3-2-2021

An Analytical Model for Circuit Reliability Estimation

Khawja Zaid Sikander
University of Windsor

An Analytical Model for Circuit Reliability Estimation

By

Khawja Zaid Sikander

A Thesis
Submitted to the Faculty of Graduate Studies
through the Department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Science
at the University of Windsor

Windsor, Ontario, Canada

2020

An Analytical Model for Circuit Reliability Estimation

by

Khawja Zaid Sikander

APPROVED BY:

_____

G. Guo
Odette School of Business

_____

S. Chowdhury
Department of Electrical and Computer Engineering

_____

C. Chen, Advisor
Department of Electrical and Computer Engineering

December 10, 2020

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

With the continuous scaling of CMOS technologies, integrated circuits are becoming more sensitive to process variations and/or external factors such as temperature or background noise and, as a result, may operate unreliably. Circuit reliability can, however, be improved by making some design changes, and this requires an efficient and accurate method for evaluation of the reliability during the design stage. Reliability of a circuit can be estimated using either simulation-based or analytical based methods.

While simulation-based methods (such as Monte-Carlo Simulation) can produce near to perfect estimated values, it takes a long time to run and the run-time increases exponentially with circuit size. On the other hand, analytical methods are relatively fast, but their accuracy level could decrease significantly if signal correlations are not properly accounted for.

In this thesis, a new method for calculating the signal reliability correlation coefficient is presented. The proposed method takes advantage of the local information available in the circuit. It is assumed that all signal probabilities of error-free circuits are already available. The proposed method to calculate the Correlation Coefficient was tested for its efficiency and accuracy by applying it on large circuits in comparison with the results produced by the Monte-Carlo Simulation. For circuits containing thousands of gates, their output reliability and probability can be calculated using the proposed method within minutes, as opposed to over 10 hours using Monte-Carlo Simulation. The average errors are as low as 1.67% when all gate reliability is set at 0.95. Throughout the thesis, various ISCAS85 benchmark circuits have been tested under different conditions such as different gate reliabilities, input signal probability and input signal reliabilities in comparison with Monte-Carlo Simulation in order to verify its validity.

ACKNOWLEDGEMENTS

I would like to show my sincerest appreciation to my research supervisor Dr. Chunhong Chen for his continued support and guidance throughout my program.

I would also like to thank Dr. Sazzadur Chowdhury and Dr. Guangrui Guo for providing me their precious feedbacks.

Finally, I would like to show my deepest gratitude towards my parents, Hafiz Sikander and Armana Sikander, for giving me the chance to study at the University of Windsor, supporting me financially to complete my studies without any worries and also encouraging me every day.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CMOS – Complementary Metal Oxide Semiconductor

MC – Monte-Carlo

PGM – Probabilistic Gate Models

PTM – Probability Transfer Matrices

SPRA – Signal Probability-based Reliability Analysis

ER – Equivalent Reliability

CHAPTER 1
Introduction

1.1 Motivation

With the continuous scaling down of the CMOS transistor size and the growing complexity of the integrated circuits, the circuits are now more prone to be affected by external and internal factors, making the operation more unreliable. The unreliable operation may be caused due to some hard or soft errors. Hard errors are the permanent errors that will be present constantly throughout the life cycle of the circuit and are normally caused by some physical factors. Soft errors are the random errors that get introduced in form of some random bit flip due to some external (such as temperature, background radiation etc.) or internal (such as signal crosstalk, bit flip due to unreliable gate etc.) factors [1].

Soft errors are not permanently present and may occur randomly at different location of the circuit, making it harder to have an absolute value. They can, however, be estimated via some computer-aided analysis. Currently there are mainly two types of analysis, i) Simulation-based Reliability Analysis and ii) Analytical-based Reliability Analysis. These analyses can be performed to give a close to accurate estimation. Simulation-based Reliability Analysis can produce estimations that are very close to real world data. However, since simulation requires a large number of iterations to give a more accurate result, the run-time for any simulation-based reliability analysis increases exponentially with the size and complexity of the circuit [2].

Throughout the thesis, we will be focusing mainly on the errors produced by the soft errors. Since we will be using the ISCAS85 combinational benchmark circuits throughout the thesis, we can assume that the only source of error that gets introduced into

the circuit is via the unreliable gates. The wires will be assumed to be 100% reliable, as an unreliable wire can be represented as a reliable wire followed by an unreliable buffer.

If no signal correlations exist in a circuit (i.e., all the signals are independent of each other), then calculating the signal reliability and probability becomes simple and can be calculated without errors using any of the analytical methods. Unfortunately, that is rarely the case in real world, thus motivating researchers to work towards finding a fast and accurate solution.

1.2 Background and Past Work

Reliability of a logic circuit is defined as the probability that the outcome of the circuit will be a correct value. Due to the possibility of error that may be present in the driving gates of any signal, the signal may become unreliable and produce an incorrect outcome. According to Von Neumann [3] the existence of error will cause the output bit to flip ($0 \rightarrow 1$ or $1 \rightarrow 0$). The probability of error will range from 0 to 0.5 (where 0 means the circuit is error-free and 0.5 means a total random operation).

There have been a lot of work in the past to calculate the reliability and probability of an integrated circuit using an analytical model. Probability Transfer Matrix (PTM) [4], Probabilistic Gate Model (PGM) [5], Signal-Probability based Reliability Analysis (SPRA) [6], Equivalent Reliability (ER) model [7], Three-point method [8] etc. are some examples of the methods that utilize equations and models to calculate the reliability and probability of an integrated circuit in linear time. Each method mentioned above has their own advantages and disadvantages.

In PTM method, the circuit is separated into levels and the signals are analyzed to find the probability matrices of each signal. The signals are then propagated from level to level, taking every possible combination into account to finally produce the PTM of the circuit. Since this method takes every scenario into account, the effect due to correlation coefficient will be addressed and so the estimated reliability and probability is very close to the Monte-Carlo Simulation results. However, since there are a lot of signal probability matrices present at the same time in each level, the method requires a large memory overhead. Even though work has been done to efficiently reduce the memory requirement while keeping the accuracy, the method becomes very expensive to run for large practical circuits [4].

SPRA uses the PTM method to calculate the probability of the signal states and once the state probability matrix (SPM) has been calculated, it gets propagated to the next gate until it reaches the final output gates. SPRA assumes that the two signals are independent of each other and so if the two signals are correlated then the output SPM will be incorrect. Because the SPM are propagated from one gate to the next, even one wrong SPM will increase the error significantly as the signal passes through the circuit. The correlation issue in SPRA was dealt with in [6] where the author generated bit stream in order to calculate the correlation coefficient where fan-out points and reconvergent gates exists. The accuracy of the method increases significantly as the number of bits generated are increased, however, that increases the computational run time significantly as well.

The ER and Three-point method both try to address the correlation coefficient of the reliabilities by using some local information in the circuit so that the reliabilities correlation can be calculated in linear time. However, the accuracy of these two methods can be improved and thus the proposed model uses the concept from ER model and improves upon it.

1.3 Organization of this Thesis

The thesis is organized as follows

Chapter 2 gives a general idea about signal probability and reliability and how to calculate it using an analytical method. The chapter then goes on to present the limitations of analytical method in the form of signal correlations and how that can lead to producing large errors in the estimation. A proposed method is then presented to deal with the signal correlation issue, hence increasing the accuracy of the estimation while maintaining a linear computation time.

Chapter 3 expands on the idea proposed on chapter 2 and explains in detail on how to obtain certain parameters and mathematically demonstrates how to calculate the correlation coefficient, reliability and probability using some real values on a small benchmark circuit that is ISCAS85 C17 benchmark circuit.

Chapter 4 presents all the simulation data and compares the proposed method with other existing methods. The comparison is done in terms of accuracy as compared to the results obtained via Monte-Carlo Simulation.

Finally, chapter 5 concludes the thesis and discusses some of the limitations this proposed method contains along with how to improve the model in the future.

The appendix section includes some of the important codes for the proposed model.

CHAPTER 2
Signal Reliability Correlation Coefficient

2.1 Reliability Calculation using Signal Probabilities

The reliability for any signal s is defined as the probability that its outcome(s) is what it is meant to be, i.e., $r_s = P\{s = s^*\}$ where s is an output signal and s^* is the correct logic value of the signal. Throughout this paper, the symbol "*" will be used to represent the correct or reliable or "error-free" value. Because the signal is a stream of 1's and 0's, the overall reliability r_s is actually a function of its reliability pair $\{r_s^0, r_s^1\}$ where, r_s^0 (r_s^1) are the conditional probability that the output of the signal is logic 0 (logic 1) given its error-free value is also logic 0 (logic 1).

$$r_s^0 = P\{s = 0 | s^* = 0\} \quad (2\text{-}1)$$

$$r_s^1 = P\{s = 1 | s^* = 1\} \quad (2\text{-}2)$$

The reliability correlation coefficient between two signals a and b is defined as the ratio of their joint reliability and the product of their individual reliability as given below:

$$C_{ab}^{ij} = \frac{r_{ab}^{ij}}{r_a^i r_b^j} \quad (2\text{-}3)$$

where r_{ab}^{ij} is the joint reliability of signal a and b given as:

$$r_{ab}^{ij} = P\{ab = ij | a^* b^* = ij\} \quad (2\text{-}4)$$

where $i, j = 1$ or 0 and r_a^i and r_b^j are the are the conditional probabilities of signals a and b being their error-free value calculated using (2-1) and (2-2):

$$r_a^i = P\{a = i | a^* = i\} \tag{2-5}$$

$$r_b^j = P\{b = j | b^* = j\} \tag{2-6}$$

If the circuit is fully reliable then the signals a and b will be replaced by a^* and b^* respectively. If these two error-free signals are the inputs to a 2-input gate, their error-free joint probability can be expressed as $P_{ij}^* = P\{a^* b^* = ij\}$ for $i, j = 1$ or 0. This error-free joint probability can be used to calculate the error-free joint probability matrix in [7].We then get the following vector:

$$\boldsymbol{P}^* = [P_{00}^* \; P_{01}^* \;\; P_{10}^* \;\; P_{11}^* \;] \tag{2-7}$$

Calculating the $\boldsymbol{P}^*$ requires the correlation of the signal and so throughout this paper this value will be assumed to be available to us. Unfortunately, this might not be the case in reality for large circuits and we may have to run a fast Monte-Carlo to obtain this value. Therefore, this gives us the opportunity to work on developing a model to account for signal correlation in the future.

In reality circuits are almost never 100% reliable. And so, the circuit will instead produce its joint probability for an unreliable circuit denoted as P_{ij} where $i, j = 1$ or 0. The joint probability matrix for an unreliable circuit can be expressed as:

$$\boldsymbol{P} = [P_{00} \; P_{01} \; P_{10} \; P_{11}] \tag{2-8}$$

This joint probability matrix for an unreliable circuit can be obtained by the product of $\boldsymbol{P}^*$ and the joint reliability matrix of the two signals a and b given as $\boldsymbol{R}$ below:

$$\boldsymbol{P} = \boldsymbol{P}^* \cdot \boldsymbol{R} \tag{2-9}$$

where

$$\boldsymbol{R} = [r_{kl}^{ij}] = \begin{bmatrix} r_{00}^{00} & \cdots & r_{00}^{11} \\ \vdots & \ddots & \vdots \\ r_{11}^{00} & \cdots & r_{11}^{11} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R_{00}} \\ \boldsymbol{R_{01}} \\ \boldsymbol{R_{10}} \\ \boldsymbol{R_{11}} \end{bmatrix} \tag{2-10}$$

where the diagonal elements $r_{ij}^{ij} = r_{ab}^{ij}$ are calculated using (2-4) and the non-diagonal elements r_{kl}^{ij} (where k and l can be 0 or 1) can be found using r_{ab}^{ij}, r_a^i and/or r_b^j. The values for r_a^i and r_b^j can be calculated using (2-5) and (2-6) respectively. For instance, $r_{00}^{01} = r_a^0 - r_{ab}^{00}$, $r_{00}^{10} = r_b^0 - r_{ab}^{00}$, and $r_{00}^{11} = 1 - r_a^0 - r_b^0 + r_{ab}^{00}$. Similarly, the matrix can be filled with the rest of the elements as shown in (2-11). If we rearrange the values in equation (2-3), the joint reliability r_{ab}^{ij} can be expressed as $r_{ab}^{ij} = r_a^i \times r_b^j \times C_{ab}^{ij}$. Therefore, the joint reliability can be calculated to the exact value as long as we have the correlation coefficient value.

$$\boldsymbol{R} = \begin{bmatrix} r_{00}^{00} & \cdots & r_{00}^{11} \\ \vdots & \ddots & \vdots \\ r_{11}^{00} & \cdots & r_{11}^{11} \end{bmatrix} = \begin{bmatrix} r_{ab}^{00} & r_a^0 - r_{ab}^{00} & r_b^0 - r_{ab}^{00} & 1 - r_a^0 - r_b^0 + r_{ab}^{00} \\ r_a^0 - r_{ab}^{01} & r_{ab}^{01} & 1 - r_a^0 - r_b^1 + r_{ab}^{01} & r_b^1 - r_{ab}^{01} \\ r_b^0 - r_{ab}^{10} & 1 - r_a^1 - r_b^0 + r_{ab}^{10} & r_{ab}^{10} & r_a^1 - r_{ab}^{10} \\ 1 - r_a^1 - r_b^1 + r_{ab}^{11} & r_b^1 - r_{ab}^{11} & r_a^1 - r_{ab}^{11} & r_{ab}^{11} \end{bmatrix}$$

Once the $\boldsymbol{P}$ and $\boldsymbol{P}^*$ are available, the output reliability and probability for any output signal of any gate with the input signals a and/or b can be calculated using the method mentioned in [7].

In [7] the method for calculating the reliabilities and probabilities for any type of gate is given in details with the joint reliability denoted as **M** instead of **R**. For example, we will be taking NAND Gate with input signals a and b and output signal c to calculate the reliability and probability as shown below:

$$r_c^0 = \boldsymbol{R_0} \cdot (\boldsymbol{1} - \boldsymbol{r_G}) \tag{2-11}$$

$$r_c^1 = \frac{[P_{00}^* \;\; P_{01}^* \;\; P_{10}^*] \cdot (\boldsymbol{R_1} \cdot \boldsymbol{r_G})}{P_c^*} \tag{2-12}$$

where $\boldsymbol{R_1} = \begin{bmatrix} \boldsymbol{R_{00}} \\ \boldsymbol{R_{01}} \\ \boldsymbol{R_{10}} \end{bmatrix}$, $\boldsymbol{R_0} = [\boldsymbol{R_{11}}]$, $\boldsymbol{r_G} = [r_g \;\; r_g \;\; r_g \;\; 1 - r_g]^T$ with r_g being the reliability of the gate and P_c^* is the probability of the output signal c for an error-free condition. Once

the conditional reliability pair $\{r_c^0, r_c^1\}$ has been calculated using (2-11) and (2-12) the overall reliability and probability can be calculated as follows:

$$r_c = r_c^1 * P_c^* + (1 - P_c^*) * r_c^0 \quad (2\text{-}13)$$

$$p_c = r_c^1 * P_c^* + (1 - P_c^*) * (1 - r_c^0) \quad (2\text{-}14)$$

In summary, all signal reliabilities in a given circuit can be found by propagating the reliabilities through all gates in their topological order.

In the following section, we will be focusing on how to estimate the value for correlation coefficient between two signals using the local information in the circuit.

2.2 Correlation Coefficient Estimation

When two signals a and b are independent of each other, then the value of their correlation coefficient, C_{ab}^{ij}, will be set to be 1 as the signals are not affected by one another. In such a case the value of their joint reliability can easily be calculated as $r_{ab}^{ij} = r_a^i \times r_b^j$. Performing this equation is fast and accurate and does not require the use of the proposed model. If the two signals share the same originating fan-out point the signals will be correlated. However, the two signals being correlated does not necessarily suggest that the reliabilities (or the conditional reliability r_a^i and/or r_b^j where $i, j = 1$ or 0) will be correlated. Therefore, the two signals having an originating fan-out point is one of the conditions for the reliabilities being correlated but is not a sufficient one.

Figure 2-1 shows a section of a circuit that demonstrates some of the local information that are available and how to utilize this information in order to calculate an estimation for the reliability correlation coefficient of any two signals a and b.

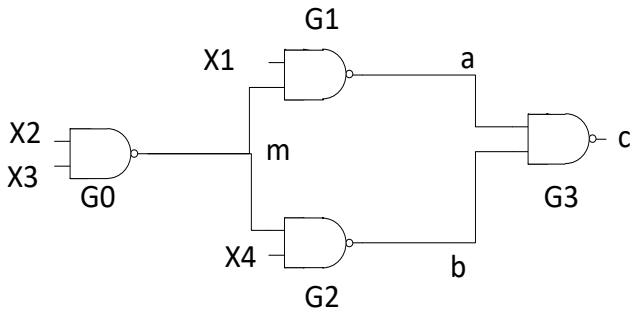

Figure 2- 1 Correlated Section of a Circuit

As mentioned previously, correlation between two signals a and b can only exist if there is a common source of origin, which in the case of Figure 2-1 is represented as the fan-out signal m. Presence of this fan-out point m is a necessary condition for the presence of reliability correlation, however, this may not be enough. The factors that determine whether this condition is enough or not will be discussed briefly in the coming sections.

Generally, the reliability correlation between two signals a and b will increase as the reliability of the fan-out point signal m decreases. This is because as the reliability of the fan-out signal decreases, more error can pass through the driving gates of the signals a and b. An error on one of the signals could appear of the other signal and hence the two signal reliabilities become more correlated.

In extreme conditions where the fan-out signal is assumed to be error-free, i.e. $r_m = 1$ where, r_m is the overall reliability of the fan-out signal m, there will be no errors that passes through the driving gates of the signals a and b. Therefore, the two signals' reliabilities will be independent of each other, even though their probabilities will be correlated. Under such condition the reliability correlation coefficient C_{ab}^{ij} will be 1 and the joint reliability of a and b can be calculated as described previously. On the other side of the spectrum, the maximum correlation will exist if both signals a and b are directly connected to the fan-out point. This will mean that the reliability of the signals does not get influenced by any other factors and so any error present on one of the signals will

definitely be present on the other signal. As a result, the reliability of the signals can be expressed as $r_m = r_a = r_b$, leading to a maximum reliability correlation value of $C_{ab} = {}^{1}/_{r_m}$ according to (2-3). This implies that in general the value of reliability correlation for any two signals can range anywhere from 1 (complete independence) to ${}^{1}/_{r_m}$ (maximum correlation).

In a real circuit, and also demonstrated in Figure 2-1, the fan-out signal passes through a number of gates along two separate paths before reaching a re-convergent gate. On the way of travel, other independent signals can join in and change the reliability and/or probability of the signal, making the two signals a and b less correlated. While it is not exactly known how the gate or other signals affect the reliability correlation between the two signals, however, with careful analysis of different circuit structure the following observations were made: 1) The reliability correlation decreases rapidly as number of unreliable gates on either signal path increases. This suggests that the number of gates (also referred to as levels and denoted as L_1 and L_2) the signal passes through are an important factor when calculating the reliability correlation coefficient. 2) As the signal passes through unreliable gates, the signal reliability decreases and as the reliability of the signal on either path decreases the two signals a and b become less related to the fan-out signal m and thus becomes more independent of each other in terms of the signal reliabilities. This suggests that when both signal reliabilities r_a^i and r_b^j are closer to the signal reliability of the fan-out point, the two signals show a relatively stronger correlation. However, it should be noted that the reliabilities of signal a and b can get affected by other independent signals as they pass through different gates and so weakening the correlation between the two signals. Therefore, when considering the relative value of signal reliability with respect to the reliability of fan-out point m, it would be more meaningful to take only the effective signal reliability with respect to r_m. This effective reliability only contains the portion of r_a^i or r_b^j, and will be explained later in this thesis.

After taking the above observations into consideration, the proposed model for calculating the correlation coefficient between two signals a and b can be expressed as:

$$C_{ab}^{ij} = 1 + \left(\frac{1}{r_m} - 1\right) e^{\alpha\left[L_1\left(1-\frac{r_{ea}^i}{r_m}\right)+L_2\left(1-\frac{r_{eb}^j}{r_m}\right)\right]} \tag{2-15}$$

where α is a constant determined by running simulations and r_{ea}^i and r_{eb}^j are the effective reliabilities of the signals a and b respectively with respect to the fan-out signal m.

Further analysis of the circuit in Figure 2-1 shows that the condition for correlation between these two signals are not limited to the mentioned parameters in (2-15). The reliability of the driving gate of signals a and b also play a major role in determining the correlation coefficient. More specifically, if the conditional reliabilities r_a^i and r_b^j are closer to the reliability of the driving gate of the signals a and b, then that implies that the reliability of the signals are controlled more by the gate rather than the inputs of the gates. Hence, the signals are less correlated of each other and so (2-15) can be further modified as:

$$C_{ab}^{ij} = 1 + \left[\frac{1}{r_m} - 1\right] e^{-x} \tag{2-16}$$

where

$$x = \alpha(1 - r_m)\left[\frac{\left(1-\frac{r_{ea}^i}{r_m}\right)L_1}{\left(1-\frac{r_a^i}{r_g}\right)} + \frac{\left(1-\frac{r_{eb}^j}{r_m}\right)L_2}{\left(1-\frac{r_b^j}{r_g}\right)}\right] \tag{2-17}$$

where r_g is the reliability of the driving gate of the signals a and b. The values for r_m, r_g, r_a^i and r_b^j can be taken directly from a specific propagation step. Because the circuits are arranged topologically, all the information will already be available to be used. L_1 and L_2 represent the number of gates the signal passes from the re-convergent point m to signals a and b, respectively, and can be found from the topologic structure of the circuit. α is a

constant, which is found to be around 1.226 for when the value of $r_m > 0.8$ and 0.55 for when the value of $r_m \leq 0.7$ and it changes linearly for when the value of r_m lies anywhere between 0.8 and 0.7, based on our simulations.

The term $(1 - r_m)$ in (2-17) is a weighting factor which is added to adjust the weight of the exponential function in (2-16), making α nearly a constant for various circuit structures.

As mentioned previously, the existence of a fan-out point is a necessary condition for the two signals a and b but not an absolute one. Under certain circumstances the signals may be considered to have independent reliabilities even though they may have correlated signal probabilities. After careful observations and backed by simulation data, we can treat the signals a and b to be independent if:

1) No reconvergent gate is found
2) The reconvergent gate is physically too far away from the fan-out point, i.e., either one of the signal passes through a large number of gates before reaching the reconvergent gate which is considered to be more than 5.
3) If the fan-out signal is an error-free signal.

For better understanding the Table1 below shows the topological signal correlation information as well as the value of L_1 and L_2 for C17 ISCAS85 benchmark circuit shown in Figure 2-2.

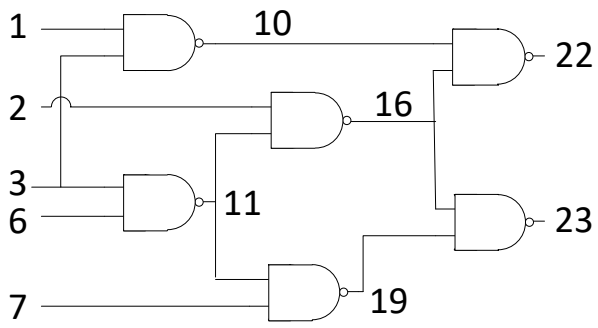

Figure 2- 2 C17 Benchmark Circuit

Table 1 Topological information about C17 benchmark circuit

Gate output signal #	Input Signals Correlation	L_1	L_2
10	N	-	-
11	N	-	-
16	N	-	-
19	N	-	-
22	Y	1	2
23	Y	1	1

2.3 Effective Reliability

In the previous section, we saw that when calculating the reliability correlation, we are interested in a parameter called the effective reliability. This is because as the signal passes through different gates from its fan-out point, the other independent signals may cause the reliability and probability of the signal to change and so the signals become less correlated of each other. Taking the effective reliability of the signal ensures that when calculating the reliability correlation, the independent signals do not invalidate the results.

To explain the effective reliability let us consider a correlated section of a circuit as shown in Figure 2-3 where the signal a is driven by the NAND gate G_1 which has two inputs, x_1 and m. Assume that the joint reliability matrix for the signals x_1 and m with this gate G_1 is given as:

$$\boldsymbol{R_{G1}} = \begin{bmatrix} \boldsymbol{R_{G1}^{00}} \\ \boldsymbol{R_{G1}^{01}} \\ \boldsymbol{R_{G1}^{10}} \\ \boldsymbol{R_{G1}^{11}} \end{bmatrix} \tag{2-18}$$

where $\boldsymbol{R_{G1}^{00}}$, $\boldsymbol{R_{G1}^{00}}$, $\boldsymbol{R_{G1}^{00}}$ and $\boldsymbol{R_{G1}^{00}}$ can be calculated as in (2-10) whether the signals are correlated or not. Since the effective reliability is calculated only for the signals that are in the reconvergent signal path, the joint reliability for x_1 and m should not include any cases where the output signal a can be controlled by x_1. Since the G_1 is a NAND gate, the output signal a can be controlled by x_1 when the joint probability of the inputs is $P_{x_1m}^{01}$. Thus, the effective joint probability matrix of the error-free inputs m^* and x_1^* can be expressed as:

$$P_{eff}^{*} = [P_{x_1m}^{00} \;\; 0 \;\; P_{x_1m}^{10} \;\; P_{x_1m}^{11}] \tag{2-19}$$

The effective reliability pair $\{r_{ea}^0, r_{ea}^1\}$ in (2-17) can now be calculated using similar equations of (2-11) and (2-12) as

$$r_{ea}^0 = \boldsymbol{R_{G1}^0} \cdot (\mathbf{1} - \boldsymbol{r_{G1}}) \tag{2-20}$$

$$r_{ea}^1 = \frac{[P_{x_1m}^{00} \;\; 0 \;\; P_{x_1m}^{10}] \cdot (\boldsymbol{R_{G1}^1} \cdot \boldsymbol{r_{G1}})}{P_a^*} \tag{2-21}$$

where $\boldsymbol{R_{G1}^0} = \begin{bmatrix} \boldsymbol{R_{G1}^{00}} \\ \boldsymbol{R_{G1}^{01}} \\ \boldsymbol{R_{G1}^{10}} \end{bmatrix}$ and $\boldsymbol{R_{G1}^1} = [\boldsymbol{R_{G1}^{11}}]$.

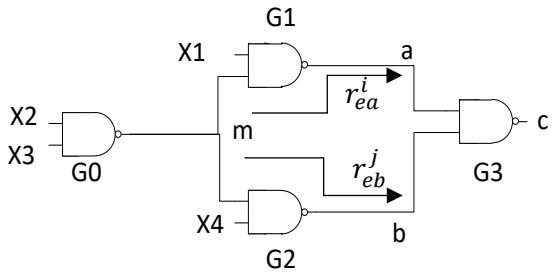

Figure 2- 3 Propagation of effective reliability

Similarly the effective reliability of signal b in Figure 2-3 can be calculated so that they can be used properly in (2-17) to calculate the reliability correlation coefficient of the signals a and b.

It is important to note that the effective reliability pair will only be calculated for the signal that is a part of the reconvergent path. If the signal passes through more than one gate before reaching the input of the reconvergent gate, as shown in Figure 2-4, the effective reliability pair needs to be propagated.

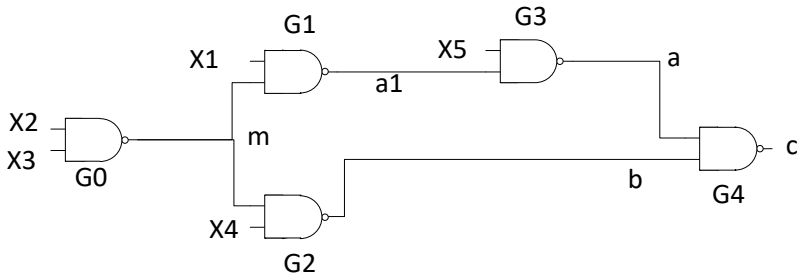

Figure 2- 4 Propagation of effective reliabilities over gates

For the section of the circuit in Figure 2-4 the effective reliability pair $\{r_{ea_1}^0, r_{ea_1}^1\}$ for the signal a_1 needs to calculated first before it is propagated to the next gate G_3 to calculate the effective reliability pair of signal a. The input x_5 in gate G_3 is not a part of the reconvergent path, hence the regular reliability pair $\{r_{x_5}^0, r_{x_5}^1\}$ will be used instead of its equivalent reliability pair when calculating the equivalent reliability pair for the signal a.

The equations used to calculate the joint reliability, effective reliability pair and the regular reliability pair of the signals varies with the type of gate the signals pass though. The exact equations and methods are described in detail on [7].

2.4 Algorithm

Throughout the thesis, in order to test and prove the validity of the model we implemented the algorithm on MATLAB. The entire process from reading the ISCAS85 Benchmark circuits into the MATLAB till we obtain the reliability and probability of every signal in the circuit is summarized in the pseudo code below:

Algorithm Description:

Step 1: Read given circuit;

Step 2: Sort and number all gates in topologic order;

Step 3: FOR each gate *k*, DO:

IF the gate inputs *a* and *b* are independent, THEN

set C_{ab}^{ij}=1

ELSE find reconvergent point, L_1, L_2 and effective reliabilities for *a* and *b*, and calculate C_{ab}^{ij} using the proposed model (2-13);

Calculate the reliability pair for the gate output;

END FOR

Step 4: Report the reliabilities for circuit outputs.

CHAPTER 3
In-Depth Analysis of the Proposed Model

3.1 Overview

In the previous chapter, a proposed model to calculate the reliability correlation coefficient between two signals was introduced. This proposed model uses the local information that are already made available within the section of the circuit in order to calculate a nearly accurate estimation. In this chapter we will be taking a closer look on how to obtain all the required parameters and then mathematically prove its validity using small ISCAS85 benchmark circuits such as C17 and C18.

3.2 Tabular representation of circuit

The main application of the reliability correlation coefficient is to calculate the joint reliability matrix of two input signals a and b of a 2-input logic gate. Unfortunately, this joint reliability matrix cannot be calculated directly for a gate that consists of more than two inputs. Such a multiple-input gate (three or more input) must be decomposed into a tree of 2-input gates followed by a buffer or an inverter depending on the nature of the gate. For example, a 4-input NOR gate as shown in Figure 3-1 a) should be decomposed into a tree of 2-input gates as shown in Figure 3-1 b).The 2-input OR gates OR_1 to OR_3 are considered to be fault-free ideal gates while the inverter at the end, NOT_1 contains all the error of the gate, i.e. the reliability of the buffer will be equal to the reliability of the pre-decomposed gate. In doing so, it guarantees that the overall reliability of the gate does not change while maintaining the operation of the gate, hence properly representing the erroneous multiple-input gate into a tree of 2-input gates followed by a 1-input gate.

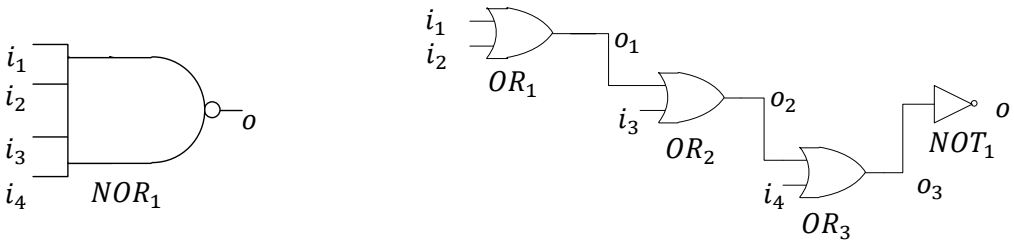

Figure 3- 1 a) 4-input NOR Gate b) Decomposed 4-input NOR gate into tree of 2-input OR gates

Once the multiple-input gates have been properly decomposed, the calculation for reliability and probability at the gate output signal can be carried out as normal and as discussed in the previous chapter.

Once all the gates are decomposed properly, the circuit is represented topologically in a tabular data structure. Table 2 shows the tabular representation of a C17 benchmark circuit. Here, since none of the gates had to be decomposed the total number of gates in the circuit did not increase. The gate reliability is represented as r_g. For the rest of the paper r_g is assumed to be same for every gate. Gate type is represented using a numerical value from 1 to 8 where each value represents a specific type of logic gate (e.g., NAND gate has gate type of 1. Readers are referred to [7] for details).

Table 2 Tabular Representation of C17 Benchmark Circuit

Signal #	Input-1	Input-2	Gate Reliability	Gate type
10	1	3	r_g	1
11	3	6	r_g	1
16	2	11	r_g	1
19	11	7	r_g	1
22	10	16	r_g	1
23	16	19	r_g	1

3.3 Mathematical Analysis of C17 benchmark circuit

Figure 3-2 shows the schematic diagram of C17 benchmark circuit. in this example, we will assume the primary inputs of signal $1, 2, 3, 6$ and 7 are all error-free signals ($r_s = 1$ and $\{r_s^0, r_s^1\} = \{1,1\}$, where $s = 1,2,3,6,7$) and that their probabilities are set at 0.5 ($p_s = 0.5$). All gates in this example are assumed to be unreliable and the reliability of the gates are set at $r_g = 0.95$.

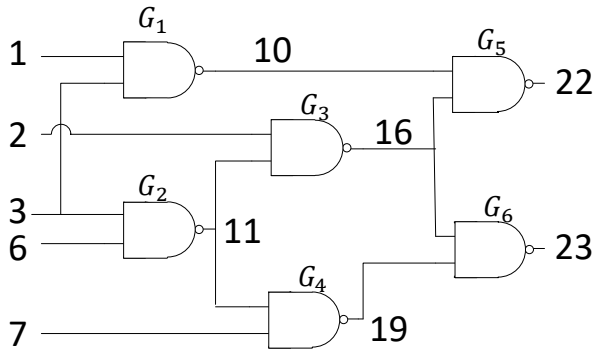

Figure 3- 2 C17 Benchmark Circuit

Signal reliability and probability needs to be calculated topologically and propagated to the next gates until we reach the primary output gate. Also, we know that the input signals of a logic gate are considered to independent unless they have the following conditions:

1. There is no fan-out point between the two input signals
2. If there is a fan-out point but the number of Level for either of the signal is more than 5

If these conditions are not met, then the reliability correlation must be calculated before moving on to calculate the reliability and probability of the output signals.

Considering gate G_1

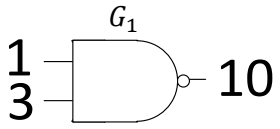

Figure 3- 3 Isolated gate G_1 from C17

It is assumed that the error-free probability matrix of the circuit is already provided to us and so we can simply write the value of the matrix $\boldsymbol{P}^*$ as:

$$\boldsymbol{P}^* = [0.25\ 0.25\ 0.25\ 0.25] \tag{3-1}$$

with $\boldsymbol{P_0^*} = [0.25\ 0.25\ 0.25]$ and $\boldsymbol{P_1^*} = [0.25]$

According to equation (2-9) the probability matrix of an unreliable circuit can be expressed:

$$\boldsymbol{P} = \boldsymbol{P}^* \times \boldsymbol{R} \tag{3-2}$$

where $\boldsymbol{R}$ is the joint reliability matrix of the input signals 1 and 3. Since, signals 1 and 3 do not satisfy any of the condition to be correlated, these two signals are considered to be independent of each other and hence the correlation coefficient for these two signals can be given as:

$$\boldsymbol{C_{1,3}} = \left[C_{1,3}^{00}\ C_{1,3}^{01}\ C_{1,3}^{10}\ C_{1,3}^{11}\right] = [1\ 1\ 1\ 1]$$

With the correlation coefficient, $\boldsymbol{C_{1,3}}$ available the joint reliability matrix for signal 1 and 3 can be calculated using equation (2-10) as:

$$\boldsymbol{R} = \begin{bmatrix} r_{1,3}^{00} & r_1^0 - r_{1,3}^{00} & r_3^0 - r_{1,3}^{00} & 1 - r_1^0 - r_3^0 + r_{1,3}^{00} \\ r_1^0 - r_{1,3}^{01} & r_{1,3}^{01} & 1 - r_1^0 - r_3^1 + r_{ab}^{01} & r_3^1 - r_{1,3}^{01} \\ r_3^0 - r_{1,3}^{10} & 1 - r_1^1 - r_3^0 + r_{ab}^{10} & r_{1,3}^{10} & r_1^1 - r_{1,3}^{10} \\ 1 - r_1^1 - r_3^1 + r_{1,3}^{11} & r_3^1 - r_{1,3}^{11} & r_1^1 - r_{1,3}^{11} & r_{1,3}^{11} \end{bmatrix} \tag{3-3}$$

$$\boldsymbol{R} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3-4}$$

According to equation (2-10) $\boldsymbol{R}$ can be split into two sub-matrices as:

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R_{00}} \\ \boldsymbol{R_{01}} \\ \boldsymbol{R_{10}} \\ \boldsymbol{R_{11}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R_0} \\ \boldsymbol{R_1} \end{bmatrix} \tag{3-5}$$

where $\boldsymbol{R_0} = [\boldsymbol{R_{00}}\ \boldsymbol{R_{01}}\ \boldsymbol{R_{10}}]^T$ and $\boldsymbol{R_1} = [\boldsymbol{R_{11}}]$

The output reliability vector of the gate, $\boldsymbol{r_g}$ is a 4×1 matrix where each element is a conditional probability for the output to be a certain value (1 or 0) given the joint input probability being ij where $i, j = 0$ or 1. The $\boldsymbol{r_g}$ for any logic gate is expressed as:

$$\boldsymbol{r_g} = \left[r_g^{00}\ r_g^{01}\ r_g^{10}\ r_g^{11}\right]^T \tag{3-6}$$

where $r_g^{ij} = P\{c = k_0 | ab = ij\}$ and $k_0, i, j = 0$ or 1. Here, c is the output signal of the logic gate while a and b are its inputs.

For a NAND gate, the output reliability vector is given as:

$$\boldsymbol{r_g} = [r_g\ r_g\ r_g\ 1 - r_g]^T \tag{3-7}$$

where r_g is the reliability of the gate. This output reliability vector will be different depending on the type of the gate.

For gate G_1 the output reliability vector will be given as:

$$\boldsymbol{r_g} = \begin{bmatrix} 0.95 \\ 0.95 \\ 0.95 \\ 0.05 \end{bmatrix} \tag{3-8}$$

Using equation (2-11) and (2-12) the reliability pair of the output signal of gate G_1 is calculated as:

$$r_{10}^0 = \boldsymbol{R_0} \cdot (1 - \boldsymbol{r_g}) \tag{3-9}$$

$$r_{10}^0 = [0\ 0\ 0\ 1] \cdot \left(1 - \begin{bmatrix} 0.95 \\ 0.95 \\ 0.95 \\ 0.05 \end{bmatrix} \right) \tag{3-10}$$

$$r_{10}^0 = 0 + 0 + 0 + 0.95 = 0.95 \tag{3-11}$$

And

$$r_{10}^1 = \frac{\boldsymbol{P_1^* \cdot R_1 \cdot r_g}}{P_{10}^*} \tag{3-12}$$

where P_c^* is the error-free output probability of signal 10 which is assumed to be available and $P_c^* = 0.75$.

$$r_{10}^1 = \frac{[0.25\ 0.25\ 0.25] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0.95 \\ 0.95 \\ 0.95 \\ 0.05 \end{bmatrix}}{0.75} \tag{3-13}$$

$$r_{10}^1 = \frac{0.25 \times 0.95 + 0.25 \times 0.95 + 0.25 \times 0.95}{0.75} = \frac{0.7125}{0.75} = 0.95 \tag{3-14}$$

Therefore, the output reliability pair for signal 10 is calculated to be $\{r_{10}^0, r_{10}^1\} = \{0.95, 0.95\}$

Now to calculate the overall probability and reliability of the signal 10 we will be using equation (2-13) and (2-14) as below:

The overall reliability will now be calculated,

$$r_{10} = (r_{10}^{1} * P_{10}^{*}) + ((1 - P_{10}^{*}) * r_{10}^{0}) \quad (3\text{-}15)$$

$$r_{10} = (0.95 * 0.75) + \big((1 - 0.75) * 0.95\big) = 0.7125 + 0.2375 = 0.95 \quad (3\text{-}16)$$

And the probability is calculated as,

$$p_{10} = (r_{10}^{1} * P_{10}^{*}) + \Big((1 - P_{10}^{*}) * (1 - r_{10}^{0})\Big) \quad (3\text{-}17)$$

$$p_{10} = (0.95 * 0.75) + \big((1 - 0.75) * (1 - 0.95)\big) = 0.7125 + 0.0125 = 0.725 \quad (3\text{-}18)$$

Finally, the parameters that are calculated from gate G_1 and will be made available to be used later are:

- $r_{10}^{0} = 0.9500$
- $r_{10}^{1} = 0.9500$
- $p_{10} = 0.7250$
- $r_{10} = 0.9500$

Once all these values are calculated we move on to the next gates topologically to calculate their corresponding values propagating the ones that are calculated from the previous stages.

Since signal 11 has identical input and output conditions, the mathematical calculation for reliability and probability will also be the same as signal 10 and so the values will also be the same.

Considering gate G_3

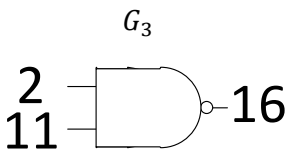

Figure 3- 4 Isolated gate G_3 from C17

Now considering the gate G_3, we have to perform the same equations to obtain its reliability and probability value. The provided error-free probability matrix for gate G_3 is given as:

$$\boldsymbol{P}^* = [0.1375\ 0.3621\ 0.1371\ 0.3634] \tag{3-19}$$

with $\boldsymbol{P_0^*} = [0.1375\ 0.3621\ 0.1371]$ and $\boldsymbol{P_1^*} = [0.3634]$

Since signals 2 and 11 do not satisfy any of the conditions to be correlated, these two signals are considered to be independent of each other and hence the correlation coefficient for these two signals can be given as:

$$\boldsymbol{C_{2,11}} = \left[C_{2,11}^{00}\ C_{2,11}^{01}\ C_{2,11}^{10}\ C_{2,11}^{11}\right] = [1\ 1\ 1\ 1] \tag{3-20}$$

With the correlation coefficient, $\boldsymbol{C_{2,11}}$ available the joint reliability matrix for signal 1 and 3 can be calculated using equation (2-10) that is similar to (3-) as:

$$\boldsymbol{R} = \begin{bmatrix} 0.95 & 0.05 & 0 & 0 \\ 0.05 & 0.95 & 0 & 0 \\ 0 & 0 & 0.95 & 0.05 \\ 0 & 0 & 0.05 & 0.95 \end{bmatrix} \tag{3-21}$$

According to equation (2-10) $\boldsymbol{R}$ can be split into two sub-matrices as:

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R_{00}} \\ \boldsymbol{R_{01}} \\ \boldsymbol{R_{10}} \\ \boldsymbol{R_{11}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R_0} \\ \boldsymbol{R_1} \end{bmatrix} \tag{3-22}$$

where $\boldsymbol{R_0} = [\boldsymbol{R_{00}}\ \boldsymbol{R_{01}}\ \boldsymbol{R_{10}}]^T$ and $\boldsymbol{R_1} = [\boldsymbol{R_{11}}]$

Using equation (2-11) and (2-12) on gate G_3 to calculate the reliability pair as:

$$r_{16}^0 = M_0 \cdot (1 - R) \tag{3-23}$$

$$r_{16}^0 = [0\ 0\ 0.05\ 0.95] \cdot (1 - [0.95\ 0.95\ 0.95\ 0.05]^T) \tag{3-24}$$

$$r_{16}^{0} = [0\ 0\ 0.05\ 0.95] \cdot \begin{bmatrix} 0.05 \\ 0.05 \\ 0.05 \\ 0.95 \end{bmatrix} \tag{3-25}$$

$$r_{16}^{0} = 0 + 0 + 0.0025 + 0.9025 = 0.905 \tag{3-26}$$

and

$$r_{16}^{1} = \frac{[0.1374\ 0.3621\ 0.1371] \cdot \begin{bmatrix} 0.95 & 0.05 & 0 & 0 \\ 0.05 & 0.95 & 0 & 0 \\ 0 & 0 & 0.95 & 0.05 \end{bmatrix} \cdot [0.95\ 0.95\ 0.95\ 0.05]^{T}}{0.6367} \tag{3-27}$$

$$r_{16}^{1} = \frac{[0.1486\ 0.3509\ 0.1302\ 0.0069] \cdot [0.95\ 0.95\ 0.95\ 0.05]^{T}}{0.6367} \tag{3-28}$$

$$r_{16}^{1} = \frac{0.1486 \times 0.95 + 0.3509 \times 0.95 + 0.1302 \times 0.95 + 0.0069 \times 0.05}{0.6367} = \frac{0.5986}{0.6367} = 0.9402 \tag{3-29}$$

Therefore, the output reliability pair for signal 10 is calculated to be $\{r_{16}^{0}, r_{16}^{1}\} = \{0.905, 0.9402\}$

Now to calculate the overall probability and reliability of the signal 10 we will be using equation (2-13) and (2-14) as below:

The overall reliability will now be calculated,

$$r_{16} = (r_{16}^{1} * P_{16}^{*}) + ((1 - P_{16}^{*}) * r_{16}^{0}) \tag{3-30}$$

$$r_{16} = (0.9402 * 0.6255) + \big((1 - 0.625) * 0.905\big) = 0.5881 + 0.3389 = 0.9270 \tag{3-31}$$

And the probability is calculated as,

$$p_{16} = (r_{16}^{1} * P_{16}^{*}) + \Big((1 - P_{16}^{*}) * (1 - r_{16}^{0})\Big) \tag{3-32}$$

$$p_{16} = (0.9402 * 0.6255) + \big((1 - 0.6255) * (1 - 0.905)\big) = 0.5881 + 0.0356 = 0.6237 \tag{3-33}$$

Finally, the parameters that are calculated from gate G_3 and will be made available to be used later are:

- $r_{16}^0 = 0.9050$
- $r_{16}^1 = 0.9402$
- $p_{16} = 0.6237$
- $r_{16} = 0.9270$

The input and output conditions for gate G_4 is the same as for gate G_3 and so the output reliability and probability values will also be the same.

Considering gate G_5

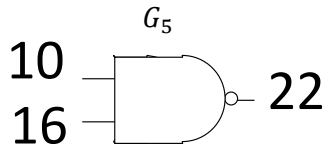

Figure 3- 5 Isolated gate G_5 from C17

The inputs of this gates have an originating fan-out point. Therefore, it does satisfy one of the conditions for the two input signals to be correlated of each other. However, the fan-out point signal is a primary input signal and so the reliability of this fan-out point signal is 1. As a result, even though the signals are correlated with respect to their probabilities, the signals are independent with respect to their reliabilities. Thus, the correlation coefficient can be expressed as:

$$\boldsymbol{C_{10,16}} = \left[C_{10,16}^{00}\ C_{10.16}^{01}\ C_{10,16}^{10}\ C_{10,16}^{11}\right] = [1\ 1\ 1\ 1]$$

The error-free input joint probability matrix is available and is given as:

$$\boldsymbol{P}^{*} = [0.1027\ 0.1722\ 0.2705\ 0.4546]$$

with $\boldsymbol{P_0^*} = [0.1027\ 0.1722\ 0.2705\ ]$ and $\boldsymbol{P_1^*} = [0.4546]$

With the correlation coefficient, $\boldsymbol{C_{10,16}}$ available the joint reliability matrix for signal 10 and 16 can be calculated using equation (2-10) as:

$$\boldsymbol{R} = \begin{bmatrix} 0.8598 & 0.0902 & 0.0453 & 0.0048 \\ 0.0567 & 0.8933 & 0.0030 & 0.0470 \\ 0.0453 & 0.0048 & 0.8598 & 0.0902 \\ 0.0030 & 0.0470 & 0.0567 & 0.8933 \end{bmatrix}$$

According to equation (2-10) $\boldsymbol{R}$ can be split into two sub-matrices as:

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R_{00}} \\ \boldsymbol{R_{01}} \\ \boldsymbol{R_{10}} \\ \boldsymbol{R_{11}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R_0} \\ \boldsymbol{R_1} \end{bmatrix}$$

where $\boldsymbol{R_0} = [\boldsymbol{R_{00}}\ \boldsymbol{R_{01}}\ \boldsymbol{R_{10}}]^{T}$ and $\boldsymbol{R_1} = [\boldsymbol{R_{11}}]$

Using equation (2-11) and (2-12) on gate G_3 to calculate the reliability pair as:

$$r_{16}^{0} = M_0 \cdot (1 - R)$$

$$r_{22}^{0} = [0.0030\ 0.0470\ 0.0567\ 0.8933] \cdot (1 - [0.95\ 0.95\ 0.95\ 0.05]^{T})$$

$$r_{22}^{0} = [0.0030\ 0.0470\ 0.0567\ 0.8933] \cdot \begin{bmatrix} 0.05 \\ 0.05 \\ 0.05 \\ 0.95 \end{bmatrix}$$

$$r_{22}^{0} = 0.00015 + 0.00235 + 0.002835 + 0.8486 = 0.8539$$

and

$$r_{22}^{1} = \frac{[0.1027\ 0.1722\ 0.2705\ ] \cdot \begin{bmatrix} 0.8598 & 0.0902 & 0.0453 & 0.0048 \\ 0.0567 & 0.8933 & 0.0030 & 0.0470 \\ 0.0453 & 0.0048 & 0.8598 & 0.0902 \end{bmatrix} \cdot [0.95\ 0.95\ 0.95\ 0.05]^{T}}{0.4546}$$

$$r_{22}^{1} = 0.8956$$

Therefore, the output reliability pair for signal 10 is calculated to be $\{r_{22}^{0}, r_{22}^{1}\} = \{0.8539,0.8956\}$

Now to calculate the overall probability and reliability of the signal 10 we will be using equation (2-13) and (2-14) as below:

The overall reliability will now be calculated,

$$r_{22} = (r_{22}^{1} * P_{22}^{*}) + ((1 - P_{22}^{*}) * r_{22}^{0})$$

$$r_{22} = (0.8956 * 0.5600) + \big((1 - 0.5600) * 0.8539\big) = 0.5015 + 0.3757 = 0.8772$$

And the probability is calculated as,

$$p_{22} = (r_{22}^{1} * P_{22}^{*}) + \Big((1 - P_{22}^{*}) * (1 - r_{22}^{0})\Big)$$

$$p_{22} = (0.8956 * 0.5600) + \big((1 - 0.5600) * (1 - 0.8539)\big) = 0.5015 + 0.0643 = 0.5658$$

Finally, the parameters that are calculated from gate G_5 and will be made available to be used later are:

- $r_{22}^{0} = 0.8539$
- $r_{22}^{1} = 0.8956$
- $p_{22} = 0.5658$
- $r_{22} = 0.8772$

Considering gate G_6

The error-free input joint probability matrix is available and is given as:

$$\boldsymbol{P}^* = [0.1390\ 0.2327\ 0.2337\ 0.3946]$$

with $\boldsymbol{P_0^*} = [0.1390\ 0.2327\ 0.233\ ]$ and $\boldsymbol{P_1^*} = [0.3946]$

The inputs in gate G_6 satisfies all the conditions to have reliability correlation between them. Due to this the reliability correlation coefficient needs to be calculated using our proposed method. The proposed model is given as:

$$C_{16,19}^{ij} = 1 + \left[\frac{1}{r_{11}} - 1\right] e^{-\alpha(1-r_{11})\left[\frac{\left(1-\frac{r_{e16}^{i}}{r_{11}}\right)L_1}{\left(1-\frac{r_{16}^{i}}{r_g}\right)} + \frac{\left(1-\frac{r_{e19}^{j}}{r_{11}}\right)L_2}{\left(1-\frac{r_{19}^{j}}{r_g}\right)}\right]}$$

The two input signals originate at a common point at signal 11. Therefore, the reliability of the fan-out point will be $r_m = 0.95$. The signal then passes through one gate on each path before reaching the reconvergent gate. Therefore, the level for the two correlated signals are $L_1 = 1$ and $L_2 = 1$. Finally, the effective reliability needs to be calculated for the two signals reaching the reconvergent gate according to equations (2-18) to (2-21).

The effective reliability for the signals 16 and 19 can be calculated as:

$$P_{eff,16}^* = [P_{2,11}^{00}\ \ 0\ \ P_{2,11}^{10}\ \ P_{2,11}^{11}] \tag{3-34}$$

The effective reliability pair $\{r_{e16}^0, r_{e16}^1\}$ in (2-17) can now be calculated using similar equations of (2-11) and (2-12) as:

$$r_{e16}^0 = \boldsymbol{R_{G3}^0} \cdot (\mathbf{1} - \boldsymbol{r_{G3}}) \tag{3-35}$$

$$r_{ea}^1 = \frac{[P_{2,11}^{00}\ \ 0\ \ P_{2,11}^{10}] \cdot (\boldsymbol{R_{G3}^1} \cdot \boldsymbol{r_{G3}})}{P_{16}^*} \tag{3-36}$$

The effective reliability pair is then calculated and is found to be $\{r_{e16}^0, r_{e16}^1\} = \{0.9050, 0.3997\}$

Since the signal passes through identical paths on both sides the effective reliability pair for signal 19 is also $\{r_{e19}^0, r_{e19}^1\} = \{0.9050, 0.3997\}$

Now the reliability correlation coefficient between signals 16 and 19 can be calculated using the proposed model as:

$$C_{16,19}^{00} = 1 + \left[\frac{1}{r_{11}} - 1\right] e^{-\alpha(1-r_{11})\left[\frac{\left(1-\frac{r_{e16}^0}{r_{11}}\right)L_1}{\left(1-\frac{r_{16}^0}{r_g}\right)} + \frac{\left(1-\frac{r_{e19}^0}{r_{11}}\right)L_2}{\left(1-\frac{r_{19}^0}{r_g}\right)}\right]} \quad (3\text{-}37)$$

$$C_{16,19}^{00} = 1 + \left[\frac{1}{0.95} - 1\right] e^{-1.226(1-0.95)\left[\frac{\left(1-\frac{0.9050}{0.95}\right)}{\left(1-\frac{0.8539}{0.95}\right)} + \frac{\left(1-\frac{0.9050}{0.95}\right)}{\left(1-\frac{0.8539}{0.95}\right)}\right]} \quad (3\text{-}38)$$

$$C_{16,19}^{00} = 1.0466 \quad (3\text{-}39)$$

Similarly, the rest of the correlation coefficient values are calculated and represented as:

$$\boldsymbol{C_{16,19}^{ij}} = [1.0466\ 1.0016\ 1.0015\ 1.0001] \quad (3\text{-}40)$$

With the correlation coefficient, $\boldsymbol{C_{16,19}}$ available the joint reliability matrix for signal 16 and 19 can be calculated using equation (2-10) as:

$$\boldsymbol{R} = \begin{bmatrix} 0.8572 & 0.0478 & 0.0478 & 0.0472 \\ 0.0527 & 0.8523 & 0.0070 & 0.0880 \\ 0.0528 & 0.0070 & 0.8522 & 0.0880 \\ 0.0036 & 0.0561 & 0.0561 & 0.8842 \end{bmatrix}$$

According to equation (2-10) $\boldsymbol{R}$ can be split into two sub-matrices as:

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R_{00}} \\ \boldsymbol{R_{01}} \\ \boldsymbol{R_{10}} \\ \boldsymbol{R_{11}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R_0} \\ \boldsymbol{R_1} \end{bmatrix}$$

where $\boldsymbol{R_0} = [\boldsymbol{R_{00}}\ \boldsymbol{R_{01}}\ \boldsymbol{R_{10}}]^T$ and $\boldsymbol{R_1} = [\boldsymbol{R_{11}}]$

Using equation (2-11) and (2-12) on gate G_3 to calculate the reliability pair as:

$$r_{23}^{0} = M_0 \cdot (1 - R)$$

$$r_{23}^{0} = [0.0036 \quad 0.0561 \quad 0.0561\ 0.8842] \cdot (1 - [0.95\ 0.95\ 0.95\ 0.05]^T)$$

$$r_{23}^{0} = [0.0036 \quad 0.0561 \quad 0.0561\ 0.8842] \cdot \begin{bmatrix} 0.05 \\ 0.05 \\ 0.05 \\ 0.95 \end{bmatrix}$$

$$r_{23}^{0} = 0.8458$$

and

$$r_{23}^{1} = \frac{[0.1390\ 0.2327\ 0.233\ ] \cdot \begin{bmatrix} 0.8572 & 0.0478 & 0.0478 & 0.0472 \\ 0.0527 & 0.8523 & 0.0070 & 0.0880 \\ 0.0528 & 0.0070 & 0.8522 & 0.0880 \end{bmatrix} \cdot [0.95\ 0.95\ 0.95\ 0.05]^T}{0.3946}$$

$$r_{23}^{1} = 0.8792$$

Therefore, the output reliability pair for signal 10 is calculated to be $\{r_{23}^{0}, r_{23}^{1}\} = \{0.8458,0.8792\}$

Now to calculate the overall probability and reliability of the signal 10 we will be using equation (2-13) and (2-14) as below:

The overall reliability will now be calculated,

$$r_{23} = (r_{23}^{1} * P_{23}^{*}) + ((1 - P_{23}^{*}) * r_{23}^{0})$$

$$r_{23} = (0.8792 * 0.5600) + \big((1 - 0.5600) * 0.8458\big) = 0.4924 + 0.3722 = 0.8645$$

And the probability is calculated as,

$$p_{23} = (r_{23}^{1} * P_{23}^{*}) + \Big((1 - P_{23}^{*}) * (1 - r_{23}^{0})\Big)$$

$$p_{23} = (0.8792 * 0.5600) + \big((1 - 0.5600) * (1 - 0.8458)\big) = 0.4924 + 0.0678 = 0.5602$$

Finally, the parameters that are calculated from gate G_5 and will be made available to be used later are:

- $r_{23}^0 = 0.8458$
- $r_{23}^1 = 0.8792$
- $p_{23} = 0.5602$
- $r_{23} = 0.8645$

The results calculated using the model is compared with Monte-Carlo Simulation to prove its validity and is shown in the Table below

Table 3 Comparison of reliability and probability between Monte-Carlo and Proposed model using C17 with r_g=0.95

Signal #	Monte Carlo		Proposed	
	Reliability	Probability	Reliability	Probability
10	0.9504	0.7249	0.9500	0.7250
11	0.9498	0.7249	0.9500	0.7250
16	0.9273	0.6235	0.9270	0.6237
19	0.9278	0.6236	0.9270	0.6237
22	0.8755	0.5638	0.8772	0.5658
23	0.8659	0.5634	0.8645	0.5602
Run time	70.2 s		0.0779s	

CHAPTER 4

Simulation Results

To verify the accuracy and time efficiency of the proposed model, a number of simulations were performed on various ISCAS85 Benchmark circuits all with different number of gates, inputs and outputs. The simulations were performed in MATLAB using a computer with 2.90GHz processor and 12GB RAM. Throughout the thesis, the results obtained by using Monte-Carlo Simulations are assumed to be accurate.

Table 4 shows the result of ISCAS85 Benchmark circuits to verify the accuracy of the proposed model in comparison with the ER (Equivalent Reliability) model. All primary inputs were assumed to be error-free and have a probability value of 0.5 with the reliability of all the gates set at 0.95.

Table 4 All signal output reliability comparison between ER, Improved ER and proposed Model on ISCAS85 circuits with $r_g = 0.95$

Circuit	# Gates	# Input	# Output	Average Signal error % Proposed Model	Average Signal error % ER - Model	Average Signal error % Improved ER - Model
C17	6	5	2	0.05	0.22	0.10
C432	160	36	7	1.36	1.63	1.05
C499	202	41	32	0.16	0.24	1.36
C880	383	60	26	1.80	2.75	1.54
C1355	546	41	32	5.14	8.83	4.11
C1908	880	33	25	3.08	5.30	4.15
C2670	1193	233	140	1.29	4.57	1.97
C3540	1669	50	22	2.05	5.46	2.29
C5315	2370	178	123	0.58	4.92	2.50
Average				1.67%	3.77%	2.12%

It can be observed from the comparison Table 4 above that the average error for the proposed model is 1.67% whereas the average error for the ER model is around 3.77%

which means the proposed model produces a more accurate estimate of the output reliability. Analyzing the table further we can see that for every individual circuit, the proposed model produces better results and for larger circuits such as C1355, C2670 and C5315, the proposed model produces a far superior output compared to the ER model. However, even though the average error for the proposed model and the Improved ER is very close (1.67% for the proposed model and 2.12% for the Improved ER), individually the proposed model produces far more accurate results for some of the circuits. For example, for C499 and C5315 the proposed model was more accurate than the Improved ER model.

In Table 5 below ISCAS85 Benchmark circuits to verify the accuracy of the proposed model in comparison with the Three-Point Method, modified Three-Point Method and PGM. All primary inputs were assumed to be error-free and have a probability value of 0.5 with the reliability of all the gates set at 0.9.

Table 5 All primary output signal reliability comparison between Three-Point Method, modified Three-Point Method, PGM and proposed Model on ISCAS85 circuits with r_g =0.9

Circuit	**# Gates**	**# Input**	**# Output**	**Average Signal error % Proposed Model**	**Average Output Signal error % Three-Point Method**	**Average Output Signal error % Modified Three-Point Method**	**Average Output Signal error % PGM**
C17	6	5	2	0.11	2.08	2.08	33.15
C432	160	36	7	1.56	2.88	4.04	4.37
C499	202	41	32	0.24	0.62	0.67	34.21
C880	383	60	26	2.41	1.83	4.86	8.63
C1355	546	41	32	0.24	0.65	0.65	21.53
C1908	880	33	25	2.02	4.56	4.56	17.31
C2670	1193	233	140	1.26	2.83	3.28	17.80
C3540	1669	50	22	1.87	2.74	3.77	19.57
C5315	2370	178	123	1.53	4.81	5.15	7.94
Average				1.25	2.56	3.23	18.28

From the results in Table 5 it can be observed that the proposed method has an average error of 1.25% when compared with Monte-Carlo Simulation. This result is better than all three of the methods it was compared with having the best results in comparison to PGM method. We do see that in C880 the proposed method generated higher error as compared to the Three-point method but otherwise the error percentage is less for all the other circuits.

The two simulation results prove that the proposed model produces results that are more accurate than the other methods it was compared with proving the validity of the model.

The estimated reliability value calculated using the proposed model will be more accurate as the reliability of the gate increases. Realistically the gates are assumed to have high reliability values usually in the range of 0.98-0.9999. Therefore, the following tables, Table 6 shows how the accuracy of the proposed model changes as we change the reliability of the gate from 0.9 to 0.99 for C432 and C499 benchmark circuits.

Table 6 All gate signal error % trend for different r_g using proposed model

Reliability of the gate, r_g	**% error for all signals in C432**	**% error for all signals in C499**
0.90	1.56	0.24
0.91	1.58	0.21
0.92	1.59	0.18
0.93	1.57	0.16
0.94	1.44	0.15
0.95	1.36	0.16
0.96	1.23	0.13
0.97	1.03	0.11
0.98	0.85	0.08
0.99	0.56	0.04

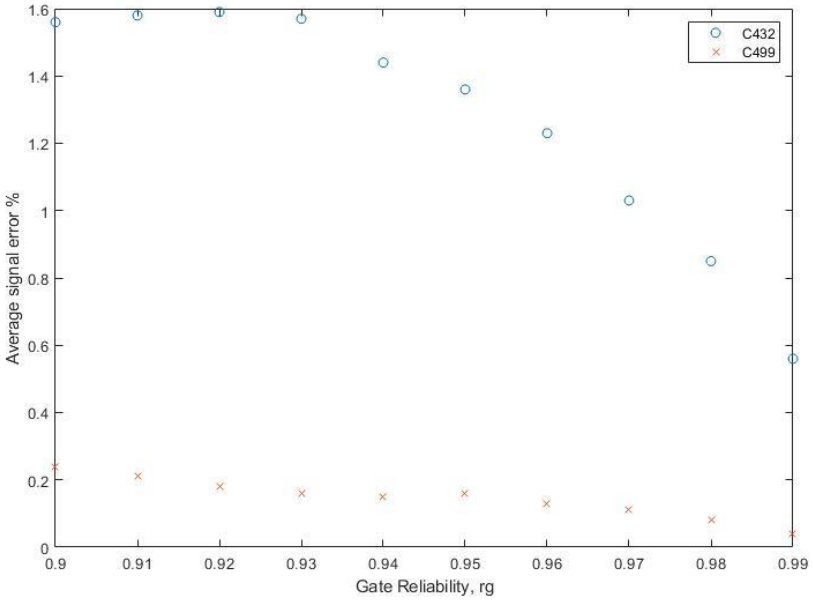

Figure 4- 1 Average signal error % trend for different r_g for C432 and C499

CHAPTER 5
Conclusion and Future Work

5.1 Conclusion

In the previous section, it has been confirmed via simulation that the proposed method gives a better result over some of the existing methods such as ER model, Three-Point Method and PGM in terms of accuracy. The results from the simulation showed that the proposed model on average had an error percentage of 1.67% while the average error percentage for ER model was 3.77% when compared under the same circuit conditions. The simulations also showed that the proposed model on average had an error percentage of 1.25% whereas the Three-Point Method had an average error percentage of 2.56% while the PGM method had an average error percentage of 18.28% under the same circuit conditions. All these data suggests that the proposed model is a more accurate analytical method for calculating the reliability of an integrated circuit.

However, due to the scope of the research it was assumed that the probability of an error-free circuit was already made available. Unfortunately, that is not always the case and the probability of an error-free circuit can be calculated using a fast Monte-Carlo Simulation.

5.2 Future Works

The proposed model is developed using the combinational circuits and so it will work perfectly for any combinational circuit. Sequential circuits, on the other hand, have a different operating principle and so as a result the model will not work on sequential circuits. Since sequential circuits require multiple cycles before producing the result, the model needs to be modified in the future so that they can work perfectly for sequential circuits.

The model also assumes that the probability of an error-free circuit is already made available. In order to make the entire reliability analysis be linear in terms of computational time, the probability of the error-free circuit also needs to be calculated using equations with the signal probability correlation coefficient calculated using a model. This probability correlation coefficient model can be made using the concept of the proposed model with new assumptions and modifications. Future works can be done in order to develop this new model and calculate the probability correlation coefficient using local information within the circuit in linear time.

REFERENCES

[1] M. R. Choudhury and K. Mohanram, "Reliability Analysis of Logic Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 28, no. 3, 2009.

[2] R. Y. Rubinstein and D. P. Kroese, Simulation and the Monte Carlo method, John Wiley & Sons, 2008.

[3] J. Von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," in *Automata Studies*, Princeton, Princeton University Press, 1946, pp. 43-98.

[4] S. Krishnaswamy, G. F. Viamontes, I. L. Markov and J. P. Hayes, "Accurate reliability evaluation and enhancement via probabilistic transfer matrices," in *Proceedings of the Design Automation and Test in Europe*, 2005.

[5] J. Han, H. Chen, E. Boykin and J. Fortes, "Reliability evaluation of logic circuits using probabilistic gate models," *Microelectronics Reliability,* vol. 51, no. 2, pp. 468-476, 2011.

[6] H. Jahanirad, "CC-SPRA: Correlation Coefficient Approach for Signal Probability-based Reliability Analysis," *IEEE Transactions on Very Large Scale Integrated Circuit (VLSI) Systems,* vol. 27, no. 4, pp. 927-939, 2019.

[7] C. Chen and R. Xiao, "A fast model for analysis and improvement of gate-level circuit reliability," *INTEGRATIONS, the VLSI Journal,* vol. 50, pp. 107-115, 2015.

[8] J. Cai, "Speeding up Reliability Analysis for Large-scale Circuits," in *M.Ap.Sc Thesis*, University of Windsor, 2016.

APPENDICES

Appendix A

Reliability analysis

```
function [r,p,t,checker,tm] =
Reliability_Proposed_Model(rgate,pi,file_name)


run Reading_benchmark_circuit.m
x = size(Gate);
checker = 0;
% p = zeros(10^5,1);
% r = zeros(10^5,3);
t = zeros(1,1000);
tm = zeros(1,1000);
for i = 1:length(Input)
    p(Input(i),1) = 0.5;
end

for i = 1:length(Input)
    r(Input(i),1) = 1;                              % Reliability of the
Signal
    r(Input(i),2) = 1;                              % R0
    r(Input(i),3) = 1;                              % R1
end



[c_t,sxx,xx] = Fd_pg(Gate,Input);
C = zeros(10^3,4);
% pxi = [0.75 0.75 0.4375];
for i = 1:x(1)
   C(i,:) = [1 1 1 1];
end
for i = 1:x(1)
    checker = checker+1;
    tf =  tic;
  if c_t(i,4)==0
    C(i,:) = [1 1 1 1];
  else
      m = c_t(i,4);
      t_model = tic;
      [rst_g,m1] = R_state_pg_v4(Gate,Input,r,p,i,C,xx,c_t);
      if m1~= 0
          try
            C(i,:) =
Correlation_Coeffiecient_NAND_v22(r(Gate(find(Gate(:,1)==m,1))),r(Gate(
find(Gate(:,1)==m,1))),Gate(i,4),c_t(i,2),c_t(i,3),rst_g(Gate(i,2),2:3)
,rst_g(Gate(i,3),2:3),r(Gate(i,2),2),r(Gate(i,2),3),r(Gate(i,3),2),r(Ga
te(i,3),3));
          catch
              C(i,:) = [1 1 1 1];
          end
```

```
      else
          C(i,:) = [1 1 1 1];
      end
       tm(i,1) = toc(t_model);
%       clear rst_g
  end
    if Gate(i,3) == 0
        [r(Gate(i,1),1),p(Gate(i,1)),r(Gate(i,1),3),r(Gate(i,1),2)] =
Gate_to_do_matrix_norb(Gate(i,5),pi(Gate(i,1)),Gate(i,4),r(Gate(i,2),2)
,r(Gate(i,2),3));

    else
        [r(Gate(i,1),1),p(Gate(i,1)),r(Gate(i,1),3),r(Gate(i,1),2)] =
Gate_to_do_matrix(Gate(i,5),p(Gate(i,2)),p(Gate(i,3)),pi(Gate(i,1)),Gat
e(i,4),C(i,:),r(Gate(i,2),2),r(Gate(i,2),3),r(Gate(i,3),2),r(Gate(i,3),
3));
    end
   t(i) = toc(tf);
end

t = nonzeros(t)';
tm = nonzeros(tm)';

end
```

Reliability Correlation Coefficient using Proposed Model

```
function [C,t] =
Correlation_Coeffiecient_Proposed_Model(rm,rm_state,rg,L1,L2,ra_state,r
b_state,ra0,ra1,rb0,rb1)



tic
x10 = 1 - (ra_state(1)/rm_state);
x11 = 1 - (ra0/rg);

x1 = L1 * (x10/x11);

if rm<=1 && rm> 0.8
    alp = 1.226;
elseif rm <= 0.8 && rm > 0.7
    alp = (6.76*rm) - 4.182;
elseif rm <= 0.7
    alp = 0.55;
end

x20 = 1 - (rb_state(1)/rm_state);
x21 = 1 - (rb0/rg);
x2 = L2 * (x20/x21);
if L1 == 0

        s = alp*(1-rm)*(x2);
```

```
elseif L2 ==0

    s = alp*(1-rm)*(x1);

else

        s = alp*(1-rm)*(x1+x2);

end
try
        s1 = abs(s);
        C(1) = 1 + (((1/rm)-1)*exp(-s1));

catch
    C(1) = 1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x10 = 1 - (ra_state(1)/rm_state);
x11 = 1 - (ra0/rg);
x1 = L1 * (x10/x11);

x20 = 1 - (rb_state(2)/rm_state);
x21 = 1 - (rb1/rg);
x2 = L2 * (x20/x21);

if L1 == 0

        s = alp*(1-rm)*(x2);


elseif L2 ==0

    s = alp*(1-rm)*(x1);

else

        s = alp*(1-rm)*(x1+x2);

end

try
        s1 = abs(s);
        C(2) = 1 + (((1/rm)-1)*exp(-s1));

catch
    C(2) = 1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x10 = 1 - (ra_state(2)/rm_state);
x11 = 1 - (ra1/rg);
x1 = L1 * (x10/x11);
```

```
x20 = 1 - (rb_state(1)/rm_state);
x21 = 1 - (rb0/rg);
x2 = L2 * (x20/x21);

if L1 == 0

        s = alp*(1-rm)*(x2);


elseif L2 ==0

    s = alp*(1-rm)*(x1);

else

        s = alp*(1-rm)*(x1+x2);

end

try
        s1 = abs(s);
        C(3) = 1 + (((1/rm)-1)*exp(-s1));

catch
    C(3) = 1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x10 = 1 - (ra_state(2)/rm_state);
x11 = 1 - (ra1/rg);
x1 = L1 * (x10/x11);

x20 = 1 - (rb_state(2)/rm_state);
x21 = 1 - (rb1/rg);
x2 = L2 * (x20/x21);

if L1 == 0

        s = alp*(1-rm)*(x2);


elseif L2 ==0

    s = alp*(1-rm)*(x1);

else

        s = alp*(1-rm)*(x1+x2);

end
```

```
try
        s1 = abs(s);
        C(4) = 1 + (((1/rm)-1)*exp(-s1));

catch
    C(4) = 1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

t = toc;
end
```

Obtaining level and fan-out information topologically for all gate signals

```
function [c_t,sxx,xx,t] = Fd_pg(Gate,Input)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
sxx = zeros(1,10);
tic
for c = 1:length(Gate(:,1))

%     c = 6
    g=c;
    % i1 = i;
    % i2 = i;
    L1 = 0;
    L2 = 0;
    x1(1) = Gate(g,1);
    x=0;
    % j=0;

    % x1(2) = Gate(j,2);
    % x1(3) = Gate(j,3);

%     x1
    i = 1;
    s = 1;
    j = 1;

    while x == 0
      try
        if ~any(x1(j)==Input)

                try
                    x1(i+1) = Gate(find(Gate(:,1)==x1(j)),2);
                catch ME
                end

                try
                    x1(i+2) = Gate(find(Gate(:,1)==x1(j)),3);
                catch ME
                end
```

```
        else
            x1(i+1) = 0;
            x1(i+2) = 0;
        end
      catch ME
      end

%     if ~any(x1(3)==Input)
%         x1(6) = Gate(find(Gate(:,1)==x1(3)),2);
%         x1(7) = Gate(find(Gate(:,1)==x1(3)),3);
%     end
      x1 = nonzeros(x1)';
      m_sor = nonzeros(unique(x1))';

          if ~isempty(m_sor(find(hist(x1,unique(x1))>1)))
              m = m_sor(find(hist(x1,unique(x1))>1));
              x = 1;
          else
              m = 0;
              x = 0;
          end
%           if x1(i+1)==x1(i+2)
%               x = 1;
%           end
%          if length(x1)<5
%           if any(x1(end-1)==Input) && any(x1(end)==Input)
%               x = 1;
%           end
%          end
          if (Gate(c,5) == 5)|| (Gate(c,5) == 6)
              m = 0;
          end
       s = s+1;
       if s==15
          x = 1;
       end
        sxx(c,1:length(m)) = m;
       if length(m)>1


           m = m(1);
       end

        i = i+2;
        j = j+1;

if  m~=0

    x2 = m;
%
%     L1s = Gate(find(Gate(:,2)==x2),1)
%
%     L2s = Gate(find(Gate(:,3)==x2),1)
%     LL1 = size(L1s);
```

```
%     LL2 = size(L2s);

    L1as = Gate(find(Gate(:,2)==x2),1);

    L2as = Gate(find(Gate(:,3)==x2),1);

    L1s = nonzeros(any(x1 == L1as).*x1);
    L2s = nonzeros(any(x1 == L1as).*x2);

    LL1 = size(L1s);
    LL2 = size(L2s);
%


%
%     if LL1(1) > 1
%         L1s = L1s(1,:);
%     end
%     if LL2(1) > 1
%         L2s = L2s(1,:);
%     end
% %     L1s
%     L2s

    x = 0;
    i = 1;
    j = 1;
    s = 0;
    while x == 0
%         clear [ns,nsx,ls,lsx]
        try
            ns = Gate(find(Gate(:,2)==L1s(i)),1);
            nsx = Gate(find(Gate(:,3)==L1s(i)),1);
        catch ME
        end
        try
        if nonzeros(any(ns==x1))
            try
                L1s(i+1) = ns;
            catch ME
            end
        elseif nonzeros(any(nsx == x1))
            try
                L1s(i+1) = nsx;
            catch ME
            end
        end
        catch ME
        end
       try
            ls = Gate(find(Gate(:,2)==L2s(i)),1);
            lsx = Gate(find(Gate(:,3)==L2s(i)),1);
       catch ME
       end

       try
```

```
        if nonzeros(any(ls==x1))
            try
                L2s(i+1) = ls;
            catch ME
            end
        elseif nonzeros(any(lsx == x1))
            try
                L2s(i+1) = lsx;
            catch ME
            end
        end
       catch ME
       end
        if any(L1s == Gate(c,1)) && any(L2s == Gate(c,1))
            x= 1;
        end
        s = s + 1;
        if s ==15
            x = 1;
        end
        i = i+1;
    end
%     clear [x2,L1s,L2s]


        if isempty(L1s)
            L1 = 0;
        else
            L1 =  length(L1s);
        end

        if isempty(L2s)
            L2 = 0;
        else
            L2 =  length(L2s) ;
        end

else
    L1 = 0;
    L2 = 0;


end



%           run R_state_runing.mlx
    end
    xx(c,:) = zeros(1,50);
    x1;
    c_t(c,1) = Gate(c,1);
    c_t(c,2) = L1;
    c_t(c,3) = L2;
    c_t(c,4) = m;
    if ~isempty(x1)
```

```
        xx(c,1:length(x1)) = x1;


    end

%     r_s(c,:) = rst_out;

    clear x1

end

t = toc;
end
```

Correlating signal paths

```
function [rst_out,m,t] = R_state_pg_v4(Gate,Input,r,p,c,C,xx,c_t)
%UNTITLED9 Summary of this function goes here
%   Detailed explanation goes here


tic
% for c = 1:length(Gate(:,1))
% t2 = zeros(1000,1);
%     c = 9
    g=c;
    % i1 = i;
    % i2 = i;
%     L1 = 0;
%     L2 = 0;
%     x1(1) = Gate(g,1);
%     x=0;
    % j=0;

    % x1(2) = Gate(j,2);
    % x1(3) = Gate(j,3);

%     x1
%     i = 1;
%     s = 1;
%     j = 1;

%     while x == 0
%       try
%         if ~any(x1(j)==Input)
%
%                 try
%                     x1(i+1) = Gate(find(Gate(:,1)==x1(j)),2);
%                 catch ME
%                 end
%
%                 try
%                     x1(i+2) = Gate(find(Gate(:,1)==x1(j)),3);
```

```
%                 catch ME
%                 end
%
%         else
%             x1(i+1) = 0;
%             x1(i+2) = 0;
%         end
%       catch ME
%       end
%
%     %     if ~any(x1(3)==Input)
%     %         x1(6) = Gate(find(Gate(:,1)==x1(3)),2);
%     %         x1(7) = Gate(find(Gate(:,1)==x1(3)),3);
%     %     end
%         x1 = nonzeros(x1)';
%         m_sor = nonzeros(unique(x1))';
%
%             if ~isempty(m_sor(find(hist(x1,unique(x1))>1)))
%                 m = m_sor(find(hist(x1,unique(x1))>1));
%                 x = 1;
%             else
%                 m = 0;
%                 x = 0;
%             end
% %             if x1(i+1)==x1(i+2)
% %                 x = 1;
% %             end
% %            if length(x1)<5
% %             if any(x1(end-1)==Input) && any(x1(end)==Input)
% %                 x = 1;
% %             end
% %            end
%             if (Gate(c,5) == 5)|| (Gate(c,5) == 6)
%                 m = 0;
%             end
%          s = s+1;
%          if s==25
%             x = 1;
%          end
%
%          if length(m)>1
%              m = m(1);
%          end
%
%           i = i+2;
%           j = j+1;
%
% %           run R_state_runing.mlx
%     end
    xx(c,:);
    m = c_t(c,4);
%     x1;

if  m~=0
    x2 = m;
    m = x2;
    L1s = zeros(1,100);
```

```
    L2s = zeros(1,100);
    L1as = Gate(Gate(:,2)==x2,1);

    L2as = Gate(Gate(:,3)==x2,1);

    x1 = nonzeros(xx(c,:))';
    LL1 = size(L1as);
    LL2 = size(L2as);
    if ~isempty(L1as)
        if LL1(1) == 1
            if ~isempty(nonzeros(any(x1 == L1as).*x1))
                L1s(1) = L1as;
            else
                L1s(1) = 0;
            end
        elseif LL1(1) > 1
             if  nonzeros(any(x1 == L1as).*x1) > 1
                 L1sof = nonzeros(any(x1 == L1as).*x1);
                 L1s(1) = L1sof(1);
             else
                 L1s(1) = 0;
             end
        end
    else
        L1s = L1as;
    end

    if ~isempty(L2as)
        if LL2(1) == 1
            if  ~isempty(nonzeros(any(x1 == L2as).*x1))
                L2s(1) = L2as;
            else
                L2s(1) = 0;
            end
        elseif LL2(1) > 1
             if  nonzeros(any(x1 == L2as).*x1) > 1
                 L2sof = nonzeros(any(x1 == L2as).*x1);
                 L2s(1) = L2sof(1);
             else
                 L2s(1) = 0;
             end
        end
    else
        L2s = L2as;
    end

%
%     if LL1(1) > 1
%         L1s = L1s(1,:);
%     end
%     if LL2(1) > 1
%         L2s = L2s(1,:);
%     end
% %     L1s
%     L2s
```

```
x = 0;
i = 1;
j = 1;
s = 0;
while x == 0
    try
        ns = Gate(Gate(:,2)==L1s(i),1);
    catch
    end
    try
        nsx = Gate(Gate(:,3)==L1s(i),1);
    catch
    end
    try
    if nonzeros(any(ns==x1))
        try
            g2es = nonzeros(any(x1==ns).*x1);
            if length(g2es) > 1
                L1s(i+1) = x1(x1==ns);
            else
                L1s(i+1) = g2es;
            end

        catch
        end
    elseif nonzeros(any(nsx == x1))
        try
            tles = nonzeros(any(x1==nsx).*x1);
            if length(tles) >1
                L1s(i+1) = x1(x1==nsx);
            else
                L1s(i+1) = tles;
            end
        catch
        end
    end
    catch
    end
   try
        ls = Gate(Gate(:,2)==L2s(i),1);
   catch
   end
   try
        lsx = Gate(Gate(:,3)==L2s(i),1);
   catch
   end

   try
    if nonzeros(any(ls==x1))
        try
            sunes = nonzeros(any(x1==ls).*x1);
            if length(sunes) > 1
                L2s(i+1) = x1(x1==ls);
            else
                L2s(i+1) = sunes;
            end
```

```
            catch
            end
        elseif nonzeros(any(lsx == x1))
            try
                mcxes = nonzeros(any(x1==lsx).*x1);
                if length(mcxes) > 1
                    L2s(i+1) = x1(x1==lsx);
                else
                    L2s(i+1) = mcxes;
                end
            catch
            end
        end
       catch
       end
        if any(L1s == Gate(c,1)) && any(L2s == Gate(c,1))
            x= 1;
        end
        s = s + 1;
        if s ==15
            x = 1;
        end
        i = i+1;
    end
%     clear [x2,L1s,L2s]

        if isempty(L1s)
            L1 = 0;
        else
            L1 =  length(L1s);
        end

        if isempty(L2s)
            L2 = 0;
        else
            L2 =  length(L2s) ;
        end
        L1s = nonzeros(L1s)';
        L2s = nonzeros(L2s)';
        try
            if L1s(1) == 0

            end
        catch
                L1 = 0;
                L2 = 0;
                m = 0;
                L1s = 0;
                L2s = 0;

        end

        try
            if L2s(1) == 0

            end
```

```
        catch
                L1 = 0;
                L2 = 0;
                m = 0;
                L1s = 0;
                L2s = 0;

        end

else
    L1 = 0;
    L2 = 0;

end


%     c_t(c,1) = Gate(c,1);
%     c_t(c,2) = L1;
%     c_t(c,3) = L2;
%     c_t(c,4) = m;
%     r_s(c,:) = rst_out;

%     clear x1

% end
        lmfao = max(Gate(:,1));
%         lmfao = 507;
        rst = zeros(lmfao,3);
        t1 = toc;
 if m~=0

    LL1 = size(L1s);
    LL2 = size(L2s);
    if LL1(1) > 1
        L1s = L1s';
    end
    if LL2(1) > 1
        L2s = L2s';
    end
%     if LL1(1) == LL2(1)
%         if ~isempty(L1s) && ~isempty(L2s)
%           rou = horzcat(L1s,L2s);
%         elseif ~isempty(L1s)
%             rou = L2s;
%         elseif ~isempty(L2s)
%             rou = L1s;
%         end
%     else
%         if ~isempty(L1s) && ~isempty(L2s)
%             rou = horzcat(L1s(1,:),L2s(1,:));
%         elseif ~isempty(L1s)
%             rou = L2s;
%         elseif ~isempty(L2s)
%             rou = L1s;
%         end
```

```
%     end
    rou = nonzeros(rou)';

    for ilx = 1:length(rou)
        tic

        in1 = (Gate((Gate(:,1)==rou(ilx)),2));
        in2 = (Gate((Gate(:,1)==rou(ilx)),3));


%         C = [1 1 1 1];
       if in1~=0
            if rst(in1,2) == 0

                    rst(in1,2) = r(in1,2);

            else

                rst(in1,2) = rst(in1,2);

            end

            if rst(in1,3) == 0

                    rst(in1,3) = r(in1,3);

            else

                rst(in1,3) = rst(in1,3);

            end
       end

       if in2~=0

            if rst(in2,2) == 0
                try
                    rst(in2,2) = r(in2,2);
                catch rst(in2,2) = 0;
                end

            else
                try
                    rst(in2,2) = rst(in2,2);
                catch rst(in2,2) = 0;
                end
            end

            if rst(in2,3) == 0
                try
                    rst(in2,3) = r(in2,3);
                catch rst(in2,3) = 0;
                end
```

```
            else
                try
                    rst(in2,3) = rst(in2,3);
                catch rst(in2,3) = 0;
                end
            end
        end

        gta = Gate(Gate(:,1)==rou(ilx),5);
         gtr = Gate(Gate(:,1)==rou(ilx),4);


         t3s(ilx) = toc;
         if (in2~=0) && (in1~=0)
            if in1 == m
                [rst(rou(ilx),2),rst(rou(ilx),3),t2x(ilx)] =
r_state_find(gta,p(in2),p(in1),gtr,rst(in2,2),rst(in2,3),rst(in1,2),rst
(in1,3),C(ilx,:));
            elseif in2 == m

                    [rst(rou(ilx),2),rst(rou(ilx),3),t2x(ilx)] =
r_state_find(gta,p(in1),p(in2),gtr,rst(in1,2),rst(in1,3),rst(in2,2),rst
(in2,3),C(ilx,:));

            elseif in1~=m && in2~=m
                if any(in1==rou)
                    [rst(rou(ilx),2),rst(rou(ilx),3),t2x(ilx)] =
r_state_find(gta,p(in2),p(in1),gtr,rst(in2,2),rst(in2,3),rst(in1,2),rst
(in1,3),C(ilx,:));
                elseif any(in2==rou)
                    [rst(rou(ilx),2),rst(rou(ilx),3),t2x(ilx)] =
r_state_find(gta,p(in1),p(in2),gtr,rst(in1,2),rst(in1,3),rst(in2,2),rst
(in2,3),C(ilx,:));
                end

            end
         else
            if in1~=0
                if gta == 6
                    [rst(rou(ilx),2),rst(rou(ilx),3),t2x(ilx)] =
Buffer_gate_Matrix(p(in1),gtr,rst(in1,2),rst(in1,3));
                elseif gta == 5
                    [rst(rou(ilx),2),rst(rou(ilx),3),t2x(ilx)] =
Buffer_gate_Matrix(p(in1),gtr,rst(in1,2),rst(in1,3));       %%
                end
            elseif in2~=0
                 if gta == 6
                    [rst(rou(ilx),2),rst(rou(ilx),3),t2x(ilx)] =
Buffer_gate_Matrix(p(in2),gtr,rst(in2,2),rst(in2,3));
                elseif gta == 5
                    [rst(rou(ilx),2),rst(rou(ilx),3),t2x(ilx)] =
Buffer_gate_Matrix(p(in2),gtr,rst(in2,2),rst(in2,3));       %%
                end
            end
```

```
        end
    end
%         rst;


        rst_out = rst;

 else

 rst_out = rst;

 end
 try
 t2 = sum(t2x);
        t3 = sum(t3s);
        t = t1+t2+t3;
 catch
     t2 = 0;
     t3 = 0;
     t = t1+t2+t3;
 end

        clear x1
% end
        clear rou
        clear L1s
        clear L2s
        clear rst
        clear L1as
        clear L2as
%         clear m
        clear L1
        clear L2

end
```

Calculating the effective reliability pair

```
function [r0_y2,r1_y2,t] =
r_state_find(Gate_type,p_a,p_b,R_g,ra0,ra1,rb0,rb1,C)
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here
switch(Gate_type)
    case{1}
        [r0_y2,r1_y2,t] = r_state_NAND(p_a,p_b,R_g,ra0,ra1,rb0,rb1,C);
    case{2}
        [r0_y2,r1_y2,t] = r_state_AND(p_a,p_b,R_g,ra0,ra1,rb0,rb1,C);
    case{3}
        [r0_y2,r1_y2,t] = r_state_NOR(p_a,p_b,R_g,ra0,ra1,rb0,rb1,C);
    case{4}
        [r0_y2,r1_y2,t] = r_state_OR(p_a,p_b,R_g,ra0,ra1,rb0,rb1,C);
    case{5}
        [x,y,r0_y2,r1_y2,t] = Buffer_gate_Matrix(p_a,R_g,ra0,ra1);
    case{6}
```

```
        [x,y,r0_y2,r1_y2,t] = NOT_gate_Matrix(p_a,R_g,ra0,ra1);
    case{7}
        [r0_y2,r1_y2,t] = r_state_XOR(p_a,p_b,R_g,ra0,ra1,rb0,rb1,C);
    case{8}
        [r0_y2,r1_y2,t] = r_state_XNOR(p_a,p_b,R_g,ra0,ra1,rb0,rb1,C);
end
```

NAND Gate

```
function [r0_y2,r1_y2,t] = r_state_NAND(p_a,p_b,R_g,ra0,ra1,rb0,rb1,C)
%R_state The output gives the e
%   Detailed explanation goes here
a = rand(1,10^6) < p_a;
b = rand(1,10^6) < p_b;
R_2 = [R_g R_g R_g 1-R_g]';


Ps_ab = P_star(a,b);
tic
M = M_out(ra0,ra1,rb0,rb1,C);

M1_y2 = M(1:3,:);
M0_y2 = M(4,:);

P1_y2 = [Ps_ab(1) 0 Ps_ab(3)];

r1_y2 = ((P1_y2*M1_y2)*R_2)/(sum(Ps_ab(1:3)));
r0_y2 = M0_y2*(1-R_2);

t = toc;

end
```

AND Gate

```
function [r0_y2,r1_y2,t] = r_state_AND(p_a,p_b,R_g,ra0,ra1,rb0,rb1,C)
%R_state The output gives the e
%   Detailed explanation goes here
a = rand(1,10^6) < p_a;
b = rand(1,10^6) < p_b;
R_2 = [R_g R_g R_g 1-R_g]';


Ps_ab = P_star(a,b);
tic
M = M_out(ra0,ra1,rb0,rb1,C);

M1_y2 = M(4,:);
M0_y2 = M(1:3,:);

P1_y2 = [Ps_ab(1) 0 Ps_ab(3)];

r0_y2 = ((P1_y2*M0_y2)*R_2)/(1 - sum(Ps_ab(4)));
```

```
r1_y2 = M1_y2*(1-R_2);

t = toc;

end
```

NOR Gate

```
function [r0_y2,r1_y2,t] = r_state_NOR(p_a,p_b,R_g,ra0,ra1,rb0,rb1,C)
%UNTITLED17 Summary of this function goes here
%   Detailed explanation goes here
a = rand(1,10^6) < p_a;
b = rand(1,10^6) < p_b;
R_2 = [1-R_g R_g R_g R_g]';


Ps_ab = P_star(a,b);

tic
M = M_out(ra0,ra1,rb0,rb1,C);

M0_y2 = M(2:4,:);
M1_y2 = M(1,:);

P0_y2 = [Ps_ab(2) 0 Ps_ab(4)];
P1_y2= Ps_ab(1);

r0_y2 = ((P0_y2*M0_y2)*R_2)/(1 - sum(Ps_ab(1)));
r1_y2 = M1_y2*(1-R_2);
t = toc;
% r_y2 = (r1_y2*pc)+((1-pc)*r0_y2);
% p_y2 = (r1_y2*pc)+((1-pc)*(1-r0_y2));
end
```

OR Gate

```
function [r0_y2,r1_y2,t] = r_state_OR(p_a,p_b,R_g,ra0,ra1,rb0,rb1,C)
%UNTITLED19 Summary of this function goes here
%   Detailed explanation goes here
a = rand(1,10^6) < p_a;
b = rand(1,10^6) < p_b;
R_2 = [1-R_g R_g R_g R_g]';


Ps_ab = P_star(a,b);

tic
M = M_out(ra0,ra1,rb0,rb1,C);

M0_y2 = M(1,:);
M1_y2 = M(2:4,:);
```

```
P0_y2 = Ps_ab(1);
P1_y2= [Ps_ab(2) 0 Ps_ab(4)];

r0_y2 = M0_y2*(1-R_2);
r1_y2 = ((P1_y2*M1_y2)*R_2)/(sum(Ps_ab(2:4)));


t = toc;
% r_y2 = (r1_y2*pc)+((1-pc)*r0_y2);
% p_y2 = (r1_y2*pc)+((1-pc)*(1-r0_y2));
end
```

XOR Gate

```
function [r0_y2,r1_y2,t] = r_state_XOR(p_a,p_b,R_g,ra0,ra1,rb0,rb1,C)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
a = rand(1,10^6) < p_a;
b = rand(1,10^6) < p_b;
R_2 = [1-R_g R_g R_g 1-R_g]';


Ps_ab = P_star(a,b);
tic
M = M_out(ra0,ra1,rb0,rb1,C);

M0_y2 = [M(1,:);M(4,:)];
M1_y2 = [M(2,:);M(3,:)];

P0_y2 = [Ps_ab(1),Ps_ab(4)];
P1_y2= Ps_ab(2:3);

r0_y2 = ((P0_y2*M0_y2)*(1-R_2))/(1 - sum([Ps_ab(1),Ps_ab(4)]));
r1_y2 = ((P1_y2*M1_y2)*(R_2))/(sum([Ps_ab(1),Ps_ab(4)]));


t = toc;
% r_y2 = (r1_y2*pc)+((1-pc)*r0_y2);
% p_y2 = (r1_y2*pc)+((1-pc)*(1-r0_y2));
end
```

XNOR Gate

```
function [r0_y2,r1_y2,t] = r_state_XNOR(p_a,p_b,R_g,ra0,ra1,rb0,rb1,C)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
a = rand(1,10^6) < p_a;
b = rand(1,10^6) < p_b;
R_2 = [1-R_g R_g R_g 1-R_g]';


Ps_ab = P_star(a,b);
```

```
tic
M = M_out(ra0,ra1,rb0,rb1,C);

M0_y2 = [M(2,:);M(3,:)];
M1_y2 = [M(1,:);M(4,:)];

P0_y2 = [Ps_ab(2),Ps_ab(3)];
P1_y2 = [Ps_ab(1),Ps_ab(4)];

r0_y2 = ((P0_y2*M0_y2)*(R_2))/(1 - sum([Ps_ab(1),Ps_ab(4)]));
r1_y2 = ((P1_y2*M1_y2)*(1-R_2))/(sum([Ps_ab(1),Ps_ab(4)]));


t = toc;
% r_y2 = (r1_y2*pc)+((1-pc)*r0_y2);
% p_y2 = (r1_y2*pc)+((1-pc)*(1-r0_y2));
end
```

Reliability pair, Probability and Overall Reliability Calculation

```
function [r_y,p_y2,r1_y2,r0_y2,t] =
Gate_to_do_matrix(Gate_type,p_a,p_b,pc,R_g,C,ra0,ra1,rb0,rb1)
%UNTITLED5 Summary of this function goes here
%   Detailed explanation goes here
switch(Gate_type)
    case{1}
        [r_y,p_y2,r1_y2,r0_y2,t] =
NAND_gate_Matrix(p_a,p_b,pc,R_g,C,ra0,ra1,rb0,rb1);
    case{2}
        [r_y,p_y2,r1_y2,r0_y2,t] =
AND_gate_Matrix(p_a,p_b,pc,R_g,C,ra0,ra1,rb0,rb1);
    case{3}
        [r_y,p_y2,r1_y2,r0_y2,t] =
NOR_gate_Matrix(p_a,p_b,pc,R_g,C,ra0,ra1,rb0,rb1);
    case{4}
        [r_y,p_y2,r1_y2,r0_y2,t] =
OR_gate_Matrix(p_a,p_b,pc,R_g,C,ra0,ra1,rb0,rb1);
    case{5}
        [r_y,p_y2,r1_y2,r0_y2,t] = NOT_gate_Matrix(p_a,ra0,ra1);
    case{6}
        [r_y,p_y2,r1_y2,r0_y2,t] = Buffer_gate_Matrix(p_a,ra0,ra1);
    case{7}
        [r_y,p_y2,r1_y2,r0_y2,t] =
XOR_gate_Matrix(p_a,p_b,pc,R_g,C,ra0,ra1,rb0,rb1);
    case{8}
        [r_y,p_y2,r1_y2,r0_y2,t] =
XNOR_gate_Matrix(p_a,p_b,pc,R_g,C,ra0,ra1,rb0,rb1);
end

end
```

NAND Gate

```
function [r_y2,p_y2,r1_y2,r0_y2,t] =
NAND_gate_Matrix(p_a,p_b,pc,R_g,C,ra0,ra1,rb0,rb1)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
a = rand(1,10^6) < p_a;
b = rand(1,10^6) < p_b;
R_2 = [R_g R_g R_g 1-R_g]';


Ps_ab = P_star(a,b);

tic
M = M_out(ra0,ra1,rb0,rb1,C);

M1_y2 = M(1:3,:);
M0_y2 = M(4,:);


    P1_y2 = Ps_ab(1:3);

try
    P0_y2= Ps_ab(4);
catch ME
end
    r1_y2 = ((P1_y2*M1_y2)*R_2)/(sum(Ps_ab(1:3)));
    r0_y2 = M0_y2*(1-R_2);

r_y2 = (r1_y2*pc)+((1-pc)*r0_y2);
p_y2 = (r1_y2*pc)+((1-pc)*(1-r0_y2));
t = toc;
end
```

AND Gate

```
function [r_y2,p_y2,r1_y2,r0_y2,t] =
AND_gate_Matrix(p_a,p_b,pc,R_g,C,ra0,ra1,rb0,rb1)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
a = rand(1,10^6) < p_a;
b = rand(1,10^6) < p_b;
R_2 = [R_g R_g R_g 1-R_g]';


Ps_ab = P_star(a,b);

tic
M = M_out(ra0,ra1,rb0,rb1,C);
```

```
M1_y2 = M(4,:);
M0_y2 = M(1:3,:);

try
     P1_y2 = Ps_ab(4);
catch ME
end

    P0_y2= Ps_ab(1:3);



    r0_y2 = ((P0_y2*M0_y2)*R_2)/(1 - sum(Ps_ab(4)));
    r1_y2 = M1_y2*(1-R_2);


    r_y2 = (r1_y2*pc)+((1-pc)*r0_y2);
    p_y2 = (r1_y2*pc)+((1-pc)*(1-r0_y2));
t = toc;
end
```

NOR Gate

```
function [r_y2,p_y2,r1_y2,r0_y2,t] =
NOR_gate_Matrix(p_a,p_b,pc,R_g,C,ra0,ra1,rb0,rb1)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
a = rand(1,10^6) < p_a;
b = rand(1,10^6) < p_b;
R_2 = [1-R_g R_g R_g R_g]';


Ps_ab = P_star(a,b);

tic
M = M_out(ra0,ra1,rb0,rb1,C);

M0_y2 = M(2:4,:);
M1_y2 = M(1,:);

P0_y2 = Ps_ab(2:4);
P1_y2= Ps_ab(1);

r0_y2 = ((P0_y2*M0_y2)*R_2)/(1 - sum(Ps_ab(1)));
r1_y2 = M1_y2*(1-R_2);

r_y2 = (r1_y2*pc)+((1-pc)*r0_y2);
p_y2 = (r1_y2*pc)+((1-pc)*(1-r0_y2));
t = toc;
end
```

OR Gate

```
function [r_y2,p_y2,r1_y2,r0_y2,t] =
OR_gate_Matrix(p_a,p_b,pc,R_g,C,ra0,ra1,rb0,rb1)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
a = rand(1,10^6) < p_a;
b = rand(1,10^6) < p_b;
R_2 = [1-R_g R_g R_g R_g]';


Ps_ab = P_star(a,b);
tic
M = M_out(ra0,ra1,rb0,rb1,C);

M0_y2 = M(1,:);
M1_y2 = M(2:4,:);

P0_y2 = Ps_ab(1);
P1_y2= Ps_ab(2:4);

r0_y2 = M0_y2*(1-R_2);
r1_y2 = ((P1_y2*M1_y2)*R_2)/(sum(Ps_ab(2:4)));

r_y2 = (r1_y2*pc)+((1-pc)*r0_y2);
p_y2 = (r1_y2*pc)+((1-pc)*(1-r0_y2));

t = toc;
end
```

NOT Gate

```
function [r_x1,p_x1,r1_x1,r0_x1,t] =
NOT_gate_Matrix(p_in1,rg,r_m0,r_m1)
%UNTITLED4 Summary of this function goes here
%   Detailed explanation goes here
tic
r0_x1 = (rg*r_m1) + ((1-r_m1)*(1-rg));
r1_x1 = (rg*r_m0) + ((1-r_m0)*(1-rg));

r_x1 = (r1_x1*p_in1) + ((1-p_in1)*r0_x1);
p_x1 = (r1_x1*p_in1) + ((1-p_in1)*(1-r0_x1));
t = toc;

% r1_x1 = (rg*r_m1) + ((1-r_m1)*(1-rg));
% r0_x1 = (rg*r_m0) + ((1-r_m0)*(1-rg));
%
% r_x1 = (r1_x1*p_in1) + ((1-p_in1)*r0_x1);
% p_1 = (r1_x1*p_in1) + ((1-p_in1)*(1-r0_x1));
% p_x1 = p_1;
end
```

BUFFER

```
function [r_x1,p_x1,r1_x1,r0_x1,t] =
Buffer_gate_Matrix(p_in1,rg,r_m0,r_m1)
%UNTITLED4 Summary of this function goes here
%   Detailed explanation goes here
tic
r0_x1 = (r_m0*rg) + ((1-rg)*(1-r_m0));
r1_x1 = ((1-r_m1)*(1-rg)) + (r_m1*rg);

r_x1 = (r1_x1*p_in1) + ((1-p_in1)*r0_x1);
p_x1 = (r1_x1*p_in1) + ((1-p_in1)*(1-r0_x1));

t = toc;

end
```

XOR Gate

```
function [r_y2,p_y2,r1_y2,r0_y2,t] =
XOR_gate_Matrix(p_a,p_b,pc,R_g,C,ra0,ra1,rb0,rb1)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
a = rand(1,10^6) < p_a;
b = rand(1,10^6) < p_b;
R_2 = [1-R_g R_g R_g 1-R_g]';


Ps_ab = P_star(a,b);

tic
M = M_out(ra0,ra1,rb0,rb1,C);

M0_y2 = [M(1,:);M(4,:)];
M1_y2 = [M(2,:);M(3,:)];

P0_y2 = [Ps_ab(1),Ps_ab(4)];
P1_y2= Ps_ab(2:3);

r0_y2 = ((P0_y2*M0_y2)*(1-R_2))/(1 - sum([Ps_ab(1),Ps_ab(4)]));
r1_y2 = ((P1_y2*M1_y2)*(R_2))/(sum([Ps_ab(1),Ps_ab(4)]));

r_y2 = (r1_y2*pc)+((1-pc)*r0_y2);
p_y2 = (r1_y2*pc)+((1-pc)*(1-r0_y2));

t = toc;
end
```

XNOR Gate

```
function [r_y2,p_y2,r1_y2,r0_y2,t] =
XNOR_gate_Matrix(p_a,p_b,pc,R_g,C,ra0,ra1,rb0,rb1)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
a = rand(1,10^6) < p_a;
b = rand(1,10^6) < p_b;
R_2 = [1-R_g R_g R_g 1-R_g]';


Ps_ab = P_star(a,b);
tic
M = M_out(ra0,ra1,rb0,rb1,C);

M0_y2 = [M(2,:);M(3,:)];
M1_y2 = [M(1,:);M(4,:)];

P0_y2 = [Ps_ab(2),Ps_ab(3)];
P1_y2 = [Ps_ab(1),Ps_ab(4)];

r0_y2 = ((P0_y2*M0_y2)*(R_2))/(1 - sum([Ps_ab(1),Ps_ab(4)]));
r1_y2 = ((P1_y2*M1_y2)*(1-R_2))/(sum([Ps_ab(1),Ps_ab(4)]));

r_y2 = (r1_y2*pc)+((1-pc)*r0_y2);
p_y2 = (r1_y2*pc)+((1-pc)*(1-r0_y2));

t = toc;
end
```

VITA AUCTORIS

NAME:	Khawja Zaid Sikander
PLACE OF BIRTH:	Dhaka, Bangladesh
YEAR OF BIRTH:	1997
EDUCATION:	Ahsanullah University of Science and Technology, B.Sc., Dhaka, Bangladesh, 2018
	University of Windsor, M.A.Sc., Windsor, ON, 2020