POST-SILICON TIMING DIAGNOSIS UNDER PROCESS VARIATIONS

by

Lin Xie

A dissertation submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

(Electrical Engineering)

at the

UNIVERSITY OF WISCONSIN – MADISON

2010
Acknowledgments

First of all, I would like to express my deepest thanks and appreciation to my academic advisor Prof. Azadeh Davoodi. During the past few years, Professor Davoodi provided me with the opportunities to work on the most cutting-edge and important research topics in VLSI Electronic Design Automation area. From her, I have learned how to formulate problems with both scientific and technological significance and how to solve them in an efficient and systematic way. She also gave me countless advice in academic writing and my future career path. Without her guidance and generous support, I could not have proceeded so far in my PhD study. I sincerely appreciate her patient help, precious advice, and consistent encouragement.

I also would like to thank my PhD committee members, Professor Kewal K. Saluja, Professor Parameswaran Ramanathan, Professor Yu Hen Hu, and Professor Vicki Bier for their precious time in reviewing my thesis and their valuable advice to improve my work. Particularly, I would give special thanks to Professor Saluja. The collaborations with him provided me a broader view of research from Testing area, and gave me an opportunity to derive his valuable advices and suggestions on how to overcome difficulties in the research.

I am grateful to both the former and present members of VLSI Electronic Au-
tomation Lab in the University of Wisconsin-Madison (WISCAD), Michael Anderson, Anuj Kumar, Shreyas Parnerkar, Meng Shi, Hamid Shojaei, and Tai-Hsuan Wu, for the assistance they have given me during my PhD study and their helpful discussions that enriched my research.

Finally, I owe thanks to my parents who have always stood by me and guided me through my pursuit of PhD degree. My life is more meaningful because of their love and care. Words cannot express the gratitude I owe them.
Abstract

With continuous technology scaling, process variations have become a major factor affecting the performance of VLSI designs. Even a small degree of variation in device parameters may amplify at the system level, resulting in significant deviations in circuit timing. Post-silicon timing diagnosis addresses the issues of estimating the timing of each fabricated chip. It aims to identify individual paths that fail to meet their required timing constraints, despite the lack of access to the inside of the chip and the limitation to at most a few thousand I/O pins from which to reason about the timing behavior of billions of nano-scale components in a circuit. The lack of both access to and knowledge of individual process variations inside a fabricated chip causes post-silicon timing diagnosis to be timing-consuming and expensive.

In this thesis, we propose a framework for automating post-silicon timing diagnosis under process variations. First, our goal is to \textit{characterize} the timing of the fabricated chips in order to isolate the failing paths in “slow” chips that fail to operate correctly at the required frequency. Moreover, our framework localizes those segments on the failing paths that show more deviation in their post-silicon delays.

The first step in our proposed framework is to identify the statistically-critical paths under process variations. These are the paths that have the highest probability
of failing their frequency requirements in the presence of process variations, and can therefore form a good pool of candidates for post-silicon timing diagnosis. We present an approach to identifying these critical paths based on computing analytical bounds. Particularly, we first define a “violation probability” for each gate or interconnect in a circuit. The violation probability measures the likelihood that a particular gate or interconnect forms part of a statistically-critical path, and can be efficiently computed as a pre-processing step. With these probabilities, we then compute lower and upper bounds on the violation probability for an arbitrary segment of consecutive gates and interconnects. The computation is of constant computational complexity if done incrementally as the segment size grows. We then use these bounds to identify those statistically-critical paths that involves only one traversal of the timing graph representing the circuit. Simulation results show that our proposed approach is both efficient and accurate.

Next, we aim to isolate those “failing paths” that cannot meet the required timing constraint. This is done by predicting the post-silicon delays of the statistically-critical paths which were identified before fabrication. The prediction requires measuring the delays of a few paths representing the statistically-critical paths after fabrication. To identify these “representative critical paths”, we take advantage of the fact that the delays of the statistically-critical paths are highly correlated, due to the large number of overlaps among these paths, and the spatial correlation among process variations. We utilize the novel concept of “effective rank” to select few representative critical paths. Simulation results again show that we can predict the timing of a few thousand statistically-critical paths from the delays of around one hundred representative critical paths in the face of around one thousand process variations.
After the failing paths are isolated, we formulate an optimization problem to diagnose those segments on the failing paths that show a large deviation in their post-silicon delays. This optimization makes use of the post-silicon delay measurements on the representative critical paths for delay prediction of segments inside a path. It forms segments that maximize a defined metric referred to as “diagnostic resolution”. A high diagnostic resolution indicates that the chosen segments are likely to have a post-silicon delay larger than their design-time estimates. We illustrate via simulation that our proposed procedure is computationally efficient, and can yield a high diagnostic resolution in identifying “failing segments”. Further, we can accurately rank these identified segments in order of their delay deviations. Diagnosis of these failing segments increases the observability inside each failing path at the post-silicon stage. It can thus help with post-silicon timing repair, and can be analyzed to find the cause of timing failures.
Contents

Acknowledgments i

Abstract iii

List of Figures x

List of Tables xii

1 Introduction 1
   1.1 Process Variations Trends ................................. 2
   1.2 Impact of Process Variations on Timing .................... 5
       1.2.1 Pre-Silicon Timing Analysis .......................... 5
       1.2.2 Post-Silicon Timing Diagnosis ....................... 7
   1.3 Our Contributions ............................................. 10

2 Preliminaries 14
   2.1 Static Timing Analysis ..................................... 14
       2.1.1 Gate Delay Model ................................. 15
       2.1.2 Path Delay Computation .......................... 16
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.3</td>
<td>Circuit Block Delay Computation</td>
<td>17</td>
</tr>
<tr>
<td>2.2</td>
<td>Process Variation Model</td>
<td>20</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Spatial Correlation Model</td>
<td>22</td>
</tr>
<tr>
<td>2.3</td>
<td>Gate/Interconnect Delay Model Incorporating Process Variations</td>
<td>26</td>
</tr>
<tr>
<td>2.4</td>
<td>Statistic Static Timing Analysis</td>
<td>28</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Monte Carlo based Statistical Static Timing Analysis</td>
<td>29</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Parameterized Statistical Static Timing Analysis</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>Statistically-Critical Path Extraction Under Process Variations</td>
<td>33</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>34</td>
</tr>
<tr>
<td>3.2</td>
<td>Problem Definition</td>
<td>37</td>
</tr>
<tr>
<td>3.3</td>
<td>Node/Edge Violation Probability</td>
<td>38</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Definition and Computation</td>
<td>38</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Computational Complexity</td>
<td>40</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Comparison with Criticality Probability</td>
<td>42</td>
</tr>
<tr>
<td>3.4</td>
<td>Bound Computation of Violation Probability for Two-Connected Edges</td>
<td>43</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Lower &amp; Upper Bound Computation</td>
<td>46</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Discussion on Tightness of the Bounds</td>
<td>49</td>
</tr>
<tr>
<td>3.4.3</td>
<td>Lower Bound Comparisons</td>
<td>51</td>
</tr>
<tr>
<td>3.5</td>
<td>Extensions for $K$ Connected Edges</td>
<td>52</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Lower &amp; Upper Bound Computation</td>
<td>52</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Discussion on Tightness of the Bounds</td>
<td>54</td>
</tr>
<tr>
<td>3.6</td>
<td>Our Proposed Algorithm</td>
<td>55</td>
</tr>
<tr>
<td>3.6.1</td>
<td>Edge Combinations</td>
<td>56</td>
</tr>
<tr>
<td>3.6.2</td>
<td>Optimal Pruning of Inferior Segments</td>
<td>57</td>
</tr>
</tbody>
</table>
3.6.3 Selecting top $n$ paths at Super Primary Output .......................... 59
3.6.4 Runtime Complexity ......................................................... 59
3.7 Extracting The Most Statistically-Critical Path ............................. 60
3.8 Experimental Results .......................................................... 62
  3.8.1 Verifying Accuracy and Efficiency of Path Extraction ......... 64
  3.8.2 Reduction for Most Statistically-Critical Path ................. 68
  3.8.3 Experiment with Non-Gaussian Process Variations ........... 69
3.9 Summary ................................................................. 70

4 Representative Critical Path Selection for Failing Path Isolation 72
  4.1 Introduction ............................................................ 73
  4.2 Motivation ............................................................... 75
  4.3 Problem Definition ....................................................... 78
  4.4 Our Proposed Algorithm ............................................... 79
    4.4.1 Exact Selection ................................................... 80
    4.4.2 Representative Critical Path Selection With Effective Rank 82
    4.4.3 Representative Critical Path Selection Procedure with $\epsilon$ . 84
    4.4.4 Complexity Analysis .............................................. 89
  4.5 Experimental Results ................................................... 90
    4.5.1 Representative Critical Path Selection ......................... 92
    4.5.2 Failing Path Isolation ............................................ 96
  4.6 Summary ............................................................... 97

5 Post-Silicon Timing Diagnosis of Segments on Failing Paths 98
  5.1 Introduction ............................................................ 99
5.2 Preliminaries ................................................. 101
5.3 Problem Definition ......................................... 103
  5.3.1 Segment Type Definition ................................. 103
  5.3.2 Diagnostic Resolution .................................. 104
5.4 Our Proposed Algorithm ................................. 107
  5.4.1 Mathematical Formulation ............................. 107
  5.4.2 Linearization ......................................... 112
5.5 Experimental Results .................................... 114
  5.5.1 Comparisons on Diagnostic Resolution .......... 116
  5.5.2 Evaluation of Ranking The Failing Segments ... 120
  5.5.3 Evaluation of Sharing in Failing Paths ........... 122
5.6 Summary .................................................. 123

6 Conclusions and Future Work .......................... 124
  6.1 Conclusions ............................................. 124
  6.2 Future Work ............................................ 127
## List of Figures

1.1 The coefficient of variation for different device parameters from 250nm to 70nm technology [63] .............................. 3
1.2 Our proposed framework for post-silicon timing diagnosis ............ 10
2.1 Delay dependence on input patterns in a 2-input NAND gate from [22]. 15
2.2 The computation of the data path delay: \( d_{SDT \rightarrow G1 \rightarrow G2 \rightarrow G3 \rightarrow FF0} \) ....... 17
2.3 (a) A simple example of circuit block; (b) The associated timing graph representation .............................................. 18
2.4 Grid based spatial correlation model for WID variations ............... 23
2.5 A two-level hierarchical spatial correlation model for WID variations . 24
3.1 Example circuit and its corresponding timing graph ....................... 41
3.2 Two connected edges \( e_{1,2} \) and \( e_{2,3} \) .......................... 43
3.3 Visual representation of node, edge, and path violation probabilities for the simple case ......................................... 45
3.4 General case of many connected edges .................................. 52
3.5 Three cases for graph modification: remove edges in solid line and intermediate nodes from \( \mathcal{E} \), and add edges in dashed line to \( \mathcal{E} \) ....... 56
3.6 Classification of the segments for \( n = 3 \) ................................. 57
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>A simple example to show path delay dependence.</td>
<td>75</td>
</tr>
<tr>
<td>4.2</td>
<td>Visualization of our representative critical path selection problem.</td>
<td>79</td>
</tr>
<tr>
<td>4.3</td>
<td>The normalized singular values of transformation matrix $A_{P_c}$ under two configurations.</td>
<td>84</td>
</tr>
<tr>
<td>4.4</td>
<td>Histograms of prediction errors for $P_m$ for S38417 with $\epsilon = 8%$.</td>
<td>96</td>
</tr>
<tr>
<td>5.1</td>
<td>Overview of our segment diagnosis framework</td>
<td>100</td>
</tr>
<tr>
<td>5.2</td>
<td>Relationship of different segment types.</td>
<td>105</td>
</tr>
<tr>
<td>5.3</td>
<td>Visualization of our segment diagnosis problem formulation.</td>
<td>106</td>
</tr>
<tr>
<td>5.4</td>
<td>Illustration of the variable definitions used in our ILP formulation.</td>
<td>108</td>
</tr>
<tr>
<td>5.5</td>
<td>Scatter plots of number of edges merged into segments, and number of actual failing edges drawn with respect to the path delay deviation.</td>
<td>120</td>
</tr>
</tbody>
</table>
## List of Tables

1.1 The nominal values of different device parameters from 250nm to 70nm technology [63] .......................... 3

3.1 Accuracy and efficiency of our bound-based algorithm: $T_0$(PSEC), Runtime(SEC) ........................................ 63

3.2 Accuracy and efficiency of our bound-based algorithm: Runtime(SEC) 65

3.3 Comparison in the number of matched paths in Case 2 ($n = 200$) .... 67

3.4 Graph pruning for $n = 1$ ................................................ 68

3.5 Accuracy and efficiency of our bound-based algorithm with non-Gaussian process variations ........................................ 69

4.1 Results for representative critical path selection with loose timing constraint. ........................................ 93

4.2 Results for representative critical path selection with tight timing constraint. ........................................ 94

5.1 Results of our segment diagnosis framework for two sets of failing paths.117

5.2 Results for the ranking of failing segments and sharing information over failing paths. .......................... 121
Chapter 1

Introduction

The reduced level of accuracy in the manufacturing process of integrated circuits (ICs) results in increasing deviation from the expected parameters of nano-scale devices on chips [4, 64]. These manufacturing-induced deviations are called process variations. Slight deviations at the device level can amplify at the system level and result in significant deviations in performance metrics, such as circuit timing and power dissipation [27].

Another source of variations that can affect the performance of modern circuit designs is environmental variations. They are mainly caused by changes in circuit operating conditions, such as fluctuations in temperature and supply voltage. To guarantee that a chip can work correctly in most operating conditions, IC designers should take environmental variations into account.

To account for the impact of variations before “silicon” (i.e., fabrication), the timing of the ICs is traditionally analyzed using a method known as corner-based Static Timing Analysis (STA), in which the worst-case values of process variations
are considered. However, with the increase of the range and dimension of process variations, which is the trend of current technology, the corner-based timing analysis becomes very time-consuming and pessimistic. This increase can also result in tremendous mismatch between the estimated timing of the designs before silicon and the observed timing of the chips after silicon. Some of the fabricated chips can be “slow” and may not meet a required frequency as expected at the design stage. To detect these slow chips and further repair them require accurate and efficient post-silicon timing diagnosis. The timing diagnosis mainly includes isolation and fixing the paths that fail to meet the operation speed. However, the lack of access to the inside of the chip and limitation to an interface of a few thousand I/O pins makes the post-silicon timing diagnosis to be very expensive and time-consuming.

This thesis focuses on the impact of process variations on post-silicon timing diagnosis, while environmental variations are not considered. In this chapter, we discuss the trends of process variations and illustrate the impact of process variations on pre- and post-silicon timing analysis. Then, we present our research work and the contributions of this thesis.

1.1 Process Variations Trends

With the continuous technology scaling, process variations in device parameters follow the trend described below:

- The dimension of process variations in device parameters greatly increases with the technology scaling. In the past, the variations in the transistor parameters were the main source of process variations. However, currently, the intercon-
Table 1.1: The nominal values of different device parameters from 250nm to 70nm technology [63]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_{eff} ) (( nm ))</td>
<td>250</td>
<td>180</td>
<td>130</td>
<td>100</td>
<td>70</td>
</tr>
<tr>
<td>( T_{ox} ) (( nm ))</td>
<td>5</td>
<td>4.5</td>
<td>4</td>
<td>3.5</td>
<td>3</td>
</tr>
<tr>
<td>( V_T ) (( V ))</td>
<td>0.5</td>
<td>0.45</td>
<td>0.4</td>
<td>0.35</td>
<td>0.3</td>
</tr>
<tr>
<td>( W_{int} ) (( \mu m ))</td>
<td>0.8</td>
<td>0.65</td>
<td>0.5</td>
<td>0.4</td>
<td>0.3</td>
</tr>
<tr>
<td>( T_{int} ) (( \mu m ))</td>
<td>1.2</td>
<td>1</td>
<td>0.9</td>
<td>0.8</td>
<td>0.7</td>
</tr>
</tbody>
</table>

![Figure 1.1: The coefficient of variation for different device parameters from 250nm to 70nm technology [63]](image)

nects are also subject to large levels of variations, especially when the width and thickness of interconnects are scaled down.

- The ranges of process variations in device parameters also greatly increase when the technology moves from 250nm to 22nm [2]. Table 1.1 lists the nominal values of different device parameters under five technologies ranging from 250nm to 70nm [63]. These parameters are effective channel length \( L_{eff} \), gate oxide thickness \( T_{ox} \), and thermal voltage \( V_T \) for transistors, and width \( W_{int} \) and thick-


ness $T_{int}$ for interconnects. Fig. 1.1 plots the Coefficient of Variation (CV) for the above device parameters, which are defined as the standard deviations divided by their means. As we can see, the CVs for all device parameters get increased with technology scaling.

- With respect to the same device parameter, the variations in different transistors on the same die now become increasingly different, and their correlations get more complicated.

The above trends will continue with the continuous technology scaling based on the International Technology Roadmap for Semiconductors (ITRS) projections [2]. Specifically, under nanometer technology, the minimum feature sizes have reached the resolution limits of photolithography systems, and fabrication processes are forced to operate in sub-wavelength lithography regime. The modern resolution enhancement techniques, such as optical proximity correction and phase shift masking can overcome some of these effects, but the level of control over feature sizes and the so-called critical dimensions get reduced [47]. Some of these variations can be modeled deterministically, but there is still a remnant which is totally random. This results in a great increase in the total number of independent process variations. In addition, device parameters are subject to more variations: for example, as devices grow smaller and the number of dopant atoms per transistor becomes in the range of 10 to 100, the level of control and number of atoms decreases, which result in more variation in the threshold voltage [47].
1.2 Impact of Process Variations on Timing

In this section, we discuss the impact of process variations on the timing of ICs at both pre- and post-silicon stages.

1.2.1 Pre-Silicon Timing Analysis

Pre-silicon timing analysis is to verify the timing of ICs at design stage (before silicon). Due to the existence of process variations, the fabricated chips corresponding to the same design can have quite different maximum operating frequency under which they can operate correctly. To incorporate this difference before silicon and guarantee that nearly all chips after silicon are able to meet the required frequency specification, two types of pre-silicon timing analysis techniques are developed. They are discussed as follows:

**Corner-Based Static Timing Analysis:** Traditional corner-based STA utilizes the worst-case values of process variations to compute the maximum circuit delay among all fabricated chips. However, this type of STA can be overly pessimistic since it is very unlikely that the worst-case “corners” of process variations can occur simultaneously. For example, we assume that there exists two independent process variations in one IC design and that both of these two variations follow Gaussian distributions with mean $\mu$ and standard deviation $\sigma$. The probability that each process variation lies out of the range $[\mu - 3\sigma, \mu + 3\sigma]$ is 0.2%, where $\mu \pm 3\sigma$ is always considered as the worst-case corners for Gaussian distributions. It further indicates that the probability that both of these two process variations fall out of the range $[\mu - 3\sigma, \mu + 3\sigma]$ is equal to $0.2\% \times 0.2\% = 0.04\%$, which is pretty small.
Therefore, this type of corner-based STA becomes highly pessimistic with the increase of the dimension of the independent process variations, which is the trend of future semiconductor technology [2].

**Statistical Static Timing Analysis (SSTA):** The SSTA assumes that the exact distributions of process variations are available. It aims to provide accurate Probability Density Function (PDF) for the circuit timing [7, 8, 15, 18, 20, 21, 23, 28, 35, 46, 68] instead of the worst-case circuit timing value. More importantly, for a given operating frequency, we can use the PDF to evaluate the probability that the fabricated chips can operate correctly. The alternative is that we can use SSTA to determine the maximum operating frequency, when a large proportion of the fabricated chips, such as 99%, can work correctly. Therefore, SSTA can greatly reduce the pessimism, which intrinsically exists in corner-based STA.

Generally speaking, there are two types of SSTA approaches: **Monte Carlo based SSTA** [18, 20, 28, 68] and **Parameterized SSTA** [7, 15, 21, 23, 35, 46]. The Monte Carlo based SSTA first generates a large number of samples for process variations, and then implements STA under each sample. Thus, this type of SSTA can be very accurate but very slow due to the high dimension of process variations. Differently, Parameterized SSTA first expresses the delay of each gate/interconnect as a low-order polynomial function with respect to process variations. Then, it analytically computes the arrival time at each node (including gates and interconnects) and model it as another polynomial. Here, we refer to arrival time as the maximum time to propagate a signal from the input of the circuit to that node. Therefore, Parameterized SSTA can be very fast but may suffer from the error in the delay modeling and analytical arrival time approximations.
Nowadays, SSTA has been fully accepted by the industry. However, with the further scaling of the submicron technology, existing SSTA techniques would have the following limitations:

- As the manufacturing process becomes more complicated, it would be very difficult to derive the exact distribution of process variations. The availability of these information is the assumption of nearly all existing SSTA techniques except the works such as [40, 71].

- The increase in the dimension and the ranges of process variations would make the widely used low-order polynomial delay models in existing SSTA to be not accurate enough [38].

- To achieve high efficiency, existing techniques rarely incorporate Multiple-Input-Switching gate delay models, cross-talk and false paths, which would degrade the performance of SSTA [36, 39, 65].

So far, we have illustrated the challenges of considering process variations during pre-silicon timing analysis. The presence of process variations also makes post-silicon timing diagnosis to be significantly expensive and time-consuming, which we will illustrate in the following section.

1.2.2 Post-Silicon Timing Diagnosis

Post-silicon diagnosis is one important step in the development of modern VLSI circuit designs. During the pre-silicon stage, engineers rely on sophisticated simulation, emulation and formal verification tools to test the behavior and performance charac-
teristics of the designs. In contrast, post-silicon diagnosis is conducted on fabricated chips to make sure that they can work as intended.

Post-silicon diagnosis has nowadays become very time-consuming and expensive, since the actual chip behaviors are very unlikely to get simulated at pre-silicon with sufficient accuracy. Post-silicon diagnosis is referred to as “dirty little secret”, which can cost $15 million to $20 million and take six months to complete [56].

In the past, “logic errors” due to the incorrect functional implementation were the main sources impacting the functional behavior of the chip. However, with the continuous technology scaling, electrical failures, which are due to the inaccuracies in modeling nano-scale non-idealities, have now become significantly more cumbersome to detect and analyze [53, 60]. Specifically, those malfunctions that manifest in the form of timing failures (which cause setup and hold time violations on logic) take the majority of the post-silicon diagnosis cycle [52]. Localizing timing failures and identifying their causes are crucial to all domains of the electronic industry including microprocessors [9, 31], System on Chips integrating Intellectual Properties [44, 57], and ASICs, in which meeting a target frequency is a necessity [48]. We classify the related research works into the following categories:

**Failing Path Isolation:** Here, we refer to failing paths as the paths which have post-silicon delays larger than circuit delay at the pre-silicon stage (with pre-silicon models). The isolation of failing paths is a crucial step in post-silicon timing diagnosis. It is required for “fixing” the failing paths [26, 62, 67]. In [32], a manually-guided technique is given to isolate failing paths with “failing test” on an equipment known as “debug tester”. Specifically, failing test is performed by gradually shrinking the clock cycle until the fabricated chip cannot work correctly. Isolation is then accomplished
using the extracted data from the debug tester under this failing frequency. In [53], a statistical learning approach is proposed to predict the failing paths by measuring the delays of a small set of representative paths at the post-silicon stage. In [59], a branch-and-bound technique for isolation of failing paths is proposed which is based on parameterized SSTA. The proposed isolation is based on the chip measurements obtained from the scan elements placed on the circuit.

There are some limitations and drawbacks in the existing works:

- The approach in [32] is not automatic and thus can be very time-consuming. It also requires expensive equipment.

- Both [53] and [59] do not discuss the selection of the measurement sites at the post-silicon stage. This is not trivial since the selection of representative paths [53] and the placement of scan elements [59] would influence the efficiency and accuracy in the failing paths isolation. Moreover, the number of measurements (i.e., representative paths and scan elements) would reflect the effort in post-silicon timing diagnosis, which should be considered as another factor.

Causality Analysis: The challenging task in post-silicon timing diagnosis is to understand the cause of timing failures in order to apply repair or tuning. This causality analysis step is performed after isolating the failing paths. In [32], the work from Intel lists some causes of timing failures in modern processor designs. They are dynamic factors, such as multiple input switching, crosstalk noises, and transient voltage droop. Except those, delay deviations in some gates and interconnects can also result in timing failure of fabricated chips [74, 75]. To identify such causes, [32] suggests performing very expensive detailed timing analysis with debug testers. [74,
1.3 Our Contributions

The goal of this thesis is to overcome the aforementioned problems in Chapter 1.2.2. We aim to provide an automatic framework for post-silicon timing diagnosis, which includes failing path isolation and causality analysis.

Fig. 1.2 gives the overview of our proposed framework. This framework mainly includes four steps, which correspond to the four blocks in Fig. 1.2, respectively. The first two steps are proposed to be performed at pre-silicon stage (before fabrication). Step 1 extracts a large number of statistically-critical paths, which may fail the timing after silicon (Block 1 in Fig. 1.2). Step 2 identifies a small set of representative critical paths and builds a model to predict the delays of a large number of statistically-critical paths (Block 2 in Fig. 1.2). The last two steps of our framework are to be performed at the post-silicon stage (after fabrication). Step 3 measures the actual delays of the representative critical paths after fabrication (Block 3 in Fig. 1.2). With these measurements, the post-silicon delays of the statistical critical paths can
be predicted, and the failing paths can get further isolated. Step 4 diagnoses the gates and interconnects in the circuit, which are top candidates to cause the timing violation in the failing paths (Block 4 in Fig. 1.2).

Particularly, Step 3 in our framework (Block 3 in Fig. 1.2) requires inexpensive infrastructures for delay measurement on representative critical paths. At the pre-silicon stage, we identify a small set of representative critical paths and insert existing structures around these paths which allow post-silicon delay measurement, such as the ones developed in [73]. We categorize Step 1, 2 and 4 in this framework as our main contributions and we will elaborate on them in detail as follows:

- **Statistically-Critical Path Extraction:** Here, we define statistically-critical paths as the paths with the highest probability to violate the circuit timing. Therefore, these paths are top candidate failing paths after silicon. To facilitate the extraction of these paths, first, we define the “violation probability” for a gate or interconnect in the circuit to measure the likelihood that a failing path would go through it. The violation probability can get efficiently pre-computed for each gate and interconnect. Then, we introduce the “violation probability” for any arbitrary segment of consecutive gates and interconnects, which evaluates the probability that a failing path would go through them simultaneously. We derive expression for the upper/lower bound of the segment violation probability and show that our bounds can be computed efficiently. Finally, we use these bounds to construct the statistically-critical paths by traversing the circuit once. This work was published in [41].

- **Representative Critical Path Selection for Failing Path Isolation:** To isolate the failing paths among all statistically-critical paths, one approach is
to derive the accurate timing of all these critical paths at the post-silicon stage and compare them with the timing constraint. However, the total number of statistically-critical paths can be extremely large and consequently, we cannot afford to directly measure them after fabrication. Fortunately, we observe that these critical paths always share a lot of gates/interconnects. Moreover, some of these paths may be located very close to each other, which indicates that they may suffer from the same degree of process variations. Due to the above facts, we propose to first select a small set of representative critical paths from the statistically-critical paths at the pre-silicon stage. We measure their delays after fabrication, and use these measurements to predict the post-silicon delays of the remaining statistically-critical paths. Particularly, we utilize a concept known as “effective rank” to help this selection. This work was published in [42].

• **Post-Silicon Segment Diagnosis on Failing Paths:** One important task after isolating failing paths is to identify the causes of these timing failures. Here, our goal is to find the failing segments (consecutive gates and interconnects) on the failing paths, which have their post-silicon segment delays larger than their estimated pre-silicon values. We also aim to accurately rank failing segments based on the degree of their delay deviations. To achieve these goals, we propose to use the delay measurements on the representative critical paths for more accurate diagnosis. Particularly, we formulate an Integer Linear Programming to diagnose the failing segments on failing paths and maximize “diagnostic resolution”. A high diagnostic resolution indicates that our identified failing segments are very likely to fail timing at the post-silicon stage. This work was published in [37].
Note that in this thesis, we assume that the cause of the timing failures is only process variations. We do not consider the dynamic factors, such as crosstalk and power grid noise, which can also result in timing failures. Thus, we focus on process variations as a first step for post-silicon timing diagnosis.

This thesis is organized as follows: Chapter 2 introduces process variation model and some related work on pre-silicon timing analysis. Chapter 3 presents a statistically-critical path extraction technique. The representative critical path selection algorithm for failing path isolation is given in Chapter 4, and a method for segment diagnosis on failing paths is proposed in Chapter 5. The last chapter concludes this thesis.
Chapter 2

Preliminaries

In this chapter, we first study the basic concepts in Static Timing Analysis in Section 2.1. To evaluate the impact of process variations on circuit timing, we introduce the process variation model and the variation-aware gate/interconnect delay models in Section 2.2 and 2.3, respectively. Section 2.4 gives an overview of the existing Statistical Static Timing Analysis approaches to compute the probability density function of the circuit timing under process variations.

2.1 Static Timing Analysis

Static Timing Analysis is to verify the timing behavior of a circuit in the deterministic case (without variations). The STA is static since the timing analysis is performed statically and does not rely on the data patterns applied at the input pins. An alternative approach for timing verification is using simulation. It requires applying a large set of data patterns to the input pins. Therefore, timing simulation is more time-consuming compared to STA.
Figure 2.1: Delay dependence on input patterns in a 2-input NAND gate from [22].

In order to conduct STA, the delay of individual circuit elements (i.e., gates and interconnects) should be incorporated in the analysis. Here, we introduce our notations for gate/interconnect delay modeling, which is used in this thesis:

**Notation 1:** For a $n$-input gate $i$, we denote the delay from its $j$-th input pin to its output pin as $d_{i,j}$ for $j = 1, \ldots, n$.

**Notation 2:** For the interconnect which starts from point $i$ and ends at point $j$, we denote its delay as $d_{i \rightarrow j}$. A point could be gate or primary input and primary output pins.

### 2.1.1 Gate Delay Model

Logic gates are the building blocks of modern VLSI designs. Their delays are highly dependent on slew (also referred as transition time) and their loading capacitance. With the increase in the slew and loading capacitance, the gate delay increases. For example, the simplest gate delay model approximates its delay $d_{i,j}$ using

$$d_{i,j} \approx c_0 + c_1 S_{i,j} + c_2 C_{L,i},$$
where $S_{i,j}$ and $C_{L,i}$ denote the slew at the $j$-th input pin of gate $i$ and its loading capacitance, respectively. The $c_0, c_1, c_2$ terms are positive constants derived from linear regression. We refer to [17] for more complex but accurate gate delay models.

To elaborate the notations, we take Fig. 2.1 as an example. It is a 2-input NAND gate, which is labeled using G0. Its loading capacitance $C_{L,G0}$ is set to 100fF as given in [22]. The notation $d_{G0,1}$ represents the delay of G0 from the input pin P1 to the output pin, and similarly for $d_{G0,2}$. Fig. 2.1 lists these delay values under different input patterns from [22]. We observe that when G0 has falling transition at its input pin, it has larger delays. In addition, the delay of G0, resulting from different input pins but same transition direction, also varies. This difference looks minor in this case, but its extent is in fact highly dependent on the size (channel width and length) of the transistors inside the gate. In this thesis, for each gate, we assume one delay per input pin. Specifically, we consider both falling and rising transition in that input pin and use the maximum delay as the final gate delay caused by that input pin. However, our framework in this thesis can be easily extended to consider more complicated delay models, such as incorporating the transition effects.

2.1.2 Path Delay Computation

To compute the delay of a combinational path $p$, which is denoted as $d_p$, we can simply add up the delays of all gates and the interconnects which form this path, and then use the summation as $d_p$.

Let us take Fig. 2.2 as an example. Consider the data path $p := \text{SDT} \rightarrow \text{G1} \rightarrow \text{G2} \rightarrow \text{G3} \rightarrow \text{FF0}$, and the delays of individual gates and interconnects are marked
Figure 2.2: The computation of the data path delay: $d_{\text{SDT} \rightarrow \text{G1} \rightarrow \text{G2} \rightarrow \text{G3} \rightarrow \text{FF0}}$

in the figure using our notations. We can simply compute $d_p$ using

$$d_{\text{SDT} \rightarrow \text{G1} \rightarrow \text{G2} \rightarrow \text{G3} \rightarrow \text{FF0}} = d_{\text{SDT} \rightarrow \text{G1}} + d_{\text{G1} \rightarrow \text{G2}} + d_{\text{G2} \rightarrow \text{G3}} + d_{\text{G3} \rightarrow \text{FF0}}.$$  \hspace{1cm} (2.1)

More generally, if we consider a segment $s := \text{SDT} \rightarrow \text{G1} \rightarrow \text{G2}$, which is a portion of the above defined path $p$, and we have its delay $d_s$ to be expressed as

$$d_{\text{SDT} \rightarrow \text{G1} \rightarrow \text{G2}} = d_{\text{SDT} \rightarrow \text{G1}} + d_{\text{G1} \rightarrow \text{G2}}.$$  \hspace{1cm} (2.2)

2.1.3 Circuit Block Delay Computation

Circuit blocks are composed of a subset of logic gates and their associated interconnects, possibly with primary input and output pins. For a certain circuit block, we can enumerate all its possible paths and use the maximum path delay as the delay of this circuit block. However, with the increase in the number of gates and intercon-
connects forming this circuit block, the number of possible paths can be extremely large. For example, the circuit C6288 in the ISCAS'85 benchmark suite has more than one million paths.

Before discussing how to compute the delay of circuit blocks more efficiently, we first introduce the following concepts and definitions. We start from the concept of timing graph representation.

**Timing Graph Representation:** We can represent any circuit block using a timing graph $\mathcal{G}$. Let us consider the circuit block given in Fig. 2.3(a), which is composed of 5 gates (G1-G5), 3 primary input pins (PI1-PI3), and 2 primary output pins (PO1 and PO2). Fig. 2.3(b) shows its timing graph. As we can see, every gate, primary input and primary output in Fig. 2.3(a) is represented using a node in Fig. 2.3(b). On the other hand, each interconnect in Fig. 2.3(a) is represented using a directed edge in Fig. 2.3(b). We associate each edge in Fig. 2.3(b) with a weight, which is equal to the summation of the corresponding gate delay and interconnect delay. For example, for the edge connecting G3 and G4 in Fig. 2.3(b), we have its weight expressed as

$$w_{G3,G4} = d_{G3\rightarrow G4} + d_{G4,2}.$$  \hspace{1cm} (2.3)
With the above weight definition, we can rewrite the path delay in Eq. (2.1) as

\[ d_{\text{SDT} \rightarrow G1 \rightarrow G2 \rightarrow G3 \rightarrow \text{FF0}} = w_{\text{SDT}, G1} + w_{G1, G2} + w_{G2, G3} + w_{G3, \text{FF0}}, \]

which is equal to the summation of the weights of the edges forming that path. Moreover, we add two super nodes SPI and SPO, which have direct connections with all primary inputs and primary outputs, respectively. In this example, we associate these connected edges with their weights to be zero. However, based on the definition for the input and output delays, we may have non-zero weights.

For each node \( i \) in timing graph \( \mathcal{G} \), such as Fig. 2.3(b), we define its Arrival Time (\( AT_i \)) and Reverse Arrival Time (\( RAT_i \)) as follows:

**Definition 2.1** \( AT_i \): It represents the maximum time to propagate a signal from the super primary input SPI to node \( i \).

**Definition 2.2** \( RAT_i \): It represents the maximum time to propagate a signal from node \( i \) to the super primary output SPO.

Let us consider the timing graph \( \mathcal{G} \) as given in Fig. 2.3(b). We assume that all edge weights in \( \mathcal{G} \) are given. Take G5 as an example, once we derive \( AT_{G1} \) and \( AT_{G2} \), we can instantly compute the arrival time at G5 as

\[ AT_{G5} = \max(AT_{G1} + w_{G1,G5}, AT_{G2} + w_{G2,G5}) \]

Following the same procedures, we can compute the arrival times at SPO and further conclude that this circuit block has delay \( d_{\text{clk}, \text{blk}} \) to be equal to \( AT_{\text{SPO}} \).
Similarly, we can compute the reverse arrival times at G2 using

\[ RAT_{G2} = \max(RAT_{G5} + w_{G2,G5}, RAT_{G4} + w_{G2,G4}) , \]

and finally compute the delay of the circuit block using \( d_{ckt,blk} = RAT_{SPI} \).

Motivated by the above example, we can conclude by traversing \( G \) through its topological order from super primary input SPI to super primary output SPO, we can compute the circuit block delay in linear runtime complexity \( O(m + n) \), where \( m \) and \( n \) denote the number of edges and nodes in \( G \), respectively. For the circuit block containing sequential elements, such as flip-flops and latches, we need to consider setup time and hold time check, which are not incorporated in this thesis. Please refer to [17] for details.

So far, we have discussed the basic concepts in Static Timing Analysis. In the following sections, we begin investigating the impact of process variations on STA by first introducing the process variation modeling.

### 2.2 Process Variation Model

Process variations become increasingly important with the scaling of feature sizes. The variations in one device parameter can be further broken into die-to-die (D2D) and within-die (WID) components. Specifically, D2D variations are the variations from lot to lot, wafer to wafer or die to die, while WID variations correspond to the variations within a single die. D2D variations affect all the devices on the same die in the same way, e.g., making all the transistor lengths on the same chip to be larger or smaller than the nominal values. WID variations may affect different devices on
the same chip in different ways, e.g., causing some transistors to have smaller-than-nominal gate lengths and other transistors have larger-than-nominal gate lengths.

In the past, D2D variations were the main concern and WID variations can be safely neglected for CMOS digital circuit design. However, with the scaling of sub-micron technology, the WID variations becomes more important. For example, for 130nm CMOS technology, the percentage of the total variation of the effective channel length that can contribute to WID variation can be up to 35% [47].

Mathematically, for any circuit element $i$ (i.e., gates and interconnects), we model its variation in certain parameter $P$ (such as effective channel length $L_{eff}$ and zero-bias threshold voltage $V_{th0}$ in a gate, or wire width $W_{int}$ and thickness $T_{int}$ in an interconnect) using

$$\Delta P(i) = P(i) - P_0(i) = \Delta P_{D2D} + \Delta P_{WID}(i), \quad (2.4)$$

where $P_0(i)$ is the nominal value of device parameter $P$ for circuit element $i$.

Similar to [3, 8, 12, 14, 47], we make the following assumptions in modeling variations in parameter $P$ in a gate or interconnect:

**Assumption 1:** We consider process variations in the following device parameters: effective channel length $L_{eff}$, zero-bias threshold voltage $V_{th0}$, dopant fluctuations $F_{dopt}$, wire width $W_{int}$, wire thickness $T_{int}$, and via resistance $R_{via}$. We assume that process variations in $L_{eff}$, $V_{th0}$, $W_{int}$, and $T_{int}$ follow Gaussian distribution, and those in $F_{dopt}$ and $R_{via}$ follow Log-Normal distribution [15].

**Assumption 2:** The process variations in the above parameters are independent from each other. For example, we have $\Delta L_{eff}(i)$ and $\Delta V_{th0}(j)$ to be
independent from each other no matter \( i \) is equal to \( j \) or not.

**Assumption 3:** For different circuit elements \( i \) and \( j \) with respect to the same parameters \( P \), we use the correlation coefficient between \( \Delta P_{WID}(i) \) and \( \Delta P_{WID}(j) \), denoted as

\[
\rho_P(i, j) \triangleq \text{Corr}(\Delta P_{WID}(i), \Delta P_{WID}(j)),
\]

(2.5)

to capture their spatial correlations, which we will illustrate in Section 2.2.1.

### 2.2.1 Spatial Correlation Model

In this section, we discuss the modeling of the spatial correlation between \( \Delta P_{WID}(i) \) and \( \Delta P_{WID}(j) \) for different circuit elements \( i \) and \( j \). Due to the intrinsic characteristics of the manufacturing process, the \( \rho_P(i, j) \) in Eq. (2.5) is highly dependent on the physical locations of circuit elements \( i \) and \( j \). Here, we mainly describe two most-frequently used models, which are as follows.

**Distance-based Correlation Model:** This model is used in [54, 58, 61]. It is based on the separation distance of the circuit elements. Specifically, we first partition the physical region of each die into \( n_{row} \times n_{col} \) grids. If the circuit elements are close to each other, they are more likely to have similar device parameters compared to those, which are located far away. We assume perfect correlation among the circuit elements in the same grid, high correlations among those in close grids, and low or zero correlations in the far-away grids. For example, G1 and G2 are located in the same grid, and we assume that their parameter variations are always identical. G1 and G3 are located in neighboring grids, and their parameter variations are not
identical but highly correlated due to their spatial proximity. G4 is located further away from G1 compared to G3, and the correlation between the parameter variation in G4 and G1 is smaller than that between the parameter variation in G1 and G3. Since G1 and G5 are very far away from each other, we assume that the parameter variations in G1 and G5 are independent.

More specifically, [58] gives an example expression for $\rho_P(i, j)$ in Eq. (2.5):

$$
\rho_P(i, j) = \begin{cases} 
1.0, & \text{if } d^2(i, j) = 0 \\
0.7, & \text{if } d^2(i, j) = 1 \\
0.4, & \text{if } d^2(i, j) = 2 \\
0, & \text{if } d^2(i, j) > 2 
\end{cases}
$$

(2.6)

where $d(i, j)$ is the distance between the grids where circuit elements $i$ and $j$ are

Figure 2.4: Grid based spatial correlation model for WID variations
Figure 2.5: A two-level hierarchical spatial correlation model for WID variations

located, and can be computed using

\[ d^2(i, j) = (C_x(i) - C_x(j))^2 + (C_y(i) - C_y(j))^2, \]

where \((C_x(i), C_y(i))\) denotes the coordinate of the grid where circuit element \(i\) is located. For example, G1 and G2 are located in the region with coordinate (1, 1). Therefore, we have \(C_x(G1) = C_x(G2) = C_y(G1) = C_y(G2) = 1\). G3 is located in the region with coordinate (G1, G2), and thus we have \(C_x(G3) = 1\) and \(C_y(G3) = 3\). Consequently, we have \(d^2(G1, G3) = 1\) and the correlation coefficient between \(\Delta P_{WID}(G1)\) and \(\Delta P_{WID}(G3)\), denoted as \(\rho_P(G1, G3)\), is equal to 0.7. Similarly, we have \(\rho_P(G1, G4) = \rho_P(G1, G5) = 0\).

**Hierarchical Correlation Model:** The distance-based correlation model illustrated above assumes that all circuit elements located in the same region would have the same process variation in the same device parameter. However, this model cannot capture the random variations associated with each circuit elements in the same region. To be more precise and general, [3] propose a hierarchical spatial correlation
In this hierarchical model, we first divide the physical area of the die into regions using an \( L \)-level quad-tree partitioning. At each level \( l \in \{1, \ldots, L\} \), we partition the die into \( 2^l \times 2^l \) squares. For example, for the first or top level, we divide the entire die into 4 regions, while the last or the bottom level \( L \), the die is composed of \( 4^L \) regions. Based on the accuracy requirement, we can adjust \( L \) to be an appropriate value. We then associate an independent random variable \( \Delta P_{(l,r)} \) to each region \( (l,r) \) to represent a component of the total WID variation in device parameter \( P \), where \( l \) denotes the level, and \( r \) denotes the region index in level \( l \). We finally express the WID variation and the total variation of circuit element \( i \) with respect to the device parameter \( P \) using

\[
\Delta P_{WID}(i) = \sum_{l,r|\in\text{region}(l,r)} \Delta P_{(l,r)} + \Delta P_R(i),
\]

\[
\Delta P(i) = \Delta P_{D2D} + \sum_{l,r|\in\text{region}(l,r)} \Delta P_{(l,r)} + \Delta P_R(i),
\]

where \( \Delta P_R(i) \) denotes the random process variation for the circuit element \( i \) and can also be considered as the mismatch between the pre-silicon and post-silicon process variation model. We assume that \( \Delta P_R(i) \) for different \( i \)s are independent from each other, and are also independent from \( \Delta P_{D2D} \) and \( \Delta P_{(l,r)} \).

Let us take G1 in Fig. 2.5 as an example, and we assume that a two-level hierarchical model is used. In this case, we have \( L = 2 \), and we can rewrite the total variation for G1 in device parameter \( P \) using

\[
\Delta P(G1) = \Delta P_{D2D} + \Delta P_{(1,1)} + \Delta P_{(2,2)} + \Delta P_R(G1),
\]
where $\Delta P_{D2D}$, $\Delta P_{(1,1)}$, $\Delta P_{(2,2)}$, and $\Delta P_R(G1)$ are all independent random variables.

Remark: Both the distance-based correlation model in Eq. (2.6) and the hierarchical correlation model in Eq. (2.7) require separating each die into grids. While process variation in device parameter $P$, such as $L_{eff}$ and $V_{th0}$, follows Gaussian distribution, we can apply Principal Component Analysis to help guide the grid separation. While process variation in device parameter $P$, such as $R_{via}$ and $F_{dopant}$, follows non-Gaussian distribution, we can use Independent Component Analysis instead. We refer the readers to [14, 23] for more details.

### 2.3 Gate/Interconnect Delay Model Incorporating Process Variations

Let us consider a chip with $N$ circuit elements (i.e., gates and interconnects). We assume process variations in $K$ types of device parameters $P^1, P^2, \ldots, P^K$, where $P^i$ can be effective channel length $L_{eff}$, zero-bias threshold voltage $V_{th0}$, etc for $i = 1, 2, \ldots, K$. For circuit element $i$ (a gate or interconnect), from Taylor series, we can express its variation-aware delay using

$$d(i) = \mu(i) + \sum_{k=1}^{K} (a_k(i) \cdot \Delta P^k(i)) + \sum_{k=1}^{K} \sum_{l=1}^{K} (b_{k,l}(i) \cdot \Delta P^k(i) \cdot \Delta P^l(i)) + \cdots , \quad (2.8)$$

where $\mu(i)$ denotes the nominal delay of circuit element $i$ (without variations). The $a_k(i)$ and $b_{k,l}(i)$ terms are equal to the 1-st and 2-nd order partial derivatives with respect to certain device parameters, respectively. They can be derived by fitting using the transistor-level simulation data.
In most of the existing works such as [7, 14, 23, 25, 55, 69], the variation-aware delay of circuit element $i$ (a gate or interconnect) is approximated using

$$d(i) \approx \mu(i) + \sum_{k=1}^{K} (a_k(i) \cdot \Delta P_k(i)),$$  \hspace{1cm} (2.9)

which is shown to have very high accuracy.

Let us further assume that the $L$-level hierarchical model in Section 2.2 can accurately capture the spatial correlation among the process variations in these $N$ circuit elements. We can then plug Eq. (2.4) and Eq. (2.7) into Eq. (2.9) and derive

$$d(i) \approx \mu(i) + \sum_{k=1}^{K} \left( a_k(i) \cdot \left( \Delta P_{D2D}^k + \sum_{l,r|\in \text{region}(l,r)} \Delta P_{(l,r)}^k + \Delta P_R^k(i) \right) \right)$$

$$= \mu(i) + \sum_{k=1}^{K} \left( a_k(i) \cdot \left( \Delta P_{D2D}^k + \sum_{l,r|\in \text{region}(l,r)} \Delta P_{(l,r)}^k \right) \right) + \sum_{k=1}^{K} (a_k(i) \cdot \Delta P_R^k(i))$$

$$\hspace{1cm} (2.10)$$

For a certain circuit element $i$, we know that $\Delta P_R^k(i)$ are independent from each other among different $k$. Thus, we can define a new random variable $R(i)$ as

$$R(i) \triangleq \sum_{k=1}^{K} (a_k(i) \cdot \Delta P_R^k(i))$$

and then rewrite Eq. (2.10) using

$$d(i) \approx \mu(i) + \sum_{k=1}^{K} \left( a_k(i) \cdot \left( \Delta P_{D2D}^k + \sum_{l,r|\in \text{region}(l,r)} \Delta P_{(l,r)}^k \right) \right) + R(i).$$  \hspace{1cm} (2.11)
Let us group all random variables which are irrelevant with \( i \) (i.e., \( \Delta P_{D2D}^k, \Delta P_{(l,r)}^k \) for all \( k = 1, 2, \ldots, K \)) into a column vector \( P \). All the remaining random variables (i.e., \( R(i) \) for all \( i \)) are grouped into another column vector \( R \). We further define \( X \triangleq [P, R] \). Then, we can rewrite Eq. (2.11) into

\[
d(i) \approx \mu(i) + \mathbf{a}^T(i) P + R(i) = \mu(i) + \mathbf{a}^T(i) X. \quad (2.12)
\]

where \( \mathbf{a}(i) \) and \( \mathbf{a}(i) \) are columns vector, representing the sensitivity of \( d(i) \) with respect to \( P \) and \( X \), respectively.

The \( d(i) \) in Eq. (2.12) is the delay of circuit element \( i \), which can be a gate delay or interconnect delay. More specifically, for delay of gate \( i \) from input pin \( j \), denoted as \( d_{i,j} \), and delay of interconnect between two points \( i \) and \( j \), denoted as \( d_{i\rightarrow j} \), we use the following notations:

\[
d_{i,j} \approx \mu_{i,j} + \mathbf{a}_{i,j}^T X, \quad d_{i\rightarrow j} \approx \mu_{i\rightarrow j} + \mathbf{a}_{i\rightarrow j}^T X, \quad (2.13)
\]

to express the variation-aware gate/interconnect delay in this thesis.

### 2.4 Statistic Static Timing Analysis

Statistical Static Timing Analysis aims to compute the PDF of the circuit timing assuming that the exact distribution of process variations (\( X \) in Eq. (2.13)) is available.

Eq. (2.1) and Eq. (2.2) have shown that both path delay and segment delay are equal to the summation over the weights of the associated edges on the circuit timing graph as defined in Section 2.1.3. Therefore, once we derive the variation-
 aware gate/interconnect delay models such as Eq. (2.13), we can instantly derive the variation-aware expressions for the edge weights using Eq. (2.3). The delays of the path $p$ and segment $s$ can be further written as

$$d_p = \mu_p + a_p^T X, \quad d_s = \mu_s + a_s^T X,$$

where $\mu_p$ and $\mu_s$ are the nominal path and segment delay (without variations), respectively. The $a_p$ and $a_s$ are sensitivity vectors of path delay $d_p$ and segment delay $d_s$ with respect to process variations $X$, respectively. These sensitivity vectors can be simply derived by adding the associated $a_{i,j}$ and $a_{i\rightarrow j}$ defined in Eq. (2.13) together.

Even after we derive the delay models for paths/segments, it is still very difficult to obtain the model for circuit delay. First, it is nearly impossible to enumerate all critical paths for current modern VLSI designs. Second, the circuit delay is defined to be equal to the maximum over all (critical) path delays. However, it is difficult to handle the MAX operation on the variation-aware delay expressions. In the remaining of this section, we will describe two types of SSTA approaches which can overcome the aforementioned problems.

### 2.4.1 Monte Carlo based Statistical Static Timing Analysis

The most straightforward approach to accomplish SSTA is to implement Monte Carlo (MC) simulation. With the available PDF of process variations in device parameters, we can draw a large number of samples. For each sample, we compute the circuit delay using fast block based static timing analysis explained in Section 2.1.3. By evaluating these circuit delay samples, we can build the PDF for the circuit timing. We can also
compute the *timing yield* using the fraction of the number of samples meeting a given timing constraint out of the total number of samples in MC simulation.

Since the number of process variations (i.e., $|X|$, the dimension of $X$ in Eq. (2.13)) is always very large, the sampling method in MC based SSTA becomes extremely important. Latin Hypercube Sampling or stratified sampling methods are investigated to accelerate the MC based SSTA approaches [6, 68]. An alternative approach to reduce the runtime complexity is to perform adjustment-based static timing analysis, which is proposed by us in [38]. Its intuition is that much fewer MC samples are required to build an adjustment model from approximate circuit timing to accurate ones, compared to directly building models of accurate circuit timing with respect to process variations. It also assumes that it is very cheap to derive approximate circuit timing. More recently, MC-based SSTA is performed on Graphic Processing Unit (GPU) and Field Programmable Gate Array (FPGA) to accelerate the runtime speed [18, 28].

### 2.4.2 Parameterized Statistical Static Timing Analysis

Different from MC based SSTA, Parameterized SSTA approaches mainly works on the analytical computation of the arrival times defined in Section 2.1.3. For example, [15] approximates all $AT_i$s under process variations as the canonical form below:

$$AT_i = \mu_i + f(P) + R_{AT_i}, \quad (2.15)$$

where $P$ is defined as in Eq. (2.12), $R_{AT_i}$ is a random variable specially associated with $AT_i$ and independent from all elements in $X$ defined in Eq. (2.12), and $f(\cdot)$ is a
deterministic function to capture the relationship between the arrival time $AT_i$ and process variations $P$. Generally, we set $f(\cdot)$ to be a low-order polynomial function, such as linear, quadratic, and cubic function.

To derive $f(\cdot)$ in Eq. (2.15) accurately and efficiently, we recall Eq. (2.4) and further express $AT_i$ as

$$AT_i = \max_{j \in \text{Fanin}(i)} (AT_j + w_{ji}), \quad (2.16)$$

where $\text{Fanin}(i)$ represents the set of nodes in the circuit timing graph, which are the fanin of node $i$.

The equality in Eq. (2.16) indicates that we need to work on ADD and MAX operations over the canonical expressions, which are in form of Eq. (2.15). Specially, ADD is a linear operation and we can handle it by simply adding the edge weight to the corresponding arrival time terms. However, the MAX operation is much more difficult. In this section, we only discuss the case when $X$ in Eq. (2.13) follows multivariate Gaussian distribution. The authors in [7] first make the following approximations: in Eq. (2.15), $f(\cdot)$ is a linear function and thus $AT_i$ also follows Gaussian distribution. Then, they use Clark’s method [33] to solve MAX operation. Briefly explaining, given two Gaussian random variables $\xi_1$ and $\xi_2$, they can approximate the maximum between $\xi_1$ and $\xi_2$, defined as $\xi$, using

$$\xi \triangleq \max(\xi_1, \xi_2) \approx t\xi_1 + (1-t)\xi_2 + \xi_3 \quad (2.17)$$
where $t$ is the tightness probability defined as

$$
t \triangleq \int_{-\infty}^{\lambda} \frac{1}{\sqrt{2\pi}} \exp \left( -\frac{x^2}{2} \right) dx \quad (2.18)
$$

and $\xi_3$ is an additional Gaussian random variable to ensure that this approximation has the same mean and covariance as those of $\xi$. In addition, in Eq. (2.18), $\lambda$ is defined as $\mu_{\xi_1-\xi_2}/\sigma_{\xi_1-\xi_2}$, where $\mu_{\xi_1-\xi_2}$ and $\sigma_{\xi_1-\xi_2}$ denote the mean and standard deviation of $\xi_1-\xi_2$, respectively. The Gaussian approximation in Eq. (2.17) makes MAX operation easy to implement, but suffers from high error. The authors in [43] pointed that the Clark’s approximation introduces high error when $\xi_1$ and $\xi_2$ have very similar mean but very different variances, and when $\xi_1$ and $\xi_2$ have very similar mean but highly negative correlation.

For more general cases, we refer the readers to [15, 23, 35] for accurate and sophisticated statistical computational methods to handle with statistical MAX operations.
Chapter 3

Statistically-Critical Path Extraction Under Process Variations

In this chapter, we introduce a bound-based approach to extract a pre-specified number of statistically-critical paths under process variations. These are the paths with the highest “violation probability”, which indicates the probability that a path would violate the given timing constraint. Specifically, our approach requires the pre-computation of the violation probability of all the nodes and edges in the circuit timing graph, which can be done efficiently using two rounds of Statistical Static Timing Analysis. Given these node/edge violation probabilities, we derive tight upper and lower bounds for any arbitrary segment of consecutive nodes and edges, which is our major contribution in this chapter. We further utilize these bounds to extract the statistically-critical paths and show constant-time for incremental update of the
bounds when extending a segment to a longer one. If our goal is to extract the single most statistically-critical path, we show a bound based reduction that can prune a large portion of circuit without losing optimality. In our simulations, we verify the correctness and accuracy of our bounds for individual paths, and compare with exact path extraction using Monte Carlo-based simulation, and an alternative which incorporates path-based Statistical Static Timing Analysis.

3.1 Introduction

Due to the existence of process variations, both deterministic and corner-based static timing analysis cannot provide a good candidate pool of paths, which have a higher probability to violate a given timing constraint under process variations. The paths that are critical in the deterministic case or at one process corner may not be critical in other points of the process variation space. More importantly, some gates and interconnects may have quite similar deterministic delays, however, due to their types or locations on the chip, they may have quite different delay sensitivity with respect to process variations.

At the pre-silicon stage, a large variety of variation-aware analysis and optimization tasks require the extraction of statistically-critical paths. These extracted paths can help to evaluate the impact of false paths under process variations [19, 39]. They can also be optimized to improve the circuit performance under process variations, such as the circuit timing yield [5, 49]. At the post-silicon stage, the identification of such statistically-critical paths can help to isolate the failing paths for post-silicon repair as we will show in Chapter 4.
A group of related works focus on the area of at-speed test to minimize the number of critical paths for post-silicon delay test [24, 34, 66, 70, 72]. The most recent work in this application is [69]. Specifically, the authors in [69] propose a branch-and-bound approach to select a minimum number of critical paths to cover the entire process variation space. At the post-silicon stage, by only testing these selected paths, they can identify whether the chip fails the timing constraint. However, the paths identified using the above-mentioned techniques cannot be used for timing-based optimization, or to incorporate the impact of false paths in timing yield analysis.

In this chapter, we target the problem of extracting the top $n$ statistically-critical paths under process variations, where $n$ can be provided as an input parameter. These paths have the highest “violation probability”, which is defined as the probability that a path fails the timing constraint.

To solve this problem, we first introduce the definitions of “node/edge violation probability”, which is the probability that a node/edge falls on a path which may violate timing constraint. We show that these violation probabilities can be computed as a pre-processing step with low computational complexity (using two rounds of SSTA followed by the probability calculation). We further derive the lower and upper bounds for violation probability of any arbitrary segment, which reflects the likelihood that this segment is a subset of a path which may violate the timing constraint. The computation of these bounds only requires node/edge violation probabilities and can be incrementally updated as the segment gets expended. We finally propose a bound-based approach to extract the top $n$ statistically-critical paths with only one traversal of the circuit timing graph.
We summarize our contribution as follows:

- We first study the case of a segment which is constructed of only two connected edges but may intersect with the other paths in the circuit. We compute bound expressions for its violation probability, from the violation probabilities of its nodes and edges.

- We then incrementally compute the bounds of a longer segment after adding an edge to the previous segment with constant time complexity.

- For the case to find a single path with the highest violation probability, we introduce a pruning scheme, which can significantly reduce the size of the timing graph without loosing optimality.

- Our derived lower/upper bound for the segment violation probability is independent of the distributions of process variations, assuming the node/edge violation probabilities are available.

Simulation results show that our algorithm has fast runtimes with an error of about 2% in cumulative violation probability compared to exhaustive path extraction using Monte Carlo-based SSTA.

The organization of this chapter is as follows. Section 3.2 gives the formal definition of our statistically-critical path extraction problem. Section 3.3 discusses the pre-computation of node/edge violation probabilities. We show incremental computation of bounds for violation probability of two- and $K$-connected edges in Sections 3.4 and 3.5, respectively. Our proposed bound-based approach is given in Section 3.6. We further discuss how to extract a single statistically-critical path in Section 3.7. Simulation results are presented in Section 3.8 followed by conclusions.
3.2 Problem Definition

Given a circuit timing graph $\mathcal{G}$ as defined in Section 2.1.3, we denote its node set and edge set as $\mathcal{N}$ and $\mathcal{E}$, respectively. For each node $n_i \in \mathcal{N}$, we further define the arrival time $AT_i$ (reverse arrival time $RAT_i$) as the maximum time to propagate a signal from any primary input (output) to this node. These definitions were given in Section 2.1. For a given timing constraint $T_0$ and any path/segment $(p/s)$ connecting an arbitrary sequence of nodes such as $n_1 \rightarrow n_2 \rightarrow \ldots \rightarrow n_K$, we define its violation probability $C_{p/s}$ (also written as $C_{1,2,...,K}$) using

$$C_{p/s} \triangleq Pr(D_{p/s} \geq T_0) = Pr(AT_1 + \sum_{i=2}^{K} w_{i-1,i} - 1 + RAT_K \geq T_0), \quad (3.1)$$

where $w_{i-1,i}$ is the variation-aware weight associated with the edge $e_{i-1,i}$ connecting $n_{i-1}$ and $n_i$ as given in Fig. 2.3, and $D_{p/s}$ (also written as $D_{1,2,...,K}$) signifies the maximum path delay passing path/segment. Thus, we can consider $C_{p/s}$ as the probability that path/segment violates the timing constraint $T_0$. Specially, for a segment, its violation probability indicates the likelihood for it to fall on a longer path which fails the timing constraint.

In this chapter, we aim to extract the top $n$ statistically-critical paths with the highest $C_p$ indicating that they have the highest probability to violate $T_0$. We formulate this problem as follows:

**Problem Definition:** Given the timing constraint $T_0$ and a circuit timing graph $\mathcal{G}$ with nodes $\mathcal{N}$ and edges $\mathcal{E}$ as defined in Section 2.1.3, we aim to extract a set of $n$ paths, denoted by $\mathcal{P}_c$, which have the largest $C_p$ among all possible paths in $\mathcal{G}$. Note that the number of target paths $n$ can be arbitrarily provided.
The above problem definition indicates that the paths in the set \( \mathcal{P}_c \) are more likely to fail the timing after considering the impact of process variations. Therefore, we can incorporate the solution of this to improve the pre-silicon timing analysis and optimization, such as variation-aware timing-based optimization (by focusing on optimizing the paths in \( \mathcal{P}_c \)). In Chapter 4, we will show how this problem can also be served as the first step in predicting the failing paths at the post-silicon stage.

### 3.3 Node/Edge Violation Probability

In this section, we define the node/edge violation probability, which is applied as a pre-processing step and can help extract the statistically-critical paths. We also discuss a technique to compute these probabilities.

#### 3.3.1 Definition and Computation

To simplify the extraction of top \( n \) statistically-critical paths, we introduce the following definitions:

**Definition 3.1 Node Violation Probability:** For a given timing constraint \( T_0 \) and any node \( n_i \in \mathcal{N} \), we define its violation probability \( C_i \) as

\[
C_i \triangleq \Pr(D_i \geq T_0) = \Pr(AT_i + RAT_i \geq T_0)
\]  

(3.2)

where \( D_i = AT_i + RAT_i \) signifies the maximum path delay going through the node \( n_i \).

**Definition 3.2 Edge Violation Probability:** For a given timing constraint \( T_0 \) and any
edge \( e_{i,j} \in \mathcal{E} \), which connects \( n_i \) and \( n_j \), we define its violation probability \( C_{i,j} \) as:

\[
C_{i,j} \triangleq Pr(D_{i,j} \geq T_0) = Pr(AT_i + RAT_j + w_{i,j} \geq T_0)
\]  

(3.3)

where \( D_{i,j} = AT_i + RAT_j + w_{i,j} \) signifies the maximum path delay going through the edge \( e_{i,j} \).

The computations of \( C_i \) and \( C_{i,j} \) defined in Eqs. (3.2) and (3.3) require two tasks: 1) applying block-based SSTA to compute (reverse) arrival time expressions of \( AT_i \) and \( RAT_j \), 2) probability computation. For the former, any block-based SSTA which generates variation-aware (reverse) arrival time expressions can be used. Please refer to Section 2.4.2 for details. For the latter, since the node/edge violation probability is analogous to the timing yield(-loss) of a circuit, any yield computation technique, such as [12, 30, 45], can be utilized.

In the following discussions, we assume that all variation-aware expressions (i.e., edge weight \( w_{i,j} \), (reverse) arrival times) are in quadratic functions with respect to process variations \( X \), which is a valid assumption as reviewed in Section 2.4.2. Consequently, the terms \( D_{p/s}, D_i \), and \( D_{i,j} \) defined in Eqs. (3.1)-(3.3) are also in quadratic relationship with respect to \( X \). We take \( D_i \) as an example to review one type of probability computation techniques known as Pearson Curve.

We describe \( D_i \) as follows

\[
D_i = \mu_i + \sum_{j=1}^{m} b_{i,j}(X_j + c_{i,j})^2,
\]  

(3.4)
where \( m \) is equal to the dimension of process variations \( X \) and \( X_j \) denotes the \( j \)-th element in \( X \). The \( \mu_i, b_{i,j} \) and \( c_{i,j} \) are constants and can be derived by applying existing parameterized SSTA approaches such as [15, 23].

Let us further assume that \( X \) follows a multivariate Gaussian distribution (similar to [16, 24, 25, 49]). We can compute the probability of \( Pr(D_i \geq T_0) \) by considering the following two cases:

- **Case 1:** When all \( b_{i,j} \)s are non-negative for \( j = 1, 2, \ldots, m \), \( D_i \) follows Chi-Square distribution [11]. We can exactly compute this violation probability analytically from the expression of Chi-Square distribution.

- **Case 2:** For the remaining cases, we can utilize Pearson Curve proposed in [33] to approximately compute the violation probability. The authors of [33] first compute the percentile points of \( D_\alpha \) for several different \( \alpha \)s, which are defined as the minimum value of \( d \) such that \( Pr(D_i < d) \geq \alpha \). Specifically, the \( \alpha \)-percentile points of the random variable \( D_i \) can be approximated by using

\[
D_\alpha = \frac{\pi_1(\sqrt{\beta_1, \beta_2})}{\pi_2(\sqrt{\beta_1, \beta_2})} \quad \text{where} \quad \pi_i(\sqrt{\beta_1, \beta_2}) = \sum \sum_{0 \leq r+s \leq 3} b_{r,s}^i (\sqrt{\beta_1})^r (\sqrt{\beta_2})^s, \quad r \text{ and } s \text{ are positive integers.}
\]

The values of \( b_{r,s}^i \) at different \( \alpha \)s can be found via a pre-stored \( 10 \times 20 \) lookup table. Then, interpolation is used to compute the violation probability of \( Pr(D_i \geq T_0) \), which proves to be very accurate in [33].

### 3.3.2 Computational Complexity

To compute the (reverse) arrival time expressions for all the nodes and edges, we only need two rounds of SSTA; one forward and one backward directed traversal of the circuit timing graph as defined in Section 2.1.3. Block-based SSTA techniques have a
linear complexity in the number of nodes and edges in the timing graph. Regardless of which SSTA technique is used, after only two rounds, all the (reverse) arrival time expressions can be computed.

Next, the violation probability should be computed at each node/edge based on its variation-aware (reverse) arrival time expressions. This can be done exactly once at each node/edge and its complexity is similar to the yield-loss computation techniques such as [12, 45]. Moreover, the violation probability computations are independent from each other, and thus can be done simultaneously and in parallel for all the nodes/edges. For the quadratic approximation case based on Pearson Curve, the table lookup allows the probability computation with constant complexity for each node/edge.

The computation of node/edge violation probabilities is a pre-processing step for our path extraction technique. We require node/edge violation probability values as the input to generate bounds on paths and then integrate these in path extraction. So in this context, our techniques will not be limited and can incorporate any existing SSTA technique which can be based on linear/nonlinear variation-aware gate/interconnect delay models with either Gaussian or non-Gaussian process variations.
3.3.3 Comparison with Criticality Probability

The definition of violation probability is different from the criticality probability given in \[16, 24, 25, 49, 66\]. To compare these two probabilities, we consider the example circuit given in Fig. 3.1. The violation probability of \(n_6\) is given by:

\[
C_6 \triangleq Pr(D_6 \geq T_0) = Pr(AT_6 + RAT_6 \geq T_0)
\]

The criticality probability at node \(n_6\) is defined with respect to a cutset in the timing graph which includes \(n_6\). For example, one (smallest) cutset at \(n_6\) is given by \(\{n_6, e_{1,7}, e_{2,7}\}\). For this cut-set, the criticality probability of \(n_6\) denoted by \(Cr_6\) is defined as \[16\]:

\[
Cr_6 \triangleq Pr \left( (D_6 \geq D_{1,7}) \cap (D_6 \geq D_{2,7}) \right)
= Pr \left( AT_6 + RAT_6 \geq \max (AT_1 + RAT_7 + w_{1,7}, AT_2 + RAT_7 + w_{2,7}) \right)
\]

In other words, criticality probability is the likelihood that a path passing through \(n_5\) has the maximum delay compared to the set of “complementary paths” that do not pass from \(n_5\) (passing through \(e_{1,7}\) and \(e_{2,7}\)).

As we can see from the above example, computing the criticality probability \(Cr_6\) implies doing more MAX operations than violation probability \(C_6\). This is because \(C_6\) only requires one comparison with timing constraint while \(Cr_6\) requires multiple comparisons with \(D_{1,7}\) and \(D_{2,7}\), the number of which depends on the cutset size and can grow to be very large in larger examples. The work in \[66\] shows that the criticality probability can still be computed with linear computational complexity.
However, apart from the runtime, the need for additional MAX operations can imply a potentially higher approximation error in accuracy than only one MAX operation [16].

3.4 Bound Computation of Violation Probability for Two-Connected Edges

We first consider two-connected edges as shown in Fig. 3.2. We assume three nodes $n_1$, $n_2$, and $n_3$ are connected by directed edges $e_{1,2}$ and $e_{2,3}$. They form a simple segment $s_{1,2,3}$. Note that we assume that there might be many other incoming or outgoing edges to/from $n_2$ except $n_1$ and $n_3$. In this section, we are interested in computing the upper and lower bounds of the violation probability for segment $s_{1,2,3}$, which is denoted as $C_{1,2,3}$ and defined the same as in Eq. (3.1).

Before the discussion on how to compute the lower/upper bound of $C_{1,2,3}$, we introduce the following theorem and analyze the special case when the intermediate node $n_2$ has single fanin or fanout.

**Theorem 3.1** For the segment $s_{1,2,3}$ formed by edges $e_{1,2}$ and $e_{2,3}$, we have its viola-
tion probability satisfying

\[ C_{1,2,3} = C_{1,2} + C_{2,3} - C_2 + I_1 - I_2, \]  \hspace{1cm} (3.5)  

where

\[
\begin{align*}
I_1 &= Pr\left( (D_2 \geq T_0) \cap (D_{1,2} < T_0) \cap (D_{2,3} < T_0) \right), \\
I_2 &= Pr\left( (D_{1,2} \geq T_0) \cap (D_{2,3} \geq T_0) \cap (D_{1,2,3} < T_0) \right),
\end{align*}
\]

and \( D_i, D_{i,j}, D_{i,j,k} \) express the maximum path delay going through node \( n_i \), edge \( e_{i,j} \), and segment \( s_{i,j,k} \).

**Proof** From basic probability theory [11], we know

\[
C_{1,2} + C_{2,3} = Pr\left((D_{1,2} \geq T_0) \cup (D_{2,3} \geq T_0)\right) + Pr\left((D_{1,2} \geq T_0) \cap (D_{2,3} \geq T_0)\right). \hspace{1cm} (3.6)
\]

Then, we can replace the first term on the right hand side of Eq. (3.6) using

\[
Pr\left((D_{1,2} \geq T_0) \cup (D_{2,3} \geq T_0)\right) = C_2 - Pr\left((D_{2} \geq T_0) \cap (D_{1,2} < T_0) \cap (D_{2,3} < T_0)\right).
\]

Similarly, we can replace the second term on the right hand side of Eq. (3.6) using

\[
Pr\left((D_{1,2} \geq T_0) \cap (D_{2,3} \geq T_0)\right) = C_{1,2,3} + Pr\left((D_{1,2} \geq T_0) \cap (D_{2,3} \geq T_0) \cap (D_{1,2,3} < T_0)\right).
\]

Then, the proof completes. \( \Box \)
Fig. 3.3 elaborates Theorem 3.1. It labels $C_2, C_{1,2}, C_{2,3}$ and $C_{1,2,3}$. The vertical and horizontal shaded areas of this figure represent $I_1$ and $I_2$, respectively. The term $I_1$ signifies the probability that the maximum-delay paths that violate timing constraint and go through node $n_2$ do not go through edge $e_{1,2}$ and $e_{2,3}$. The term $I_2$ corresponds to probability that maximum-delay paths that violate timing constraint go through edge $e_{1,2}$ or $e_{2,3}$ but do not go through segment $s_{1,2,3}$. This can happen for paths, which go through segments $s_{1,2,j}$ and $s_{i,2,3}$.

Specifically, if node $n_2$ has single fanin or fanout, we can simplify Eq. (3.5) as follows:

**Lemma 3.1** For segment $s_{1,2,3}$ as shown in Fig. 3.2, we have:

1. If $n_2$ has single fanin and multiple fanouts, then $C_{1,2,3} = C_{2,3}$ holds.
2. If $n_2$ has single fanout and multiple fanins, then $C_{1,2,3} = C_{1,2}$ holds.
3. If $n_2$ has single fanin and fanout, then $C_{1,2,3} = C_2$ holds.

**Proof** In all these 3 cases above, we have $I_1 = I_2 = 0$. In addition,

- For Case 1, we have $C_{1,2} = C_2$. Eq. (3.5) then simplifies to $C_{1,2,3} = C_{2,3}$. 
For Case 2, we have $C_{2,3} = C_2$. Eq. (3.5) then simplifies to $C_{1,2,3} = C_{1,2}$.

For Case 3, we have $C_{1,2,3} = C_{1,2} = C_{2,3} = C_2$.

The proof then completes.

Lemma 3.1 illustrates that we can express the violation probability for two-connected edges directly using pre-computed node/edge violation probabilities, if the intermediate node has single fanin or fanout. For the single fanout case, we will show in Section 3.5 that we can generalize the above observation to multiple connected edges. For the single fanin case, we show in Section 3.6 that the results can be applied as a pre-processing step in our proposed algorithm for statistically-critical path extraction.

In the remainder of this section, we will resume to finding bounds for the violation probability of two connected edges when the intermediate node has multiple fanins and fanouts. Our approach is by finding bounds for $I_1$ and $I_2$ defined in Theorem 3.1. We will show that we can express the lower and upper bounds of $C_{1,2,3}$ in terms of $C_2$ (node violation probability), $C_{1,2}$ and $C_{1,3}$ (edge violation probabilities), as well as some new quantities that we will introduce along the way and can be efficiently pre-computed.

### 3.4.1 Lower & Upper Bound Computation

Lemma 3.1 gives the lower bound of $C_{1,2,3}$ when $n_2$ has single fanin or fanout. Here, Lemma 3.2 gives the lower bound of $C_{1,2,3}$ for the more general case when $n_2$ has multiple fanins and fanouts.
Lemma 3.2 The lower bound for $C_{1,2,3}$ is given as below:

$$C_{1,2,3} \geq C_{1,2} + C_{2,3} - C_2 - Pr \left( (D_{1,2} \geq T_0) \bigcap (D_{2,3} \geq T_0) \bigcap (D_{2,3} < \max_{j \neq 3} D_{2,j}) \right)$$

Proof Given $I_2$ defined in Theorem 3.1, we have

$$I_2 = Pr \left( (D_{1,2} \geq T_0) \bigcap (D_{2,3} \geq T_0) \bigcap (D_{2,3} < T_0) \right)$$

$$= Pr \left( (D_{1,2} \geq T_0) \bigcap (D_{2,3} \geq T_0) \bigcap (D_{2,3} < T_0) \bigcap (D_{2,3} < \max_{j \neq 3} D_{2,j}) \right)$$

$$\leq \min \left( Pr(D_{1,2,3} < T_0), Pr \left( (D_{1,2} \geq T_0) \bigcap (D_{2,3} \geq T_0) \bigcap (D_{2,3} < \max_{j \neq 3} D_{2,j}) \right) \right)$$

$$= Pr \left( (D_{1,2} \geq T_0) \bigcap (D_{2,3} \geq T_0) \bigcap (D_{2,3} < \max_{j \neq 3} D_{2,j}) \right) \quad (3.7)$$

In the above equation, the first equality is due to the fact that $D_{1,2} \geq D_{1,2,3}$ always holds. Since $D_{1,2} = \max_j (D_{1,2,j})$, where $j$ is the fanout of $n_2$, we can directly conclude that $D_{2,3} < \max_{j \neq 3} D_{2,j}$ and add it to the term for $I_2$. The second inequality is due to the fact that $Pr(A \bigcap B) \leq \min(Pr(A), Pr(B))$, where $A$ and $B$ represent the event $(D_{123} < T_0)$ and $((D_{1,2} \geq T_0) \bigcap (D_{2,3} \geq T_0) \bigcap (D_{2,3} < \max_{j \neq 3} D_{2,j}))$, respectively. The third equality is based on our assumptions: $Pr(D_{1,2,3} < T_0) > 0.5$ and $Pr((D_{1,2} \geq T_0) \bigcap (D_{2,3} \geq T_0)) \leq Pr(D_{2,3} \geq T_0) < 0.5$. These assumptions mean that the circuit should have a timing yield of at least 50%, which is valid for nowadays designs. Since probability $I_1 \geq 0$ and given Eq. (3.7), we further have

$$I_1 - I_2 \geq -Pr \left( (D_{1,2} \geq T_0) \bigcap (D_{2,3} \geq T_0) \bigcap (D_{2,3} < \max_{j \neq 3} D_{2,j}) \right). \quad (3.8)$$
Given the above bound for $I_1 - I_2$ and Eq. (3.5), the proof completes.

**Computational Complexity:** Lemma 3.2 illustrates that the lower bound of $C_{1,2,3}$ can be computed by using the node/edge violation probability and a new probabilistic term $\epsilon \triangleq Pr((D_{1,2} \geq T_0) \cap (D_{2,3} \geq T_0) \cap (D_{2,3} < \max_{j \neq 3} D_{2,j}))$. We can further express this probabilistic term as follows:

$$
\begin{align*}
\epsilon &= Pr\left((D_{1,2} \geq T_0) \cap (D_{2,3} \geq T_0) \cap (D_{2,3} < \max_{j \neq 3} D_{2,j})\right) \\
&= Pr\left((D_{1,2} \geq T_0) \cap (D_{2,3} \geq T_0) \cap (D_{2,3} < \max_{j \neq 3} D_{2,j})\right) \\
&= Pr\left((D_{1,2} \geq T_0) \cap (D_{2,3} \geq T_0) \cap (T_0 + D_{2,3} - \max_{j \neq 3} D_{2,j} < T_0)\right) \\
&= Pr\left(\min (D_{1,2}, D_{2,3}) \geq T_0\right) - Pr\left(\min \left(D_{1,2}, D_{2,3}, T_0 + D_{2,3} - \max_{j \neq 3} D_{2,j}\right) \geq T_0\right)
\end{align*}
$$

Due to the fact that $\min(\xi_1, \xi_2) = -\max(-\xi_1, -\xi_2)$, we can compute the probabilistic term $\epsilon$ using three MAX operations. In addition, since $\epsilon$ is only relevant with the maximum path delay going through each edge, we can pre-compute this probabilistic term before the path extraction. Therefore, we conclude that we can compute the lower bound for any two-connected edges in constant computational complexity.

We also compute the upper bound of the violation probability for two connected edges in Fig. 3.2 as follows.

**Lemma 3.3** The upper bound of $C_{1,2,3}$ is given by

$$
C_{1,2,3} \leq \min(C_{1,2}, C_{2,3}). \quad (3.9)
$$
Proof  From Fig. 3.3, we know that $C_{1,2,3} \leq C_{1,2}$ and $C_{1,2,3} \leq C_{2,3}$. Intuitively, any path that violates timing constraint and goes through $s_{1,2,3}$ must include edge $e_{1,2}$ and $e_{2,3}$ as well. Therefore, the proof completes.

**Computational Complexity:** Lemma 3.3 indicates that given the edge violation probabilities, we can compute the upper bound of $C_{1,2,3}$ with constant computation complexity.

### 3.4.2 Discussion on Tightness of the Bounds

In Lemma 3.2, two places can introduce the looseness of the lower bound of $C_{1,2,3}$.

The first place is in Eq. (3.8) and we bound $I_1$ using $I_1 \geq 0$. From the definition of $I_1$ given in Theorem 3.1, we can have

$$I_1 = Pr\left((D_2 \geq T_0) \cap (D_{1,2} < T_0) \cap (D_{2,3} < T_0)\right) \leq Pr\left((D_2 \geq T_0) \cap (D_{2,3} < T_0)\right) = 1 - Pr\left((D_2 < T_0) \cup (D_{2,3} \geq T_0)\right). \quad (3.10)$$

In Eq. (3.10), we have to evaluate whether $Pr((D_2 < T_0) \cup (D_{2,3} \geq T_0))$ is close to 1, which results in $I_1 \to 0$, indicating the lower bound of $I_1$ is tight. We consider both balanced and unbalanced circuits. The former one has many timing paths with very similar delays, while the latter one has few dominant ones.

For balanced circuits, it is very likely that the maximum path delay going through node $n_2$ is very close to that going through edge $e_{2,3}$, which indicates that $D_2 \approx D_{2,3}$. Therefore, we can conclude that $Pr((D_2 < T_0) \cup (D_{2,3} \geq T_0))$ is close to 1, and then
$I_1 \approx 0$. For unbalanced circuits, if node $n_3$ is selected to bind with edge $e_{1,2}$ to form the critical segment, then $e_{23}$ is dominant among all the other fanout edges of $n_2$, which also results in $I_1 \approx 0$.

The second place to result in the looseness in the lower bound of $C_{1,2,3}$ is the second inequality in Eq. (3.7):

$$\epsilon - I_2 = Pr\left( (D_{1,2} \geq T_0) \cap (D_{2,3} \geq T_0) \cap (D_{2,3} < \max_{j \neq 3} D_{2,j}) \cap (D_{1,2,3} \geq T_0) \right)$$

$$\leq Pr\left( (D_{2,3} < \max_{j \neq 3} D_{2,j}) \cap (D_{1,2,3} \geq T_0) \right)$$

$$\leq \min\left( Pr\left( D_{1,2,3} < \max_{j \neq 3} D_{1,2,j} \right), Pr(D_{1,2,3} \geq T_0) \right)$$

We expect $Pr(D_{1,2,3} < \max_{j \neq 3} D_{1,2,j})$ to be very small since during our path extraction process, we always intend to select the fanout node with high probability to violate the timing. Therefore, $D_{1,2,3}$ always turns out to be one for which the right-hand-side of the inequality is very small.

The above two discussions indicate that our derived lower bound for $C_{1,2,3}$ in Lemma 3.2 is tight, as we will also verify in our simulation results.

With respect to the upper bound of $C_{1,2,3}$, its expression is simple and only depends on the pre-computed edge violation probabilities. It is possible to derive a tighter upper bound, if we include more information and follow similar procedure to that of the lower bound computation. However, we will show in our simulation results that the current upper bound is already very tight.
3.4.3 Lower Bound Comparisons

In [41], we computed the lower bound of $C_{1,2,3}$ as

$$C_{1,2,3} \geq C_{1,2} + C_{2,3} - C_2 - Pr\left(\left(D_{2,3} \geq T_0\right) \cap \left(D_{2,3} < \max \left\{ D_{2,j} \right\}_{j \neq 3}\right)\right) \quad (3.11)$$

For the convenience of presentations, let us first denote the lower bound of $C_{1,2,3}$ in Eq. (3.11) and Eq. (3.7) as $LB_{old}$ and $LB_{new}$, respectively, and then define the difference between these two bounds as follows:

$$\Delta_{1,2,3} \triangleq LB_{new} - LB_{old}$$

$$= Pr\left((D_{1,2} < T_0) \cap (D_{2,3} \geq T_0) \cap \left(D_{2,3} < \max \left\{ D_{2,j} \right\}_{j \neq 3}\right)\right) \geq 0 \quad (3.12)$$

To elaborate the indications of $\Delta_{1,2,3}$ defined above, we consider the example of Fig. 3.1. Suppose that we are computing the lower bound of violation probability $C_{4,6,7}$, corresponding to segment $s_{4,6,7}$ ($n_4 \rightarrow n_6 \rightarrow n_7$), we can express the lower bound difference defined above as

$$\Delta_{4,6,7} = Pr\left((D_{4,6} < T_0) \cap (D_{6,7} \geq T_0) \cap (D_{6,7} < D_{6,8})\right).$$

This $\Delta_{4,6,7}$ term is equal to the probability of an event, which should satisfy: 1) the maximum-delay paths going through $n_4 \rightarrow n_6$ do not violate the timing constraint but the paths going through $n_6 \rightarrow n_7$ or $n_6 \rightarrow n_8$ can violate the timing constraint; 2) the path going through $n_4 \rightarrow n_6 \rightarrow n_7$ has smaller delay than that going through $n_4 \rightarrow n_6 \rightarrow n_8$. The above event may have non-negligible probability to occur, which indicates that $\Delta_{4,6,7}$ cannot be ignored. Therefore, we can conclude that the lower
bound derived in Lemma 3.2 is tighter than that in our previous work [41].

3.5 Extensions for \( K \) Connected Edges

In this section, we consider the more general case as shown in Fig. 3.4. In this case, the nodes \( n_1, n_2, \ldots, n_K, \) and \( n_{K+1} \) are connected using \( K \) edges \( (e_1, 2, \ldots, e_{K-1}, K, e_{K, K+1}) \) and form a long segment \( s_{1,2,\ldots,K+1} \). We aim to compute the lower and upper bounds for the violation probability of this segment, which is denoted by \( C_{1,2,\ldots,K+1} \).

3.5.1 Lower & Upper Bound Computation

Based on Lemma 3.2, we can express the lower bound of the violation probability \( C_{1,2,\ldots,K+1} \) as follows.

**Lemma 3.4** The lower bound of \( C_{1,2,\ldots,K+1} \) is given by:

\[
C_{1,2,\ldots,K+1} \geq \sum_{i=1}^{K} C_{i,i+1} - \sum_{i=2}^{K} C_i - \sum_{i=2}^{K} \Pr \left( (D_{i-1,i} \geq T_0) \bigcap (D_{i,i+1} \geq T_0) \bigcap \left( D_{i,i+1} < \max_{\forall j \neq i+1} D_{i,j} \right) \right)
\]
Proof For the convenience of notations, we first denote the right hand side of the inequality in Lemma 3.4 as $L_{K+1}$, and then use induction to prove this lemma:

- Base Case ($K = 2$): We have already proved this case in Lemma 3.2.
- Inductive Step: Given the induction hypothesis in which $C_{1,2,...,i} \geq L_i$ we need to prove $C_{1,2,...,i+1} \geq L_{i+1}$.

For the inductive step, we consider the segment $s_{1,2,...,i}$ as a long edge. By applying Lemma 3.2, we can derive

\[
C_{1,2,...,i+1} \\
\geq C_{1,2,...,i} + C_{i,i+1} - C_i - Pr \left( (D_{1,2,...,i} \geq T_0) \cap (D_{i,i+1} \geq T_0) \cap \left( D_{i,i+1} < \max_{\forall j \neq i+1} D_{i,j} \right) \right) \\
\geq C_{1,2,...,i} + C_{i,i+1} - C_i - Pr \left( (D_{i-1,i} \geq T_0) \cap (D_{i,i+1} \geq T_0) \cap \left( D_{i,i+1} < \max_{\forall j \neq i+1} D_{i,j} \right) \right) \\
\geq L_i + C_{i,i+1} - C_i - Pr \left( (D_{i-1,i} \geq T_0) \cap (D_{i,i+1} \geq T_0) \cap \left( D_{i,i+1} < \max_{\forall j \neq i+1} D_{i,j} \right) \right) \\
\geq L_i + C_{i,i+1} - C_i - Pr \left( (D_{i-1,i} \geq T_0) \cap (D_{i,i+1} \geq T_0) \cap \left( D_{i,i+1} < \max_{\forall j \neq i+1} D_{i,j} \right) \right) ,
\]

(3.13)

which gives our lower bound. The proof then completes. \qed

Similar to Lemma 3.3, we can obtain the upper bound of $C_{1,2,...,K+1}$ using the following lemma.

Lemma 3.5 We denote the upper bound of $C_{1,2,...,K+1}$ by $U_{K+1}$. Then, we have:

\[
C_{1,2,...,K+1} \leq \min (U_K, C_{K,K+1}) \leq \min_{\forall i = \{1,2,...,K\}} C_{i,i+1}
\]

(3.14)
The proof is straightforward and similar to Lemma 3.3. So we do not repeat here.

**Computational Complexity:** Lemma 3.5 shows that we can incrementally compute the upper bound of $C_{1,2,...,i+1}$ using $U_i$ which is the upper bound of $C_{1,2,...,i}$, and the available edge violation probability $e_{i,i+1}$. Meanwhile, to compute the lower bound of $C_{1,2,...,i+1}$ using Eq. (3.13) requires $L_i$ which is the lower bound of $C_{1,2,...,i}$, the available node/edge violation probabilities $C_i$ and $C_{i,i+1}$, as well as a probabilistic term which is similar to $\epsilon$ in Lemma 3.2. We have shown in Section 3.4 that this probabilistic term can be computed in constant time. Consequently, if $L_i$ is given for $C_{1,2,...,i}$, we can incrementally compute the lower bound of $C_{1,2,...,i+1}$ in constant time.

### 3.5.2 Discussion on Tightness of the Bounds

The last inequality in Eq. (3.13) shows that every time we add an edge $e_{i,i+1}$ to a segment $s_{1,2,...,i}$ to form a longer segment $s_{1,2,...,i+1}$, we replace the segment violation probability $C_{1,2,...,i}$ using its lower bound $L_i$. It indicates that every time the lower bound of the segment violation probability would become looser as the segment size grows. However, as illustrated in Section 3.4, we expect that the lower bound for two-connected edges is tight. In addition, as we will show in our proposed algorithm, if the difference between the lower and upper bounds of the segment violation probability grows larger than a threshold, we will update the lower/upper bounds using the actual violation probability. We will elaborate this in Section 3.6.
3.6 Our Proposed Algorithm

The outline of our algorithm to extract the top \( n \) statistically-critical paths is as follows:

\textbf{Algorithm 3:} Top \( n \) statistically-critical path extraction under process variations

\textbf{Input:} Target number of extracted paths \( n \), and circuit timing graph \( \mathcal{G} = (\mathcal{N}, \mathcal{E}) \).

\textbf{Output:} The top \( n \) statistically-critical paths \( \mathcal{P}_c \).

1. Apply pre-processing to combine some edges in \( \mathcal{E} \) into segments and derive new circuit timing graph \( \mathcal{G}' \).

2. Initialize the segment set stored at the super primary input in \( \mathcal{G}' \) as empty.

3. Visit nodes in \( \mathcal{G}' \) in topological order from primary inputs to primary outputs.
   
   3.1 At each node \( n_i \), add edge \( e_{j,i} \) to all segments in the set \( \mathcal{S}_j \) stored at each fanin node \( n_j \) of \( n_i \).
   
   3.2 Merge all segments in \( \mathcal{S}_j \) for each fanin \( n_j \) of \( n_i \) and remove the inferior segments using the bound-based pruning to form \( \mathcal{S}_i \).

4. At the super primary output in \( \mathcal{G}' \), select the top \( n \) paths from the pool of all non-inferior paths to form the target set of statistically-critical paths \( \mathcal{P}_c \).

More specifically, we first combine some edges into segments at Step 1 to derive a new circuit timing graph \( \mathcal{G}' \). At the super primary input, we initialize the stored segment set as empty (Step 2). Then, we visit all nodes in \( \mathcal{G}' \) in a topological ordering at Step 3. At each node, we add the edges to the segments stored at its fanin nodes (Step 3.1), and then perform bound-based pruning to remove the inferior paths, which cannot belong to the top \( n \) statistically-critical paths (Step 3.2). Particularly, we use the upper/lower bounds derived in Sections 3.4 and 3.5 to help the pruning. At the super primary output, we pick up the final \( n \) paths as our solution \( \mathcal{P}_c \) (Step 4).
the remaining of this section, we will discuss the details of Steps 1, 3.2, 4 and the runtime complexity of our proposed algorithm.

3.6.1 Edge Combinations

Lemma 3.1 shows that when the intermediate nodes, such as node $n_2$ in Fig. 3.2, have single fanin or fanout, we can directly express the segment violation probability using edge violation probability. Therefore, before visiting the nodes from primary inputs to primary outputs, we first combine some edges to segments and remove the intermediate nodes (Step 1). This helps to potentially reduce the accumulation of the looseness in the lower bound of the segment violation probability, which in turn can result in the reduction of the runtime complexity of our proposed algorithm.

Fig. 3.5 shows that we can incorporate three cases for edge combinations. However, in our proposed algorithm, we only consider Cases (a) and (c). The reasons that we do not consider Case (b) are as follows. First, we have shown in Section 3.4 that our computed lower bound can handle this case accurately even without combination. Second, even after combination of the edges, we still need to merge the paths stored at multiple fanins and select the superior ones.
3.6.2 Optimal Pruning of Inferior Segments

This pruning is performed at Step 3.2 and based on the lower/upper bounds discussed in Section 3.4 and 3.5. Given the set of segments $\mathcal{S}_i$ stored at node $n_i$, we first divide all the segments in $\mathcal{S}_i$ into the following categories:

1. The $B$ segments: For a given $n$ which is the target number of paths to be extracted, the $B$ segments refer to any segments $s_l \in \mathcal{S}_i$ such that its lower bound denoted as $LB_{s_l}$ is among the top $n$ largest lower bounds of all segments in $\mathcal{S}_i$ (See Fig. 3.6). We also denote the minimum of the lower bounds of the segments in $B$ by $LB_B = \min_{s_l \in B}(LB_{s_l})$.

2. The $R$ segments: They are the segments $s_l \in \mathcal{S}_i$ such that its upper bound denoted as $UB_{s_l}$ is smaller than $LB_B$. In other words, we have $R = \{s_l | UB_{s_l} \leq LB_B\}$. In addition, the $R$ segments never overlap with the $B$ segments.

3. The remaining segments in $\mathcal{S}_i$ are the $G$ segments, which might overlap with both $R$ and $B$; $G = \{s_l|s_l \notin R, s_l \notin B, s_l \in \mathcal{S}_i\}$. 

Figure 3.6: Classification of the segments for $n = 3$
Then, at each node $n_i$, we prune the $R$ segments and keep all the $G$ and $B$ segments, which can keep the optimality of our proposed algorithm. Notice that we need to memorize $|G| + |B|$ segments at each intermediate node. However, Eq. (3.13) has shown that the difference between the segment violation probability and our computed lower bound increases with the addition of the edges to form a longer segment. This indicates that the number of segments memorized at some intermediate nodes (i.e., $|G| + |B|$) may go very high, which may deteriorate the performance of our proposed approach. Therefore, we selectively update the lower/upper bounds for the segments stored at some intermediate nodes.

The selection criterion for the bound update is as follows. If the number of segments stored at node $n_i$ is larger than a pre-specified threshold $\alpha \cdot n$, we check the difference between the lower and upper bounds of all segments (i.e., $s_i$) stored at $n_i$. For segment $s_i$, if the lower bound of its violation probability becomes smaller than $\beta < 1$ times of its upper bound, we update both lower and upper bounds of this segment violation probability using the actual violation probability. Note that we update the bound for the segments only when both the above two conditions are satisfied. Using the parameterized expressions for the $RAT$ at $n_i$ and the delays of all gates and interconnects which form the segment, we can analytically compute the actual segment violation probability using Clark’s method [7] or Pearson Curve [33]. Note that as discussed in Section 3.3.2, we have already computed these parameterized expressions. Therefore, this update will not increase the computational overhead of our proposed algorithm.
3.6.3 Selecting top $n$ paths at Super Primary Output

Since we consecutively construct segments from the super primary input to super primary output, all segments stored at super primary output are in fact paths in the circuit. Here, we discuss two criteria to select the final top $n$ statistically-critical paths among all the candidates stored at the super primary output (Step 4). The first criterion is to compute the actual violation probability for all the candidate paths and then select the ones with the highest path violation probabilities. This criterion can guarantee that our selection is accurate without error, however, the total number of paths memorized at the super primary output may be very large, which results in low runtime. The second criterion is to simply select the top $n$ paths with the highest upper or lower bound values. This selection can eliminate the runtime of computing the actual violation probabilities for all candidate paths.

3.6.4 Runtime Complexity

In Algorithm 3, we visit each node exactly once and at each node, we merge the segments stored at all its fanins. We update the upper/lower bounds of each path. Computing upper and lower bounds for the violation probability of each segment can be incrementally done with constant computation complexity. In addition, we introduce the bound update criterion, so that we can bound the maximum number of stored segments over all the nodes in the circuit timing graph using $\gamma n$, where $\gamma$ is a finite constant and $n$ is the target number of paths to be extracted. Then, we conclude that the complexity of our algorithm is $O(n|\mathcal{N}|)$ where $|\mathcal{N}|$ denotes the number of nodes in the circuit timing graph $\mathcal{G}$. 
3.7 Extracting The Most Statistically-Critical Path

Here, we consider the case when our goal is to extract the most statistically-critical path ($n = 1$). We first loosen the lower bound derived in Lemma 3.4. Since we aim to extract only a single path, we can tolerate a looser lower bound, which is still tight enough as shown in our simulation results. We start deriving the new bound from considering the simple case: two-connected edges. From Lemma 3.2, we can derive

$$C_{1,2,3} \geq C_{1,2} + C_{2,3} - C_2 - Pr\left((D_{1,2} \geq T_0) \cap (D_{2,3} \geq T_0) \cap (D_{2,3} < \max_{j \neq 3} D_{2,j})\right)$$

$$\geq C_{1,2} + C_{2,3} - C_2 - Pr\left((D_{2,3} \geq T_0) \cap (D_{2,3} < \max_{j \neq 3} D_{2,j})\right)$$

$$= C_{1,2} + Pr\left((D_{2,3} \geq T_0) \cap (D_{2,3} \geq \max_{j \neq 3} D_{2,j})\right) - C_2$$

$$= C_{1,2} + Pr\left(D_{2,3} \geq \max_{i \neq \{3\}} (D_{2,j}, T_0)\right) - C_2,$$

where the computation of $Pr(D_{2,3} \geq \max_{i \neq \{3\}} (D_{2,j}, T_0))$ can be pre-processed very efficiently using several statistical MAX operations. Following the same procedures in Section 3.5, for any $K$ connected edges in the circuit, we can derive

$$C_{1,2,\ldots,K+1} \geq \sum_{i=1}^{K} C_{i,i+1} + \sum_{i=2}^{K} \left(Pr\left(D_{i-1,i} \geq \max_{j \neq \{i\}} (D_{i,j}, T_0)\right) - C_i\right),$$

which is a little looser than the lower bound in Lemma 3.4.

With the new lower bound given in Eq. (3.16), we introduce the following Lemmas:

**Lemma 3.6** Given a circuit timing graph $G = (N, E)$ as defined in Section 2.1.3, for
each edge $e_{i,j}$ in $\mathcal{E}$, we define its length using

$$l_{i,j} = \begin{cases} 
C_{i,j} + Pr \left( D_i \geq \max_{\forall k \neq j} (D_{i,k}, T_0) \right) - C_i & \forall i \notin PI \\
C_{i,j} & \forall i \in PI
\end{cases}.$$

Then, the maximum attainable lower bound of the path violation probability, denoted by $LB_{max}$, is equal to the length of the longest path in $\mathcal{G}$.

**Proof** Based on the definition of the edge length given above, we can observe that for any path in $\mathcal{G}$, its length is equal to the lower bound of its violation probability given in Eq. (3.16). Therefore, we conclude that the maximum attainable lower bound for the path violation probability $LB_{max}$ is equal to the length of the longest path in $\mathcal{G}$. The proof then completes.

**Lemma 3.7** To extract the most statistically-critical path, the edge $e_{i,j}$ can be removed from $\mathcal{G}(\mathcal{N}, \mathcal{E})$ if its violation probability $C_{i,j} \leq LB_{max}$, which is defined in Lemma 3.6.

The proof is simple and thus skipped.

Lemmas 3.6 and 3.7 illustrate that before the extraction of the most statistically-critical path, we can first perform graph pruning to remove all edge $e_{i,j}$ with their edge violation probability smaller than $LB_{max}$. The computation of $LB_{max}$ can be formulated into a longest path problem. Then, we can utilize Algorithm 3 to extract the most statistically-critical path. This pruning can greatly reduce the circuit size, which we will show in Section 3.8.
3.8 Experimental Results

We synthesize ISCAS’85 and ISCAS’89 benchmark suites using Synopsys Design Compiler with a 45nm technology library [1]. This synthesis is conducted for minimum area under a stringent timing constraint to ensure that the design is optimized and many paths are critical after optimization. We assume process variations in effective channel length and zero-body bias threshold voltage with standard deviations of 5% and 10% of their means, respectively. We use the hierarchical model of [3] illustrated in Section 2.2.1 to capture the spatial correlations. We further assume that these process variations follow Gaussian distributions, and use a linear model to express the variation-aware gate delays in terms of process variations as in Eq. (2.13). The above assumptions are consistent with most recent literatures on path extraction, such as [16, 66]. All simulations are ran on a 2.2GHz processor with 2GB memory running Linux.

For each benchmark, we first compute the node/edge violation probability defined in Eqs. (3.2) and (3.3), where we set nominal circuit delay (without variations) as the timing constraint $T_0$. As we have already illustrated in Section 3.3.1, we can accurately compute these terms using two rounds (one forward and one backward) of any accurate parameterized SSTA approaches. After deriving the canonical forms of $AT$ and $RAT$ at each node, we can analytically compute these probabilistic terms using Pearson Curves in Section 3.3.1 or other techniques such as [30], accurately and efficiently. However, in our simulation, we compute these terms using two rounds of MC based SSTA with 10,000 samples. This selection guarantees that these violation probabilities are pre-processed without any error and can facilitate the evaluations of the correctness of our bound in simulation. Note that the SSTA with linear ap-
Table 3.1: Accuracy and efficiency of our bound-based algorithm: $T_0$(PSEC), Runtime(SEC)

<p>| Benchmark Info | Preprocess | Case I ($n = 100$) | | |
|----------------|------------|---------------------|| | |</p>
<table>
<thead>
<tr>
<th>#Gate</th>
<th>#Edge</th>
<th>Runtime</th>
<th>$N_{rem}$</th>
<th>$N_{update}$</th>
<th>$N_{update}$ (%)</th>
<th>$\frac{AC_p - ALB_{p}}{n}$</th>
<th>$\frac{UB_{p} - AC_{p}}{n}$</th>
<th>$\frac{AC^{MC}<em>{p} - AC</em>{p}}{n}$</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>C432</td>
<td>338</td>
<td>551</td>
<td>2.18</td>
<td>249</td>
<td>20</td>
<td>8.03</td>
<td>0.0056</td>
<td>0.0158</td>
<td>0.0063</td>
</tr>
<tr>
<td>C499</td>
<td>691</td>
<td>1107</td>
<td>4.61</td>
<td>513</td>
<td>45</td>
<td>8.77</td>
<td>0.0859</td>
<td>0.0945</td>
<td>0.0564</td>
</tr>
<tr>
<td>C880</td>
<td>683</td>
<td>1162</td>
<td>4.97</td>
<td>520</td>
<td>16</td>
<td>3.08</td>
<td>0.0192</td>
<td>0.0866</td>
<td>0.0349</td>
</tr>
<tr>
<td>C1355</td>
<td>702</td>
<td>1121</td>
<td>4.78</td>
<td>515</td>
<td>44</td>
<td>8.54</td>
<td>0.0827</td>
<td>0.0771</td>
<td>0.0658</td>
</tr>
<tr>
<td>C1908</td>
<td>707</td>
<td>1149</td>
<td>5.11</td>
<td>501</td>
<td>59</td>
<td>11.78</td>
<td>0.0478</td>
<td>0.0857</td>
<td>0.0429</td>
</tr>
<tr>
<td>C2670</td>
<td>971</td>
<td>1424</td>
<td>7.46</td>
<td>684</td>
<td>7</td>
<td>10.23</td>
<td>0.1354</td>
<td>0.1114</td>
<td>0.0288</td>
</tr>
<tr>
<td>C3540</td>
<td>1273</td>
<td>2272</td>
<td>15.18</td>
<td>991</td>
<td>93</td>
<td>9.38</td>
<td>0.0634</td>
<td>0.0053</td>
<td>0.0013</td>
</tr>
<tr>
<td>C5315</td>
<td>2220</td>
<td>3613</td>
<td>32.08</td>
<td>1656</td>
<td>54</td>
<td>3.26</td>
<td>0.0227</td>
<td>0.0602</td>
<td>0.0399</td>
</tr>
<tr>
<td>C7552</td>
<td>2706</td>
<td>4383</td>
<td>43.52</td>
<td>1920</td>
<td>84</td>
<td>4.38</td>
<td>0.0356</td>
<td>0.0288</td>
<td>0.0168</td>
</tr>
<tr>
<td>S15850</td>
<td>1007</td>
<td>1572</td>
<td>12.24</td>
<td>768</td>
<td>0</td>
<td>0</td>
<td>0.0377</td>
<td>0.0734</td>
<td>0.0037</td>
</tr>
<tr>
<td>S35932</td>
<td>16202</td>
<td>25622</td>
<td>578.95</td>
<td>12213</td>
<td>0</td>
<td>0</td>
<td>0.0082</td>
<td>0.0463</td>
<td>0.0037</td>
</tr>
<tr>
<td>S38417</td>
<td>14772</td>
<td>24848</td>
<td>474.32</td>
<td>11789</td>
<td>136</td>
<td>11.54</td>
<td>0.0484</td>
<td>0.0232</td>
<td>0.0072</td>
</tr>
<tr>
<td>S38584</td>
<td>11944</td>
<td>20996</td>
<td>463.98</td>
<td>9858</td>
<td>0</td>
<td>0</td>
<td>0.0510</td>
<td>0.0581</td>
<td>0.0144</td>
</tr>
<tr>
<td>Ave</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
proximation of arrival times [7], although extremely fast, would result in inaccuracy of the evaluation of our derived lower/upper bound for segment/path violation probability. In Table 3.1, Columns 2-3 lists the total number of nodes and edges for each benchmark. Column 4 lists the runtime of this preprocessing step, which we observe to be proportional to the size of the circuit (Columns 2-3). Even though MC simulation is time-consuming, it has already proved to be able to get highly accelerated via parallelization using GPU and FPGA platforms [18, 28] and when combined with intelligent sampling schemes [6, 20, 68].

3.8.1 Verifying Accuracy and Efficiency of Path Extraction

In this subsection, we consider two cases: they both have nominal circuit delay as the timing constraint $T_0$ but different target numbers for the statistical critical paths to be extracted, which are $n = 100$ and $n = 200$, respectively.

In Case 1, we extract the top 100 statistically-critical paths using Algorithm 3, where the control parameters are set as $\alpha = 3$ and $\beta = 0.3$. It indicates that at each node in the circuit, when the total number of stored segments is larger than $\alpha n = 300$, we have to check the following condition. If the segment violation probability has its lower bound smaller than 0.3 times of its upper bound, we need to update it using its actual value as discussed in Section 3.6.3. At the super primary output, we can use two different criteria to extract the top 100 statistically-critical paths among all candidate paths. The first one is to extract the top 100 paths with the highest upper bound with respect to their violation probabilities. The second one is to first compute the actual violation probabilities for all the paths stored at the super primary output and then select the top ones based on their actual path violation probabilities. Since
Table 3.2: Accuracy and efficiency of our bound-based algorithm: Runtime(SEC)

<table>
<thead>
<tr>
<th>Case II</th>
<th>$N_{rem}$</th>
<th>$N_{update}$</th>
<th>$N_{update}/N_{rem}$ (%)</th>
<th>$AC_{p} - ALB_{p}/n$</th>
<th>$AU_{B_{p}} - AC_{p}/n$</th>
<th>$AC_{p}^{NSC} - AC_{p}/n$</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>C432</td>
<td>249</td>
<td>16</td>
<td>6.43</td>
<td>0.0049</td>
<td>0.0158</td>
<td>0.0042</td>
<td>2.54</td>
</tr>
<tr>
<td>C499</td>
<td>513</td>
<td>34</td>
<td>6.63</td>
<td>0.1198</td>
<td>0.0994</td>
<td>0.0479</td>
<td>10.12</td>
</tr>
<tr>
<td>C880</td>
<td>520</td>
<td>6</td>
<td>1.15</td>
<td>0.0276</td>
<td>0.0798</td>
<td>0.0153</td>
<td>0.74</td>
</tr>
<tr>
<td>C1355</td>
<td>515</td>
<td>36</td>
<td>6.99</td>
<td>0.0693</td>
<td>0.0644</td>
<td>0.0446</td>
<td>11.37</td>
</tr>
<tr>
<td>C1908</td>
<td>501</td>
<td>33</td>
<td>6.59</td>
<td>0.0502</td>
<td>0.0762</td>
<td>0.0396</td>
<td>5.11</td>
</tr>
<tr>
<td>C2670</td>
<td>684</td>
<td>4</td>
<td>0.58</td>
<td>0.1017</td>
<td>0.1151</td>
<td>0.0243</td>
<td>0.50</td>
</tr>
<tr>
<td>C3540</td>
<td>991</td>
<td>68</td>
<td>6.86</td>
<td>0.0560</td>
<td>0.0052</td>
<td>0.0018</td>
<td>7.94</td>
</tr>
<tr>
<td>C5315</td>
<td>1656</td>
<td>48</td>
<td>2.90</td>
<td>0.0186</td>
<td>0.0577</td>
<td>0.0319</td>
<td>5.77</td>
</tr>
<tr>
<td>C7552</td>
<td>1920</td>
<td>49</td>
<td>2.55</td>
<td>0.0298</td>
<td>0.0095</td>
<td>0.0024</td>
<td>8.64</td>
</tr>
<tr>
<td>S15850</td>
<td>768</td>
<td>0</td>
<td>0</td>
<td>0.0252</td>
<td>0.0493</td>
<td>0.0001</td>
<td>0.04</td>
</tr>
<tr>
<td>S35932</td>
<td>12213</td>
<td>0</td>
<td>0</td>
<td>0.0044</td>
<td>0.0302</td>
<td>0.0009</td>
<td>0.77</td>
</tr>
<tr>
<td>S38417</td>
<td>11789</td>
<td>125</td>
<td>1.06</td>
<td>0.0447</td>
<td>0.0447</td>
<td>0.0226</td>
<td>58.28</td>
</tr>
<tr>
<td>S38584</td>
<td>9858</td>
<td>0</td>
<td>0</td>
<td>0.0358</td>
<td>0.0441</td>
<td>0.0135</td>
<td>0.60</td>
</tr>
<tr>
<td>Ave</td>
<td></td>
<td></td>
<td>3.48</td>
<td>0.0455</td>
<td>0.0534</td>
<td>0.0192</td>
<td></td>
</tr>
</tbody>
</table>

our computed lower/upper bound are exact, the second criterion guarantees that the extracted paths are exactly the same as the theoretical top 100 statistically-critical paths. But in our simulation, we use the first one and will show that this criterion is still accurate while it is faster.

Simulation results for Case 1 are given in Table 3.1. Particularly, Column 5 shows the total number of remaining nodes (i.e., $N_{rem}$) after Step 1 of Algorithm 3. Column 6 shows the total number of nodes whose upper/lower bound of the violation probability should be updated at Step 3.2 of Algorithm 3 (i.e., $N_{update}$). We can see in Column 7, $N_{update}$ is about 4.95% of $N_{rem}$ on average, which indicates that our lower/upper bound computation can help to prune the inferior paths at the intermediate nodes of the circuit.
To show the correctness of our derived lower/upper bounds, we conduct MC simulation to compute the actual path violation probability $C_{p_i}$ for all the extracted $n = 100$ statistically-critical paths where $i = 1, 2, \ldots, n$. The number of MC samples is equal to 10,000. For each path $p_i$, we compare $C_{p_i}$ with its upper/lower bounds, which are denoted as $LB_{p_i}$ and $UB_{p_i}$, respectively. We observe that $LB_{p_i} \leq C_{p_i} \leq UB_{p_i}$ holds for all 100 extracted paths. To further show the tightness of the derived lower/upper bounds, we compute the following accumulated terms $AC_p = \sum_{i=1}^{n} C_{p_i}$, $ALB_p = \sum_{i=1}^{n} LB_{p_i}$, and $AUB_p = \sum_{i=1}^{n} UB_{p_i}$, where $n$ is equal to 100. Columns 8-9 list the average difference of $ALB_p$ and $AUB_p$ compared to $AC_p$ over all the extracted paths. Since the maximum possible average difference is 1, we can conclude that both upper and lower bounds in this case are close to the actual path violation probability.

To evaluate the quality of our extracted statistically-critical paths, we implement exhaustive MC simulation. Specifically, we enumerate all possible paths in the circuit. By running MC simulation over all paths, we select $n = 100$ paths with the highest violation probability, and sum these $n$ violation probabilities together to denote $AC_p^{MC}$. Columns 10 in Table 3.1 shows that the average difference between $AC_p^{MC}$ and $AC_p$ is 0.0265, which indicates that our extracted paths are very accurate. Runtime of our approach is reported in Column 11, which is less than 5 SEC for all the circuits except the very large one S38417.

In Case 2, we set $n = 200$, $\alpha = 3$ and $\beta = 0.3$. Table 3.2 gives the simulation results in this case. We can see that all the differences illustrated in Case 1 ($n = 100$) are still small and the runtime is less than 12 SEC except S38417. We further compare Table 3.2 with Table 3.1, and observe that the runtime in the $n = 200$ case is higher than that in the $n = 100$ case. It is because at each intermediate node in the circuit,
Table 3.3: Comparison in the number of matched paths in Case 2 ($n = 200$)

<table>
<thead>
<tr>
<th>BENCH</th>
<th>$T_{0_{\text{min}}}$ (PSEC)</th>
<th>$n$</th>
<th>#Matched paths</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ours</td>
</tr>
<tr>
<td>C432</td>
<td>975.53</td>
<td>200</td>
<td>166</td>
</tr>
<tr>
<td>C499</td>
<td>1037.10</td>
<td>200</td>
<td>89</td>
</tr>
<tr>
<td>C880</td>
<td>728.38</td>
<td>200</td>
<td>146</td>
</tr>
<tr>
<td>C1355</td>
<td>1036.22</td>
<td>200</td>
<td>117</td>
</tr>
<tr>
<td>C1908</td>
<td>1203.42</td>
<td>200</td>
<td>109</td>
</tr>
<tr>
<td>C2670</td>
<td>888.36</td>
<td>200</td>
<td>132</td>
</tr>
<tr>
<td>C3540</td>
<td>1521.36</td>
<td>200</td>
<td>183</td>
</tr>
<tr>
<td>C5315</td>
<td>1115.55</td>
<td>200</td>
<td>116</td>
</tr>
<tr>
<td>C7552</td>
<td>1004.01</td>
<td>200</td>
<td>184</td>
</tr>
<tr>
<td>S15850</td>
<td>649.39</td>
<td>200</td>
<td>184</td>
</tr>
<tr>
<td>S35932</td>
<td>562.22</td>
<td>200</td>
<td>162</td>
</tr>
<tr>
<td>S38417</td>
<td>1086.40</td>
<td>200</td>
<td>158</td>
</tr>
<tr>
<td>S38584</td>
<td>773.87</td>
<td>200</td>
<td>151</td>
</tr>
</tbody>
</table>

we need to rank the paths based on the lower bound of their violation probabilities and the number of stored paths at each node is larger in Case 2 than that in Case 1.

For Case 2, we also report the number of matched statistically-critical paths which are identical to those generated using exhaustive MC simulation. The results are given in Table 3.3. Column 3 lists the target number of extracted statistically-critical paths ($n = 200$). Column 4 reports the number of common extracted paths by using our proposed approach and exhaustive MC simulation. We can see that in many cases, a large portion of the extracted paths are identical. We further make comparisons with an alternative approach. In this alternative approach, we first extract all the paths with their nominal delay larger than $T_{0_{\text{min}}}$ given in Column 2, which corresponds to $\alpha = 93\%$ of the timing constraint $T_0$. We then compare these paths with the 200 statistically-critical paths derived from exhaustive MC simulation, and record
the number of matched paths in Column 6. We can see that in some cases such as C432, C880, C2670 and S35932, our method significant outperforms the alternative approach in the number of matched paths.

### 3.8.2 Reduction for Most Statistically-Critical Path

Here, we aim to extract most statistically-critical path where $n$ is equal to 1. We still set $T_0$ to be the nominal circuit delay. We first apply the graph pruning proposed in Lemma 3.7, and then select the most statistically-critical path with the highest $C_p$ using MC. Column 2 in Table 3.4 shows the ratios of the number of pruned edges to the total number of edges (i.e., $|\mathcal{E}|$) in the timing graph $\mathcal{G}$ for each circuit. On average, we can prune 92.76% of the edges in $\mathcal{E}$. In Column 3, we report $LB_{max}$, which is the maximum attainable lower bound of the path violation probability de-
Table 3.5: Accuracy and efficiency of our bound-based algorithm with non-Gaussian process variations

<table>
<thead>
<tr>
<th>BENCH</th>
<th>n</th>
<th>(N_{rem})</th>
<th>(N_{update})</th>
<th>(\frac{\text{N}<em>{\text{update}}}{\text{N}</em>{\text{rem}}} \text{(%)})</th>
<th>(\frac{\Delta C_p - \Delta LB_p}{n})</th>
<th>(\frac{\Delta UB_p - \Delta C_p}{n})</th>
<th>(\frac{\Delta C_{MC} - \Delta C_p}{n})</th>
</tr>
</thead>
<tbody>
<tr>
<td>C432</td>
<td>100</td>
<td>249</td>
<td>20</td>
<td>8.03</td>
<td>0.0058</td>
<td>0.0156</td>
<td>0.0064</td>
</tr>
<tr>
<td>C432</td>
<td>200</td>
<td>249</td>
<td>16</td>
<td>6.43</td>
<td>0.0047</td>
<td>0.0158</td>
<td>0.0042</td>
</tr>
<tr>
<td>C499</td>
<td>100</td>
<td>513</td>
<td>44</td>
<td>8.58</td>
<td>0.0875</td>
<td>0.0907</td>
<td>0.0526</td>
</tr>
<tr>
<td>C499</td>
<td>200</td>
<td>513</td>
<td>34</td>
<td>6.63</td>
<td>0.1200</td>
<td>0.0943</td>
<td>0.0433</td>
</tr>
</tbody>
</table>

fined in Lemma 3.6. Column 4 gives the actual violation probability of the most statistically-critical path computed using MC. Compare Column 3 with Column 4, and we can observe that our computed maximum lower bound is very close to the violation probability of the most statistically-critical path.

We also perform another experiment. We first identify the edge \(e_{i,j}\) with the maximum edge violation probability (i.e., \(C_{max}^{i,j}\)), and then select the path \(p\) among all the paths going through this edge \(e_{i,j}\) using MC, which has the highest path violation probability \(C_p\). The results are given in Column 5, which are identical to Column 4. This confirms a similar assumption made in [66] that the most critical path goes through the edge with the highest criticality.

3.8.3 Experiment with Non-Gaussian Process Variations

To evaluate our bound-based algorithm in the presence of non-Gaussian process variations, we conduct the following experiment. We assume that the process variations in zero-bias threshold voltage follow Log-Normal distribution and that the process variations in effective channel length still follow Gaussian distribution. Particularly, for zero-bias threshold voltage, we first generate an auxiliary random variable
\(\xi_2 \triangleq \exp(\xi_1),\) where \(\xi_1\) follows Gaussian distribution with mean 0 and variance 0.2. Then, we derive another random variable \(\xi_3 \triangleq \frac{\xi_2 - \mu_{\xi_2}}{\sigma_{\xi_2}}\) where \(\mu_{\xi_2}\) and \(\sigma_{\xi_2}\) denote the mean and standard deviation of the random variable \(\xi_2\), respectively. The above definitions enable random variable \(\xi_3\) to follow Log-Normal distribution with mean 0, variance 1, and skewness 1.59. Consequently, we use this random variable \(\xi_3\) for the normalized variation in zero-bias threshold voltage. Note that the actual variances of the process variations in both effective channel length and zero-bias threshold voltage in this case are the same as in the previous experiments. All other assumptions and correlation modeling remain the same as in the previous experiments.

In this experiment, we select C432 and C499 for validation. We still use nominal circuit delay as the timing constraint \(T_0\) and extract the top \(n\) statistically-critical paths. Table 3.5 gives the simulation results for \(n = 100\) and \(n = 200\). We can observe that our bound-based approach can still get very high accuracy. Furthermore, let us compare Column 4 in Table 3.5 with Column 6 in Table 3.1 and Column 3 in Table 3.2. We can see that the number of updates remain similar in these cases. This can be an indication on the tightness of our bounds and its independence on the distribution of process variations.

3.9 Summary

We propose a bound-based approach to extract the top \(n\) statistically-critical paths under process variations, which are the paths with the top \(n\) highest timing violation probabilities. Using node and edge violation probabilities, we can compute tight lower and upper bounds for the violation probability of any arbitrary segment in the
circuit. We show that the lower/upper bounds can be updated in an incremental manner with constant computational complexity. By incorporating these bounds, we can gradually construct the top $n$ statistically-critical paths with very high accuracy and efficiency. In our simulations, we consider the case when $n$ is not too large, which is practical for incremental optimization at the pre-silicon stage. We show that our proposed algorithm can extract the top 200 statistically-critical paths efficiently.
Chapter 4

Representative Critical Path

Selection for Failing Path Isolation

In this chapter, we introduce a novel approach to identify a small set of representative critical paths for failing path isolation, where we define failing paths as the paths violating timing constraint at the post-silicon stage. To achieve this, we first extract a large number of statistically-critical paths, as defined in Chapter 3, to form a set of candidate failing paths. We then select the representative critical paths from this candidate set. The selected paths should have their delays to be highly correlated with the remaining failing paths. By directly measuring the delays of these selected representative critical paths at the post-silicon stage, we can predict the post-silicon delay of the candidate failing paths accurately, and help isolate the failing paths. Simulation results show that we can predict up to a few thousand candidate failing paths accurately using the post-silicon delays of less than 150 representative critical paths in the presence of more than 1,000 process variations.
4.1 Introduction

Failing paths are the paths which violate the required timing constraint at the post-silicon stage and thus are identified for each chip individually. In Chapter 3, we propose an efficient method to extract the top $n$ statistically-critical paths under process variations with the highest timing violation probability, which is defined as the probability to violate the giving timing constraint. These paths can thereby become the candidates of the failing paths at the post-silicon stage. However, in reality, a large number of paths, especially for circuits with balanced topology, would have non-negligible violation probability, which indicates that $n$ can be very large.

To help isolate the failing paths at the post-silicon stage, among the related literature, [53] proposes a statistical learning approach to predict the post-silicon delay of a pre-specified set of candidate failing paths. The prediction is with the aid of measuring the delays of a small set of representative critical paths. However, [53] does not discuss how to select these representative critical paths. It is not trivial since these paths should be selected such that their delays highly correlate with the other failing paths in the circuit.

To help identify such representative critical paths, [51] proposes a technique which relies on defining a set of basic features (e.g., the number/types of gates) to rank and cluster the target failing paths. This helps to define a smaller subset of representative ones to be used for prediction of timing failures at the post-silicon stage. However, it is not clear to what extent these features can actually “bind” paths to their representative ones in the presence of process variations.

Another related work that can be considered relevant is [55], which proposes design and synthesization of a representative critical path so that its delay highly correlates
with the circuit delay. By directly measuring the delay of this representative critical path at post-silicon stage, the overall chip frequency can be predicted. However, this approach cannot isolate the failing paths.

In this chapter, we assume that the source of uncertainty in the post-silicon delays is process variations (and not environmental factors) and the actual values of these variations for each chip are unknown. We study the identification of the set of representative critical paths from the candidate failing paths. The delays of these paths will be measured at the post-silicon and be used to predict the actual delays of a large pool of target failing paths. The goal is to select a minimum number of representative critical paths such that their delays would highly correlate with the delays of the candidate failing paths. This helps to more quickly isolate those failing paths and facilitates the diagnosis process. Simulation results show that for over 1,000 random variations and up to several thousand candidate failing paths, at most 150 paths are required for accurate delay prediction. Although there is an error associated with the post-silicon target path delay prediction, we show that the guardband for post-silicon timing analysis is very small on average, and it can be used to accelerate post-silicon timing diagnosis.

The organization of this chapter is as follows. We discuss the motivation behind our framework and our problem formulation in Section 4.2 and Section 4.3, respectively. We present the representative critical path selection using effective rank in Section 4.4. Simulation results are presented in Section 4.5 followed by conclusions.
4.2 Motivation

Let us first assume process variations in device parameters, such as effective channel length and zero-bias threshold voltage. We use the hierarchical model given in Section 2.2.1 to de-correlate these spatially correlated process variations into $m$ random variables, which are denoted by $\mathbf{X} \triangleq [X_1, X_2, \ldots, X_m] \in \mathbb{R}^{m \times 1}$, similar to Eq. (2.12). Each entry $X_i$ falls into one of the variation categories: die-to-die, within-die and random variations. Each random variation is specific to one gate or interconnect. After scaling these random variable, we assume that all entries $X_i$s in $\mathbf{X}$ are independent from each other and follow Gaussian distribution with mean 0 and variance 1. In addition, we assume that the actual value of $\mathbf{X}$ for each fabricated chip is unknown at the post-silicon stage.

Given a set of candidate failing paths $\mathcal{P}_c$ of size $n$, we follow the procedures illustrated in Section 2.4, and model their variation-aware delays, denoted by $\mathbf{d}_{\mathcal{P}_c} = [d_{p_1}, \ldots, d_{p_n}] \in \mathbb{R}^{n \times 1}$, using a linear function with respect to $\mathbf{X}$:

$$\mathbf{d}_{\mathcal{P}_c} = \mathbf{\mu}_{\mathcal{P}_c} + \mathbf{A}_{\mathcal{P}_c} \mathbf{X},$$

(4.1)

where $\mathbf{\mu}_{\mathcal{P}_c} \in \mathbb{R}^{n \times 1}$ represents the nominal delays of all paths in the set $\mathcal{P}_c$ and
$A_{\mathcal{P}_c} \in \mathbb{R}^{n \times m}$ is a sensitivity matrix. For example, we express the delay of the $i$-th path, $p_i$, using $d_{p_i} = \mu_{p_i} + \sum_{j=1}^{n_i} a_{ij} X_j$, where $\mu_{p_i}$ is the nominal path delay, $a_{ij}$ is the element in the $(i, j)$-th entry of $A_{\mathcal{P}_c}$. If $X_j$ is not related to any gate or interconnect on $p_i$, then we set $a_{ij} = 0$. Otherwise, $a_{ij}$ is the sensitivity of $d_{p_i}$ with respect to $X_j$.

Let us consider the circuit shown in Fig. 4.1. We assume the candidate failing paths to be $\mathcal{P}_c = \{p_1, p_2, p_3, p_4\}$ where the paths are $p_1 : \text{PI2} \rightarrow G1 \rightarrow G3 \rightarrow G4 \rightarrow \text{PO1}$, $p_2 : \text{PI2} \rightarrow G1 \rightarrow G3 \rightarrow G5 \rightarrow \text{PO2}$, $p_3 : \text{PI3} \rightarrow G2 \rightarrow G3 \rightarrow G4 \rightarrow \text{PO1}$, and $p_4 : \text{PI3} \rightarrow G2 \rightarrow G3 \rightarrow G5 \rightarrow \text{PO2}$. Without considering process variations, we can express the path delays using

\[
\begin{align*}
    d_{p_1} &= w_{\text{PI2,G1}} + w_{G1,G3} + w_{G3,G4} \\
    d_{p_2} &= w_{\text{PI2,G1}} + w_{G1,G3} + w_{G3,G5} \\
    d_{p_3} &= w_{\text{PI3,G2}} + w_{G2,G3} + w_{G3,G4} \\
    d_{p_4} &= w_{\text{PI3,G2}} + w_{G2,G3} + w_{G3,G5}
\end{align*}
\]

where $w_{i,j}$ is the edge weight and defined as given in Fig. 2.3. In addition, the primary outs do not have delay, which indicates that $w_{G4,\text{PO1}} = w_{G5,\text{PO2}} = 0$. From the above equation, we can derive

\[
d_{p_1} = d_{p_2} + d_{p_3} - d_{p_4}.
\]

It indicates that we can form a set of representative critical paths $\mathcal{P}_r = \{p_2, p_3, p_4\}$ to model the delays of the remaining path $p_1$ with zero error. However, the dimension of $\mathcal{P}_r$ can still be very large (3 out of 4).
For the simplicity of explanation, we further assume process variations in transistor channel length $L$. There are no other process variations such as in interconnect parameters. We use the zero-level hierarchical model in Section 2.2.1 for simplification to capture the spatial correlation. Then, follow the notations in Eq. (2.12), we define $X \triangleq [\Delta L_{D2D}, R(G1), R(G2), \ldots, R(G5)]$. In this case, we rewrite the variation-aware delays of the derived representative critical paths (i.e., $P_r = \{p_2, p_3, p_4\}$) using

$$d_{P_r} \triangleq [d_{p_2}, d_{p_3}, d_{p_4}] = A_{P_r}X$$

where we define

$$A_{P_r} \triangleq \begin{bmatrix}
- a_{G1,G3}^{L_{D2D}} + a_{G1,G3}^{L_{D2D}} + a_{G1,G3}^{L_{D2D}} & a_{G1,G3}^{R(G1)} & 0 & a_{G1,G3}^{R(G3)} & 0 & a_{G1,G3}^{R(G5)} \\
- a_{G2,G3}^{L_{D2D}} + a_{G2,G3}^{L_{D2D}} + a_{G2,G3}^{L_{D2D}} & 0 & a_{G2,G3}^{R(G2)} & a_{G2,G3}^{R(G3)} & a_{G2,G3}^{R(G4)} & 0 \\
- a_{G3,G5}^{L_{D2D}} + a_{G3,G5}^{L_{D2D}} + a_{G3,G5}^{L_{D2D}} & 0 & a_{G3,G5}^{R(G2)} & a_{G3,G5}^{R(G3)} & 0 & a_{G3,G5}^{R(G5)}
\end{bmatrix}$$

Each entry in the matrix $A_{P_r}$ represents the sensitivity of the path delay with respect to the corresponding process variations. For example, $a_{G2,G3}^{L_{D2D}}$ and $a_{G2,G3}^{R(G3)}$ denotes the sensitivity of $w_{G2,G3}$ with respect to $\Delta L_{D2D}$ and $R(G3)$.

Compare the 2-nd and 3-rd rows of matrix $A_{P_r}$, and we can find that their differences lie in 1-st, 5-th and 6-th terms (i.e., $A_{P_r}(2, i)$ and $A_{P_r}(3, i)$ for $i = 1, 5, 6$). First, Fig. 4.1 shows that G4 and G5 are of the same type. If their load capacitances are similar, then we have $a_{G3,G4}^{L_{D2D}}$ to be close to $a_{G3,G5}^{L_{D2D}}$. In addition, if $p_3$ and $p_4$ share more common gates, the difference between $A_{P_r}(2, 1)$ and $A_{P_r}(3, 1)$ could be quite small. Similarly, the difference in $d_{p_3}$ and $d_{p_4}$, which results from the 5-th and 6-th terms of the 2-nd and 3-rd rows of $A_{P_r}$, can be ignored. Another reason is that
the random variation takes a very small proportion over the total variation in gate parameter $L$. Therefore, we may be able to use $d_{p_3}$ to approximate $d_{p_4}$. It further indicates that we may use the representative critical paths $\mathcal{P}_r = \{p_2, p_3\}$ to predict the post-silicon delays of $p_1$ and $p_4$ by allowing a small error tolerance. We would like to note that in the more general case, we can use $d_{p_2}$ and $d_{p_3}$ to build a model for $d_{p_4}$ instead of directly approximating it using $d_{p_3}$. In addition, the above illustrations are not restricted to the very simple process variation model, and will be also valid under more general process variation models.

The above example shows that by incorporating both the structural and process variation correlation in the circuit, it is possible to utilize a small number of representative critical paths to predict the post-silicon delays of a large pool of candidate failing paths with a tolerable prediction error.

### 4.3 Problem Definition

Motivated by Section 4.2, we define our problem as follows:

**Problem Formulation:** Given a large pool of candidate failing paths denoted by $\mathcal{P}_c$ with size $n$, we aim to select a set of representative critical paths $\mathcal{P}_r \subset \mathcal{P}_c$ with size $r$. Here, we denote the set of remaining $(n - r)$ paths in $\mathcal{P}_c$ as $\mathcal{P}_m$ and their delays as $d_{\mathcal{P}_m}$. Assuming that the actual delays of $\mathcal{P}_r$, denoted by $d_{\mathcal{P}_r}$, are available at the post-silicon stage, we aim to build a prediction model that maps $d_{\mathcal{P}_r}$ to $d_{\mathcal{P}_m}$, which is the actual delays of paths in $\mathcal{P}_m$ at the post-silicon stage. Our objective is to minimize $r$ and identify the representative critical paths $\mathcal{P}_r$, such that the error of the delay prediction model is upper bounded by a sufficiently small tolerance $\epsilon$. 

Assumptions on accurate path delay measurements: In our problem definition, we assume that accurate post-silicon delay information on a small set of representative critical paths can be available. To derive these accurate measurement, we propose to use the inexpensive special-type scan flipflops developed in [73], which should be inserted at design stage along the representative critical paths.

Fig. 4.2 gives the visualization of our representative critical path selection problem. We aim to predict the post-silicon timing of the candidate failing paths $\mathcal{P}_c$ within an error tolerance $\epsilon$ using delay measurements on $\mathcal{P}_r$.

4.4 Our Proposed Algorithm

Given $n$ candidate failing paths $\mathcal{P}_c$, we aim to select $r$ representative critical paths $\mathcal{P}_r$ from $\mathcal{P}_c$ to predict the post-silicon delays of the remaining $n - r$ paths $\mathcal{P}_m$ within an error tolerance $\epsilon$. To solve this problem, in this section, we show that a recently-introduced idea in [10] known as effective rank of the transformation matrix between process variations and candidate failing path delays ($\mathbf{A}_{\mathcal{P}_c}$ in Eq. (4.1)) can be applied. While the rank of $\mathbf{A}_{\mathcal{P}_c}$ identifies the number of representative critical paths
which can exactly predict the delays of the candidate paths, the effective rank of $A_{P_c}$ is determined for a given tolerance error $\epsilon$ and can help to reduce the number of representative critical paths. We start from discussing exact prediction with $\epsilon = 0$.

4.4.1 Exact Selection

Let $A_{P_r}$ be the rows of $A_{P_c}$ corresponding to the $r$ paths that are selected, and $A_{P_m}$ be the remaining rows. We rewrite Eq. (4.1) as

$$
\mathbf{d}_{P_c} = \left[ \begin{array}{c} d_{P_r} \\ d_{P_m} \end{array} \right] = \left[ \begin{array}{c} \mu_{P_r} \\ \mu_{P_m} \end{array} \right] + \left[ \begin{array}{c} A_{P_r} \\ A_{P_m} \end{array} \right] \mathbf{X}. \quad (4.2)
$$

From the above equation, we can derive the following Theorem:

**Theorem 4.1** The smallest $r$ for which we can exactly express $d_{P_m}$ as a linear combination of $d_{P_r}$ is $r = \text{rank}(A_{P_c})$.

**Proof** From Eq. (4.1), we have $\mathbf{d}_{P_c} - \mu_{P} = A_{P_c} \mathbf{X}$. Given the definition of matrix rank in [13], we can conclude that $\mathbf{d}_{P_m} - \mu_{P_m}$ can be exactly written as a linear combination of $\mathbf{d}_{P_r} - \mu_{P_r}$ with $r = \text{rank}(A_{P_c})$. Therefore, the smallest $r$ to express the $d_{P_m}$ in terms of $d_{P_r}$ is $r = \text{rank}(A_{P_c})$. This completes the proof. \[\square\]

The above theorem shows that for $r = \text{rank}(A_{P_c})$, the $r$ paths corresponding to any $r$ linearly independent rows of $A_{P_c}$ will suffice as the representative critical paths. This is from the definition of matrix rank since the $r$ rows span all the remaining rows of $A_{P_c}$, and consequently the entire vector of path delay $d_{P_c}$ can be exactly recovered. To select these $r = \text{rank}(A_{P_c})$ representative critical paths, we can utilize Algorithm 4.2, which we will illustrate in Section 4.4.3.
To give an idea about \( \text{rank}(A_{P_c}) \), we define \( S \) to be a set of non-overlapping segments, which can cover all edges in \( P_c \). Furthermore, each segment is made of consecutive edges and the intermediate nodes on the segment all have an out-degree of 1. In this case, we can rewrite Eq. (4.1) using

\[
d_{P_c} = Gd_S = G (\mu_S + A_S X),
\]

(4.3)

where \( d_S \) denotes the delays of all segments in \( S \) under process variations, \( \mu_S \) is the nominal segment delay (without variations), \( A_S \) is the sensitivity matrix of segment delay with respect to process variation \( X \), and \( G \) is a zero-one matrix. Specifically, if the \( j \)-th segment in \( S \) belongs to the \( i \)-th path in \( P_c \), then we set the element in the \((i, j)\)-th entry of \( G \) to be 1. Otherwise, we set it to be 0.

The above definition for segment set \( S \) indicates that \( n_S \), which is the total number of segments, is at most equal to the number of edges in the circuit timing graph as defined in Section 2.1.3, since the segments are lumped representation of the edges. The number of edges in turn can be much smaller than the total number of candidate failing paths. Then, we derive the following Lemma:

**Lemma 4.1** For the smallest number of representative critical paths, we have \( r = \text{rank}(A_{P_c}) \leq n_S \), where \( n_S = |S| \).

**Proof** Compare Eq. (4.1) and Eq. (4.3), we have

\[
\mu_{P_c} = G \mu_S, \quad A_{P_c} = GA_S,
\]

which indicates that \( \text{rank}(A_{P_c}) \leq \text{rank}(G) \leq n_S \). Then the proof completes. \( \square \)
The above lemma states that to derive an exact linear expression that maps \( d_{P_c} \) to \( d_{P_m} \), we need at most \( n_S \) representative critical paths.

To further illustrate Theorem 4.1 and Lemma 4.1, we consider circuit S1423. In this circuit, we extract 644 statistically-critical paths, which indicates \( n = |P_c| = 644 \). These paths cover 415 gates and 255 segments. Since \( \text{rank}(A_{P_c}) = 122 \) holds, we only need 122 paths to exactly recover the delays of all remaining paths.

### 4.4.2 Representative Critical Path Selection With Effective Rank

The previous section shows that for exact path delay prediction, the minimum number of required paths is \( \text{rank}(A_{P_c}) \), where \( A_{P_c} \) is the transformation matrix between \( d_{P_c} \) and \( X \). Here, we will show that by allowing a small prediction error tolerance \( \epsilon \), we can greatly reduce the number of required paths by using the novel idea of effective rank proposed in [10].

We first explain the idea intuitively. Consider the example of S1423 again. Since the extracted 644 paths can only cover 255 segments, many paths are forced to share the segments. It indicates that many of the rows in \( G \) are similar. That is, for many pairs of paths in \( P_c \) such as \( p_i \) and \( p_k \), few entries such as \( G(i, j) \) and \( G(k, j) \) in \( G \) might be different, indicating that the two paths only differ in a few segments (i.e., segment \( s_j \)). Correspondingly, this results in similarity between the \( i \)-th and \( k \)-th rows of matrix \( A_{P_c} \) since \( A_S \) is a constant sensitivity matrix (See Eq. (4.3)). Therefore, intuitively, it seems that we may need much less than \( \text{rank}(A_{P_c}) = 122 \) to predict the remaining paths with high prediction accuracy.

Formally, we can perform singular value decomposition (SVD) over \( A_{P_c} \in \mathbb{R}^{n \times m} \).
and obtain $A_{P_c} = U \Delta V^T$ where $U$ and $V$ are $n \times n$ and $m \times m$ orthogonal matrices, respectively, and $\Delta$ is a $n \times m$ diagonal matrix. The diagonal elements $\lambda_i$ in $\Delta$ are singular values and follow $\lambda_i \geq \lambda_{i+1}$. The rank of $A_{P_c}$ is equal to the largest $i$ such that $\lambda_i > 0$. These $\lambda_i$s can also provide other insights into the structure of $A_{P_c}$.

Let us denote the energy as $E = \sum_{i=1}^{n} \lambda_i$, we define the effective rank of $A_{P_c}$ to be $\left\lceil \arg \min_k \left( \sum_{i=1}^{k} \lambda_i \geq (1 - \eta)E \right) \right\rceil$, where $\eta$ is specified as a threshold, for example 5%. So it is the index of the smallest singular value which marks the points exceeding $(1 - \eta)E$. The effective rank is shown to be closely related to prediction error $\epsilon$ in [10].

For matrix $A_{P_c}$, its effective rank can be much smaller than its rank. If its singular values $\lambda_i$ drop with a very fast rate, only a few singular values are dominant and the effective rank of $A_{P_c}$ can be very small. It indicates that very few representative critical paths are required for prediction under a given error tolerance. Fig. 4.3 (a) plots the normalized singular values of $A_{P_c}$, which is equal to $\lambda_i / \sum \lambda_i$, on the $y$-axis using log-linear scale for S1423. Simulation configuration is given in Section 4.5. In this figure, we sort the singular values of $A_{P_c}$ in non-increasing order and only plot the first 30 singular values. This figure shows that the sorted singular values with index larger than 20 are less $10^{-3}$. In addition, from the large gap that separates the singular values into subsets of large and small singular values, we can conclude that we may need less than 20 paths to predict remaining paths with very high accuracy.

However, with the further scaling in deep submicron technology, both the dimension and the extent of random variations with respect to the total variation (including die-to-die, within-die, and random variations) would greatly increase. In this case, the number of representative critical paths would dramatically grow. As an example, we only increase the sensitivity of the independent random variations in $A_{P_c}$ by 3X.
and plot its normalized singular values in Fig. 4.3 (b). We can see that the drop rate of the singular values of matrix $A P_c$ decreases quite a lot compared to Fig. 4.3 (a), which indicates that more representative critical paths are required for prediction of $d P_c$. We can show similar plots if we increase the number of random variations.

In the next subsection, we discuss the representative critical path selection assuming $\epsilon$ is provided. Specifically, we use effective rank to select representative critical paths so that the delay prediction error for remaining paths is bounded by $\epsilon$.

### 4.4.3 Representative Critical Path Selection Procedure with $\epsilon$

We give the high-level algorithm for representative critical path selection with error tolerance $\epsilon$ as below. We start Algorithm 4.1 by selecting $r = \text{rank}(A P_c)$ paths as explained in Section 4.4.1. The initial error in this case is $\epsilon_r = 0$. We decrement the number of target paths by one and select $r - 1$ representative critical paths in Step 2.2. This introduces a new error which we compute in Step 2.3 and update $\epsilon_r$. If the new error is still smaller than given $\epsilon$, we repeat another iteration and further decrement $r$ until the error tolerance is reached.
Algorithm 4.1: Representative Critical Path Selection

**Input:** error tolerance $\epsilon$, $d_{P_c} - \mu_{P_c} = A_{P_c}X$

**Output:** The set of representative critical paths $P_r$ and its dimension $r$.

1. Select $r = \text{rank } (A_{P_c})$ representative critical paths exactly and set error $\epsilon_r = 0$
2. While ($\epsilon_r \leq \epsilon$)
   2.1 $r \leftarrow r - 1$
   2.2 Select $r$ representative critical paths from $P_c$ to form $P_r$
   2.3 Update error $\epsilon_r$ for the newly selected paths.

In the remaining of this section, we will explain Step 2.2 of Algorithm 4.1 to select $r$ representative critical paths. We will also discuss building a model between $d_{P_r}$ and $d_{P_m}$ and the computation of error $\epsilon_r$, which are included in Step 2.3 of our algorithm.

**Step 2.2: Selection of $r$ Representative Critical Paths**

The selection of representative critical paths is a combinatorial optimization problem which is NP-complete [13]. From algorithmic perspective, it is equivalent to the subset selection problem in computational linear algebra. One procedure to solve this problem approximately is “QR decomposition using column pivoting” which we discuss below from [13].

Algorithm 4.2: Selection of $r$ Representative Critical Paths

**Input:** Matrix $A_{P_c}$ and $r \leq \text{rank}(A_{P_c})$.

**Output:** Matrix $A_{P_r}$ corresponding to the representative critical paths $P_r$.

1. Perform SVD on $A_{P_c} = U\Delta V^T$.
2. Perform “QR with column pivoting” on matrix $U_r$ composed by the first $r$ columns of $U$ and get: $U_r^T P_r = QR$, where $P_r$ is a $n \times n$ permutation matrix.
3. Take $A_{P_r}$ to be the sub-matrix formed by the first $r$ rows of $P_r^T A_{P_c}$

We first perform SVD on $A_{P_c}$ to obtain matrix $U$. Then we apply “QR decomposition with column pivoting” on $U$ [13]. The input of this procedure is $U_r$, a
sub-matrix formed by the first \( r \) columns of \( \mathbf{U} \). The matrices \( \mathbf{Q} \) and \( \mathbf{R} \) are found during the procedure and help identify the output permutation matrix \( \mathbf{P}_r \). After obtaining \( \mathbf{P}_r \), to identify the \( r \) representative critical paths, we compute \( \mathbf{P}_r^T \mathbf{A}_{\mathbf{P}_c} \) and take the sub-matrix formed by the first \( r \) rows which in turn relates to \( r \) path delays from vector \( \mathbf{d}_{\mathbf{P}_c} \).

Note that in Algorithm 4.1, as we decrement \( r \) at each iteration of the while loop, we apply the QR decomposition using column pivoting. This procedure can also be implemented incrementally based on the result of the previous iteration. For more details, we refer the reader to [10].

**Step 2.3: \( \mathbf{d}_{\mathbf{P}_r} \rightarrow \mathbf{d}_{\mathbf{P}_m} \) Model and Error Computation**

After selecting the \( r \) representative critical paths, we use the following Theorem to build a model from the delays of representative critical paths \( \mathbf{d}_{\mathbf{P}_r} \) to the delays of remaining paths \( \mathbf{d}_{\mathbf{P}_m} \). We assume all entries in \( \mathbf{X} \) are independent and follow Gaussian distribution with mean 0 and variance 1 as in [55].

**Theorem 4.2** The linear estimate of \( \mathbf{d}_{\mathbf{P}_m} \) is given as follows:

\[
\mathbf{d}'_{\mathbf{P}_m} = \mu_{\mathbf{P}_m} + \mathbf{A}_{\mathbf{P}_m} \mathbf{A}^T_{\mathbf{P}_r} (\mathbf{A}_{\mathbf{P}_r} \mathbf{A}^T_{\mathbf{P}_r})^{-1} (\mathbf{d}_{\mathbf{P}_r} - \mu_{\mathbf{P}_r}),
\](4.4)

where \((-1)^{-1}\) denotes the pseudo-inverse operator, and \( \mathbf{A}_{\mathbf{P}_r}, \mathbf{A}_{\mathbf{P}_m}, \mu_{\mathbf{P}_r}, \) and \( \mu_{\mathbf{P}_m} \) are defined in Eq. (4.2). The prediction error for this estimate can be computed using

\[
\Delta_r \triangleq \mathbf{d}'_{\mathbf{P}_m} - \mathbf{d}_{\mathbf{P}_m} = \mathbf{A}_{\mathbf{P}_m} \mathbf{A}^T_{\mathbf{P}_r} (\mathbf{A}_{\mathbf{P}_r} \mathbf{A}^T_{\mathbf{P}_r})^{-1} \mathbf{A}_{\mathbf{P}_r} \mathbf{X} - \mathbf{A}_{\mathbf{P}_m} \mathbf{X} = \Omega_r \mathbf{X},
\](4.5)

where \( \Omega_r \triangleq \mathbf{A}_{\mathbf{P}_m} \mathbf{A}^T_{\mathbf{P}_r} (\mathbf{A}_{\mathbf{P}_r} \mathbf{A}^T_{\mathbf{P}_r})^{-1} \mathbf{A}_{\mathbf{P}_r} - \mathbf{A}_{\mathbf{P}_m} \) is constant after selection is performed.
It indicates that $\Delta$, also follows multivariate Gaussian distribution.

**Proof**  Let us first consider the multi-dimensional random variable $d_{Pr}$, its covariance matrix can be written as

\[
\text{Cov} \left( \begin{bmatrix} d_{Pr} \\ d_{Pm} \end{bmatrix} \right) = \begin{bmatrix} \Sigma_{r,r} & \Sigma_{r,m}^T \\ \Sigma_{m,r} & \Sigma_{m,m} \end{bmatrix} = \begin{bmatrix} A_{Pr}A_{Pr}^T & A_{Pr}A_{Pm}^T \\ A_{Pm}A_{Pr}^T & A_{Pm}A_{Pm}^T \end{bmatrix}.
\]

Then, for the multi-dimensional random variable $d_{Pm}$, its distribution conditioned on $d_{Pr}$ is a multivariate normal distribution with mean $\bar{\mu}_{Pm}$ and covariance matrix $\bar{\sigma}_{Pm}^2$ as follows [50]:

\[
\bar{\mu}_{Pm} = \mu_{Pm} + \Sigma_{r,m}\Sigma_{r,r}^{-1}(d_{Pr} - \mu_{Pr})
\]

\[
\bar{\sigma}_{Pm}^2 = \Sigma_{m,m} - \Sigma_{r,m}\Sigma_{r,r}^{-1}\Sigma_{r,m}^T
\]

and we can further simplify the conditional mean $\bar{\mu}_{Pm}$ to

\[
\bar{\mu}_{Pm} = \mu_{Pm} + A_{Pm}A_{Pr}^T(A_{Pr}A_{Pr}^T)^{-1}(d_{Pr} - \mu_{Pr}). \quad (4.6)
\]

From the above equation, we can observe that this linear predictor is in fact equal to the conditional mean of $d_{Pm}$, which proves to be an optimal linear predictor in [50]. The remaining of the above theorem is straightforward, and thus the proof completes.

\[\blacksquare\]

**Error Definition:** Eq. (4.5) shows that we can compute the deviation of predicted path delays compared to exact path delays once we determine the representative
critical paths $\mathcal{P}_r$. In this chapter, we define the error $\epsilon_r$ used in Algorithm 4.1 as

$$
\epsilon_r = \max_{i=1,2,...,n-r} \frac{WC(|\Delta_r^{(i)}|)}{T_0},
$$

(4.7)

where $T_0$ denotes the circuit timing constraint, and $\Delta_r^{(i)}$ denotes the $i$-th entry of $\Delta_r$ given in Eq. (4.5). The function $WC(y)$ denotes the “worst-case” value of random variable $y$. For example, it can be considered as the 99-th percentile of $y$. Then, Eq. (4.7) indicates that the “worst-case” prediction error (deviation from the actual path delay) cannot be larger than $\epsilon_r T_0$ for all paths in $\mathcal{P}_m$. Since $\Delta_r$ follows multivariate-Gaussian distribution with known mean and covariance matrix, we can compute $WC(|\Delta_r^{(i)}|)$ analytically.

**Failing Path Isolation:** Here, we discuss how to use the prediction model derived in Theorem 4.2 for failing path isolation. Let us first define guardband for each remaining statistically-critical path $p_i \in \mathcal{P}_m$ as follows:

$$
\phi_i = WC(|\Delta_r^{(i)}|) \leq \epsilon_r T_0,
$$

(4.8)

where the inequality is from Eq. (4.7).

Eq. (4.8) indicates that for each remaining path in $\mathcal{P}_m$, we have an individual guardband. This guardband can help isolate failing paths at the post-silicon stage. More specifically, for any given chip $k$, we can incorporate the guardband defined in Eq. (4.8) to derive the worst-case and best-case delays of each remaining paths. Let us denote the post-silicon delay measurement on the representative critical paths $\mathcal{P}_r$ for chip $k$ as $d_{\mathcal{P}_r}(k)$. From the model in Eq. (4.5), we can compute the estimate of
actual delays of the remaining paths $P_m$ as

$$d'_{p_m}(k) = \mu_{p_m} + A_{p_m} A_{P_r}^T (A_{P_r} A_{P_r}^T)^{-1}(d_{P_r}(k) - \mu_{P_r}). \quad (4.9)$$

For instance, we consider the $i$-th remaining path in $P_m$. We denote its delay estimate as $d'_p(k)$ which corresponds to the $i$-th element in $d'_{p_m}(k)$ as given in Eq. (4.9). Combined with the guardband defined in Eq. (4.8), we can compute its worst-case and best-case post-silicon delays as

$$d_{p_i}^{(WC)}(k) \triangleq d'_p(k) + \phi_i, \quad d_{p_i}^{(BC)}(k) \triangleq d'_p(k) - \phi_i.$$

Consequently, for this given chip $k$ and each remaining path $p_i \in P_m$, we define the following three cases:

- Case 1: If $d_{p_i}^{(WC)}(k) \leq T_0$, then path $p_i$ on this chip $k$ cannot fail the timing.
- Case 2: If $d_{p_i}^{(WC)}(k) > T_0$, then path $p_i$ on this chip $k$ may fail the timing.
- Case 3: If $d_{p_i}^{(BC)}(k) > T_0$, then path $p_i$ on this chip $k$ would fail the timing.

The above categorization can help isolate failing paths. We will elaborate in our simulation results (See Section 4.5.2) that our isolated paths which may fail the timing (Case 2) would be very likely to fail the timing.

### 4.4.4 Complexity Analysis

Within Algorithm 4.1, we evoke Algorithm 4.2 for at most $r = \text{rank}(A_{P_c})$ times. Each call requires one SVD and one QR decomposition as its dominant computing
requirements. Sophisticated algorithms exist to solve these procedures. We use svd() and qr() functions from Matlab in our simulations. Generally, if the number of candidate failing paths is very large, we can apply a clustering procedure to form clusters of paths of smaller size for speedup. Furthermore, building the model between \(d_{\mathcal{P}_r}\) and \(d_{\mathcal{P}_m}\) and evaluating the error are all done analytically and very efficiently by using Theorem 4.2.

### 4.5 Experimental Results

We synthesize ISCAS’89 benchmarks using 90nm TSMC library and Synopsys Design Compiler for minimum area under a stringent timing constraint to ensure that the circuits are optimized. We assume process variations in effective channel length \(L_{\text{eff}}\) and zero-bias threshold voltage \(V_{th0}\), which follows Gaussian distribution with standard deviation equal to 10% of their mean. To capture the spatial correlation among these process variations, we use the hierarchical model illustrated in Section 2.2.1 and [3], which defines rectangular regions on the chip. Column 3 in Table 4.1 gives the total number of regions (\(|R|\)) of each benchmark. As shown in this table, for smaller benchmarks, we use a 3-level model resulting in \(\sum_{i=0}^{2} 4^i = 21\) regions. For larger ones, we use a 5-level model resulting in \(\sum_{i=1}^{4} 4^i = 341\) regions. This assumption is consistent with [55]. In addition, each gate has a random variation term, which is 6% of the total variations. Note that this cannot be captured by [55].

To evaluate the performance of our proposed approaches, we first adopt the variation of the algorithm given in Chapter 3 also known as [41] to extract the candidate failing paths \(\mathcal{P}_c\). Specifically, we extract all paths with path yield smaller than a
given threshold since they are more likely to fail the timing at the post-silicon stage. Note that our approaches can incorporate with any other path extraction algorithms.

Then, we generate \( N_S = 10,000 \) samples for process variations. Under each sample, we first compute the delays of the representative critical paths, and use them to predict the delays of the remaining paths. We compare the predicted delays with their actual delay values provided by the samples, and evaluate the following metrics:

**Metric:** We first define errors \( \varepsilon_i \) and \( \hat{\varepsilon}_i \), which indicate the maximum and average relative prediction errors for \( i \)-th remaining path, respectively. The \( e_1 \) and \( e_2 \) indicate the average of \( \varepsilon_i \) and \( \hat{\varepsilon}_i \) over all remaining paths \( P_m \), respectively:

\[
\varepsilon_i = \max_{k=1}^{N_S} \frac{|d'_{pi}(k) - d_{pi}(k)|}{d_{pi}(k)}, \quad e_1 = \frac{\sum_{i=1}^{n-r} \varepsilon_i}{n-r},
\]
\[
\hat{\varepsilon}_i = \frac{1}{N_S} \sum_{k=1}^{N_S} \frac{|d'_{pi}(k) - d_{pi}(k)|}{d_{pi}(k)}, \quad e_2 = \frac{\sum_{i=1}^{n-r} \hat{\varepsilon}_i}{n-r},
\]

where \( d'_{pi}(k) \) and \( d_{pi}(k) \) are the predicted and actual delays of the \( i \)-th remaining path in \( P_m \) for sample \( k \). They are also corresponding to the \( i \)-th element in \( d'_{P_m}(k) \) and \( d_{P_m}(k) \), respectively.

We also define three match rations as follows:

\[
m_{rf} = \frac{\sum_{k=1}^{N_S} \sum_{i=1}^{n-r} I_{d_{pi}(k) \leq T_0}}{\sum_{k=1}^{N_S} \sum_{i=1}^{n-r} I_{d_{pi}(k) \leq T_0}}, \quad m_{pf} = \frac{\sum_{k=1}^{N_S} \sum_{i=1}^{n-r} I_{d_{pi}(k) > T_0}}{\sum_{k=1}^{N_S} \sum_{i=1}^{n-r} I_{d_{pi}(k) > T_0}}, \quad m_f = \frac{\sum_{k=1}^{N_S} \sum_{i=1}^{n-r} I_{d_{pi}(BC)(k) > T_0}}{\sum_{k=1}^{N_S} \sum_{i=1}^{n-r} I_{d_{pi}(BC)(k) > T_0}},
\]

(4.10)

where \( I_E \) is an indicator function. If the event \( E \) holds, then \( I_E \) is equal to 1. Otherwise, it is equal to 0. Based on the best-case and worst-case delay definitions,
we have all these three match ratios to be upper bounded by 1. In addition, high match ratios indicate that we have good performance in failing path isolation. For example, high $mr_{nf}$ indicates that our predicted non-failing path set is very close to the actually non-failing path set. Others are in similar fashion.

### 4.5.1 Representative Critical Path Selection

To evaluate the performance of our representative critical path selection approaches proposed in Section 4.4, we first set nominal circuit delay (without variations) as the timing constraint and extract all paths with a timing yield-loss greater than $0.01(1 - Y)$, where $Y$ is circuit timing yield. We use these extracted paths as $\mathcal{P}_c$. In addition, we set a threshold $\epsilon = 5\%$ for approximate path selection (See Algorithm 4.1).

In Table 4.1, Columns 2 and 3 give the total number of gates and regions in each circuit ($|G|$ and $|R|$), respectively. The number of candidate failing paths denoted as $|\mathcal{P}_c|$ is given in Column 4. We can observe that for some benchmarks, we cannot extract many paths, since these circuits are intrinsically unbalanced as also stated in [55]. Column 5 shows the number of representative critical paths found from the exact approach, which is denoted as $|\mathcal{P}_r|$. This number is much smaller than the number of candidate failing paths $|\mathcal{P}_c|$ in Column 4. The prediction error in this case is zero. Column 6 shows the number of representative critical paths using the approximate approach when error tolerance $\epsilon$ in Algorithm 4.1 is set to 5%. We can observe significant reduction in the number of representative critical paths compared to Column 5. Furthermore, the number of representative critical paths is scalable to the size of $\mathcal{P}_c$ and relevant to the circuit topology. Generally, when the paths in $\mathcal{P}_c$ are more correlated, we can achieve higher reduction after Algorithm 4.1. Specifically,
Table 4.1: Results for representative critical path selection with loose timing constraint.

<table>
<thead>
<tr>
<th>BENCH</th>
<th>Configurations</th>
<th>Exact</th>
<th>Approximate</th>
<th>Inaccurate Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$</td>
<td>G</td>
<td>$</td>
<td>$</td>
</tr>
<tr>
<td>S1423</td>
<td>1045</td>
<td>21</td>
<td>644</td>
<td>122</td>
</tr>
<tr>
<td>S1488</td>
<td>702</td>
<td>21</td>
<td>16</td>
<td>14</td>
</tr>
<tr>
<td>S1494</td>
<td>711</td>
<td>21</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>S5378</td>
<td>1911</td>
<td>21</td>
<td>228</td>
<td>51</td>
</tr>
<tr>
<td>S9234</td>
<td>1688</td>
<td>21</td>
<td>608</td>
<td>53</td>
</tr>
<tr>
<td>S13207</td>
<td>2185</td>
<td>341</td>
<td>52</td>
<td>30</td>
</tr>
<tr>
<td>S15850</td>
<td>1007</td>
<td>341</td>
<td>53</td>
<td>29</td>
</tr>
<tr>
<td>S35932</td>
<td>16202</td>
<td>341</td>
<td>576</td>
<td>289</td>
</tr>
<tr>
<td>S38417</td>
<td>14772</td>
<td>341</td>
<td>692</td>
<td>190</td>
</tr>
<tr>
<td>S38584</td>
<td>11944</td>
<td>341</td>
<td>264</td>
<td>90</td>
</tr>
<tr>
<td>Ave</td>
<td>6895</td>
<td>341</td>
<td>324</td>
<td>86</td>
</tr>
</tbody>
</table>
Table 4.2: Results for representative critical path selection with tight timing constraint.

<table>
<thead>
<tr>
<th>BENCH</th>
<th>Configurations</th>
<th>Exact</th>
<th>Approx. Path Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(</td>
<td>G</td>
</tr>
<tr>
<td>S1423</td>
<td></td>
<td>1045</td>
<td>21</td>
</tr>
<tr>
<td>S1488</td>
<td></td>
<td>702</td>
<td>21</td>
</tr>
<tr>
<td>S1494</td>
<td></td>
<td>711</td>
<td>21</td>
</tr>
<tr>
<td>S5378</td>
<td></td>
<td>1911</td>
<td>21</td>
</tr>
<tr>
<td>S9234</td>
<td></td>
<td>1688</td>
<td>21</td>
</tr>
<tr>
<td>S13207</td>
<td></td>
<td>2185</td>
<td>341</td>
</tr>
<tr>
<td>S15850</td>
<td></td>
<td>1007</td>
<td>341</td>
</tr>
<tr>
<td>S35932</td>
<td></td>
<td>16202</td>
<td>341</td>
</tr>
<tr>
<td>S38417</td>
<td></td>
<td>14772</td>
<td>341</td>
</tr>
<tr>
<td>S38584</td>
<td></td>
<td>11944</td>
<td>341</td>
</tr>
<tr>
<td>Ave</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For balanced circuit S38417, we can extract 692 statistically-critical paths and 190 paths are required for exact selection with zero prediction error. With error tolerance \(\epsilon = 5\%\), we can reduce the number of representative critical paths \(|\mathcal{P}_r|\) to 44. We report \(e_1\) and \(e_2\) in Columns 7-8, which are rather small.

To further evaluate the performance of our representative critical path selection, we select a tighter timing constraint and extract more statistically-critical paths (Note that \(|\mathcal{P}_c|\) in Table 4.1 is too small). We still use the same path yield threshold to extract \(\mathcal{P}_c\). Simulation results are given in Table 4.2. Columns 2-6 report the total number of gates and regions of the circuit (\(|G|\) and \(|R|\)), the number of gates and regions covered by \(\mathcal{P}_c\) (\(|G_C|\) and \(|R_C|\)), and the number of extracted statistically-critical paths \(|\mathcal{P}_c|\), respectively. Take S38417 as an example. We extract 3,507 statistically-critical paths which cover 1,386 gates and 157 regions. Thus, for S38417, we have 1,700(= \(1,386 + 157 \times 2\)) independent process variations, where 1,386 denotes the
random variation for each gate, 157 denotes the variations in $P$ as given in Eq. (2.12) associated with one parameter, and 2 denotes the parameters of $L_{eff}$ and $V_{th0}$.

Column 7 in Table 4.2 lists the number of representative critical paths for exact prediction. We require measuring the delays of more representative critical paths compared to Column 5 in Table 4.1. The number of selected representative critical paths when $\epsilon = 8\%$ is set in Algorithm 4.1 is given in Column 8. The associated errors $e_1$ and $e_2$ are reported in Columns 9-10, respectively. We can observe that with this setting, we can still reduce the number of representative critical paths to less than 150 nearly all benchmarks. Especially for balanced circuit S38417 and S1423, we can use the delay measurement on less than 40 representative critical paths to accurately predict the post-silicon delays of around 3500 statistically-critical paths.

So far, we assume our pre-silicon model is accurate, which indicates that Eq. (4.1) can accurately capture the relationship between path delays and process variation with zero error. However, this assumption may not always hold. Here, we introduce random error in the elements of both the vector $\mu_p_c$ and matrix $A_p_c$. These errors are generated uniformly between -5\% and 5\%. In addition, we would like to note that the errors in $\mu_p_c$ can be considered to be used to capture the nonlinear relationship between $d_{p_c}$ and process variations $X$. We still consider the nominal circuit delay as the timing constraint $T_0$ and $\epsilon = 5\%$. With this simulation configuration, we perform approximate selection and report the errors $e_1$ and $e_2$ in Columns 12 and 13 of Table 4.1, respectively. Compare with Columns 7-8, we can observe that these errors get increased, but are still very small.
4.5.2 Failing Path Isolation

We first consider the nominal circuit delay as the timing constraint and $\epsilon$ to be 5%. Columns 9-11 of Table 4.1 report the match ratios we have defined in Eq. (4.10) after we perform approximate selection under this simulation configuration. We can see that these match ratios are high. Especially for $m_{nf}$, we have it to be 90.98% on average. Note that the maximum possible match ratios are 100%. High match ratios indicate good performance on failing path isolation.

To further illustrate the indications of the prediction errors and error tolerance $\epsilon$, we consider the simulation configuration with tighter timing constraint and $\epsilon = 8\%$ as in Table 4.2. Fig. 4.4 plots the histograms of $\varepsilon_i$ and $\hat{\varepsilon}_i$ for the remaining paths $p_i \in \mathcal{P}_m$. Recall that $\varepsilon_i$ and $\hat{\varepsilon}_i$ indicates the average prediction error and maximum prediction error for path $p_i$ among $N_S = 10,000$ MC samples. It shows that for different path, its maximum prediction error can be quite different, which indicates that we can vary the guardband for different paths separately.
4.6 Summary

We present a framework for post-silicon timing prediction under process variations. Given the actual delay measurements on a small set of carefully selected representative critical paths, we can accurately predict the actual timings of a large pool of candidate failing paths at the post-silicon stage. Simulation results show that we can use very few selections to predict up to 3,500 paths with very high prediction accuracy even when the dimension of process variations is larger than 1,000.
Chapter 5

Post-Silicon Timing Diagnosis of Segments on Failing Paths

In this chapter, we study the post-silicon timing diagnosis of segments on failing paths due to process variations. We propose a formal procedure that is applied after failing paths are isolated, which incorporates post-silicon path delay measurements. Our goal is to identify “failing segments” and then rank them according to their degree of failure. Here, the failing segments are referred to be those with post-silicon delays larger than their nominal delay values from the pre-silicon stage. The diagnosis of these failing segments alleviates the problem of lack of observability inside a path. Moreover, the analysis to find the cause of timing failure, and post-silicon tuning or repair, can be done more effectively by focusing on the failing segments.

We propose an Integer Linear Programming formulation to breakdown each failing path into segments, which maximizes a metric we define and refer to as “Diagnostic Resolution”. Our algorithm is fast and can yield a very high “diagnosis resolution” in identifying failing segments, and in ranking them.


5.1 Introduction

In Chapter 4, we present an efficient framework to help to isolate failing paths. In this framework, we propose to first use the method introduced in Chapter 3 to extract a large pool of statistically-critical paths to form the candidate failing paths. Due to the structural correlation among the candidate failing paths and the spatial correlation among the process variations in parameters, we are able to select a small set of representative critical paths at the design stage. The delays of these representative critical paths are accurately measured at the post-silicon stage and can help predict the actual delays of the candidate failing paths with high accuracy, and further isolate the failing paths for each fabricated chip.

The major challenge after isolating the failing paths is to discover the cause of failure in order to more effectively predict other failing paths, or to apply repair or tuning, if such infrastructure is available, for example as discussed in [29]. This “root-cause” analysis task is particularly challenging due to lack of observability inside the chip, and inability to measure segments of failing paths at the post-silicon stage.

To tackle the aforementioned problem, we present an approach to form segments on the failing paths such that their post-silicon delays are larger than their pre-silicon values. We refer to such segments as “failing segments”. Our approach is based on a formal optimization formulation to identify these “failing” segments. We also rank them according to the degree of their delay deviations. Identifying such failing segments helps towards addressing the above-stated challenges to drive a more focused analysis when access to inside a path is not possible. In addition, these identified segments can also be helpful to find layout patterns that are more timing-sensitive.

We first simplify this problem by assuming that the source of timing failure is
process variations; i.e., dependency on specific degree of process variations in a fabricated chip. We also inject a random noise representing unknown fabrication behavior in our analysis. While dynamic factors, such as crosstalk and power grid noise, are also important causes of timing failure, they are not considered in this thesis. In addition, process variations is still an important cause of the failures.

We then assume that the failing paths can be isolated using the framework such as our method discussed in Chapter 4. Delay measurements on the representative critical paths are also made using existing techniques such as [32, 73].

In this chapter, we formulate a novel Integer Linear Programming, which breaks down each failing path into provably non-failing segments, leaving the remaining edges on the path to be potentially-failing candidates. We show a very high “diagnostic resolution” in identifying the failing segments. We also provide a ranking of the failing segments based on the degree of delay deviations from the pre-silicon models, and show this ranking to be highly accurate. We categorize our contribution as in Fig. 5.1. This is the first work on segment diagnosis on failing paths.

The organization of this chapter is as follows. We introduce the preliminaries and our segment diagnosis problem formulation in Section 5.2 and Section 5.3, respectively. We then discuss our Integer Linear Programming formulation in Section 5.4. Simulation results are presented in Section 5.5 followed by conclusions.
5.2 Preliminaries

Given a timing graph $\mathcal{G}$ as defined in Section 2.1.3, we define “segment” to be a sequence of connected edges on $\mathcal{G}$. We assume that each primary input $\text{PI}_i$ and primary output $\text{PO}_j$ in this graph have zero delay, which indicates that $w_{\text{SPI}, \text{PI}_i} = w_{\text{PO}_j, \text{SPO}} = 0$.

Let us further assume process variations in parameters such as effective channel lengths. We denote $X$ to be a column vector, which includes all independent process variations, similar to Eq. (2.12). Each element in $X$ follows an independent standard Normal distribution (after scaling), which is always assumed in the existing literature such as [14, 55, 66, 69]. We use the linear model in Eq. (2.13) to approximate both gate and interconnect delays under process variations. As illustrated in Eq. (2.14), we can further express the delay of segment $s$ using the following linear expression:

$$d_s = \mu_s + a_s^T X,$$

(5.1)

where $\mu_s$ is the nominal segment delay and $a_s$ is the sensitivity vector with respect to process variations $X$. In the special case when a segment is an edge in $\mathcal{G}$, we can still get a similar linear expression. In addition, the delay of a path in the timing graph can also be written as the summation of the expressions of the associated edge weights, which are defined in Eq. (2.3). However, we would like to note that in this chapter, we do not use the linear model to approximate the circuit delay with respect to $X$, which is more prone to error than segment and path delays.

For each fabricated chip, we assume that the degree of process variations (i.e., $X$) is unknown. However, we can derive the accurate delay information on $r$ representative
critical paths $\mathcal{P}_r = \{p_1, \ldots, p_r\}$ by direct post-silicon measurement. We denote the actual post-silicon delays of these paths by vector $d_a$, while their variation-aware delays are given by

$$d_{\mathcal{P}_r} = \mu_{\mathcal{P}_r} + \mathbf{A}_{\mathcal{P}_r} \mathbf{X},$$

(5.2)

where $\mu_{\mathcal{P}_r}$ is the expected delay of the paths in $\mathcal{P}_r$, and $\mathbf{A}_{\mathcal{P}_r}$ is the matrix representing the sensitivities of the path delay in $\mathcal{P}_r$ with respect to $\mathbf{X}$.

Let us define another random variable for segment $s$:

$$\bar{d}_s \triangleq d_s | (d_{\mathcal{P}_r} = d_a).$$

(5.3)

As we can see, $\bar{d}_s$ is a conditional random variable to express the delay of segment $s$ when the $r$ representative critical paths have their post-silicon delays equal to $d_a$.

Since we assume that process variation $\mathbf{X}$ follows a Multivariate Gaussian distribution, we can conclude that $\bar{d}_s$ also follows a Gaussian distribution with mean $\bar{\mu}_s$ and standard deviation $\bar{\sigma}_s$, which can be computed using the equation below [55]:

$$\bar{\mu}_s = \mu_s + \mathbf{a}_s^T \mathbf{A}_{\mathcal{P}_r}^T \Sigma_{\mathcal{P}_r}^{-1} (d_a - \mu_{\mathcal{P}_r})$$

$$\bar{\sigma}_s^2 = \sigma_s^2 - \mathbf{a}_s^T \mathbf{A}_{\mathcal{P}_r}^T \Sigma_{\mathcal{P}_r}^{-1} \mathbf{A}_{\mathcal{P}_r} \mathbf{a}_s,$$

(5.4)

where $\Sigma_{\mathcal{P}_r} \triangleq \mathbf{A}_{\mathcal{P}_r} \mathbf{A}_{\mathcal{P}_r}^T$, and $\sigma_s$ is the standard deviation of $d_s$ at the pre-silicon stage. Since all the parameters in the above equation are available, once the representative critical path delay measurements are made, we can compute the mean and variance of $\bar{d}_s$ instantly. More importantly, the above equation shows that $\bar{\sigma}_s^2 \leq \sigma_s^2$ always
holds, which indicates that we can more accurately predict the delay for segment $s$ at the post-silicon stage when provided by the delay measurements on $\mathcal{P}_r$.

5.3 Problem Definition

In this section, we first give the definitions for different segment types. Then, we introduce the concept of “diagnostic resolution” and further formulate our problems.

5.3.1 Segment Type Definition

For each fabricated chip, we denote the degree of process variations by $X = K$ ($K$ is a constant vector). The actual delay of a segment $s$ is now denoted by the term $\bar{d}_s|_{(X=K)}$, which is derived by evaluating segment delay $d_s$ at $X = K$.

Now, we define failing and non-failing segments on a path as follows, which will be used in our segment diagnosis problem.

**Definition 5.1** Segment $s$ is defined to be “failing” $\iff \bar{d}_s|_{(X=K)} > \mu_s$.

Definition 5.1 indicates that the failing segments should have their post-silicon delays larger than their expected delays given by pre-silicon models. A segment that is not failing is referred to as “non-failing”. This definition is rather intuitive. As we will discuss later, our objective is to accurately diagnose the failing segments on the failing paths without the knowledge of the degree of process variations and only relying on post-silicon delay measurements on the representative critical paths $\mathcal{P}_r$.

The following lemma helps us to identify the non-failing segments of a given path without requiring the knowledge of the degree of process variations (i.e., $X$) in a fabricated chip.
Lemma 5.1  A sufficient condition for \( s \) to be non-failing is \( \bar{\mu}_s + 3\bar{\sigma}_s \leq \mu_s \).

Proof  Let us denote the left-hand-side of the above inequality by \( \bar{d}_s^{(wc)} \). It can be considered as the “worst-case” delay of segment \( s \), given the path delay measurements on \( \mathcal{P}_r \) at the post-silicon stage. The right-hand-side is the expected delay of segment \( s \) based on pre-silicon model in Eq. (5.1). If \( \bar{d}_s^{(wc)} \leq \mu_s \) holds, then based on Definition 5.1, we can conclude that segment \( s \) is non-failing.

Similarly, we can conclude that segment \( s \) is failing if \( \bar{\mu}_s - 3\bar{\sigma}_s > \mu_s \) holds.

The above lemma provides sufficient condition to identify some of the non-failing segments. However, it is not a necessary condition for a segment to be non-failing; i.e., a segment \( s \) can be non-failing when \( \bar{d}_s^{(wc)} > \mu_s \) holds. In this case, the conditional variance of \( d_s \) (i.e., \( \bar{\sigma}^2_s \)) is too large and the actual measurements cannot provide enough information to interpret \( \bar{d}_s \). Note that with the increase in \( r \), the number of post-silicon measurements, \( \bar{\sigma}^2_s \) generally decreases, which allows for higher number of segments to be identified as failing/non-failing.

Motivated by the above illustrations, in the absence of knowledge about the degree of process variations in a fabricated chip, we define the following types of segments:

Definition 5.2  Segment \( s \) is “definitely non-failing” \( \iff \bar{\mu}_s + 3\bar{\sigma}_s \leq \mu_s \).

Definition 5.3  Segment \( s \) is “potentially failing” \( \iff \bar{\mu}_s + 3\bar{\sigma}_s > \mu_s \).

We can see that each segment can be categorized into one of the above two types.

5.3.2 Diagnostic Resolution

For each fabricated chip, let us assume that we first measure the delays of representative critical paths \( \mathcal{P}_r \) and further isolate a set of failing paths \( \mathcal{P}_f \). Then, we break
down these failing paths into a set of non-overlapping segments $\mathcal{S}$, which can cover all the paths in $\mathcal{P}_f$. Based on Definitions 5.1-5.3, we can identify the types for each segment in $\mathcal{S}$. We denote the set of all failing segments, all definitely non-failing segments, and all potential failing segments as $\mathcal{S}_f$, $\mathcal{S}_{dnf}$, and $\mathcal{S}_{pf}$, respectively. Fig. 5.2 shows the relationship among these sets, where the dashed area corresponds to $\mathcal{S}_{dnf}$. As we can see, $\mathcal{S}_f$ is a subset of $\mathcal{S}_{pf}$. The union of $\mathcal{S}_{pf}$ and $\mathcal{S}_{dnf}$ is equivalent to $\mathcal{S}$, and $\mathcal{S}_{pf}$ is disjoint from $\mathcal{S}_{dnf}$. Note that these observations are consistent with our segment type definitions.

Before introducing the concept of diagnostic resolution, for the convenience of the notations, we first denote $L$ as the number of edges covered by $\mathcal{P}_f$. The number of edges covered by $\mathcal{S}_{dnf}$, $\mathcal{S}_{pf}$, and $\mathcal{S}_f$ are denoted by $L_{dnf}$, $L_{pf}$, and $L_f$, respectively. Fig. 5.2 illustrates that $L_{pf} + L_{dnf} = L$ and $L_{pf} \geq L_f$ always hold.

**Definition 5.4** For a given set of non-overlapping segments covering the failing paths $\mathcal{P}_f$, we define diagnostic resolution, $DR$, as

$$DR \triangleq \frac{L_f}{L_{pf}}.$$  (5.5)
The above definition shows that \textit{the value of DR varies with respect to segment formation method}. In addition, a high $DR$ indicates that the set of potentially failing edges in $S_{pf}$ is close to the actually failing ones in $S_f$, and that this segment formation $S_{pf}$ for the failing paths is a good one.

Now we formally define our segment diagnosis problem as follows.

**Problem Definition:** Given post-silicon delay measurements of $r$ representative critical paths $\mathcal{P}_r$, and $m$ failing paths $\mathcal{P}_f$ which have been isolated, our objective is to divide $\mathcal{P}_f$ into a set of non-overlapping segments and form $S_{pf}$ such that diagnostic resolution given in Eq. (5.5) is maximized.

Fig. 5.3 gives the visualization of our above mentioned problem formulation. Note that our focus in this chapter is on segment diagnosis, assuming that the failing paths and the delay measurements on the representative critical paths are provided. We intend to form potentially failing segments such that $DR$ is maximized.
5.4 Our Proposed Algorithm

In this section, we first introduce a mathematical formulation for our segment diagnosis problem, and then discuss how it is linearized into an Integer Linear Programming (ILP) formulation.

5.4.1 Mathematical Formulation

The objective of our segment diagnosis problem is to maximize $DR$ as defined in Eq. (5.5). Since $L_f$, the numerator of $DR$, is unknown for an instance of process variations, we alternatively minimize the denominator $L_{pf}$ which is the number of edges in the potentially-failing segments. On the other hand, since $L_{pf} + L_{dnf} = L$ holds, minimizing $L_{pf}$ is equivalent to maximizing $L_{dnf}$ which is the number of edges covered by the definitely non-failing segments. Therefore, we translate this segment diagnosis problem to forming the definitely non-failing segments set $S_{dnf}$ so that they can cover more edges on the failing paths. This leaves the remaining edges to be potentially-failing segments (of length 1).

Our Proposed Approach: We first introduce a formulation for the simple case when we solve the segment diagnosis problem on a single failing path, given post-silicon delay measurements on $r$ representative critical paths. Then, we can iteratively apply this formulation on all the $m$ failing paths in $P_f$.

The above approach solves the segment diagnosis problem from path to path, which indicates that the ordering of the paths in solving this problem may influence the quality of the solutions (derived $DR$). In order to alleviate the influence, when we solve the simple case for a single failing path, we describe a definitely non-failing
segment as a consecutive connection of edges, and ensure that distinct definitely non-failing segments are apart by at least one edge. Let us consider the case when there exist two paths $p_1$ and $p_2$, and they share some edges which can form definitely non-failing segments. Our requirement for the consecutiveness makes it very likely that these shared edges will be grouped together to form definitely non-failing segments when solving the segment diagnosis problem no matter on path $p_1$ or on path $p_2$. Here, we also use Fig. 5.4(a) as an example to show a valid solution for solving the segment diagnosis problem for a single failing path. The failing path given in this figure is composed of two definitely non-failing segments, while the remaining edges $e_3$ and $e_8$ are potentially-failing segments of length 1.

In the remaining of this subsection, we will introduce a mathematical formulation for the simple case when we solve the segment diagnosis problem for a single failing path. We start from giving our notations with an example.

**Notations:** Consider one failing path with $L$ edges $e_1, \ldots, e_L$, and all edges are labeled in an increasing order from the path input to its output. For the definitely non-failing segment set $\mathcal{S}_{\text{dnf}}$, we denote its $j$-th segment by $s_{\text{dnf}_j}$, and assume that these segments are indexed in increasing order from the path input towards its output.

Since we require that the definitely non-failing segments be apart from each other
by at least one edge, an upper bound on the number of definitely non-failing segments is \(N_m = \lceil \frac{L}{2} \rceil\), where \([x]\) denotes the smallest integer not less than \(x\). The remaining edges between these segments belong to \(S_{pf}\). We further define binary variables \(x_{ij}\) for \(i = 1, 2, \ldots, L\) and \(j = 1, 2, \ldots, N_m\). If edge \(e_i\) belongs to the \(j\)-th definitely non-failing segment \(s_{dnf_j}\), we set \(x_{ij}\) to 1. Otherwise, \(x_{ij}\) is equal to 0, and the edge \(e_i\) belongs to the potentially-failing segments set \(S_{pf}\).

Since the number of definitely non-failing segments is unknown and we only have its upper bound, we define binary variables \(\Delta_j\) to represent whether \(s_{dnf_j}\) exists for \(j = 1, 2, \ldots, N_m\). If \(\Delta_j\) is equal to 1, then it indicates that \(s_{dnf_j}\) has been formed. Otherwise, it means that the \(j\)-th definitely non-failing segment is not formed.

We further define variables \(l_j\) and \(u_j\) to help identify the starting and ending indices of \(s_{dnf_j}\) for \(j = 1, \ldots, N_m\). If \(s_{dnf_j}\) does not get formed (i.e., \(\Delta_j = 0\)), then we force \(l_j = u_j = L\). Otherwise, we can form \(s_{dnf_j}\), and \(u_j\) returns the edge index of the last edge covered by \(s_{dnf_j}\) while \(l_j\) gives the index of the edge immediately before the first edge covered by \(s_{dnf_j}\). Fig. 5.4(b) illustrates the definitions of \(l_j\) and \(u_j\).

**Example:** Consider Fig. 5.4(a). This failing path includes \(L = 8\) edges. The maximum number of definitely non-failing segments \(N_m\) is equal to \(L/2 = 4\). For the particular solution shown in this figure, we have two definitely non-failing segments \(s_{dnf_1}\) and \(s_{dnf_2}\). The remaining edges, i.e., \(e_3, e_4\) and \(e_8\), belong to \(S_{pf}\). Therefore, we have \(\Delta_1 = \Delta_2 = 1\) and \(\Delta_3 = \Delta_4 = 0\). For \(s_{dnf_2}\), we have \(l_2 = 4\) and \(u_2 = 7\). We also have \(u_3 = l_3 = u_4 = l_4 = L = 8\).

Now we describe the mathematical formulation for our segment diagnosis problem for a single path as below:
max \[ \sum_{i=1}^{L} \sum_{j=1}^{N_m} x_{ij} \] \hspace{1cm} (5.6)

\[ \text{s.t.} \quad \sum_{j=1}^{N_m} x_{ij} \leq 1, \quad \forall i = 1, \ldots, L \] \hspace{1cm} (5.7)

\[ \check{\mu}_s + 3\check{\sigma}_s \leq \mu_s, \quad \forall j = 1, \ldots, N_m \] \hspace{1cm} (5.8)

\[ \sum_{i=1}^{L} x_{ij} = u_j - l_j, \quad \forall j = 1, \ldots, N_m \] \hspace{1cm} (5.9)

\[ l_j = L - \max_{i=1, \ldots, L} (L + 1 - i)x_{ij}, \quad \forall j = 1, \ldots, N_m \] \hspace{1cm} (5.10)

\[ u_j \geq \max_{i=1, \ldots, L} (ix_{ij}), \quad \forall j = 1, \ldots, N_m \] \hspace{1cm} (5.11)

\[ u_j + \Delta_{j+1} \leq l_{j+1}, \quad \forall j = 1, \ldots, N_m - 1 \] \hspace{1cm} (5.12)

\[ \frac{l_{j+1} - (L - 1)}{L} \leq 1 - \Delta_{j+1} \leq \frac{l_{j+1}}{L}, \quad \forall j = 2, \ldots, N_m - 1 \] \hspace{1cm} (5.13)

\[ x_{ij} \in \{0, 1\}, \quad \forall i = 1, \ldots, L; j = 1, \ldots, N_m \] \hspace{1cm} (5.14)

\[ \Delta_j \in \{0, 1\}, \quad \forall j = 1, \ldots, N_m \] \hspace{1cm} (5.15)

Eq. (5.6) in the above formulation expresses our objective. We aim to maximize DR, which alternatively turns into maximizing \( L_{dnf} \). The \( L_{dnf} \) is equal to the total number of edges covered by the set \( S_{dnf} \). Note that based on our definition for \( x_{ij} \), the edges that are covered by \( S_{dnf} \) will have \( x_{ij} = 1 \) and otherwise 0.

Eq. (5.7) indicates that each edge on the path can be covered by at most one definitely non-failing segment. For edge \( e_i \), if all \( x_{ij}s \) are equal to 0 for \( j = 1, 2, \ldots, N_m \), then \( e_i \) belongs to the set \( S_{pf} \) of potentially-failing edges.

Eq. (5.8) ensures that the \( j \)-th segment formed by our formulation is \( s_{dnf,j} \). This is consistent with the definition of definitely non-failing segment as explained in Section 5.3.1.

Eq. (5.9) ensures that each definitely non-failing segment is composed by one con-
tinuous ordering of consecutive edges. This can be expressed based on our definition of variables $l_j$ and $u_j$ given for segment $s_{dnf_j}$. The inequality basically states that the summation of the number of edges covered by $s_{dnf_j}$ should be equal to $u_j - l_j$.

Eq. (5.10) and (5.11) describe our definitions of variables $l_j$ and $u_j$ in terms of $x_{ij}$. The highest edge index on segment $s_{dnf_j}$, denoted as $u_j$, should satisfy Eq. (5.11). Similarly, Eq. (5.10) gives the expression for $l_j$. In the example of Fig. 5.4, we verify these expressions as $l_2 = 8 - \max(4, 3, 2) = 4$, and $u_2 = \max(5, 6, 7) = 7$. Note that even though the expression for $u_j$ is an inequality, Eq. (5.9) forces this to become an equality for formed segments $s_{dnf_j}$.

Eq. (5.12) uses variable $\Delta_{j+1}$ to ensure that the distinct definitely non-failing segments are at least apart by one edge. Recall that if the $(j + 1)$-th segment $s_{dnf_{j+1}}$ is formed, then we have $\Delta_{j+1} = 1$. Consider the case when we have $\Delta_{j+1} = 1$, and we can derive $u_j + 1 \leq l_{j+1}$, indicating that the formed segments $s_{dnf_j}$ and $s_{dnf_{j+1}}$ are apart by at least one potentially-failing edge.

In our formulation, we define $N_m$ number of $\Delta_j$ variables, where $N_m$ is the maximum number of $S_{dnf}$ segments we can possibly form. If we are not able to form $N_m$ number of definitely non-failing segments, then the $\Delta_j$s with higher indexes of $j$ will be automatically set to 0 for satisfying Eq. (5.12) such that we have $u_j \leq l_{j+1}$. Combine this case with Eq. (5.9), and we can conclude that when $\Delta_{j+1} = 0$, we have $u_k = l_k = L$ for $k \geq j + 1$. We use this special relationship to enforce the definition of $\Delta_{j+1}$ variable which we discuss next.

The binary variable $\Delta_{j+1}$ is defined using Eq. (5.13). When $\Delta_{j+1} = 0$, we have $u_j \leq l_{j+1} = L$. The inequality in Eq. (5.13) turns into $\frac{L - L + 1}{L} \leq 1 \leq \frac{L}{L}$, which always holds. When $\Delta_{j+1} = 1$, we have $\frac{l_j - L}{L} \leq 0 \leq \frac{l_{j+1} - L}{L}$, which is also a valid inequality.
Therefore, we enforce our definition of $\Delta_{j+1}$ using the alternative conditions of $u_j \leq l_{j+1} = L$ (when $\Delta_{j+1} = 0$) and $u_j + 1 \leq l_{j+1}$ (when $\Delta_{j+1} = 1$).

### 5.4.2 Linearization

In the mathematical formulation presented in Section 5.4.1, Eqs. (5.8), (5.10), (5.11) are in non-linear form. Here, we describe how to linearize these constraints without approximation.

The Eqs. (5.10) and (5.11) are nonlinear due to the max operation. In general, we can define an auxiliary variable $y_{\text{max}}$ to replace the max expressions, which are in the form of $\max_{i=1,\ldots,a}(y_i)$. Then, we enforce $y_{\text{max}}$ to satisfy: $y_1 \leq y_{\text{max}}, \ldots, y_a \leq y_{\text{max}}$.

Now, we are left to explain how Eq. (5.8) can be linearized. First, we equivalently rewrite Eq. (5.8) as

$$\mu_{s_j} - \bar{\mu}_{s_j} \geq 0, \quad \forall j = 1, 2, \ldots, N_m$$

$$\left(\mu_{s_j} - \bar{\mu}_{s_j}\right)^2 \geq 9\bar{\sigma}_{s_j}^2, \quad \forall j = 1, 2, \ldots, N_m$$

where Eq. (5.16) can be transformed into

$$\mu_{s_j} - \bar{\mu}_{s_j} \geq 0 \iff \sum_{i=1}^{L} x_{ij} (\mu_{e_i} - \bar{\mu}_{e_i}) \geq 0.$$

The $\mu_{e_i}$ and $\bar{\mu}_{e_i}$ in the above equation are defined using Eqs. (5.1) and (5.4), when the segment is just a single edge. It also makes use of the linearity property in expectation on conditional random variables in writing $\mu_{s_j}$ and $\bar{\mu}_{s_j}$ in terms of $\mu_{e_i}$ and $\bar{\mu}_{e_i}$, respectively. Note that we can precompute $\mu_{e_i}$ and $\bar{\mu}_{e_i}$ for all edges using
Eqs. (5.1) and (5.4). Consequently, Eq. (5.16) becomes in linear form.

To linearize Eq. (5.17), we first rewrite its left-hand-side in the following linear form:

\[
(\mu_s - \bar{\mu}_s)^2 = \sum_{k=1}^{L} \sum_{i=1}^{L} x_{ij} x_{kj} (\mu_{ei} - \bar{\mu}_{ei}) (\mu_{ek} - \bar{\mu}_{ek}).
\]  

(5.19)

Then, we introduce auxiliary variables \(\min(x_{ij}, x_{kj})\) to replace \(x_{ij} x_{kj}\) since both \(x_{ij}\) and \(x_{kj}\) are binary variables. In addition, we need to introduce additional linear constraints for these auxiliary variables expressing the min operation, which is very straightforward and similar to the max operation as illustrated before.

Now, we are left to transform the right-hand-side of Eq. (5.17) into a linear form. Using Eq. (5.4), we can express \(9\bar{\sigma}^2_s\) as

\[
9\bar{\sigma}^2_s = 9 \left( \sigma^2_s - a^T_s A^T_P \sum^{-1} P_r A_P a_s \right) = 9a^T_s T a_s
\]  

(5.20)

where \(T = I - A^T_P \sum^{-1} P_r \) is a constant matrix and can be pre-computed.

Let us denote the element in the \((i,j)\)-th entry of \(T\) and that in the \(k\)-th entry of \(a_s\) by \(t_{ij}\) and \(a_{sk}\), respectively. We can then rewrite \(\bar{\sigma}^2_s\) into

\[
\bar{\sigma}^2_s = \sum_{k=1}^{m} a_{sk}^2 t_{kk} + \sum_{i \neq k}^{m} a_{si} a_{sk} (t_{ik} + t_{ki}),
\]  

(5.21)

where we have two non-linear terms \(a_{sk}^2\) and \(a_{si} a_{sk}\). Due to the similarity, here we only discuss how to linearize the first term. We can follow the same procedures to
linearize the other one. For \( a_{sjk}^2 \), we have

\[
a_{sjk}^2 = \left( \sum_{i=1}^{L} a_{eik} x_{ij} \right)^2 = \sum_{i=1}^{L} \sum_{l=1}^{L} a_{eik} a_{elk} x_{ij} x_{lj}
\]  

(5.22)

Note that we have already introduced auxiliary variables to linearize \( x_{ij} x_{lj} \), as discussed for Eq. (5.19). Thus, we do not need additional variables and constraints to linearize \( a_{sjk}^2 \).

So far, we have shown that by introducing auxiliary variables, we can transform the mathematical formulation presented in Section 5.4.1 into an Integer Linear Programming (ILP) without any approximation. Next, we will show in our simulation results that solving ILP for each path is extremely fast (fraction of a second). Therefore, even for a very large number of failing paths at the post-silicon stage, we can still afford to iteratively apply the ILP for each path in order to solve our segment diagnosis problem. This method can still give us high accuracy and efficiency.

5.5 Experimental Results

We implement our framework using C++, and use CPLEX 9.0 to solve the generated ILP formulation. In our experimental flow, we start by synthesizing ISCAS’89 benchmarks using 90nm TSMC library and Synopsys Design Compiler for minimum area under a stringent timing constraint to ensure having many critical paths. We assume process variations in effective channel length \( L_{\text{eff}} \) and zero-bias threshold voltage \( V_{\text{th0}} \) to be Gaussian distributed with standard deviations of 5% and 10% of their mean, respectively.
To capture spatial correlation between the varying parameters, we use the multi-level hierarchical model as illustrated in Section 2.2.1, which defines rectangular regions on the chip. The gates/interconnects in the same region or in physically close-by regions will share all or some of their process variations and be correlated to each other using this hierarchical model.

Similar to Section 4.5 and [55], for smaller benchmarks (S1423 to S9234), we use a 3-level hierarchical model, resulting in 21 regions. In the remaining benchmarks, we use a 5-level model with 341 regions for each benchmark. For each region, we have two distinct random variables associated with $L_{eff}$ and $V_{th0}$.

To derive a “golden model” capturing post-silicon behavior in our simulations, we assume the degree of process variation $X$ is known for each simulation case corresponding to one fabricated chip. We then experiment by analyzing many such $X$ generated using MC simulation to reflect various post-silicon fabrication cases. Furthermore, for each case, we assume an additional Gaussian-distributed error of $\epsilon_i$ is introduced for the delay of each gate $i$ representing a mismatch between pre- and post-silicon delay models. We randomly generate this error for each gate and assume it to be at most 6% of the nominal gate delay in our simulations.

Our framework requires as input, post-silicon delay measurements on representative critical paths (which can include failing ones). While our framework can take as input any method for selecting and measuring representative critical paths, the important factor in our simulations is to ensure that the measured paths are indeed representative so they can more effectively reflect the silicon impact. We briefly summarize our procedure to get representative critical paths for measurement:

1. We first extract a large pool of critical paths $P_c$, using pre-silicon timing models
for the nominal case (without variations). These paths are selected if their nominal delays fall within a 20% window of the timing constraint.

2. We then use the procedure presented in Chapter 4 (also given in [42]) to identify a subset of these paths denoted by $\mathcal{P}_r$, as representative critical paths ($\mathcal{P}_r \subseteq \mathcal{P}_c$). These paths are highly correlated with the target critical paths in presence of process variations. We indeed verify that our representative critical paths could predict the delays of critical paths with an average error of less than 3% over a large set of process variation cases.

3. We assume the measured delays of these representative critical paths are obtained using the explained “golden model”. So we assume an instance of process variations $\mathbf{X}$ and an instance of random noise $\mathbf{e}_i$ for each gate $i$, and incorporate these in Eq. (5.2) to find the delay of each path for each simulation.

5.5.1 Comparisons on Diagnostic Resolution

We generate MC samples for process variations in device parameters, which follow Gaussian distributions. Due to the very large dimension of process variations, we generate several MC sample sets and each one includes 2,500 samples. We chose the sample set, which yields the largest number of failing samples, for performance validation. We declare an instance to be a “failing sample” if at least one path in $\mathcal{P}_C$ fails the timing. Recall that we assume failing paths are identified and isolated using existing techniques, such as our proposed approach in Chapter 4 and the one in [59]. Here, we use our golden model to isolate the set of failing paths $\mathcal{P}_f$ for each variation instance, while the representative critical paths are determined once and
Table 5.1: Results of our segment diagnosis framework for two sets of failing paths.

| BENCH  | $| \mathcal{P}_r |$ | $N_{FS}$ | $N_{FP}$ | $DR$ (%) | $\frac{L_{FS}}{L_{FP}}$ (%) | $L_{FP}$ | $\Delta N_e$ | $DR_p$ (%) | $N_{FS}$ | $N_{FP}$ | $DR$ (%) | $DR_p$ (%) |
|--------|----------------|----------|----------|------------|-----------------|--------|-------------|------------|--------|----------|------------|------------|
| S1423  | 119            | 1581     | 23.56    | 86.72      | 68.76           | 61.14  | 18.22       | 18         | 92.20  | 12.65    | 90.84      | 94.81      |
| S1488  | 24             | 1289     | 2.09     | 85.46      | 74.74           | 59.91  | 10.16       | 9          | 86.92  | 1.79     | 89.61      | 91.07      |
| S1494  | 11             | 1016     | 1.32     | 72.18      | 83.64           | 80.81  | 11.48       | 6          | 73.64  | 1.21     | 81.75      | 80.96      |
| S5378  | 61             | 1547     | 19.36    | 87.27      | 69.12           | 62.80  | 9.99        | 7          | 90.89  | 1041     | 16.03      | 91.44      |
| S9234  | 53             | 1228     | 141.80   | 71.23      | 60.32           | 55.38  | 11.27       | 9          | 76.35  | 835      | 122.10     | 72.97      |
| S13207 | 37             | 1935     | 4.58     | 75.57      | 74.37           | 70.32  | 11.21       | 9          | 81.70  | 1137     | 2.90       | 82.79      |
| S15850 | 34             | 1294     | 13.30    | 62.84      | 59.64           | 54.83  | 8.81        | 9          | 66.55  | 756      | 12.87      | 63.75      |
| S35932 | 295            | 624      | 267.16   | 71.12      | 53.63           | 46.12  | 7.37        | 4          | 83.84  | 468      | 267.97     | 71.96      |
| S38417 | 173            | 1950     | 56.09    | 82.06      | 53.77           | 40.59  | 11.76       | 14         | 86.95  | 1122     | 40.24      | 86.19      |
| S38584 | 101            | 1286     | 72.56    | 73.45      | 44.69           | 31.07  | 8.44        | 6          | 79.89  | 826      | 69.59      | 75.25      |
| Ave    |                |          |          |            |                 |        |             |            | 81.89  |          | 80.65      | 85.54      |
remain the same over all variation instances. The number of failing samples $N_{FS}$ is given in Column 3 of Table 5.1. Note this number is high compared to the total number of samples for some benchmarks. It is because 2,500 samples are selected to cover a high number of failing samples, representing various failure possibilities. In addition, this number also differs among benchmarks due to their topologies.

For each failing sample, we first compute the delays of representative critical paths $\mathcal{P}_r$ according to our golden model and consider them as post-silicon measurements. We then implement our framework using these measurements to diagnose the segments on those identified failing paths. Here, we consider two sets of failing paths $\mathcal{P}_{f,1}$ and $\mathcal{P}_{f,2}$. For an instance of process variations, $\mathcal{P}_{f,1}$ includes all failing paths in $\mathcal{P}_f$ with timing violation less than 3% of the timing constraint. It represents small violation case. Similarly, $\mathcal{P}_{f,2}$ includes all failing paths with timing violation between 3% and 6% of the target circuit delay. It represents large violation case. Note that the paths in $\mathcal{P}_{f,1}$ and those in $\mathcal{P}_{f,2}$ are different for each variation case and are defined based on the post-silicon delays.

In Table 5.1, Column 2 provides the number of representative critical paths ($|\mathcal{P}_r|$) for direct measurement. Columns 3-10 show the performance of our framework for small violation case where the segment diagnosis is performed over the set of $\mathcal{P}_{f,1}$. Column 4 gives the average number of failing paths ($N_{FP}$) in our MC simulation. Column 5 gives the average diagnostic resolution ($DR$) after the ILP formulation. As we can see, the average $DR$ over all the circuits is very high, which is 76.79%. *Note that in our simulations, the actual failing segment set $\mathcal{S}_f$ was indeed contained in the potentially failing segment set $\mathcal{S}_{pf}$.*

Columns 6-7 give the ratios of the average number of potential failing edges ($L_{pf}$)
with respect to the total number of edges covered by the failing paths ($L$). We compute this percentage before ILP (where each edge is a segment of length 1 of either definitely non-failing or potentially failing types). We also compute this percentage after ILP (where some edges are merged into definitely non-failing segments). As can be seen, after ILP, we can reduce this percentage by on-average 8% in small and 13% in large benchmarks.

We also report the average length of the failing paths ($L_{FP}$) in Column 8. Column 9 gives the maximum number of edges which can be merged ($\Delta N_e$) by using ILP in our MC Simulation. Compared with Column 8, we sometimes merge a large proportion of edges into non-failing segments.

Column 10 gives the average diagnostic resolution of a path ($DR_p$) over MC samples. The $DR_p$ is defined with respect to each path; the ILP is solved for each path and the diagnostic resolution measured for each path separately, and then averaged over the number of paths and the MC samples. As can be seen, $DR_p$ is very high, indicating that the set of potentially-failing edges per path is very close to the actually-failing ones, if the paths are individually considered.

Columns 11-14 give simulation results for large violation case (the set of failing paths is $P_{f,2}$). In this case, we can observe that the values of $DR$ and $DR_p$ are higher than those in the small violation case. This could be due to the fact that in the large violation case, there are likely more number of failing edges on the paths ($L_f$ is larger).

To further show the effectiveness of our ILP formulation, we perform another experiment considering all the failing paths (for all the post-silicon delay violation cases). For each path, we record the number of edges merged into definitely non-
failing segments after applying our ILP formulation. We also record the number of actual failing edges (obtained using the golden model) over all the violation cases. Fig. 5.5 shows the scatter plots of these two quantities for S1488. As shown, for the cases of small delay violations, we can merge more edges into non-failing segments. For the cases of large violations, since more edges are prone to failure, there is less benefit in solving the ILP.

Note that solving ILP for each path is extremely fast (fraction of a second), which makes our algorithm practical.

5.5.2 Evaluation of Ranking The Failing Segments

So far, we have shown in simulation results that we can achieve very high diagnostic resolution, which indicates that our diagnosed potentially failing segments are very likely to fail the timing at the post-silicon stage. In this subsection, we will show that we can also accurately rank the failing segments based on their degree of failures (i.e., post-silicon delay deviations from their estimated pre-silicon values).
Table 5.2: Results for the ranking of failing segments and sharing information over failing paths.

<table>
<thead>
<tr>
<th>BENCH</th>
<th></th>
<th>MRRL (%)</th>
<th>Sharing Info</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>20%</td>
<td>40%</td>
</tr>
<tr>
<td>S1423</td>
<td>1045</td>
<td>93.98</td>
<td>94.23</td>
</tr>
<tr>
<td>S1488</td>
<td>702</td>
<td>95.71</td>
<td>95.70</td>
</tr>
<tr>
<td>S1494</td>
<td>711</td>
<td>89.97</td>
<td>89.97</td>
</tr>
<tr>
<td>S5378</td>
<td>1911</td>
<td>92.52</td>
<td>92.38</td>
</tr>
<tr>
<td>S9234</td>
<td>1688</td>
<td>84.51</td>
<td>85.88</td>
</tr>
<tr>
<td>S13207</td>
<td>2185</td>
<td>87.77</td>
<td>87.86</td>
</tr>
<tr>
<td>S15850</td>
<td>1007</td>
<td>82.16</td>
<td>82.77</td>
</tr>
<tr>
<td>S35932</td>
<td>16202</td>
<td>89.42</td>
<td>91.26</td>
</tr>
<tr>
<td>S38417</td>
<td>14772</td>
<td>90.65</td>
<td>90.97</td>
</tr>
<tr>
<td>S38584</td>
<td>11944</td>
<td>91.68</td>
<td>92.46</td>
</tr>
<tr>
<td>Ave</td>
<td>89.84</td>
<td>90.35</td>
<td>93.20</td>
</tr>
</tbody>
</table>

We consider small violation case (i.e., the set of failing paths is \(\mathcal{P}_{f,1}\)). We define two ranked lists of 1) potentially failing segments, and 2) actually failing ones. This allows us to evaluate a match ratio as we discuss later. The two lists are constructed as follows. After applying our framework, we obtain a set of potentially failing segments \(\mathcal{S}_{pf}\), and for each one compute the expected segment delay given the path measurements (i.e., \(\bar{\mu}_s\)) using Eq. (5.4). We rank these segments in non-increasing order in terms of the segment delay deviation \((\bar{\mu}_s - \mu_s)\), and derive a rank list \(\mathcal{L}_1\). We also rank all the segments in \(\mathcal{S}_f\) in terms of their actual post-silicon delay deviations from \(\mu_s\) in non-increasing order to get the list \(\mathcal{L}_2\). Note that the length of all segments in \(\mathcal{L}_1\) and \(\mathcal{L}_2\) is 1.

Using a control parameter \(\eta \in (0, 1]\), we define a window of size \(m\) to compare the segments of the two ranked lists, where \(m \triangleq \max(\eta|\mathcal{L}_2|, \min(10, |\mathcal{L}_2|))\). This ensures that for fair comparison, the window size is at least 10 segments, or the entire list (if
the list size is less than 10 segments).

We then consider the first $m$ segments of the two ranked lists $\mathcal{L}_1$ and $\mathcal{L}_2$ and define the metric “match ratio of the rank list” as $\text{MRRL}(\eta) = \frac{n}{m}$, where $n$ is the number of segments in the window of $\mathcal{L}_2$ which were contained in the window of $\mathcal{L}_1$. A high value of MRRL indicates a high accuracy of our framework. Table 5.2 gives MRRL for $\eta = 20\%$, $40\%$, and $100\%$. Column 2 gives the total number of gates in the entire circuit. Columns 3-5 show that for nearly all cases, MRRL is larger than $85\%$ indicating a large matching. It indicates that our framework can determine the top $\eta$ ($\%$) failing segments very accurately, even when $\eta$ is small.

5.5.3 Evaluation of Sharing in Failing Paths

In this section, we provide the sharing information among the failing paths. For the failing paths, first we consider the failing edges and compute the average number of failing paths going through each failing edge (over all the paths and MC samples). We denote this by $n_{fe}$. Next, we consider all the failing and non-failing edges of the failing paths. We then compute the average number of failing paths going through each edge. We denote this by $n_e$. Columns 6-7 in Table 5.2 give the simulation results. As we can see, $n_{fe}$ is higher than $n_e$, which indicates that the edges which are covered by a higher number of failing paths are more probable to fail. We plan to incorporate this observation to improve our framework in future.
5.6 Summary

We present a framework for fast post-silicon segment diagnosis on failing paths, which is based on our proposed ILP formulation. We show in our simulation results that we can achieve a very high diagnostic resolution of failing segments on the failing paths. A failing segment has a post-silicon delay larger than its pre-silicon one. We further rank the failing segments based on the degree in their post-silicon delay deviation and show a high accuracy in this ranking.
Chapter 6

Conclusions and Future Work

With continuous technology scaling, the timing of fabricated ICs becomes very sensitive to process variations. Pre-silicon timing analysis can only provide the statistical information of the timing over an entire population of the fabricated chips. However, after fabrication, the designers may require the actual timing information for the “slow” chips to localize their timing failure and perform post-silicon repair. However, the process becomes increasingly complicated due to the lack of access to the inside of the chip after fabrication. Moreover, it is very expensive or nearly impossible to derive the actual values for all device parameter variations in each fabricated chip. Therefore, post-silicon timing diagnosis becomes very time-consuming and costly.

6.1 Conclusions

In this thesis, we provide an automatic framework to help localize the timing failures for fabricated chips, which fails to meet a required frequency constraint. We assume that the cause of the timing failures is only process variations, which follow multi-
variate Gaussian distributions. Although some dynamic factors, such as crosstalk and power grid noise can also result in timing failures, we believe that focusing on process variations can be considered as a first step for post-silicon timing diagnosis. We summarize our provided framework as follows:

First, we present a bound-based approach to efficiently and accurately extract the statistically-critical paths under process variations. These paths are the paths with the highest probability to violate the circuit timing, and thereby can be considered to form a good pool of candidate failing paths after silicon. In our methods, we first define violation probability for each node/edge in the circuit timing graph to measure the likelihood that the failing paths would go through them. These probabilities can be pre-computed by performing two rounds of Statistical Static Timing Analysis. Then, we further define segment violation probability to be the likelihood that the failing paths would go through this segment. We show that we can derive tight upper/lower bound for the violation probability of any arbitrary segment with constant computational complexity by making use of node/edge violation probabilities. Therefore, we are able to develop a very efficient approach to construct the statistically-critical paths from the primary inputs to the primary outputs with one traversal of the circuit timing graph. Experimental results show that our developed approach can extract the statistically-critical paths efficiently and accurately.

Then, we propose a novel scheme to select a small set of representative critical paths from the above extracted statistically-critical paths. These selected representative critical paths should have their delays highly correlated with the delays of the remaining statistically-critical paths. To facilitate this selection, we consider both the path structure correlations (the sharing of the gates/interconnects among different
paths) and the spatial correlation among the process variations in device parameters. We utilize the concept of “effective rank” to efficiently identify the representative critical paths and build a model from the delays of the representative critical paths to the remaining statistically-critical paths. In addition, our selection of representative critical paths keep unchanged for different fabricated chips with the same design. Simulations show that for each fabricated chip, by directly measuring the delays of around 150 representative critical paths after fabrication, we can predict the post-silicon delays of around 3,000 remaining statistically-critical paths accurately. Since our selection and prediction are performed over statistically-critical paths, which are more prone to violate the timing after fabrication compared to other paths in the circuit, our approach can help isolate the failing paths at the post-silicon stage.

Finally, we introduce a framework to diagnose the segments on the failing paths, which is proposed to be performed by using the post-silicon measurements on representative critical paths after the failing paths are isolated. In our introduced framework, our goal is to identify a set of segments on the failing paths, which are very likely to have their post-silicon delays larger than their pre-silicon values. All remaining edges should be able to form into segments which have post-silicon delays smaller than their pre-silicon values. Moreover, we maximize “diagnostic resolution”, which is defined as the ratio of the total number of failing edges among our identified segments divided by the total number of edges covered by our identified segments. Particularly, we propose an Integer Linear Programming formulation to identify these segments and further evaluate this formulation using Monte Carlo simulation. We show in experimental results that we can achieve very high diagnostic resolution and further rank these segments based on the degree of their failures accurately.
6.2 Future Work

Future extensions to improve our research work presented in this thesis include:

- In this thesis, we assume that process variations always follow Gaussian distributions, and that the gate/interconnect delays have a linear relationship with respect to those variations. However, these assumptions may not be accurate enough with further technology scaling. We can generalize our framework to handle non-Gaussian and nonlinear cases as follows:
  - In Chapter 3, we have illustrated that our proposed statistically-critical path extraction approach is only dependent on node/edge violation probabilities. These probabilities can be pre-computed using any SSTA approach, which can handle nonlinear case and non-Gaussian variations.
  - In Chapter 4, we use “effective rank” to select representative critical paths. The procedures are only relevant with the transformation matrix $A_P$ in Eq. (4.1), which models the relationship between the path delays $d_P$ and process variations $X$. Therefore, our proposed approach can handle non-Gaussian process variations easily, but handling non-linear case requires further investigation. In addition, for both non-Gaussian and non-linear cases, instead of using linear model as given in Theorem 4.2, we may need more complicated models, such as quadratic one, to predict the delays of remaining paths $d_m$ using the delays of representative critical paths $d_r$.
  - In Chapter 5, both our proposed mathematical formulation in Section 5.4.1 and linearization in Section 5.4.2 require the assumptions of linear delay model and Gaussian process variations. Extending this step remains a challenge, but may be an important problem to solve.
In Chapter 4, we propose to measure the delays of the representative critical paths at the post-silicon stage, and then further use these information to predict the delays of the remaining statistically-critical paths. However, at the design stage, we can construct a number of “custom” test structures on a chip. If these structures have their delays highly correlated with the representative segments covered by the statistically-critical paths, then we may be able to reduce the number of measurements and accurately predict the post-silicon delays of the statistically-critical paths. Developing a solution to the above problem is another exciting direction to explore.

In Chapter 5, we solve segment diagnosis problem from one failing path to another, which requires the definitely non-failing segments to be formed by consecutive edges. We can release this requirement by introducing a new mathematical formulation, where for each failing path we have a constraint. More specifically, this constraint expresses that for each failing path, all the edges on this path which do not belong to potentially failing segments would be bound together as one entity satisfying the definition of definitely non-failing segment. This can potentially improve the solution quality, but may increase the computational complexity, especially when the number of failing paths is large.

In conclusion, post-silicon timing diagnosis is still an open problem and extremely difficult to solve. In addition to process variations, many other factors, such as crosstalk and power-grid noise, would also effect the diagnosis quality. However, we expect that the work presented in this thesis can help to alleviate the pressures for chip designers in localizing timing failures after fabrication.
Bibliography


