the availability of p-values (PV) of the NIST tests showing results for probabilistic LFSR model with accurate AND gate in the feedback

NIST tests	<i>Accurate</i>		<i>Simple</i>		<i>Topol₁</i>		<i>Topol₂</i>		<i>Topol₃</i>		<i>Topol₄</i>	
	<i>PV</i>	<i>PP</i>	<i>PV</i>	<i>PP</i>	<i>PV</i>	<i>PP</i>	<i>PV</i>	<i>PP</i>	<i>PV</i>	<i>PP</i>	<i>PV</i>	<i>PP</i>
Frequency	✓	0.9	✓	1	✗	0	✗	0.1	✗	0	✗	0
B. Frequency	✗	0	✗	0	✗	0	✗	0	✗	0	✗	0
C. Sums	✓	0.9	✓	1	✗	0	✗	0	✗	0	✗	0
Runs	✓	0.9	✓	1	✗	0	✗	0.1	✗	0.1	✗	0
Longest Run	✗	0.2	✗	0.2	✗	0.4	✗	0.6	✓	0.8	✗	0
Rank	✗	0.5	✓	1	✓	0.9	✓	1	✓	1	✓	1
FFT	✗	0	✗	0	✗	0.2	✗	0.1	✓	0.9	✗	0.1
N. O. Temp.	✓	1	✓	1	✓	1	✓	1	✓	1	✓	1
O. Temp.	✓	0.9	✓	0.9	✓	0.8	✗	0.6	✓	0.8	✓	1
Universal	✗	1	✗	1	✗	1	✗	1	✗	1	✗	1
Serial	✗	0	✗	0	✗	0.3	✗	0.4	✓	0.8	✗	0
L. Complex.	✓	1	✓	1	✓	1	✓	1	✓	1	✓	0.9

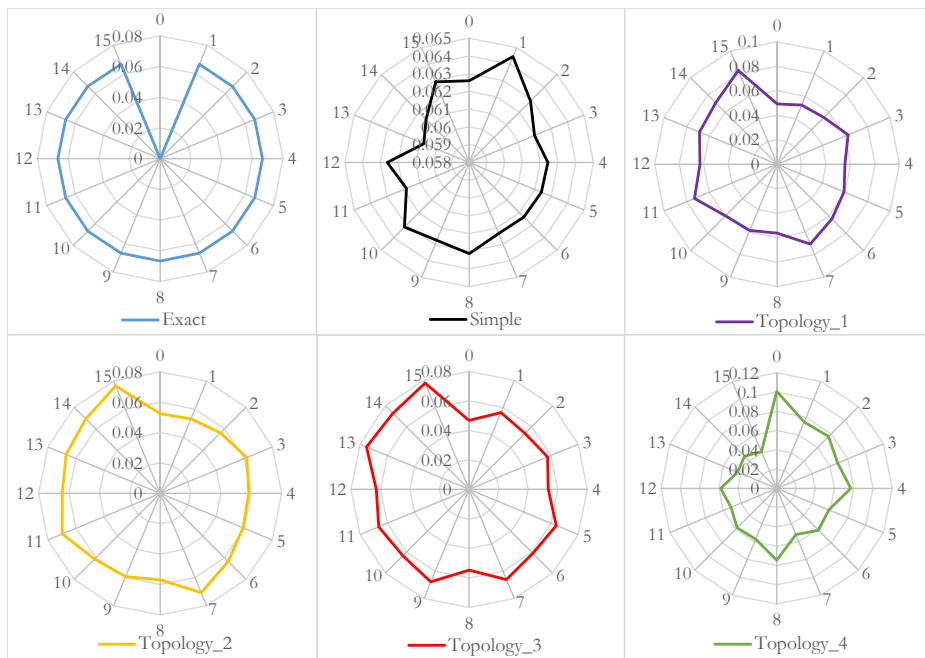
Table 5.3: The proportion value (PP), and the availability of p-values (PV) of the NIST tests showing results for probabilistic LFSR model with probabilistic AND gate in the feedback

5.3.1.3 Probabilities of Order-4 LFSR

This part investigates the probability of LFSR outputs for various probabilistic XOR models and accurate AND gates, and the statistical quantities of each topology. Order-4 is selected for this test to cover all of the generated data from [0 : 15]. Figure 5.3i show the number probabilities of all different XOR topologies in one diagram which show that the simple XOR model almost have a uniform probability for all the numbers beside accurate XOR. There is an advantage for the probabilistic models over the accurate one represented in the generation of '0' in the sequence with capability to cover from it, conversely to the accurate model which considers '0' as a forbidden state which will lock it from random numbers generation.



(i)



(ii)

Figure 5.3: Diagrams for number probabilities of order-4 programmable LFSR

The detailed diagrams for each topology is shown in figure 5.3ii which clearly presents the number probabilities is uniformly distributed for the accurate model with zero probability for '0'. The simple probabilistic model shows probability 0.06 for number '13' which is the minimum, maximum probability 0.064 for '1', and same probability for most of the numbers between {0.062,0.063}. These probabilities show almost the equally probability for 16 numbers which is 0.0625. Table 5.4 presents the probabilities of generated numbers and some statistical key values to describe these probabilities.

The mean value which interprets the average of these probabilities is the same for all data as they have the same number of iterations and the probabilities are distributed between 16 numbers. The midpoint of the data is explained by median value, it is clear the probabilities in LFSRs with accurate, simple probabilistic, complex topology-III XOR models are balanced which means the equally weighted probability is in the middle between the other probabilities in case they aren't distributed equally. Topologies I and II are still have better median rather than topology-IV. The assumption for these median values are related to the symmetry in the complex XOR gates. The standard deviation shows the variation in obtained probabilities around the mean, LFSR with the accurate model shows greater spread in these probabilities cause '0' can not determined in the sequence at all. In contrast the variation in LFSR with simple probabilistic model is the least cause the generated data are almost equally distributed.

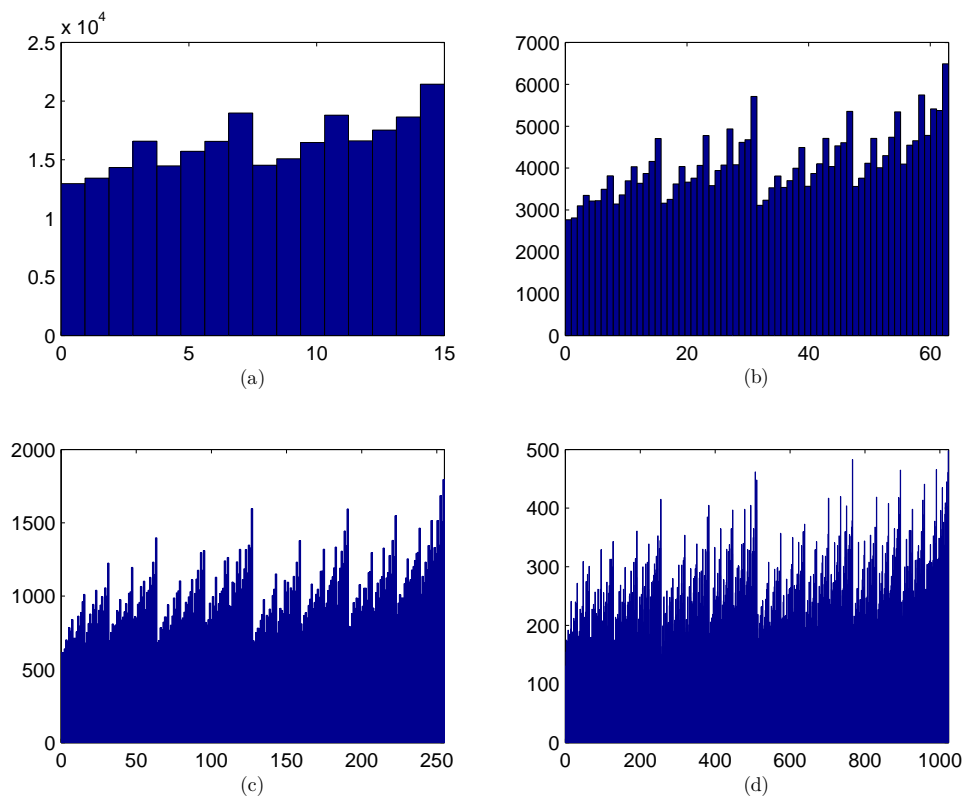
<i>Number</i>	<i>XOR_{acc}</i>	<i>XOR_{BB}</i>	<i>XOR_{Imp1}</i>	<i>XOR_{Imp2}</i>	<i>XOR_{Imp3}</i>	<i>XOR_{Imp4}</i>
0	0.000	0.063	0.049	0.052	0.047	0.101
1	0.067	0.064	0.052	0.053	0.056	0.075
2	0.067	0.063	0.054	0.056	0.054	0.076
3	0.067	0.062	0.063	0.062	0.058	0.069
4	0.067	0.062	0.055	0.058	0.054	0.077
5	0.067	0.062	0.059	0.059	0.064	0.058
6	0.067	0.062	0.063	0.063	0.061	0.061
7	0.067	0.062	0.071	0.071	0.066	0.052
8	0.067	0.063	0.056	0.057	0.055	0.075
9	0.067	0.063	0.059	0.059	0.068	0.057
10	0.067	0.063	0.060	0.061	0.064	0.058
11	0.067	0.062	0.073	0.070	0.066	0.051
12	0.067	0.063	0.063	0.065	0.063	0.058
13	0.067	0.061	0.069	0.067	0.075	0.044
14	0.067	0.061	0.071	0.069	0.073	0.047
15	0.067	0.063	0.083	0.077	0.078	0.041
Mean	0.063	0.063	0.063	0.063	0.063	0.063
St. Dev.	0.016	0.001	0.009	0.007	0.008	0.015
Median	0.067	0.063	0.061	0.061	0.063	0.058

Table 5.4: Number probabilities for different topologies of order-4 programmable LFSR with important statistical values

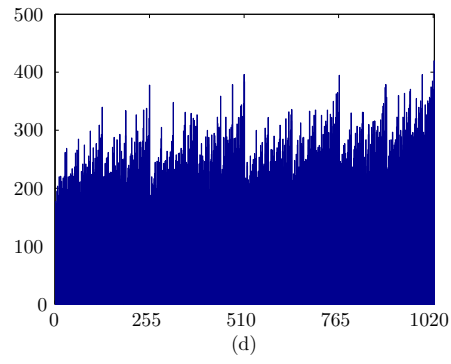
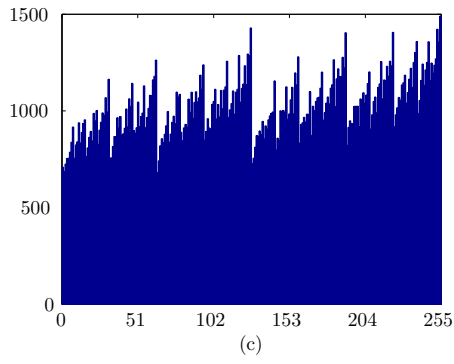
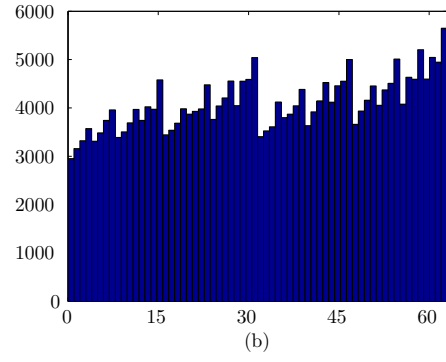
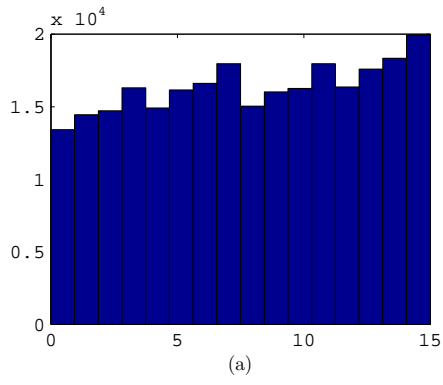
5.3.2 Different Orders

Programmable LFSRs with different complex probabilistic XOR gates tested for orders (4, 6, 8, and 10) to check if there is an effect for LFSR order width on the obtained histograms in different topologies section. Figure 5.4 shows the histograms for various complex XOR models used in the probabilistic LFSR. All of the histograms show same distribution behavior in different orders for the probabilistic models which confirms that there is no effect for LFSR order in the generated numbers distribution. On other words, the shown histograms are impacted by the probabilistic gates only, and there is no effect for the applied order in the distribution of generated data.

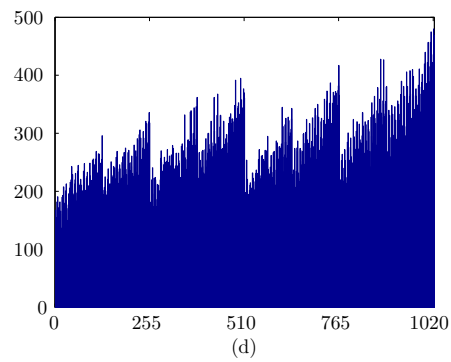
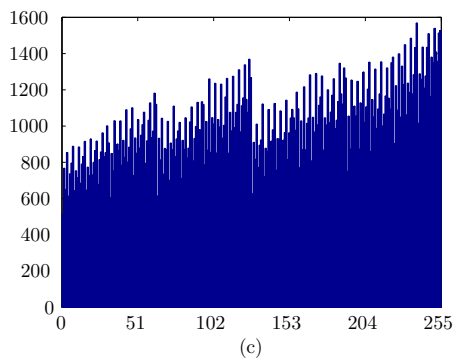
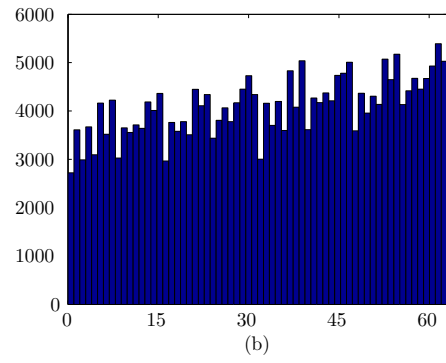
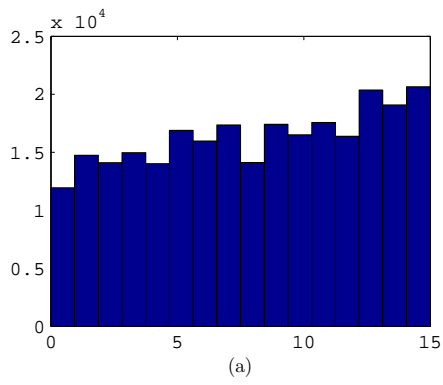
Same environment (number of bits and streams) used in testing different topologies is established for The NIST test results for these different orders of LFSR with the probabilistic complex XOR topology-III as shown in table 5.5. Linear complexity test for all the orders is perfect and this confirms the generated sequences of these topologies are complex to be treated as true random numbers. Most orders passed same NIST tests which means that the probabilistic LFSR implementation will be stable.



(i) XOR Topology-I



(ii) XOR Topology-II



(iii) XOR Topology-III

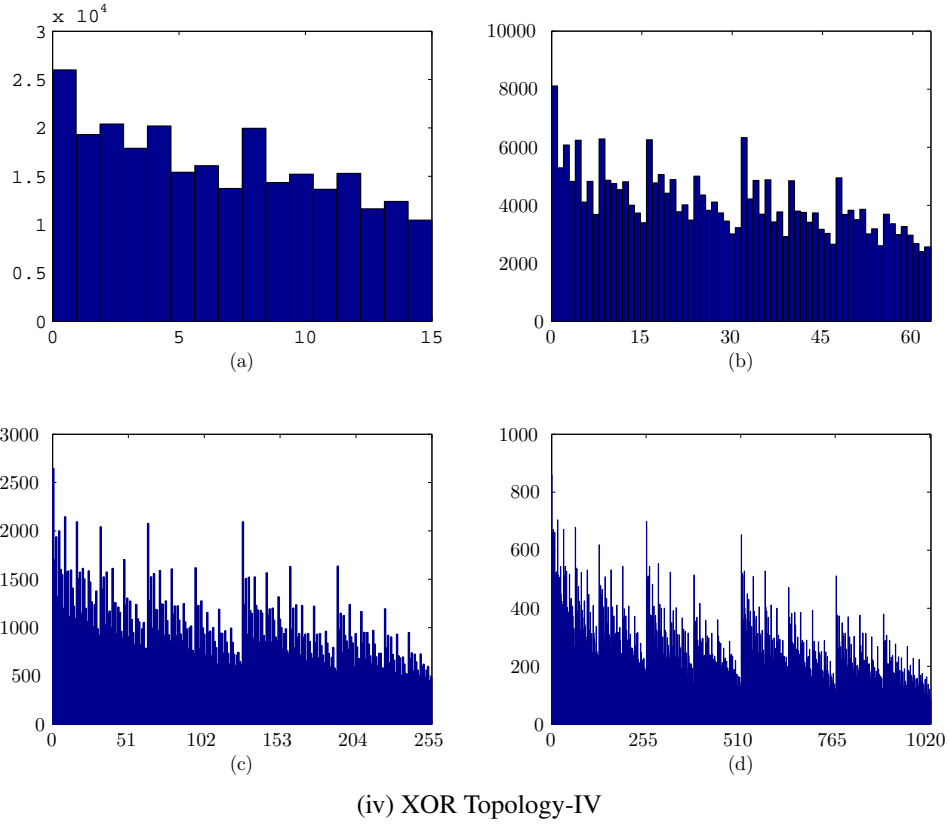


Figure 5.4: Histogram of the symbols is plotted for different PRNG Orders, (a) Order-4, (b) Order-6, (c) Order-8, and (d) Order-10

NIST tests	Order 4		Order 6		Order 8		Order 10	
	<i>PV</i>	<i>PP</i>	<i>PV</i>	<i>PP</i>	<i>PV</i>	<i>PP</i>	<i>PV</i>	<i>PP</i>
Frequency	✗	0	✗	0	✗	0	✗	0.1
B. Frequency	✗	0	✗	0	✗	0	✗	0
C. Sums	✗	0	✗	0	✗	0	✗	0
Runs	✗	0	✗	0.1	✗	0	✗	0
Longest Run	✗	0.1	✓	0.8	✗	1	✓	0.8
Rank	✗	1	✓	1	✓	1	✓	1
FFT	✗	0.3	✓	0.9	✓	1	✗	0.7
N. O. Temp.	✓	1	✓	1	✓	1	✓	1
O. Temp.	✓	0.5	✓	0.9	✓	0.9	✓	0.8
Universal	✗	0	✗	1	✗	1	✗	0
Serial	✗	0	✓	0.6	✓	0.6	✓	1
L. Complex.	✓	1	✓	1	✓	1	✓	1

Table 5.5: The proportion value (PP), and the availability of p-values (PV) of the NIST tests showing results for probabilistic LFSR model for different orders

5.3.3 Different Polynomials

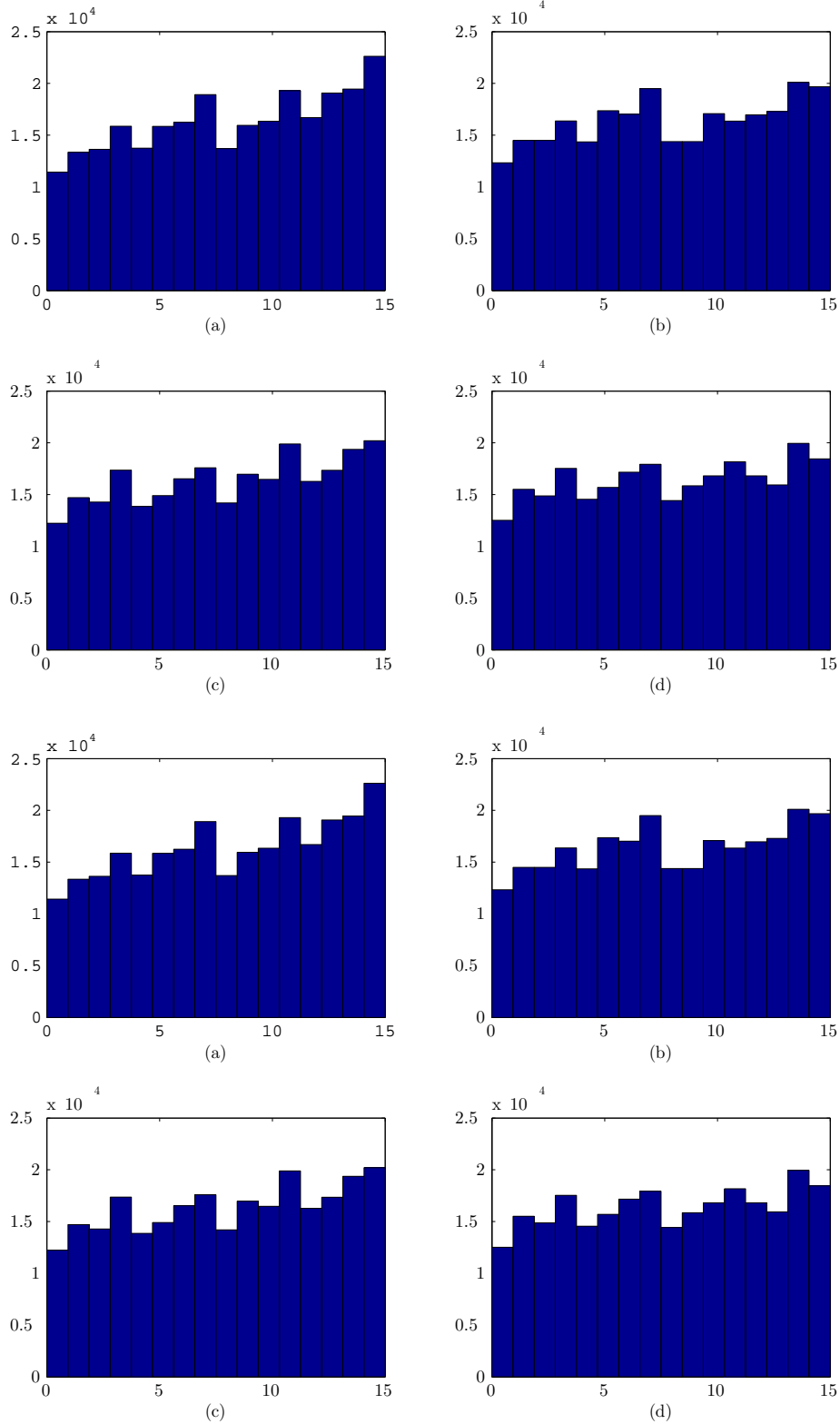


Figure 5.5: Histogram of the symbols is plotted for different polynomials, (a) 1000, (b) 1001, (c) 1010, (d) 1011, (e) 1100, (f) 1101, (g) 1110, and (h) 1111

Figure 5.5 presents the histograms for some polynomials to test order-4 LFSR with XOR topology-III. These histograms show almost same distribution of the data as 4 regions can be noticed. Any polynomial can be used in the sequence generation which means the LFSR is not restricted with specific polynomial to achieve maximal sequence length as while using LFSR in PRNG. Using accurate AND logic gates in the feedback may be useful to control the propagation of probabilistic XOR through the chain of LFSR.

5.4 Conclusion

This chapter introduced programmable LFSR technique which is a method to implement PRNG. It was tested by replacing its logic gates by the proposed probabilistic models for simple (AND, XOR), and complex (Four XOR topologies) to check the effect of these models on the generated sequences. Three different histograms were obtained to assess the probabilistic LFSR model. Also, NIST statistical tests are applied on the model which show better results specially in linear complexity feature rather than the accurate model.

Chapter 6

Conclusion and Future Work

6.1 Summary of The Work

The main objective of this thesis is to present a model that can describe inaccurate approximate gates and imprecise probabilistic gates in various inexact computing systems. Such model can be useful in modeling the effect of these computing systems in EDA tools, which will help in designing different circuits and implementation on abstract level based on either injecting the probability distribution for physical error of the used device, or mapping the wrong inputs combination for a gate which replace the accurate one. In this thesis the following objectives were accomplished:

- A new metric (OPE) that helps in the model definition by giving a percentage for the inexact output results compared with accurate model.
- An abstract model for inexact gates which depends on a source of error that is injected in the model. This error can be deterministic related to input combinations or non-deterministic as a probability distribution function that describes the physical behavior in a device.
- The proposed model is extended to proof its validity on basic simple gates and four different complex XOR topologies. Simple gates show the correctness of the model by generating OPEs same as CDFs of the fitted PDFs for the injected probabilistic distributions. Also, complex gates extend the exactness of the model to show almost identical OPEs for the same type gates under the same conditions as the previous OPE shape and stage location.
- The inexact models utilized in chain of logic gates from the same type. OPE for the final stage show performance same as the probabilistic distribution injected as error in the logic gate.
- Different application as RCA and RNG are tested using the proposed probabilistic models to check their performance. The inexact RCA models show a correct behavior in MED for two different types of adders: LOA as an approximate techniques, PFA as a probabilistic technique.
- The programmable LFSR which is the key element in PRNG implementation is utilized by adding the imprecise XOR models rather than the accurate ones, tested for different topologies, orders, and polynomials to check the generated histograms and the effects of these characteristics on the data distribution, and examined for NIST tests that confirm a higher complexity and true randomness in the generated sequences.

6.2 Impact on The Digital Flow

There is a great motivation to utilize the inexact effects in latest technologies according to the road maps reports issued by International Technology Roadmap for Semiconductors (IRTS) [62], and study effects on CAD digital design flow from different perspectives, as follows:

1. **RTL Implementation:** The proposed model may be so useful in RTL implementation. It is so promising to integrate different logic gates in a library or package for HDL languages. Once adding this library, the inexact gates can be used in a simple way to model different real data sets of these gates on RTL level.
2. **RTL Verification:** Modeling the inaccuracy in different verification methodologies is investigated specially how to use monitor and scoreboard in checking accuracy of the generated outputs. This may need a new definitions and parameters to make the verification methodology behaves in the correct way.
3. **Standard Cells:** It is expected that using these inexact logic gates will make the standard cell definition different as there is a need to model the effect of error, and if it will change the static timing analysis in the internal paths which may lead to different slacks and critical paths cause of the inaccuracy.
4. **Gate Level Synthesis:** Different Algorithms were proposed to synthesize the design such as: automatic pruning tool which depends on editing the netlist according to studying the effect of switching activity, and ranking the nodes according to their activity product.
5. **Place and Route:** Modeling power of the placed cells, and the delays of interconnects between these cells. As interconnect becomes a dominant factor in this step which requires different wire models.

6.3 Future Work

In this thesis the modeling of different inexact computing techniques was studied. The suggested future work is:

- Extend the model to determine the number of wrong 0's and 1's then mapping them to probabilities which can be involved in defining OPE equations for each gate based on these probabilities.
- Apply extracted data sets that represents real distributions for switching probability of devices as error function in the model, this will add more reliability to the adder.
- Use post processing techniques to random number generator to get uniform results.
- Model the delay and power to mimic device in real operation.

References

- [1] S. Mittal, “A survey of techniques for approximate computing,” *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, p. 62, 2016.
- [2] A. K. Mishra, R. Barik, and S. Paul, “iact: A software-hardware framework for understanding the scope of approximate computing,” in *Workshop on Approximate Computing Across the System Stack (WACAS)*, 2014.
- [3] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, *et al.*, “In-datacenter performance analysis of a tensor processing unit,” in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pp. 1–12, ACM, 2017.
- [4] T. Moreau, J. San Miguel, M. Wyse, J. Bornholt, A. Alaghi, L. Ceze, N. E. Jerger, and A. Sampson, “A taxonomy of general purpose approximate computing techniques,” *IEEE Embedded Systems Letters*, vol. 10, no. 1, pp. 2–5, 2018.
- [5] S. Misailovic, S. Sidiroglou, H. Hoffmann, and M. Rinard, “Quality of service profiling,” in *Software Engineering, 2010 ACM/IEEE 32nd International Conference on*, vol. 1, pp. 25–34, IEEE, 2010.
- [6] S. Sidiroglou-Douskos, S. Misailovic, H. Hoffmann, and M. Rinard, “Managing performance vs. accuracy trade-offs with loop perforation,” in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pp. 124–134, ACM, 2011.
- [7] E. F. Moore and C. E. Shannon, “Reliable circuits using less reliable relays,” *Journal of the Franklin Institute*, vol. 262, no. 3, pp. 191–208, 1956.
- [8] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, “Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 4, pp. 850–862, 2010.
- [9] B. E. Akgul, L. N. Chakrapani, P. Korkmaz, and K. V. Palem, “Probabilistic cmos technology: A survey and future directions,” in *Very Large Scale Integration, 2006 IFIP International Conference on*, pp. 1–6, IEEE, 2006.
- [10] “Numerical analysis for engineering.” <https://ece.uwaterloo.ca/~dwharder/NumericalAnalysis/01Error/PrecisionAccuracy/>.
- [11] L. Monroe, “Probabilistic and approximate computing,” Dec 2015. <https://rebootingcomputing.ieee.org/images/files/pdf/RCS4MonroeThu845.pdf>.
- [12] J. Von Neumann, “Probabilistic logics and the synthesis of reliable organisms from unreliable components,” *Automata studies*, vol. 34, pp. 43–98, 1956.

- [13] B. R. Gaines, "Stochastic computing systems," in *Advances in information systems science*, pp. 37–172, Springer, 1969.
- [14] W. Poppelbaum, C. Afuso, and J. Esch, "Stochastic computing elements and systems," in *Proceedings of the November 14-16, 1967, fall joint computer conference*, pp. 635–644, ACM, 1967.
- [15] B. R. Gaines, "Stochastic computing," in *Proceedings of the April 18-20, 1967, spring joint computer conference*, pp. 149–156, ACM, 1967.
- [16] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," *ACM Transactions on Embedded computing systems (TECS)*, vol. 12, no. 2s, p. 92, 2013.
- [17] J. Han, H. Chen, J. Liang, P. Zhu, Z. Yang, and F. Lombardi, "A stochastic computational approach for accurate and efficient reliability evaluation," *IEEE Transactions on Computers*, vol. 63, no. 6, pp. 1336–1350, 2014.
- [18] I. Present, "Cramming more components onto integrated circuits," *Readings in computer architecture*, vol. 56, 2000.
- [19] J. M. Shalf and R. Leland, "Computing beyond moore's law," *Computer*, vol. 48, no. 12, pp. 14–23, 2015.
- [20] K. Natori and N. Sano, "Scaling limit of digital circuits due to thermal noise," *Journal of applied physics*, vol. 83, no. 10, pp. 5019–5024, 1998.
- [21] L. B. Kish, "End of moore's law: thermal (noise) death of integration in micro and nano electronics," *Physics Letters A*, vol. 305, no. 3-4, pp. 144–149, 2002.
- [22] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *Proceedings of the 40th annual Design Automation Conference*, pp. 338–342, ACM, 2003.
- [23] B. E. Akgul, L. N. Chakrapani, P. Korkmaz, and K. V. Palem, "Probabilistic cmos technology: A survey and future directions," in *Very Large Scale Integration, 2006 IFIP International Conference on*, pp. 1–6, IEEE, 2006.
- [24] S. Hamdioui, S. Kvatinsky, G. Cauwenberghs, L. Xie, N. Wald, S. Joshi, H. M. Elsayed, H. Corporaal, and K. Bertels, "Memristor for computing: Myth or reality?," in *Proceedings of the Conference on Design, Automation & Test in Europe*, pp. 722–731, European Design and Automation Association, 2017.
- [25] G. Papandroulidakis, I. Vourkas, N. Vasileiadis, and G. C. Sirakoulis, "Boolean logic operations and computing circuits based on memristors," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 12, pp. 972–976, 2014.
- [26] I. Vourkas and G. C. Sirakoulis, "Emerging memristor-based logic circuit design approaches: A review," *IEEE Circuits and Systems Magazine*, vol. 16, no. 3, pp. 15–30, 2016.
- [27] I. Vourkas and G. C. Sirakoulis, "A novel design and modeling paradigm for memristor-based crossbar circuits," *IEEE Transactions on Nanotechnology*, vol. 11, no. 6, pp. 1151–1159, 2012.

- [28] S. Gaba, P. Knag, Z. Zhang, and W. Lu, “Memristive devices for stochastic computing,” in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*, pp. 2592–2595, IEEE, 2014.
- [29] R. Naous and K. N. Salama, “Approximate computing with stochastic memristors,” in *CNNA 2016; 15th International Workshop on Cellular Nanoscale Networks and their Applications; Proceedings of*, pp. 1–2, VDE, 2016.
- [30] E. Linn, R. Rosezin, S. Tappertzhofen, U. Böttger, and R. Waser, “Beyond von neumann—logic operations in passive crossbar arrays alongside memory operations,” *Nanotechnology*, vol. 23, no. 30, p. 305205, 2012.
- [31] H. Alahmadi, *Memristive Probabilistic Computing*. PhD thesis, KAUST, 2017.
- [32] S. H. Jo, K.-H. Kim, and W. Lu, “Programmable resistance switching in nanoscale two-terminal devices,” *Nano letters*, vol. 9, no. 1, pp. 496–500, 2008.
- [33] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, “Impact: imprecise adders for low-power approximate computing,” in *Proceedings of the 17th IEEE/ACM international symposium on Low-power electronics and design*, pp. 409–414, IEEE Press, 2011.
- [34] Y. Liu, T. Zhang, and K. K. Parhi, “Computation error analysis in digital signal processing systems with overscaled supply voltage,” *IEEE transactions on very large scale integration (VLSI) systems*, vol. 18, no. 4, pp. 517–526, 2010.
- [35] D. Mohapatra, V. K. Chippa, A. Raghunathan, and K. Roy, “Design of voltage-scalable meta-functions for approximate computing,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, pp. 1–6, IEEE, 2011.
- [36] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, “Approximate xor/xnor-based adders for inexact computing,” in *Nanotechnology (IEEE-NANO), 2013 13th IEEE Conference on*, pp. 690–693, IEEE, 2013.
- [37] Z. Yang, J. Han, and F. Lombardi, “Transmission gate-based approximate adders for inexact computing,” in *Nanoscale Architectures (NANOARCH), 2015 IEEE/ACM International Symposium on*, pp. 145–150, IEEE, 2015.
- [38] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, “Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 4, pp. 850–862, 2010.
- [39] J. Liang, J. Han, and F. Lombardi, “New metrics for the reliability of approximate and probabilistic adders,” *IEEE Transactions on computers*, vol. 99, no. 1, 2012.
- [40] J. Liang, J. Han, and F. Lombardi, “On the reliable performance of sequential adders for soft computing,” in *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2011 IEEE International Symposium on*, pp. 3–10, IEEE, 2011.
- [41] C. K. Koc, “About cryptographic engineering,” in *Cryptographic engineering*, pp. 1–4, Springer, 2009.

- [42] I. Vasyltsov, E. Hambarzumyan, Y.-S. Kim, and B. Karpinsky, “Fast digital trng based on metastable ring oscillator,” in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 164–180, Springer, 2008.
- [43] A. Cherkaoui, V. Fischer, L. Fesquet, and A. Aubert, “A very high speed true random number generator with entropy assessment,” in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 179–196, Springer, 2013.
- [44] K. H. Lundberg, “Noise sources in bulk cmos,” *Unpublished paper*, vol. 3, 2002.
- [45] B. Jun and P. Kocher, “The intel random number generator,” *Cryptography Research Inc. white paper*, 1999.
- [46] J.-L. Danger, S. Guilley, and P. Hoogvorst, “High speed true random number generator based on open loop structures in fpgas,” *Microelectronics journal*, vol. 40, no. 11, pp. 1650–1656, 2009.
- [47] R. Shaltiel, “An introduction to randomness extractors,” in *International Colloquium on Automata, Languages, and Programming*, pp. 21–41, Springer, 2011.
- [48] B. Barak, R. Shaltiel, and E. Tromer, “True random number generators secure in a changing environment,” in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 166–180, Springer, 2003.
- [49] J. Eichenauer and J. Lehn, “A non-linear congruential pseudo random number generator,” *Statistische Hefte*, vol. 27, no. 1, pp. 315–326, 1986.
- [50] J. Liang, J. Han, and F. Lombardi, “On the reliable performance of sequential adders for soft computing,” in *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2011 IEEE International Symposium on*, pp. 3–10, IEEE, 2011.
- [51] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, “Approximate xor/xnor-based adders for inexact computing,” in *Nanotechnology (IEEE-NANO), 2013 13th IEEE Conference on*, pp. 690–693, IEEE, 2013.
- [52] J. Han, H. Chen, J. Liang, P. Zhu, Z. Yang, and F. Lombardi, “A stochastic computational approach for accurate and efficient reliability evaluation,” *IEEE Transactions on Computers*, vol. 63, no. 6, pp. 1336–1350, 2014.
- [53] A. Alaghi and J. P. Hayes, “Survey of stochastic computing,” *ACM Transactions on Embedded computing systems (TECS)*, vol. 12, no. 2s, p. 92, 2013.
- [54] J. S. Marron and M. P. Wand, “Exact mean integrated squared error,” *The Annals of Statistics*, pp. 712–736, 1992.
- [55] C. E. Rasmussen, “The infinite gaussian mixture model,” in *Advances in neural information processing systems*, pp. 554–560, 2000.
- [56] “Linear-feedback shift register - wikipedia.” https://en.wikipedia.org/wiki/Linear-feedback_shift_register.
- [57] M. Murase, “Linear feedback shift register,” Feb. 18 1992. US Patent 5,090,035.

- [58] “Randomness tests: A literature survey.” <http://www.ciphersbyritter.com/RES/RANDTEST.HTM>.
- [59] R. G. Brown, “dieharder.” <http://webhome.phy.duke.edu/~rgb/General/dieharder.php>.
- [60] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker, “A statistical test suite for random and pseudorandom number generators for cryptographic applications,” tech. rep., Booz-Allen and Hamilton Inc Mclean Va, 2001.
- [61] P. FIPS, “140-2,” *Security Requirements for Cryptographic Modules*, vol. 25, 2001.
- [62] “International technology roadmap for semiconductors.” <http://www.itrs2.net/>.

Appendix A

Memristive Logic Gates

This appendix shows the truth tables for sequential OR, NAND, and NOR as memristive logic gates in deterministic mode. Also, it presents the analysis diagrams for these gates in probabilistic mode.

A.1 Deterministic Sequential logic

Operation	Terminal	Cycle 1	Cycle 2	Cycle 3
OR	T_1	0	in_1	in_2
	T_2	1	0	0
in_2	in_1	$State_0$	$State_1$	$State_2$
0	0	OFF	No change	No change
0	1	OFF	ON	No change
1	0	OFF	No change	ON
1	1	OFF	ON	ON

Table A.1: The truth table for sequential OR logic gate

Operation	Terminal	Cycle 1	Cycle 2	Cycle 3
AND	T_1	0	1	1
	T_2	1	in_1	in_2
in_2	in_1	$State_0$	$State_1$	$State_2$
0	0	OFF	ON	ON
0	1	OFF	No change	ON
1	0	OFF	ON	No change
1	1	OFF	No change	No change

Table A.2: The truth table for sequential NAND logic gate

Operation	Terminal	Cycle 1	Cycle 2	Cycle 3
AND	T_1	0	1	0
	T_2	1	in_1	in_2
	in_2	in_1	$State_0$	$State_1$
	0	0	OFF	ON
	0	1	OFF	No change
	1	0	OFF	ON
	1	1	OFF	No change

Table A.3: The truth table for sequential NOR logic gate

Tables A.1, A.2, and A.3 show the the truth tables for sequential OR, NAND, and NOR respectively as memristive logic gates in deterministic mode.

A.2 Probabilistic Sequential logic

The detailed analysis diagrams for the different applied inputs in probabilistic sequential OR, NAND, and NOR gates through the operating cycles are presented in figures A.1, A.2, and A.3 respectively.

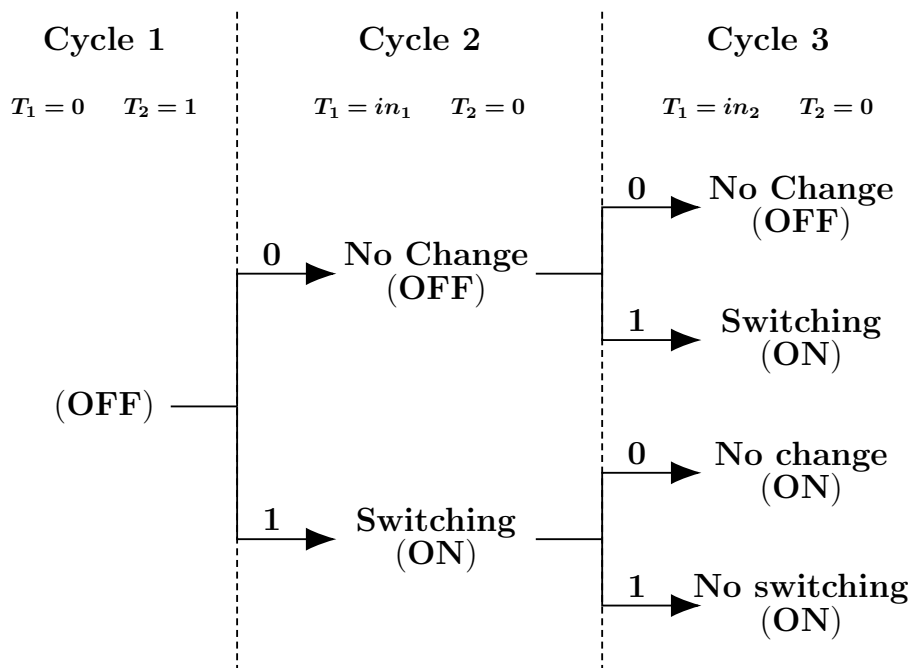


Figure A.1: Diagram for the applied inputs through the cycles of sequential OR logic gate

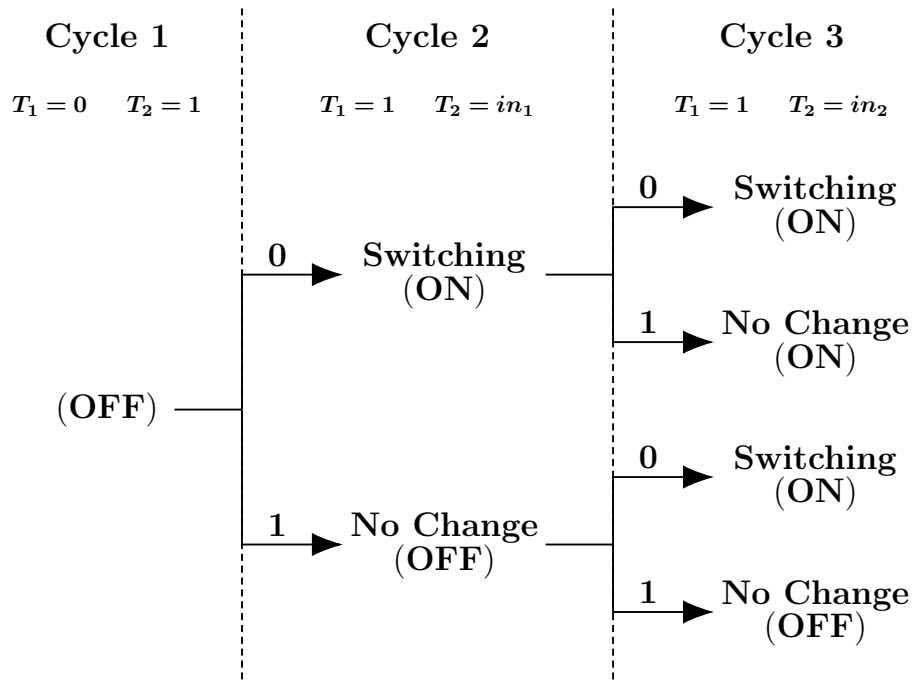


Figure A.2: Diagram for the applied inputs through the cycles of sequential NAND logic gate

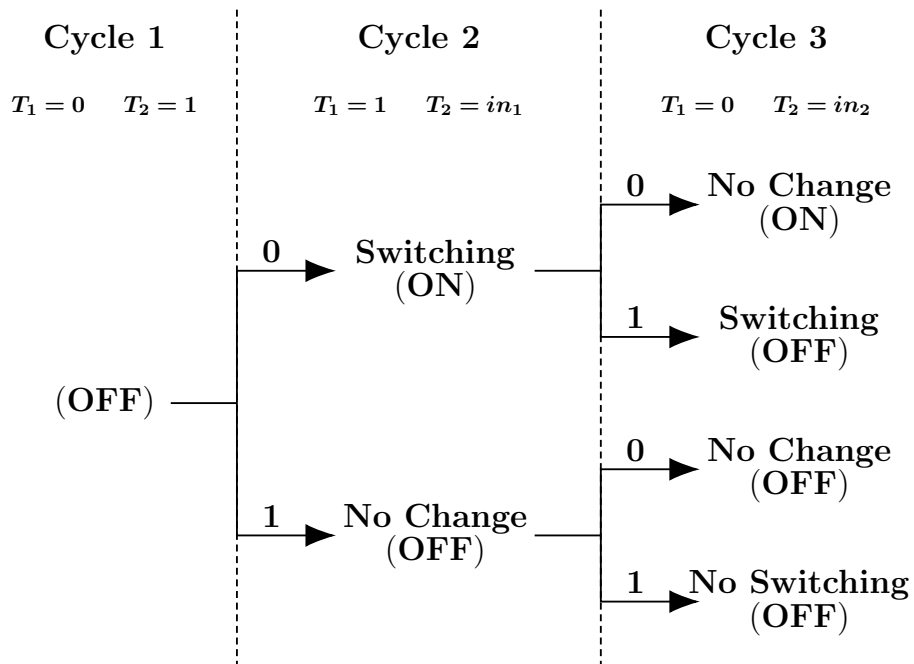


Figure A.3: Diagram for the applied inputs through the cycles of sequential NOR logic gate

الملخص

أصبحت لأنظمة الحوسبة أهمية كبيرة في التعامل مع الوسائط والبيانات. النتيجة الدقيقة ليست ضرورية في معالجة الوسائط مثل الصوت والصورة والفيديو والرسومات والتعرف على الأنماط واستخراج البيانات. في هذه التطبيقات، يكفي الحصول على نتيجة تقريبية أو أقل من المستوى الأمثل. على سبيل المثال، لا يعتبر تعرف الإنسان حساسًا للتغيرات عالية التردد في تطبيقات معالجة الإشارات. أيضًا، ليست هناك حاجة لتحديد كمية الإشارة في كميات أعلى بمجرد أن تكون أرضية الضوضاء مقبولة.

هناك العديد من المصادر التي تتحمل النتائج غير الدقيقة مثل قدرة الدماغ البشري على اكتشاف المعلومات المفقودة، وتكرار بيانات الإدخال التي تسمح للخوارزمية أن تكون كافية على الرغم من عدم كفاءتها. نُبِت أن هذا التحمل للنتائج غير الدقيقة مفيد في تقليل الطاقة لهذه التطبيقات. يتم تقديم أنظمة الحوسبة المختلفة لهذا الغرض كحوسبة تقريبية، احتمالية، عشوائية.

في هذه الرسالة، يتم تقديم مراجعة بحثية لنماذج مختلفة من الأخطاء في أنظمة الحوسبة. أيضًا، يتم إجراء تحقيق لمعرفة تأثير السلوك الاحتمالي لبعض المكونات الإلكترونية مثل مقاوم الذاكرة (Memristor) وشبه موصل أكسيد الفلز المكمل الاحتمالي (PCMOS). نقترح نموذجًا تجريديًا لبوابات غير الدقيقة مع قياس جديد للخطأ في النسبة المئوية للمخرجات، والذي يعطي نتيجة مئوية لإجمالي المخرجات غير الدقيقة مقارنة بالنواتج المولدة في بوابات منطقية دقيقة. يمكن أن يكون هذا النموذج مفيدًا لأدوات لتصميم الإلكتروني الألي (EDA) في نمذجة تأثير بوابات المنطق غير الدقيقة على أداء التطبيقات المعقدة.

نناقش نماذج البوابة التقريبية والعشوائية المقترحة، وكذلك نمذجة البوابات البسيطة المختلفة، وأربعة بنيات معقدة من بوابات عدم التطابق (XOR). علاوة على ذلك، يتم توفير تحليل مفصل لسلاسل مختلفة مثل بوابة العاكس (Inverter)، وبوابات عدم التطابق (XOR) متصلة بشكل تسلسلي لفحص التبعيات على تلك السلاسل. ونناقش تطبيقين مختلفين؛ تحقيق نموذج حسابي للجمع، وأداء مولد الأرقام العشوائية الحقيقية القابلة للبرمجة باستخدام بوابات بسيطة من عملية عدم التطابق (XOR) العشوائية. أخيرًا، تم ذكر بعض المجالات المحتملة للعمل في المستقبل.



محمد علاء الدين خليل أبو العلا

١٩٩٢١١٠١٢٤

مصري

٢٠١٤١١٠١١

٢٠١٩

هندسة الإلكترونيات والاتصالات الكهربائية
ماجستير العلوم

مهندس:

تاريخ الميلاد:

الجنسية:

تاريخ التسجيل:

تاريخ المنح:

القسم:

الدرجة:

المشرفون:

أ.م.د. أحمد خطاب فتحى خطاب

أ.د. حسام على حسن فهمى

أ.د. عمرو جلال الدين أحمد وصال

المتحنون:

(المشرف الرئيسي)

أ.م.د. أحمد خطاب فتحى خطاب

(مشرف)

أ.د. حسام على حسن فهمى

(مشرف)

أ.د. عمرو جلال الدين أحمد وصال

(المتحن الداخلى)

أ.د. أحمد حسين محمد خليل

(المتحن الخارجى)

أ.د. محمد واثق الخراشى

(الأستاذ بكلية الهندسة – جامعة عين شمس)

عنوان الرسالة:

نموذج جديد للبوابات الغير دقيقة وتحليل لتطبيق شبه مولد أرقام عشوائية حقيقية

الكلمات الدالة:

النمذجة، الخوارزمية، العائلات المنطقية، الدوائر الحسابية، مولدات الأرقام العشوائية.

ملخص الرسالة:

يهدف هذا العمل إلى اقتراح نموذج جديد للبوابات المنطقية غير الدقيقة. تم التحقيق في استخدام هذه البوابات المنطقية كخلايا بسيطة في دوائر معقدة، وكذلك التدقيق في أداء التطبيقات المختلفة مثل دوائر الجمع ومولدات الأرقام العشوائية. يساعد ذلك العمل في تجريد الخطأ في البوابات المنطقية من خلال نمذجة ذلك الخطأ في دالة كثافة الاحتمال، وحقن هذا الخطأ في إخراج البوابة. كما يقدم دليلاً لحل مشكلة عدم التوحيد في مولد الأرقام العشوائية للعمل في المستقبل.

نموذج جديد للبوابات الغير دقيقة وتحليل لتطبيق شبه مولد
أرقام عشوائية حقيقية

إعداد

محمد علاء الدين خليل أبو العلا

رسالة مقدمة إلى كلية الهندسة - جامعة القاهرة
كجزء من متطلبات الحصول على درجة
ماجستير العلوم
في
هندسة الإلكترونيات والاتصالات الكهربائية

يعتمد من لجنة الممتحنين:

المشرف الرئيسي

أ.م.د. أحمد خطاب فتحى خطاب

مشرف

أ.د. حسام على حسن فهمى

مشرف

أ.د. عمرو جلال الدين أحمد وصال

الممتحن الداخلي

أ.د. أحمد حسين محمد خليل

الممتحن الخارجي

أ.د. محمد واثق الخراشى

الأستاذ بكلية الهندسة - جامعة عين شمس

كلية الهندسة - جامعة القاهرة

الجيزة - جمهورية مصر العربية

٢٠١٩

نموذج جديد للبوابة الغير دقيقة وتحليل لتطبيق شبه مولد
أرقام عشوائية حقيقية

إعداد

محمد علاء الدين خليل أبو العلا

رسالة مقدمة إلى كلية الهندسة - جامعة القاهرة
كجزء من متطلبات الحصول على درجة
ماجستير العلوم
في
هندسة الإلكترونيات والاتصالات الكهربائية

تحت إشراف

أ.د. حسام على حسن فهمي

أ.م.د. أحمد خطاب فتحي خطاب

أستاذ

قسم هندسة الإلكترونيات والاتصالات الكهربائية
كلية الهندسة - جامعة القاهرة

أستاذ مساعد

قسم هندسة الإلكترونيات والاتصالات الكهربائية
كلية الهندسة - جامعة القاهرة

أ.د. عمرو جلال الدين أحمد وصال

أستاذ

قسم هندسة الحاسبات
كلية الهندسة - جامعة القاهرة

كلية الهندسة - جامعة القاهرة

الجيزة - جمهورية مصر العربية

٢٠١٩



نموذج جديد للبوابات الغير دقيقة وتحليل لتطبيق شبه مولد
أرقام عشوائية حقيقية

إعداد

محمد علاء الدين خليل أبو العلا

رسالة مقدمة إلى كلية الهندسة - جامعة القاهرة
كجزء من متطلبات الحصول على درجة
ماجستير العلوم
في
هندسة الإلكترونيات والاتصالات الكهربائية

كلية الهندسة - جامعة القاهرة
الجيزة - جمهورية مصر العربية

٢٠١٩