ERROR ANALYSIS OF CORRELATION DEPENDENT STOCHASTIC CIRCUIT

Thesis is submitted in the partial fulfilment of the requirements for the degree of

MASTER IN ELECTRONICS AND

TELECOMMUNICATION ENGINEERING

by

SUJIT TUDU

Class Roll Number: 002010702004

Registration Number: 127703 of 2014-2015

Examination Roll Number: M4ETC22004 of 2020-22

Under the Guidance of

Prof. MRINAL KANTI NASKAR

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

KOLKATA-700032

August, 2022

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

This is to certify that **Mr. SUJIT TUDU** (Class Roll No. **002010702004** and Registration No. **127703** of **2014-15**) bearing Examination Roll No. **M4ETC22004** has completed and submitted his thesis entitled, "**ERROR ANALYSIS OF CORRELATION DEPENDENT STOCHASTIC CIRCUIT**", in partial fulfilment of the requirements for the degree of "**Master in Computer Engineering**" under Electronics and Telecommunication Engineering Department of Jadavpur University. The thesis work has been carried out by him under my guidance and supervision. The project, in my opinion, is worthy of its acceptance for the partial fulfilment of the requirement for the degree of Master of Electronics and Telecommunication Engineering.

SUPERVISOR
Prof. Mrinal Kanti Naskar
Professor
Department of Electronics and Telecommunication Engineering
Jadavpur University
Kolkata -700032

Prof. Manotosh Biswas
Head of the Department
Electronics and Telecommunication Engineering
Jadavpur University
Kolkata - 700032

Prof. Chandan Mazumdar
Dean
Faculty of Engineering and Technology
Jadavpur University
Kolkata - 700032

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

CERTIFICATE OF APPROVAL

The foregoing thesis, entitled "**ERROR ANALYSIS OF CORRELATION DEPENDENT STOCHASTIC CIRCUIT**", is hereby approved as a creditable study of "**Master in Computer Engineering**" under Electronics and Telecommunication Engineering department and presented by **Mr. SUJIT TUDU** bearing Examination Roll No. **M4ETC22004** in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion therein but approve this thesis only for the purpose for which it is submitted.

Committee on Final examination for evaluation of thesis:

Signature of Examiners

Additional Examiner

Supervisor

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains literature survey and original work by the undersigned candidate as part of his Master of Electronics and Telecommunication studies.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name	**: SUJIT TUDU**
Examination Roll No	**: M4ETC22004**
Class Roll No	**: 002010702004**
Thesis Title	**: Error Analysis of Correlation Dependent Stochastic Circuit**

Signature of the Candidate

Acknowledgement

Firstly, I would like to express my special gratitude to my project guide, Prof. Mrinal Kanti Naskar, for his supervision, invaluable suggestions, and continuous encouragement throughout the entire course of the research work. His valuable guidance and feedback has helped me in completing this project.

I am very grateful to Mr. Arijit Chowdhury (Research Scholar, Jadavpur University) for his necessary suggestions and valuable help during my project work. I also wish to express my thanks to all my teachers who taught me during the tenure of my course work.

Above all, I extend my deepest gratitude to my parents for their invaluable love, affection, encouragement, and support and for their struggle to make me a successful person in spite of many obstacles. Their help continues to guide me in my life.

Lastly, I would like to thank Prof. Manotosh Biswas, head of the department, Electronics and Telecommunication Engineering department, Jadavpur University, for his kind cooperation throughout the entire course.

SUJIT TUDU

Abstract

Stochastic computing has emerged as an important research area because of its low hardware requirements and error tolerance. In stochastic computing, many things are important, like random number generation, derandomizer unit, correlation among input values, etc. Here, each number is represented as a sequence of random bit streams where the probability of 1 is taken as its value. Stochastic circuits perform well under severe error conditions. Here, the design of stochastic computing circuits is such that many errors have a very negligible impact because if one bit flip occurs in the entire bit stream, then there is a negligible impact on the result. However, this is relevant that noisy channels, such as binary symmetric channels and binary asymmetric channels, will impact the processes of stochastic computing. We can also expect some robustness against those types of errors in stochastic circuits.

However, Alaghi showed that we can expert correlation to make some complicated functions using very simple circuits. In this thesis, it is shown that conventional basic circuits like AND, OR, and XOR can be used with the help of correlation to get certain functions like absolute difference, multiplication, addition, etc. This type of circuit depends on correlation. Hence, apart from channel noise, there is an additional source of noise that is due to the correlation value. Suppose we generate an approximate circuit for a complex function that is dependent on the correlation value to be +1, 0, or in a minor case, say 0.6. But, there is no guarantee that the generation process is capable of generating exact values with high precision enough to meet the exact correlation. Hence, the input sequence is slightly different from the correlation needed to achieve the complex functionality. Here, the functions of the circuits depend on the correlation itself. An XOR gate is used to perform absolute subtraction between two bit streams. There, correlation plays an important role because we want the inputs to be perfectly correlated, which will be +1. It might so happen that the correlation gets slightly reduced. That result of a slight error will impact the calculation of absolute subtraction. So, the same error will propagate everywhere this output is used. Whenever such a scenario happens, the output will be slightly erroneous and any other functionality that uses the same output will see the same error propagate through multiple stages of the stochastic circuit. Hence, the output of a circuit design exploiting correlation is dependent on the particular correlation value of the input sequences. Thus, it is important to study how the output performs when the actual correlation between the inputs varies. This thesis stochastically analysed the error pattern and suggested what level of deviation from the required correlation value (to perform a particular operation) can be tolerated.

Contents

List of figures

CHAPTER 1 INTRODUCTION

Stochastic computing is a special type of computation where random bits are used to compute numbers and the requirements of hardware components are low compared to conventional binary circuits for the same operation. Stochastic computing was introduced in 1969 by Gaines [11] as a low-power, small-size, and fault-tolerant circuit as compared to conventional binary computing.

A long random bit stream of 0s and 1s is used to represent the data. The probability of 1 appearing in the data bit stream represents the data in binary. For example, the 8-bit stochastic number ***10010101*** represents ***4/8***, which means 0.5 in binary, and the 16-bit stochastic number ***0001 0000 1100 0101*** represents ***5/16*** in binary. A measurement of the similarity of bits between two bit streams is used as a correlation measurement. Correlation can be dependent or independent. For example, ***px = 00001111*** and ***py = 00110011*** are uncorrelated, ***px = 00001111*** and ***py = 00001111*** are positively correlated, and ***px = 00001111*** and ***py = 11110000*** are negatively correlated. All ***px*** and ***py*** represent stochastic numbers of probability value 0.5.

In [3], Alaghi created a new measure for measuring the correlation of a stochastic bit stream, which is much more appropriate in the case of stochastic computing, and they created useful circuit operations. However, unlike other types of circuits, the output of this type of circuit depends on the errors caused by noisy channels, such as binary symmetric channels or binary asymmetric channels that will impact the processes of stochastic computing. Apart from this, the output of these circuits depends on the value of the correlation ***(r)***. As shown by Alaghi, the output of a circuit depends on a particular ***r***. However, there is a shift in that value in real applications. The output will vary. How much it suffers—that is the main part of my work where the error is analysed only due to variation in correlation.

The thesis is organised into the following 6 chapters: The theoretical aspects of stochastic computing are discussed in chapter 2. In chapters 3, 4 and 5, theoretical and practical analysis are done on AND, XOR and OR gates, respectively. In chapter 6, an application for edge detection is demonstrated, and the thesis is concluded in chapter 7.

CHAPTER 2 BACKGROUND

In this chapter, the basics of stochastic computing are covered. The functions of some logic gates in stochastic computing are discussed.

2.1 Stochastic Computing:

Random bits of sequences are used in stochastic computing, a sort of computer processing, to calculate certain complicated operations on modest hardware. Here, computational data (in bitstreams) is processed as digitised probabilities after being represented as stochastic sequences. For example, while performing a multiplication operation, two stochastic numbers are given into an AND gate, which then outputs the result.

2.2 Stochastic Numbers:

The stochastic numbers are represented by bit streams containing 0 and 1 and interpreted through probability. The representation of bit-streams is random and independent, but can be dependent on the requirements of arithmetic operations. In stochastic computing, binary numbers are converted into stochastic numbers in order to perform operations.

2.3 Stochastic Coding format:

Two forms can be used to code stochastic representation:

1. SC-unipolar
2. SC-bipolar

In SC-unipolar, the representations and all the involved computations always lie within the real-number interval ***[0, 1].***

Example:- A bit-stream S containing **75%** 1s and **25%** 0s denotes the number ***p = 0.75***, reflecting the fact that the probability of observing a 1 at an arbitrary bit position is ***p***.

Neither the length nor the structure of S need be fixed; **(1,1,1,0), (1,0,1,1),** and **(0,1,0,1,1,1,1,1)** are all possible representations of 0.75.

Here, ***p*** depends on the ratio of 1s to the length of the bit-stream, not on their positions, we will refer to bit-streams of this type and the probabilities they represent as **stochastic numbers**.

A 2's complement binary input bit-stream **$\{0011\}_2$** is represented in stochastic bit-streams S, consisted of 3 of bit '1' out of $\boldsymbol{2^4}$ ***= 16*** bits (remaining bits are zeros). This stochastic bit-streams ***S***, is also interpreted as ***p(S = 1) = 3/16***. Such coding is termed as SC-unipolar format.

In SC-bipolar, the representations and all the involved computations always lie within the real-number interval ***[-1, 1].***

There are two more that represent stochastic numbers. These are inverted bipolar (IBP) and ratios of 1s and 0s of bit streams. Inverted bipolar is the inverse representation of bipolar. If **(0, 1, 1, 1)** represents ***+1/2*** in bipolar representation, then it represents ***-1/2*** in inverted bipolar representation. The representations of IBP and all the involved computations always lie within the real-number interval ***[-1, 1].***

Ratios of 1s and 0s of bit streams uses the ratio of the number of 1's and 0's to represent any number in the range [0, ∞].

2.4 General SC architecture:

Three main components of stochastic computing are

2.4.1 Stochastic number generator (SNG)

The stochastic number generator works as a converter from stochastic numbers to digital binary numbers. A comparator is used to transform the deterministic binary value into stochastic bit-streams with the help of a (pseudo) random number generator. A comparator compares the random numbers produced by the random number generator with the input binary number shown in the image.

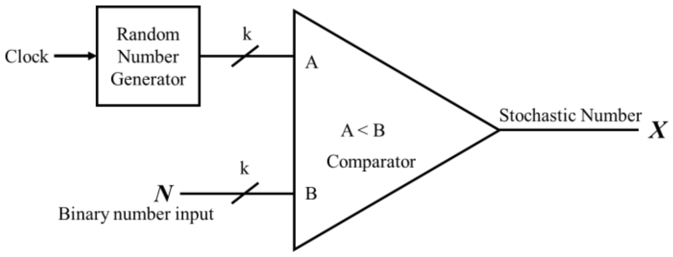

Fig. 2.1 Binary-to-stochastic converter

2.4.2 Stochastic computing elements (SCE)

The stochastic computing elements, which include multiplier, adder, subtractor, and so on, are used to compute the desired arithmetic operators from random stochastic numbers.

2.4.3 De-randomizer Unit (DRU)

The De-randomizer unit converts computed stochastic numbers into deterministic binary numbers. A DRU is usually a binary counter which counts the 1s from the output stochastic number and, after that, converts it into a binary number.

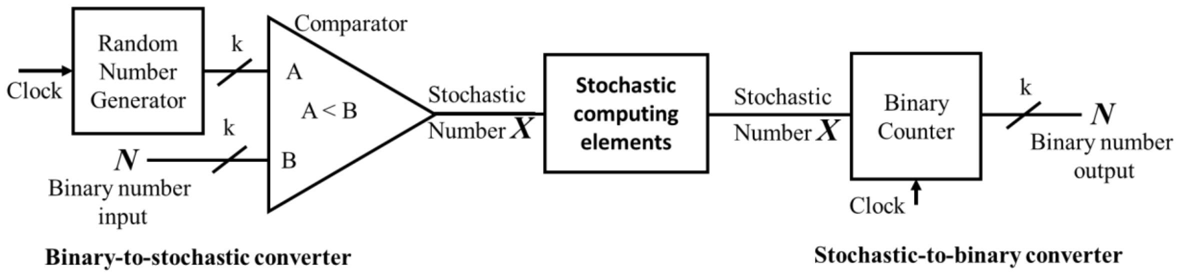

Fig 2.2: General Architecture of Stochastic Computing (SC) system

2.5 Basics operations of stochastic computing elements

Some basics but important operations of stochastic computing are -

1. Addition
2. Multiplication
3. Absolute subtraction

2.5.1 Addition

In addition, a multiplexer is used. The typical addition procedure becomes problematic since the sum of two numbers from the interval [0, 1] resides in [0, 2], while stochastic numbers naturally fall into the range [0, 1]. For this reason, results from [0, 2] are mapped to [0, 1] via a particular scaled addition operation in SC.

When applied to its data inputs ***S1*** and ***S2***, a two-way multiplexer can compute the sum of two stochastic numbers, ***p(X) = x*** and ***p(Y) = y***. A third number with the constant value ***p(W) = w = 1/2*** is also needed and is applied to the multiplexer's third (select) input; this can be provided by a (pseudo) random number generator. Here, the correlation between the two input numbers should be zero.

The probability of a 1 appearing at the output ***z = p(Z)*** is then equal to the probability of 1 at ***p(W)*** multiplied by the probability of 1 at ***p(X),*** plus the probability of 0 at ***p(W)*** multiplied by the probability of 1 at ***p(Y).***More formally,

$$z = xw + (1 - w)y = (x + y) / 2$$

Here, W effectively scales the sum by 1/2

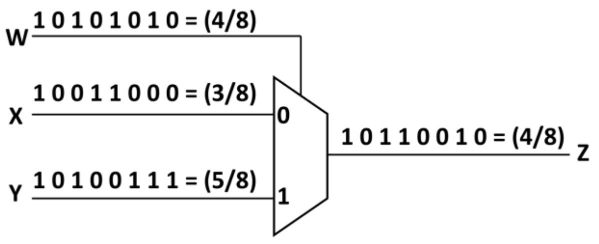

Fig. 2.3

In the figure, we can see that ***x = 3/8*** is added with ***y = 5/8***. With the help of the 3rd input, which is a random bit-stream of ***w = 4/8***, the multiplier performs addition of ***x*** and ***y*** and produces the output, which is ***z = 4/8***.

$$z = (x + y) / 2$$

$$\text{or, } z = (4/8 + 6/8) / 2$$

$$\text{or, } z = 5/8$$

We can see here that the addition operation performs correctly.

2.5.2 Multiplication

Two numbers are multiplied using a two-input AND gate. ***p(Y) = y*** and ***p(X) = x***. When two stochastic numbers, X and Y, are applied to the inputs of an AND gate, the result is

$$z = p(Z) = p(X)\, p(Y) = xy.$$

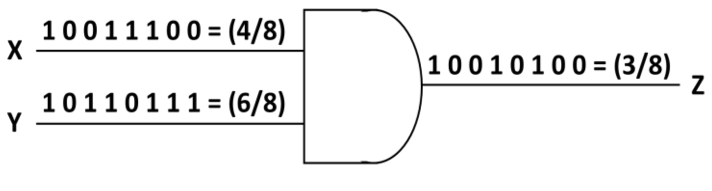

Fig. 2.4

Consider two independent random bit streams ***X*** and ***Y*** as inputs to a two-input AND gate where,

$$p(X) = x = 4/8$$

$$p(Y) = y = 6/8$$

The output random bit stream is then given by,

$$z = p(Z) = p(X , Y) = p(X)\, p(Y) = xy = 3/8$$

Thus, one AND gate can perform stochastic multiplication.

2.5.3 Absolute subtraction

Subtraction can be implemented by a single XOR. Here, inputs ***p(X)*** and ***p(Y)*** (of ***x*** and ***y,*** respectively) are highly correlated. Because of the maximum overlapping of 0s and 1s due to high correlation, the probability of receiving two 1s (or two 0s) on ***X*** and ***Y*** is ***min (p (X), p (Y))*** (or ***min (1 - p (X), p (Y))***), implying that the probability of receiving different values on ***x*** and ***y*** is |***p (X) - p(Y)***|. If stochastic numbers ***p(X)*** and ***p(Y)*** are uncorrelated, then XOR implements an entirely different function.

Function of XOR gate (highly correlated inputs) is

$$z = p(Z) = p(X)\,(1 - p(Y)) + p(Y)\,(1 - p(X)) = |p(X) - p(Y)| = |x - y|$$

In the figure, we can see that, ***p(X) = 2/8,*** and ***p(Y) = 7/8*** is subtracted and after the subtraction, we get ***p(Z) = 1/8***

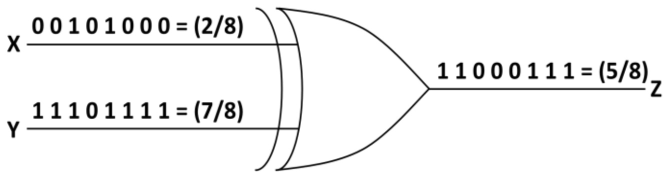

Fig. 2.5

$$p(Z) = |p(X) - p(Y)|$$

$$\text{or, } p(Z) = |2/8 - 7/8|$$

$$\text{or, } p(Z) = 5/8$$

2.6 Advantages and disadvantages of stochastic computing:

There are many advantages to using stochastic circuits over combinational circuits in some operations. With the help of examples, we can see that operations of addition, subtraction, or multiplication can be performed with the help of one multiplier, one XOR gate, or one AND gate, respectively. This significantly reduces the hardware size compared to conventional circuits for the same operation.

Another advantage of using stochastic circuit over conventional circuit is that a single bit-flip in the conventional 2's complement computation will result in a huge error, especially if the bit-flip occurs on a higher-order bit. But, a single bit-flip in a long bit-stream causes only a minor change in the original logical value. Thus, stochastic computing has high fault tolerance.

There are also disadvantages to using stochastic computing. An increase in the precision of a stochastic computation requires an exponential increase in bit-stream length, implying a corresponding exponential increase in computation time. For example, to change the numerical precision of a stochastic computation from 4 to 8 bits requires increasing the bit-stream length from ***2^4 =16*** bits to ***2^8 = 256*** bits.

Variations in the representation of the numbers being processed can lead to inaccurate results. If identical bit-streams denoting some number p are applied to the AND gate, the result is p, rather than the numerically correct product $p \times p$.

2.7 Correlation of stochastic numbers:

Correlation refers to how two bit streams are similar to each other bit-wise. As discussed in detail in [3], the Pearson correlation coefficient can be measured using the following formula:

$$\boldsymbol{\rho(X,\, Y) = \frac{ad - bc}{\sqrt{(a+b)(a+c)(b+d)(c+d)}}}$$

Where ***X*** and ***Y*** are two bit streams of length ***n, a*** refers to the number of overlapping of 1s, ***b*** refers to the number of overlapping of 1s in ***X*** and 0s in ***Y***, ***c*** refers to the number of overlapping of 0s in ***X*** and 1s in ***Y*** and ***d*** refers to the number of overlapping of 0s. But, this fails to capture the stochastic nature of bit streams for which Alaghi proposed SCC (Stochastic coefficient correlation).

For two stochastic numbers ***X*** and ***Y***, Stochastic Coefficient Correlation ***SCC(X, Y)*** is given by

$$\boldsymbol{SSC\,(X,\, Y) = \begin{cases} \dfrac{p_{x\wedge y} - p_x p_y}{\min(p_x,\, p_y) - p_x p_y} & if\ p_{x\wedge y} > p_x p_y \\ \dfrac{p_{x\wedge y} - p_x p_y}{p_x p_y - \max(p_x + p_y - 1, 0)} & otherwise \end{cases}}$$

Where $\boldsymbol{x \wedge y}$ is bit-wise AND function, and $\boldsymbol{p_{x\wedge y}}$ is probability of 1's.

When stochastic numbers $\boldsymbol{X}$ and $\boldsymbol{Y}$ are uncorrelated to each other, $\boldsymbol{p_{x\wedge y} - p_x p_y = 0}$, that means, $\boldsymbol{SSC\,(X,\, Y) = 0}$, when $\boldsymbol{X}$ and $\boldsymbol{Y}$ are similar to each other (or maximum similarity), $\boldsymbol{SSC\,(X,\, Y) = +1}$, and when $\boldsymbol{X}$ and $\boldsymbol{Y}$ are different to each other (or maximum difference), $\boldsymbol{SSC\,(X,\, Y) = -1}$. The following table shows some examples of bit streams with their SCC values. In this thesis, correlation is referred to as **r.**

Stochastic numbers	**SC correlation** $\boldsymbol{SCC\,(X,\, Y)}$
X = 1111111100000000 Y = 1111000011110000	0
X = 1111111100000000 Y = 1111111100000000	+1
X = 1111111100000000 Y = 0000000011111111	-1

Table 2.1

2.8 Function of combinational circuits in presence of Correlations:

As discussed in details in [3], AND gate implements multiplication function, function of AND gate is, $\boldsymbol{p_z = F(p_x, p_y) = p_x * p_y}$, when ***SCC(X, Y) = 0***. But, in presence of correlations, circuit behaviours changes, when ***SCC(X, Y) = +1***, $\boldsymbol{p_z = F(p_x, p_y) = min(p_x, p_y)}$, when ***SCC(X, Y) = -1***, $\boldsymbol{p_z = F(p_x, p_y) = max(p_x + p_y - 1, 0)}$.

The following table shows the functions of some basic circuits with various correlation levels.

Circuits	**SCC = 0**	**SCC = +1**	**SCC = –1**
AND	$p_x \times p_y$	$min(p_x, p_y)$,	$max(p_x + p_y - 1, 0)$.
XOR	$(p_x+p_y - 2 \times p_x \times p_y)$	$\|p_x - p_y\|$	$min(2 - p_x - p_y, p_x + p_y)$
OR	$(p_x + p_y - p_x \times p_y)$	$max(p_x, p_y)$	$min(p_x + p_y, 1)$
Multiplexer	$½(p_x + p_y)$	$½(p_x + p_y)$	$½(p_x + p_y)$

Table 2.2

From the table, we can see that AND works as a multiplier when correlation is +1, XOR gate works as absolute subtractor when correlation is +1, Multiplexer works as a scaled adder irrespective of correlation.

CHAPTER 3

AND GATE AS STOCHASTIC CIRCUIT ELEMENT

An AND gate is a basic digital gate that works as a multiplier of two bit streams in stochastic computing. In this chapter, the theoretical outputs of AND gate is presented as how a two-input AND gate behaves within a correlation range from 0 to 1.

3.1 Usability and analytics of AND gate for stochastic computing operation

Let us consider a function,

$$AND(x, y) = f_{AND}(x, y) \qquad \ldots(3.1)$$

For any correlation, r > 0

$$f_{AND}(x, y) = xy(1 - r) + r\min(x, y) \qquad \ldots(3.2)$$

$$\tilde{e} = \%\ error = \frac{r\,(min(x,y) - xy)}{xy} = r\left(\frac{1}{max(x,y)} - 1\right) \qquad \ldots(3.3)$$

For any correlation, r < 0

$$f_{AND}(x, y) = xy(1 - r) + r\max(x + y - 1, 0) \qquad \ldots(3.4)$$

$$\tilde{e} = \%\ error = \frac{r\,(max(x + y - 1, 0) - xy)}{xy} \qquad \ldots(3.5)$$

This will not happen in reality. Errors will fluctuate around this particular value (ẽ).

In practical, ẽ = % error = {e | e = error of 100 simulation for a given pair of x and y}

Let’s consider an example (1), x = 0.5, y = 0.7

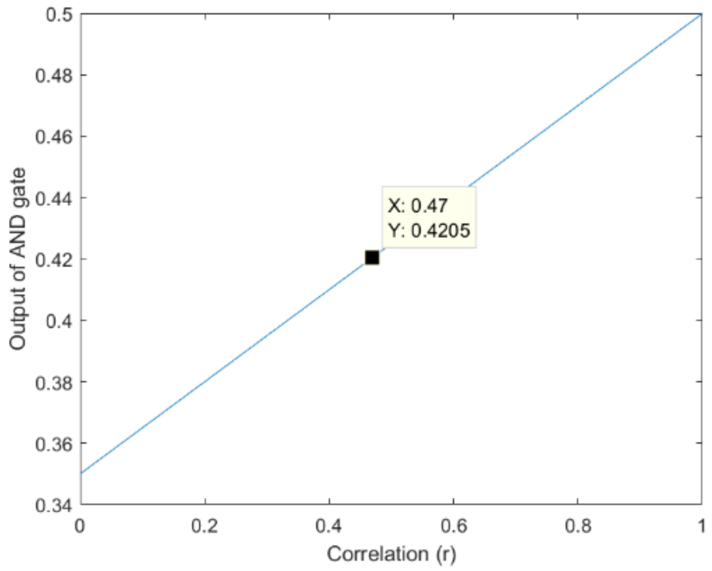

Fig 3.1: Ideal output of AND gate for correlation 0 to 1

The ideal AND gate output for 0 correlation is $\boldsymbol{x \times y = 0.5 \times 0.7 = 0.35}$. When correlation is +1, the output will be $\boldsymbol{\min(x, y) = \min(0.5, 0.7) = 0.5}$. In any other correlation, the output will be a weighted average of the two output values shown in the graph.

We want to achieve 0.35 as output. However for correlation +1, we get 0.5, and for other correlation, we are getting values representing a line shown in the graph which is in between 0.35 and 0.5.

We can consider a maximum tolerance output, beyond which output will not be useable. Putting $\boldsymbol{x = 0.5}$ and $\boldsymbol{y = 0.7}$ in equation 2, we get,

$$\boldsymbol{\tilde{e} = r\left(\frac{1}{0.7} - 1\right) = r\frac{0.3}{0.7} = \frac{3r}{7}} \qquad \ldots(3.6)$$

If we take our maximum tolerance to be 20%, then 20% of 0.35 = 0.07. So, the maximum output value that can be tolerated is 0.42. Therefore,

$$\frac{3r^*}{7} = 0.2\,,$$

$$or\; r^* = \frac{1.4}{3} = 0.47$$

We will get an output between 0.35 and 0.42 for any correlation r less than $r^* = 0.47$. We can use that. But as an example, if we analyse further, we will not get the exact value. When we run 100 simulations, 100 times output is generated for 0.5 and 0.7. For this particular correlation value, we get this.

Px	Py	Expected	r = 0	r = 0.25	r = 0.5	r = 0.75	r = 1
0.5	0.7	0.35	0.35073	0.38894	0.42468	0.46396	0.49938

Although, from theory, we get that any value of r less than r* is usable. We can clearly see that, due to random fluctuations, this will not always be guaranteed. So, for this r* value correlation, sometimes we will get much more. This happens due to imperfection in the system, because x and y are never truly dependent on the length of bits, the simulation, and other random things. So we did 100 simulations and we can see that.

Let us consider more examples, example (2), if X and Y are 0.3 and 0.8, the output is 0.24 when correlation is 0 and 0.3 when correlation is 1. In this case,

$$r^*\left(\frac{1}{\min(0.3, 0.8)} - 1\right) = 0.2$$

$$\text{Or, } r^* = 0.85$$

And example (3), if X and Y are 0.5 and 0.9, the output is 0.45 when correlation is 0 and 0.5 when correlation is 1. In this case,

$$r^*\left(\frac{1}{\min(0.5, 0.9)} - 1\right) = 0.2$$

$$\text{Or, } r^* = 0.2$$

In example 2, for a 20% error tolerance, r^* is 0.85 and in example 3, r^* = 0.2. With these examples, we can clearly see that r^* is depend on specific X and Y.

For correlation, r < 0

Let's consider an example (1), x = 0.5, y = 0.7

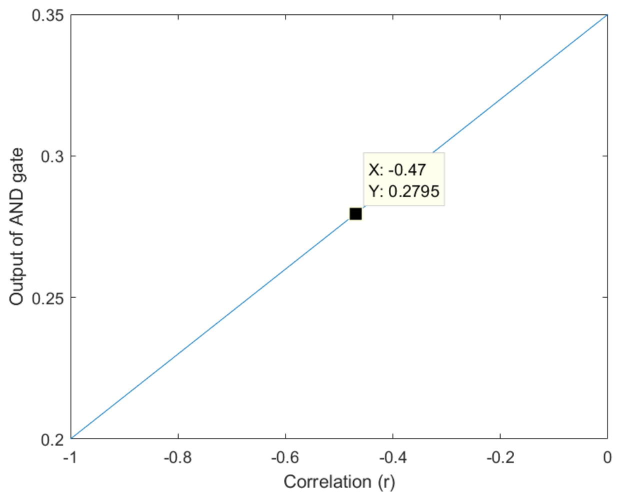

Fig. 3.2: Ideal output of AND gate for correlation -1 to 0

The ideal AND gate output for 0 correlation is ***x × y = 0.5 ×0.7 = 0.35***, while the output for -1 correlation is ***max (x + y - 1, 0) = max (0.5 + 0.7-1, 0) = 0.2***. Any other correlation, the output will be a weighted average of the two output values shown in the graph.

We want to achieve 0.35 as an output. However, for correlation -1, we get 0.2, and for other correlations, we are getting values representing a line shown in the graph, which is in between 0.2 and 0.35.

We can consider a maximum tolerance output, beyond which the output will not be usable. Putting x = 0.5 and y = 0.7 in equation 3, we get,

$$\tilde{e} = \frac{r\,(max(0.5+0.7-1{,}0)-0.5*0.7)}{0.5*0.7} = \frac{r(0.2-0.35)}{0.35} = -r\frac{0.15}{0.35} = -\frac{3r}{7} \qquad \ldots(3.7)$$

If we assume our maximum tolerance is 20%, then 20% of 0.35 = 0.7. So, the minimum output value that can be tolerated is 0.28. Therefore,

$$-\frac{3r^*}{7} = 0.2\,,$$

$$or\ r^* = \frac{1.4}{3} = -0.47$$

For any correlation, r greater than r^* = **-0.47**, we will get an output of between 0.35 and 0.2. We can use that.

Similarly, if X and Y are 0.3 and 0.8, the output is 0.24 when correlation is 0 and 0.1 when correlation is -1. In this case,

$$\frac{r^*\,(\max(0.3\,+\,0.8-\,1{,}0)-0.3*0.8)}{0.5*0.7} = 0.2$$

$$or, \boldsymbol{r}^{*} = \mathbf{0.5}$$

Now, for any application, we will know what percentage of errors can be tolerated. But we will not know the x and y. That requires a greater analysis. So, we need to find r* for a case when x and y are unknown. So, when x and y are unknown, we will replace the entire search space with uniform probability where x and y can take any value from 0 to 1, and for all possible values, we will run the simulation and try to get the value of r*. r** is independent of x and y values.

3.2 Definition of '80th percentile error':

In this thesis, 100 simulations have been done on all possible **Px** and **Py** for particular SCC for a particular circuit. For every pair of **Px** and **Py**, we get 100 errors. After that, we take the average of these errors. But, considering the worst case scenario for our simulations, we have taken the 80th percentile. This 80th percentile is referred to as the **'80th percentile error'**.

3.3 Simulation results

Here, an analysis has been done on the '**80th percentile error'** on AND gate. From 100 samples of every combination of **Px** and **Py** with every SCC from 0 to 1, **80 percentile error** is used for our worst case scenario.

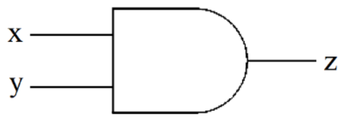

Fig 3.3. AND gate

In this analysis, AND is operated as a multiplier, which happens due to the perfect uncorrelation of Px and Py (SCC = 0). However, in this report we will study the deviation of this ideal behaviour when correlation is close to 0 but not necessarily 0. The AND gate's function is shown below.

$$\boldsymbol{p(z) = p(x) \times p(y)} \tag{3.8}$$

where p(z) is output and p(x) and p(y) are inputs.

Following 2 tables are the simulation done on different SCC ranges from r = 0 to r = 1 and different Px and Py from 0 to 1.

Table 3.1

Px	**Py**	**r = 0**	**r = 0.1**	**r = 2**	**r = 0.3**	**r = 0.4**	**r = 0.5**
0	0	NaN	NaN	NaN	NaN	NaN	NaN
0	0.1	NaN	NaN	NaN	NaN	NaN	NaN
0	0.2	NaN	NaN	NaN	NaN	NaN	NaN
0	0.3	NaN	NaN	NaN	NaN	NaN	NaN
0	0.4	NaN	NaN	NaN	NaN	NaN	NaN
0	0.5	NaN	NaN	NaN	NaN	NaN	NaN
0	0.6	NaN	NaN	NaN	NaN	NaN	NaN

0	0.7	NaN	NaN	NaN	NaN	NaN	NaN
0	0.8	NaN	NaN	NaN	NaN	NaN	NaN
0	0.9	NaN	NaN	NaN	NaN	NaN	NaN
0	1	NaN	NaN	NaN	NaN	NaN	NaN
0.1	0.1	26.95	124.61	222.27	319.92	398.05	534.77
0.1	0.2	17.19	61.135	114.84	144.14	183.2	239.35
0.1	0.3	13.933	39.973	64.388	93.685	114.84	139.26
0.1	0.4	12.305	28.174	41.602	58.693	73.34	90.43
0.1	0.5	5.468	26.954	26.954	44.532	52.344	65.039
0.1	0.6	9.05	17.188	27.767	31.837	38.347	45.671
0.1	0.7	11.607	12.305	19.977	22.769	27.651	33.929
0.1	0.8	4.98	10.474	13.525	19.629	18.409	22.07
0.1	0.9	6.8794	13.389	9.5922	9.0494	12.306	13.933
0.1	1	7.91	6.45	7.91	6.45	7.42	7.91
0.2	0.2	13.525	56.25	102.64	138.04	178.32	223.49
0.2	0.3	9.05	34.278	61.133	80.667	103.45	130.31
0.2	0.4	7.4225	23.291	43.431	54.419	71.512	86.156
0.2	0.5	5.96	17.19	29.395	39.65	48.925	57.715
0.2	0.6	8.2333	13.933	22.475	26.546	34.275	40.383
0.2	0.7	7.0714	10.907	15.793	21.025	23.464	28.696
0.2	0.8	7.425	8.6438	9.8625	13.525	14.441	18.103
0.2	0.9	5.5222	6.8778	6.3389	9.05	9.05	13.389
0.2	1	4.98	5.955	4.49	5.955	6.2	3.515
0.3	0.3	9.5922	31.289	57.333	81.211	102.91	129.49
0.3	0.4	9.05	24.917	38.35	51.367	69.275	84.733
0.3	0.5	6.4467	15.887	26.303	34.767	49.74	56.577
0.3	0.6	6.6083	11.492	18.817	28.039	32.378	38.617
0.3	0.7	3.7	9.281	14.862	18.119	21.605	26.49
0.3	0.8	6.6083	8.2375	9.8625	11.896	15.356	16.779
0.3	0.9	3.263	6.337	5.4333	8.1444	6.5185	10.678
0.3	1	3.84	4.1667	3.5167	4.0033	2.7017	5.9567
0.4	0.4	4.3687	20.544	36.106	52.281	67.844	83.106
0.4	0.5	4.49	17.432	26.465	36.23	45.265	56.005
0.4	0.6	4.3708	10.067	18.204	25.121	31.225	36.923
0.4	0.7	3.5857	9.5143	12.479	18.407	21.896	25.907
0.4	0.8	1.9281	5.2844	8.4891	12.153	14.441	17.494
0.4	0.9	3.625	4.7083	5.5222	6.0667	7.9639	9.9986
0.4	1	3.5162	3.0275	3.2725	3.2725	3.515	2.1738
0.5	0.5	4.492	14.844	25.196	35.352	44.142	54.296
0.5	0.6	4.8167	11.165	16.863	23.21	31.183	38.02
0.5	0.7	2.68	6.8629	12.026	16.629	20.257	25.837
0.5	0.8	3.515	5.835	8.3975	10.474	13.037	16.455
0.5	0.9	2.4311	4.0578	5.0356	5.7944	7.4222	8.1811
0.5	1	2.832	2.344	3.028	3.028	2.832	2.637
0.6	0.6	2.8111	9.4556	16.644	22.614	30.208	36.989
0.6	0.7	3.2357	6.4929	10.676	15.329	20.443	24.86

0.6	0.8	2.7427	4.5729	7.524	10.169	12.304	15.967
0.6	0.9	2.9	3.7148	4.3481	5.7944	7.512	7.9648
0.6	1	1.8883	1.8883	1.5633	2.3767	2.5383	1.4817
0.7	0.7	3.0367	7.2224	12.006	16.192	19.679	23.565
0.7	0.8	2.8009	5.6786	7.6839	9.6884	11.782	15.095
0.7	0.9	2.3063	3.081	4.3992	4.8643	6.8794	7.4222
0.7	1	2.0507	1.6321	1.7014	2.26	1.2136	1.8414
0.8	0.8	1.6234	4.5234	7.3453	8.9484	11.847	14.289
0.8	0.9	1.725	2.675	3.6236	4.9806	5.6583	6.8792
0.8	1	1.3794	1.44	1.4406	1.44	1.3794	1.5625
0.9	0.9	1.3938	2.358	3.2623	4.2877	5.9753	7.1204
0.9	1	1.02	0.85722	0.80333	1.02	0.74889	0.74889
1	1	0	0	0	0	0	0

Table 3.2

Px	**Py**	**r = 0.6**	**r = 0.7**	**r = 0.8**	**r = 0.9**	**r = 1**
0	0	NaN	NaN	NaN	NaN	NaN
0	0.1	NaN	NaN	NaN	NaN	NaN
0	0.2	NaN	NaN	NaN	NaN	NaN
0	0.3	NaN	NaN	NaN	NaN	NaN
0	0.4	NaN	NaN	NaN	NaN	NaN
0	0.5	NaN	NaN	NaN	NaN	NaN
0	0.6	NaN	NaN	NaN	NaN	NaN
0	0.7	NaN	NaN	NaN	NaN	NaN
0	0.8	NaN	NaN	NaN	NaN	NaN
0	0.9	NaN	NaN	NaN	NaN	NaN
0	1	NaN	NaN	NaN	NaN	NaN
0.1	0.1	608.01	691.02	788.67	896.09	993.8
0.1	0.2	280.86	319.92	346.78	400.5	432.25
0.1	0.3	165.3	181.58	212.5	240.17	251.57
0.1	0.4	111.18	119.73	134.38	156.35	173.45
0.1	0.5	77.734	81.64	97.266	107.04	118.76
0.1	0.6	49.74	57.878	66.833	71.717	79.042
0.1	0.7	33.929	40.904	43.7	49.271	53.457
0.1	0.8	23.291	28.175	29.4	31.837	37.331
0.1	0.9	15.022	17.733	17.189	19.356	23.156
0.1	1	9.38	5.96	10.35	7.42	7.42
0.2	0.2	266.21	302.82	341.9	389.5	421.24
0.2	0.3	153.9	178.32	199.48	232.03	248.3
0.2	0.4	105.08	118.5	133.76	149.64	161.22
0.2	0.5	70.9	81.64	88.48	102.15	111.91
0.2	0.6	45.263	52.992	62.763	68.458	77.408
0.2	0.7	31.486	36.371	41.6	46.136	49.971
0.2	0.8	23.294	24.513	27.259	28.478	30.919
0.2	0.9	12.85	13.933	13.933	16.644	17.458

0.2	1	5.955	5.955	5.225	5.7125	5.225
0.3	0.3	152.82	178.87	198.93	224.43	246.13
0.3	0.4	97.346	113.22	129.49	143.33	159.6
0.3	0.5	66.993	76.76	88.153	97.92	107.68
0.3	0.6	45.4	52.994	59.506	67.1	72.797
0.3	0.7	31.836	35.324	40.671	45.786	48.345
0.3	0.8	20.038	22.071	25.325	26.954	29.598
0.3	0.9	12.485	10.496	13.57	14.293	14.837
0.3	1	4.33	4.4933	4.8167	4.1667	3.6783
0.4	0.4	98.669	111.79	126.74	143.84	160.62
0.4	0.5	65.77	75.295	87.99	96.29	107.77
0.4	0.6	44.45	51.977	58.692	65.2	71.304
0.4	0.7	29.568	35.15	39.161	44.043	47.355
0.4	0.8	18.866	21.308	24.053	26.8	29.853
0.4	0.9	10.406	10.812	12.306	13.254	14.882
0.4	1	4.1262	2.7825	3.6375	3.15	4.005
0.5	0.5	64.648	75.196	84.96	94.92	105.08
0.5	0.6	44.53	51.53	58.04	65.202	71.062
0.5	0.7	29.604	33.65	37.277	42.3	46.346
0.5	0.8	18.164	21.582	22.68	26.22	28.663
0.5	0.9	9.7	11.22	12.522	12.413	14.041
0.5	1	3.223	2.344	1.563	3.126	2.54
0.6	0.6	42.822	50.961	57.064	63.846	70.356
0.6	0.7	28.348	33.348	36.719	43.113	46.833
0.6	0.8	17.696	19.934	22.681	25.427	27.767
0.6	0.9	8.687	10.315	11.31	11.672	13.842
0.6	1	2.62	2.05	2.7017	2.1317	1.6442
0.7	0.7	28.249	32.335	36.918	41.502	45.688
0.7	0.8	17.362	20.152	22.07	24.861	27.214
0.7	0.9	8.6619	10.057	10.367	12.383	13.079
0.7	1	1.4929	1.3529	1.9114	2.26	2.4693
0.8	0.8	17.492	19.323	21.536	24.283	26.648
0.8	0.9	7.6931	8.8458	10.474	11.626	12.711
0.8	1	1.7456	1.1962	1.1962	1.0737	1.3187
0.9	0.9	7.663	8.6877	10.075	11.22	12.365
0.9	1	1.02	1.1289	0.80333	0.58556	0.58556
1	1	0	0	0	0	0

As we can see, the result obtained after simulation, when r is 0.1, the percentage error is pretty low. However, the error depends on the specific Px and Py values when the correlation is 0.1. For some values, the percentage error is high. For example, when Px and Py are 0.2 and 0.3, the percentage error is 34.278, and when Px and Py are 0.3 and 0.4, the percentage error is 24.917. But, for other cases, if Px and Py are 0.5 and 0.8, then the percentage error is 5.835, and the accuracy is greater. However, since Px and Py are not known in advance, we have to look for the average error for all possible Px and Py. That is done and shown in the following plots.

From the simulation that was represented in tables 1 and 2, we can conclude the 80 percentile percentage error value for a particular r for a range of Px, Py that error is used in the following box plot.

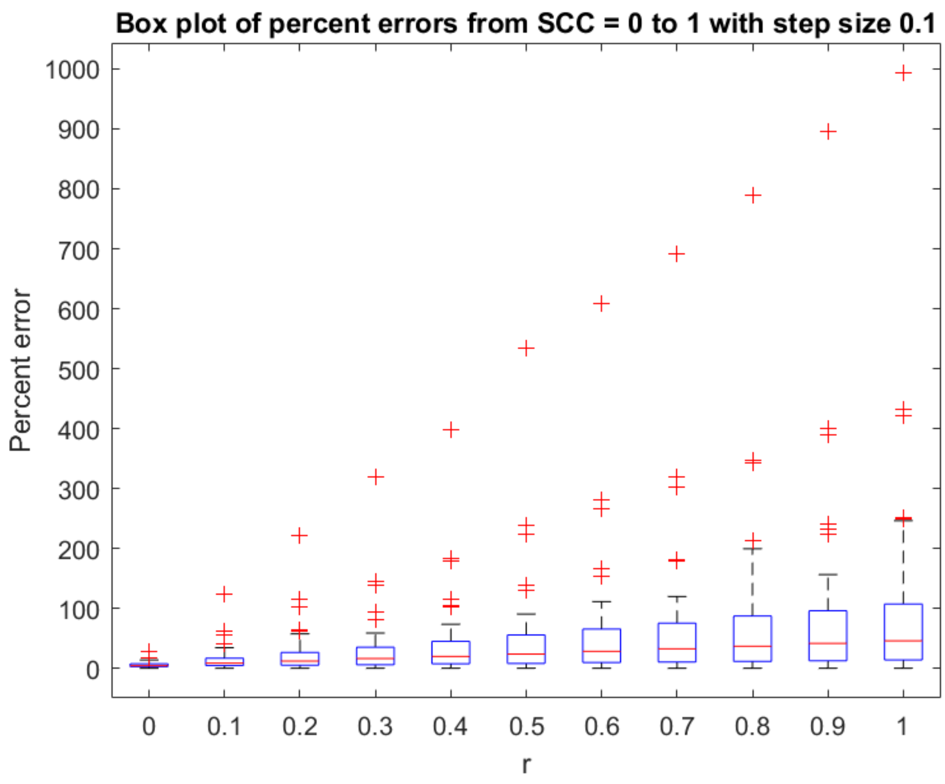

Fig 3.4

In the graph shown in 3.2, it can be seen that the highest error for a particular SCC is greater compared to the mean of 80 percentile percent error, and the gap between the mean of percent error and the highest error is increased with the increase of SCC.

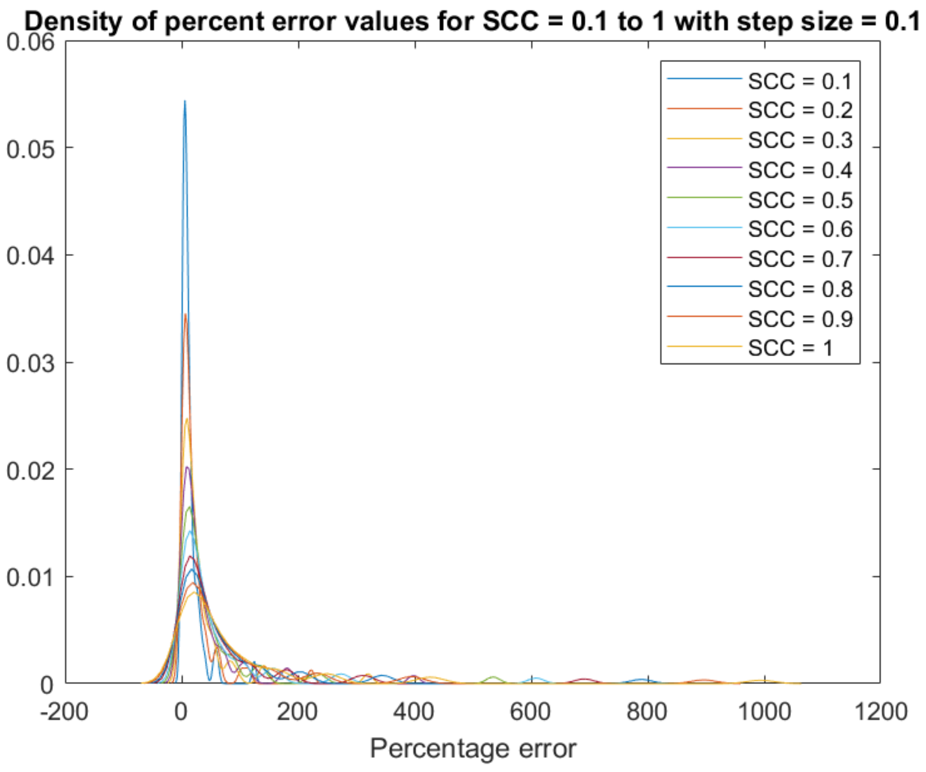

Fig. 3.5

In the graph shown in fig 3.4, it can be seen that the errors are increasing with the increase of SCC.

The following table represents the mean, 1 sigma limit, and 80th percentile for each column representing particular SCC from table 3.1 and table 3.2.

Table 3.3

	r = 0	**r = 0.1**	**r = 0.2**	**r = 0.3**	**r = 0.4**	**r = 0.5**	**r = 0.6**	**r = 0.7**	**r = 0.8**	**r = 0.9**	**r = 1**
Mean	4.73	12.13	19.44	26.29	32.81	41.00	47.79	54.21	61.17	69.02	75.49
1 sigma limit	9.42	30.81	53.02	73.31	91.98	118.84	137.65	156.43	176.96	200.96	220.51
80th percentile	7.42	17.19	26.61	35.62	46.36	56.18	66.14	75.73	88.04	96.78	107.71

We plot table 3.3 in the following graph.

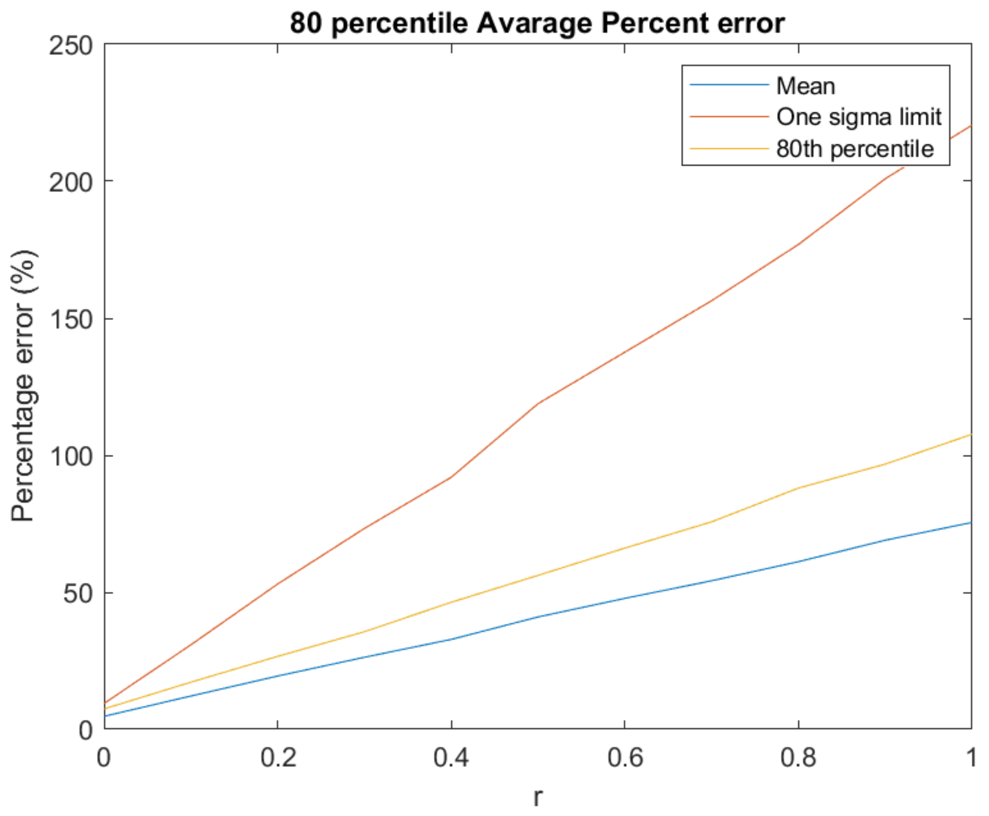

Fig. 3.6

We can see the average percentage error. So, the average error is pretty small, so we can work with it. As expected for all these cases, the error increases and the circuit becomes unusable when the correlation is above a certain level. But still, if we assume that the 20% error is permissible without degrading the performance of the entire system, we can see that 0.16. We can use it.

If our criteria are strict enough so that the maximum error is not more than 20%, in that case, we have to take a 1 sigma limit. The 1 sigma limit is the summation of the mean and standard deviation. If we consider a 1 sigma limit for our 20% permissible error, our maximum SCC value would be 0.04.

3.3 Summary:

We can conclude from the simulation done in this chapter that the range of SCC up to 0.16 can be used for any observed output value. But, if the criteria are strict, the maximum permissible limit will be 0.04. From table 3 of this analysis, the decision can be made more clearly without any observed output value that the observed output is useful or not.

CHAPTER 4

XOR GATE AS STOCHASTIC CIRCUIT ELEMENT

An XOR gate is a basic digital gate that works as an absolute subtractor of two bit streams in stochastic computing. In this chapter, similar studies are being performed as AND gate. Theoretical outputs of the XOR gate are presented as how a two-input XOR gate behaves within a correlation range from 0 to 1.

4.1 Usability and analytics of XOR gate for stochastic computing operation

Let us consider a function,

$$\boldsymbol{XOR(x, y) = f_{XOR}(x, y)} \tag{4.1}$$

SCC(x, y) = r > 0 (Here, the XOR gate works as a substractor when the correlation is +1. So, we will only study correlation from 0 to +1.)

For any correlation, r > 0

$$\boldsymbol{f_{XOR}(x, y) = (x+y-2xy)(1-r) + r\,|x-y|} \tag{4.2}$$

$$\boldsymbol{\tilde{e} = \%\ error = \frac{(1-r)\,|\,|x-y| - (x+y-2xy)|}{|x-y|}} \tag{4.3}$$

This will not happen in reality. Errors will fluctuate around this particular value (ẽ).

In practical, ẽ = % error = {e | e = error of 100 simulation for a given pair of x and y}

Let's consider an example (1), x = 0.5, y = 0.7

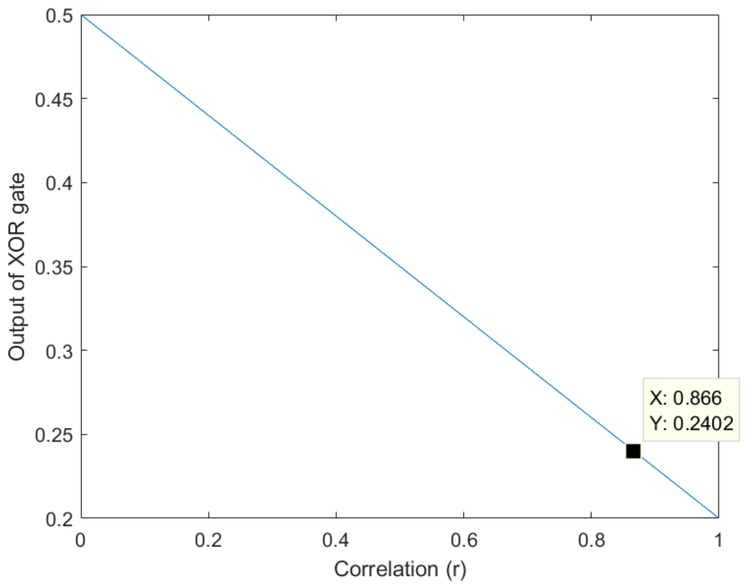

Fig. 4.1: Ideal output of XOR gate for correlation 0 to 1

The ideal XOR gate output for +1 correlation is ***|x-y| = |0.5-0.7| = 0.2***, while the output for 0 correlation is ***(x + y - 2xy) = (0.5 + 0.7 – 2 × 0.5 × 0.7) = 0.5***. Any other correlation, the output will be a weighted average of the two output values shown in the graph.

We want to achieve 0.2 as output. However, for correlation 0, we get 0.5, and for other correlations, we are getting values representing a line shown in the graph, which is between 0.2 and 0.5.

We can consider a maximum tolerance output, beyond which the output will not be usable. Putting x = 0.5 and y = 0.7 in equation 3, we get,

$$\tilde{e} = (\mathbf{1}-\boldsymbol{r})\left(\frac{\mathbf{0.5+0.7-2*0.5*0.7}}{|\mathbf{0.5-0.7}|}-\mathbf{1}\right) = (\mathbf{1}-\boldsymbol{r})\frac{\mathbf{0.3}}{\mathbf{0.2}} = \frac{\mathbf{3(1-r)}}{\mathbf{2}} \qquad (4.4)$$

If we take our maximum tolerance to be 20%, then 20% of 0.2 = 0.04. So, the maximum output value that can be tolerated is 0.24. Therefore,

$$\frac{\mathbf{3(1-}\ \boldsymbol{r}^*\mathbf{)}}{\mathbf{2}} = \mathbf{0.2}\,,$$

$$\boldsymbol{or\ r}^* = \mathbf{1} - \frac{\mathbf{0.4}}{\mathbf{3}} = \frac{\mathbf{2.6}}{\mathbf{3}} = \mathbf{0.867}$$

For any correlation, r greater than $r^* = 0.867$, we will get an output of between 0.24 and 0.2. We can use that. But as an example, if we analyse further, we will not get the exact value. When we run 100 simulations, 100 times output is generated for 0.5 and 0.7. For this particular correlation value, we get this.

Px	Py	Expected	r = 0	r = 0.25	r = 0.5	r = 0.75	r = 1
0.5	0.7	0.2	0.50055	0.42245	0.34979	0.27407	0.20241

Although, from theory, we get that any value of r greater than r* is usable. We can see that due to random fluctuations, this will not always be guaranteed. So, for this r* value correlation, sometimes we will get much less. This happens due to imperfection in the system because x and y are never truly dependent on the length of bits, the simulation, and other random things. So we did 100 simulations and we can see that.

Let us consider more examples, example (2), if X and Y are 0.3 and 0.8, the output is 0.5 when correlation is +1 and 0.62 when correlation is 0. In this case,

$$(1-r^*)\left(\frac{0.3+0.8-2*0.3*0.8}{|0.3-0.8|}-1\right) = 0.2$$

$$\text{Or, } r^* = 0.167$$

And example (3), if X and Y are 0.5 and 0.9, the output is 0.4 when correlation is +1 and 0.9 when correlation is 0. In this case,

$$(1-r^*)\left(\frac{0.5+0.9-2*0.5*0.9}{|0.5-0.9|}-1\right) = 0.2$$

$$\text{Or, } r^* = 0.84$$

In example 2, for 20% error tolerance, r^* is 0.167 and in example 3, r^* = 0.84. With these examples, we can see that r^* is depend on specific X and Y.

Now, for any application, we will know what percentage of error can be tolerated. But we will not know the x and y. That requires a greater analysis. So, we need to find r* for a case when x and y are unknown. So, when x and y are unknown, we will replace the entire search space with uniform probability where x and y can take any value from 0 to 1, and for all possible values, we will run the simulation and try to get the value of r*. r** is independent of x and y values.

4.2 Simulation results

An analysis has been done on the **‘80th percentile error’** on AND gate. In this chapter, a similar study has been done on XOR gate. From 100 samples of every combination of Px and Py with every SCC from 0 to 1, **80 percentile** error is used for our worst case scenario.

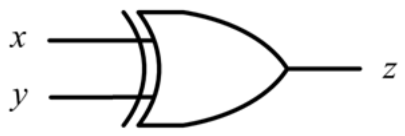

Fig.4.2 XOR gate

In this analysis, XOR is operated as a subtractor, which happens due to the perfect correlation of Px and Py (SCC = +1). However, in this report we will study the deviation of this ideal behaviour when correlation is close to +1 but not necessarily +1, and how the behaviour shifts when correlation drops from +1. The function of an XOR gate is given below,

$$\boldsymbol{p(z) = |p(x) - p(y)|} \tag{4.5}$$

where p(z) is output and p(x) and p(y) are inputs.

The following 2 tables are the results of the simulation done on different SCC ranges from r = 0 to r = 1 with step size 0.1 and different Px and Py from 0 to 1 with step size 0.1. Here, each cell corresponds to 100 simulations for Px and Py, and the table has detailed results. However, if we look for one particular entry, for example, 0.2 and 0.4, in this case, the expected output is | 0.2 – 0.4 | = 0.2, but the obtained output is not the same for different values of r. When we calculate the percentage error, we see that errors are very high for r less than 0.5. For example, when Px and Py are 0.2 and 0.4, and r = 0.5, the error is 66.5%. The circuit becomes unusable. But, for Px and Py being 0.6 and 1, the circuit is usable for any value of r because the error values are more or less 3 %.

Table 4.1

Px	Py	r = 0	r = 0.1	r = 0.2	r = 0.3	r = 0.4	r = 0.5
0	0	NaN	NaN	NaN	NaN	NaN	NaN
0	0.1	7.42	6.45	6.935	6.45	5.47	6.45

0	0.2	4.005	4.98	4.735	5.47	6.69	3.515
0	0.3	3.5167	4.655	3.5167	2.54	4.4933	3.19
0	0.4	3.515	3.6375	2.7825	2.7838	3.15	2.7825
0	0.5	3.028	2.54	2.442	2.539	2.247	2.344
0	0.6	1.5633	1.9692	1.7258	2.2133	2.62	2.1317
0	0.7	1.9114	1.6321	1.4229	1.2829	2.0507	1.7014
0	0.8	1.3794	1.135	1.5625	1.1962	1.7456	1.3794
0	0.9	1.02	0.91111	0.74889	0.91111	0.96556	1.02
0	1	0	0	0	0	0	0
0.1	0.1	Inf	Inf	Inf	Inf	Inf	Inf
0.1	0.2	170.99	151.95	138.28	120.7	106.54	91.895
0.1	0.3	75.05	69.92	61.377	53.81	49.902	41.357
0.1	0.4	45.345	41.277	35.58	33.138	27.117	23.373
0.1	0.5	28.54	26.342	23.169	20.24	18.286	15.235
0.1	0.6	18.75	16.992	15.82	13.086	12.402	10.254
0.1	0.7	11.735	11.165	9.7817	9.4567	7.91	6.9333
0.1	0.8	7.2829	6.7243	6.0964	5.7471	4.7714	4.6314
0.1	0.9	3.5769	3.5162	2.9662	2.905	2.5387	2.295
0.1	1	1.02	0.80333	0.91111	0.74889	0.69444	0.85722
0.2	0.2	Inf	Inf	Inf	Inf	Inf	Inf
0.2	0.3	295.5	267.19	236.42	205.66	176.85	152.44
0.2	0.4	126.07	114.6	101.66	90.675	78.225	66.505
0.2	0.5	69.923	64.713	58.53	51.043	44.53	38.347
0.2	0.6	43.921	39.648	35.865	31.592	26.952	23.29
0.2	0.7	26.562	23.828	21.68	19.238	17.871	14.16
0.2	0.8	15.316	14.177	12.63	11.979	10.84	8.9683
0.2	0.9	7.7014	6.9336	6.515	5.3986	5.3286	4.4929
0.2	1	1.3187	1.3187	1.3794	1.0737	1.3794	1.2575
0.3	0.3	Inf	Inf	Inf	Inf	Inf	Inf
0.3	0.4	372.66	336.52	300.39	264.75	230.08	194.43
0.3	0.5	155.61	142.19	126.07	110.69	96.532	78.467
0.3	0.6	84.082	76.757	69.108	61.297	51.693	44.53
0.3	0.7	49.415	43.8	38.672	35.988	30.005	25.732
0.3	0.8	26.074	23.73	21.68	19.14	18.067	15.137
0.3	0.9	12.305	11.653	9.7	8.7233	7.7467	7.6658
0.3	1	1.8414	1.2829	1.2829	1.8414	1.5629	1.7014
0.4	0.4	Inf	Inf	Inf	Inf	Inf	Inf
0.4	0.5	412.21	371.19	330.17	295.51	252.54	211.03
0.4	0.6	166.36	150.49	134.37	119.72	103.61	86.28
0.4	0.7	84.407	76.433	69.108	60.97	52.995	44.693
0.4	0.8	43.31	39.404	35.01	31.714	29.15	23.169
0.4	0.9	18.164	16.406	15.137	13.672	12.304	10.449
0.4	1	2.4575	2.0508	2.5383	2.2133	2.7017	2.5383
0.5	0.5	Inf	Inf	Inf	Inf	Inf	Inf
0.5	0.6	413.19	373.63	330.66	294.53	249.61	210.55
0.5	0.7	156.11	141.7	127.29	111.91	94.825	79.932

0.5	0.8	71.713	64.063	56.738	50.88	45.02	38.347
0.5	0.9	27.442	25.977	23.901	20.727	18.165	16.455
0.5	1	2.246	2.442	1.856	3.516	2.344	2.637
0.6	0.6	Inf	Inf	Inf	Inf	Inf	Inf
0.6	0.7	369.24	336.03	300.39	263.28	228.13	191.02
0.6	0.8	127.29	112.64	101.66	90.92	77.245	65.527
0.6	0.9	44.043	40.463	36.882	33.302	28.093	24.187
0.6	1	2.54	2.6613	3.2725	4.1262	4.2475	3.15
0.7	0.7	Inf	Inf	Inf	Inf	Inf	Inf
0.7	0.8	292.09	262.3	238.87	209.08	180.27	147.07
0.7	0.9	76.758	69.677	62.11	55.273	49.902	41.845
0.7	1	5.1433	4.1667	4.33	4.0033	4.655	4.1667
0.8	0.8	Inf	Inf	Inf	Inf	Inf	Inf
0.8	0.9	167.58	156.84	137.3	121.19	105.08	89.45
0.8	1	4.2475	4.735	5.47	4.2475	5.47	4.735
0.9	0.9	Inf	Inf	Inf	Inf	Inf	Inf
0.9	1	9.38	9.38	5.47	7.91	8.4	6.45
1	1	NaN	NaN	NaN	NaN	NaN	NaN

Table 4.2

Px	Py	r = 0.6	r = 0.7	r = 0.8	r = 0.9	r = 1
0	0	NaN	NaN	NaN	NaN	NaN
0	0.1	10.35	9.865	7.91	6.45	7.91
0	0.2	4.49	6.445	4.98	4.735	4.98
0	0.3	5.47	2.8633	3.5167	4.4933	4.4917
0	0.4	3.76	2.4175	2.6613	3.2725	3.0275
0	0.5	2.442	2.734	2.051	1.954	2.539
0	0.6	2.2133	2.4575	2.62	2.2942	2.3767
0	0.7	2.12	1.2829	1.5629	1.7014	1.4921
0	0.8	1.0131	1.3187	1.44	1.3187	1.8675
0	0.9	0.91111	0.85722	0.69444	0.85722	0.96556
0	1	0	0	0	0	0
0.1	0.1	Inf	Inf	Inf	Inf	NaN
0.1	0.2	75.78	57.715	41.6	25.98	11.815
0.1	0.3	34.275	24.51	18.65	12.305	5.225
0.1	0.4	20.605	16.537	12.142	8.0733	3.3533
0.1	0.5	13.525	10.107	7.91	5.835	3.515
0.1	0.6	8.692	6.836	5.274	4.102	3.028
0.1	0.7	5.8758	4.0033	4.1667	3.1083	2.295
0.1	0.8	3.3064	3.5157	3.0971	2.4693	1.4929
0.1	0.9	2.295	2.1725	2.1725	1.6237	1.0737
0.1	1	1.02	1.02	0.58556	0.96556	0.80333
0.2	0.2	Inf	Inf	Inf	Inf	NaN
0.2	0.3	122.17	95.31	66.02	37.695	8.89
0.2	0.4	54.295	41.845	30.86	16.21	4.2475
0.2	0.5	30.372	24.673	17.187	10.35	3.5167

0.2	0.6	19.385	14.624	12.305	6.445	4.2475
0.2	0.7	12.012	10.254	7.52	5.274	2.832
0.2	0.8	7.8283	6.2017	5.0617	3.0267	2.1317
0.2	0.9	3.9343	3.3064	2.8879	2.33	1.4929
0.2	1	1.3794	0.9525	1.5625	1.3794	1.7456
0.3	0.3	Inf	Inf	Inf	Inf	NaN
0.3	0.4	154.88	118.75	82.62	44.53	4.98
0.3	0.5	65.525	52.1	35.01	23.047	4.735
0.3	0.6	36.23	28.58	20.443	12.303	3.3533
0.3	0.7	20.117	16.21	11.695	6.9325	3.515
0.3	0.8	12.012	9.863	7.032	4.785	2.344
0.3	0.9	6.2017	4.8175	4.1667	3.5967	2.7017
0.3	1	1.9814	1.7014	1.9814	1.9814	1.8414
0.4	0.4	Inf	Inf	Inf	Inf	NaN
0.4	0.5	171.48	127.54	90.43	51.37	7.42
0.4	0.6	70.165	51.855	36.962	21.58	4.735
0.4	0.7	35.743	27.442	21.257	12.467	3.3533
0.4	0.8	19.63	15.235	10.717	8.0325	3.3938
0.4	0.9	9.082	7.129	5.957	3.71	2.637
0.4	1	1.8883	2.295	1.8883	2.9458	1.6442
0.5	0.5	Inf	Inf	Inf	Inf	NaN
0.5	0.6	172.46	131.45	89.45	46.48	6.45
0.5	0.7	65.04	53.32	36.475	20.605	4.735
0.5	0.8	30.533	24.512	17.84	10.84	3.19
0.5	0.9	13.769	10.474	8.155	6.5675	2.54
0.5	1	2.832	2.734	1.856	1.954	2.148
0.6	0.6	Inf	Inf	Inf	Inf	NaN
0.6	0.7	153.91	120.21	78.71	45.51	7.42
0.6	0.8	53.81	41.115	28.905	17.19	6.2
0.6	0.9	20.767	15.56	12.63	9.05	3.84
0.6	1	3.3938	2.7825	2.7825	3.0275	3.2725
0.7	0.7	Inf	Inf	Inf	Inf	NaN
0.7	0.8	126.56	97.27	64.06	38.67	8.4
0.7	0.9	35.01	26.955	19.63	11.815	4.735
0.7	1	3.6783	3.84	4.0033	4.33	3.6783
0.8	0.8	Inf	Inf	Inf	Inf	NaN
0.8	0.9	74.315	58.2	41.115	25	8.89
0.8	1	6.69	5.225	6.445	5.955	5.47
0.9	0.9	Inf	Inf	Inf	Inf	NaN
0.9	1	8.4	7.42	10.35	6.45	8.4
1	1	NaN	NaN	NaN	NaN	NaN

As we can see from the result obtained after simulation, when r is 0.9, the percentage error is pretty low. However, the error depends on the specific Px and Py values when the correlation is 0.9. For some values, the average error is high. For example, when Px and Py are 0.3 and 0.4, the percentage error is 44.53, and when Px and Py are 0.6 and 0.7, the percentage error is 45.51. But, in other cases, if Px and Py are 0.4 and 0.8, then the percentage error is 8.0325, and the accuracy is greater. But, for some Px and Py, examples are 0.4, 0.9; 0.3, 0.8 or 0.8, 1, the accuracy is greater for any value of r. That means, for these Px and Py combinations, the accuracy is greater compared to the other combinations. From the tables, we can see that the percentage error value varies from 300% to 2%. Such variation is expected because when we are looking for percentage error where output is very low, any small error can have a huge impact on percentage error.

However, since Px and Py are not known in advance and we only know the correlation and, based on that, we want to see the suitability of the circuit. We look for the average along column to infer about the r value and, using that, we get our following plots and tables.

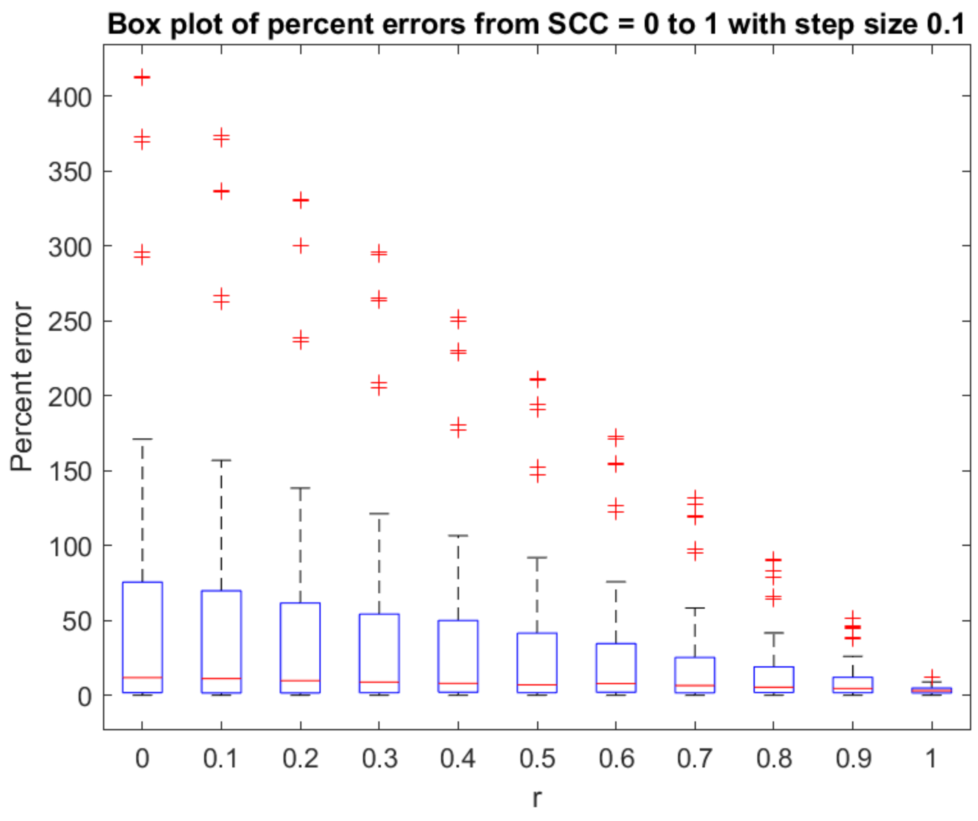

Fig. 4.3

In the graph shown in fig 4.2, it can be seen that the highest error for a particular SCC is greater compared to the mean of 80th percentile percent error, and the gap between the mean of percent error and the highest error is decreased with the increase of SCC. When SCC = 1, the error is the smallest.

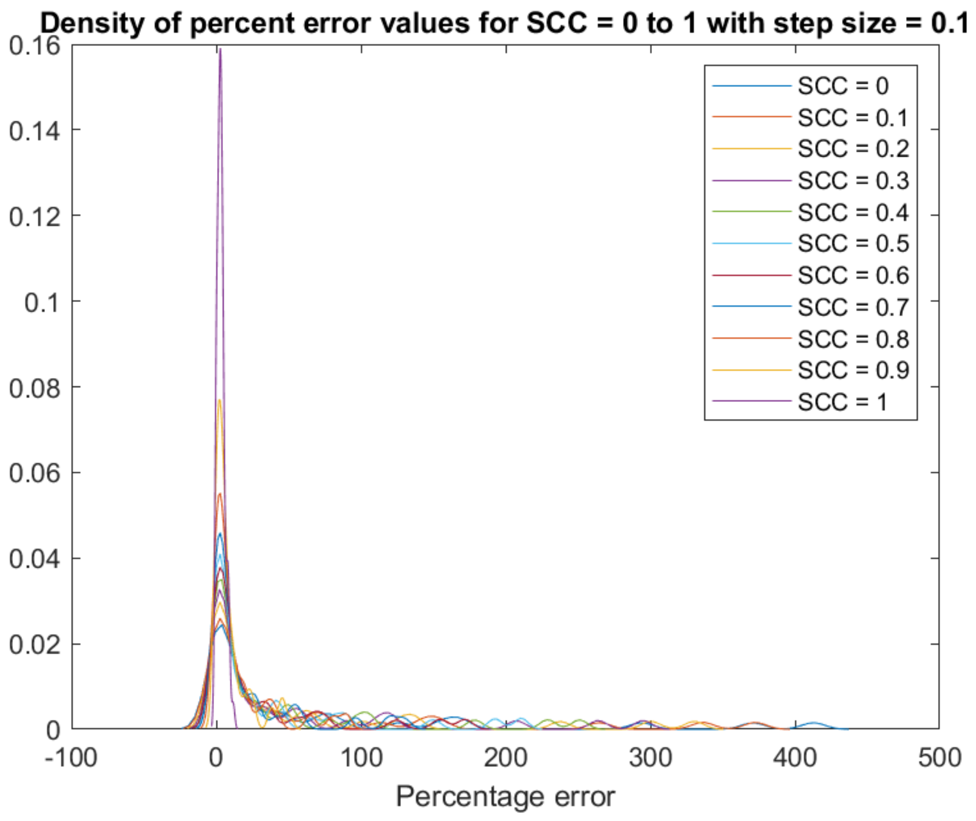

Fig 4.4

In the graph shown in fig 4.3, it can be seen that the errors are decreasing with the increase of SCC.

The following table represents the mean, 1 sigma limit, and 80th percentile for each column representing particular SCC from table 4.1 and table 4.2.

Table 4.3

	r = 0	r = 0.1	r = 0.2	r = 0.3	r = 0.4	r = 0.5	r = 0.6	r = 0.7	r = 0.8	r = 0.9	r = 1
Mean	64.21	58.27	52.06	46.2	40.19	33.88	28.09	21.82	15.68	9.62	3.34
1 sigma limit	170.67	154.47	137.82	121.88	105.12	88.43	72.33	55.64	38.69	22.39	5.98
80th percentile	105.24	94.70	85.38	75.99	65.12	55.11	45.02	34.85	25.08	14.34	4.98

We plot table 4.3 in the following graph.

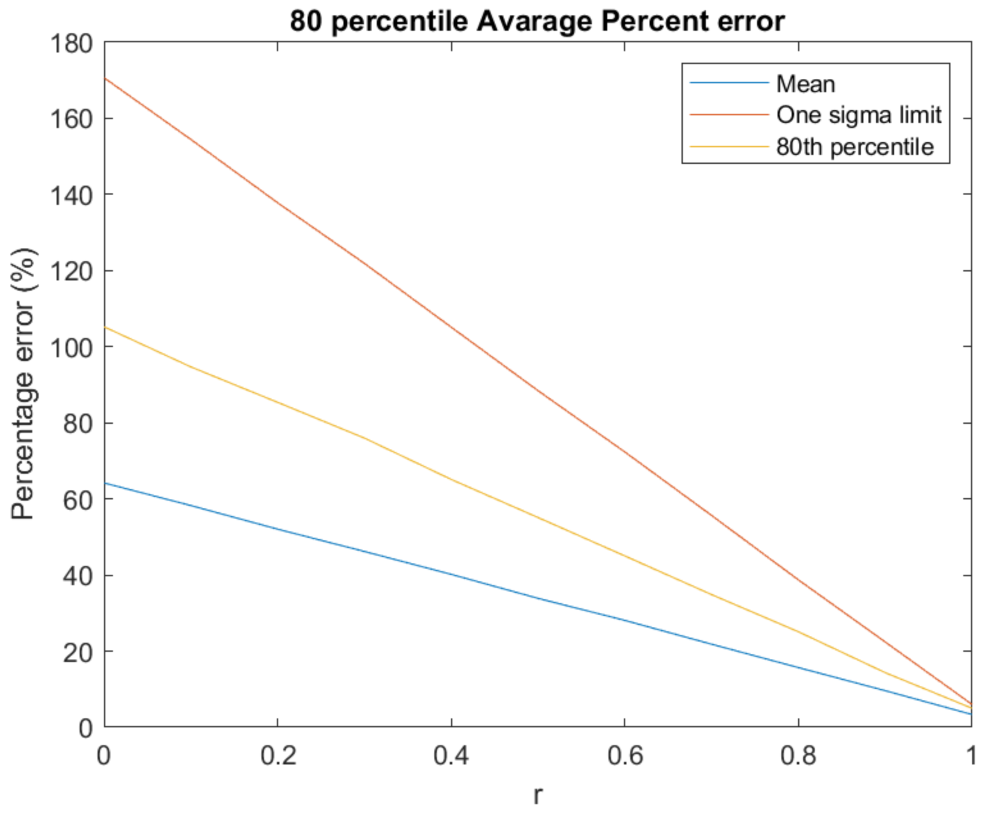

Fig. 4.5

As expected for all these cases, the error increases and the circuit becomes unusable when correlation is below a certain level. But still, if we assume that the 20% error is permissible without degrading the performance of the entire system, we can use SCC greater or equal to 0.72. However, the average error does not guarantee that it will always be less than 20%. If the circuit has a variety of inputs, this is more likely to be true. But, we have to be cautious because we could use the 80th percentile error. Then our usable SCC range will be in the range from 0.85 to 1.

If our criteria are strict enough so that the maximum error is not more than 20%, in that case, we have to take a 1 sigma limit. The 1 sigma limit is the summation of the mean and standard deviation. If we consider a 1 sigma limit for our 20% permissible error, our minimum SCC value would be 0.91.

4.3 Summary:

We can conclude from the simulation done in this chapter that the range of SCC from 0.72 to 1 can be used for any observed output value. If our criteria are strict for our system, the range of SCC will be from 0.91 to 1. In the previous analysis done on the AND gate, errors were minimised when SCC was low, but in this analysis done on the XOR gate, errors were minimised when SCC was high because the expected behaviour is with +1 SCC.

CHAPTER 5

OR GATE AS STOCHASTIC CIRCUIT ELEMENT

In this chapter, the theoretical outputs of OR gates are presented as how a two-input OR gate behaves within a correlation range from 0 to 1.

5.1 Usability and analytics of OR gate for stochastic computing operation

Let us consider a function,

$$\boldsymbol{OR(x, y) = f_{OR}(x, y)} \tag{5.1}$$

For any correlation,1> r > 0

$$\boldsymbol{f_{OR}(x, y) = (x+y-xy)(1-r) + r\, max(x, y)} \tag{5.2}$$

$$\boldsymbol{\tilde{e} = \%\ error = \frac{r\,((x+y-xy)-max(x,y))}{(x+y-xy)} = r\left(1-\frac{max(x,y)}{(x+y-xy)}\right)} \tag{5.3}$$

For any correlation, -1<r < 0

$$\boldsymbol{f_{OR}(x, y) = (x+y-xy)\ (1 - r) + r\, min(x + y, 1)} \tag{5.4}$$

$$\boldsymbol{\tilde{e} = \%\ error = = \frac{r\,((x+y-xy)-min(x+y,1))}{(x+y-xy)} = r\left(1-\frac{min(x+y,1)}{(x+y-xy)}\right)} \tag{5.5}$$

This will not happen in reality. Errors will fluctuate around this particular value (ẽ).

In practical, ẽ = % error = {e | e = error of 100 simulation for a given pair of x and y}

Let’s consider an example (1), x = 0.5, y = 0.7

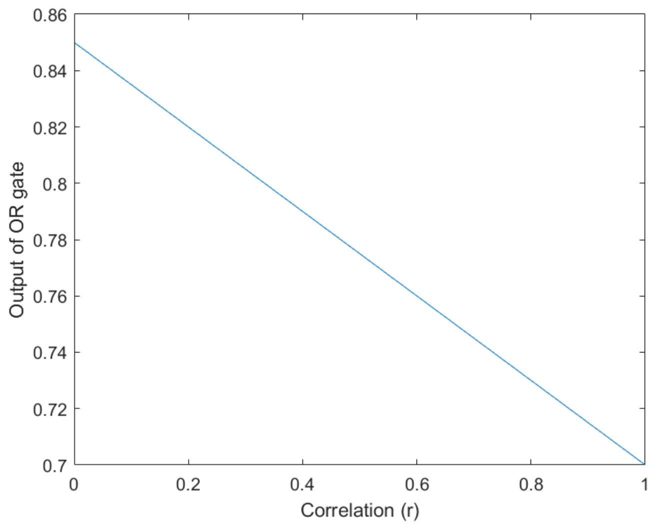

Fig. 5.1: Ideal output of OR gate for correlation 0 to 1

For 0 correlation, the ideal OR gate output is $\boldsymbol{x + y - x \times y = 0.5 + 0.7 - 0.5 \times 0.7 = 0.85}$, while for 1 correlation, the output is $\boldsymbol{max\ (x, y) = max\ (0.5, 0.7) = 0.7}$. In any other correlation, the output will be a weighted average of the two output values shown in the graph.

We want to achieve 0.85 as output. However, for correlation 1, we get 0.7, and for other correlations, we are getting values representing a line shown in the graph, which is in between 0.85 and 0.7.

We can consider a maximum tolerance output, beyond which output will not be useable. Putting x = 0.5 and y = 0.7 in equation 2, we get,

$$\tilde{e} = r\left(1 - \frac{0.7}{0.85}\right) = r\frac{0.15}{0.85} = \frac{3r}{17} \tag{5.6}$$

If we assume our maximum tolerance is 20%, then 20% of 0.85 = 0.17. So, the maximum output value that can be tolerated is 0.68, which is beyond our range, not in between 0.85 and 0.7. If we want to calculate r*, then r* will be

$$\frac{3r^*}{17} = 0.2\,,$$

$$or\; r^* = \frac{3.4}{3} = 1.13$$

Here our calculated correlation, $r^* = 1.13$, which is out of the range of 0 to 1. So, we can conclude that for any value of r we can use 0.5 and 0.7 as inputs.

When we run 100 simulations, 100 times output is generated for 0.5 and 0.7. For this particular correlation value, we get this.

Px	Py	Expected	r = 0	r = 0.25	r = 0.5	r = 0.75	r = 1
0.5	0.7	0.85	0.84913	0.81408	0.7731	0.73546	0.70001

Although, from theory, we get that any value of r can be used. Here, we also did 100 simulations, and we see whether we will get this or not.

Let us consider more examples, example (2), if X and Y are 0.3 and 0.8, the output is 0.86 when correlation is 0 and 0.8 when correlation is 1. If we consider that acceptable error is not more than 20%, then here 20% of 0.86 is 0.172 and our minimum acceptable error could be 0.688. In this case,

$$\mathbf{r}^*\left(1 - \frac{\max(0.3, 0.8)}{(0.3 + 0.8 - 0.3 * 0.8)}\right) = 0.2$$

$$\text{Or, } \mathbf{r}^* = 2.87$$

And in example (3), if X and Y are 0.5 and 0.9, the output is 0.45 when correlation is 0 and 0.5 when correlation is 1. In this case,

$$\mathbf{r}^*\left(\frac{1}{\min(0.5, 0.9)} - 1\right) = 0.2$$

$$\text{Or, } \mathbf{r}^* = 2.87$$

Again, we can see that, r* is beyond the correlation range of 0 to 1.

For correlation, r < 0

Let's consider an example (1), x = 0.5, y = 0.7

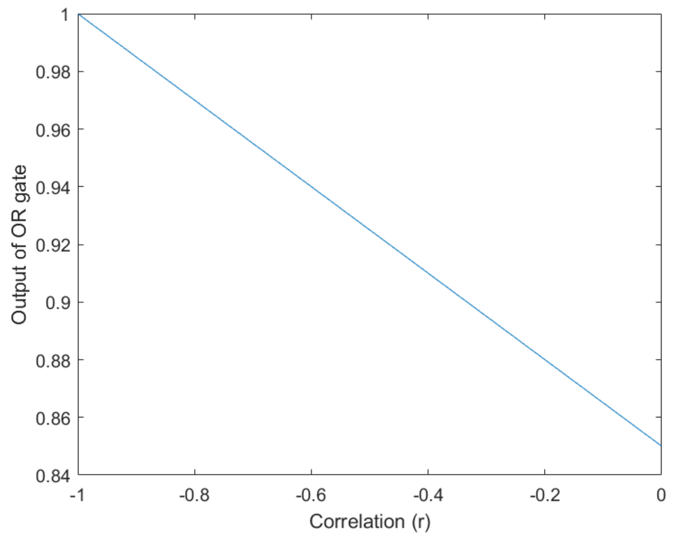

Fig. 5.2: Ideal output of OR gate for correlation -1 to 0

For a 0 correlation, the ideal OR gate output is ***x + y - x × y = 0.5 + 0.7 - 0.5 × 0.7 = 0.85***, while for a -1 correlation, the output is ***min (x + y, 1) = min (0.5 + 0.7, 1) = 1.0***. For any other correlation, the output will be a weighted average of the two output values shown in the graph.

We want to achieve 0.85 as output. However, for correlation -1, we get 0.2, and for other correlations, we are getting values representing a line shown in the graph, which is in between 1 and 0.85.

We can consider a maximum tolerance output, beyond which the output will not be usable. Putting ***x = 0.5*** and ***y = 0.7*** in equation 3, we get,

$$\tilde{e} = r\,\frac{(0.5+0.7-0.7*0.5)-min(0.5+0.7,1)}{0.5+0.7-0.5*0.7} = \frac{r(0.85-1)}{0.85} = -r\,\frac{0.15}{0.85} = -\frac{3r}{17} \qquad (5.7)$$

If we assume our maximum tolerance is 20%, then 20% of 0.85 = 0.17. So, the minimum output value that can be tolerated is 1.02. But, our range is 0.85 to 1, which is out of our range. So for any value of r*, we can use it. If we want to calculate r*, then

$$-\frac{3r^*}{17} = 0.2\,,$$

$$or\ r^* = \frac{3.4}{3} = -1.33$$

Again, r^* = -1.33, which is not in the range of 0 and 1. So, any value of correlation can be used.

Similarly, if X and Y are 0.3 and 0.8, the output is 0.86 when correlation is 0 and 0.1 when correlation is -1. In this case,

$$r^{*}\frac{(\mathbf{0.3+0.8-0.3*0.8})\ \boldsymbol{-min}(\mathbf{0.3+0.8,1})}{\mathbf{0.3+0.8-0.3*0.8}}=\mathbf{0.2}$$

$$\boldsymbol{or}, \boldsymbol{r}^{*}=\mathbf{1.23}$$

Now, in theory, for any application, we can use an OR gate for any correlation from -1 to 1. This is a theoretical analysis. In practical, a simulation has been done for all possible values of X and Y, so that we can see the OR gate behaves similarly as we have studied in theory.

5.2 Simulation results

In this chapter, a similar study has been done on OR gate. In this scenario, a similar worst-case scenario has been considered on the OR gate for our simulation. For our worst case scenario, we use an 80 percentile error from 100 samples of every combination of Px and Py with step size 0.1 and every SCC from 0 to 1 with step size 0.1.

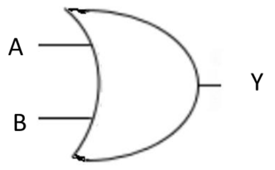

Fig. 5.3 OR gate

In this analysis, function of OR gate is given below,

$$\boldsymbol{p(z) = 1 - (1 - p(x))(1 - p(y))} \tag{5.8}$$

where p(z) is output and p(x) and p(y) are inputs.

The following 2 tables are the results of the simulation done on different SCC ranges from r = 0 to r = 1 with step size 0.1 and different Px and Py from 0 to 1 with step size 0.1.

Table 5.1

Px	Py	r = 0	r = 0.1	r = 0.2	r = 0.3	r = 0.4	r = 0.5
0	0	NaN	NaN	NaN	NaN	NaN	NaN
0	0.1	8.89	8.4	8.4	6.935	6.45	7.42
0	0.2	6.2	5.225	4.98	5.225	4.2475	4.005
0	0.3	4.4933	4.655	3.5167	5.1433	3.3517	5.1433
0	0.4	3.515	2.54	4.2475	2.7838	2.4175	1.9287
0	0.5	1.954	2.539	2.93	2.247	2.832	2.93
0	0.6	3.0267	2.4575	2.4575	2.1317	1.9692	1.8067
0	0.7	1.4229	1.2829	1.8414	1.5629	1.9814	1.6321
0	0.8	1.2575	1.0737	1.1962	1.5625	1.3187	1.5012
0	0.9	0.74889	0.85722	1.02	0.69444	0.80333	0.80333
0	1	0	0	0	0	0	0
0.1	0.1	6.3947	0.28947	3.1158	10.311	14.937	19.305

0.1	0.2	4.2821	1.3179	1.1232	5.4821	6.8786	11.064
0.1	0.3	3.3311	0.82432	0.23243	2.2122	4.7189	6.8297
0.1	0.4	2.963	1.5837	0.32717	0.64565	2.9804	3.4043
0.1	0.5	2.5391	1.4736	0.052727	0.30182	1.7218	2.5218
0.1	0.6	1.9281	0.63125	0.78438	0.054687	0.13125	1.1992
0.1	0.7	1.8699	1.1342	0.86712	0.46575	0.069178	0.064384
0.1	0.8	1.2884	1.2293	0.57439	0.87195	0.45488	0.080488
0.1	0.9	0.76813	0.66154	0.44615	0.5	0.76813	0.55385
0.1	1	0	0	0	0	0	0
0.2	0.2	3.7597	2.0722	5.7347	9.8028	13.736	18.756
0.2	0.3	3.2045	0.34545	3.675	6.1159	9.4455	11.776
0.2	0.4	3.2904	0.47308	2.25	4.9731	7.7894	8.9163
0.2	0.5	2.4575	0.065	1.205	3.4017	3.9708	6.4942
0.2	0.6	1.8206	0.81618	0.47647	1.7691	2.4875	3.9235
0.2	0.7	1.4467	0.74079	0.22368	0.67368	1.8941	1.8296
0.2	0.8	1.144	0.3881	0.15595	0.25119	0.65833	1.2393
0.2	0.9	0.94674	0.68152	0.25707	0.044565	0.11413	0.22065
0.2	1	0	0	0	0	0	0
0.3	0.3	2.8255	1.4824	5.7902	9.3333	13.928	17.278
0.3	0.4	2.5397	1.081	3.775	6.7207	9.9207	12.614
0.3	0.5	2.1631	0.61615	2.9446	5.1985	7.0762	9.1046
0.3	0.6	1.4542	0.0375	1.0556	3.0222	4.9889	6.6833
0.3	0.7	1.0557	0.19051	0.67468	2.0962	3.2089	4.2595
0.3	0.8	1.2331	0.43837	0.3	0.86744	1.3785	2.1733
0.3	0.9	0.85914	0.38602	0.12366	0.070968	0.45376	0.82097
0.3	1	0	0	0	0	0	0
0.4	0.4	2.6922	1.5813	5.7	9.2867	12.72	16.992
0.4	0.5	1.9114	1.5071	3.9479	6.6686	9.3186	12.179
0.4	0.6	1.1895	0.48026	2.6651	4.7855	6.4553	8.6395
0.4	0.7	1.5268	0.25976	1.5098	2.9988	4.4878	6.2146
0.4	0.8	1.0966	0.31932	1.0676	1.567	2.5659	3.5648
0.4	0.9	0.7734	0.35745	0.1617	0.78511	0.94149	1.617
0.4	1	0	0	0	0	0	0
0.5	0.5	1.4973	1.8227	5.0787	8.138	11.979	14.453
0.5	0.6	1.1962	1.55	3.7475	6.25	8.4475	11.255
0.5	0.7	1.0459	0.85059	2.3435	4.0671	5.7329	7.5141
0.5	0.8	0.80333	0.17389	1.4756	2.2356	3.7	4.6767
0.5	0.9	0.74	0.082105	0.28737	0.85316	1.4189	1.9842
0.5	1	0	0	0	0	0	0
0.6	0.6	0.96964	1.8202	4.4369	7.2845	9.9	12.807
0.6	0.7	0.875	0.95682	3.2318	5.2295	7.171	9.0023
0.6	0.8	0.84022	0.64565	1.7065	2.9804	4.1478	5.4217
0.6	0.9	0.50417	0.10625	0.61458	1.2755	2.0385	2.2417
0.6	1	0	0	0	0	0	0
0.7	0.7	0.71484	1.4857	3.3632	5.778	8.1923	10.5
0.7	0.8	0.72128	0.57766	2.1362	3.6426	5.0447	6.6032

0.7	0.9	0.47526	0.2299	0.88402	1.6892	2.2428	2.8974
0.7	1	0	0	0	0	0	0
0.8	0.8	0.40313	1.1229	2.5979	4.276	5.65	7.5312
0.8	0.9	0.29694	0.40051	1.048	1.8954	2.6923	3.3398
0.8	1	0	0	0	0	0	0
0.9	0.9	0.31919	0.56768	1.3081	2.1465	3.1333	3.8727
0.9	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0

Table 5.2

Px	Py	r = 0.6	r = 0.7	r = 0.8	r = 0.9	r = 1
0	0	NaN	NaN	NaN	NaN	NaN
0	0.1	5.47	10.84	8.4	9.38	12.3
0	0.2	4.98	5.225	5.955	5.955	4.735
0	0.3	4.8167	4.8167	4.1667	3.5167	4.1667
0	0.4	4.2475	3.3938	3.15	3.2725	3.2725
0	0.5	2.734	2.832	2.734	2.93	2.344
0	0.6	2.05	1.725	2.4575	2.2133	2.2133
0	0.7	1.9107	1.4929	1.6321	1.7014	1.8414
0	0.8	1.135	1.685	1.5625	1.44	1.2575
0	0.9	0.91111	0.74889	0.91111	0.80333	0.80333
0	1	0	0	0	0	0
0.1	0.1	22.132	28.813	33.184	37.295	42.689
0.1	0.2	13.157	17.339	19.432	22.921	24.318
0.1	0.3	7.8865	9.9973	11.714	13.165	15.541
0.1	0.4	4.7848	5.3152	7.9696	7.7576	10.835
0.1	0.5	3.9418	4.2964	3.9418	5.6291	7.3145
0.1	0.6	1.7336	3.0305	3.1062	4.0984	4.0219
0.1	0.7	0.80548	1.0726	1.9425	2.3438	2.4781
0.1	0.8	0.2	0.31951	0.91463	0.85488	1.689
0.1	0.9	0.23187	0.17802	0.036264	0.017582	0.14396
0.1	1	0	0	0	0	0
0.2	0.2	23.367	28.114	32.59	37.067	41.406
0.2	0.3	15.439	20.432	23.43	26.202	28.534
0.2	0.4	11.171	13.423	15.865	18.119	20.654
0.2	0.5	8.04	9.4233	10.888	12.435	14.713
0.2	0.6	5.0721	6.5801	8.0162	9.0221	10.171
0.2	0.7	2.7289	4.0145	4.7211	5.4276	6.0697
0.2	0.8	1.7625	2.053	2.7506	3.0994	3.506
0.2	0.9	0.27391	0.53913	1.0696	1.0168	1.2826
0.2	1	0	0	0	0	0
0.3	0.3	21.684	25.992	30.875	34.704	38.822
0.3	0.4	16.571	19.434	22.717	25.495	29.367
0.3	0.5	11.658	14.213	16.617	18.945	21.198
0.3	0.6	8.5153	9.9389	11.363	12.787	14.958
0.3	0.7	5.1873	6.6089	7.5354	9.019	10.317

0.3	0.8	3.1953	3.593	4.5012	4.8419	5.7506
0.3	0.9	1.136	1.2935	1.714	1.8183	2.3962
0.3	1	0	0	0	0	0
0.4	0.4	20.426	24.24	27.52	31.717	35.303
0.4	0.5	15.527	17.899	21.108	24.247	26.619
0.4	0.6	11.274	13.137	14.422	16.928	19.241
0.4	0.7	7.1671	9.2512	9.847	12.05	13.3
0.4	0.8	4.3409	5.4511	6.283	7.2261	7.3375
0.4	0.9	1.8245	2.2394	2.7074	2.7596	3.3309
0.4	1	0	0	0	0	0
0.5	0.5	17.904	21.093	25	27.864	32.227
0.5	0.6	13.453	15.65	18.335	20.776	23.279
0.5	0.7	9.0071	11.018	12.454	14.235	16.015
0.5	0.8	5.2733	6.7922	7.8233	8.9622	9.7222
0.5	0.9	2.5495	2.8579	3.68	3.9884	4.4
0.5	1	0	0	0	0	0
0.6	0.6	16.294	19.143	21.526	23.793	26.99
0.6	0.7	10.999	13.052	14.828	17.103	19.378
0.6	0.8	6.6957	8.3413	9.35	10.57	11.685
0.6	0.9	2.9542	3.7677	4.2255	4.6833	5.5479
0.6	1	0	0	0	0	0
0.7	0.7	12.538	15.007	17.099	19.729	21.499
0.7	0.8	7.9537	9.5117	10.967	12.577	13.564
0.7	0.9	3.5015	4.3577	4.9608	5.7165	6.3711
0.7	1	0	0	0	0	0
0.8	0.8	9.1594	10.94	12.109	13.941	15.365
0.8	0.9	4.2367	5.1337	5.8316	6.7286	7.2265
0.8	1	0	0	0	0	0
0.9	0.9	4.8101	5.747	6.3879	7.4727	8.4596
0.9	1	0	0	0	0	0
1	1	0	0	0	0	0

As we can see, the results obtained from our simulation show that percentage errors are low for maximum cases even when r = 1. For Px and Py are 0.2 and 0.7, when r = 0.1, percentage error is 0.74 and when r = 1, error is 6.07. Similarly, for Px and Py are 0.3 and 0.7, when r = 0.1, error is 0.19 and when r = 1, error is 10.32. That means accuracy is greater. But, in cases like when Px and Py are 0.4 and 0.4, and r = 1, the error is 35.3. From the tables, we can see that percentage error value is varies from 43% to 1%. Although the percentage variation is not that much as we have seen in previous chapters.

However, since Px and Py are not known in advance, we have to look for the average error for all possible Px and Py. That is done and shown in the following plots.

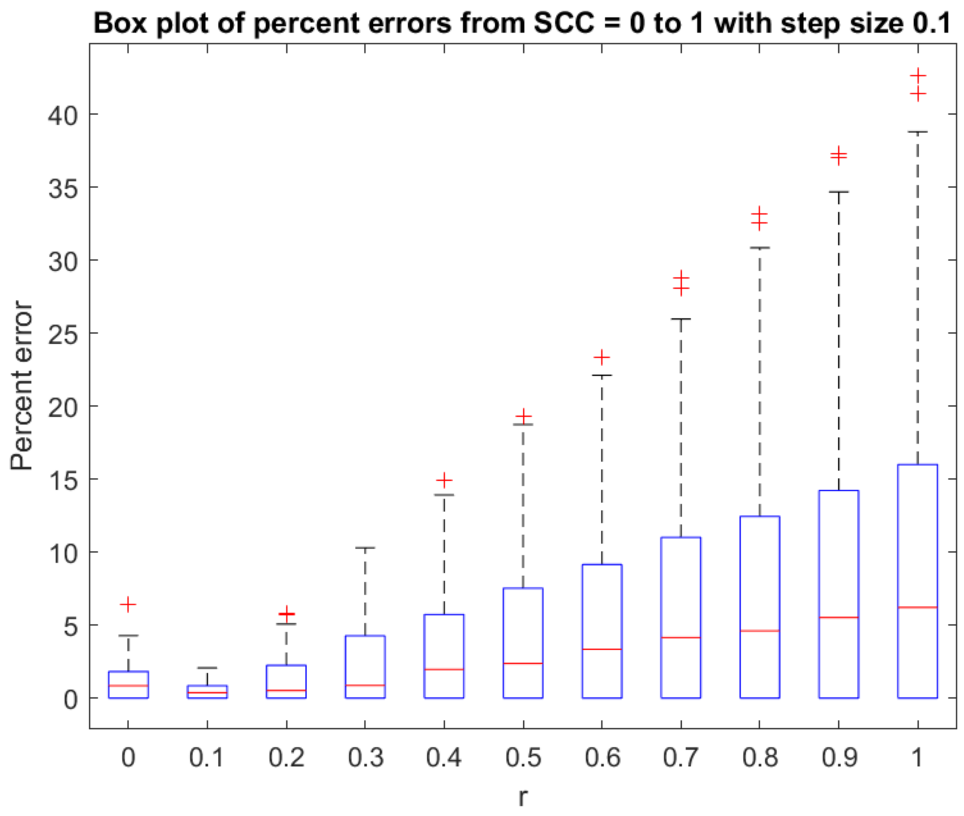

Fig. 5.4

In the graph shown in fig. 5.2, it can be seen that the highest error for a particular SCC is greater compared to the mean of 80 percentile percent error, and the gap between the mean of percent error and the highest error is increased with the increase of SCC. When SCC = 0.1, the error is the smallest.

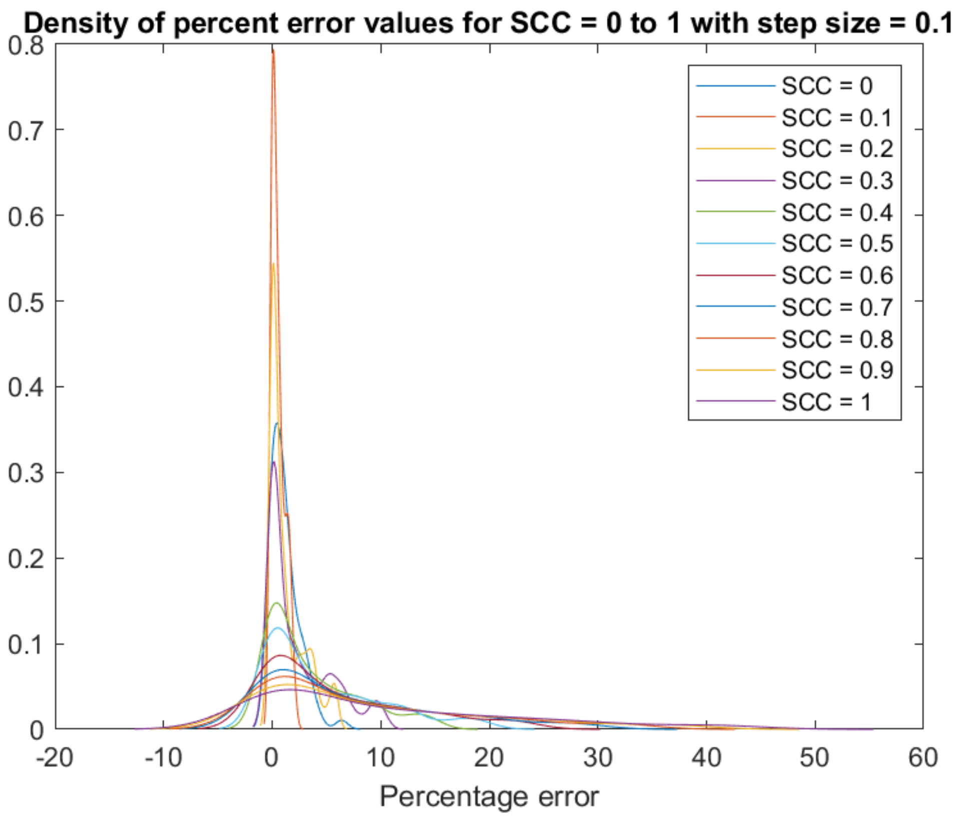

Fig. 5.5

In the graph shown in fig 5.3, it can be seen that the errors are increasing with the increase of SCC.

The following table represents the mean, 1 sigma limit, and 80th percentile for each column representing particular SCC from table 5.1 and table 5.2.

Table 5.3

	r = 0	r = 0.1	r = 0.2	r = 0.3	r = 0.4	r = 0.5	r = 0.6	r = 0.7	r = 0.8	r = 0.9	r = 1
Mean	1.63	0.98	1.76	2.84	3.89	5.02	6.16	7.48	8.55	9.67	10.88
1 Sigma Limit	3.31	2.33	3.64	5.71	7.87	10.19	12.50	15.21	17.48	19.83	22.30
80th Percentile	2.73	1.49	3.41	5.31	7.10	9.03	11.39	13.66	16.09	18.37	20.82

We plot table 5.3 in the following graph.

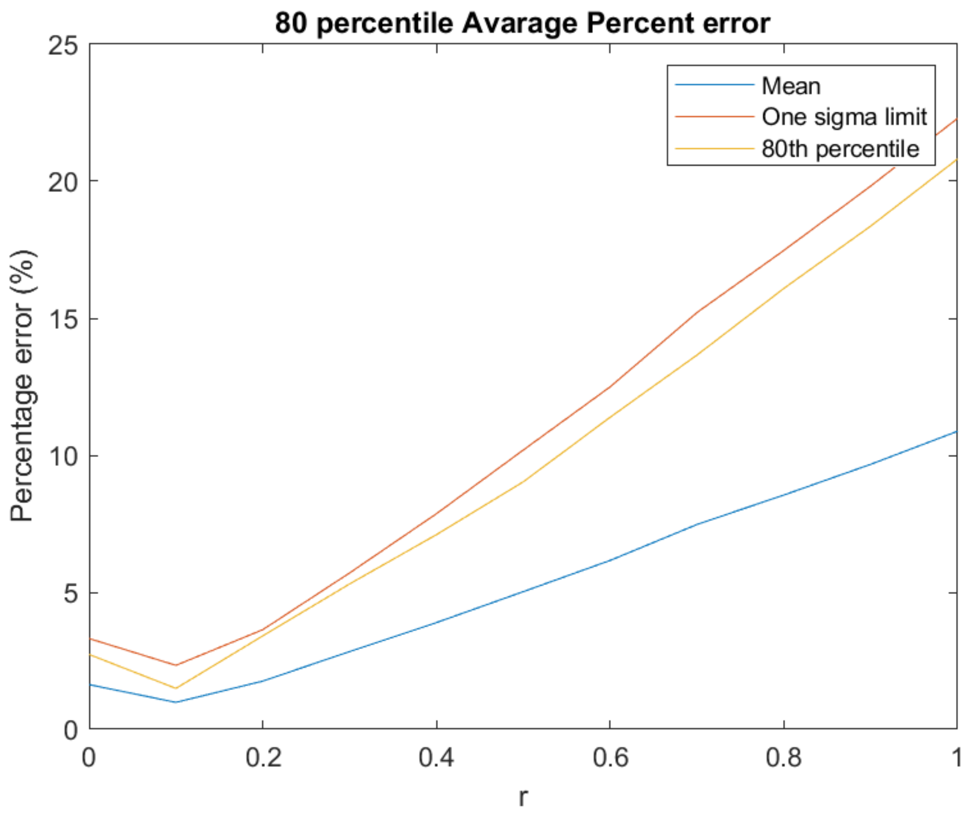

Fig. 5.6

As expected in all these cases, the error increases when correlation increases, but the circuit is somehow usable for all correlations. But for strict criteria, that means a 20% error threshold, we can use any ACC value from 0 to 0.9.

5.3 Summary:

We can conclude from the simulation done in this chapter that any range of SCC can be used for any observed output value. When SCC = 1, the error is a maximum of 10.38 percent, which is within the 20 percent threshold. Previous analysis done on the AND gate and XOR gate, we can conclude that when AND can be used when SCC is minimum, XOR can be used when SCC is maximum, but the OR gate can be used for any SCC value.

CHAPTER 6 APPLICATION CASE STUDIES

Edge detection functions are a very important part of image processing, which is very useful for machine learning, deep learning, and other important scenarios. Towards that, stochastic computing is found to be very useful [4]. However, in this application case study of image edge detection, there is a requirement for absolute subtraction. For that, Sobel and Canny edge detectors are used. Since that is achieved in stochastic computing by perfect correlation among the input values. There lies a scope of analysis of the results that we obtained in this thesis and check the tolerance of the mentioned circuit. The perfect correlation scenario is not maintained due to randomness.

6.1 Image Edge detection:

From the simulation done in chapter 4, it is clear that the range of SCC from 0.72 to 1 can be used for any observed output value for the XOR gate. If our criteria are strict for our system, the range of SCC will be from 0.91 to 1. Here, in this application, the XOR gate and stochastic adder are used for our edge detection.

For our computation purposes in this chapter, we have used the images given in Figure 6.1 and Figure 6.8. We created new 256x256 resized images and then used the circuit given in figure 6.2 to detect edges. To compare these with the benchmark result, we used Canny edge detection and Sobel edge detection. We considered each pixels as(x_i, y_j), where 1st pixel is(x_1, y_1), 2nd pixel is (x_1, y_2) and so on. Here, the value of each pixel, which is 0 to 256, is scaled to 0 to 1 for our simulation purpose.

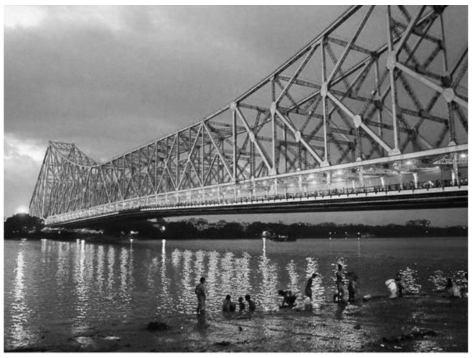
Fig. 6.1

The formula used for edge detection is

$$\boldsymbol{I(x_i, y_j)' = \frac{1}{2}\{|(x_i, y_j) - (x_{i-1}, y_{j-1})| + |(x_{i-1}, y_j) - (x_i, y_{j-1})|\}}$$

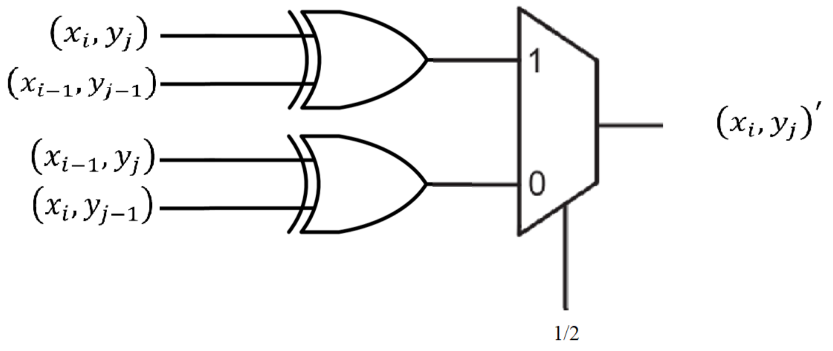

Fig 6.2 Circuit Diagram of the Edge detection circuit

Here, 1024 bits are used in stochastic computing.

The below images are the output of Sobel and Canny edge detection methods in MATLAB for the image shown in figure 6.1.

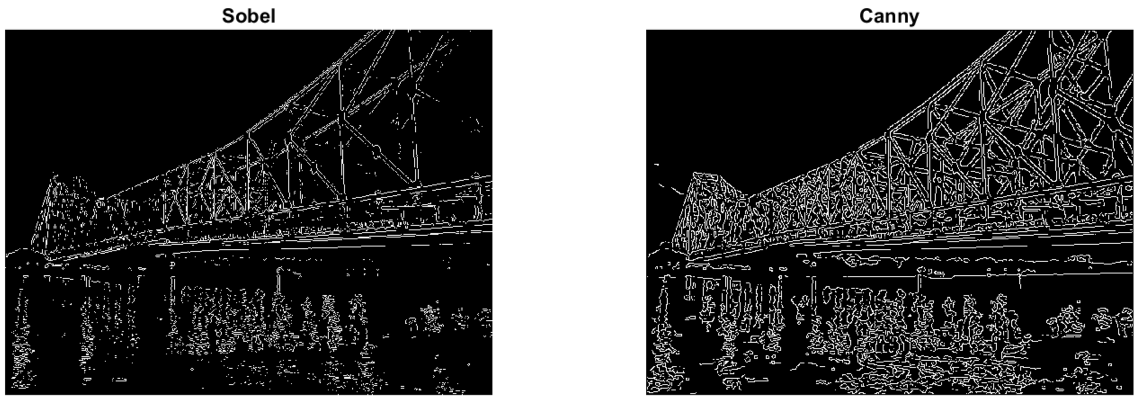

Fig 6.3 Output images of 2 edge detection operator from MATLAB for figure 6.1, Sobel (left) and Canny (right)

We will consider these outputs as our target outputs.

For uncorrelated inputs, that means when correlation, r = 0, we get the following result.

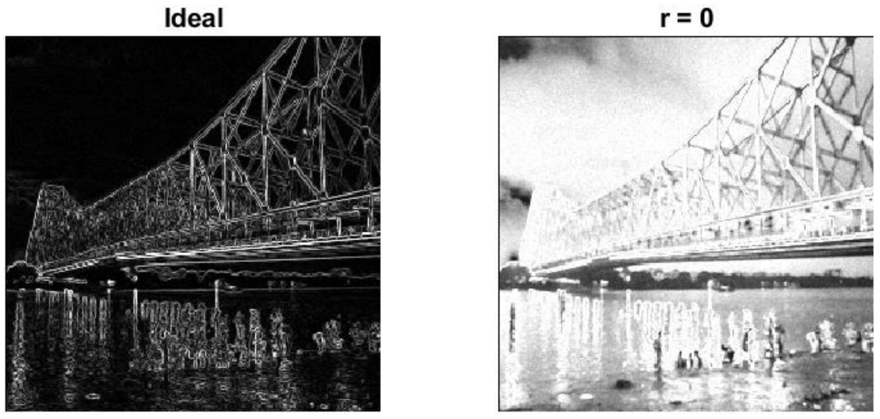

Fig 6.4 Comparison of 2 output images ideal (left) with r = 0 (right)

When correlation r = 0.5, the output is unusable.

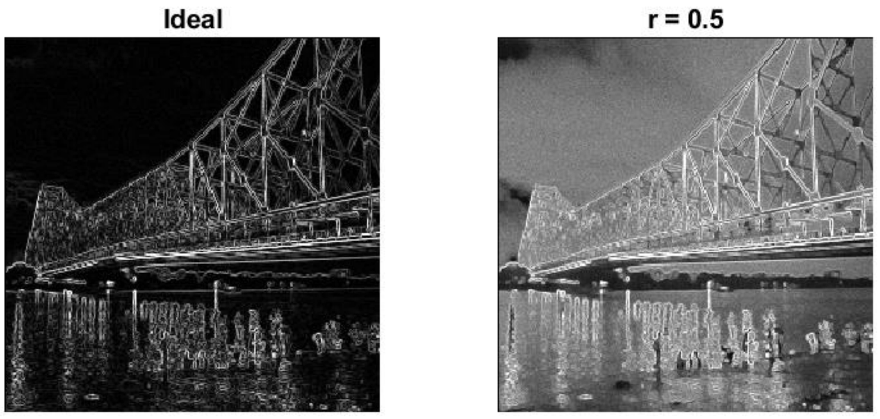

Fig 6.5 Comparison of 2 output images ideal (left) with r = 0.5 (right)

We can see in the two images shown above that when correlation r = 0.5, that means when r < 0.72, much noise is present in the output after simulation.

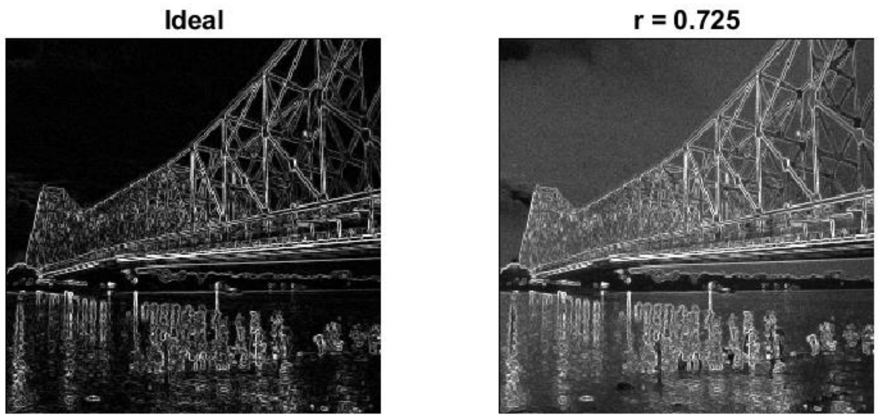

Fig 6.6 Comparison of 2 output images ideal (left) with r = 0.725 (right)

In the figure shown above, we can see that there is little noise present when correlation r = 0.725, which is greater than 0.72. But, noise is not much, and the circuit is usable if our error threshold is 20%.

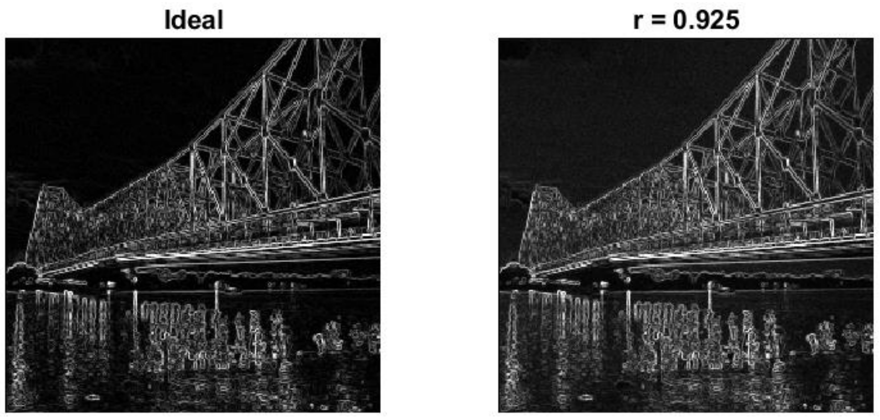

Fig 6.7 Comparison of 2 output images ideal (left) with r = 0.925 (right)

In the figure shown above correlation is 0.925, if our criteria is strict which means correlation is greater than 0.91, we can use this output as visible similar to ideal output shown in the figure.

Let's see some more examples.

The below images are outputs of the edge detection method of Sobel and Canny in MATLAB for a different image shown in figure 6.8.

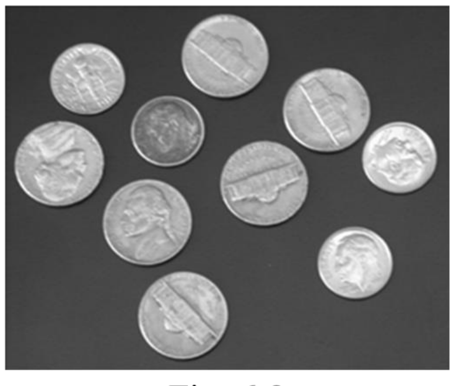

Fig. 6.8

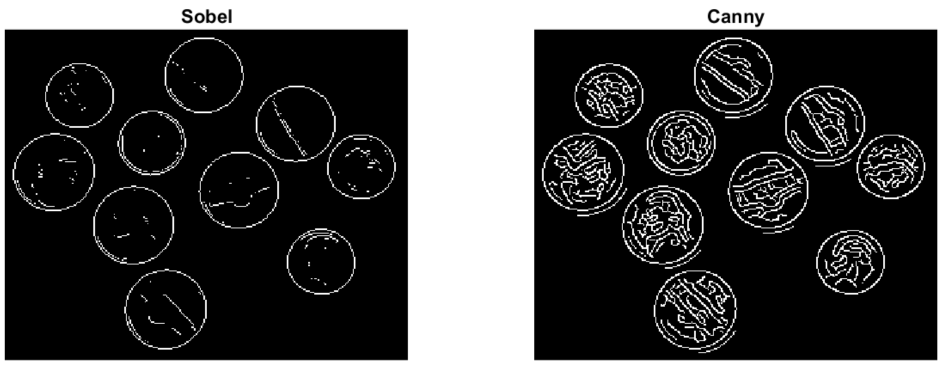

Fig 6.9 Output images of 2 edge detection operation in MATLAB, Sobel (left) with Canny (right)

In the following example, we can see the same as in the previous example when correlation r = 0.

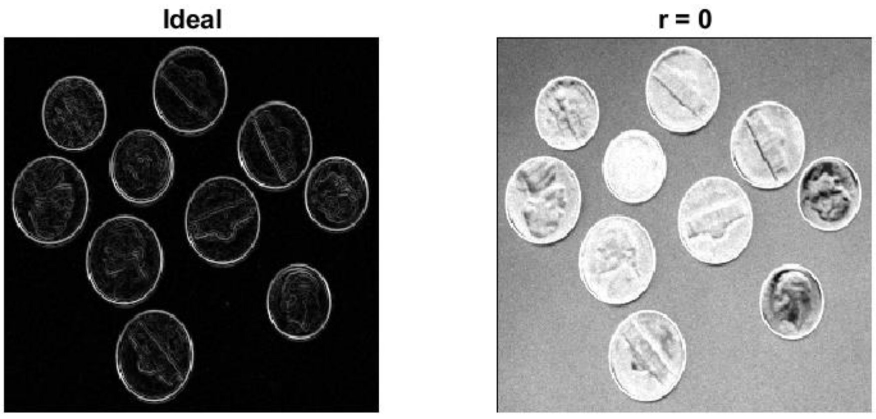

Fig 6.10 Comparison of 2 output images ideal (left) with r = 0 (right)

When correlation r = 0.5, the output is not usable.

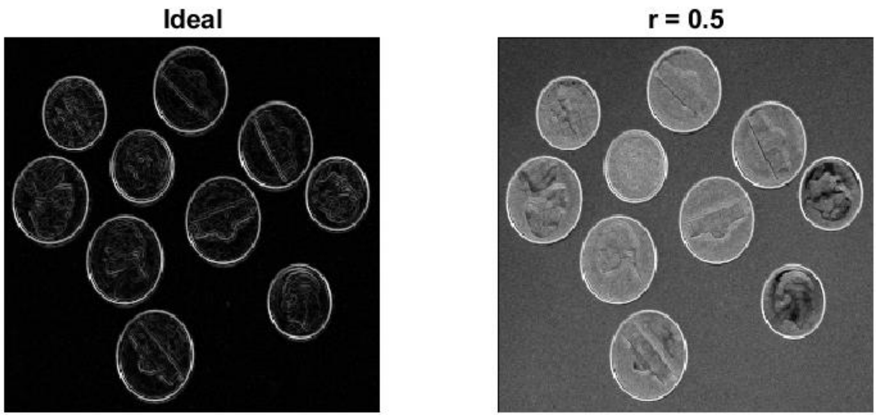

Fig 6.11 Comparison of 2 output images ideal (left) with r = 0.5 (right)

But, when correlation r = 0.725, that means more than 0.72, output can be used as shown in the following image.

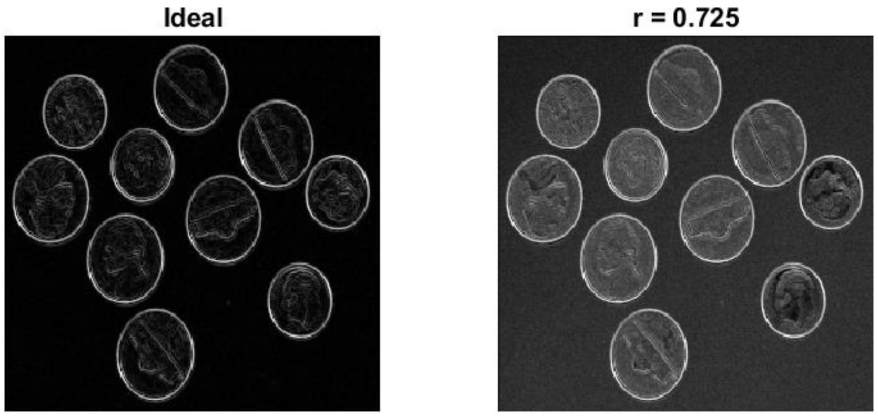

Fig 6.12 Comparison of 2 output images ideal (left) with r = 0.725 (right)

But, but there is some noise present when the correlation is 0.725. If we use the strict criteria limit of correlation greater than 0.91, our output is noise free, as shown below.

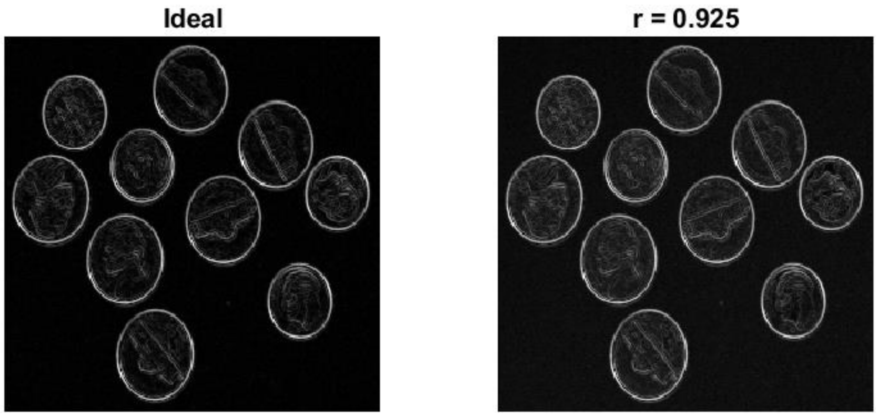

Fig 6.13 Comparison of 2 output images ideal (left) with r = 0.925 (right)

Similarly, in the figure shown above, the correlation is 0.925. If our criteria are strict, which means the correlation is greater than 0.91, we can use this output as visible similar to ideal output shown in the figure.

6.2 Summary

From the above simulations, we can conclude that practically the entire range of SCC from 0.72 to 1 can be used for any observed output value for an XOR gate, and the range of SCC from 0.91 to 1 can be used for strict criteria.

CHAPTER 7 CONCLUSION AND FUTURE SCOPE

In this thesis, it was shown that under the change in correlation value, we can always find a certain threshold for different gates like AND, OR, XOR, and those results and figures that are mentioned in the chapters of respective gates can lead to good results even if the correlation value is not exactly what is required. We can conclude from the simulation which is done for the AND gate, that the SCC value up to +0.04 can be used for the operation of multiplication. This limit can be extended up to 0.16 if our criteria are not strict. For the XOR gate for the operation of absolute subtraction, our limit is +0.91 from +1 and can be extended to +0.72 if the criteria are not strict. For the OR gate for the operation of the function x + y - xy, any value can be used if our criteria are not strict. In our analysis, our error limit is 20% of the ideal output. The results of these values are shown in tables 3.3, 4.3, and 5.3, respectively. From these tables, another limit of SCC can be used if our error in actual output can be tolerated more or less than 20% as our requirements.

In today's world, more complex operations are required for our tasks. It can be one particular complex operation or a multi-stage operation. With the help of the analysis done in this thesis, multi-stage operations can also be done. A similar type of study can also be done for the same gates for different functions with the change in SCC values. Stochastic computing has great potential in small functional circuits, like IoT devices, as their function requirements are limited.

REFERENCES

[1] Armin Alaghi and John.P.Hayes, “Survey of Stochastic Computing,” ACM Trans Embed. Comp. Syst., 12(2s):92:1-92:19, May 2013

[2] Te-Hsuan Chen, Armin Alaghi, and John P. Hayes, “Behavior of stochastic circuits under severe error conditions”, it - Information Technology, 56(4): 182–191, July 21, 2014

[3] A. Alaghi and J.P. Hayes, “Exploiting Correlation in Stochastic Circuit Design,” Proc. Intl Conf. on Computer Design (ICCD), pp. 39–46, Oct. 2013.

[4] Alaghi, A. Li, C. and Hayes, J.P., “Stochastic circuits for real-time image-processing applications,” Proc. DAC, paper 136, 2013.

[5] A. Alaghi and J.P. Hayes, “A Spectral Transform Approach to Stochastic Circuits,” Proc. Intl. Conf. Computer Design (ICCD), pp. 315–312, 2012.

[6] Bahram Dehghan , Naser Parhizgar, “Survey the stochastic computing and probabilistic transfer matrix (PTM) in logic circuits”, International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 12, December 2014

[7] Krishnaswamy, S., Viamontes, G.F., Markov, I.L. and Hayes, J.P., “Probabilistic transfer matrices in symbolic reliability analysis of logic circuits,” ACM Trans. Design Automation of Electronic Systems, 13, no. 1, pp.8:1-8:35, Jan. 2008.

[8] A. Alaghi and J. P. Hayes, “Fast and Accurate Computation Using Stochastic Circuits,” Proc. Design, Automation, and Test in Europe Conf. (DATE), pp. 1-4, 2014.

[9] Z. Zhao and W. Qian, “A General Design of Stochastic Circuit and its Synthesis,” Proc. Design, Automation and Test in Europe Conf. (DATE), pp. 1467-1472, 2015.

[10] Li, P. and Lilja, D.J., “Using stochastic computing to implement digital image processing algorithms,” Proc. ICCD, pp. 154-161, 2011.

[11] B.R. Gaines, “Stochastic Computing Systems,” Advances in Information Systems Science, vol. 2, J.T. Tou (ed.), Springer-Verlag, pp. 37-172, 1969.