Reliability Analysis and Optimization Models for Large Scale Combinational Circuits

Suoyue Zhan
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
https://scholar.uwindsor.ca/etd/7591

This online database contains the full-text of PhD dissertations and Masters’ theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.
Reliability Analysis and Optimization Models for Large Scale Combinational Circuits

by

Suoyue Zhan

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

© 2018 Suoyue Zhan
Reliability Analysis and Optimization Models for Large Scale Combinational Circuits

by

Suoyue Zhan

APPROVED BY:

______________________________________________
X.Nie
Department of Mechanical, Automotive and Materials Engineering

______________________________________________
S.Erfani
Department of Electrical and Computer Engineering

______________________________________________
C.Chen, Advisor
Department of Electrical and Computer Engineering

September 7, 2018
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 increasingly high density, today’s integrated circuit chips become sensitive to minor effects such as temperature and environmental noises, which may lead to unreliable operation. While circuit reliability can be improved at various design levels, this usually requires additional costs (such as more area and/or more power consumption). For large-scale circuits, the first step towards reliability improvement is to do reliability estimation efficiently and accurately. This is followed by reliability optimization for given cost or budget constraints, which can be done either locally or globally.

In this thesis, we propose an asymmetrical reliability model (ARM) to do quick reliability estimation with a high level of accuracy, in comparison with true reliability values produced by Monte-Carlo simulations. Thanks to its linear-time complexity, this model can be well applied to large-scale circuits. For instance, for benchmark circuit C7552 with 3512 logic gates, it only takes a few seconds to estimate the circuit output reliability with an average error of as low as 0.5%.

The only shortcoming of ARM model is that error-free signal probabilities are assumed to be available, which may not always be the case. This motivates us to develop another method to estimate both upper and lower bounds of circuit reliability regardless of signal probabilities.

It is also found that the actual average output reliability is strongly correlated with the average upper bound of reliability. This allows us to do fast analysis on circuit reliability with different gate reliabilities, and to develop an efficient gate reliability allocation algorithm which assigns specific reliabilities to individual gates with considerations of given budget constraints. All these methods are verified through simulation results with benchmark circuits.
ACKNOWLEDGEMENTS

I would like to show my sincere appreciation to my academic supervisor Dr. Chunhong Chen for his guidance and support during my Master Program.

I would like to thank Dr. Xueyue Nie and Dr. Shervin Erfani for providing precious feedbacks.

I am grateful to Ms. Andria Turner for her administrative assistance.

Finally, I would like to thank my parents: Zheng He, Yi Zhan, and stepparents Xiaojun Zhang, Limin Jiang for their love and support on my study and life.
TABLE OF CONTENTS

DECLARATION OF ORIGINALITY ........................................................................................................ iii

ABSTRACT........................................................................................................................................ iv

ACKNOWLEDGEMENTS ................................................................................................................... v

LIST OF FIGURES ............................................................................................................................ viii

LIST OF TABLES ............................................................................................................................... ix

CHAPTER 1. INTRODUCTION ........................................................................................................ 1

1.1 Background ................................................................................................................................. 1
1.2 Previous work ............................................................................................................................ 1
1.3 Organization of Thesis ............................................................................................................... 2

CHAPTER 2. ASYMMETRICAL RELIABILITY MODEL ............................................................ 4

2.1 Reliability Asymmetry ................................................................................................................ 4
2.2 Signal Correlation ...................................................................................................................... 7
2.3 Propagation of Conditional Reliabilities .................................................................................. 9
  2.3.1 Independent Case ................................................................................................................ 9
  2.3.2 Full-Correlation Case ......................................................................................................... 10
  2.3.3 General Case .................................................................................................................... 12
  2.3.4 Other Considerations ......................................................................................................... 16
2.4 Time Complexity ...................................................................................................................... 17
2.5 Simulation Results ................................................................................................................... 17

CHAPTER 3. ESTIMATION OF RELIABILITY BOUNDS ......................................................... 20

3.1 Upper and Lower Bounds ........................................................................................................ 20
3.2 Propagation of Conditional Reliability Bounds ................................................................. 25
3.3 Algorithm for Reliability Bounds ............................................................................................ 29
3.4 Simulation Results .................................................................................................................. 29
CHAPTER 4. RELIABILITY ALLOCATION .......................................................... 33
  4.1 Budget .................................................................................................. 33
  4.2 Allocation Model .................................................................................. 34
  4.3 Simulation results ............................................................................... 38
CHAPTER 5. CONCLUSION AND FUTURE WORK ........................................... 39
  5.1 Conclusion .......................................................................................... 39
  5.2 Future work ....................................................................................... 39
APPENDIX 1. IMPORTANT CODES ............................................................... 40
  Reliability Estimation ............................................................................. 40
  Reliability Bounds .................................................................................. 49
  Reliability Allocation ............................................................................. 57
REFERENCES .............................................................................................. 63
VITA AUCTORIS ........................................................................................... 65
LIST OF FIGURES

Figure 2-1 .................................................................................................................. 8
Figure 2-2 .................................................................................................................. 9
Figure 2-3 .................................................................................................................. 11
Figure 2-4 ................................................................................................................. 16
Figure 2-5 ................................................................................................................. 17
Figure 3-1 .................................................................................................................. 23
Figure 3-2 .................................................................................................................. 24
Figure 3-3 .................................................................................................................. 25
Figure 3-4 .................................................................................................................. 33
Figure 3-5 .................................................................................................................. 34
Figure 4-1 .................................................................................................................. 37
Figure 4-2 .................................................................................................................. 38
Figure 4-3 .................................................................................................................. 39
LIST OF TABLES

Table 2-1.................................................................................................................8
Table 2-2..................................................................................................................20
Table 2-3..................................................................................................................21
Table 2-4..................................................................................................................21
Table 4-1..................................................................................................................36
Table 4-2..................................................................................................................40
CHAPTER 1. INTRODUCTION

1.1 Background

As electronic device size keeps shrinking, integrated circuits are becoming more sensitive to unexpected effects (such as temperature variation and background noises), which lead to unreliable operation. While it is generally true that the higher component reliabilities can promise more reliable circuits, this comes at exponentially increasing costs (including more area penalty and/or higher power consumption). Therefore, it is necessary to have a preview of component reliability assignment under certain cost or budget constraints, before an efficient output reliability evaluation is conducted. If the circuit could not meet the reliability requirement otherwise, a costly redesign process would be needed.

For digital combinational circuits, the only components considered are logic gates, assuming all wire connections are 100% reliable. Then the problem becomes: 1. How to do general allocation of logic gate reliabilities before the design within certain budgets. 2. How to evaluate output reliabilities after the design accurately and efficiently.

1.2 Previous work

As for reliability estimation, researchers have already done a lot of work which provide fast and approximate results, as reported in [1]. Monte-Carlo simulation is considered as a typical method for circuit performance estimation and reliability evaluation, but it is computationally expensive for large-scale circuits [2, 3]. Many investigators tried to improve algorithm efficiency, but at a cost of reduced accuracy level. Examples include observability-based method [4], probabilistic
gate model (PGM) [5], probability transfer matrix (PTM) [6], conditional probability matrix (CPM) [7], Bayesian network 8], and Boolean difference-based error calculator (BDEC) [9]. Readers are referred to [9] for a review of related work on this topic.

References [10] and [11] introduced certain algorithms on reliability bounds estimation within acceptable accuracy and tolerance. The time complexity of these algorithms, however, is not strictly linear to the number of gates. [12] presented some further discussions on the effects of input vector probabilities on circuit reliability.

Due to the fact that reliability allocation is aiming at maximizing the output reliability within certain constraints such as cost or area, [13,14,15] produced certain models for reliability optimization, taking into considerations three masking issues (i.e., logic masking, electrical masking and time masking), and provided local solutions by gate resizing and partial circuit restructuring.

### 1.3 Organization of Thesis

The organization of thesis is as follows.

Chapter 2 presents a new asymmetrical reliability model (ARM) to deal with the reliability estimation issue with high accuracy level and linearly increasing processing time proportional to circuit size, assuming the error-free nodes probability are already available. Comparison of PGM and ARM are also shown in this chapter by MATLAB simulation.

Chapter 3 focuses on further improvement of efficiency by relaxing the requirements of ARM model. Without knowing the error-free signal reliabilities, people are still able to achieve reliability bounds estimation for a quick look at circuit performance. The bounds are verified to be a true bound through simulations.
Chapter 4 is devoted to reliability allocation issues before completing the design. By taking advantage of the fact that real average output reliabilities are proportional, though not strictly, to average output upper bounds, we derive a fast, near-optimal allocation model which allows further modifications for a better performance.

Conclusions as well as future work are given in Chapter 5.
CHAPTER 2. ASYMMETRICAL RELIABILITY MODEL

2.1 Reliability Asymmetry

For any signal $s$ in a circuit, its probability is defined as $P_s = Pr\{s = "1"\}$, and its reliability $r_s$ is defined as the probability that it generates an intended logic value, i.e., $r_s = Pr\{s = s^*\}$, where $s^*$ is the error-free version of $s$. Throughout this paper, the symbol "*" is used to indicate “error-free”. If $s$ is a primary output of the circuit, then $r_s$ represents the reliability of this output signal. Similarly, for a logic gate $g$, its reliability (denoted by $r_g$) is defined as the probability that an intended output logic value is produced for given input signals.

Considering the probability of error-free signal $s^*$, we express the reliability $r_s$ as:

\[
r_s = Pr\{s = s^*\}
\]

\[
= Pr\{s = '1' \land s^* = '1'\} + Pr\{s = '0' \land s^* = '0'\}
\]

\[
= Pr\{s = '1'| s^* = '1'\} \cdot P_{s^*} + Pr\{s = '0'| s^* = '0'\} \cdot (1 - P_{s^*})
\]

\[
= k_s \cdot P_{s^*} + l_s \cdot (1 - P_{s^*}) \quad (2-1)
\]

where

\[
k_s = Pr\{s = '1'| s^* = '1'\}
\]

\[
l_s = Pr\{s = '1'| s^* = '1'\} \quad (2-2)
\]

which represent conditional reliabilities (or probabilities) for ‘1’ patterns and ‘0’ patterns, respectively. In other words, the reliability of any signal $s$ is associated with a pair of conditional reliabilities $\{k_s, l_s\}$ by (2-1). Calculation of error-free signal probability (i.e., $P_{s^*}$) has been well
documented in literature [4, 10, 11], and is beyond the scope of our discussions. For small circuits, the conditional reliabilities \( \{k_s, l_s\} \) for any signal can be found using Monte-Carlo simulations. As an example, Fig. 2-1 shows the schematic of circuit Benchmark Circuit C17 with six 2-input NAND gates, where all primary inputs (i.e., signals #1~#5) are assumed to be independent and reliable. Table 2-1 shows some of our simulation results for the conditional reliability pairs at both output signals \( \text{Out1} \) and \( \text{Out2} \) assuming different primary input probabilities (\( P_{in} \)) and gate reliability (\( r_g \)). It can be seen from the table that \( k_s \) and \( l_s \) have different values (i.e., signal reliabilities are asymmetrical in general), which vary with input signal probabilities and/or gate reliabilities. For the signal \( \text{Out2} \) with \( P_{in} = 0.9 \) and \( r_g = 0.9 \) in particular, \( k_s \) and \( l_s \) differ by more than 15%.

For large circuits with potential signal correlations, it would be impractical to find \( k_s \) and \( l_s \) directly and accurately by using an exhaustive and time-consuming approach. This is because both of those parameters depend on input signal probabilities, signal correlations, as well as gate reliabilities. However, with some approximation, the values of both \( k_s \) and \( l_s \) can be propagated gate by gate throughout the circuit, as detailed in the next section. This will make the reliability estimation very efficient with the time complexity linear to circuit size \( N \) (i.e., the number of gates). The main challenge here is, among other things, how to capture signal correlations while propagating \( k_s \) and \( l_s \), which will be the focus of our discussions that follow.
Figure 2-1. Example circuit C17.

Table 2-1. Asymmetric Conditional Reliabilities with C17

<table>
<thead>
<tr>
<th>$P_m$</th>
<th>$r_g$</th>
<th>conditional reliability pair ${k_s, l_s}$</th>
<th>$r_s = k_s P_s^* + l_s (1-P_s^*)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$Out1$</td>
<td>$Out2$</td>
</tr>
<tr>
<td>0.5</td>
<td>0.9</td>
<td>${0.780, 0.741}$</td>
<td>${0.790, 0.722}$</td>
</tr>
<tr>
<td>0.5</td>
<td>0.8</td>
<td>${0.679, 0.576}$</td>
<td>${0.662, 0.557}$</td>
</tr>
<tr>
<td>0.9</td>
<td>0.9</td>
<td>${0.830, 0.709}$</td>
<td>${0.816, 0.686}$</td>
</tr>
</tbody>
</table>

Figure 2-2. General logic gate with two correlated inputs.
2.2 Signal Correlation

Consider a two-input logic gate (with gate reliability of $r_g$) in any combinational circuit, as shown in Fig. 2-2 where $a$, $b$ and $c$ denote the two input signals and one output, respectively. They are associated with conditional reliabilities $\{k_a, l_a\}$, $\{k_b, l_b\}$ and $\{k_c, l_c\}$, respectively. For the two error-free signals $a^*$ and $b^*$ (i.e., signals $a$ and $b$ when all gate reliabilities are 1), we define an error-free probability vector $P^* = \begin{bmatrix} P_{00} \, P_{01} \\ P_{10} \, P_{11} \end{bmatrix}$ = \begin{bmatrix} Pr\{a^*b^*= 00\} \\ Pr\{a^*b^*= 01\} \\ Pr\{a^*b^*=10\} \\ Pr\{a^*b^*=11\} \end{bmatrix}$. In order to deal with signal correlations, from [16], we take signal correlation factor between $a^*$ and $b^*$ as:

$$\theta_{ij} = \frac{Pr\{a^*b^*=ij\}−Pr\{a^*=i\}Pr\{b^*=j\}}{\sqrt{Pr\{a^*=i\}Pr\{b^*=j\}(1−Pr\{a^*=i\})(1−Pr\{b^*=j\})}} \tag{2-3}$$

where $-1 \leq \theta_{ij} \leq 1$ with $i, j = 0$ or 1. The positive or negative sign of $\theta_{ij}$ represents a positive or negative signal correlation. Both $P^*$ and $\theta_{ij}$ can be found using error-free signal probabilities (i.e., $P_{a^*}$, $P_{b^*}$ and $P_{c^*}$) at both inputs and output of the gate, depending on the gate type. For instance, if the gate is a NAND gate, $P^* = [2−P_{a^*}−P_{b^*}−P_{c^*} \quad P_{b^*}+P_{c^*}−1 \quad P_{a^*}+P_{c^*}−1 \quad 1−P_{c^*}]$, and $\theta_{aa} = \theta_{ab} = \theta_{ac} = -\theta_{ba} = (1−P_{a^*}−P_{b^*}−P_{c^*})/\sqrt{P_{a^*}(1−P_{a^*})P_{b^*}(1−P_{b^*})}$. Again, take C17 of Fig. 1 for example. Assuming primary input probabilities of 0.5, we found $P_{a^*} = P_{b^*} = 0.750$, $P_{b^*} = P_{g^*} = 0.625$ and $P_{c^*} = P_{out1^*} = 0.563$ for signals #6 and #8 (i.e., two inputs of NAND gate G5 in Fig. 2-1), and thus $P^* = [0.063 \ 0.188 \ 0.312 \ 0.437]$ and $|\theta_{ij}| \approx 0.148$. For input signals #8 and #9 of NAND gate G6 with stronger correlation, it is found that $P_{a^*} = P_{g^*} = 0.625$, $P_{b^*} = P_{g^*} = 0.625$ and $P_{c^*} = P_{out2^*} = 0.563$, which lead to $P^* = [0.188 \ 0.188 \ 0.187 \ 0.437]$ and $|\theta_{ij}| \approx 0.2$. Generally speaking, $0 \leq |\theta_{ij}| \leq 1$, and the signal correlation gets stronger as the value of $|\theta_{ij}|$ increases. However, special attention shall be given to two extreme cases where $\theta_{ij} = 0$ or $|\theta_{ij}| = 1$. The former case happens if both $a^*$ and $b^*$ are independent, and the latter implies the full correlation between $a^*$ and $b^*$, as shown in Fig. 2-3(a).
and (b), respectively, where $r_{g1}$ and $r_{g2}$ denote the reliability of two buffers.

Figure 2-3. Logic gate with independent inputs (a) and full-correlation inputs (b).
2.3 Propagation of Conditional Reliabilities

Propagation of conditional reliability pairs for the gate of Fig. 2-2 can be done by finding \( k_c, l_c \) for given \( k_a, l_a \) and \( k_b, l_b \). If this is done, all conditional reliabilities at this circuit’s outputs can be found by repeating the propagation process for all \( N \) gates in a topological order. The output reliabilities are finally given by (2-1). In the following sections, we first look at the two extreme cases in Fig. 2-3, and then extend our results to the general case of Fig. 2-2.

2.3.1 Independent Case

Before discussing conditional reliabilities, it would be necessary to begin with a specific conditional probability \( P_0^0 = Pr\{ab = ij | a^* b^* = 00\} \), where \( i, j = 0 \) or 1. For the independent case of Fig. 2-3 (a) with \( \theta_{ij} = 0 \), we have

\[
p_{00}^{ij} = \begin{cases} 
Pr\{a = 0 | a^* = 0\} \cdot Pr\{b = 0 | b^* = 0\} = l_a l_b, & \text{if } i = j = 0 \\
l_a(1 - l_b), & \text{if } i = 0, j = 1 \\
l_b(1 - l_a), & \text{if } i = 1, j = 0 \\
1 - l_a - l_b + l_a l_b, & \text{if } i = j = 1 
\end{cases}
\]  

(2-4)

Similarly, we can find \( P_{01}^{ij} = Pr\{ab = ij | a^* b^* = 01\} \), \( P_{10}^{ij} = Pr\{ab = ij | a^* b^* = 10\} \), or \( P_{11}^{ij} = Pr\{ab = ij | a^* b^* = 11\} \) with the above \( l_a \) and \( l_b \) in (2-4) being replaced by \( l_a \) and \( k_b \), \( k_a \) and \( l_b \), or \( k_a \) and \( k_b \), respectively. For instance, \( P_{01}^{ij} \) is expressed as

\[
p_{01}^{ij} = \begin{cases} 
Pr\{a = 0 | a^* = 0\} \cdot Pr\{b = 1 | b^* = 1\} = l_a k_b, & \text{if } i = 0, j = 1 \\
l_a(1 - k_b), & \text{if } i = j = 0 \\
(1 - l_a)(1 - k_b), & \text{if } i = 1, j = 0 \\
(1 - l_a)k_b, & \text{if } i = j = 1 
\end{cases}
\]  

(2-5)

In other words, we can obtain a \( 4 \times 4 \) conditional probability matrix (for input signals \( a \) and \( b \)) as follows:
\[
M = \begin{bmatrix}
P_{00} & P_{01} & P_{10} & P_{11} \\
00 & 01 & 10 & 11 \\
00 & 01 & 10 & 11 \\
00 & 01 & 10 & 11 \\
00 & 01 & 10 & 11 \\
\end{bmatrix} = \begin{bmatrix}
M_{00} \\
M_{01} \\
M_{10} \\
M_{11} \\
\end{bmatrix}
\]

with \(\sum_{i,j} P_{ij} = \sum_{i,j} P_{ij} = \sum_{i,j} P_{ij} = 1\). With the availability of \(M\) and \(P^*\), \(k_c\) and \(l_c\) in Fig. 3 (a) can be found analytically, depending on the gate type. For example, if the logic gate in Fig. 3 (a) is a NAND gate, then we have

\[
k_c = Pr(c = 1' | c^* = 1') = [P_{00}^* P_{01}^* P_{10}^*] \cdot \begin{bmatrix}
M_{00} \\
M_{01} \\
M_{10} \\
\end{bmatrix} \cdot \begin{bmatrix}
1 - r_g \\
r_g \\
\end{bmatrix} / (1 - P_{11})
\]

\[
l_c = Pr(c = 0' | c^* = 0') = M_{11} \cdot [1 - r_g 1 1 - r_g r_g] \\
\]

For any other gates, the conditional reliability pair \(\{k_c, l_c\}\) can be found in a similar way.

### 2.3.2 Full-Correlation Case

For the full-correlation case (\(|\theta| = 1\)) of Fig. 3 (b) where \(\{k, l\}\) is the conditional reliability pair of buffer’s input, we have

\[
l_a = l \cdot r_g + (1 - l)(1 - r_g) \\
l_b = l \cdot r_g + (1 - l)(1 - r_g)
\]

Assuming \(r_{g1}\) and \(r_{g2}\) are independent, the conditional probability \(P_{00}^a\) is given by

\[
P_{00}^a = l \cdot r_{g1} \cdot r_{g2} + (1 - l)(1 - r_{g1})(1 - r_{g2})
\]

Since both gate reliabilities of \(r_{g1}\) and \(r_{g2}\) are generally very close to 1, combination of (2-8) and
(2-9) gives

\[ p_{00} = \min\{l_a \cdot r_{g2} - \Delta_a^{00}, l_b \cdot r_{g1} - \Delta_b^{00}\} \]  

(2-10)

where

\[ \Delta_a^{00} = (1 - l_a)(1 - r_{g1})(2r_{g2} - 1) \]
\[ \Delta_b^{00} = (1 - l_b)(1 - r_{g2})(2r_{g1} - 1) \]  

(2-11)

The conditional probability \( p_{ij}^{00} \) is expressed as

\[
p_{ij}^{00} = \begin{cases} 
  p_{00}^{00}, & \text{if } i = j = 0 \\
  l_a - p_{00}^{00}, & \text{if } i = 0, j = 1 \\
  l_b - p_{00}^{00}, & \text{if } i = 1, j = 0 \\
  1 - l_a - l_b + p_{00}^{00}, & \text{if } i = j = 1 
\end{cases}
\]  

(2-12)

where \( p_{00}^{00} \) is given by (2-10). It can be proved that the value of \( p_{00}^{00} \) is greater than or equal to \( l_a \cdot l_b \), but less than or equal to either \( l_a \) or \( l_b \). This ensures that \( 0 \leq p_{00}^{00} \leq 1 \) in (2-12). Also, if the values of \( r_{g1}, r_{g2}, l_a, l_b, k_a \) and \( k_b \) are all between 0.5 and 1, which is the case in general, then \( p_{00}^{ij} \leq p_{00}^{00} \) for \( i, j = 0 \) or 1. Similarly, we can find \( p_{01}^{ij}, p_{10}^{ij}, \) or \( p_{11}^{ij} \) with the above \( l_a \) and \( l_b \) in (2-10), (2-11) and (2-12) being replaced by \( l_a \) and \( k_b, k_a \) and \( l_b, k_a \) and \( k_b \), respectively. For instance, \( p_{01}^{01} \) is expressed as

\[
p_{01}^{01} = \begin{cases} 
  l_a - p_{01}^{01}, & \text{if } i = j = 0 \\
  p_{01}^{01}, & \text{if } i = 0, j = 1 \\
  1 - l_a - k_b + p_{01}^{01}, & \text{if } i = 1, j = 0 \\
  k_b - p_{01}^{01}, & \text{if } i = j = 1 
\end{cases}
\]  

(2-13)

where \( p_{01}^{01} \) is given by
\[ p_{01}^{01} \approx \min \{ l_a \cdot r_{g2} - \Delta_a^{01}, l_b \cdot r_{g1} - \Delta_b^{01} \} \]  
(2-14)

and

\[
\begin{align*}
\Delta_a^{01} &= (1 - l_a)(1 - r_{g1})(2r_{g2} - 1) \\
\Delta_b^{01} &= (1 - k_b)(1 - r_{g2})(2r_{g1} - 1)
\end{align*}
\]  
(2-15)

and \(0 \leq p_{01}^{ij} \leq 1\). By comparing (2-4)~(2-5) and (2-10)~(2-15), one can see that when it comes to propagation of conditional reliabilities, the only difference between Fig. 3 (a) and (b) is the way to calculate the matrix \(M\) in (2-6). Eq. (2-7) for calculating \(\{k_c, l_c\}\) always works regardless of signal correlations.

There are two other full-correlation cases which are worth mentioning, as shown in Fig. 2-4. For the case of Fig. 4 (a) where the inputs \(a\) and \(b\) are connected to a same signal, we have: \(r_{g1} = r_{g2} = 1\), \(l_a = l_b\) and \(k_a = k_b\). Thus, \(p_{00}^{00} = l_a\) and \(p_{11}^{11} = k_a\). However, both \(p_{01}^{01}\) and \(p_{10}^{10}\) are immaterial because \(P_{01}^* = P_{10}^* = 0\) in this case. On the other hand, for the case of Fig. 4 (b) where \(r_{g2} = 1\), we have \(p_{01}^{01} \approx \min \{ l_a - (1-l_a)(1-r_{g1}), k_b \cdot r_{g1} \} \) and \(p_{10}^{10} \approx \min \{ k_a - (1-k_a)(1-r_{g1}), l_b \cdot r_{g1} \}, \) while both \(p_{00}^{ij}\) and \(p_{11}^{ij}\) are immaterial in this particular case due to \(P_{00}^* = P_{11}^* = 0\). Both cases of Fig. 4 rarely appear in a circuit implementation. However, even if they do, they can be identified during the reliability propagation with proper setting for \(r_{g1}\) and/or \(r_{g2}\), as mentioned above.

2.3.3 General Case

The general case of signal correlations is illustrated in Fig. 2-2, where the correlation factor \(\theta_{ij}\) of (2-3) stays somewhere in between the above extreme cases (i.e., \(0 < |\theta_{ij}| < 1\)). The key issue in finding all elements in \(M\) of (2-6) for this case is to calculate its diagonal elements (i.e., \(p_{ii}^\#, i, j = 0\) or 1). Unfortunately, an exact value of \(p_{ii}^\#\) is unknown due to complex signal correlations.
However, based on the above discussions with independent and full-correlation cases, $P_{ij}$ for the general case can be approximated as follows:

\[
\begin{align*}
P_{00}^0 &= l_a \cdot l_b + \theta_{00}^2 \cdot (\min \{l_a \cdot r_{g2} - \Delta_a^{00}, l_b \cdot r_{g1} - \Delta_b^{00}\} - l_a l_b) \\
P_{01}^0 &= l_a \cdot k_b + \theta_{01}^2 \cdot (\min \{l_a \cdot r_{g2} - \Delta_a^{01}, k_b \cdot r_{g1} - \Delta_b^{01}\} - l_a k_b) \\
P_{10}^0 &= k_a \cdot l_b + \theta_{10}^2 \cdot (\min \{k_a \cdot r_{g2} - \Delta_a^{10}, l_b \cdot r_{g1} - \Delta_b^{10}\} - k_a l_b) \\
P_{11}^0 &= k_a \cdot k_b + \theta_{11}^2 \cdot (\min \{k_a \cdot r_{g2} - \Delta_a^{11}, k_b \cdot r_{g1} - \Delta_b^{11}\} - k_a k_b)
\end{align*}
\]

where $\Delta_a^{00} = \Delta_a^{01}, \Delta_a^{10} = \Delta_a^{11}, \Delta_b^{00} = \Delta_b^{10},$ and $\Delta_b^{01} = \Delta_b^{11}$. The $P_{ij}$ for the independent case (refer to (2-4) and (2-5)) or full-correlation case (refer to (2-10) and (2-14)) is a special case of (2-16) when $|\theta_{ij}| = 0$ or 1. The positive or negative signal correlation (i.e., the sign of $\theta_{ij}$) is taken into account by using the conditional reliability pairs of \{k_a, l_a\} and \{k_b, l_b\}.

![Logic gate diagram](image-url)
Figure 2-4. Two special cases with full-correlation signals.

Figure 2-5 (a) $r_{g^2} = 1$. 
Once $P_{ij}$ is available with $i, j = 0$ or $1$, all off-diagonal elements in $M$ of (2-6) can be obtained with help of $\{k_a, l_a\}$ and $\{k_b, l_b\}$ (for example, using (2-12) and (2-13) for $P_{00}^{a}$ and $P_{01}^{a}$, respectively). With the availability of $M, P^*$ and gate type, the conditional reliability pair $\{k_c, l_c\}$ at the gate output can be found (refer to (2-7) for instance). Results reported in Section 4 show that the accuracy level of (2-16) is very high with an average error of typically 2% in estimating the circuit reliability, depending on specific circuits and gate reliabilities.

It should be noted that in addition to considerations for the two cases of Fig. 2-4, (2-16) shall be modified under other two cases: i) If one of the two inputs $a$ (or $b$) in Fig. 2-2 is a primary input (refer to Fig. 2-5 (a) where $b$ is a primary input), one shall let $r_g = 1$ (or $r_{g2} = 1$) in (2-16) with the assumption that all primary inputs are reliable; ii) If the two signals $a$ and $b$ happen to be an input and output of a same gate, one shall also let $r_{g1} = 1$ as shown in Fig. 2-5 (b), where the signal $a$ is an input of gate $G$ while the signal $b$ is the output of gate $G$. In case both signals $a$ and $b$ in Fig. 2-
2 are primary inputs, we have $|\theta_{ij}| = 0$ for $i, j = 0$ or 1 (assuming all primary inputs are independent), and thus (2-16) gives a same value of $p^t_q$ regardless of $r_{g1}$ and $r_{g2}$.

2.3.4 Other Considerations

In the above analysis, we only deal with 2-input gates. For an inverter with input $a$ and output $c$, the conditional reliabilities at the output are simply expressed as

$$
\begin{align*}
k_c &= r_g \cdot l_a + \left(1 - r_g\right)\left(1 - l_a\right) \\
l_c &= r_g \cdot k_a + \left(1 - r_g\right)\left(1 - k_a\right)
\end{align*}
$$

(2-17)

where $r_g$ is the gate reliability and $\{k_a, l_a\}$ is the conditional reliability pair at the input. For a logic gate with $q$ inputs ($q > 2$), the conditional probability matrix $M$ will be in size of $2^q \times 2^q$, and involves the signal correlations among $q$ inputs. While our model could be extended theoretically to this case, the propagation of conditional reliabilities would become much more complicated. A quick solution instead is to decompose the gate into a few two-input gates (e.g., decomposition of 3-input AND gate to two 2-input gates). Fortunately, majority of gates in real-world circuits have no more than two inputs, and most logic synthesis tools also provide an option of doing the decomposition with 2-input gates only. Further discussions on handling multi-input gates are beyond the scope of this work.

It should also be mentioned that when analyzing circuit reliability, one needs to consider electrical masking, temporal masking and logic masking in general. Electrical masking shall be included in evaluating gate reliability $r_g$ which is assumed to be available in this work, and temporal masking shall be considered in sequential circuits which are not discussed here. The focus of this work is to deal with logic masking, signal correlations as well as their roles in determining the reliability of large combinational circuits. In particular, while the conditional reliabilities ($k_c$ or $l_c$) at the
output of a gate are always less than or equal to the gate reliability \(r_g\), they could be greater than the conditional reliabilities \((k_a, l_a, k_b \text{ or } l_b)\) at inputs of the gate due to logic masking and/or input signal correlations. In other words, the output signal reliability for a gate could be higher than its input reliabilities if \(r_g\) is relatively large. Therefore, when signal reliabilities propagate through the whole circuit, they may not necessarily diminish, or at least not do as quickly as one might think.

### 2.4 Time Complexity

The asymmetric reliability model (ARM) presented in the previous section allows us to derive any logic gate’s output reliability directly from the conditional reliabilities at its inputs. This makes the circuit reliability analysis significantly fast with \(O(N)\) time complexity (assuming the availability of error-free signal probabilities), where \(N\) is the total number of gates in the circuit. This analysis efficiency is important not only for large circuits, but also for repeated reliability evaluations with different gate reliabilities which are required for reliability improvement/optimization. The model is also able to take signal correlations into account without the need for exhaustively exploring the potential impacts of all transitive fan-ins on the circuit output reliability. This is possible mainly by introducing the asymmetrical reliability pair \((k, l)\) as well as using the approximation in (2-16).

### 2.5 Simulation Results

Table 2-2 and Table 2-3 show the detailed results for every single output regarding to benchmark circuit C17. The Monte-Carlo results are calculated with \(10^7\) iterations, which is considered as accurate value. One can see that for small circuits, the performance of ARM is quite acceptable. In general, the input vector probability doesn’t have much effect on the performance, while a
higher gate reliability will provide a more accurate result. Therefore, in the following simulations, we simply put input vector to be $P_{in} = 0.5$

**Table 2-2. Simulation results on output reliabilities for C17 with $r_g = 0.99$ and different values of $P_{in}$**

<table>
<thead>
<tr>
<th>Output</th>
<th>$P_{in} = 0.1$</th>
<th>$P_{in} = 0.5$</th>
<th>$P_{in} = 0.9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal</td>
<td>ARM</td>
<td>MC</td>
<td>ARM</td>
</tr>
<tr>
<td>Out1</td>
<td>0.971</td>
<td>0.971</td>
<td>0.973</td>
</tr>
<tr>
<td>Out2</td>
<td>0.971</td>
<td>0.971</td>
<td>0.973</td>
</tr>
<tr>
<td>Average error</td>
<td>0.0%</td>
<td>0.1%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

**Table 2-3. Simulation results on output reliabilities for C17 with $P_{in} = 0.5$ and different values of $r_g$**

<table>
<thead>
<tr>
<th>Output</th>
<th>$r_g = 0.9$</th>
<th>$r_g = 0.99$</th>
<th>$r_g = 0.999$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal</td>
<td>ARM</td>
<td>MC</td>
<td>ARM</td>
</tr>
<tr>
<td>Out1</td>
<td>0.775</td>
<td>0.775</td>
<td>0.973</td>
</tr>
<tr>
<td>Out2</td>
<td>0.772</td>
<td>0.760</td>
<td>0.973</td>
</tr>
<tr>
<td>Average error</td>
<td>0.6%</td>
<td>0.1%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>
Table 2-4. Comparison of Average Estimation Errors (%) for Reliability Analysis on Benchmark Circuits ($P_{in} = 0.5$)

<table>
<thead>
<tr>
<th>Circuit</th>
<th># Gates</th>
<th># PIs</th>
<th># POs</th>
<th>Average Percentage Errors (%)</th>
<th>Average Percentage Errors (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$r_g = 0.99$</td>
<td>$r_g = 0.999$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PGM</td>
<td>ARM</td>
</tr>
<tr>
<td>C432</td>
<td>160</td>
<td>36</td>
<td>7</td>
<td>34.8</td>
<td>0.7</td>
</tr>
<tr>
<td>C499</td>
<td>202</td>
<td>41</td>
<td>32</td>
<td>44.5</td>
<td>0.1</td>
</tr>
<tr>
<td>C880</td>
<td>383</td>
<td>60</td>
<td>26</td>
<td>35.3</td>
<td>0.8</td>
</tr>
<tr>
<td>C1355</td>
<td>546</td>
<td>41</td>
<td>32</td>
<td>39.0</td>
<td>5.9</td>
</tr>
<tr>
<td>C1908</td>
<td>880</td>
<td>36</td>
<td>25</td>
<td>35.4</td>
<td>8.4</td>
</tr>
<tr>
<td>C2670</td>
<td>1193</td>
<td>233</td>
<td>140</td>
<td>43.4</td>
<td>1.1</td>
</tr>
<tr>
<td>C3540</td>
<td>1669</td>
<td>50</td>
<td>22</td>
<td>24.2</td>
<td>6.7</td>
</tr>
<tr>
<td>C5315</td>
<td>2829</td>
<td>178</td>
<td>123</td>
<td>36.8</td>
<td>1.8</td>
</tr>
<tr>
<td>C7552</td>
<td>4042</td>
<td>207</td>
<td>108</td>
<td>30.9</td>
<td>2.4</td>
</tr>
<tr>
<td>Average</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td><strong>36.0</strong></td>
<td><strong>3.1</strong></td>
</tr>
</tbody>
</table>

The overall error comparing to MC value is far less than PGM. The reason is that PGM provide very high error when output reliability is approaching 0.5, especially the case in large circuit such as Benchmark Circuits C5315 and C7552. Also, though not listed, the CPU time running ARM process is linearly proportional to circuit size, which could be taken from the algorithm itself.
CHAPTER 3. ESTIMATION OF
RELIABILITY BOUNDS

3.1 Upper and Lower Bounds

Consider a generic logic gate with output c and two inputs a and b, as shown in Fig. 3-1 where \( r_g \) is the gate reliability, and \( \{k_a, l_a\}, \{k_b, l_b\} \) and \( \{k_c, l_c\} \) represent the conditional reliability pair for signals a, b and c, respectively. We define a conditional probability for the two inputs (i.e., a and b) as follows:

\[
P_{ij}^{uv} = \Pr\{ 'ab' = 'uv' \mid 'a^*b^*' = 'ij' \}
\]

(3-1)

where \( i, j, u, v = '0' \) or \('1'\), and \( a^* \) and \( b^* \) are an error-free version of a and b, respectively. This conditional probability can be expressed as

\[
P_{ij}^{uv} = \begin{cases} 
   p_{ij}, & \text{if } u = i \text{ and } v = j \\
   q_a - p_{ij}, & \text{if } u = i \text{ and } v \neq j \\
   q_b - p_{ij}, & \text{if } u \neq i \text{ and } v = j \\
   1 - q_a - q_b + p_{ij}, & \text{if } u \neq i \text{ and } v \neq j
\end{cases}
\]

(3-2)

where

\[
q_a = \begin{cases} 
   l_a, & \text{if } i = '0' \\
   k_a, & \text{if } i = '1'
\end{cases} \quad \text{and} \quad q_b = \begin{cases} 
   l_b, & \text{if } j = '0' \\
   k_b, & \text{if } j = '1'
\end{cases}
\]

(3-3)

with \( p_{ij}^{00} + p_{ij}^{01} + p_{ij}^{10} + p_{ij}^{11} = 1 \).

Finding the exact value of \( p_{ij}^{ij} \) in (2-4) could be difficult due to possible complex signal (reliability)
correlations between $a$ and $b$. However, its bounds can be estimated by considering the following two extreme cases: (1) signals $a$ and $b$ are fully-independent, and (2) they are fully-correlated. Let us take $P_{00}^{00}$ for example. Under this independent case, we have $P_{00}^{00} = l_a l_b$. The value of $P_{00}^{00}$ increases as signals $a$ and $b$ get more correlated and reaches its maximum when they are fully-correlated. A general case of full-correlation is illustrated in Fig. 3-2 where both $a$ and $b$ are driven by a buffer with reliability of $r_{g1}$ and $r_{g2}$, respectively. We have

\[ \{k_a, l_a\} \]
\[ a \]

\[ \{k_b, l_b\} \]
\[ b \]

\[ \text{logic gate with } r_g \]

\[ \{k_c, l_c\} \]
\[ c \]

\[ \{k, l\} \]
\[ \text{logic gate with } r_g \]

\[ \{k, l\} \]
\[ \{k, l\} \]

Figure 3-1. A generic 2-input logic gate.

Figure 3-2. A general case of full correlation between two signals $a$ and $b$. 
\[
\begin{align*}
    l_a &= l \cdot r_{g1} + (1 - l)(1 - r_{g1}) \\
    l_b &= l \cdot r_{g2} + (1 - l)(1 - r_{g2})
\end{align*}
\]  
(3-4)

where \(0 \leq l \leq 1\) (refer to Fig. 3-2). Since \(r_{g1}\) and \(r_{g2}\) are independent, the conditional probability \(P_{00}^{00}\) is given by

\[
P_{00}^{00} = l \cdot r_{g1} \cdot r_{g2} + (1 - l)(1 - r_{g1})(1 - r_{g2})
\]  
(3-5)

Combination of (3-4) and (3-5) gives

\[
P_{00}^{00} = l_a \cdot r_{g2} - \frac{r_{g1} - l_a}{2r_{g1} - 1}(1 - r_{g1})(2r_{g2} - 1),
\]

or

\[
l_b \cdot r_{g1} - \frac{r_{g2} - l_b}{2r_{g2} - 1}(1 - r_{g2})(2r_{g1} - 1).
\]  
(3-6)

Since the typical value of \(r_{g1}\) or \(r_{g2}\) is close to 1, the second term in (3-6) would be negligibly small.

Thus, we have

\[
P_{00}^{00} \leq l_a \cdot r_{g2}, \quad \text{or} \quad P_{00}^{00} \leq l_b \cdot r_{g1}.
\]  
(3-7)

In order to ensure \(0 \leq P_{00}^{uv} \leq 1\), where \(P_{00}^{uv}\) is given by (3-2) with \(u, v = 0\) or \(1\), the value of \(P_{00}^{00}\) is no greater than \(\min\{l_a \cdot r_{g2}, l_b \cdot r_{g1}\}\). In other words, \(P_{00}^{00}\) is bounded as:

\[
l_a \cdot l_b \leq P_{00}^{00} \leq \min\{l_a \cdot r_{g2}, l_b \cdot r_{g1}\}
\]  
(3-8)

where \(l_a \leq r_{g1}\) and \(l_b \leq r_{g2}\), as can be seen from (3-1).

Similarly, the bounds of \(P_{ij}^{ij}\) for any logic value of \(i\) and \(j\) can be derived, and is expressed generally as

\[
q_a \cdot q_b \leq P_{ij}^{ij} \leq \min\{q_a \cdot r_{g2}, q_b \cdot r_{g1}\}
\]  
(3-9)
where \( q_a \) and \( q_b \) are given by (3-3) (i.e., \( q_a \) equals to either \( l_a \) or \( k_a \), and \( q_b \) equals to either \( l_b \) or \( k_b \), depending on the value of \( i \) and \( j \)). It should also be noted in (3-10) that \( q_a \leq r_{g1} \) and \( q_b \leq r_{g2} \).

As shown in Figure 3-2, \( r_{g1} \) and \( r_{g2} \) in (3-9) generally represent the reliability of gates driving signals \( a \) and \( b \), respectively. However, some modifications are needed under a few special cases of correlation between \( a \) and \( b \), which are illustrated in Figure 3-3. First, if \( b \) is an input of the gate driving \( a \), or vice versa (refer to Figure 3-3 (a) and (b)), we shall set \( r_{g2} = 1 \) or \( r_{g1} = 1 \) in (3-9).

Secondly, if \( a \) and \( b \) are a same signal as shown in Figure 3-3 (c),

![Figure 3-3 (a) r_{g2} = 1](image)

![Figure 3-3 (b) r_{g1} = 1](image)
Figure. 3-3 (c) \( r_{g1} = r_{g2} = 1 \)

Figure. 3-3 (d) \( r_{gI} = 1 \)

Figure. 3-3. Considerations of some special cases for equation (3-9).

we shall set \( r_{g1} = r_{g2} = 1 \) instead. Finally, if \( a \) and/or \( b \) is a primary input, then \( r_{g1} \) and/or \( r_{g2} \) would be unavailable because they have no driving gates. In this work, we assume that all primary inputs are independent and reliable, and thus set \( r_{g1} = q_a = 1 \) and/or \( r_{g2} = q_b = 1 \) in (3-6) for this particular case. Since either \( a \) or \( b \) is reliable, this case is equivalent to an independent case for which both lower and upper bounds in (3-9) would become equal. Figure. 3-3 (d) shows an example where \( a \) is a primary input (PI) with \( r_{gI} = 1 \).
3.2 Propagation of Conditional Reliability Bounds

The above equations (3-1), (3-2), (3-3) and (3-9) describe the general relationship between the conditional probability $P_{ij}^{uv}$ and conditional reliabilities $q_a$ and $q_b$ for the two inputs $a$ and $b$ of a generic logic gate shown in Figure. 3-3. In this section, we show how conditional reliabilities propagate from the two inputs to the output for different logic gates. Particularly, we are interested in propagation of conditional reliability bounds using (3-9). We will begin with two-input AND and XOR logic gates, and then extend the results to other types of gates.

(i) AND Gate: Assume the joint probability of error-free inputs (i.e., $a^*$ and $b^*$) is $P_{ij}^* = \Pr \{a^*b^*\} = ij$, where $i, j = ‘0’$ or ‘1’. If the logic gate of Figure. 3-1 is an AND gate with reliability of $r_g$, the conditional reliability pair $\{k_c, l_c\}$ for the output $c$ is expressed as

\[
k_c = P_{11}^{11} \cdot r_g + (1 - P_{11}^{11})(1 - r_g)
\]

\[
= (2r_g - 1)P_{11}^{11} + (1 - r_g)
\]

and

\[
l_c = \sum_{ij=00,01,10} [P_{ij}^*(1 - P_{ij}^{11})r_g + P_{ij}^{11}(1 - r_g)]/(1 - P_{11}^{*1})
\]

\[
= \sum_{ij=00,01,10}P_{ij}^*[r_g + (1 - 2r_g)P_{ij}^{11}]/(1 - P_{11}^{*1}).
\]

According to (3-9), we have $P_{11}^{11} \leq \min\{k_a \cdot r_g2, k_b \cdot r_g1\}$. Since $0.5 < r_g \leq 1$ in general, the upper bound of $k_c$ in (3-10) is given by

\[
k_c^U (AND) = (2r_g - 1)\min\{k_a^U \cdot r_g2, k_b^U \cdot r_g1\} + (1 - r_g)
\]

(3-12)
where \( k_a^u \geq k_a \) and \( k_b^u \geq k_b \) represent the upper bounds of \( k_a \) and \( k_b \), respectively. Also from (3-9), the lower bound of \( P_{11}^{11} \) is \( k_a \cdot k_b \), and thus the lower bound of \( k_c \) in (3-10) is given by

\[
k_c^L(\text{AND}) = (2r_g - 1)k_a^L \cdot k_b^L + (1 - r_g)
\]  

(3-13)

where \( k_a^L \leq k_a \) and \( k_b^L \leq k_b \) represent the lower bounds of \( k_a \) and \( k_b \), respectively.

According to (3-2), (3-11) can be rewritten as

\[
l_c = \{P_{00}^*r_g + (1 - 2r_g)(1 - l_a - l_b + P_{00}^{00})\} + \{P_{01}^*r_g + (1 - 2r_g)(k_b - P_{01}^{01})\}
\]

\[
+ P_{10}^*r_g + (1 - 2r_g)(k_a - P_{10}^{10})\}/(1 - P_{11}^*)
\]  

(3-14)

Since \( P_{00}^* + P_{01}^* + P_{10}^* = 1 - P_{11}^* \), we have

\[
l_c \leq r_g + (1 - 2r_g) \cdot P_{\text{min}}
\]  

(3-15)

where

\[
P_{\text{min}} = \min\{1 - l_a - l_b + P_{00}^{00}, k_b - P_{01}^{01}, k_a - P_{10}^{10}\}.
\]  

(3-16)

By using (3-9) again, we have

\[
P_{\text{min}} \geq \min\{(1 - l_a^u)(1 - l_b^u), k_b^L - \min\{l_a^u \cdot r_{g2}, k_b^L \cdot r_{g1}\}, k_a^L - \min\{k_a^L \cdot r_{g2}, l_b^u \cdot r_{g1}\}\}
\]  

(3-17)

where \( l_a^u \geq l_a \) and \( l_b^u \geq l_b \) represent the upper bounds of \( l_a \) and \( l_b \), respectively. Combining (3-15) and (3-17) gives the upper bound of \( l_c \) as:

\[
l_c^U(\text{AND}) = r_g + (1 - 2r_g) \min\{(1 - l_a^u)(1 - l_b^u), k_b^L - \min\{l_a^u \cdot r_{g2}, k_b^L \cdot r_{g1}\}, k_a^L - \min\{k_a^L \cdot r_{g2}, l_b^u \cdot r_{g1}\}\}
\]  

(3-18)

Similarly, the lower bound of \( l_c \) in (3-14) can be derived as

\[
l_c^L(\text{AND}) = r_g + (1 - 2r_g) \cdot P_{\text{max}}
\]  

(3-19)

where
\[ P_{\text{max}} = \max\{1 - l_{a}^{l} - l_{b}^{l} + \min\{l_{a}^{u} \cdot r_{g2} + l_{b}^{u} \cdot r_{g1}, k_{b}^{u}(1 - l_{a}^{u}), k_{a}^{u}(1 - l_{b}^{u})\} \} \]  

(3-20)

and \( l_{a}^{l} \leq l_{a} \) and \( l_{b}^{l} \leq l_{b} \) represent the lower bounds of \( l_{a} \) and \( l_{b} \), respectively.

(ii) XOR Gate: If the logic gate of Fig. 3-1 is an XOR gate, the conditional reliability pair \( \{k_{c}, l_{c}\} \) for the output \( c \) is expressed as

\[ k_{c} = \sum_{ij=01,10}(P_{ij}^{a}[r_{g} + (1 - 2r_{g})(P_{ij}^{00} + P_{ij}^{11})]}/(P_{01}^{a} + P_{10}^{a}) \]  

(3-21)

and

\[ l_{c} = \sum_{ij=00,11}(P_{ij}^{a}[r_{g} + (1 - 2r_{g})(P_{ij}^{01} + P_{ij}^{10})]}/(P_{00}^{a} + P_{11}^{a}) \]  

(3-22)

in comparison with (3-10) and (3-21) for AND gate. According to (3-2), (3-21) is bounded by

\[ k_{c} \leq r_{g} + (1 - 2r_{g}) \cdot \min\{l_{a} + k_{b} - 2P_{01}^{01}, k_{a} + l_{b} - 2P_{10}^{10}\} \]  

(3-23)

and

\[ k_{c} \geq r_{g} + (1 - 2r_{g}) \cdot \max\{l_{a} + k_{b} - 2P_{01}^{01}, k_{a} + l_{b} - 2P_{10}^{10}\} \]  

(3-24)

Similarly, (3-22) is bounded by

\[ l_{c} \leq r_{g} + (1 - 2r_{g}) \cdot \min\{l_{a} + l_{b} - 2P_{00}^{00}, k_{a} + k_{b} - 2P_{11}^{11}\} \]  

(3-25)

and

\[ l_{c} \geq r_{g} + (1 - 2r_{g}) \cdot \max\{l_{a} + l_{b} - 2P_{00}^{00}, k_{a} + k_{b} - 2P_{11}^{11}\} \]  

(3-26)

By applying (2-11) to (3-23)~(3-26), we derive the upper and lower bounds of both \( k_{c} \) and \( l_{c} \) for XOR gate (without proof) as follows:

\[ k_{c}^{u}(\text{XOR}) = r_{g} + (1 - 2r_{g}) \min\{\max\{k_{b}^{u} + (1 - 2r_{g2}) \cdot l_{a}^{u}, l_{a}^{l} + (1 - 2r_{g1}) \cdot k_{b}^{u}, 0\}, \] 

\[ \max\{ l_{b}^{u} + (1 - 2r_{g2}) \cdot k_{a}^{u}, k_{a}^{l} + (1 - 2r_{g1}) \cdot l_{b}^{u}, 0\}\} \]  

(3-27)

\[ k_{c}^{l}(\text{XOR}) = r_{g} + (1 - 2r_{g}) \max\{\min\{k_{b}^{l} \cdot (1 - 2l_{a}^{u}) + l_{b}^{l} \cdot (1 - 2l_{a}^{u}) + l_{a}^{u}, \] 

\[ \max\{l_{b}^{l} \cdot (1 - 2l_{a}^{u}) + k_{a}^{l} \cdot (1 - 2l_{a}^{u}) + k_{b}^{u}, l_{b}^{u} \cdot (1 - 2l_{a}^{u}) + k_{b}^{u}, 1\}, \] 

\[ \min\{\max\{l_{b}^{u} \cdot (1 - 2l_{a}^{u}) + k_{a}^{u} \cdot (1 - 2l_{a}^{u}) + k_{b}^{u}, l_{b}^{u} \cdot (1 - 2l_{a}^{u}) + l_{a}^{u}, 1\}\} \]  

(3-28)
\[ l_c^U (XOR) = r_g + (1 - 2r_g) \min \{ \max \{ l_b^U + (1 - 2r_g) \cdot t_a^U, l_a^U + (1 - 2r_g) \cdot t_b^U \}, 0 \}, \]

\[ \max \{ k_b^U + (1 - 2r_g) \cdot k_a^U, k_b^U + (1 - 2r_g) \cdot k_b^U, 0 \} \] (3-29)

\[ l_c^L (XOR) = r_g + (1 - 2r_g) \max \{ \min \{ l_b^U \cdot (1 - 2l_a^U) + l_b^U, l_b^U \cdot (1 - 2l_a^U) + l_b^U \}, \max \{ k_a^U \cdot (1 - 2k_b^U) + k_a^U, k_a^U \cdot (1 - 2k_b^U) + k_a^U \}, \min \{ k_b^U \cdot (1 - 2k_a^U) + k_b^U, k_b^U \cdot (1 - 2k_a^U) + k_b^U \}, \max \{ l_a^U \cdot (1 - 2l_b^U) + l_b^U, l_a^U \cdot (1 - 2l_b^U) + l_b^U \}, 1 \} \] (3-30)

(iii) Extension to Other Gates: If the logic gate is NAND gate, the conditional reliability bounds at its output can be obtained by switching \( k_c \) with \( l_c \) in the above equations (3-12), (3-13), (3-18) and (3-24) derived for AND gate. For NOR gate, one can instead switch \( k_a \) and \( k_b \) with \( l_a \) and \( l_b \), respectively, in equations (3-12), (3-13), (3-18) and (3-19). For OR gate, simply switch \( k_c \) with \( l_c \) in the equations obtained for NOR gate. The conditional reliability bounds for XNOR are found by switching \( k_c \) with \( l_c \) in the above equations (3-27)~(3-30) derived for XOR gate. For an inverter with input signal \( a \), propagation of conditional reliability bounds is done by simply using

\[
\begin{align*}
k_c^U (INV) &= (1 - r_g) + (2r_g - 1) \cdot l_a^U \\
l_c^U (INV) &= (1 - r_g) + (2r_g - 1) \cdot k_a^U \\
k_c^L (INV) &= (1 - r_g) + (2r_g - 1) \cdot l_a^U \\
l_c^L (INV) &= (1 - r_g) + (2r_g - 1) \cdot k_a^U
\end{align*}
\] (3-31)

where \( r_g \) is the reliability of the inverter.
3.3 Algorithm for Reliability Bounds

It can be seen from the above discussions that the key idea in propagating conditional reliability bounds is to use the bounds of $P_{ij}$ in (3-9) by considering only two extreme cases: independent case and full-correlation case. This ensures that no conditional reliabilities at the output of a logic gate would go beyond their lower and upper bounds regardless of input vectors (or the specific value of $P_{ij}^*$), eliminating the need for an exhaustive search for worst-case and/or best-case input vectors. Once the propagation of bounds for both $k$ and $l$ is done, the upper and lower bounds of the reliability at any primary output $F$ are given by:

$$
\begin{align*}
    r^U_F &= \max\{k^U_F, l^U_F\} \\
    r^L_F &= \max\{k^L_F, l^L_F\}
\end{align*}
$$

(3-32)

Therefore, the whole computation process is very efficient with the time complexity of $O(N)$, where $N$ is the number of logic gates in the circuit.

3.4 Simulation Results.

To perform the detailed results of upper and lower bound for every single output, Benchmark Circuit C432 would be a good example where there are 7 outputs in total.
Figure 3-4 Reliability distribution of different input vectors and Reliability Bounds when all gate reliabilities are 0.8
Figure 3-5 (g)

Figure 3-5 Reliability distribution of different input vectors and Reliability Bounds when all gate reliabilities are 0.99

The black and white triangles represent the lower and upper bound for outputs, respectively. And the bars represent the distribution of specific output reliability under randomly generated input vector probabilities, in percentage, which is coming from MC simulation.

From these results, one can firstly tell that our reliability bounds are true bound for all outputs and secondly, the bound is tight enough for most outputs. With a higher standard $r_g=0.99$, the overall distribution is approaching upper bound comparing to $r_g=0.8$. 
CHAPTER 4. RELIABILITY

ALLOCATION

4.1 Budget

Theoretically, gate reliabilities could be any value no larger than 1. However, in practical cases, the cost, including area and power consumption, is increasing exponentially to the gate reliability, and reaches infinity when $r_g=1$. Without losing generality, we define the cost using the following equation:

$$C = erf^{-1}(r_g)$$

(4-1)

where $C$ is the cost and $r_g$ is the reliability of a certain gate.

Here

$$erf^{-1}(z) = \sum_{k=0}^{\infty} \frac{c_k}{2k+1} \left( \frac{\sqrt{\pi}}{2} z \right)^{2k+1}$$

and

$$c_k = \sum_{m=0}^{k-1} \frac{c_m c_{k-1-m}}{(m+1)(2m+1)}.$$

The total budget of a given circuit is submission of all gate costs:

$$B = \sum_{i=1}^{N} C_i$$

(4-2)

where $N$ is the total number of gates included in the circuit.

The reason why we choose $erf^{-1}$ function is that when reliability is approaching 1, the cost should be infinity in real world. It should be noted that this function could vary regarding to specific case, relating to technic, material or some other factors.
4.2 Allocation Model

The reliability allocation is looking for an optimized assignment of gate reliabilities within certain budget to generate the maximum average output reliability of a specific circuit. The necessity of this behavior could be proved by the following table:

**Table 4-1 Output Reliability of Different Allocation on Benchmark Circuit C17**

<table>
<thead>
<tr>
<th>C17 Reliability</th>
<th>Allocation 1</th>
<th>Allocation 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output1</td>
<td>0.635</td>
<td>0.592</td>
</tr>
<tr>
<td>Output2</td>
<td>0.656</td>
<td>0.584</td>
</tr>
</tbody>
</table>

Where allocation 1 is given as [0.7,0.7,0.8,0.8,0.9,0.9] on reliability from G1-G6 in order, and allocation 2 as [0.9,0.9,0.8,0.8,0.7,0.7].

To do allocation appropriately, we firstly divide all gates into different levels, which is decided by the length of the shortest path from gate to the closest output. For instance, gates directly connected to the output are treated as level 1, and those whose outputs are inputs of level-1 gates would be defined as Level 2, and so on.

Based on the level division, we assign the gate reliabilities by through two parameters: \( \alpha \) and \( r_{L_1} \), where \( \alpha \) is the decrement factor, within [0.5,1.5] and \( r_{L_1} \) is the reliability for Level 1 gates, within [0.5,1]. The reliability of Level \( i \) gates are calculated by \( r_{L_1}/\alpha^i \). Then (4-2) could be rewritten as:

\[
B = \sum_{i=1}^{C_L} N_i \text{erf}^{-1}\left(\frac{r_{L_i}}{\alpha^i}\right)
\]  

(4-3)

Where \( N_i \) represent the number of gates on level \( i \).

Our simulation shows that upper bound has exact the same trends with real average reliability over all outputs. Figure 4-1 shows the average output reliability upper bounds for different \( \alpha \) when
given same budget $B$.

![Figure 4-1. Average output Reliability for different $\alpha$ with same budget](image)

From the simulation, we can conclude that an $\alpha > 1$ is necessary during the allocation procedure for better performance at the output side. In following optimization process, we set $\alpha$ to be within range $[1, 1.5]$.

Assume the average output reliability is a quasi-quadratic function of $\alpha$ and $r_{L_1}$ as follows:

$$R_{out} = f(\alpha, r_{L_1}) = A\alpha^2 + B\alpha + C r_{L_1}^2 + D r_{L_1} + E \alpha r_{L_1} + F.$$  \hfill (4-4)

By randomly generating 1000 pairs of $(\alpha, r_{L_1})$ values without considering about budget limit first, and $R_{out}$ is coming from MC simulation, all the other parameters could be derived through regression analysis.
Budget constraints will be applied after the quadratic function regression analysis. The optimization problem becomes: what is the maximum value of (4-4) with subject to:

\[
\begin{cases}
B \geq \sum_{i=1}^{CL} erf^{-1}\left(\frac{r_{L1}}{\alpha}\right) \\
\alpha \geq 1 \\
1 \geq r_{L1} \geq 0.5
\end{cases}
\]  

(4-5)

Once optimized \((\alpha, r_{L1})\) are reached, we can reach the actual average output reliability by applying Monte-Carlo simulation. The comparison of proposed allocation and random allocations with same fixed budget are:
Figure 4-3 shows that the random generated reliability allocation, in most cases, is producing a worse average reliability than proposed \((\alpha, r_{L_1})\) allocation with the same given budget. Due to the two assumptions we made here: 1. the general gate reliability distribution is increasing by a certain factor \(\alpha\), and 2. all gates on the same level has the same reliability assigned, there is no guarantee that the \((\alpha, r_{L_1})\) model is the best. What makes this allocation significant is its efficiency and.

Without considering signal correlations, the time complexity of proposed method is linearly proportional to circuit size. Comparing to local adjustments, which focus only on specific gate or part of entire circuit, there is no need to do detailed analysis of gate importance, which is the most time-consuming part. Based on this global allocation, a further detailed allocation tuning method, which is called fine-tuning, could also be applied for further performance improvement.
By differ those gates in same level by the number of output they produce, a slightly increasement could be applied to those more important ones at the expense of decrement on less important ones.

### 4.3 Simulation results

The performance of proposed method is shown in Table 4-2.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>#Gates</th>
<th>#Inputs</th>
<th>#Outputs</th>
<th>Cover Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C17</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>97%</td>
</tr>
<tr>
<td>C432</td>
<td>160</td>
<td>36</td>
<td>7</td>
<td>98%</td>
</tr>
<tr>
<td>C1908</td>
<td>425</td>
<td>33</td>
<td>25</td>
<td>95%</td>
</tr>
<tr>
<td>C3540</td>
<td>901</td>
<td>50</td>
<td>22</td>
<td>98%</td>
</tr>
<tr>
<td>C7552</td>
<td>2171</td>
<td>207</td>
<td>108</td>
<td>96%</td>
</tr>
</tbody>
</table>

The cover rate represents the percentage of allocation results which are better than the maximum value of random generated results under same budget constrains. It is important to mention that, when the average gate reliability is above 0.9, our model usually provides better cover rate, which means it is applicable in real designs with the reality that all gate reliabilities are around 0.99 or higher.
CHAPTER 5. CONCLUSION AND FUTURE WORK

5.1 Conclusion

Generally, the proposed ARM model can provide better results than PGM when MC is considered as the correct value in terms of both accuracy level and CPU processing time. However, it also requires certain pre-calculations to provide the error-free probabilities $P^*$ of all signals inside the circuit, which is the most time-consuming part of the entire algorithm. This is where upper bound and lower bound come in as a quick estimation of target circuit reliabilities. Considering the fact that the upper bound is proportional to the real average reliability, we have proposed an approximate model for a near-optimal gate reliability allocation subject to a given budget, which could help designers to do quick global assignment before applying partial analysis of the circuit, improving the performance significantly. All methods mentioned above have linear time complexity to the number of gates in circuits, which makes our ARM model accurate, fast, and practical.

5.2 Future work

Currently, the ARM only applies to combinational circuits. Sequential circuits could be the next move. Also, during our analysis, we assume all gate reliabilities are constants. However, the gate reliability could have certain variations due to different input patterns and aging issues. Therefore, a dynamic reliability model should be considered to make our work more practical.
APPENDIX 1. IMPORTANT CODES

Reliability Estimation

clear;
clc;
Num_input=5;
Num_node=23;
Num_gate=6;
Num_output=2;
f=fopen('C17.txt');

%Num_input=1;
%Num_node=4;
%Num_gate=3;
%Num_output=1;
%f=fopen('BuffAND.txt');

%Num_input=4;
%Num_node=7;
%Num_gate=3;
%Num_output=1;
%f=fopen('C3.txt');

%Num_input=36;
%Num_node=432;
%Num_gate=160;
%Num_output=7;
%f=fopen('C432t.txt');

%Num_input=41;
%Num_node=755;
%Num_gate=202;
%Num_output=32;
%f=fopen('C499.txt');

%Num_input=60;
%Num_node=932;
%Num_gate=383;
%Num_output=26;
%f=fopen('C880.txt');

%Num_input=41;
%Num_node=1399;
%Num_gate=546;
%Num_output=32;
%f=fopen('C1355.txt');

%Num_input=33;
%Num_node=2899;
%Num_gate=880;
%Num_output=25;
f=fopen('C1908.txt');

%Num_input=233;
%Num_node=11704;
%Num_gate=1400;
%Num_output=140;
f=fopen('C2670_2in.txt');

%Num_input=178;
%Num_node=9035;
%Num_gate=2830;
%Num_output=123;
f=fopen('C5315_2in.txt');

%Num_input=50;
%Num_output=22;
%Num_gate=1983;
%Num_node=8654;
f=fopen('C3540_2in.txt');

%Num_input=32;
%Num_output=32;
%Num_gate=2416;
%Num_node=6288;
f=fopen('C6288_2in.txt');

%Num_input=128;
%Num_node=86000;
%Num_gate=84535;
%Num_output=128;
f=fopen('Divisor.txt');

%Num_input=207;
%Num_output=108;
%Num_gate=4042;
%Num_node=12192;
f=fopen('C7552_2in.txt');

%Num_input=32;
%Num_output=32;
%Num_gate=45083;
%Num_node=45157;
f=fopen('log_verilog.txt');

MC=10^5;
T1;
P_in=0.5;
rg=0.99;
kiv=1;
liv=1;
k=100*ones(Num_node,1);
l=100*ones(Num_node,1);
ks=100*ones(Num_node,1);
ls=100*ones(Num_node,1);
r=100*ones(Num_node,1);
rs=100*ones(Num_node,1);
ks_counter=zeros(Num_node,1);
ls_counter=zeros(Num_node,1);
rs_counter=zeros(Num_node,1);
rs_counter(Input)=MC;
P_=zeros(Num_node,1);
P_counter=zeros(Num_node,1);
theta11=100*ones(Num_gate,1);
counter0=zeros(Num_node,1);
counter0_=zeros(Num_node,1);
counter1=zeros(Num_node,1);
counter1_=zeros(Num_node,1);
errk=zeros(Num_node,1);
errl=zeros(Num_node,1);
errr=zeros(Num_node,1);
r_counter=zeros(Num_node,1);
counter00=0;
cz11=0;
Countert=0;
Countert2=0;
rtest=0;
cta=0;
ctb=0;
mod=0;
\%Circuit level
\%CL=Circuit_Level(Gates,Num_gate,Num_node);
e=2;
tic;
for i=1:Num_input
    k(Input(i))=kiv;
    l(Input(i))=liv;
    ks(Input(i))=kiv;
    ls(Input(i))=liv;
end
\%P*
tic;
for i=1:MC
    NodevecP=zeros(Num_node,1);
    for j=1:Num_input
        if (rand(1)<P_in)
            NodevecP(Input(j),1)=1;
        else
            NodevecP(Input(j),1)=0;
        end
        P_(Input(j),1)=P_in;
    end
    for j=1:Num_gate
        x=Gates(j,:);
        switch x(5)
        case {1}
            NodevecP(x(1),1)=1-NodevecP(x(2),1)*NodevecP(x(3),1);
        case {2}
            NodevecP(x(1),1)=NodevecP(x(2),1)*NodevecP(x(3),1);
        case {3}
            NodevecP(x(1),1)=1-or(NodevecP(x(2),1),NodevecP(x(3),1));
        case {4}
            NodevecP(x(1),1)=or(NodevecP(x(2),1),NodevecP(x(3),1));
        end
Nodevec_(x_(1),1)=Nodevec_(x_(2),1)*(1-
Nodevec_(x_(3),1))+Nodevec_(x_(3),1)*(1-Nodevec_(x_(2),1));
end
if rand(1)<rg
switch x_(5)
case {1}
  Nodevec(x_(1),1)=1-Nodevec(x_(2),1)*Nodevec(x_(3),1);
case {2}
  Nodevec(x_(1),1)=Nodevec(x_(2),1)*Nodevec(x_(3),1);
case {3}
  Nodevec(x_(1),1)=1-or(Nodevec(x_(2),1),Nodevec(x_(3),1));
case {4}
  Nodevec(x_(1),1)=or(Nodevec(x_(2),1),Nodevec(x_(3),1));
case {5}
  Nodevec(x_(1),1)=1-Nodevec(x_(2),1);
case {6}
  Nodevec(x_(1),1)=Nodevec(x_(2),1);
case {7}
  Nodevec(x_(1),1)=Nodevec(x_(2),1)*(1-
Nodevec(x_(3),1))+(1-
Nodevec(x_(3),1))*(1-Nodevec(x_(2),1));
%case {8}
  %Nodevec(x_(1),1)=xor(N0devec(x_(2),1),Nodevec(x_(3),1);
end
else
switch x_(5)
case {1}
  Nodevec(x_(1),1)=Nodevec(x_(2),1)*Nodevec(x_(3),1);
case {2}
  Nodevec(x_(1),1)=1-Nodevec(x_(2),1)*Nodevec(x_(3),1);
case {3}
  Nodevec(x_(1),1)=or(Nodevec(x_(2),1),Nodevec(x_(3),1));
case {4}
  Nodevec(x_(1),1)=1-or(Nodevec(x_(2),1),Nodevec(x_(3),1));
case {5}
  Nodevec(x_(1),1)=Nodevec(x_(2),1);
case {6}
  Nodevec(x_(1),1)=1-Nodevec(x_(2),1);
case {7}
  Nodevec(x_(1),1)=Nodevec(x_(2),1)*Nodevec(x_(3),1)+(1-
Nodevec(x_(3),1))*(1-Nodevec(x_(2),1));
%case {8}
  %Nodevec(x_(1),1)=xor(N0devec(x_(2),1),Nodevec(x_(3),1);
end
end
if (Nodevec(x_(1),1)==1)
counter1_ (x_(1),1)=counter1_ (x_(1),1)+1;
if (Nodevec(x_(1),1)==1)
  ks_counter(x_(1),1)=ks_counter(x_(1),1)+1;
  rs_counter(x_(1),1)=rs_counter(x_(1),1)+1;
end
elseif Nodevec(x_(1),1)==0
  counter0_ (x_(1),1)=counter0_ (x_(1),1)+1;
if Nodevec(x_(1),1)==0
  ls_counter(x_(1),1)=ls_counter(x_(1),1)+1;
  rs_counter(x_(1),1)=rs_counter(x_(1),1)+1;
end
end
if Nodevec_(16)==0&&Nodevec_(19)==0
counter00=counter00+1;
if Nodevec(16)==0&&Nodevec(19)==0
cz11=cz11+1;
end
end
z11=cz11/counter00;
if Nodevec(23,1)==Nodevec_(23,1)
 r_counter=r_counter+1;
end
for jr=1:100
for j=1:Num_gate
 xt=Gates(j,:);
 if rand(1)<rg
 %Nodevec(xt(1),1)=1-Nodevec(xt(2),1)*Nodevec(xt(3),1);
 else
 %Nodevec(xt(1),1)=Nodevec(xt(2),1)*Nodevec(xt(3),1);
 %end
 if (Nodevec_(xt(1),1)==0)&&(Nodevec(xt(1),1)==0)
 %ls_counter(xt(1),1)=ls_counter(xt(1),1)+1;
 elseif (Nodevec_(xt(1),1)==1)&&(Nodevec(xt(1),1)==1)
 %ks_counter(xt(1),1)=ks_counter(xt(1),1)+1;
 %end
 %end
 %end
end
for i=1:Num_node
 if counter1_(i)~=0
 ks(i)=ks_counter(i)/counter1_(i);
 end
 if counter0_(i)~=0
 ls(i)=ls_counter(i)/counter0_(i);
 end
end
theta=((1-P_(23))-P_(16)*P_(19))/sqrt(P_(16)*(1-P_(16))*P_(19)*(1-P_(19)));
delta00=min(ls(16)*rg,ls(19)*rg);
rs=rs_counter/MC;
T_MC=toc;
%k,l estimation
tic
xp=Gates(:,1:3);
for i=1:Num_gate
 x=Gates(i,:);
 conA=isempty(find(Input==x(2)));
 conB=isempty(find(Input==x(3)));
 conC=isempty(find(xp(find(xp(:,1)==x(2)),2:3)==x(3)));
 conD=isempty(find(xp(find(xp(:,1)==x(3)),2:3)==x(2)));
 rg1=rg;
 rg2=rg;
 if conA&&conB&&conC&&conD
 switch x(5)
 case {1}

 [k(x(1),1),l(x(1),1),theta11(i,1)]=NANDKL(k(x(2),1),l(x(2),1),P_(x(2),1),k(x(3),1),l(x(3),1),P_(x(3),1),P_(x(1),1),rg,e,mod);
 case [2]
[k(x(1),1),l(x(1),1),theta11(i,1)]=ANDKL(k(x(2),1),l(x(2),1),P_(x(2),1),k(x(3)
),1),l(x(3),1),P_(x(3),1),P_(x(1),1),rg,e,mod);  
    case [3]

[k(x(1),1),l(x(1),1),theta11(i,1)]=NORKL(k(x(2),1),l(x(2),1),P_(x(2),1),k(x(3)
),1),l(x(3),1),P_(x(3),1),P_(x(1),1),rg,e,mod);  
    case [4]

[k(x(1),1),l(x(1),1),theta11(i,1)]=ORKL(k(x(2),1),l(x(2),1),P_(x(2),1),k(x(3)
),1),l(x(3),1),P_(x(3),1),P_(x(1),1),rg,e,mod);  
    case [5]

[k(x(1),1),l(x(1),1)]=NOTKL(k(x(2),1),l(x(2),1),rg);  
    case [6]

[k(x(1),1),l(x(1),1)]=BUFFKL(k(x(2),1),l(x(2),1),rg);  
    case [7]

%case [8]

end

elseif ~conC&&conD
    rg1=1;
    switch x(5)
    case [1]

[k(x(1),1),l(x(1),1),theta11(i,1)]=NANDKL_I(k(x(2),1),l(x(2),1),P_(x(2),1),k(x(3)
),1),l(x(3),1),P_(x(3),1),P_(x(1),1),rg,rg1,rg2,mod);  
    case [2]

[k(x(1),1),l(x(1),1),theta11(i,1)]=ANDKL_I(k(x(2),1),l(x(2),1),P_(x(2),1),k(x(3)
),1),l(x(3),1),P_(x(3),1),P_(x(1),1),rg,rg1,rg2,mod);  
    case [3]

[k(x(1),1),l(x(1),1),theta11(i,1)]=NORKL_I(k(x(2),1),l(x(2),1),P_(x(2),1),k(x(3)
),1),l(x(3),1),P_(x(3),1),P_(x(1),1),rg,rg1,rg2,mod);  
    case [4]

[k(x(1),1),l(x(1),1),theta11(i,1)]=ORKL_I(k(x(2),1),l(x(2),1),P_(x(2),1),k(x(3)
),1),l(x(3),1),P_(x(3),1),P_(x(1),1),rg,rg1,rg2,mod);  
    case [5]

[k(x(1),1),l(x(1),1)]=NOTKL(k(x(2),1),l(x(2),1),rg);  
    case [6]

[k(x(1),1),l(x(1),1)]=BUFFKL(k(x(2),1),l(x(2),1),rg);  
    case [7]

%case [8]

end

elseif ~conD&&conC
    rg2=1;
    switch x(5)
case {1}
[k(x(1),1),l(x(1),1),theta1(i,1)]=NANDKL_I(k(x(2),1),l(x(2),1),P_(x(2),1),k(x(3),1),l(x(3),1),P_(x(3),1),P_(x(1),1),rg,rg1,rg2,mod);
case [2]
[k(x(1),1),l(x(1),1),theta1(i,1)]=ANDKL_I(k(x(2),1),l(x(2),1),P_(x(2),1),k(x(3),1),l(x(3),1),P_(x(3),1),P_(x(1),1),rg,rg1,rg2,mod);
case [3]
[k(x(1),1),l(x(1),1),theta1(i,1)]=NORKL_I(k(x(2),1),l(x(2),1),P_(x(2),1),k(x(3),1),l(x(3),1),P_(x(3),1),P_(x(1),1),rg,rg1,rg2,mod);
case [4]
[k(x(1),1),l(x(1),1),theta1(i,1)]=ORKL_I(k(x(2),1),l(x(2),1),P_(x(2),1),k(x(3),1),l(x(3),1),P_(x(3),1),P_(x(1),1),rg,rg1,rg2,mod);
case [5]
[k(x(1),1),l(x(1),1)]=NOTKL(k(x(2),1),l(x(2),1),rg);
case [6]
[k(x(1),1),l(x(1),1)]=BUFFKL(k(x(2),1),l(x(2),1),rg);
case [7]
end

elseif ~conA&&~conB
rg1=1;
rg2=1;
switch x(5)
    case [1]
[k(x(1),1),l(x(1),1),theta1(i,1)]=NANDKL_I(k(x(2),1),l(x(2),1),P_(x(2),1),k(x(3),1),l(x(3),1),P_(x(3),1),P_(x(1),1),rg,rg1,rg2,mod);
case [2]
[k(x(1),1),l(x(1),1),theta1(i,1)]=ANDKL_I(k(x(2),1),l(x(2),1),P_(x(2),1),k(x(3),1),l(x(3),1),P_(x(3),1),P_(x(1),1),rg,rg1,rg2,mod);
case [3]
[k(x(1),1),l(x(1),1),theta1(i,1)]=NORKL_I(k(x(2),1),l(x(2),1),P_(x(2),1),k(x(3),1),l(x(3),1),P_(x(3),1),P_(x(1),1),rg,rg1,rg2,mod);
case [4]
[k(x(1),1),l(x(1),1),theta1(i,1)]=ORKL_I(k(x(2),1),l(x(2),1),P_(x(2),1),k(x(3),1),l(x(3),1),P_(x(3),1),P_(x(1),1),rg,rg1,rg2,mod);
case [5]
[k(x(1),1),l(x(1),1)]=NOTKL(k(x(2),1),l(x(2),1),rg);
case [6]
[k(x(1),1),l(x(1),1)]=BUFFKL(k(x(2),1),l(x(2),1),rg);
case [7]
end
% [k(x(1),1), l(x(1),1)] = XNORKL(k(x(2),1), l(x(2),1), P_(x(2),1),
k(x(3),1), l(x(3),1), P_(x(3),1), P_(x(1),1), rg);
end

elseif ~conA && conB
rg1=1;
switch x(5)
case [1]
[k(x(1),1), l(x(1),1), theta{i,1}] = NANDKL_I(k(x(2),1), l(x(2),1), P_(x(2),1),
k(x(3),1), l(x(3),1), P_(x(3),1), P_(x(1),1), rg, rg1, rg2, mod);
case [2]
[k(x(1),1), l(x(1),1), theta{i,1}] = ANDKL_I(k(x(2),1), l(x(2),1), P_(x(2),1), k(x(3),1), l(x(3),1), P_(x(3),1), P_(x(1),1), rg, rg1, rg2, mod);
case [3]
[k(x(1),1), l(x(1),1), theta{i,1}] = NORKL_I(k(x(2),1), l(x(2),1), P_(x(2),1),
k(x(3),1), l(x(3),1), P_(x(3),1), P_(x(1),1), rg, rg1, rg2, mod);
case [4]
[k(x(1),1), l(x(1),1), theta{i,1}] = ORKL_I(k(x(2),1), l(x(2),1), P_(x(2),1), k(x(3),1), l(x(3),1), P_(x(3),1), P_(x(1),1), rg, rg1, rg2, mod);
case [5]
[k(x(1),1), l(x(1),1)] = NOTKL(k(x(2),1), l(x(2),1), rg);
case [6]
[k(x(1),1), l(x(1),1)] = BUFFKL(k(x(2),1), l(x(2),1), rg);
case [7]
[k(x(1),1), l(x(1),1), theta{i,1}] = XORKL_I(k(x(2),1), l(x(2),1), P_(x(2),1),
k(x(3),1), l(x(3),1), P_(x(3),1), P_(x(1),1), rg, rg1, rg2, mod);
case [8]
end

elseif ~conB && conA
rg2=1;
switch x(5)
case [1]
[k(x(1),1), l(x(1),1), theta{i,1}] = NANDKL_I(k(x(2),1), l(x(2),1), P_(x(2),1),
k(x(3),1), l(x(3),1), P_(x(3),1), P_(x(1),1), rg, rg1, rg2, mod);
case [2]
[k(x(1),1), l(x(1),1), theta{i,1}] = ANDKL_I(k(x(2),1), l(x(2),1), P_(x(2),1), k(x(3),1), l(x(3),1), P_(x(3),1), P_(x(1),1), rg, rg1, rg2, mod);
case [3]
[k(x(1),1), l(x(1),1), theta{i,1}] = NORKL_I(k(x(2),1), l(x(2),1), P_(x(2),1),
k(x(3),1), l(x(3),1), P_(x(3),1), P_(x(1),1), rg, rg1, rg2, mod);
case [4]
[k(x(1),1), l(x(1),1), theta{i,1}] = ORKL_I(k(x(2),1), l(x(2),1), P_(x(2),1), k(x(3),1), l(x(3),1), P_(x(3),1), P_(x(1),1), rg, rg1, rg2, mod);
case [5]
[k(x(1),1), l(x(1),1)] = NOTKL(k(x(2),1), l(x(2),1), rg);
case [6]
[k(x(1),1), l(x(1),1)] = BUFFKL(k(x(2),1), l(x(2),1), rg);
case {7}

[k(x(1),1),l(x(1),1),theta11(i,1)]=XORKL_I(k(x(2),1),l(x(2),1),P_(x(2),1),k(x(3),1),l(x(3),1),P_(x(3),1),P_(x(1),1),rg,rg1,rg2,mod);
%case {8}

%[k(x(1),1),l(x(1),1)]=XNORKL(k(x(2),1),l(x(2),1),P_(x(2),1),k(x(3),1),l(x(3),1),P_(x(3),1),P_(x(1),1),rg);
end
end

if k(x(1))<0.5||l(x(1))<0.5
  %fprintf('k or l is small than 0.5\n');
ei=i;
%break;
elseif k(x(1))>1||l(x(1))>1
  fprintf('k or l is larger than 1\n');
ei=i;
%break;
elseif isnan(k(x(1)))||isnan(l(x(1)))
  fprintf('k or l NAN\n');
ei=i;
break;
end
end

%T_Z=toc;
for i=1:Num_node
  %rs(i,1)=ks(i,1)*P_(i,1)+ls(i,1)*(1-P_(i,1));
  %r(i,1)=k(i,1)*P_(i,1)+l(i,1)*(1-P_(i,1));
  errk(i,1)=100*(ks(i,1)-k(i,1))/ks(i,1);
  errl(i,1)=100*(ls(i,1)-l(i,1))/ls(i,1);
  errr(i,1)=100*(rs(i,1)-r(i,1))/rs(i,1);
end

raver=sum(abs(errr(Output)))/Num_output
rmax=max(abs(errr(Output)))
errr(Output);
PQ=ks.*P_+(1-ls).*(1-P_);
rPGM=PQ.*P_+(1-PQ).*(1-P_);
errPGM=100*(rs-rPGM)/rs;
raverPGM=sum(abs(errPGM(Output)))/Num_output
rmaxPGM=max(abs(errPGM(Output)))
rmaxMC=max(rs(Output))
rmminMC=min(rs(Output))

Reliability Bounds

T_l;
k_i=1;
l_i=1;
k_up=ones(Num_node,1);
l_up=ones(Num_node,1);
k_low=ones(Num_node,1);
l_low=ones(Num_node,1);
r_up=ones(Num_node,1);
r_low=ones(Num_node,1);
ks = 100 * ones(Num_node, 1);
ls = 100 * ones(Num_node, 1);
r = 100 * ones(Num_node, 1);
rs = 100 * ones(Num_node, 1);
ks_counter = zeros(Num_node, 1);
ls_counter = zeros(Num_node, 1);
rs_counter = zeros(Num_node, 1);
rs_counter(Input) = MC;
r_mul_counter = 0;
r_pro = 1;
P = zeros(Num_node, 1);
P_counter = zeros(Num_node, 1);
theta11 = 100 * ones(Num_gate, 1);
thetaAND = zeros(Num_output * (Num_output - 1) / 2, 1);
thetaOut = zeros(Num_output, 1);
counter0 = zeros(Num_node, 1);
counter0_ = zeros(Num_node, 1);
counter1 = zeros(Num_node, 1);
counter1_ = zeros(Num_node, 1);
errk = zeros(Num_node, 1);
errl = zeros(Num_node, 1);
errr = zeros(Num_node, 1);
r_counter = zeros(Num_node, 1);
cz11 = 0;
Countert = 0;
Countert2 = 0;
rtest = 0;
cta = 0;
cb = 0;
mod = 0;

% Circuit level
% Cl = Circuit_Level(Gates, Num_gate, Num_node);
e = 2;

% separate gates matrix into two
v_norm = find(Gates(:, 1) == Output(size(Output, 1)), 1);
Num_gate_norm = v_norm;
Num_gate_AND = v_norm + Num_output * (Num_output - 1) / 2;
Gates_norm = Gates(1:v_norm, :);
Gates_out = Gates(v_norm + 1:Num_gate, :);
tic;
for i = 1:Num_input
    k(Input(i)) = kiv; l(Input(i)) = liv; ks(Input(i)) = kiv; ls(Input(i)) = liv;
end

% P*
tic;
for i = 1:MC
    NodevecP = zeros(Num_node, 1);
    for j = 1:Num_input
        if (rand(1) < P_in)
            NodevecP(Input(j), 1) = 1;
        else
            NodevecP(Input(j), 1) = 0;
        end
        P_(Input(j), 1) = P_in;
    end
    for j = 1:Num_gate
x=Gates(j,:);
switch x(5)
    case {1}
        NodevecP(x(1),1)=1-NodevecP(x(2),1)*NodevecP(x(3),1);
    case {2}
        NodevecP(x(1),1)=NodevecP(x(2),1)*NodevecP(x(3),1);
    case {3}
        NodevecP(x(1),1)=1-or (NodevecP(x(2),1),NodevecP(x(3),1));
    case {4}
        NodevecP(x(1),1)=or (NodevecP(x(2),1),NodevecP(x(3),1));
    case {5}
        NodevecP(x(1),1)=1-NodevecP(x(2),1);
    case {6}
        NodevecP(x(1),1)=NodevecP(x(2),1);
    case {7}
        NodevecP(x(1),1)=xor (NodevecP(x(2),1),NodevecP(x(3),1));
end
if (NodevecP(x(1),1)==1)
    P_counter(x(1),1)=P_counter(x(1),1)+1;
end
end
P_=P_counter/MC+P_;
T_P_=toc;
% k,l  simulation
tic;
for i=1:MC
    Nodevec=zeros(Num_node,1);
    for ci=1:Num_input
        if (rand(1)<P_in)
            Nodevec(Input(ci),1)=1;
        else
            Nodevec(Input(ci),1)=0;
        end
    end
    Nodevec_=Nodevec;
    for j_=1:Num_gate
        x_=Gates(j_,:);
        switch x_(5)
        case {1}
            Nodevec_ (x_(1),1)=1-Nodevec_ (x_(2),1)*Nodevec_ (x_(3),1);
        case {2}
            Nodevec_ (x_(1),1)=Nodevec_ (x_(2),1)*Nodevec_ (x_(3),1);
        case {3}
            Nodevec_ (x_(1),1)=1-or (Nodevec_ (x_(2),1),Nodevec_ (x_(3),1));
        case {4}
            Nodevec_ (x_(1),1)=or (Nodevec_ (x_(2),1),Nodevec_ (x_(3),1));
        case {5}
            Nodevec_ (x_(1),1)=1-Nodevec_ (x_(2),1);
        case {6}
            Nodevec_ (x_(1),1)=Nodevec_ (x_(2),1);
        case {7}
            Nodevec_ (x_(1),1)=Nodevec_ (x_(2),1)*Nodevec_ (x_(3),1)+(1-Nodevec_ (x_(3),1))* (1-Nodevec_ (x_(2),1));
        end
        if rand(1)<rg
            switch x_(5)
case {1}
    Nodevec(x_{1},1)=1-Nodevec(x_{2},1)*Nodevec(x_{3},1);
case {2}
    Nodevec(x_{1},1)=Nodevec(x_{2},1)*Nodevec(x_{3},1);
case {3}
    Nodevec(x_{1},1)=1-or(Nodevec(x_{2},1),Nodevec(x_{3},1));
case {4}
    Nodevec(x_{1},1)=or(Nodevec(x_{2},1),Nodevec(x_{3},1));
case {5}
    Nodevec(x_{1},1)=1-Nodevec(x_{2},1);
case {6}
    Nodevec(x_{1},1)=Nodevec(x_{2},1);
case {7}
    Nodevec(x_{1},1)=Nodevec(x_{2},1)*(1-Nodevec(x_{3},1))+(1-Nodevec(x_{2},1))
end
else
switch x_{5}
case {1}
    Nodevec(x_{1},1)=Nodevec(x_{2},1)*Nodevec(x_{3},1);
case {2}
    Nodevec(x_{1},1)=1-Nodevec(x_{2},1)*Nodevec(x_{3},1);
case {3}
    Nodevec(x_{1},1)=or(Nodevec(x_{2},1),Nodevec(x_{3},1));
case {4}
    Nodevec(x_{1},1)=1-or(Nodevec(x_{2},1),Nodevec(x_{3},1));
case {5}
    Nodevec(x_{1},1)=Nodevec(x_{2},1);
case {6}
    Nodevec(x_{1},1)=1-Nodevec(x_{2},1);
case {7}
    Nodevec(x_{1},1)=Nodevec(x_{2},1)*(1-Nodevec(x_{3},1))+(1-Nodevec(x_{2},1))
end
end
if (Nodevec(x_{1},1)==1)
counter1(x_{1},1)=counter1(x_{1},1)+1;
if (Nodevec(x_{1},1)==1)
    ks_counter(x_{1},1)=ks_counter(x_{1},1)+1;
    rs_counter(x_{1},1)=rs_counter(x_{1},1)+1;
end
elseif Nodevec(x_{1},1)==0
counter0(x_{1},1)=counter0(x_{1},1)+1;
if Nodevec(x_{1},1)==0
    is_counter(x_{1},1)=is_counter(x_{1},1)+1;
    rs_counter(x_{1},1)=rs_counter(x_{1},1)+1;
end
end
if Nodevec(Output)==Nodevec(Output)
    r_mul_counter=r_mul_counter+1;
end
end
for i=1:Num_node
    if counter1(i)==0
        ks(i)=ks_counter(i)/counter1(i);
    end
if counter0_(i)~=0
    ls(i)=ls_counter(i)/counter0_(i);
end

rs=rs_counter/MC;
r_joint=r_mul_counter/MC;
T_MC=toc;

% k,l upper bound
tic;
xp=Gates(:,1:3);
for i=1:Num_gate_norm
    x=Gates(i,:);
    conA=isempty(find(Input==x(2)));  
    conB=isempty(find(Input==x(3)));  
    conC=isempty(find(xp(find(xp(:,1)==x(2)),2:3)==x(3)));  
    conD=isempty(find(xp(find(xp(:,1)==x(3)),2:3)==x(2)));  
    rg1=rg;
    rg2=rg;
    if ~conC&&conD
        rg1=1;
    elseif ~conD&&conC
        rg2=1;
    elseif ~conB&&conA
        rg2=1;
    elseif ~conB&&~conA
        rg1=1;
    elseif ~conA&&conB
        rg1=1;
    end
    switch x(5)
        case {1}
            [k_up(x(1),1),l_up(x(1),1)]=NANDKL_up(k_up(x(2),1),l_up(x(2),1),k_low(x(2),1),l_low(x(2),1),k_up(x(3),1),l_up(x(3),1),k_low(x(3),1),l_low(x(3),1),rg,rg1,rg2);
        case {2}
            [k_up(x(1),1),l_up(x(1),1)]=ANDKL_up(k_up(x(2),1),l_up(x(2),1),k_low(x(2),1),l_low(x(2),1),k_up(x(3),1),l_up(x(3),1),k_low(x(3),1),l_low(x(3),1),rg,rg1,rg2);
        case {3}
            [k_up(x(1),1),l_up(x(1),1)]=NORKL_up(k_up(x(2),1),l_up(x(2),1),k_low(x(2),1),l_low(x(2),1),k_up(x(3),1),l_up(x(3),1),k_low(x(3),1),l_low(x(3),1),rg,rg1,rg2);
        case {4}
            [k_up(x(1),1),l_up(x(1),1)]=ORKL_up(k_up(x(2),1),l_up(x(2),1),k_low(x(2),1),l_low(x(2),1),k_up(x(3),1),l_up(x(3),1),k_low(x(3),1),l_low(x(3),1),rg,rg1,rg2);
        case {5}
            [k_up(x(1),1),l_up(x(1),1)]=NOTKL_up(k_up(x(2),1),l_up(x(2),1),rg);
        case {6}
            [k_up(x(1),1),l_up(x(1),1)]=BUFFKL_up(k_up(x(2),1),l_up(x(2),1),rg);
end
case {7}

[k_up(x(1),1),l_up(x(1),1)]=XORKL_up(k_up(x(2),1),l_up(x(2),1),k_low(x(2),1),l_low(x(2),1),
k_up(x(3),1),l_up(x(3),1),k_low(x(3),1),l_low(x(3),1),rg,rg1,rg2);
%case {8}
%[k(x(1),1),l(x(1),1)]=XNORKL(k(x(2),1),l(x(2),1),P_(x(2),1),k(x(3),1),l(x(3),1),P_(x(3),1),P_(x(1),1),rg);
end

if k_up(x(1))<0.5||l_up(x(1))<0.5
fprintf('k or l up is small than 0.5\n'); ei=i; %break;
elseif k_up(x(1))>1||l_up(x(1))>1
fprintf('k or l up is larger than 1\n'); ei=i; %break;
elseif isnan(k_up(x(1)))||isnan(l_up(x(1)))
  fprintf('k or l up NAN\n');
iei=1;
break;
end

 tic;
 xp=Gates(:,1:3);
 for i=1:Num_gate_norm
 x=Gates(i,:);
 conA isempty(find(Input==x(2)));
 conB isempty(find(Input==x(3)));
 conC isempty(find(xp(find(xp(:,1)==x(2)),2:3)==x(3)));
 conD isempty(find(xp(find(xp(:,1)==x(3)),2:3)==x(2)));
 rg1=rg;
 rg2=rg;
 if ~conC&conD
  rg1=1;
 elseif ~conD&conC
  rg2=1;
 elseif ~conB&conA
  rg2=1;
 elseif ~conB&~conA
  rg1=1;
  rg2=1;
 elseif ~conA&conB
  rg1=1;
 end
 switch x(5)
 case {1}
 [k_low(x(1),1),l_low(x(1),1)]=NANDKL_low(k_up(x(2),1),l_up(x(2),1),k_low(x(2),1),l_low(x(2),1),
k_up(x(3),1),l_up(x(3),1),k_low(x(3),1),l_low(x(3),1),rg,rg1,rg2,flag);
 case {2}
 [k_low(x(1),1),l_low(x(1),1)]=ANDKL_low(k_up(x(2),1),l_up(x(2),1),k_low(x(2),1),l_low(x(2),1),
k_up(x(3),1),l_up(x(3),1),k_low(x(3),1),l_low(x(3),1),rg,rg1,rg2,flag);
 case {3}
 [k_low(x(1),1),l_low(x(1),1)]=NORKL_low(k_up(x(2),1),l_up(x(2),1),k_low(x(2),1),l_low(x(2),1),

1), l_low(x(2),1), k_up(x(3),1), l_up(x(3),1), k_low(x(3),1), l_low(x(3),1), rg, rg1, rg2, flag);
    case {4}
[k_low(x(1),1), l_low(x(1),1)] = ORKL_low(k_up(x(2),1), l_up(x(2),1), k_low(x(2),1), l_up(x(2),1), k_low(x(3),1), l_up(x(3),1), k_low(x(3),1), l_low(x(3),1), rg, rg1, rg2, flag);
    case {5}
[k_low(x(1),1), l_low(x(1),1)] = NOTKL_low(k_low(x(2),1), l_low(x(2),1), rg);
    case {6}
[k_low(x(1),1), l_low(x(1),1)] = BUFFKL_low(k_low(x(2),1), l_low(x(2),1), rg);
    case {7}
[k_low(x(1),1), l_low(x(1),1)] = XORKL_low(k_up(x(2),1), l_up(x(2),1), k_low(x(2),1), l_low(x(2),1), k_up(x(3),1), l_up(x(3),1), k_low(x(3),1), l_low(x(3),1), rg, rg1, rg2);
%case {8}
%[k(x(1),1), l(x(1),1)] = XNORKL(k(x(2),1), l(x(2),1), P_(x(2),1), k(x(3),1), l(x(3),1), P_(x(3),1), P_(x(1),1), rg);
end

if k_low(x(1)) < 0.5 || l_low(x(1)) < 0.5
    fprintf('k or l low is smaller than 0.5
'); ei = i; %break;
elseif k_low(x(1)) > 1 || l_low(x(1)) > 1
    fprintf('k or l low is larger than 1
'); ei = i; %break;
elseif isnan(k_low(x(1))) || isnan(l_low(x(1)))
    fprintf('k or l low NAN
');
    ei = i;
    break;
end
end

if k_low(x(1)) < 0.5 || l_low(x(1)) < 0.5
    fprintf('k or l low is smaller than 0.5
');
else if k_low(x(1)) > 1 || l_low(x(1)) > 1
    fprintf('k or l low is larger than 1
');
elseif isnan(k_low(x(1))) || isnan(l_low(x(1)))
    fprintf('k or l low NAN
');
    ei = i;
    break;
end

% % Reliability of outputs
% for i = Num_gate_norm+1:Num_gate_AND
%     x = Gates(i,:);
%     if P_(x(2)) ~= 0 && P_(x(3)) ~= 0
%         thetaAND(i-Num_gate_norm,1) = (P_(x(1)) - P_(x(2)) * P_(x(3))) / sqrt(P_(x(2)) * (1 - P_(x(2))) * P_(x(3)) * (1 - P_(x(3)))�
%     else
%         thetaAND(i-Num_gate_norm,1) = 0;
%     end
% end
% for i = 2:Num_output
%     thetaOut(i) = max(abs(thetaAND((i-1)*(i-2)/2+1:i*(i-1)/2)));
% end
% thetaOut = ones(Num_output,1);
% multiple output
% for i = Num_gate_AND+1:Num_gate
%     x = Gates(i,:);
%     conA = isempty(find(Input == x(2)));
%     conB = isempty(find(Input == x(3)));
%     conC = isempty(find(xp(find(xp(:,1) == x(2)),2:3) == x(3)));
%     conD = isempty(find(xp(find(xp(:,1) == x(3)),2:3) == x(2)));
%     if i == Num_gate_AND+1

55
% \texttt{rg1=rg;}
% \texttt{rg2=rg;}
% \texttt{if conA\&\&conB\&\&conC\&\&conD}
% \texttt{[k(x(1),1),l(x(1),1),\texttt{theta11(i,1)}]=XORKL\_Out(k(x(2),1),l(x(2),1),P_\{x(2),1\},k(x(3),1),l(x(3),1),P_\{x(3),1\},P_\{x(1),1\},1,\texttt{rg1,rg2,mod,thetaOut(i-Num\_gate\_AND+1)});}
% \texttt{else ~conC\&\&conD}
% \texttt{rg1=1;}
% \texttt{[k(x(1),1),l(x(1),1),\texttt{theta11(i,1)}]=XORKL\_OutI(k(x(2),1),l(x(2),1),P_\{x(2),1\},k(x(3),1),l(x(3),1),P_\{x(3),1\},P_\{x(1),1\},1,\texttt{rg1,rg2,mod,thetaOut(i-Num\_gate\_AND+1)});}
% \texttt{else ~conD\&\&conC}
% \texttt{rg2=1;}
% \texttt{[k(x(1),1),l(x(1),1),\texttt{theta11(i,1)}]=XORKL\_OutI(k(x(2),1),l(x(2),1),P_\{x(2),1\},k(x(3),1),l(x(3),1),P_\{x(3),1\},P_\{x(1),1\},1,\texttt{rg1,rg2,mod,thetaOut(i-Num\_gate\_AND+1)});}
% \texttt{else ~conA\&\&~conB}
% \texttt{rg1=1;}
% \texttt{rg2=1;}
% \texttt{[k(x(1),1),l(x(1),1),\texttt{theta11(i,1)}]=XORKL\_OutI(k(x(2),1),l(x(2),1),P_\{x(2),1\},k(x(3),1),l(x(3),1),P_\{x(3),1\},P_\{x(1),1\},1,\texttt{rg1,rg2,mod,thetaOut(i-Num\_gate\_AND+1)});}
% \texttt{else ~conA\&\&conB}
% \texttt{rg1=1;}
% \texttt{[k(x(1),1),l(x(1),1),\texttt{theta11(i,1)}]=XORKL\_OutI(k(x(2),1),l(x(2),1),P_\{x(2),1\},k(x(3),1),l(x(3),1),P_\{x(3),1\},P_\{x(1),1\},1,\texttt{rg1,rg2,mod,thetaOut(i-Num\_gate\_AND+1)});}
% \texttt{else ~conB\&\&conA}
% \texttt{rg2=1;}
% \texttt{[k(x(1),1),l(x(1),1),\texttt{theta11(i,1)}]=XORKL\_OutI(k(x(2),1),l(x(2),1),P_\{x(2),1\},k(x(3),1),l(x(3),1),P_\{x(3),1\},P_\{x(1),1\},1,\texttt{rg1,rg2,mod,thetaOut(i-Num\_gate\_AND+1)});}
% \texttt{end}
% \texttt{else}
% \texttt{rg1=1;}
% \texttt{rg2=1;}
% \texttt{if conA\&\&conB\&\&conC\&\&conD}
% \texttt{[k(x(1),1),l(x(1),1),\texttt{theta11(i,1)}]=XORKL\_Out2(k(x(2),1),l(x(2),1),P_\{x(2),1\},k(x(3),1),l(x(3),1),P_\{x(3),1\},P_\{x(1),1\},1,\texttt{rg1,rg2,mod,thetaOut(i-Num\_gate\_AND+1)});}
% \texttt{else ~conC\&\&conD}
% \texttt{rg1=1;}
% \texttt{[k(x(1),1),l(x(1),1),\texttt{theta11(i,1)}]=XORKL\_OutI2(k(x(2),1),l(x(2),1),P_\{x(2),1\},k(x(3),1),l(x(3),1),P_\{x(3),1\},P_\{x(1),1\},1,\texttt{rg1,rg2,mod,thetaOut(i-Num\_gate\_AND+1)});}
% \texttt{else ~conD\&\&conC}
% \texttt{rg2=1;}
% \texttt{[k(x(1),1),l(x(1),1),\texttt{theta11(i,1)}]=XORKL\_OutI2(k(x(2),1),l(x(2),1),P_\{x(2),1\},k(x(3),1),l(x(3),1),P_\{x(3),1\},P_\{x(1),1\},1,\texttt{rg1,rg2,mod,thetaOut(i-Num\_gate\_AND+1)});}
% \texttt{end}
,k(x(3),1),l(x(3),1),P_(x(3),1),P_(x(1),1),1,rg1,rg2,mod,thetaOut(i-Num_gate_AND+1));
%     elseif ~conA&&~conB
%         rg1=1;
%         rg2=1;
%     [k(x(1),1),l(x(1),1),thetaOut(i,1)]=XORKL_OutI2(k(x(2),1),l(x(2),1),P_(x(2),1)
%     ,k(x(3),1),l(x(3),1),P_(x(3),1),P_(x(1),1),1,rg1,rg2,mod,thetaOut(i-
%     Num_gate_AND+1));
%     elseif ~conA&&conB
%         rg1=1;
%     [k(x(1),1),l(x(1),1),thetaOut(i,1)]=XORKL_OutI2(k(x(2),1),l(x(2),1),P_(x(2),1)
%     ,k(x(3),1),l(x(3),1),P_(x(3),1),P_(x(1),1),1,rg1,rg2,mod,thetaOut(i-
%     Num_gate_AND+1));
%     elseif ~conB&&conA
%         rg2=1;
%     [k(x(1),1),l(x(1),1),thetaOut(i,1)]=XORKL_OutI2(k(x(2),1),l(x(2),1),P_(x(2),1)
%     ,k(x(3),1),l(x(3),1),P_(x(3),1),P_(x(1),1),1,rg1,rg2,mod,thetaOut(i-
%     Num_gate_AND+1));
% end
% end
% T_Z=toc;
for i=1:Num_node
    r_up(i,1)=max(k_up(i),l_up(i));
    r_low(i,1)=min(k_low(i),l_low(i));
end
for i=1:Num_output
    r_pro=r_pro*rs(Output(i),1);
end
Reliability Allocation
% T1;
% circuit level development
Circuit_level;
Cir_level=max(Gates(:,6));
%ro_up=ones(Num_output,41);
% circuit polynomial development
p=zeros(1,max(Gates(:,6)));
p1=zeros(1,max(Gates(:,6)));
rs_counter=zeros(Num_node,1);
% p=zeros(21,2);
for iAU=1:regression_fac
    if iAU == 1
        alpha(1,iAU) = 1+0.2*rand(1);
        Rout(1,iAU) = 0.95+0.05*rand(1);
    elseif iAU == 1
        alpha(1,iAU) = 1;
        Rout(1,iAU) = 1;
    end
end
\[ r_0 = \frac{R_{out}(1,iAU)}{\alpha(1,iAU)^{(Cir\_level-1)}}; \]

\begin{verbatim}
for irg = 1:Cir_level
    rg(irg,1) = r0*\alpha(1,iAU)^{(Cir\_level-irg)};
end
for irgx = 1:Num\_gate
    index = Gates(irgx,:);
    rg\_ex(irgx,1) = rg(index(6));
end
rg\_std = std\_gate*ones(Num\_gate,1);
Gates1 = [Gates,rg\_ex];
Gates3 = [Gates,rg\_std];
\end{verbatim}

\% MC\_RealValue;
\begin{verbatim}
r\_up=ones(Num\_node,1);
r\_low=ones(Num\_node,1);
k\_up=ones(Num\_node,1);
k\_upstd=ones(Num\_node,1);
l\_up=ones(Num\_node,1);
l\_upstd=ones(Num\_node,1);
k\_low=ones(Num\_node,1);
l\_low=ones(Num\_node,1);
l\_lowstd=ones(Num\_node,1);
xp=Gates1(:,1:3);
for i=1:Num\_gate
    x=Gates1(i,:);
    conA=isempty(find(Input==x(2)));
    conB=isempty(find(Input==x(3)));
    conC=isempty(find(xp(find(xp(:,1)==x(2)),2:3)==x(3)));
    conD=isempty(find(xp(find(xp(:,1)==x(3)),2:3)==x(2)));
    rg1=x(7);
    rg2=x(7);
    if ~conC&&conD
        rg1=1;
    elseif ~conD&&conC
        rg2=1;
    elseif ~conB&&conA
        rg1=1;
    elseif ~conB&&~conA
        rg1=1;
    elseif ~conA&&conB
        rgl=1;
    end
    switch x(5)
        case {1}
        [k\_up(x(1),1),l\_up(x(1),1)]\=NANDKL\_up(k\_up(x(2),1),l\_up(x(2),1),k\_low(x(2),1)
        ,l\_low(x(2),1),k\_up(x(3),1),l\_up(x(3),1),k\_low(x(3),1),l\_low(x(3),1),x(7),rgl
        ,rg2);
    case {2}
        [k\_up(x(1),1),l\_up(x(1),1)]\=ANDKL\_up(k\_up(x(2),1),l\_up(x(2),1),k\_low(x(2),1)
        ,l\_low(x(2),1),k\_up(x(3),1),l\_up(x(3),1),k\_low(x(3),1),l\_low(x(3),1),x(7),rg1,
        rg2);
    case {3}
\end{verbatim}
\[\text{case } \{4\}\]

\[\text{case } \{5\}\]

\[\text{case } \{6\}\]

\[\text{case } \{7\}\]

\[\text{end}\]

\[\text{end}\]

\[\text{xp=Gates1(:,1:3)};\]

\[\text{for } i=1:\text{Num\_gate}\]

\[x=Gates1(i,:);\]

\[\text{conA=isempty(find(Input==x(2))));}\]

\[\text{conB=isempty(find(Input==x(3))));}\]

\[\text{conC=isempty(find(xp(find(xp(:,1)==x(2)),2:3)==x(3))));}\]

\[\text{conD=isempty(find(xp(find(xp(:,1)==x(3)),2:3)==x(2))));}\]

\[\text{rg1=(x(6));}\]

\[\text{rg2=(x(6));}\]

\[\text{if } \sim\text{conC}\&\&\text{conD}\]

\[\text{rg1=1;}\]

\[\text{elseif } \sim\text{conD}\&\&\text{conC}\]

\[\text{rg2=1;}\]

\[\text{elseif } \sim\text{conB}\&\&\text{conA}\]

\[\text{rg2=1;}\]

\[\text{elseif } \sim\text{conB}\&\&\sim\text{conA}\]

\[\text{rg1=1;}\]

\[\text{rg2=1;}\]

\[\text{elseif } \sim\text{conA}\&\&\text{conB}\]

\[\text{rg1=1;}\]

\[\text{end}\]

\[\text{switch } x(5)\]

\[\text{case } \{1\}\]

\[\text{case } \{2\}\]

\[\text{case } \{3\}\]
\[ \text{k_low}(x(1), 1), \text{l_low}(x(1), 1) = \text{NORKL}_\text{low}(\text{k_up}(x(2), 1), \text{l_up}(x(2), 1), \text{k_low}(x(2), 1), \text{l_low}(x(2), 1), \text{k_up}(x(3), 1), \text{l_up}(x(3), 1), \text{k_low}(x(3), 1), \text{l_low}(x(3), 1), x(7), \text{rg}_1, \text{rg}_2, \text{flag}); \]

\text{case} \{4\}

\[ \text{k_low}(x(1), 1), \text{l_low}(x(1), 1) = \text{ORKL}_\text{low}(\text{k_up}(x(2), 1), \text{l_up}(x(2), 1), \text{k_low}(x(2), 1), \text{l_low}(x(2), 1), \text{k_up}(x(3), 1), \text{l_up}(x(3), 1), \text{k_low}(x(3), 1), \text{l_low}(x(3), 1), x(7), \text{rg}_1, \text{rg}_2, \text{flag}); \]

\text{case} \{5\}

\[ \text{k_low}(x(1), 1), \text{l_low}(x(1), 1) = \text{NOTKL}_\text{low}(\text{k_low}(x(2), 1), \text{l_low}(x(2), 1), x(7)); \]

\text{case} \{6\}

\[ \text{k_low}(x(1), 1), \text{l_low}(x(1), 1) = \text{BUFFKL}_\text{low}(\text{k_low}(x(2), 1), \text{l_low}(x(2), 1), x(7)); \]

\text{case} \{7\}

\[ \text{k_low}(x(1), 1), \text{l_low}(x(1), 1) = \text{XORKL}_\text{low}(\text{k_up}(x(2), 1), \text{l_up}(x(2), 1), \text{k_low}(x(2), 1), \text{l_low}(x(2), 1), \text{k_up}(x(3), 1), \text{l_up}(x(3), 1), \text{k_low}(x(3), 1), \text{l_low}(x(3), 1), x(7), \text{rg}_1, \text{rg}_2); \]

```matlab
end
end
xp=Gates3(:,1:3);
for i=1:Num_gate
x=Gates3(i,:);
conA=isempty(find(Input==x(2)));
conB=isempty(find(Input==x(3)));
conC=isempty(find(xp(find(xp(:,1)==x(2)),2:3)==x(3)));
conD=isempty(find(xp(find(xp(:,1)==x(3)),2:3)==x(2)));
rg1=x(7);
rg2=x(7);
if ~conC&&conD
    rg1=1;
elseif ~conD&&conC
    rg2=1;
elseif ~conB&&conA
    rg2=1;
elseif ~conB&&~conA
    rg1=1;
    rg2=1;
else
    ~conA&&conB
    rg1=1;
end
switch x(5)
    case \{1\}
        \[ \text{k_upstd}(x(1), 1), \text{l_upstd}(x(1), 1) = \text{NANDKL}_\text{up}(\text{k_upstd}(x(2), 1), \text{l_upstd}(x(2), 1), \text{k_lowstd}(x(2), 1), \text{l_lowstd}(x(2), 1), \text{k_upstd}(x(3), 1), \text{l_upstd}(x(3), 1), \text{k_lowstd}(x(3), 1), \text{l_lowstd}(x(3), 1), x(7), \text{rg}_1, \text{rg}_2); \]
    case \{2\}
        \[ \text{k_upstd}(x(1), 1), \text{l_upstd}(x(1), 1) = \text{ANDKL}_\text{up}(\text{k_upstd}(x(2), 1), \text{l_upstd}(x(2), 1), \text{k_lowstd}(x(2), 1), \text{l_lowstd}(x(2), 1), \text{k_upstd}(x(3), 1), \text{l_upstd}(x(3), 1), \text{k_lowstd}(x(3), 1), \text{l_lowstd}(x(3), 1), x(7), \text{rg}_1, \text{rg}_2); \]
    case \{3\}
end
```

60
\[ k_{\text{upstd}}(x(1),1), l_{\text{upstd}}(x(1),1) = NORKL_u \]
\[ p(k_{\text{upstd}}(x(2),1), l_{\text{upstd}}(x(2),1), k_{\text{lowstd}}(x(2),1), l_{\text{lowstd}}(x(2),1), k_{\text{upstd}}(x(3),1), l_{\text{upstd}}(x(3),1), k_{\text{lowstd}}(x(3),1), l_{\text{lowstd}}(x(3),1), x(7), rg1, rg2); \]

\[ \text{case } \{4\} \]

\[ k_{\text{upstd}}(x(1),1), l_{\text{upstd}}(x(1),1) = ORKL_u (k_{\text{upstd}}(x(2),1), l_{\text{upstd}}(x(2),1), k_{\text{lowstd}}(x(2),1), l_{\text{lowstd}}(x(2),1), k_{\text{upstd}}(x(3),1), l_{\text{upstd}}(x(3),1), k_{\text{lowstd}}(x(3),1), l_{\text{lowstd}}(x(3),1), x(7), rg1, rg2); \]

\[ \text{case } \{5\} \]

\[ k_{\text{upstd}}(x(1),1), l_{\text{upstd}}(x(1),1) = NOTKL_u (k_{\text{upstd}}(x(2),1), l_{\text{upstd}}(x(2),1), x(7)); \]

\[ \text{case } \{6\} \]

\[ k_{\text{upstd}}(x(1),1), l_{\text{upstd}}(x(1),1) = BUFFKL_u (k_{\text{upstd}}(x(2),1), l_{\text{upstd}}(x(2),1), x(7)); \]

\[ \text{case } \{7\} \]

\[ k_{\text{upstd}}(x(1),1), l_{\text{upstd}}(x(1),1) = XORKL_u (k_{\text{upstd}}(x(2),1), l_{\text{upstd}}(x(2),1), k_{\text{lowstd}}(x(2),1), l_{\text{lowstd}}(x(2),1), k_{\text{upstd}}(x(3),1), l_{\text{upstd}}(x(3),1), k_{\text{lowstd}}(x(3),1), l_{\text{lowstd}}(x(3),1), x(7), rg1, rg2); \]

end

\[ \text{end} \]

\[ xp = \text{Gates3}(:,1:3); \]

\[ \text{for } i = 1: \text{Num\_gate} \]
\[ \text{x = } \text{Gates3}(:,i,:); \]
\[ \text{conA} = \text{isempty}(\text{find(Input} == x(2))); \]
\[ \text{conB} = \text{isempty}(\text{find(Input} == x(3))); \]
\[ \text{conC} = \text{isempty}(\text{find}\text{xp}(\text{find(xp}(:,1) == x(2)),2:3) == x(3))); \]
\[ \text{conD} = \text{isempty}(\text{find}\text{xp}(\text{find(xp}(:,1) == x(3)),2:3) == x(2))); \]
\[ \text{rg1} = x(6); \]
\[ \text{rg2} = x(6); \]
\[ \text{if } \neg\text{conC} \& \& \text{conD} \]
\[ \text{rg1} = 1; \]
\[ \text{elseif } \neg\text{conD} \& \& \text{conC} \]
\[ \text{rg2} = 1; \]
\[ \text{elseif } \neg\text{conB} \& \& \text{conA} \]
\[ \text{rg2} = 1; \]
\[ \text{elseif } \neg\text{conB} \& \& \neg\text{conA} \]
\[ \text{rg1} = 1; \]
\[ \text{rg2} = 1; \]
\[ \text{elseif } \neg\text{conA} \& \& \neg\text{conB} \]
\[ \text{rg1} = 1; \]
\[ \text{end} \]

\[ \text{switch } x(5) \]
\[ \text{case } \{1\} \]
\[ \text{[k_{\text{lowstd}}(x(1),1), l_{\text{lowstd}}(x(1),1)] = NANDKL\_low (k_{\text{upstd}}(x(2),1), l_{\text{upstd}}(x(2),1), k_{\text{lowstd}}(x(2),1), l_{\text{lowstd}}(x(2),1), k_{\text{upstd}}(x(3),1), l_{\text{upstd}}(x(3),1), k_{\text{lowstd}}(x(3),1), l_{\text{lowstd}}(x(3),1), x(7), rg1, rg2, flag); \]
\[ \text{case } \{2\} \]
\[ \text{[k_{\text{lowstd}}(x(1),1), l_{\text{lowstd}}(x(1),1)] = ANDKL\_low (k_{\text{upstd}}(x(2),1), l_{\text{upstd}}(x(2),1), k_{\text{lowstd}}(x(2),1), l_{\text{lowstd}}(x(2),1), k_{\text{upstd}}(x(3),1), l_{\text{upstd}}(x(3),1), k_{\text{lowstd}}(x(3),1), l_{\text{lowstd}}(x(3),1), x(7), rg1, rg2, flag); \]
\[ \text{case } \{3\} \]
for j=1:Num_node
    r_up(j,1)=max(k_up(j),l_up(j));
    r_low(j,1)=min(k_low(j),l_low(j));
    r_upstd(j,1) = max(k_upstd(j),l_upstd(j));
end
for j=1:Num_output
    ro_up(j,iAU)=r_up(Output(j));
    ro_upstd(j,iAU) = r_upstd(Output(j));
end
ro_upave = sum(ro_up)/Num_output;
ro_upavestd = sum(ro_upstd)/Num_output;
REFERENCES


Suoyue Zhan, born in 1993, received his bachelor’s degree of Electrical Engineering from Department of Electrical Engineering in University of Electrical and Science Technology of China in 2015. He is currently a candidate for Master of Applied Science Degree in Department of Electrical and Computer Engineering in University of Windsor and is expected to graduate in Fall 2018.