FAULT MODELING, DELAY EVALUATION AND PATH SELECTION FOR DELAY TEST UNDER PROCESS VARIATION IN NANO-SCALE VLSI CIRCUITS

A Dissertation

by

XIANG LU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2005

Major Subject: Computer Engineering

FAULT MODELING, DELAY EVALUATION AND PATH SELECTION FOR DELAY TEST UNDER PROCESS VARIATION IN NANO-SCALE VLSI CIRCUITS

A Dissertation

by

XIANG LU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Weiping Shi
Committee Members,	Duncan M. Walker
	Gwan Choi
	Andreas Klappenecker
Head of Department,	Costas Georghiades

December 2005

Major Subject: Computer Engineering

ABSTRACT

Fault Modeling, Delay Evaluation and Path Selection for Delay Test Under Process Variation in Nano-scale VLSI Circuits. (December 2005)

Xiang Lu, B.S., Xi'an Jiaotong University;

M.S., Xi'an Jiaotong University

Chair of Advisory Committee: Dr. Weiping Shi

Delay test in nano-scale VLSI circuits becomes more difficult with shrinking technology feature sizes and rising clock frequencies. In this dissertation, we study three challenging issues in delay test: fault modeling, variational delay evaluation and path selection under process variation. Previous research of fault modeling on resistive spot defects, such as resistive opens and bridges in the interconnect, and resistive shorts in devices, lacked an accurate fault model. As a result it was difficult to perform fault simulation and select the best vectors. Conventional methods to compute variational delay under process variation are either slow or inaccurate. On the problem of path selection under process variation, previous approaches either choose too many paths, or missed the path that is necessary to be tested.

We present new solutions in this dissertation. A new fault model that clearly and comprehensively expresses the relationship between electrical behaviors and resistive spots is proposed. Then the effect of process variations on path delays is modeled with a linear function and a fast method to compute coefficients of the linear function is also derived. Finally, we present the new path pruning algorithms that efficiently prune unimportant paths for test, and as a result we select as few as possible paths for test while the fault coverage is satisfied. The experimental results show that the new solutions are efficient and accurate.

DEDICATION

To my parents

ACKNOWLEDGMENTS

It has been a great pleasure working with the faculty, staff, and students at Texas A&M University, College Station, during my Ph.D. program starting from the fall of 2000. The doctoral study has been possible with the help of a lot of people.

This dissertation would never have been completed if I were not given plenty of freedom to pursue my research interests, thanks in large part to the kindness and advising provided by Dr. Weiping Shi, my long-time advisor and the committee chair. He patiently provided the vision, encouragement and advice necessary for me to proceed through the doctoral program and complete my dissertation. I have learned from him to proceed with my research interests in the future. I really appreciate his kindness and help, and consider myself fortunate to have been one of his students.

I would like to thank Dr. Duncan M. (Hank) Walker for his great help and advice on my research work. His intuitive ideas and industrial concerns guided me in the correct research directions and focus. It was a great time working with him in the Semiconductor Research Corporation (SRC) project under contract 2000-TJ-844.

Special thanks to my committee, Dr. Weiping Shi, Dr. Duncan M. (Hank) Walker, Dr. Gwan Choi, and Dr. Andreas Klappenecker, for their support, guidance and helpful suggestions.

I also benefited much from the faculty in Computer Engineering, Dr. Jiang Hu, Dr. Sunil Khatri, and Dr. Peng Li. I have learned a lot from their research and ideas presented in the weekly seminar, and the discussions with them were very helpful to my research work.

I am grateful to Zhuo Li and Wangqi Qiu for their great help in my research work. Zhuo Li helped me solve difficult research problems and provided a lot of experimental data to verify research results. He also did a good job in fault modeling of resistive bridges to make the fault modeling work complete. Wangqi Qiu was a great partner in the SRC project under contract 2000-TJ-844. We had a good time working together and my research benefited much from him. I would also like to thank Jing Wang and Senthikumar Veluswami. Their research work, timing analysis under power noise and statistical timing analysis, provided helpful attempts to extend parts of my research work.

Also, thanks to Mrs. Carolyn A. Warzon for her help in the group of Computer Engineering. I thank Mrs. Tammy Carda and the Department of Electrical Engineering for providing me with a Teaching Assistant position in the spring of 2005.

My research was funded in part by the SRC grant 000-TJ-844, NSF grants CCR-0098329, CCR-0113668, EIA-0223785, and ATP grant 512-0266-2001. I thank these corporations for providing financial support.

I had a great team of officemates: Zhijun Cai, Xiaonan Ma, Zhiquan Qiu, Dongming Peng, Cheng-Ta Chiang, Shu Yan, Zhuo Li, C. N. Sze, Yong Liu, Haiyun You, Chuan He, Le Zou, Wentao Zhao, etc. I thank them for making me very comfortable in the group of Computer Engineering and for making the study in Werc111 a memorable one.

Finally, I want to thank my wife Ziding Yue and my parents for their constant support in my doctoral program the past 5 years. Their support and believing in my abilities were indispensable for me to complete the dissertation. I would also like to thank my brother Tao Lu, my sister-in-law Chun Yue and my parents-in-law for their support and patience.

TABLE OF CONTENTS

LIST OF FIGURES

Page

Page

LIST OF TABLES

I. INTRODUCTION

The 2002 International Technology Roadmap for Semiconductors (ITRS) [1] projects the at-speed test as an increasingly difficult problem. With the shrinking feature size in VLSI technology and the rising clock frequencies, traditional functional and delay test approaches are becoming either infeasible or inefficient. More challenging issures, such as fault modeling for resistive spot defects and effect of process variations, are necessary to be considered in at-speed test.

1. Resistive Spot Defect

There are two types of spot defects in the interconnect – opens and bridges. Opens are unintended impedance of a wire, and are considered a wire cut-off if the impedance becomes infinite. Bridges are unintended electrical connections between interconnects. The spot defects inside MOS transistors, gate oxide shorts, are unintended electrical connections through the gate oxide between the gate and the source, drain, or channel of a MOS transistor. Gate oxide shorts have been identified as a major type of fabrication defect and, in some CMOS processes, the dominant defect [2].

In traditional logic tests, spot defects are considered to have zero resistance and are modeled as stuck-at faults, such as [3][4]. However, spot defects have a wide range of resistance values, which plays an important role in both logic and delay behaviors. For example, Hawkins and Soden [2] showed that the experimentally measured gate oxide short

---

This dissertation follows the style of *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*.

resistance is from 0.8k to 4.7k ohms. They also found that gate oxide shorts cause degraded voltage levels and increased propagation delays. Hao and McCluskey [5] studied circuit behaviors of resistive shorts, and the dependence on input signals of the faulty gate and the driving gate. Renovell *et al*. [6] showed that the voltage behavior of gate oxide shorts related to the short resistance, short locations and short sizes. In Degraeve *et al.* [7], a method to determine the breakdown position in short channel NMOSFETs is introduced.

To detect resistive spot defects in delay test, an accurate and realistic fault model that considers the short resistance is necessary. Gaitonde and Walker [8] studied problems faced in mapping spot defects to changes of circuit topology, and found it is more difficult to map gate oxide shorts than to map interconnect shorts. Segura *et al.* [9] developed electrical models of gate oxide shorts, and showed that, depending upon location and transistor type, the short can be resistive, diode, parasitic MOSFET or parasitic BJT. Hao and McCluskey [10] modeled the gate oxide short as a resistor connecting the gate and the source or the drain of a transistor. Based on the model, they showed the change of the logic and delay under different signal patterns. However, the model does not give a quantitative relationship between the delay change and the short resistance, and cannot be used to measure the quality of the test pattern in delay test. Sar-Dessai and Walker [11] presented several logic fault models for resistive bridge faults. Moore *et al.* [12] presented comprehensive delay fault analysis for resistive bridge models and coupling effects, but the delay calculation was not given. Other techniques, such as the mixed-mode simulation method by Chuang and Hajj [13] and neural network techniques by Shaw *et al*. [14], give more accurate bridge fault models. But these methods are not efficient for large circuits due to their high time complexity. In [15], we proposed a circuit level fault model for resistive opens and bridges in the interconnect. However, resistive shorts in MOS transistors are more complex and the approach in [15] is insufficient.

2. Process Variation

Process variations are unintended deviations from the requested manufacturing parameters. There are two distinct sources of variation: *environmental factors* and *physical factors* [16]. Environment factors include variations in the power supply voltage and the temperature. These factors are highly design dependent and exhibit time constants similar in scale to the clock frequency. Physical factors include variations in the electrical and physical parameters characterizing the behaviors of wires and devices. These variations are caused by processing and mask imperfections and various wear-out mechanisms. Physical factors exhibit long time constants, typically measured in years, and can be further divided into *inter-die* variations and *intra-die* variations. Inter-die variations are independent of the design implementation, and are considered globally in die-to-die, wafer-to-wafer and lot-to-lot areas. Intra-die variations are dependent of design implementation and are considered locally.

With the decreasing feature size of VLSI technology and the rising clock frequencies, the impact of process variations is increasingly felt. A great amount of research has been done recently on process variations, such as the clock skew analysis under process variation [17][18][19], statistical performance analysis [20][21][22], worst case performance analysis [23][24], parametric yield estimation [25], impact analysis on micro architecture [26] and delay fault test under process variation [27][28][29][30].

In previous research, the variational path delay under process variation is modeled either as a function of process variables [17] [18][20][24][25][30], or as a random variable of certain distribution [19][22][28][31]. However, the conventional methods to compute path delay are either slow or inaccurate. The response surface method (RSM), which performs multiple simulations and curve-fittings, is used in [20] [24] [25] [31]. To achieve high accuracy, the RSM method must perform multiple parasitic extractions and delay evaluations under different process conditions. Due to the large number of metal layers in the modern technology, there are many interconnect process variables. For example, for a k-layer technology, there are 3k process variables related to interconnect, corresponding to the metal width, metal thickness and inter-layer dialectic thickness of each layer. As a result, the traditional RSM has prohibitive running cost for large circuits. Orshansky *et al.* [23] derived delay sensitivity to gate length variation based on a simple model, and expressed delay as a function of gate length. Their method does not automatically apply to interconnect process variation due to the lack of a similar model for the interconnect. For methods that treat the path delay as a random variable, it is also necessary to extract parasitic RCs and compute the path delay under different process conditions [22] [28], resulting in too much timing cost.

3. Path Selection

Delay test of digital integrated circuits is to ensure that the signal from any primary input to any primary output is propagated in less time than the specification. A circuit is considered faulty if the delay of any path exceeds the specification. A delay increase due to a local defect, such as a resistive bridge or a resistive open, may cause a timing violation on the path through the defect, which is modeled as a delay fault [15][32]. Such a delay increase is localized to a gate input, output or an interconnect wire in the circuit, where the localized position is called a *local fault site*. Testing the longest path through the local fault site will capture the delay increase due to the fault. When process variation is not considered, the problem of finding the longest and testable paths that cover all local fault sites has been extensively studied [33][34][35]. In these methods, only one path with the maximum delay is tested for each local fault site.

When process variation is considered, the path delay becomes a function of process variables. Among all paths through a fault site, there are often multiple paths whose delay can be the maximum under different process conditions [36]. For each fault site s, we call a path *longest* for s if the path has the maximum delay among all paths through s under some process conditions. On the other hand, we call a path *redundant* for s if the path can never be longest for s under any process condition. For example in Figure 1, there are four longest paths, P_1, P_2, P_3 and P_4, through a fault site in ISCAS85 circuit c432 using TSMC 180 nm technology. Two process variables, x_2 and x_3, represent metal thickness variations of Metal 2 and Metal 3 respectively. In this example, four paths form the upper bound of the delay for all paths through the fault site within the range of process variation. Any path whose delay is below this bound is redundant for the fault site.

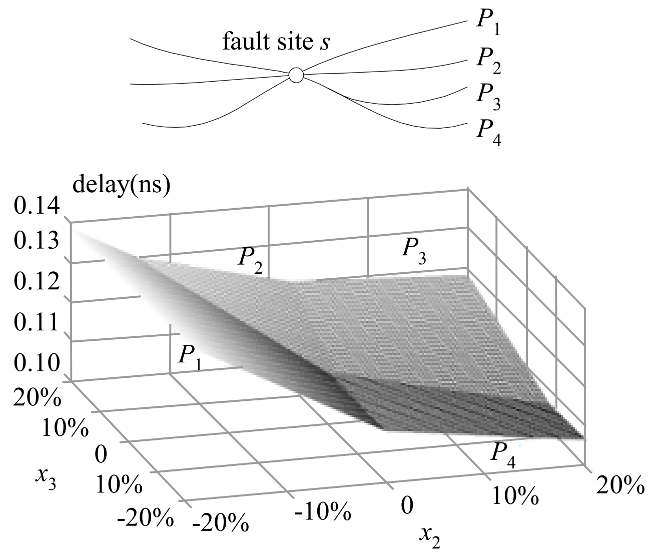


Figure 1. Delay of four longest paths under process variation.

Traditionally, tests are only performed on the longest paths under the nominal or worst-case process condition. However, this might be insufficient. In Figure 1, P_1 is the longest under the nominal process condition ($x_2 = 0$, $x_3 = 0$) and also the worst-case process condition ($x_2 = -20\%$ and $x_3 = 20\%$). However, under process condition $x_2 = 20\%$ and $x_3 = -20\%$, P_1 is much shorter than P_4. Since it is hard to know the actual process variation for a chip under test, we must test all paths that could be longest under any process condition. In this dissertation, we propose techniques to select all longest paths through each fault site to maximize the fault coverage.

Modern delay optimization tools tend to make many paths critical or near critical [37], resulting in too many paths for test. Pruning some of the paths based on structural correlation and process variation correlation is an effective approach to reduce the number of paths. If two paths share some nets or gates, there is a *structural correlation* between them. Similarly, if two nets run on the same metal layer, there is a *process correlation* between them. Luong and Walker [27] proposed a pruning technique using both the structural correlation and the process correlation. As a result, they significantly reduced the number of paths. However, they only considered the longest paths for the entire circuit, instead of the longest paths through every local fault site. Furthermore, they did not consider interconnect delay. Tani *et al*. considered the longest paths through every local fault site [38]. They used a min-max comparison method, with the help of the structural correlation but not process correlation. As a result, their approach is overly pessimistic and produces too many paths. Liou *et al*. [28] used Monte Carlo simulation to select a set of critical paths that maximizes the probability of covering all critical paths under all process conditions. However, Monte Carlo simulation is very slow for large circuits and no running time is given for their method.

4. Solutions

In this dissertation, we put attention to the impact of these three issues on delay test, and present our new solutions.

On resistive spot defects, we propose an accurate and realistic fault model for delay test in CMOS circuits. The restive spot defect is modeled as a logic fault or a delay fault depending on the resistance and input patterns. The delay change is approximated as a function of the short resistance. Based on the new fault model, we present a method to select the best test vectors to cause the worst-case effect of the short on the circuit. As an application, we quantitatively evaluate the performance of the delay test and the logic test, and show the improvement using the new fault model.

On the computation of variation delay due to process variation, we present a new method PARADE for fast PARAmetric Delay Evaluation using analytical formulae and pre-characterized lookup tables. We first model variational path delays as linear functions of process variables. By analyzing a small sample of parasitic capacitance extracted from any commercial parasitic extraction tools, we derive an efficient method to compute the effect due to process variation for parasitic capacitance. Then the variational path delay is evaluated efficiently, based on the lumped C delay model and based on the effective capacitance delay model respectively. No multiple parasitic extractions and multiple delay evaluations are needed for both methods, resulting in a significant speedup over the traditional RSM. The efficiency of our methods makes it possible to comprehensively analyze circuit performance on all interconnect and device process variables for large circuits. We do experiments on ISCAS85 circuits under TSMC 180nm 1.8V 6-metal layer technology. Experiments show that our methods achieve high accuracy and efficiency. Compared to the traditional RSM, the delay error is within 5% using analytical methods, and is within 3% using the table lookup method.

On longest path selection, we present a new method to select longest paths for each local fault site in the circuit. To maximize fault coverage, we want to find as many longest paths as possible. On the other hand, to minimize test costs, we want to find as few paths as possible. Given a set of testable paths, our method first models the path delay as a linear function of process variation variables, then uses two pruning algorithms to remove paths that are redundant or almost redundant. We repeat the process for each fault site in the circuit, and the remaining paths are longest paths for delay test. Experiments on the ISCAS circuits show that the new method is efficient and significantly reduces the number of paths for test, compared to the previous best method proposed in [38]. We consider process variations of devices and interconnect in our work, and the method can also be applied in path selection under operating variations of supply voltages and temperature [39].

5. Organization

The dissertation is organized as follows. In Section II we analyze electrical behaviors of resistive open/bridge in the interconnect and resistive short in devices, and present the new fault model for each type of the spot defect. In Section III, the new method to compute variational delay under process variation is presented. In Section IV we propose the path pruning algorithms to select longest paths for test, and show the experimental result presents using the new algorithms. We conclude our work in Section V.

II. FAULT MODELING OF INTERCONNECT RESISTIVE SHORTS

1. Resistive Open in Interconnect

Resistive opens can be classified as strong opens (>10MΩ) and weak opens (≤10MΩ) [32]. Strong opens cause stuck-at faults, while weak opens cause delay change in the interconnect. Strong opens cause stuck-at faults and, can be detected by regular stuck-at patterns. Weak opens cause delay faults and thus may not be detected by regular stuck-at patterns [2][12]. Montanes and Gyvez [32] showed that in modern nano-scale circuits, the percentage of weak opens is high and delay test for such defects is necessary.

The resistive open fault model is shown in Figure 2. In this model, a resistive open is represented by a resistor r_o in a net at the location where the open defect may occur. The input buffer B_1 and output buffer B_2 represent arbitrary CMOS gates.

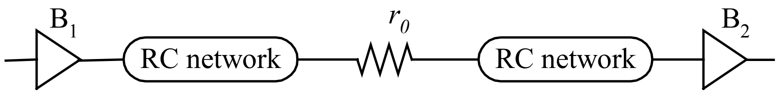


Figure 2. Resistive open fault model.

From extensive SPICE simulation, we found the delay increases almost linearly with the open resistance. Figure 3 shows the SPICE simulation result of a typical net using TSMC 250nm technology.

The open resistance is modeled as an increased delay *d'* in the net. The increased delay *d'* is approximated by a linear function $d' = r_0 / R_{nominal} \cdot d$, where r_0 is the open resistance, $R_{nominal}$ is the interconnect resistance without open and *d* is the nominal delay. Above a certain value, depending on the clock frequency of the circuit the open becomes a stuck-open fault.

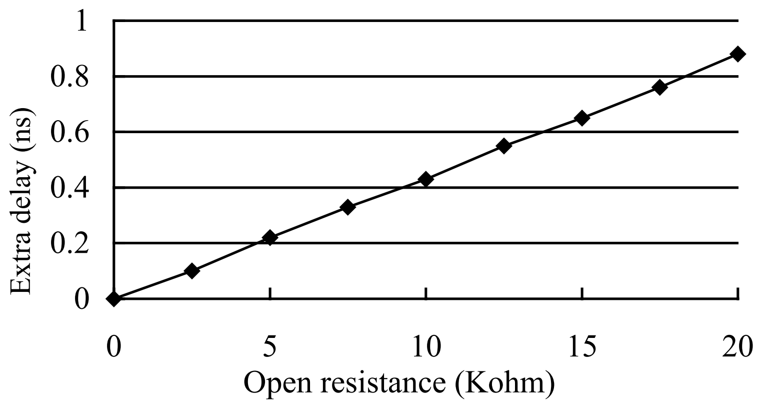


Figure 3. Delay increases linearly with open resistance.

2. Resistive Bridge in Interconnect

The resistive bridge circuit model is shown in Figure 4. Each gate B_i is an arbitrary CMOS gate. To simplify the analysis, CMOS devices in B_1 and B_2 are replaced by switches and linear resistors in Figure 4, and B_3 and B_4 are replaced by buffers. We use a simple RC interconnect model that lumps interconnect parasitic capacitance with the load capacitance.

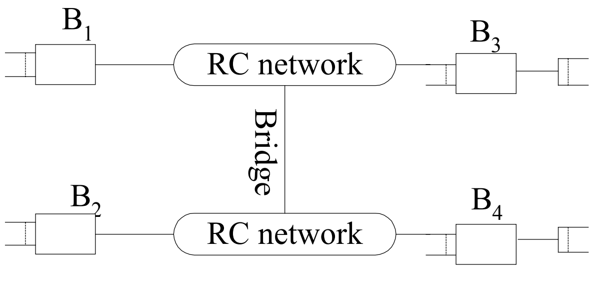


Figure 4. The resistive bridge circuit model.

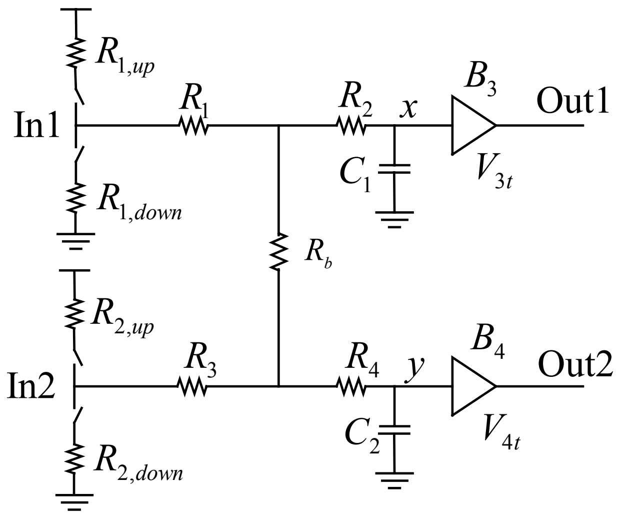


Figure 5. The simplified resistive bridge circuit model.

Circuit parameters in Figure 5 include pull-up and pull-down resistances $R_{1,up}$, $R_{1,down}$, $R_{2,up}$ and $R_{2,\mathrm{down}}$ of B_1 and B_2, interconnect parasitic resistances R_1, R_2, R_3 and R_4, bridge resistance R_b, capacitances C_1 and C_2 that includes interconnect and sink capacitances, and logic interpretation voltages V_{3t} , V_{4t} of B_3 and B_4. The logic interpretation voltage V_t of a buffer is defined as follows. If the input of the buffer is below V_t, the output is considered logic low. If the input of the buffer is above V_t, the output is considered logic high. For inverters or other gate types, the definition is similar with some “high” and “low” exchanged. Inputs of B_3 and B_4 are denoted as x and y, respectively. For simplicity, the delay of Out1 (Out2) means the delay at x (y).

A. Static Analysis

In static analysis, it is assumed that input signals remain constant and output signals are stable. Therefore, all interconnect parasitic capacitances and sink capacitances are ignored. There are four possible cases of input patterns in the static analysis. When In1 and In2 are both high, or both low, the bridge has no impact on the circuit. When In1 is low and In2 is high, the circuit is shown in Figure 6.

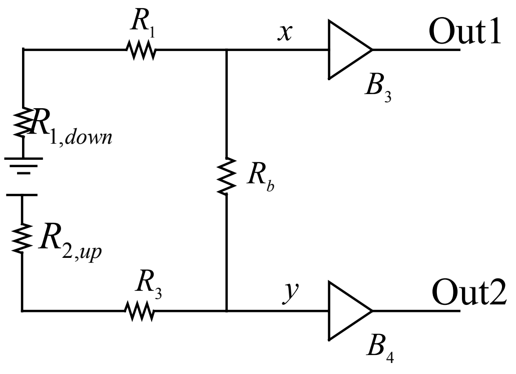


Figure 6. The circuit model when In1 is low and In2 is high.

Define the Bridge Threshold Resistance (BTR) for Out1 as

$$R_{1,Vss} = \frac{Vdd(R_1 + R_{1,down})}{V_{3t}} - (R_1 + R_3 + R_{1,down} + R_{2,up}) \tag{2.1}$$

When $R_b < R_{1,Vss}$, the voltage at x is greater than V_{3t} and Out1 is high, which is a logic fault. When $R_b > R_{1,Vss}$, there is no logic fault, but there might be an increased delay, which is discussed in the Section 2.2. The relationship between Out1 and R_b is illustrated in Figure 6.

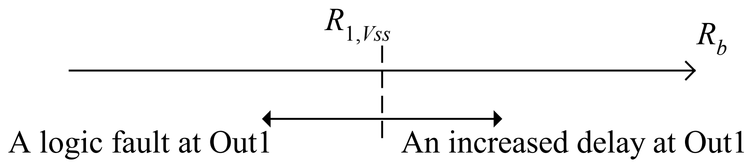


Figure 7. The relationship between R_b and Out1.

Similarly for Out2, the BTR is

$$R_{2,Vdd} = \frac{V_{4t}(R_3 + R_{1,up})}{Vdd - V_{4t}} - (R_1 + R_{1,down}) . \tag{2.2}$$

The case when In1 is high and In2 is low is symmetric. The corresponding BTRs are given as follows:

$$R_{1,Vdd} = \frac{V_{3t}(R_1 + R_{1,up})}{Vdd - V_{3t}} - (R_3 + R_{2,down}) , \tag{2.3}$$

$$R_{2,Vss} = \frac{Vdd(R_3 + R_{2,down})}{V_{4t}} - (R_1 + R_3 + R_{2,down} + R_{1,up}) . \tag{2.4}$$

It is known that for Boolean functions with two inputs, only four are monotone and non-constant. Therefore, the behavior of Out1 in Figure 5 can only be one of the four in Figure 8. Table 1 summarizes the above analysis and shows which model the circuit behaves. The concept of Bridge Threshold Resistance is similar to the concept of critical (limit, detectable) resistance studied in previous work [6][9][10]. In this dissertation, simple formulas to compute BTRs are presented.

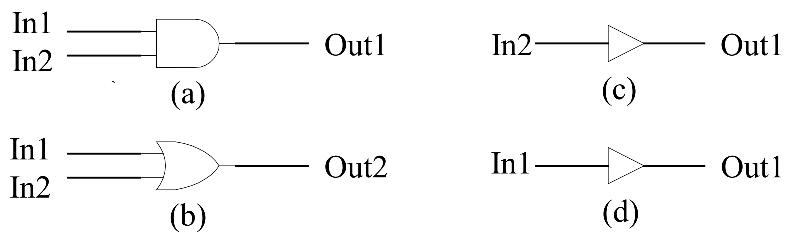


Figure 8. Four basic resistive bridge fault models for static analysis.

TABLE 1. The bridge fault model for Out1.

R_b range	Out1 Model
$R_b \leq \min(R_{1,Vdd}, R_{1,Vss})$	(c)
$R_{1,Vss} < R_{\mathrm{b}} < R_{1,Vdd}$ (if $R_{1,Vss} < R_{1,Vdd}$)	(a)
$R_{1,Vdd} < R_b < R_{1,Vss}$ (if $R_{1,Vdd} < R_{1,Vss}$)	(b)
$R_b \geq \max(R_{1,Vdd}, R_{1,Vss})$	(d)

Some useful properties can be derived directly from the models. For example, it is impossible for all BTRs to be greater than zero. Here is a simple proof. If all BTRs are set to be greater than zero, then from Eq. (2.1) and (2.2), we can derive that $V_{4t} > V_{3t}$. Similarly, from Eq. (2.3) and (2.4), we can derive that $V_{3t} > V_{4t}$, which is a contradiction. Therefore, all BTRs cannot be greater than zero (or less than zero with a similar proof) at the same time. Thus, Out1 and Out2 cannot behave as the model shown in Figure 8(c) simultaneously, that is, there exist some input vectors that make logic values of two outputs not be swapped. When $R_b < \max(R_{1,Vdd}, R_{1,Vss}, R_{2,Vdd}, R_{2,Vss})$, Out1 and Out2 cannot behave as the model shown in Figure 8(d) simultaneously, that is, there must be a logic fault at either Out1 or Out2.

B. Dynamic Analysis

In the dynamic analysis, there are four types of input signals: high, low, rising (from low to high), and falling (from high to low). According to the static analysis, the output behavior eventually settles down to one of the four fault models in Figure 8, determined by BTR values. There are totally 16 cases of input type combinations for In1 and In2. The analysis for all cases is similar to the following case.

Consider the case when In1 is rising and In2 is low. The circuit in Figure 5 can be simplified to the circuit in Figure 9. If $R_b \leq R_{1,Vdd}$, the static analysis shows that there is a logic fault for Out1, which can be detected by logic tests. If $R_b > R_{1,Vdd}$, there is no logic fault.

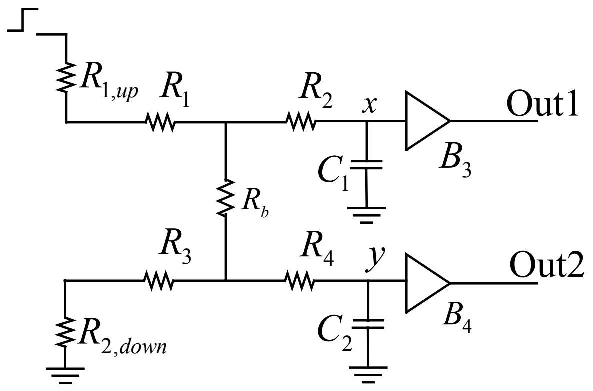


Figure 9. The circuit model when In1 is rising and In2 is low.

From the circuit analysis by matching the second moment of transfer function [40], we found that when there is a rising input on In1, the behavior of x in Figure 9 can be approximated by the behavior of x in Figure 10, where

$$C_e = C_1 + \frac{(R_{2,down} + R_3)^2}{(R_{2,down} + R_3 + R_b)^2} \cdot \frac{C_2}{1 + |R_4 - R_2| / (R_{1,up} + R_1 + R_2)}$$

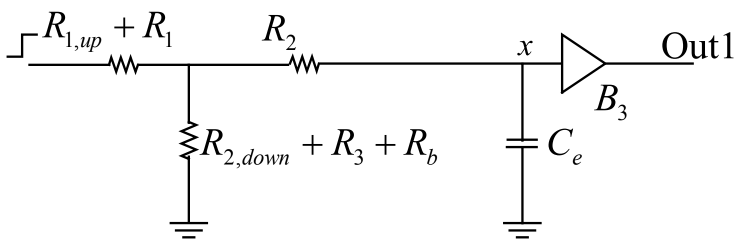


Figure 10. The approximation circuit model for Out1 when In1 is rising and In2 is low.

The coefficient $\frac{1}{1 + |R_4 - R_2| / (R_{1,up} + R_1 + R_2)}$ is achieved experimentally to balance the interconnect resistance R_4 and R_2. When $R_4 = R_2$, the following fact is true: the first two moments of driving admittance [41] in Figure 9 and 10 are the same, which is

$$\frac{1}{R_{2,down} + R_3 + R_b + R_{1,up} + R_1} + \frac{(R_{2,down} + R_3)^2 C_2 + (R_{2,down} + R_3 + R_b)^2 C_1}{(R_{2,down} + R_3 + R_b + R_{1,up} + R_1)^2} s + \mathrm{O}(s^2)$$

In the approximation, B_3 is only regarded as a sink capacitance that is included in C_1 in Figure 9 and C_e in Figure 10. In Figure 9, if the bridge does not exist in the circuit, then the equivalent circuit is shown in Figure 11(a), where $R_{line} = R_2 + R_1 + R_{1,up}$.

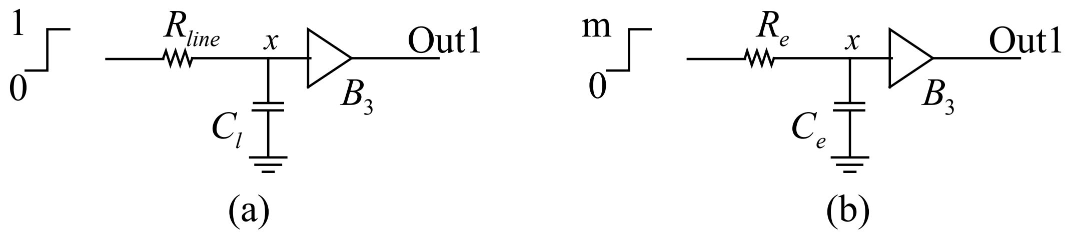


Figure 11. (a) The equivalent circuit of Figure 9 when there is no bridge. (b) The equivalent circuit of Figure 10.

Define the delay at *x* in Figure 11(a) as d_1, then

$$d_1 = -R_{line} \cdot C_1 \cdot \ln(0.5) \tag{2.5}$$

Similarly, the equivalent circuit of Figure 10 is shown in Figure 11(b), where

$$R_e = R_2 + (R_1 + R_{1,up}) // (R_{2,down} + R_3 + R_b),$$

$$m = \frac{R_{2,down} + R_3 + R_b}{R_1 + R_{1,up} + R_{2,down} + R_3 + R_b}$$

Here, the symbol "//" represents the parallel computation of two resistances. Define the delay at x in the Figure 11(b) as d_2, then we can get

$$d_2 = -R_e \cdot C_e \cdot \ln(1 - \frac{0.5}{m}). \tag{2.6}$$

Since the peak voltage at x in Figure 11(b) is only a fraction of Vdd, there is an increased delay at Out1. Intuitively, m can be seen as the voltage division ratio, and R_e can be seen as the effective resistance from upstream to *x*. When $R_b \to \infty$, $C_e = C_1$, $R_e = R_{line}$, $m = 1$ and $d_2 = d_1$.

From Eq. (2.5) and (2.6), the increased delay $d' = d_2 - d_1$ can be computed as

$$d' = -a \cdot b \cdot \log_2(1 - 0.5/m) \cdot d_1, \tag{2.7}$$

where $a = C_e/C_1$, $b = R_e/R_{line}$. In Eq. (2.7), a can be seen as the ratio between the effective capacitance with and without the bridge, b can be seen as the ratio between the effective resistance with and without the bridge. When the input pattern is (falling, low), then $d' = -a \cdot b \cdot \log_2(0.5/m) \cdot d_1$, where all parameters have similar meanings to parameters in (rising, low) case except for different values. Generally, if the initial voltage value of a rising input in Figure 11(b) is defined as g, the static value after the dynamic process is defined as h, then

$$d' = -a \cdot b \cdot \log_2(1 - \frac{0.5 - g}{h - g}) \cdot d_1 = -a \cdot b \cdot \log_2(\frac{0.5 - h}{g - h}) \cdot d_1 \ . \tag{2.8}$$

If the initial value of a falling input is g, the static value after the dynamic process is h, it is interesting that d_0 can be written in the same format as in the rising case except for different parameter values.

In Eq (2.8), we write d_0 as the function of d_1 since we can calculate d_0 from d_1, which is a more accurate value such as the delay including the cell delay from SPICE simulation or delay tables. Our equation can also be easily modified when there is a ramp input in Figure 8. We can abstract the ramp input by a step input applied at the instant when the ramp crosses the 50% point and an extra delay $\tau/2$, where τ is the slope of the ramp input [42]. Now Eq. (2.8) is modified to

$$d_2 = \frac{-a \cdot b \cdot \log_2((0.5 - h)/(g - h))}{1 + (\tau/2/(-R_{line} \cdot C_1 \cdot \ln(0.5)))} \cdot d_1 \tag{2.9}$$

Through SPICE simulation the bridge resistance can increase or decrease the delay (d'

can be greater or less than zero) depending on input patterns. This was also mentioned in previous work [12]. Figure 12 shows the SPICE simulation of two interconnect segments from the layout of the ISCAS85 circuit c432. The bridge resistance is 1 KΩ. There is an increased delay at Out1 when input pattern (In1, In2) is (falling, high), and a decreased delay at Out1 when the input pattern is (rising, high). The decreased delay may cause a hold time violation or race at Out1. This type of fault cannot be detected by the current delay test.

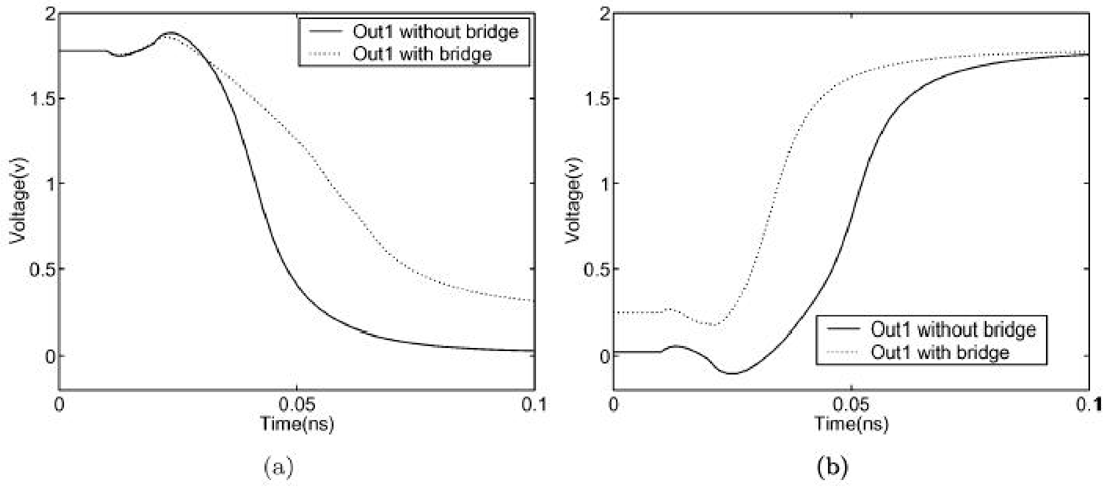


Figure 12. (a) A bridge causes an increased delay at Out1 when input pattern is (falling, high). (b) The bridge causes a decreased delay at Out1 when input pattern is (rising, high).

C. Modeling Procedure

Based on the above analysis, we derive the bridge fault model as follows. All logic and delay faults are included in the model. Previous fault models such as the aggressor-victim model [12] are special cases of this model.

(1) Compute $R_{1,up}$, $R_{1,down}$, $R_{2,up}$, $R_{2,down}$, V_{3t} and V_{4t} from the cell library and the input pattern for cells other than inverter/buffers. Compute R_1, R_2, R_3, R_4, C_1 and C_2 from the interconnect parasitics.

(2) Compute BTR values $R_{1,Vdd}$, $R_{1,Vss}$, $R_{2,Vdd}$ and $R_{2,Vss}$ according to equations (2.1) to (2.4).

(3) For the fault simulation, R_b is given. Use R_b to choose a fault model from Figure 12 according to Table 1. When there is a delay fault, compute

$$d' = (-l \cdot \log_2((0.5-h)/(g-h)) - 1) \cdot d_1, \tag{2.10}$$

where d_1 is the nominal delay of Out1, l, g and h are chosen according to Table 2.

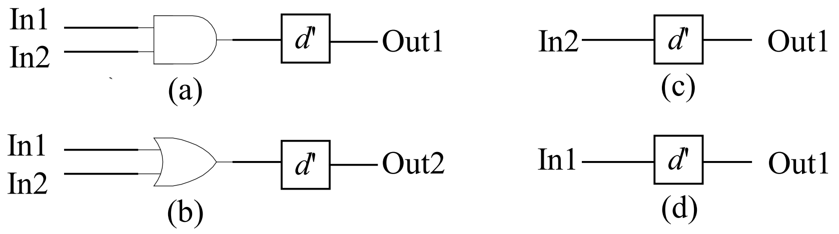


Figure 13. Four basic resistive bridge fault models.

In Table 2, behaviors at Out1 for Figure 12(d) with all input patterns are presented. If both In1 and In2 change, it is assumed that two inputs change simultaneously. If two inputs do not change simultaneously, we treat the case as the combination of two cases happening sequentially. For example, if both inputs are rising and In1 is faster, then this case is consistent with the combination of (r, 0) and (1, r).

TABLE 2. Increased Delay(ID) or Decreased Delay (DD) at Out1 for Figure 10(d), *r* means rising, *f* means falling, 1 means high, 0 means low, and other variables are defined in equation series.

Input Pattern (In1, In2)	Out1 Model
Both static (0, 0) , (0, 1), (1, 0), (1, 1)	No ID nor DD
Same direction (*r*, *r*), (*f*, *f*)	
In1 static (0, *r*), (0, *f*), (1, *r*), (1, *f*)	
(*r*, 0)	ID, $g = 0$, $h = m_1$, $l = a_1 \cdot b_1$
(*f* , 0)	DD, $g = m_1$, $h = 0$, $l = a_1 \cdot b_1$
(*r*, *f*)	ID or DD, $g = m_2$, $h = m_1$, $l = a_1 \cdot b_1$
(*f*, *r*)	ID or DD, $g = m_1$, $h = m_2$, $l = a_2 \cdot b_2$
(*r*, 1)	DD, $g = m_2$, $h = 1$, $l = a_2 \cdot b_2$
(*f* , 1)	ID, $g = 1$, $h = m_2$, $l = a_2 \cdot b_2$

Some constants in Table 2 are given as follows. Constants a_i's, b_i's and m_i's have similar meanings to those explained above.

$$a_1 = 1 + \frac{(R_{2,down} + R_3)^2}{(R_{2,down} + R_3 + R_b)^2} \cdot \frac{C_2 / C_1}{1 + |R_4 - R_2| / (R_{1,up} + R_1 + R_2)}$$

$$a_2 = 1 + \frac{(R_{2,up} + R_3)^2}{(R_{2,up} + R_3 + R_b)^2} \cdot \frac{C_2 / C_1}{1 + |R_4 - R_2| / (R_{1,down} + R_1 + R_2)}$$

$$b_1 = \frac{R_2 + (R_1 + R_{1,up}) // (R_{2,down} + R_3 + R_b)}{R_{1,up} + R_1 + R_2}$$

$$b_2 = \frac{R_2 + (R_1 + R_{1,down}) // (R_{2,up} + R_3 + R_b)}{R_{1,down} + R_1 + R_2}$$

$$m_1 = \frac{R_{2,down} + R_3 + R_b}{R_{1,up} + R_1 + R_{2,down} + R_3 + R_b}$$

$$m_2 = \frac{R_{1,down} + R_1}{R_{2,up} + R_1 + R_{1,down} + R_3 + R_b}$$

For other models in Figure 13, the same delay formula can be used to compute the increased or decreased delay, except for some input vectors causing logic faults at Out1. Similar results for Out2 can be easily derived from Figure 12and Table 2 by replacing Out2 with Out1, input pattern (In2, In1) with (In1, In2), $R_{2,VDD}$ with $R_{1,VDD}$ and $R_{2,VSS}$ with $R_{1,VSS}$. All the equations in (2.10) and (2.11) need to be recomputed by exchanging all the superscript 1 with 2, R_3 with R_1, R_4 with R_2 and C_1 with C_2 in the right hand side. For example, when input pattern (In2, In1) is (r, 0),

$$a_1 = 1 + \frac{(R_{2,down} + R_3)^2}{(R_{1,down} + R_1 + R_b)^2} \cdot \frac{C_1 / C_2}{1 + |R_4 - R_2| / (R_{2,up} + R_3 + R_4)}$$

When the input is a ramp signal, the modeling procedure is the same except the delay formula (2.10) needs to be modified in the way shown in Section 2.2. We use “Step input delay model” and “Ramp input delay model” to distinguish two different formulas, though in real applications only one formula is needed and the effect of the ramp input is considered as one coefficient.

Since driving resistances are dependent on input patterns for cells other than inverters/buffers, in the static fault simulation in which input patterns are unknown, resistances are chosen to maximize (minimize) the delay effect that gives optimistic (pessimistic) estimation.

In Table 2, there are two input patterns, (r, f) and (f, r), which may cause an increased delay or a decreased delay at Out1 and Out2 simultaneously. However, it can be easily derived from equation series (2.10) and (2.11) that the delay of Out1 with input (*r*, 0) is greater than the delay with input(*r*, *f*), and the delay with input (*f*, 1) is greater than the delay with input (*f*, *r*). Therefore, to maximize the delay of Out1, the best input patterns are (*r*, 0) and (*f*, 1). Whether the former is better or the latter is better depends on the parameters. To maximize the delay of Out2, the best input patterns are (0, *r*) and (1, *f*). Similarly, to minimize the delay at Out1, the best input patterns are (*r*, 1) and (*f*, 0) and to minimize the delay at Out2, (1, *r*) and (0, *f*). To maximize or minimize the delay at both output simultaneously, the best input patterns may be (*r*, *f*) and (*f*, *r*). Also, the delay formula we derived helps to choose input patterns at the previous stage. As shown in Figure 11, when the output of B_1 is rising, and the output of B_2 stays low, then there will be an increased delay. There are three input patterns, (1, *f*), (*f*, 1) and (*f*, *f*), which set the output of B_1 to a rising signal. However, (*f*, *f*) will produce less delay than the other two patterns since it decreases the pull-up resistance. Therefore, we should choose best patterns at previous stage to maximize or minimize the driving resistance.

For cases (*r*, 0) and (*f*, 1), simulation results of SPICE and our delay model on the example circuit in Figure 14 with the step input are shown in Figure 15 and Figure 16. In Figure 14, the input vector assignment for (*r*, 0) is shown. The technology is TSMC 180 nm 1.8 V. The PMOS size (width/length) is 540nm/180 nm, and the NMOS size is 270nm/180 nm. Pull-up/down resistances range from 1.5 KΩ to 3.2 KΩ, which are computed based on the linear region CMOS U-I curve [43]. Sink capacitances are 2.2fF and 3.1fF for the inverter and NAND gate, respectively. The logic interpretation voltage is 0.9V. There are 8 RC segments in each interconnect (same for the two lines), in which the total resistance is

17.4 Ω and the total capacitance is 4fF. The bridge locates in the middle of the two nets. BTRs $R_{1,Vss}$ for Out1 is greater than zero and $R_{1,Vdd}$ is less than zero. Out1 can only behave as the model in Figure 13(d) or (b) based on TABLE 1.

When $R_b > R_{1,Vss}$, in both Figure 15 and Figure 16, Out1 behaves as Figure 13(d) and an increased delay exists. Bridge faults falling in this range may be detected by delay tests with our fault model, but may not be detected by traditional logic tests with infinitely slow speed.

When $R_b \leq R_{1,Vss}$, Out1 behaves as Figure 13(b) in both figures but appears as an increased delay in Figure 14 and a logic fault in Figure 16. In this case, even though there is a delay fault for Out1 with some input patterns, bridge faults in this range can still be detected by traditional logic tests with other input patterns. Both delay and logic tests may detect these bridge faults.

For the case (r, 0), simulation results on the example circuit with the ramp input are shown in Figure 17. The slope of the ramp input is 0.01ns, which is almost half of the nominal delay of Out1. Results of SPICE, step input delay model and ramp input delay model are compared and we can see that the ramp input delay model considering the slope effect gives more accurate result when R_b increases.

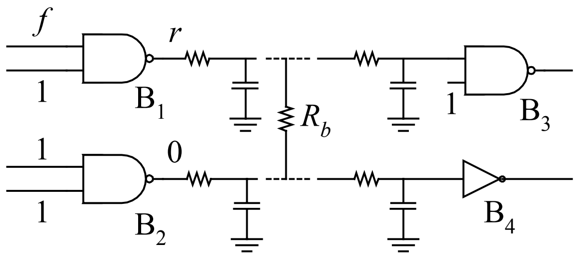


Figure 14. Example circuit for simulation.

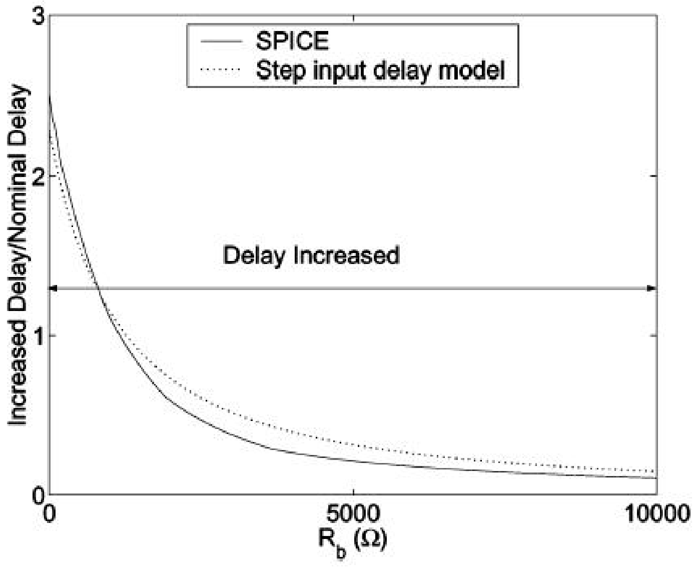


Figure 15. An example relationship between R_b and the increased delay at Out1 with the rising step input.

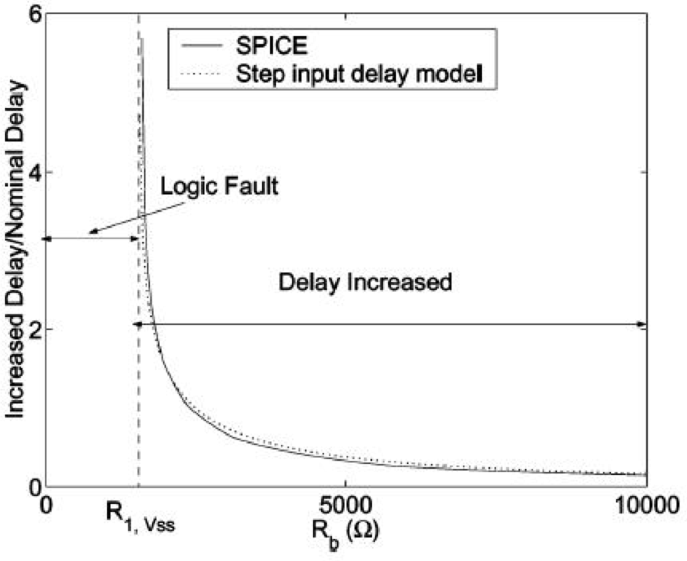


Figure 16. An example relationship between *R_b* and the increased delay at Out1 with the falling step input.

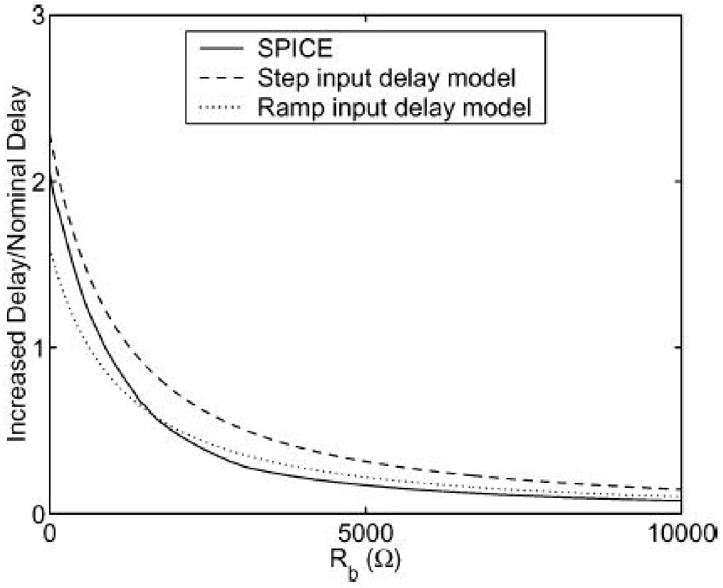


Figure 17. An example relationship between *R_b* and the increased delay at Out1 with the rising ramp input.

From all figures, our model shows a good match with SPICE simulation results. However, there are still some errors, which come from following sources: cell delay errors due to the linear resistance model and lumped capacitance model for driving gates, and interconnect errors due to simple RC interconnect and the approximation from Figure 8 to Figure 10.

In our experiment, when the bridge location is not in the middle of the two nets, the average delay varies by 0.1%. In general, if we do not know the exact location of a bridge, we will assume it locates in the middle of the two nets. It is a good approximation in practice. Some previous work also showed that the delay effect of a bridge fault has little relation with its location [42].

D. Application

The resistive bridge model d has been implemented in the CodSim delay fault simulator [44]. In the experiments, the bridge sites are assumed to be between two nets where large coupling capacitances exist. Such net pairs can be extracted using commercial capacitance extraction tools. The ISCAS85 benchmark circuits are used and the circuit layout is done with the Cadence Silicon Ensemble in TSMC 250 nm 3 V 3-metal technology. Commercial parasitic extraction tools are used to extract parasitics and compute net delays. The logic interpretation voltages are from 1.4 V to 1.5 V, and pull-up/down resistances of all gates are from 1 KΩ to 4 KΩ. For multi-input gates, pull-up/down resistances are computed assuming only one input changes at any time. The clock period is set to be 5% longer than the delay of the longest structural path.

Table 3 shows the simulation results for the ISCAS85 circuits, using 10,000 random vectors. Circuit c2670 is not included due to a parasitic extraction tool problem. The bridge resistance is approximately uniformly distributed from 0 Ω to 40 KΩ [45]. Columns 3 and 4 show the fault coverage using full-speed and half-speed tests, respectively. The fault coverage is computed by averaging the detected bridge resistance range over the potentially detectable resistance range for each bridge site. The half-speed tests can be considered fast logic tests and the full-speed tests can be considered the at-speed built-in self-tests (BIST), whose fault coverage is 1–5% higher than the half-speed tests. Using our model, for the first time it becomes possible to estimate the benefit from at-speed tests for resistive bridge faults. Our model is independent of the bridge resistance distribution, and therefore more accurate fault coverage can be computed if a more accurate distribution is known. Column 6 shows the simulation time and indicates that our bridge model is computational efficient.

TABLE 3. Delay fault simulation results for 10000 random vectors using our bridge model.

Circuit	Total Bridges	Resistive Bridge Model FC (%)		Time (s)
		Full –Speed	Half-Speed	
c432	821	88.1	84.4	1.4
c499	1,102	93.5	89.4	2.2
c880	1,412	90.0	86.2	2.4
c1355	2,488	88.6	84.2	7.0
c1908	4,007	92.0	91.9	5.1
c3540	8,919	87.0	86.7	17.9
c5315	12,168	94.3	94.0	18.6
c6288	14,17	91.6	91.4	22.5
c7522	12,156	87.2	86.6	25.7

III. FAULT MODELING OF DEVICE RESISTIVE SHORTS

The objective of the fault model is to transform the effect of resistive shorts in the circuit into a delay fault or a logic fault, and to compute the delay change due the short resistance R_b. We consider four types of gate oxide shorts (NMOS gate-to-drain short, NMOS gate-to-source short, PMOS gate-to-drain short and PMOS gate-to-source short), which are shown in Figure 18. This abstraction was first proposed in Hao and McCluskey [10]. Note that there is a diode in series with the short resistor for PMOS transistors, since the polysilicon gate is doped with N-type dopant for long channel transistors. We ignore the gate-to-channel short following [5], since its effect is little on the majority of transistors in logic gates.

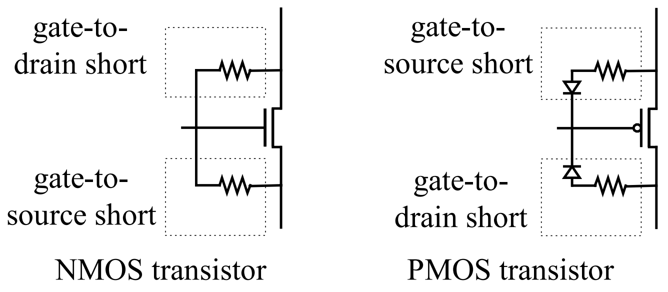


Figure 18. Gate-to-source and gate-to-drain shorts in NMOS and PMOS transistors.

To evaluate the short effect, we consider the CMOS circuit in Figure 19. There is a driving cell with output node *A* and a faulty cell with output node *B*. The potential defect is a gate-to-drain short inside an NMOS transistor of the faulty cell. Circuits for other defect types are similar, except that the faulty transistor may be PMOS or the short may be an NMOS gate-to-source short, or the faulty transistor may be directly connected to the gate output or the supply.

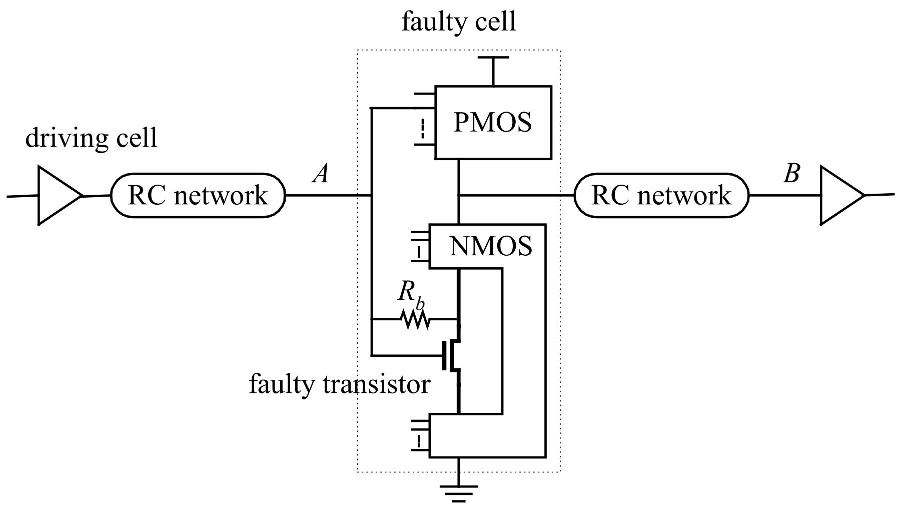


Figure 19. The circuit with an NMOS gate-to-drain short.

We assume that there is only one short in the faulty cell. We also assume shorts are only on input transistors, i.e. transistors driven by external signals. For shorts on other transistors, we can divide the faulty cell and make the faulty transistor driven by an external signal, then perform the similar analysis. For example, an AND gate can be partitioned into a NAND gate driving an inverter, in order to analyze the short at one of the inverter transistors.

When a signal transition at *A* causes a transition at *B*, we consider the sum of the driving cell delay and the faulty cell delay. On the other hand, when the signal at *A* is static, and a signal at other inputs of the faulty cell causes a transition at *B*, we only consider the faulty cell delay. Here delay is computed at 50% of V_{dd} on the signal waveform, and denoted as d_{Rb}. When there is no short, the driving cell delay is denoted as D_A, and the faulty cell delay is denoted as D_B. Both D_A and D_B are pre-computed by any commercial tools.

We present two methods to analyze the circuit behavior with gate-to-source/drain shorts. One is based on the analysis of resistive circuits, and the other is based on the analysis of the output waveforms properties. The former method uses the linear resistance of a transistor to simplify the transistor-level circuit into a resistive circuit, then derives output voltages and delay changes. The method is simple and fast, but the error is large for small short resistance due to the inaccurate circuit modeling. The latter method uses table-based current model for the U-I property of a transistor in the circuit, and achieves satisfied accuracy. The speed of the method is slow compared with the former method, but is still faster than the method using complex current models.

1. Linear Resistance Based Approach

The linear resistance based approach uses Shichman-Hodgs (SPICE Level-1) MOSFET model to compute the linear resistance of a transistor as $R_{linear} = 1/\beta/(V_{gs} - V_t)$, where β is the transistor gain factor, V_{gs} is the voltage between the gate and the source of the transistor, and V_t is the threshold voltage. R_{linear} is then used to compute the pull-up/down resistance in static analysis and the dynamic analysis

A. NMOS Gate-to-drain Short

i. Static Analysis

The circuit with static signals is simplified in Figure 20. The gate-to-drain short of the faulty NMOS transistor connects the driving cell and the faulty cell, and also breaks the pull-down path of the faulty cell into two parts. We use linear resistance R_{up1} and R_{down1} to represent the pull-up and pull-down resistance of the driving cell. The pull-up path of the faulty cell is represented by R_{up2}. The pull-down path of the faulty cell is divided into resistor R_{down2} and R_{down3} by R_b. The linear resistance of the faulty transistor is included in R_{down2}. Switches s_1, s_2, s_3 and s_4 indicate whether the pull-up path or the pull-down path is conducted. R_1 and R_2 are lumped resistance of the RC network of the interconnect.

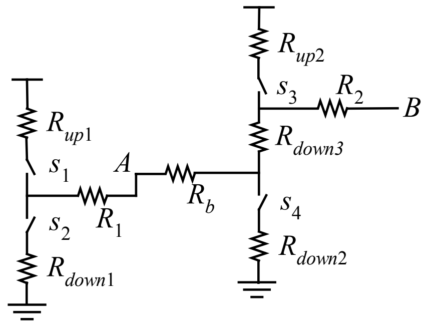


Figure 20. The circuit for static analysis.

In the case of A=0 and B=1, the pull-down path of the driving cell is conducted and s_1 is off, s_2 is on. The low voltage of A causes the pull-up path of the faulty cell conducted, then s_3 is on and s_4 is off. The voltage of A can be derived as:

$$V_{A,low} = \frac{R_1 + R_{down1}}{R_{down1} + R_1 + R_b + R_{down3} + R_{up2}} V_{dd} . \tag{3.1}$$

Define Short Threshold Resistance (STR) $R_{A,low}$ for the logic value of A, such that when $R_b < R_{A,\mathrm{low}}$, $V_{A,low}$ is greater than the logic low threshold voltage V_{LL}. Then according to (3.1),

$$R_{A,low} = \frac{(V_{dd} - V_{LL})(R_1 + R_{down1})}{V_{LL}} - R_{down3} - R_{up2} . \tag{3.2}$$

Therefore when $R_b < R_{A,low}$, $V_{A,low} > V_{LL}$, there is a logic fault at A. When $R_b > R_{A,low}$, there will be no logic fault at A, but there might be a delay fault, which will be analyzed later. Similarly, we can derive the voltage of B, $V_{B,high}$, which is logic high in a fault-free circuit, and then derive the corresponding STR $R_{B,high}$ for B. When a pull-up path in the driving cell is formed, similar voltage expressions $V_{A,high}$ at A, $V_{B,low}$ at B and corresponding BTRs can be derived.

ii. Dynamic Analysis

In dynamic analysis, there are four types of input signal transitions: high, low, rising and falling. Transitions may occur in both the driving cell and the faulty cell, or occur in the faulty cell only. We analyze circuit behaviors through two typical transition patterns. Analysis for other patterns can be derived similarly. In each pattern, R_b is assumed to be large enough such that no logic fault exists. To simplify the computation, we use C_1 and C_2 to represent the load capacitance of the driving cell and the faulty cell respectively. Both C_1 and C_2 include the lumped interconnect parasitic capacitance and the input gate capacitance.

a. Pattern 1: Falling at A, Rising at B

The circuit in dynamic analysis is illustrated in Figure 21. When the signal at A is falling and the signal at B is rising, the initially high voltage of A decreases to $V_{A,low}$ and the initial low voltage of B increases to $V_{B,high}$. Then the faulty NMOS transistor is cut off due to the low voltage at A, and the pull-up path of the faulty cell is conducted. If there is no R_b, C_1 is discharged only through R_{down1}, and C_2 is charged only through R_{up2}. However, in the presence of R_b, a path through R_b connects A and B. Then C_1 is partially discharged to C_2 because the initial voltage of A is higher than that of B. At the same time, C_1 is charged through the pull-up path of the faulty cell and R_b.

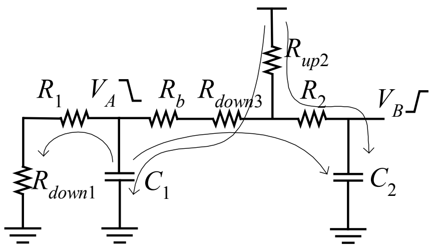


Figure 21. A current path connects *A* and *B* through *R*$_b$.

The delay $_{computation}$ works as follows. We first derive the formulas of driving cell delay $d_{A,Rb}$ and faulty cell delay $d_{B,Rb}$ according to the linear circuit analysis. Then considering both delays are functions of R_b, we compute the ratio between the delay with a fixed value of R_b and the delay with $R_b = \infty$, and approximate d_{Rb} as

$$d_{R_b} = \frac{d_{A,R_b}}{d_{A,R_b=\infty}} D_A + \frac{d_{B,R_b}}{d_{B,R_b=\infty}} D_B . \tag{3.3}$$

The value of $d_{A,Rb}$ in the formula is computed in the following procedure.

In order to transform the circuit in Figure 21 to a first-order linear system and derive the delay formula directly, we introduce an effective resistance $R_{A,eff}$ to replace the discharging path R_b-R_{down3}-R_2-C_2. The value of $R_{A,eff}$ is computed by equalizing the average current of the two circuits in Figure 22, during time interval T_d, witch is the time for the voltage of C_1 in the circuit of Figure 22(a), dropping from its initial voltage V_0 to 50% of the total swing. The equivalent current method is also used to calculate effective capacitance in [46]. In this work we use it to compute $R_{A,eff}$.

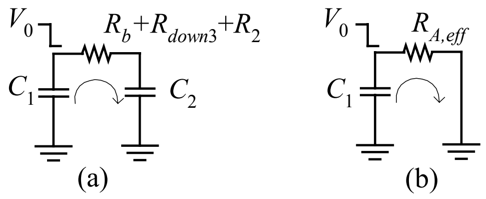


Figure 22. $R_{A,eff}$ is computed by equalizing the average current in the two circuits.

Then we get

$$R_{A,eff} = \frac{C_2}{C_1 + C_2} \cdot \frac{\ln(2)}{\ln(\frac{2(C_1 + C_2)}{2C_1 + C_2})} \cdot (R_b + R_{down3} + R_2) . \tag{3.4}$$

Since the circuit becomes a first order linear system, the driving cell delay can be derived directly as

$$d_{A,R_b} = d_{f1} + \ln(\frac{V_{A,low} - V_{A,high}}{V_{A,low} - 0.5V_{dd}}) \cdot R_A \cdot C_1 , \tag{3.5}$$

where d_{f1} is the intrinsic falling delay of the driving cell, R_A=(R_{down1}+R_1)//(R_b+R_{down3}+R_{up2})//$R_{A,eff}$. Note that when R_b is infinity, $d_{A,Rb=\infty}$ is the sum of intrinsic gate delay and gate load delay under the lumped C delay model:

$$d_{A,R_b=\infty} = d_{f1} + \ln 2 \cdot (R_{down1} + R_1) \cdot C_1 \ . \tag{3.6}$$

The computation of $d_{B,Rb}$ is similar. For the rising signal transition at B, path C_1-R_b-R_{down3}-R_2 is a charging path. We insert a resistance $R_{B,eff}$ between A and B to equalize the charging effect of the path. The value of $R_{B,eff}$ is computed similarly. Here the value of T_d is the time for the voltage of C_2 in the circuit of Figure 23(a), rising to 50% of the total swing.

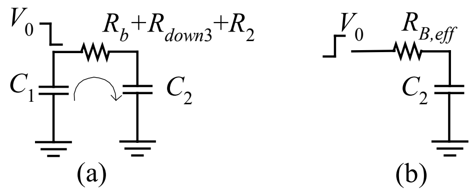


Figure 23. $R_{B,eff}$ is computed by equalizing the average current in the two circuits.

Then we get

$$R_{B,eff} = \frac{C_1}{C_1 + C_2} \cdot \frac{\ln(2)}{\ln(\frac{2(C_1 + C_2)}{C_1 + 2C_2})} \cdot (R_b + R_{down3} + R_2)\,. \tag{3.7}$$

The faulty cell delay is derived as

$$d_{B,R_b} = \alpha \cdot d_{r2} + \ln(\frac{V_{B,down} - V_{B,up}}{V_{B,down} - 0.5V_{dd}}) \cdot R_B \cdot C_2\,, \tag{3.8}$$

where d_{r2} is the intrinsic rising delay of the faulty cell, $\alpha=(V_{dd} - V_{B,low})/V_{dd}$ is introduced to scale d_{r2} because the initial voltage for the rising transition at B is not zero, and $R_B=(R_{down1}+R_1+R_{B,eff}+R_b+R_{down3})//R_{up2}+R_2$.

The approximation is verified on an inverter driving a NAND gate. The resistive gate-to-drain short of an NMOS transistor connects the inverter and the NAND gate. We use Cadence Spectre to run SPICE simulation under TSMC 180 nm 1.8V technology. Results of delay vs. R_b computed by our model and by SPICE simulation are shown in Figure 24. We assume V_{LL} = 0.9V, then compute the Bridge Threshold Resistance (STR) $R_{A,down}$ = 1284Ω. When $R_b < R_{A,low}$, a logic fault occurs on A. Otherwise, a decreasing delay change occurs, which means the delay in the presence of R_b is less than the delay in a short-free circuit. Our model provides a reasonably accurate approximation of the delay change. In the figure, the two curves of SPICE simulation and our approximation are pretty close for larger R_b. The inaccuracy for smaller R_b is due to non-linear properties of MOS transistors. Similar results are also found in other gates, as long as the circuit is CMOS circuit.

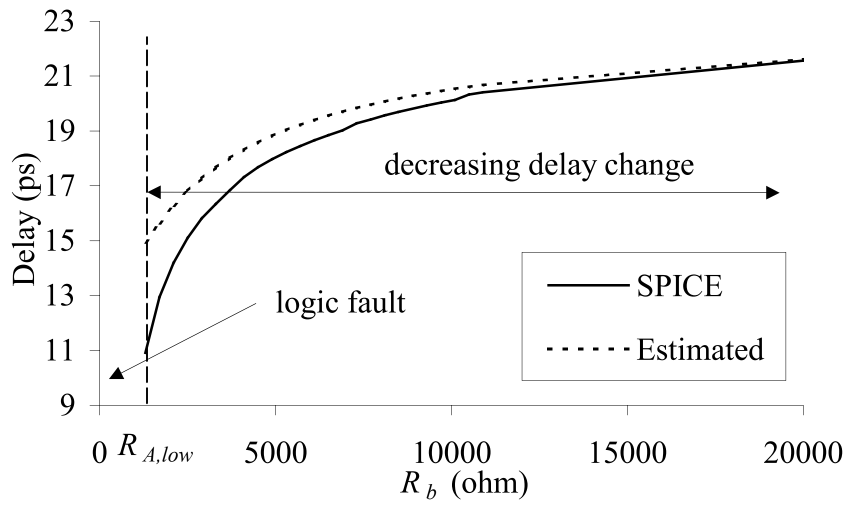


Figure 24. Decreasing delay estimated and computed by SPICE vs. R_b.

b. Pattern 2: Static at A, Falling at B

Consider an NMOS transistor T_g, which is in series with the faulty NMOS transistor T_f. Assume there is a rising signal driving T_g, resulting in a falling transition at B, while the signal at A is static.

During the transition, a pull-down path through T_f and T_g is formed. The circuit in transition is shown in Figure 25, where R_{down3} represents the linear resistance of T_f and the NMOS block connecting the output of the faulty cell, and R_{down2} represents the linear resistance of the NMOS block connecting T_f and GND.

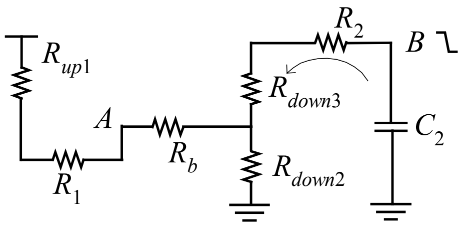


Figure 25. C_2 is discharging through the pull-down path in the faulty cell.

The final voltage at B is dependent on the value of R_b, and can be derived as

$$V_{B,low} = \frac{R_{down2}}{R_{up1} + R_1 + R_b + R_{down2}} V_{dd} . \tag{3.9}$$

Then the faulty cell delay is computed as

$$d_{B,R_b} = d_{f2} + \ln\left(\frac{V_{dd} - V_{B,low}}{0.5V_{dd} - V_{B,low}}\right) \cdot (R_{down2} // (R_{up1} + R_1 + R_b) + R_{down3} + R_2) \cdot C_2 , \tag{3.10}$$

where d_{f2} is the intrinsic rising delay of the faulty cell without the short.

Similarly, we approximate d_{Rb} to be

$$d_{R_b} = \frac{d_{B,R_b}}{d_{B,R_b=\infty}} \cdot D_B . \tag{3.11}$$

Experimental results on the same circuit are shown in Figure 26. The BTR for $V_{B,low}$ is $R_{B,low}$ = 1798Ω. The figure shows that when $R_b < R_{B,low}$, there is a logic fault at A and when $R_b \geq R_{B,low}$, there is an increasing delay change in the faulty cell in the presence of R_b.

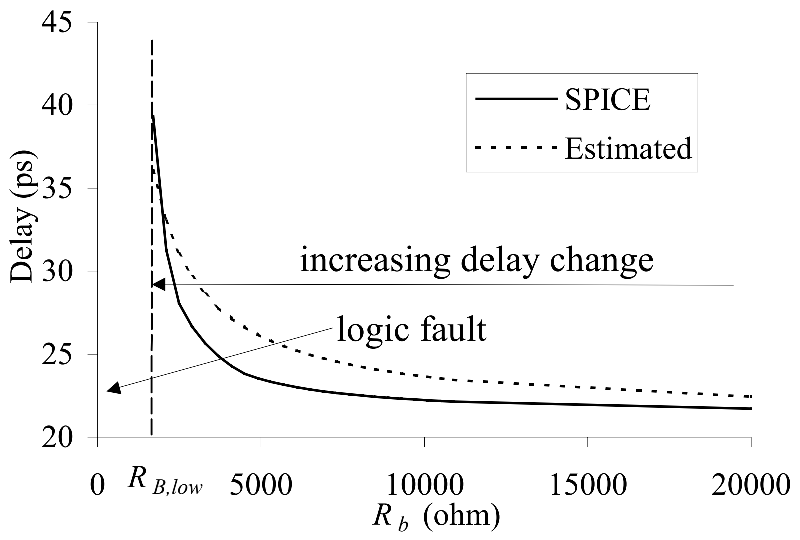


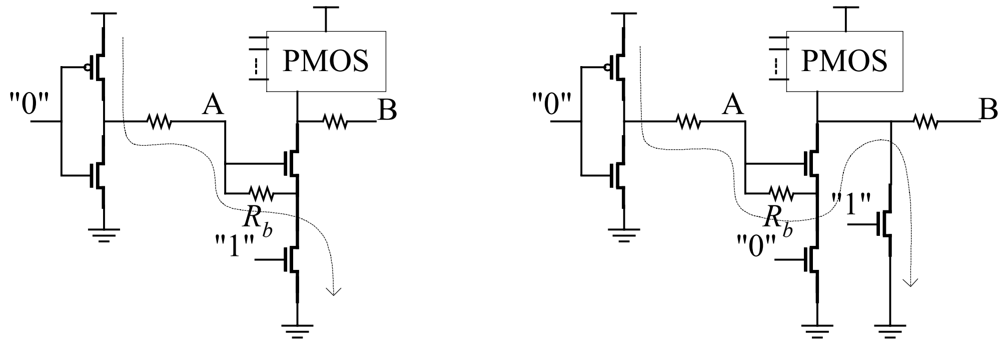
Figure 26. Increasing delay estimated and computed by SPICE vs. *R_b*.

B. NMOS Gate-to-source Short

An NMOS gate-to-source short may affect circuit behaviors in two ways.

1) The short acts as a resistor connecting *A* to GND. For example, given the signals shown in Figure 27(a), the current flows through R_b to GND, while the faulty NMOS transistor is cut off. The resistor can be represented by R_b+R_{down}, where R_{down} represents the linear resistance of the NMOS block between the faulty transistor and GND.

2) The short connects *A* and *B* through the faulty NMOS transistor, and currents flow through *A* to the faulty cell output, then through some NMOS transistors in parallel with the faulty transistor to GND, as the example shown in Figure 27(b). Therefore the voltage of *B* is pulled up incorrectly, and may cause a logic fault or a delay fault. The corresponding STR values and the delay change can be computed similarly.



(a) Currents flow through R_b to GND (b) Currents flow through R_b and the faulty transistor

Figure 27. The NMOS gate-to-source short may connect *A* to GND or connect to *B* under different the input signals.

C. PMOS Gate-to-source/drain Short

The gate-to-source short in a PMOS transistor can be modeled as a bridge resistor in series with a diode between interconnect and VDD. The diode is always forward-biased and is represented by a certain voltage drop across the bridge. The behavior of the diode can be analyzed similarly to NMOS gate-to-source shorts, except currents can never flow through *A* to *B*. PMOS gate-to-drain shorts can be analyzed in the similar way with NMOS gate-to-drain shorts.

2. Current Table Based Approach

A. Static Analysis

In static analysis, input and output signal voltages of the circuit keep static. In the short-free circuit, we assume that the output voltages of the driving cell V_a and the faulty cell V_b are VSS or VDD. In the presence of the short, those voltages may not exactly be VSS or VDD, but dependent on the short resistance and the input signals. It is essential to calculate static voltages fast and accurately. The voltage indicates if a logic fault exists, i.e., if it is out of the range for a correct digital signal. Moreover, it is one important component in delay approximation, which will be discussed in dynamic analysis.

i. Static Voltage Approximation

Consider an NMOS gate-source short in a 2-input NAND gate in Figure 28. We use an inverter as the driving cell. Transistors in the circuit are labeled by M_1, M_2, …, M_6. The voltage of the inverter input is V_g, node *A* voltage is V_a, node *B* voltage is V_b, and the voltage at the other terminal of the short is V_c. Signals on transistor M_4 and M_6 are non-controlling signals, and in static VDD. The task of the static analysis is to approximate V_a and V_b, given short resistance R_b and static input voltage V_g.

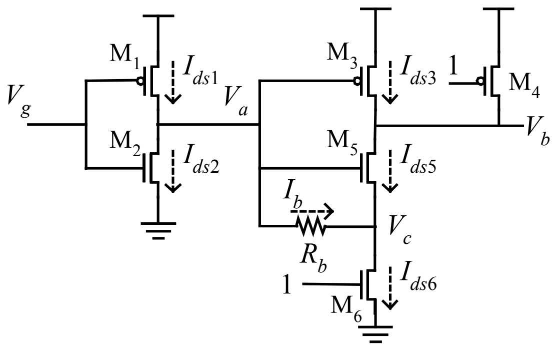


Figure 28. The circuit with an NMOS gate-to-source short in static analysis.

Assume that the current through each transistor is I_{ds1}, I_{ds2}, I_{ds3}, I_{ds5}, and I_{ds6}, and the current through R_b is I_b, as shown in the figure. Note that we assume there is no current through transistor M_4 because it is in cut-off state. Then we have:

$$I_{ds1} = I_{ds2} + I_b, \tag{3.12}$$

$$V_a - V_c = R_b \cdot I_b, \quad (3.13)$$

$$I_b + I_{ds5} = I_{ds6}, \quad (3.14)$$

$$I_{ds3} = I_{ds5}. \quad (3.15)$$

On the other hand, we can express the drain-to-source current of a MOS transistor I_{ds} as a function of voltage V_{gs} between the transistor gate and source, and voltage V_{ds} between the transistor drain and source, i.e. $I_{ds} = F(V_{gs}, V_{ds})$. Therefore we have:

$$I_{ds1} = F_1(\mathrm{VDD} - V_g, \mathrm{VDD} - V_a), \quad (3.16)$$

$$I_{ds2} = F_2(V_g, V_a), \quad (3.17)$$

$$I_{ds3} = F_3(\mathrm{VDD} - V_a, \mathrm{VDD} - \mathrm{V_b}), \quad (3.18)$$

$$I_{ds5} = F_5(V_a - V_c, V_b - V_c), \quad (3.19)$$

$$I_{ds6} = F_6(\mathrm{VDD}, V_c). \quad (3.20)$$

The value of V_a and V_b can be derived based on the above equations Eq. (3.12) to Eq. (20). The speed and accuracy depend on the current model in use. For example, the Shichman-Hodgs (SPICE Level-1) MOSFET model provide simple piecewise equations, which express I_{ds} as a polynomial of V_{gs} and V_{ds} under different conditions. Then we can use non-linear regression approaches to derive V_a and V_b. However, experiments show that using Shichman-Hodgs model introduces more than 10% errors. More accurate models, such as the Alpha-Power Law MOSFET model [47], are too complicated to be applied. Instead, in this paper, we use a two-dimension table to describe $I_{ds} = F(V_{gs}, V_{ds})$. Each entry of the table is a value of I_{ds} under some certain V_{gs} and V_{ds}, which is pre-computed by SPICE simulation. Accurate current models also can be used to setup the table. We only need to compute the table for the smallest size of PMOS and NMOS transistors respectively.

For other size of transistors, the table is scaled proportionally with the scaled transistor size.

The table-based approximation on gate-source shorts works as follows. First we initialize a vector of V_a, which is sampled from 0 to VDD. Then we derive a vector of I_b corresponding to V_a based on Eg. (3.12), and then derive a vector of V_c according to Eq. (3.13). Corresponding values of I_{ds6} are computed based on Eq. (3.20). Then we use Eq. (3.14) to obtain values of I_{ds5}, and derive V_b using Eq. (3.19). Therefore, values of I_{ds3} on transistor M_3 are computed according to Eq. (3.18). Since we have $I_{ds3} = I_{ds5}$ in Eq. (3.15), the data curve of I_{ds3} and I_{ds5} on sampled V_a must have a cross point, which corresponds to the solution of V_a. Assume the number of sample points of V_a is n, then the complexity of the approximation is $O(n^2)$, since for each sample point we need take $O(n)$ complexity of interpolation in the current table.

As an example, we show the approximated V_a and V_b curves over a series of R_b, and compare them with SPICE simulation results in Figure 29. Experiments was performed on the circuit with PMOS transistor of size W=0.7um and L=0.2um, and NMOS transistor of size W=0.7um and L=0.2um under TSMC 180nm technology. The input signal V_g is 0. The current table size is 19x19, and the size of the initial V_a vector is 19. Results show that the approximated voltage is well matched with that of SPICE simulation.

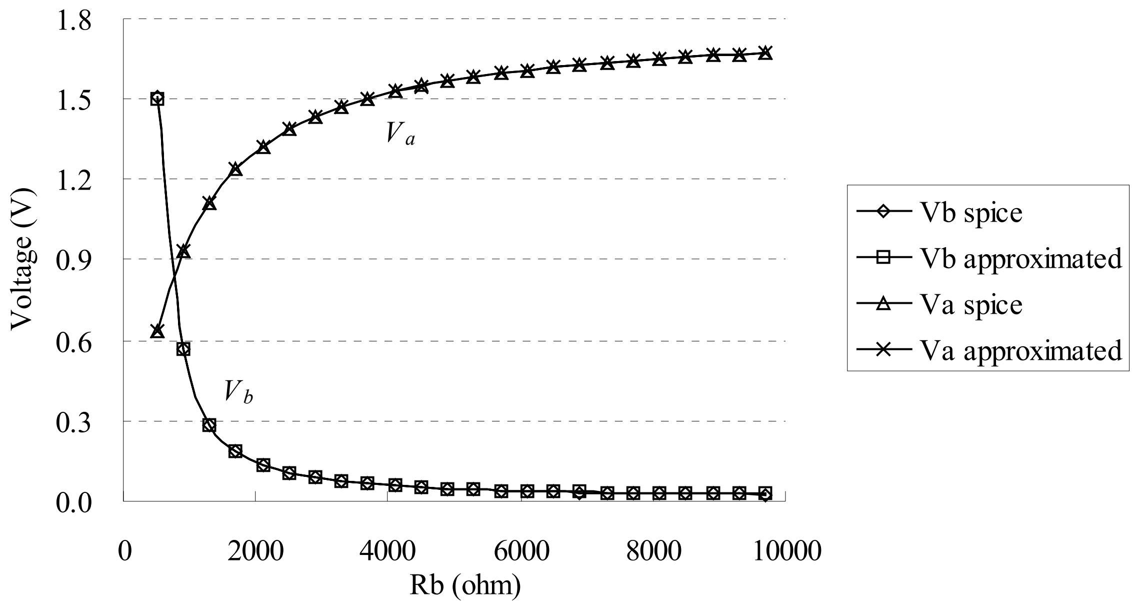


Figure 29. Static voltage approximation compared with SPICE simulation.

ii. Short Threshold Resistance

Assume the logic low threshold voltage is V_{LL} and the logic high threshold voltage is V_{HH}. Then define the threshold short resistance $R_{A,low}$ for the logic value of A, such that when $R_b < R_{A,\mathrm{low}}$, $V_{A,low}$ is greater than V_{LL}. Therefore, when $R_b < R_{A,\mathrm{low}}$, we consider a logic fault on A, else we consider a delay fault. The relationship is simply shown in Figure 30.

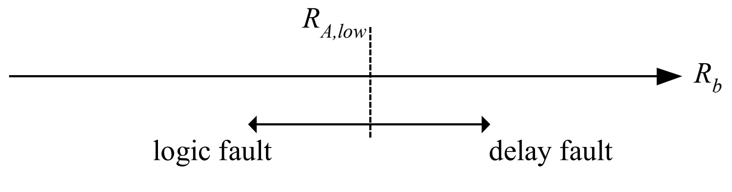


Figure 30. Threshold short resistance $R_{A,low}$.

The threshold short resistance can be obtained using the proposed voltage approximation approach. That is, once the curve of V_a is computed, $R_{A,low}$ can be found in corresponding to V_{LL}.

Similarly, we can compute threshold resistance $R_{A,high}$ corresponding to V_{HH}, and the threshold resistance for signal at B. These resistance values are to be used to indicate if a logic fault happens.

B. Dynamic Analysis

In dynamic analysis, we consider circuit behaviors during signal transitions, and explore the relationship between delay changes and the short resistance. Transitions may occur in both the driving cell and the faulty cell, or occur in the faulty cell only, depending on input signal patterns. In this paper, we focus on the transition on both cells, since it is more complex due to the interaction of the two cells. Similar analysis and results can be simply applied to the transition that only occurs on the faulty cell.

We first analyze the transition procedure in the presence of R_b, and show how delay changes under different input signals. Then we present a new delay model, and use it in our approximation approach. Finally, we compare our approximation result and SPICE simulation. In dynamic analysis, R_b is assumed to be bigger than the threshold short resistance such that no logic fault exists. And to simplify the computation, we use C_1 and C_2 to represent the load capacitance of the driving cell and the faulty cell respectively. Both C_1 and C_2 include the lumped interconnect parasitic capacitance and the input gate capacitance. Because the analysis of NMOS gate oxide shorts can be applied dually to the PMOS gate oxide shorts, in this section we only study the NMOS gate oxide shorts. For

PMOS gate oxide short, the diode in the short model allows current to flow in only one direction. If it is conducted, then it is treated as a certain voltage drop on the current path. If not, then the current path is cut-off.

i. Transition Procedure

The presence of the short resistor provides an extra current channel for the load capacitor to be charged and discharged. Consider the circuit with an NMOS gate-to-source short in Figure 31, and assume there is one rising signal for 0 to VDD on the input of the driving cell. The circuit in transition is shown in Figure 6.

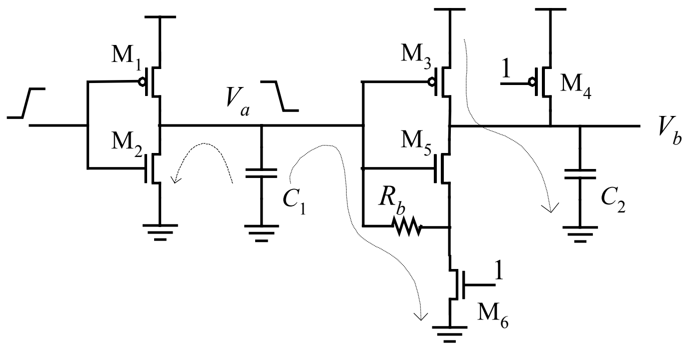


Figure 31. The circuit with an NMOS gate-to-source short in transition.

Initially, transistor M_1 is conducted and M_2 is cut-off due to the low voltage on the input of the inverter. Transistor M_6, which is connected to node A by R_b, is conducted due to the high voltage on its gate. Therefore, a current path (M_1-R_b-M_6) is formed. Then V_a is pulled-down from VDD. Accordingly, V_b may not keep 0 because M_3 can be in non-saturation state for $V_a > V_{pt}$, where V_{pt} is the threshold voltage of the PMOS transistor M_3.

When the transition begins, transistor M_2 is becoming conducted. Capacitor C_1 is discharged through M_2, and through R_b and M_6 in a parallel path. Here the presence of R_b provides an extra discharging path for C_1. For smaller Rb, the timing constant of the extra discharging path is smaller, which means C_1 will be discharged even faster. In that scenario, the driving cell delay is decreasing. In the end of the transition, C_1 is totally discharged and $V_a = 0$. On the faulty cell, capacitor C_2 is charged through M_3. Although there is no extra current path for C_2 to be charged faster, the initially lowered voltage on the input of the faulty cell accelerate the process. Finally V_b is pulled-up to VDD.

In Figure 32, we show the waveforms of V_a under three different values of R_b in SPICE simulation, where VDD=1.8v and the input signal is a ramp rising with edge rate 30ps. The initial voltage of V_a becomes smaller with smaller Rb, and the delay on the 50% of VDD decreases as well.

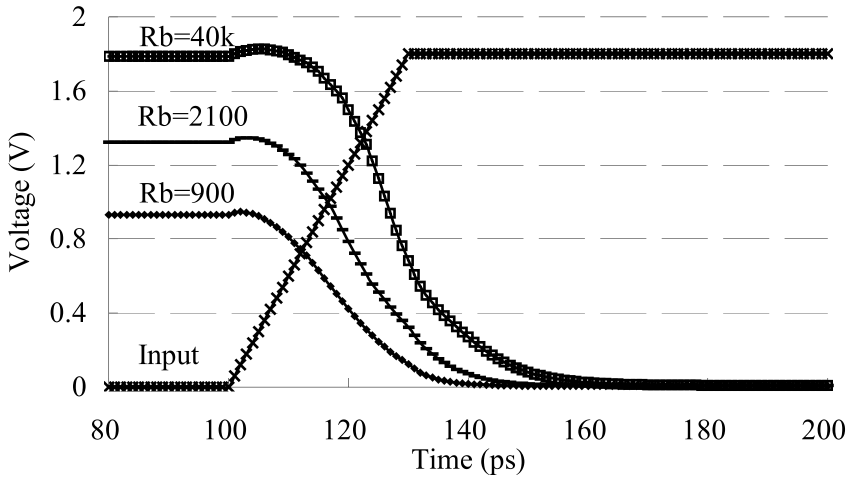


Figure 32. Delay decreases with smaller R_b, under a rising input signal.

Now consider the transition triggered by a falling signal at the driving cell input.

Initially, V_a=0 and V_b=VDD. Then capacitor C_1 is charged through transistor M_1 and V_a increases. On the other hand, for capacitor C_1, the discharging path through R_b and transistor M_6 exist during both the static and transition period. That slows down the charging of C_1, and finally lowers down V_a from VDD. Accordingly, because of the slow increase of V_a, the discharging process of C_2 becomes slow as well.

In Figure 33, we show the waveforms of V_a under the falling input signals. The delay increases with smaller R_b.

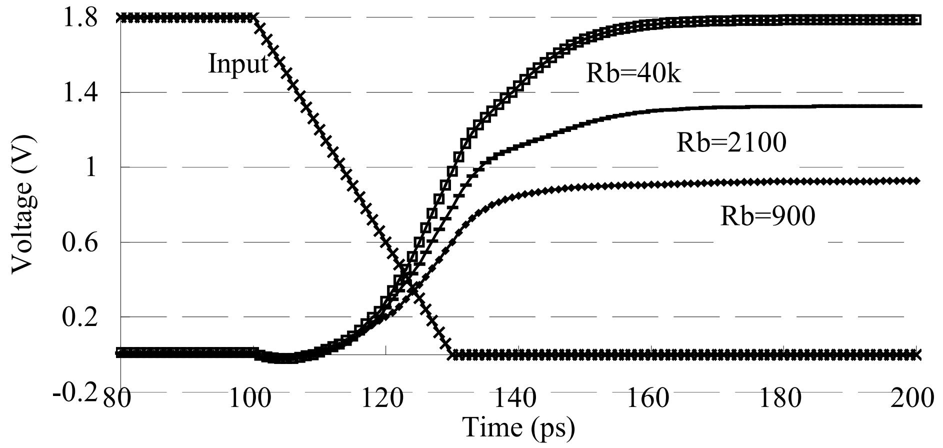


Figure 33. Delay increases with smaller *R_b*, under a falling input signal.

In conclusion, the short resistance R_b affects delay changes in two ways. One is that R_b provides an additional current path in the pull-up/down path. Therefore the load capacitor is charged or discharged faster when the path helps the process, and slower when the path deters it. The other is that the initial voltage may change. That makes the transistor convert from the cut-off state to the non-saturation state faster.

We conclude the relationship between delay changes and the output waveform shape in Table 4. The initial and the final voltage of the signal are V_0 and V_1 respectively. In the table, “off VDD” means that the voltage is lowered down from VDD due to the short, and “off 0” means the voltage is pulled-up from VSS. The offset initial voltage speedups the transition, while the offset final voltage indicates that an extra current path slows it down. When both V_0 and V_1 are offset from VSS or VDD, in general cases delay decreases. However, for R_b close to the threshold short resistance, delay may be greater than the delay in the fault-free circuit. Therefore we put “undetermined” in that waveform shape.

TABLE 4. Relationship between delay changes and output signal waveform.

Type	V0	V1	Delay change
Rising	VSS	Off VDD	increasing
Falling	Off VDD	VSS	decreasing
Rising	Off VSS	VDD	decreasing
Falling	VDD	Off VSS	increasing
Rising	Off VSS	Off VDD	undetermined
Falling	Off VDD	Off VSS	undetermined

ii. Delay Approximation

The delay approximation approach is based on a new delay model in the following:

$$D = D_{sleep} + D_{intrinsic} + D_{RC}, \tag{3.21}$$

where D is the cell delay, D_{sleep} is the time for the transistor to convert from the cut-off status to the non-saturation status, $D_{intrinsic}$ is the delay with zero output load, and D_{RC} is the extra delay caused by the output load.

a. D_{sleep} Approximation

D_{sleep} is determined by the input signal waveforms. It is used to account for the time that a transistor starts to be conducted, and is computed as the time for $V_{gs} < V_t$, where V_t is the threshold voltage of the transistor to be conducted. For a slow input signal, D_{sleep} is more significant. For fast input signals, D_{sleep} is small, and especially, $D_{sleep} = 0$ under a step input.

b. $D_{intrinsic}$ Approximation

$D_{intrinsic}$ is computed from the time that the transistor starts to be conducted, to the time that the output waveform across the 50% of VDD, for the cell with zero loads. Note it is different from the traditional delay model, in that the starting point is not the time for the input waveform to across the 50% of VDD.

Assume R_b is infinity, i.e., in a short-free circuit, we can compute the intrinsic delay using any commercial tools, and name it as D_{int}. Then considering the presence of the short, we treat the transistor in transition as a first-order linear system under a step stimulate, and write the output voltage as:

$$V(t) = V_1 + (V_0 - V_1)e^{-t/\tau}, \tag{3.22}$$

where V_0 is the initial voltage and V_1 is the final voltage, τ is the time constant in the system. Then the delay for the waveform to across the 50% of VDD can be expressed as:

$$d_{50\%Vdd} = \ln\left(\frac{V_0 - V_1}{50\%Vdd - V_1}\right) \cdot \tau . \tag{3.23}$$

Note that in the short-free circuit, $d_{50\%VDD} = D_{int} = \ln 2\ \tau$. Therefore, we approximate the intrinsic delay with the presence of R_b as a scaled D_{int}:

$$D_{\text{int}rinsic} = \ln(\frac{V_0 - V_1}{50\%\text{VDD} - V_1}) / \ln 2 \cdot D_{\text{int}} \,. \tag{3.24}$$

Note that in the above formula, we assume that the time constant τ does not change with R_b. That is reasonable since the transistor is in non-saturation status during most of the transition time, and is not affected by the change of V_0 and V_1.

c. D_{RC} Approximation

D_{RC} is used to account for the delay due to the output load C_l, and can be expressed as

$$D_{RC} = \ln(\frac{V_0 - V_1}{50\%\text{VDD} - V_1}) \cdot R_d C_l ,. \tag{3.25}$$

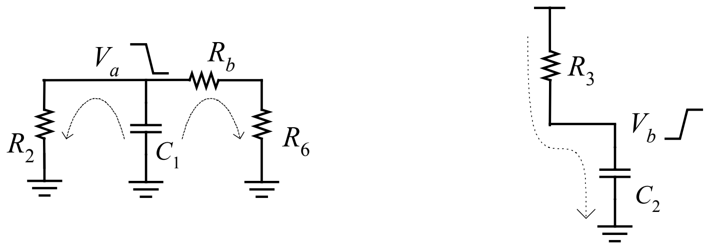
where R_d is the equivalent driving resistance of the pull-up/down path.

In the short-free circuit, we name it as D_{RC0} =ln2R_dC_l, and can be computed by subtracting D_{sleep} and $D_{\text{intrinsic}}$ from the delay with output load. With the known of output load C_l, we can compute the driving resistance $R_d = D_{RC0}/\ln2/C_l$. This value accounts for the transistor driving ability, and keeps consistent in the faulty circuit.

With the presence of R_b, voltage V_0 and V_1 change, as well as R_d. The value of R_d depends on the type of the gate-oxide short and input signals. In the following, we derive R_d. in two typical cases. For other cases, the value of R_d can be derived similarly. We assume the driving resistance of the transistor M_i is R_i, which is pre-computed in the fault-free circuit.

Case 1. NMOS gate-to-source short under rising input

Consider the circuit in Figure 6, the rising input of the driving cell causes a falling transition at *A* and a rising transition at *B*. The simplified circuit for R_d. computation is shown in Figure 34.



(a) Rising transition at A (b) Falling tansition at B

Figure 34. Simplified circuit for RC delay approximation.

Then for the rising transition at A, the equivalent driving resistance $R_{d,A} = R_2//(R_b + R_6)$, where "//" means two resistors in parallel, and for the falling transition at B, the equivalent driving resistance $R_{d,B} = R_3$.

Such approximation also can be applied to NMOS gate-to-source short under falling input, and PMOS gate-to-source short under rising and falling input, except that the conducted transistor is different.

Case 2. NMOS gate-to-drain short under rising input

Consider the NMOS gate-to-drain short of an inverter in the circuit of Figure 35.

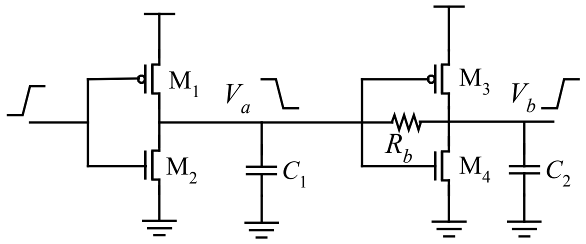


Figure 35. Circuit with an NMOS gate-to-drain short under a rising input.

Initially, M_1 is conducted and M_2 is cut-off. Dependent on the initial voltage of V_a, M_3 and M_4 may be in the cut-off or the conducted status. Under the rising input, V_a is falling and V_b is rising. Then the faulty NMOS transistor becomes cut-off due to the low V_a, and M_3 becomes conducted. If there is no R_b, C_1 is discharged only through M_2, and C_2 is charged only through M_3. However, in the presence of R_b, a path through R_b connects A and B. Then C_1 is partially discharged to C_2 because the initial voltage of A is higher than that of B. At the same time, C_1 is charged through the pull-up path of the faulty cell and R_b. By replacing transistors with the driving resistors, we show the circuit in transition in Figure 36.

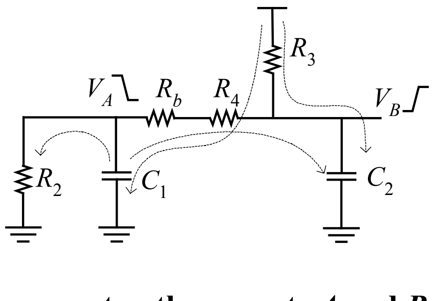


Figure 36. A current path connects *A* and *B* through R_b.

The value of R_d for V_a falling and for V_b rising is computed similar with the analysis in linear resistance based method in Eq. (3.4) and Eg. (3.7) respectively.

Then we get

$$R_{A,eff} = \frac{C_2}{C_1 + C_2} \cdot \frac{\ln(2)}{\ln(\frac{2(C_1 + C_2)}{2C_1 + C_2})} \cdot (R_b + R_4). \tag{3.26}$$

Then we get $R_{d,A}$=R_2//(R_b+R_4)//$R_{A,eff}$, which is the parallel resistance of R_2, R_b+R_4, and $R_{A,eff}$.

The computation of $R_{d,B}$ is similar.

$$R_{B,eff} = \frac{C_1}{C_1 + C_2} \cdot \frac{\ln(2)}{\ln(\frac{2(C_1 + C_2)}{C_1 + 2C_2})} \cdot (R_b + R_4). \tag{3.27}$$

The driving resistance of the faulty cell is $R_{d,B}$=(R_2+ $R_{B,eff}$+R_b+R_4)//R_3.

d. Approximation Procedure and Experimental Results

We summarize the procedure of the delay approximation approach as follows.

Step 1. Compute cell delays in the fault-free circuit, and derive the driving resistance of the transistor, and intrinsic delay.

Step 2. Using static analysis, compute the initial voltage V_{a0} and V_{b0} and the final voltage V_{a1} and V_{b1}.

Step 3. Approximate the delay at A, using the delay model in formula (10).

Step 4. Use V_a as the input signal of the driving cell, determine D_{sleep} for the delay approximation of the faulty cell. This step is similar with step 3, except the delay is computed at the time of $V_{gs} = V_t$.

Step 5. Approximate the delay at B, using the delay model in formula (10).

To verify our delay approximation approach, we estimate the delay of a faulty circuit consisting of two inverters in series. TSMC 180nm technology is used. PMOS transistor size is Wp=1.4um and Lp=0.2um. NMOS transistor size is Wn=0.7um and Ln=0.2um. Load capacitance C_1=C_2=1fF. We show the approximated delay on the output of the circuit and SPICE simulation result in the following figures. Each figure represents a different case of the transition and is summarized in Table 5.

TABLE 5. Approximated delay and SPICE simulation results in Figure 37 – Figure 40.

Short type	Input	Normal delay	Worst delay	Threshold short resistance	Delay change
NMOS gate-to-source	rising	19.7ps	0.6ps	RA,high=853Ω	decreasing
NMOS gate-to-source	falling	20.9ps	48.0ps	RA,high=853Ω	increasing
NMOS gate-to-drain	rising	19.7ps	12.3ps	RB,high=775Ω	decreasing
NMOS gate-to-drain	falling	20.9ps	5.8ps	RB,high=775Ω	decreasing

In Figure 37, we consider the delay of the circuit with an NMOS gate-to-source short and a rising input. The threshold short resistance on A is $R_{A,high}$=853Ω. When R_b< $R_{A,high}$, a logic fault occurs on A, otherwise, the circuit delay is less than the normal circuit delay. With decreasing R_b, the delay change becomes worse.

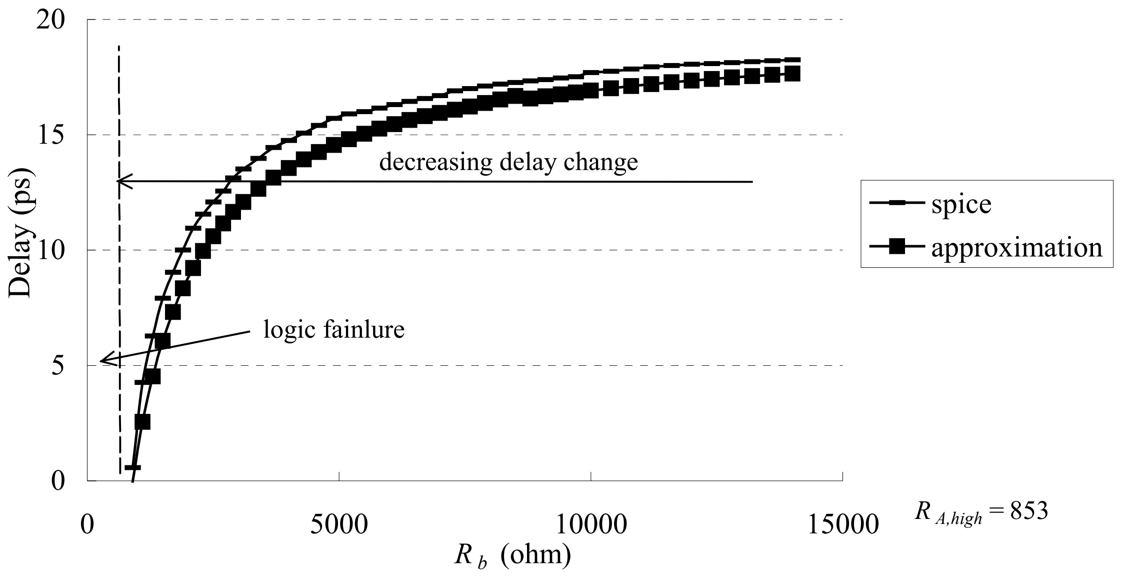


Figure 37. Approximated circuit delay compared with SPICE simulation for NMOS gate-to-source short, under a rising input.

When we use a falling input signal, the delay curves are shown in Figure 38, where an increasing delay change happens.

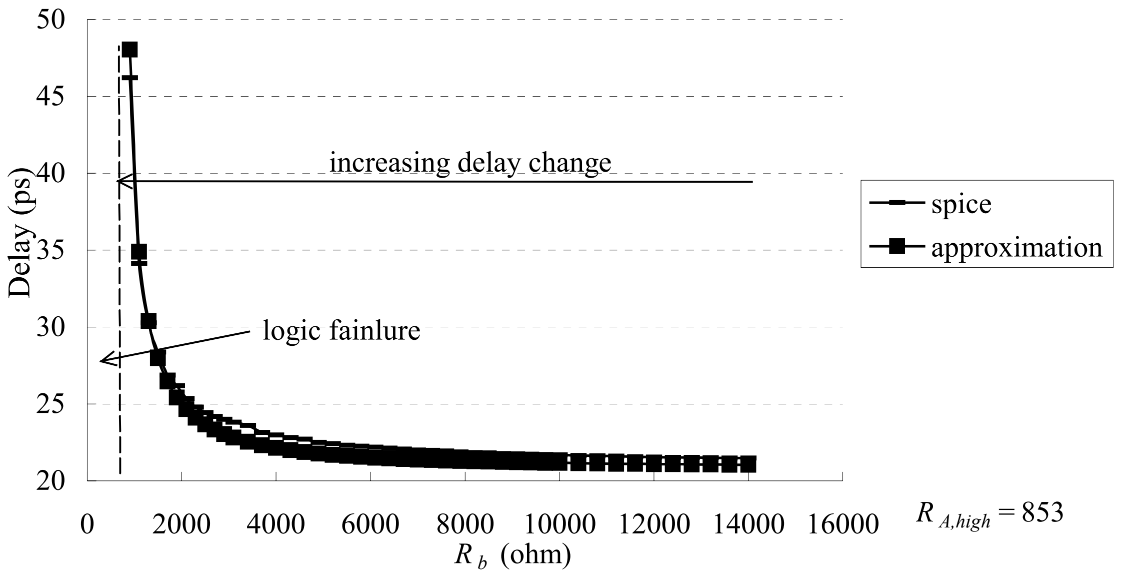


Figure 38. Approximated circuit delay compared with SPICE simulation for NMOS gate-to-source short, under a falling input.

In Figure 39 and Figure 40, we show the results of an NMOS gate-to-drain short under a rising input and a falling input respectively. The SPICE simulation shows that, with R_b close to the threshold short resistance, delay changes to increase. That is because the output waveform slope becomes more close to 0, resulting in the significant increasing of the delay at the 50% of VDD.

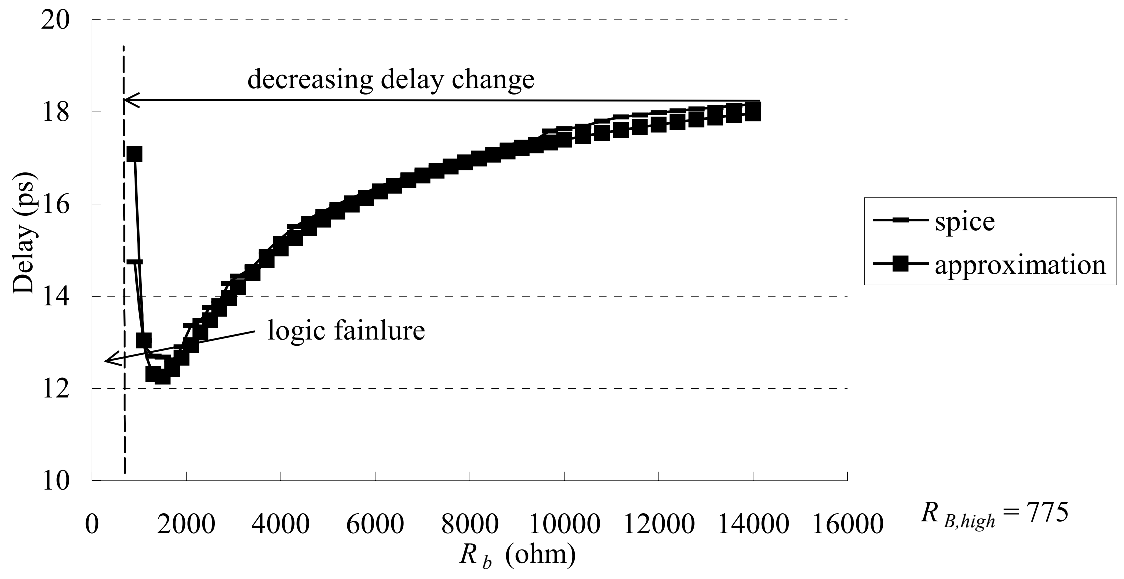


Figure 39. Approximated circuit delay compared with SPICE simulation for NMOS gate-to-drain short, under a rising input.

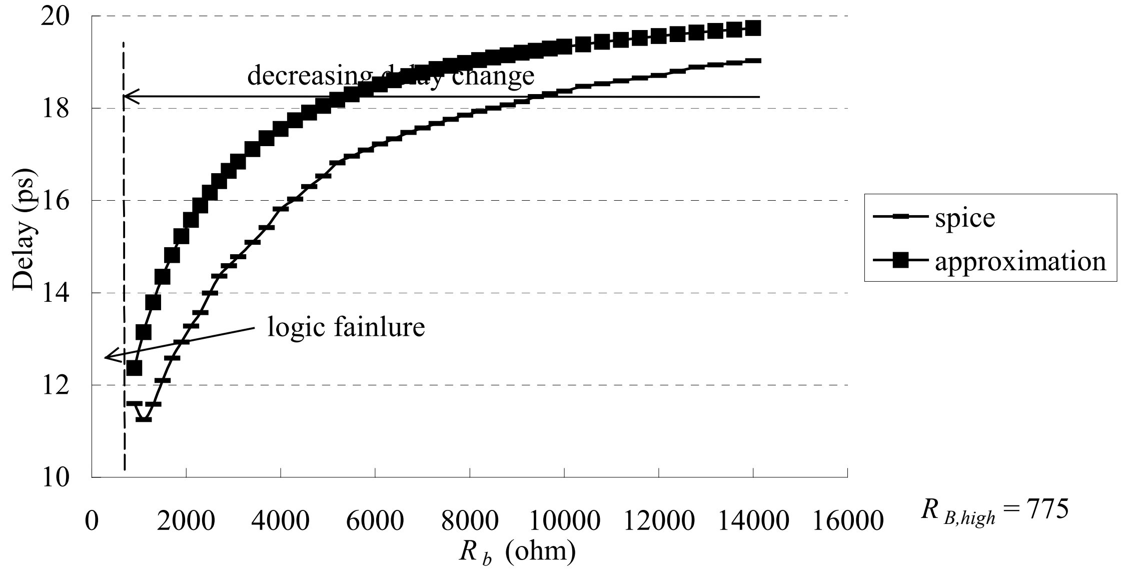


Figure 40. Approximated circuit delay compared with SPICE simulation for NMOS gate-to-drain short, under a falling input.

3. Fault Behavior and Vector Selection

We summarize circuit behaviors of resistive gate oxide shorts in Table 6. For each type of short, we list delay changes for different input signal patterns. In the table, T_f is the faulty transistor, and T_g is another transistor in the same NMOS/PMOS block with T_f. We assume that the rising/falling transition at the output of the faulty cell is only provoked by the signals on T_g and T_f, where signals on other transistors in the same NMOS/PMOS block keep static. "$T_f \mid T_g$" means the two transistors are in parallel, "$T_f \sim T_g$" means the two transistors are in series, "decrease/increase" means the delay decreases/increases compared to the short-free circuit, and "0" means the delay dose not change even in the presence of the short. The "static" status means the signal keeps logic high or logic low, to guarantee a pull-up/down path to conduct when T_g and T_f are in series, or to make the transistor cut off when T_g and T_f are in parallel. The table enumerates all input signal patterns that cause increasing/decreasing delay change at the output of the faulty cell. The amount of delay change is given by formulae such as (3.3) and (3.11).

TABLE 6. Behaviors for each type of short under different input signals.

Short type	Input on T_f	Input on T_g	Delay change
NMOS gate-to-source	Rising	Static	increase
	Falling	Static	decrease
	Static	Rising	$0(T_f \mid T_g)$, increase$(T_f \sim T_g)$
	Static	Falling	$0(T_f \mid T_g)$, decrease$(T_f \sim T_g)$
NMOS gate-to-drain short	Rising	Static	decrease
	falling	Static	decrease
	Static	Rising	decrease$(T_f \mid T_g)$, increase$(T_f \sim T_g)$
	Static	Falling	Increase$(T_f \mid T_g)$, decrease$(T_f \sim T_g)$
PMOS gate-to-drain short	Rising	Static	decrease
	falling	Static	Increase
	Static	Rising	$0(T_f \mid T_g)$, decrease$(T_f \sim T_g)$
	Static	Falling	$0(T_f \mid T_g)$, increase$(T_f \sim T_g)$
PMOS gate-to-source short	Rising	Static	decrease
	Falling	Static	increase
	Static	Rising	$0(T_f \mid T_g)$, $0(T_f \sim T_g)$
	Static	Falling	$0(T_f \mid T_g)$, increase$(T_f \sim T_g)$

Our fault model can be used to select the best vector to cause the greatest delay change. Note that in this dissertation we only consider delay fault due to the increasing delay change based on the path delay fault model [48]. Analysis of the decreasing delay change is similar. The necessary input signals on T_f and T_g to cause an increasing delay change can be found in Table 6. To further refine the vector selection, input signals of the driving cell must be considered. That is because input signals will affect the pull-up/down resistance of the driving cell, then further affect the value of delay changes. Such dependence is also observed and called "pattern dependence" in Hao and McCluskey [5], but it is used for logic fault model in their work and is insufficient for delay fault model. In our fault model, the dependence is explicitly expressed in formulas such as in formula (3.1) and (3.8).

We list the best test vectors in Table 7. In the table we use "strong" and "weak" to indicate the preferred input signals of the driving cell. For example, a strong pull-up driving strength for a NAND gate requires all inputs are logic 0, while a weak pull-up driving strength requires only one input is logic 0. Choosing the strong or the weak driving strength depends on the input signals of the faulty cells, as well as the network of the transistors structure. For example, for an NMOS gate-to-source short, we choose the weak driving strength when the input of T_f is rising and the input of T_g is static. However, when the input of T_f is static and the input of T_g is rising, and if T_f and T_g are in parallel, we can choose the strong ability.

TABLE 7. Vectors to cause the greatest increasing delay change.

Short type	Input of T_f	Input of T_g	Driving strength
NMOS gate-to-source	Rising	Static	Weak
	Static	Rising	strong($T_f \sim T_g$)
NMOS gate-to-drain	Static	Rising	strong($T_f \sim T_g$)
	Static	Falling	weak($T_f \mid T_g$)
PMOS gate-to-drain	Falling	Static	Strong
	Static	Falling	strong ($T_f \sim T_g$)
PMOS gate-to-source	Falling	Static	Weak
	Static	Falling	weak($T_f \sim T_g$)

4. Test Performance Improvement

To evaluate the benefit of the proposed fault model in delay test, we designed a circuit-level delay fault simulator for both delay test and logic test. We run experiments on ISCAS85 and ISCAS89 benchmark circuits. TSMC 180nm 1.8V technology is used. Shorts are assigned between gate and source or between gate and drain for each transistor in the circuit. For each run of the simulation, we assumed that there is only one short present, and evaluated the short resistance that makes the circuit fail in logic or timing requirements. We generated the input signal pattern that causes the worst effect of the resistive short according to Table 7. Then we used an ATPG tool proposed in [49] and [50] to generate the critical path satisfying the input signal pattern.

We calculated the delay of the longest path through the faulty transistor in the short-free circuit, and represented it as $D_{Rb=\infty}$. A delay fault is considered to have occurred if the short causes a delay increase of more than 10% of $D_{Rb=\infty}$. Therefore, the resistance value to cause 10% of $D_{Rb=\infty}$ is the maximum value that a delay test is able to detect. On the other hand, for logic test, we only need to calculate the BTR value for each short, which is the maximum value for a short to cause a logic fault. Normally the value of BTR is less than the value to cause a delay fault, which means the delay test will detects more potential defects than the logic test. Using our fault model and a kind of resistance distribution, we can not only illustrate this fact, but also numerically evaluate how much the delay test coverage exceeds the logic test coverage, which cannot be done using previous fault models.

As an example, in Figure 41, we show the circuit failure distribution vs. the short resistance for one ISCAS89 benchmark circuit s1488. There are 3829 potential fault sites for gate oxide shorts in the circuit. A circuit is considered faulty if the test detects a fault on one fault site. For a certain value of the short resistance, we perform delay test and logic test, and compute the percentage of faulty circuits, assuming all shorts are equally likely. In the figure, the X-axis indicates the short resistance to be evaluated, and the Y-axis indicates the percentage of faulty circuits for that value of the short resistance. As expected, when the short resistance is large, most circuits are fault free, so there is little difference between the two tests. And similarly, when the resistance is low, most circuits have a functional fault, and so are detected. For a certain value of the short resistance, we always found more faulty circuits in delay test. For example, for a short resistance of 1159Ω, 44.7% of circuits are found faulty in delay test, while only 22.9% are found faulty in logic test. The difference, 21.8%, is also the most benefit that the delay test exceeds the logic test in the circuit. In Figure 42, we show the similar simulation results of another ISCAS89 circuit s38417, which has 101,489 potential fault sites.

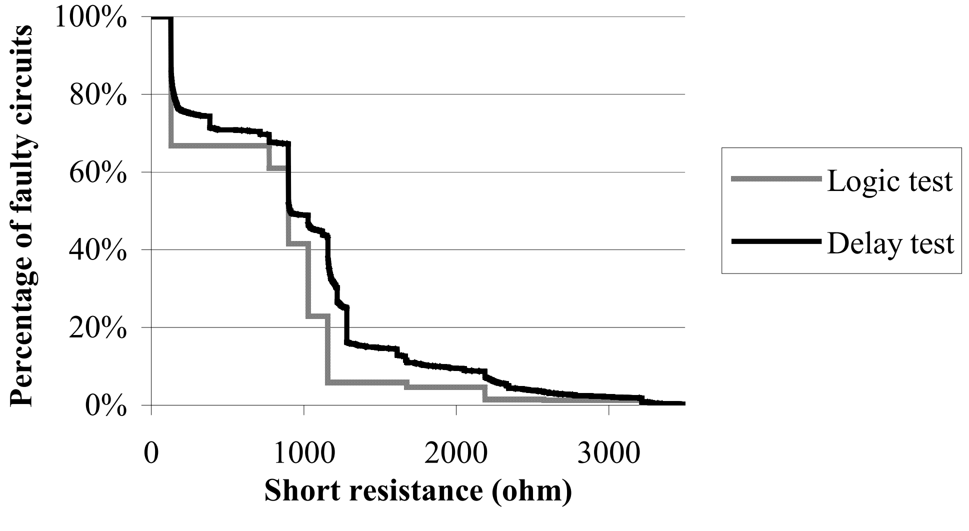


Figure 41. The circuit failure distribution vs. the short resistance for ISCAS89 circuit s1488.

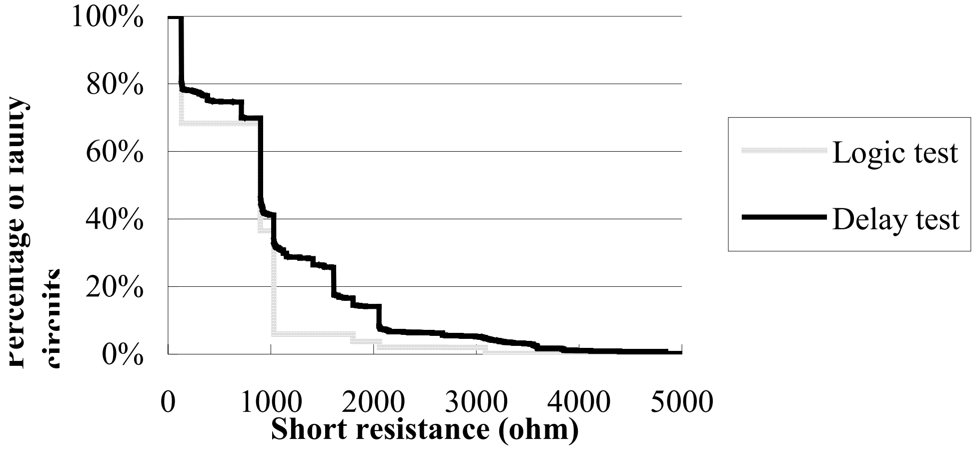


Figure 42. The circuit failure distribution vs. the short resistance for ISCAS89 circuit s38417.

IV. PARAMETRIC DELAY EVALUATION

1. ISCAS85/89 Benchmark Circuits

ISCAS85/89 benchmark circuits are proposed in Brglez *et al.* [51] for 10 combinational circuits, and in Brglez *et al*. [52] for 26 sequentail circuits. Traditionally, they are designed to verify the performance of the logic test, and only netlists in the stuctual level are provided. However, for delay test and timing analysis under nano-scale technologies, it is necessary to provide more realistic timing information. Especially, in order to research on the process variation impact on circuit delays, the layout and parasitic information must be provided.

In this work, we developed a standard cell library using TSMC 180nm technology, and generated layout and parasitic information for ISCAS85/89 circuits.

A. Standard Cell Library

The standard cell library is developed using the TSMC 180nm technology. The MOSIS DEEP rule SCN6M_SUBM (6 Metal, 1 Poly, 1.8V/3.3V, and λ=100nm) is used. The library The standard cell library consists of 28 standard cells, and contains all the cells that ISCAS benchmark circuits have. The content is listed in Table 8.

TABLE 8. Cell list in standard cell library.

Cell Name	Function
buf_1	Noninverting buffer, drive strength 1
inv_1	Inverter, drive strength 1
and2_1	2-input AND gate, drive strength 1
and3_1	3-input AND gate, drive strength 1
and4_1	4-input AND gate, drive strength 1
and5_1	5-input AND gate, drive strength 1
and8_1	8-input AND gate, drive strength 1
and9_1	9-input AND gate, drive strength 1
nand2_1	2-input NAND gate, drive strength 1
nand3_1	3-input NAND gate, drive strength 1
nand4_1	4-input NAND gate, drive strength 1
nand5_1	5-input NAND gate, drive strength 1
nand8_1	8-input NAND gate, drive strength 1
nand9_1	9-input NAND gate, drive strength 1
or2_1	2-input OR gate, drive strength 1
or3_1	3-input OR gate, drive strength 1
or4_1	4-input OR gate, drive strength 1
or5_1	5-input OR gate, drive strength 1
or8_1	8-input OR gate, drive strength 1
or9_1	9-input OR gate, drive strength 1
nor2_1	2-input NOR gate, drive strength 1
nor3_1	3-input NOR gate, drive strength 1
nor4_1	4-input NOR gate, drive strength 1
nor5_1	5-input NOR gate, drive strength 1
nor8_1	8-input NOR gate, drive strength 1
nor9_1	9-input NOR gate, drive strength 1
xor2-1	2-input XOR gate, drive strength 1
Dff	D flip-flop, drive strength 1

For each cell, the transistor gate length is 2λ. The width varies according to drive strength. For lowest drive strength 7λ (NMOS), 14λ(PMOS). For hi-drive-strength may use 14λ (NMOS), 28λ (PMOS), $W_p/W_n=2$ always for every primitive gates. The layout of each cell is provided with GDSII format and LEF format. The current version of the standard cell library contains only drive strength 1 cells. It can be expanded to contain more standard cells with different drive strength.

B. Delay Table

For each standard cell, we build up a two-dimensional delay table using SPICE simulation. One dimension is the input signal slew rate, in range of (20 ~ 1000ps) by 9 uneven samples. The other is the output load, in range of (5fF ~ 500fF) by 9 uneven samples. The range of input slew rate and the output load are chosen according to the parasitic information from the layout of ISCAS circuits.

The table is provided by "*.lib*" format used in Synopsys standard timing format, and is provided by "*.tlf*" format used in Cadence timing library format.

C. Layout and Parasitic Information

Based on the standard cell library, we generate the layout of each cell using Cadence Silicon Ensemble. The Cadence Hyper-extractor, a 2.5D parasitic extraction tool, is used to generated distributed interconnect parasitic information. The coupled capacitance between interconnect is generated using another parasitic extraction tool, Synopsys Arcadia. The input capacitance of each standard cell is pre-characterized in the library.

2. Linear Delay Modeling

There are many forms of process variation, see for example Nassif [16] and Stine *et al.* [39]. In this dissertation, we consider the systematic process variation, such as the variation on gate length, and the variation of metal width, metal thickness, and inter-layer-dielectric (ILD) thickness related to each interconnect layer. Our methods can be extended to include other process variation such as the threshold voltage, the supply voltage and the temperature, as long as the approximated delay can be expressed as a linear function of the process variables within their variation ranges.

In order to calculate the path delay under process variation, we first compute the buffer-to-buffer delay. The buffer-to-buffer delay is defined as the delay from the input pin of a cell to the input pin of a downstream cell. After each buffer-to-buffer delay in the circuit is computed, the delay of any path can be easily obtained by adding up buffer-to-buffer delays along the path.

We approximate the buffer-to-buffer delay as a linear function of process variables:

$$d(\mathbf{x}, s) \approx d_0(s) + b_1(s)x_1 + b_2(s)x_2 + \ldots + b_p(s)x_p, \tag{4.1}$$

where $d_0(\mathrm{s})$ is the nominal delay, $\mathbf{x}=(x_1, x_2, \ldots, x_\mathrm{p})$ is the vector of process variables, each representing the deviation from the nominal value, s is the input signal slew, and $b_i(\mathrm{s})=\partial d/\partial x_\mathrm{i}$ is the delay sensitivity to process variable x_i. We assume both the nominal delay and delay sensitivities are functions of input signal slew s.

The validity of the linear model is supported by extensive simulation. We performed multiple parasitic extraction and SPICE simulation under different process conditions. It is found that for any single process variation variable, its effect on delay is approximately linear within its small variation range. In Figure 43 we show the SPICE simulation result on a buffer-to-buffer segment in the circuit for several typical process variation variables. Each variable changes within its typical range (metal width ±5%, metal thickness ±20%, ILD thickness 40%, and gate length ±5%). In addition, since the systematic process variables in our consideration are determined at different stages of the manufacturing process, we can assume they are independent of each other. Furthermore, within the small variation range of each variable, the effect of each variable is additive. For example, considering width variation on metal 2 and metal 3, we denote the delay variation under metal 2 width variation and under metal 3 width variation as Δd_{w2} and Δd_{w3} respectively, and denote delay variation under both variations happening as Δd_{w1+w2}. The width changes on both metal 2 and metal 3 are 5% of the nominal metal width. Then we use $\Delta d_{w1}+\Delta d_{w2}$ to approximate Δd_{w1+w2}. In Figure 44 we show the error distribution over 160 buffer-to-buffer delays in circuit c432. From the figure, for most of buffer-to-buffer segments, the error is considerably smaller. Because of the effect of layer overlapping between metal 2 and metal 3 in the layout, the error is over 20% for few buffer-to-buffer segments. It is interesting to study more complex models to compensate for such segments and keep the additive property. Nevertheless, for most buffer-to-buffer segments the effect of metal 2 and metal 3 width variation can be considered as additive. The similar result is found in other process variables.

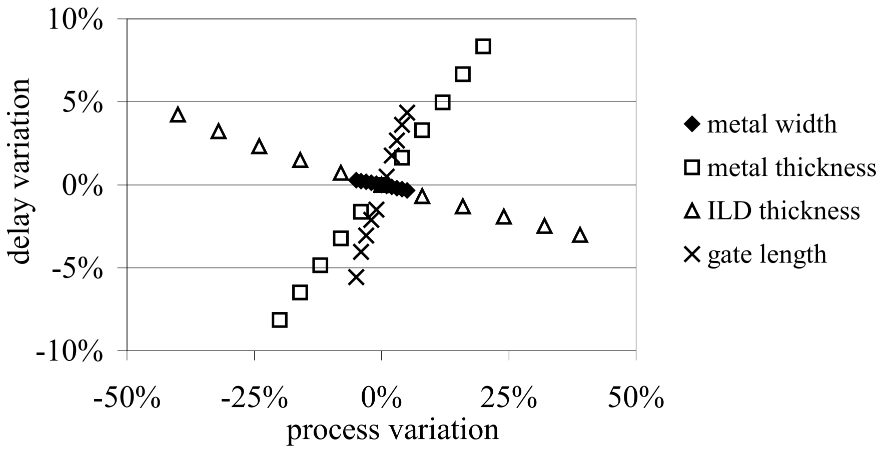


Figure 43. Delay variations due to process variation are linear in SPICE simulation. The x-axis indicates process variation and the y-axis indicates the percentage deviation from the nominal delay.

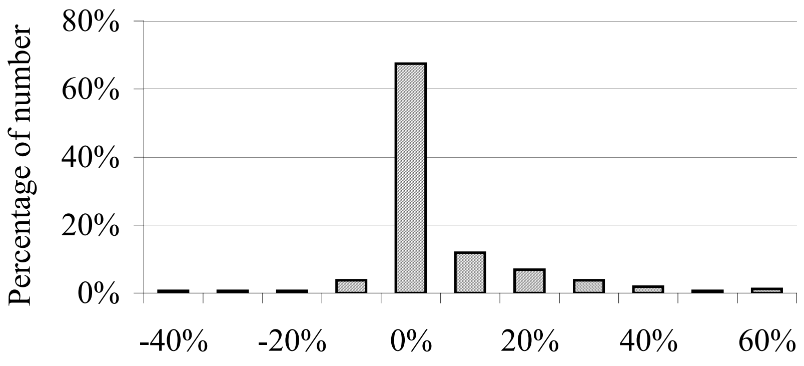


Figure 44. The delay effect of process variation is additive, which is demonstrated by the error distribution of approximating Δd_{w1+w2} with $\Delta d_{w1}+\Delta d_{w2}$ over 160 buffer-to-buffer segments in circuit c432.

The effect of signal slew has been studied in previous research, for example, in variational delay evaluation [16]and in static timing analysis [53]. We assume the effect of process variation on output signal slew is small and propagate signal slews under the nominal process condition. The computation of nominal delay and signal slew can be done by any commercial tool, and is not the focus of this paper. The key issue is to efficiently compute delay sensitivities b_1, b_2, …, b_p.

A buffer-to-buffer segment in a circuit is represented by a cell driving an RC circuit, which consists of distributed R_1, …, R_n and distributed C_1, …, C_n on interconnect and sink capacitance C_s for each downstream cell. The RC circuit can be a tree-like structure or a path-like structure. Parasitic RCs are generated by commercial parasitic extraction tools, and each pair of parasitic (R_i, C_i) is related to one metal segment or a contact/via on interconnect.

A. Computation on RC Variations

The sink capacitance C_s is only related to device parameters of the downstream cell. In this work, we ignore the variation of C_s. Thus $\partial C_s/\partial x_i$ is zero for all process variables.

Parasitic RCs on interconnect vary in different process conditions. The value of $\partial R_j/\partial x_i$ can be easily derived from the basic resistance computation formula $R = \rho L/(WT)$, where ρ is the resistive constant, L, W and T is the length, width and thickness the metal segment respectively.

However, it is more difficult to compute $\partial C_j/\partial x_i$. This is because the parasitic capacitance of a metal wire depends not only on the wire itself, but also on the neighboring condition. Formula-based methods for parasitic extraction are no longer used and are replaced by more accurate 2.5D/3D tools. For these tools, there is no explicitly capacitance formula we can use. To make our method widely applicable to different design flows, the computation of $\partial C_j/\partial x_i$ must be independent of any particular parasitic extraction tools. At the same time, we need to avoid multiple extractions on the whole circuit for different process variable.

To get $\partial C_j/\partial x_i$ for any process variable x_i efficiently and accurately under any complex neighboring condition, we introduce the concept of *unit capacitance variation u_{ik}*, which is an estimate of the percentage variation of parasitic capacitance on metal k, with respect to process variable x_i. In practice, we randomly choose n parasitic capacitance on metal k in a circuit, and calculate u_{ik} by:

$$u_{ik} = \frac{1}{n}\sum_j \frac{\Delta C_{jk} / \Delta x_i}{C_{jk}}, \tag{4.2}$$

where Δx_i is a small change of process variable x_i, C_{jk} indicates a parasitic capacitance on metal k under the nominal condition, andΔC_{jk} is the variation of C_{jk} due to Δx_i.

For a given process technology, the value of $(\Delta C_{jk}/\Delta x_i)/C_{jk}$ is in a considerable small range. In Figure 32 we show the distribution of $(\Delta C_{jk}/\Delta x_i)/C_{jk}$ due to the wire width variation on metal 2 in ISCAS85 circuit c432 for 406 sample capacitance. From the figure, we can see that for most parasitic capacitance C_{jk}, the value of $(\Delta C_{jk}/\Delta x_i)/C_{jk}$ is around 0.61 with small deviations.

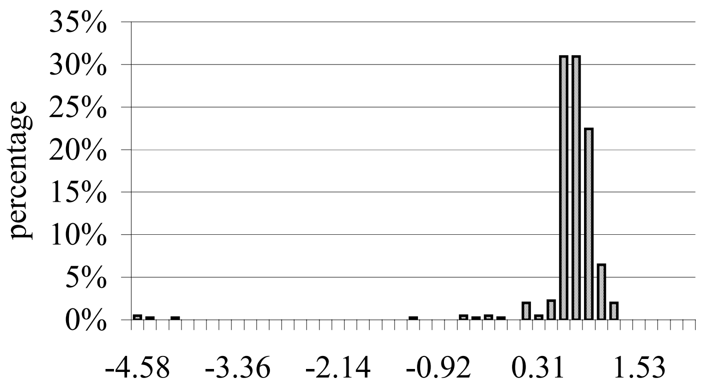


Figure 45. The distribution of $(\Delta C_{jk}/\Delta x_i)/C_{jk}$ due to metal 2 width variation on 406 samples in ISCAS85 circuit c432. The x-axis indicates values of $(\Delta C_{jk}/\Delta x_i)/C_{jk}$.

The unit capacitance variation u_{ik} is pre-computed for each metal layer with respect to each interconnect process variable, and is used to estimate the variation for any parasitic capacitance on metal k under a small change of x_i. For any C_j is on metal k, we have:

$$\partial C_j/\partial x_i = u_{ik}\cdot C_j. \tag{4.3}$$

B. Computation on Delay Sensitivity

We assume a *k*-factor table of delay with respect to input slew *s* and load C_L is given. If such a table is not available, we construct one using existing technology. The delay table under the nominal gate length is named as *nominal table*. We then build another *k*-factor table with the same indices of the first table, where each entry is the delay under a small change of gate length. The change of gate length ΔL_g is 3% of the nominal gate length in our experiments. This table is named as *variational table*.

We apply two delay models in delay sensitivity computation. One is lumped C delay model, and the other is effective capacitance delay model.

i. Lumped C Delay Model

In the lumped C delay model, all parasitic resistance on interconnect are removed, and all parasitic *capacitance* and the sink capacitance are lumped into one single load capacitance C_L. Then we have $C_L=\sum C_j$. Then we refer to the nominal table and generate delay d according to s and C_L.

For delay sensitivity to gate length variation, we refer to the variational table according to s and C_L, and generate delay d'. Then we calculate $\partial d/\partial x_i = \partial d/\partial L_g = (d-d')/\Delta L_g$.

For delay sensitivity to interconnect process variable x_i, we first calculate variation of C_L under a small change of x_i as $\Delta C_L=\Delta x_i\cdot\sum (u_{ik}\cdot C_j)$, where Δx_i is 5% of the nominal value of x_i. Then we refer to the variational table and calculate delay d' according to $C_L+\Delta C_L$. Therefore, we calculate $\partial d/\partial x_i = (d-d')/\Delta x_i$.

ii. Effective Capacitance Delay Model

For each buffer-to-buffer segment in the circuit, effective capacitance C_{eff} rather than lumped capacitance C_L is used to refer to the table. Thus we can consider the interconnect resistance shielding effect more accurately. There are several effective capacitance methods can be used, such as iterative method [46] and non-iterative method [54][55]. For the speed concern, we use non-iterative method here. The method proposed in [55] is used for RC interconnect and is difficult to be applied in buffer-to-buffer segment. Thus we apply the method proposed in [54], which evaluates effective capacitance by matching the delay of a cell with a Π load and the delay of a cell with a single effective capacitance load. The delay under s and C_{eff} is named as d.

For delay sensitivity to gate length variation, we first compute the effective capacitance under the variational gate length. Under the gate length change ΔL_g, effective capacitance C'_{eff} is recalculated using the method proposed in [54]. Note here the Π load does not change with ΔL_g. Then we use C'_{eff} and s to refer to the variational table, and generate delay d'. Then the delay sensitivity to gate length variation is calculated by $\partial d/\partial x_i = \partial d/\partial L_g = (d - d')/\Delta L_g$.

For delay sensitivity to interconnect process variables, we need variational effective capacitance to refer to the nominal table. Thus we have to compute the new effective capacitance C'_{eff} under process variation Δx_i. However, it costs too much to derive a new Π load and compute C'_{eff} accordingly. Instead we use $\Delta C_{eff} = C_{eff} \cdot \Delta C_L/C_L$ to approximate the change of C_{eff} due to Δx_i, where C_L is the lumped capacitance, and ΔC_L is variation of C_L and is calculated by $\Delta C_L = \Delta x_i \cdot \sum (u_{ik} \cdot C_j)$. Therefore $C'_{eff} = C_{eff} + \Delta C_{eff}$ is used to refer to the nominal table and generate delay d', then the delay sensitivity to x_i is calculated by $\partial d/\partial x_i = (d - d')/\Delta x_i$.

3. Experimental Results

We apply our methods to ISCAS85 circuits using a UNIX server running on Solaris 2.7. The systematic process variation variables considered in our paper are variations of the transistor gate length, the width of 5 metal layers, the thickness of 5 metal layers and the thickness of 5 inter-layer-dielectrics (ILD). We apply the following manufacturing ranges of these variables: gate length ±6%, metal width ±5%, metal thickness ±20%, and ILD thickness ±40%. The range of delay variation is about ±10% of the nominal delay.

We first show the running time comparison between the traditional RSM and new method in Table 9. For each circuit we perform RSM and our new method respectively to generate the parasitic delay model for all buffer-to-buffer segments in the circuit. RSM is implemented by SPICE simulation with its running time listed in the third column. The path delay is computed by summing buffer-to-buffer delays. The running time of our method is listed in followed columns. Compared to RSM, our method achieves significant speedup. The running time of the method based on lumped C delay model is faster than the method based on effective capacitance delay model by 2-5 times. The reason is that the method based on effective capacitance method spends more cost on Ceff computation.

TABLE 9. Running time comparison between the traditional RSM and new methods for ISCAS85 circuits.

Circuit	# of buffer-to-buffer delays	Running time		
		RSM (hh:mm)	New Methods (s)	
			Lumped C	Effective C
c432	343	0:41	0.014	0.020
c499	440	1:03	0.017	0.026
c880	755	1:30	0.014	0.053
c1355	1096	2:13	0.044	0.084
c1908	1523	2:48	0.075	0.304
c2670	2292	4:19	0.108	0.456
c3540	2961	5:39	0.143	0.466
c5315	4509	>8 hr	0.196	0.785
c6288	4832	>9 hr	0.200	0.846
c7552	6253	>10 hr	0.308	1.600

To evaluate the accuracy of our method, we perform RSM and our method on the longest path of each circuit. Results are compared under the corner condition. In our experiments, the path delay under the nominal process condition d_0 is computed by SPICE simulation. Under the corner condition, the parametric variational delay computed by the traditional RSM is denoted as d' and the parametric variational delay calculated by our method is denoted as d'' using function (4.1). Then the delay error under the corner condition is computed by $(d''-d')/(d_0+d')$. This value indicates the result of our method is how close to the result of RSM.

The results are shown in Table 10, where the number of cells in the longest path is listed in the second column, the path delay computed by RSM is listed in the third column and the delay variation under the worst case corner condition is listed in the fourth column. From the table, we can conclude that the method based on effective capacitance delay model is more accurate. Its delay error is less than 3% and for most circuits the error is around 1% of the path delay, where the delay error of the method based on lumped C model is less than 5%.

TABLE 10. Accuracy comparison between the traditional RSM and new methods for ISCAS85 circuits.

Circuit	# of cells in path	Worst case delay computed by RSM (ps)	Delay Var. (%)	Delay error under worst case corner (%)	
				Lumped C	Effective C
c432	17	698.5	10.80	-4.38	-0.01
c499	11	464.6	9.83	-2.09	-0.60
c880	24	530.3	9.54	-2.00	-0.36
c1355	24	609.1	10.37	-3.64	-2.98
c1908	40	724.5	11.02	-2.46	-1.55
c2670	32	947.6	10.83	-2.84	-0.65
c3540	47	1103.1	9.97	-0.33	-1.24
c5315	49	994.5	10.02	-2.50	-1.15
c6288	124	2853.4	9.53	-0.11	-1.71
c7552	41	690.9	10.28	-2.86	-1.32

V. LONGEST PATH SELECTION

1. Delay Test Using Longest Paths

A. Delay Test Basics

Delay test of combinational circuits is to ensure that the signal from any primary input to any primary output is propagated in less time than the system clock cycle time. A circuit is considered faulty if the delay of any path exceeds the specification. The delay increase due to a local defect, such as a resistive bridge or a resistive open, may cause a timing violation on the path through the defect, and can be modeled as a delay fault [15][32]. Such delay increase is localized to a gate output or an interconnect wire in the circuit, where the localized position is called a *local fault site* in this paper. Generally, the local delay fault is modeled as an additional delay Δ along the path through the fault size.

Testing the longest path through the local fault site will capture the delay increase due to the fault. For example, in the combinational circuit of Figure 46, there are two paths P_1 and P_2 through a common local defect. If there is no defect, the delay of P_2 is larger than P_1. Then with the additional delay caused by the local defect, the delay of P_2 is more likely to exceed the timing specification T_{spec}. Therefore test on the longest path is the most likely to capture the delay increase due to the fault.

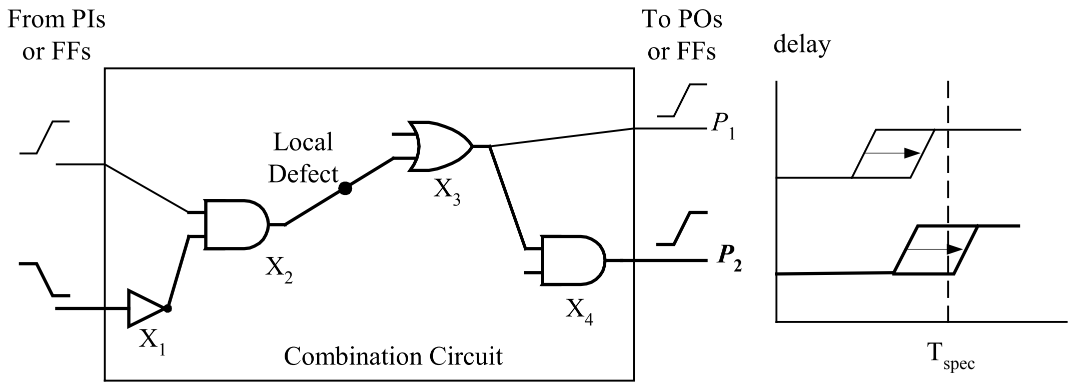


Figure 46. Test on the longer path is more likely to capture delay defect.

Modern delay optimization tools tend to make many paths critical or near critical [37], resulting in too many paths for test. Pruning some of the paths based on structural correlation and process variation correlation is an effective approach to reduce the number of paths. If two paths share some nets or gates, there is a *structurally correlation* between them. Similarly, if two nets run on the same metal layer, there is a *process correlation* between them. Luong and Walker [27] proposed a pruning technique using both the structural correlation and the process correlation. As a result, they significantly reduced the number of paths. However, they only considered the longest paths for the entire circuit, instead of the longest paths through every local fault site. Furthermore, they did not consider interconnect delay. Tani *et al*. considered the longest paths through every local fault site [38]. They used a min-max comparison method, with the help of the structural correlation but not process correlation. As a result, their approach is overly pessimistic and produces too many paths. Liou *et al*. [28] used Monte Carlo simulation to select a set of critical paths that maximizes the probability of covering all critical paths under all process conditions. However, Monte Carlo simulation is very slow for large circuits and no running time is given for their method.

B. Longest Path Redefined

When variations are not considered, there is only one path whose delay is the maximum in the combinational circuit, and the problem of finding the longest and testable paths that cover all local fault sites has been extensively studied [33][34][35].

When process variation is considered, the path delay becomes a function of process variables. Among all paths through a fault site, there are often multiple paths whose delay can be the maximum under different process conditions [36]. For each fault site *s*, we call a path *longest* for *s* if the path has the maximum delay among all paths through *s* under some process conditions. On the other hand, we call a path *redundant* for *s* if the path can never be longest for *s* under any process condition.

C. Test All Longest Paths

Traditionally, tests are only performed on the longest paths under the nominal or worst-case process condition. However, this might be insufficient. As an example in Figure 47, we show the delay of four path through one common local fault size under one process parameter *x*. Under the nominal process variation, the delay of path P_2 is the maximum. However, under the worst-case corners (min and max), the delay of P_3 and P_1 is the maximum respectively. Obviously, only testing the path under one special process corner cannot maximize the fault coverage, because we do not what the actual process condition is for a chip in test. On the other hand, it is inefficient to test all these paths. For instance, test on P_4 is wasteful because it cannot be a longest path in any process condition. Therefore, testing on all longest paths under any process conditions is the only way to satisfy the fault coverage, as well as to minimize the test cost.

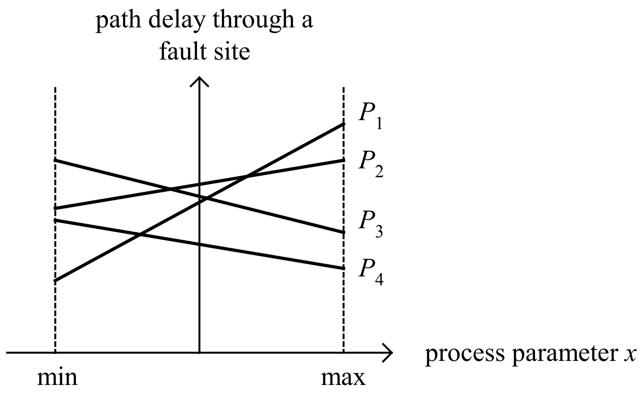


Figure 47. Four path delays under one process parameter x.

In the following we present a new method to select longest paths for each local fault site in the circuit. To maximize fault coverage, we want to find as many longest paths as possible. On the other hand, to minimize test costs, we want to find as few paths as possible. Given a set of testable paths, our method first models the path delay as a linear function of process variation variables, then uses two pruning algorithms to remove paths that are redundant or almost redundant. We repeat the process for each fault site in the circuit, and the remaining paths are longest paths for delay test. Experiments on the ISCAS circuits show that the new method is efficient and significantly reduces the number of paths for test, compared to the previous best method. We consider process variations of devices and interconnect in our work, and the method can also be applied in path selection under operating variations of supply voltages and temperature [39].

2. Path Pruning Algorithms

Based on the linear delay model presented by PARADE, the delay of a path can be derived as a linear function by accumulating all buffer-to-buffer delays defined in (5.1) along the path:

$$D(\mathbf{x}) = d_0 + d_1x_1 + d_2x_2 + \cdots + d_px_p, \tag{5.1}$$

where d_0 is the nominal path delay, and d_1, d_2, …, d_p are coefficients for process variation variables, $\mathbf{x}=(x_1, x_2, \ldots, x_\mathrm{p})$ is the vector of process variables, each representing the deviation from the nominal value.

Let $\boldsymbol{P} = \{P_1, P_2, \ldots, P_n\}$ be a set of testable paths through a local fault site in the circuit, and let the delay of each path P_i be $D_i(\mathbf{x})=d_{i0}+ d_{i1}x_1+ \cdots +d_{ip}x_p$. The range of all process variation variables is defined as $\boldsymbol{G} \subset \Re^p$, where $\boldsymbol{G}=\{(x_1, \ldots, x_p) \mid l_j \le x_j \le h_j, j=1, \ldots, p\}$, and l_j and h_j are the lower and upper bounds of x_j respectively. Then, any path P_q is a longest path in $\boldsymbol{P}$ if and only if there exists $\mathbf{x}' \in \boldsymbol{G}$ such that:

$$D_q(\mathbf{x}') \ge D_1(\mathbf{x}'), D_2(\mathbf{x}'), \ldots, D_n(\mathbf{x}'). \tag{5.2}$$

Verifying whether the set of inequalities (5.2) can be satisfied is known as the feasibility problem of linear programming (LP). When the dimension p is fixed, LP can be solved in $O(n)$ time [56]. However, the constant factor in the time complexity is exponential with the dimension p, resulting in high costs for large p. To reduce the running time in the case of large p, we replace LP with two heuristics. Heuristic H_1 prunes redundant paths using less strict constraints, while Heuristic H_2 refines outputs of Heuristic H_1 to further reduce the number of paths for delay test.

A. Heuristic H_1

To determine if path P_q is longest, we define its *rough domain* of process variable x_k, with respect to path P_i as:

$R_{ki} = [l_{ki}, h_{ki}]$, where

$l_{ki} = \min_{\mathbf{x}\in\mathbf{G}}\{x_k | D_q(\mathbf{x}) \geq D_i(\mathbf{x})\}$,

$h_{ki} = \max_{\mathbf{x}\in\mathbf{G}}\{x_k | D_q(\mathbf{x}) \geq D_i(\mathbf{x})\}$.

Intuitively, R_{ik} specifies the possible values of x_k such that P_q is longer than P_i. The computation of l_{ki} and h_{ki} is straightforward.

The heuristic is as follows:

```
$H_1$: Prune redundant paths
Input: path set $\boldsymbol{P}$, process range $G$
1 For each path $P_q \in \boldsymbol{P}$, do
2     For each process variable $x_k$, do
3         Initial rough domain $R_k = [l_k, h_k]$.
4         For each path $P_i$, $i$=1, …, $n$, $i \neq q$, do
5             Compute the rough domain $R_{ki}$.
6             Update $R_k = R_k \cap R_{ki}$.
7         End
8         If $R_k = \emptyset$, $P_q$ is "redundant" and pruned.
9     End
10 End
```

Heuristic H_1 prunes path P_q if the intersection of rough domains of any process variable for P_q with respect to other paths is empty. This is because if the intersection is empty, there does not exist any $\mathbf{x}\in\boldsymbol{G}$ such that P_q is the longest under process condition $\mathbf{x}$. Then according to the definition, P_q is redundant. The worst-case time complexity of the heuristic is $O(n^2p^2)$, since there are $O(n^2p)$ rough ranges and each takes $O(p)$ time to compute. Note that although Heuristic H_1 prunes a large number of redundant paths, some redundant paths may escape when these paths are shorter than the combination of other paths.

B. Heuristic H_2

Among the longest paths, some paths are only slightly longer than others under every process condition. A path P_q is called *insignificant* if there is a longest path P_i such that their maximum delay difference is small:

$$\max_{\mathbf{x}\in G}\{D_q(\mathbf{x}) - D_i(\mathbf{x})\} \le \varepsilon,$$

where ε is a user-specified threshold. If ε is small, say 1% of the maximum nominal path delay, then testing P_q after testing P_i achieves little delay test coverage improvement. Therefore P_q should be pruned. The following heuristic prunes insignificant paths.

```
$H_2$: Prune insignificant paths
Input: Path set $\boldsymbol{P}$, pre-specified threshold $\varepsilon$
1 For each path $P_q \in \boldsymbol{P}$, do
2     For any other path $P_i \neq P_q$, do
3         If $\max_{x\in G}\{D_q(\mathbf{x}) - D_i(\mathbf{x})\} < \varepsilon D_i(\mathbf{0})$
4             Prune $P_q$ as insignificant.
5     End
6 End
```

Heuristic H_2 compares the delay difference between each pair of paths under the worst-case process corners. The time complexity is $O(n^2p)$. We perform H_2 on the output of H_1, and the remaining paths are kept for delay test.

3. Longest Path Generation

Given a set of testable paths, we generate the set of longest paths by pruning redundant paths from it. Testable paths are generated by the algorithm in [49] and [50] and are ranked in the order of non-increasing nominal delays. The path *with* the largest nominal delay has index 0, and the path with the second largest nominal delay has index 1, etc. For each fault site, we first request a batch of *K* longest paths, indexed from 0 to *K*–1. Then the path pruning algorithms are applied to prune all redundant paths. Finally, the probability that a path in the next batch could be longest is estimated. If the probability is less than a specified criteria value, for example 0.1%, the procedure stops. Otherwise, we request the next batch of paths from the path generator, indexed from *K* to 2*K*–1. The above procedure is repeated until the stop criterion is satisfied. The flowchart of longest path generation is shown in Figure 48.

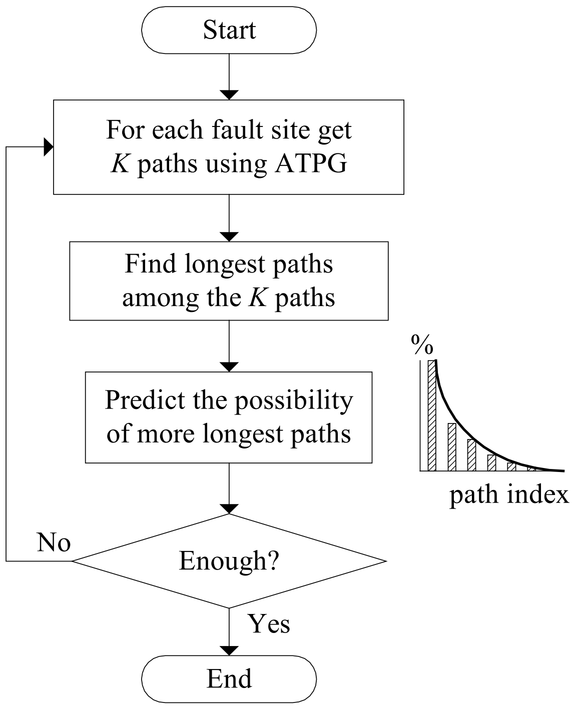

Start
For each fault site get *K* paths using ATPG
Find longest paths among the *K* paths
Predict the possibility of more longest paths
%
path index
No
Enough?
Yes
End

Figure 48. Flowchart of longest path generation.

To estimate the probability that longest paths could exist in the next batch of paths, we consider the distribution of the already generated *longest* paths versus path indexes for all fault sites. Let $f(k)$ be the percentage of fault sites where the path with index k is a longest path. Because of the non-increasing order of the nominal path delay and path delay correlations, paths with greater indexes are less likely to be longest paths. Thus, the value of $f(k)$ decreases with the increasing of index k, and can be modeled by a rational function:

$$f(k) = \frac{1}{1 + ak + bk^2}, \tag{5.3}$$

where parameters a and b can be computed by performing curve fitting on the distribution of already generated longest paths. Let the index of path batch be l, we estimate the maximum percentage in the next batch of paths as $f(l \cdot K)$. If the value of $f(l \cdot K)$ is greater than 0.1%, we consider the next batch of paths and recalculate $f(k)$. Otherwise, the procedure stops. Although parameters of $f(k)$ changes when a new batch is considered, experiments show that they vary little when a proper K is used. As an example, we show the actual longest path distribution versus path indexes in Figure 49 for an ISCAS85 circuit, where the x-axis indicates the path index and the y-axis indicates the percentage of longest paths. The *batch* size K is 10 and four batches are used in the whole procedure. In Table 11 we show parameters of $f(k)$ for each time a new path batch is considered, and we also show the estimated and the actual maximum percentage of longest paths in the next path batch.

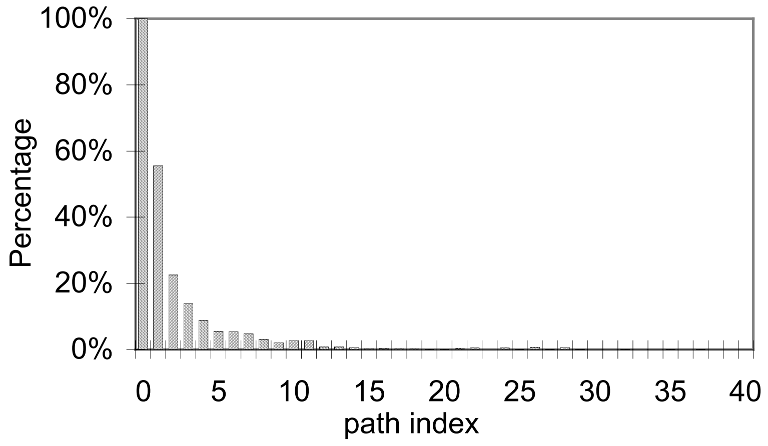


Figure 49. Distribution of longest paths versus path indexes.

TABLE 11. Parameters of *f*(*k*) and the estimated and the actual maximum percentage of longest paths.

Index of batch (l)	A	B	f(k) at $k = l \cdot K$	Maximum percentage in batch $l + 1$
1	-0.1959	0.7446	1.36%	2.69%
2	-0.2060	0.7534	0.34%	0.67%
3	-0.2063	0.7536	0.15%	0.12%
4	-0.2067	0.7540	0.08%	0.01%

Because a longest path through a fault site is very likely to be a longest path through another fault site in the circuit, a path collapsing procedure is performed to discard the shared paths among all fault sites in the circuit, after the longest path set of each fault site is generated. The procedure can be implemented in linear time in terms of the number of paths. The collapsed path set is the longest path set that covers all delay fault sites in the circuit and must be tested.

4. Experimental Results

The experiments were performed for all ISCAS85 combinational circuits and the three largest ISCAS89 sequential circuits. We used Cadence Silicon Ensemble^{TM} for circuit layout generation and parasitic extraction under TSMC 180 nm 1.8 V 5-metal technology. We implemented heuristics in C on a 2.8 GHz Pentium 4 with 1 GB memory running at WindowsXP. The process variations considered are variations in transistor gate length, metal width, metal thickness and inter-layer-dielectric (ILD) thickness. There are a total of 16 variables for the 5-metal layer technology. The ranges of process variation variables are as follows: gate length ±5%, metal width ±5%, metal thickness ±20%, and ILD thickness ±40%. Under such variation ranges, path delays vary within ±10% in our experimental circuits.

We first compare the performance of our new method with the min-max method, which is the previous best method for the problem [38]. Considering path structural correlation, the min-max method first identifies shared gates between different paths and eliminates the delay of shared gates from path delays. Then min-max comparison is performed on remaining delays. In experiments of the min-max method, we used parameters α=1.0 and β=10% to achieve a ±10% min-max delay range. In our new method, path structural correlation is implicitly considered in the formation of delay inequalities, and process correlation is handled by using the same set of variables in each delay function. Therefore, the new method is able to identify and prune more redundant paths than the min-max method does.

We assumed the output of each cell in ISCAS85 circuits as a possible fault site. For sequential ISCAS89 circuits, we considered the combinational circuit between any pair of flip-flops and assumed there can be a delay fault at the output of the driving flip-flop, as well as gate outputs. A path generator [49] [50] was used to provide critical and testable paths in batches of 50, where paths are indexed from 0 and sorted in the order of non-increasing nominal delay. For each batch we applied path pruning heuristics and collected remaining paths into the path set for test. Because of the non-increasing order of the nominal path delay, paths with greater indexes are less likely to be longest paths. Then if most paths in a batch are pruned, e.g. 45 out of 50 are pruned, which means the probability for the next batch to contain a longest path is small, the procedure stops. The stopping threshold is user-defined, and we used 90% in experiments. Although it is possible that a longest path exists in the following batch, the number of "escaped" paths is very small if we stop when 90% or more are pruned in a batch of 50 paths. In the experiments, the number of paths in the second batch is at most 1.79% of all longest paths we selected, and no longest path is found in the third batch. The reason for this behavior is that path delay correlation is high enough that two paths of very different index are unlikely to both be longest and still pass heuristic H_2. Delay test coverage will not be significantly degraded if only a small number of longest paths escape, since these paths will be only slightly longer than the tested paths. Luong and Walker [27] used a similar batch-based method to decide when to stop global longest path generation.

The comparison between our method and the min-max method is shown in Table 12. In the table, column "# of critical paths" indicates the total number of paths through each fault site within 20% of the nominally longest path delay. In column "paths for test" we list the total and the average number of longest paths for all fault sites in the circuit. The percentage of longest paths in the critical paths is shown in column “percentage”. The running time is shown in column “time (s)”, where the time for the path generator is not included. We do not list the result of the min-max method for circuit c6288 and larger circuits because the running time is more that several hours. As shown in the table, the number of longest paths selected by the new method is only 1%-6% of that selected by the min-max method. This indicates that only a small percentage of paths are actually longest when structural and process correlations are used, compared to just using structural correlation in the min-max approach. The maximum average number of paths to be tested by the new method is 4.4. That means only a few paths need to be tested for each fault site. In addition, the new method is 300-3000 times faster than the min-max method. This is because the min-max method takes too much time identifying shared gates among paths. We used LP to verify the results of the new method and found that only 5% of the selected paths are pruned. That indicates that our method achieves close to the minimal test set at much lower cost.

TABLE 12. Performance comparison between the min-max method and the new method.

Circuits	# of fault sites	# of top 20% paths	Min-max method				New method			
			Paths for test		Percentage (%)	Time (s)	Paths for test		Percentage (%)	Time (s)
			Total	Avg.			Total	Avg.		
c432	140	9263	5946	42.5	64.2	21	249	1.8	2.7	0.06
c499	202	5962	5384	26.7	90.3	47	204	1.0	3.4	0.09
c880	383	19641	9340	24.4	47.6	237	399	1.0	2.0	0.16
c1355	546	236771	185803	340.3	78.5	1070	598	1.1	0.2	0.30
c1908	845	136530	93061	110.1	68.2	1414	868	1.0	0.6	0.49
c2670	1246	80407	26095	20.9	32.5	1377	1253	1.0	1.6	0.70
c3540	1629	92617	30785	18.9	33.2	3431	1636	1.0	1.8	1.05
c5315	2278	129560	70394	30.9	54.3	2449	2312	1.0	1.8	1.19
c7552	3434	180045	87822	25.6	48.8	1872	3483	1.0	1.9	3.14
c6288	2384	760550	N/A	N/A	N/A	>3 hr	2384	1.0	0.3	6.35
s35932	7491	151204	N/A	N/A	N/A	N/A	8364	1.1	5.5	2.40
s38417	33418	987587	N/A	N/A	N/A	N/A	93593	2.8	9.5	23.91
s38584	18664	147952	N/A	N/A	N/A	N/A	30238	1.6	20.4	3.71

In Table 13, we show the distribution of the path set size for all fault sites in three largest circuits. The distribution shows that, for most of fault sites in circuit s35932, no more than 3 paths must be tested. In a worse case, circuit s38584, no more than 5 paths must be tested for 90% of the fault sites. In the worst case, circuit s38417, the number is 8.

TABLE 13. Path set size distribution for all fault sites in three largest circuits.

Set size / circuit	1	2	3	4	5	6	7	8	> 8
s35932	95.6	2.5	1.8	0.0	0.0	0.0	0.0	0.0	0.0
s38417	36.0	10.9	13.0	6.2	7.0	5.5	6.9	4.5	10.0
s38584	44.1	26.6	14.1	5.1	1.8	0.4	1.8	0.6	5.5

The path selection distribution in circuit s38417 is shown in Figure 50, where the X-axis indicates the path index, and the Y-axis indicates the percentage of local fault sites where the indexed path is selected. The path with index 0 is the longest under the nominal process condition and is selected for all fault sites. With increasing path index, the percentage selected decreases, as a path with lower nominal delay is less likely to be longest. The distribution also shows that, most of the longest paths are selected within the first batch of 50 paths, while no paths are selected in the second batch. For most fault sites, the path selection procedure stops at the second path batch.

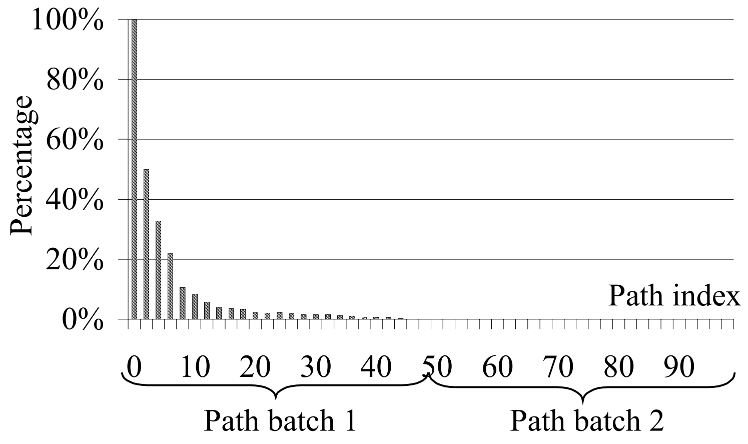


Figure 50. Path distribution vs. path indexes in s38417 using the new method.

To compare the efficiency between the two methods, in Figure 51and Figure 52 we show the path selection distribution in circuit c432 using the min-max method and the new method respectively. As shown in Figure 34, using the min-max method, the distribution goes to 0 after more than 150 paths, while for the new method the distribution goes to 0 after only 15 paths in Figure 52.

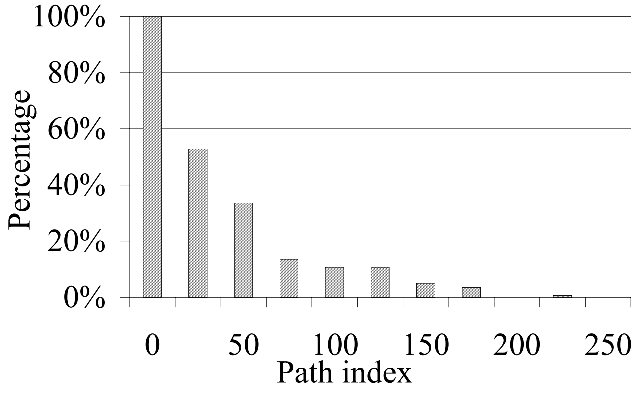


Figure 51. Path distribution vs. path indexes in c432 using the min-max method.

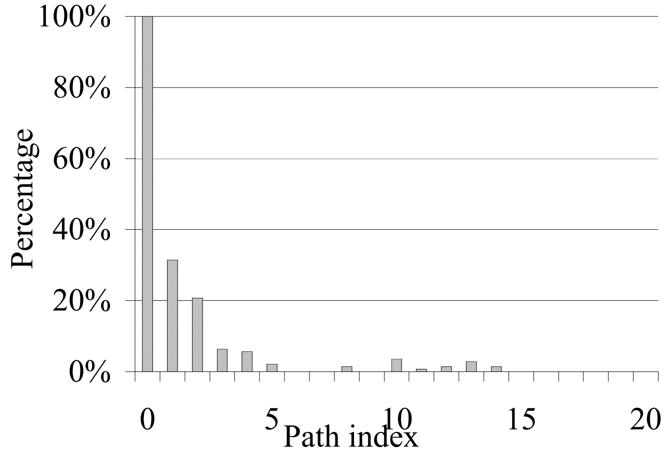


Figure 52. Path distribution vs. path indexes in c432 using the new method.

VI. SUMMARY AND CONCLUSIONS

In this dissertation, we study three challenging issues for delay test in nano-scale VLSI circuits: fault modeling of resistive spot defects, variational delay evaluation, and path selection under process variation. We present our new solutions and show the improvement in experimental results.

The electrical behaviors of resistive spot defects are comprehensively analyzed. The defect is modeled as a functional fault or a delay fault according to the input signal patterns and the resistance. We derived close-form expressions for the relationship between the delay change and the resistance. Based on the fault model, we are able to numerically compare the performance of different input vectors and choose the one to improve the fault coverage. The fault model is combined into a circuit-level fault simulator and results show the benefits of the delay test over the logic test.

To fast compute effects of process variation on circuit delays, we propose a linear delay model that incorporates the effect of process variations into a linear function. A fast parametric delay evaluation method PARARDE is presented to compute coefficients of the linear function. Our method avoids multiple parasitic extractions and multiple delay evaluations as did in the traditional RSM, and result in a significant speedup. The method based on effective capacitance delay model achieves higher accuracy. Experiments on ISCAS85 circuits show that our methods are effective and accurate for the parametric delay evaluation under process variation. And our new estimation method for capacitance sensitivity computation is applicable for any commercial parasitic extraction tools.

On path selection for delay test under process variation, we present a novel and efficient method to find the set of longest paths. For the first time, we consider both path structural correlation and process correlation, and consider process variation in both devices and interconnect. Two heuristics are proposed to prune redundant paths and insignificant paths. Experimental results show the heuristics are very efficient and effective. Our method can significantly reduce the number of paths and test patterns for delay test, compared with the previous best method. Experiments on ISCAS circuits show that the new method reduces the number of paths for test to 1%-6% of the results using the min-max method [36], without decreasing the fault coverage in delay test. The significant reduction indicates that considering both structural correlation and process correlation is much more effective than considering path structural correlation alone. In addition, the new method runs 300-3000 times faster than the min-max method, mainly because the min-max method examines far more paths.

The work described above only considers die-to-die process variation. Systematic within-die variation, such as computed by lithography simulation tools, can be incorporated into the delay model, as it only affects the delay equation coefficients. Random within-die variation will be incorporated into the model in the future. This requires the addition of more process variables and a spatial correlation structure. The lower path correlation will result in more paths selected for testing. But these test sets will still be significantly smaller than the min-max test sets, which assume only structural correlation between path delays.

REFERENCES

[1] "2002 Update Tables", *International Technology Roadmap for Semiconductors, 2002 Update*, Semiconductor Industries Association: San Jose, CA, 2002, pp. 16-150.

[2] C. F. Hawkins, J. M. Soden, A. W. Righter, F. J. Ferguson, "Defect Classes – An Overdue Paradigm for CMOS IC Testing", in *Proc. IEEE International Test Conference*, Washington DC, Oct. 1994, pp. 413-425.

[3] P. Banerjee and J. A. Abraham, "Fault Characterization of VLSI MOS Circuits", in *Proc. IEEE International Conference on Circuits and Computers*, New York, Sept. 1982, pp. 564-568.

[4] P. Banerjee and J. Abraham, "Generating Tests for Physical Failures in MOS Logic Circuits", in *Proc. IEEE International Test Conference*, Philadelphia, Oct. 1983, pp. 554-559.

[5] H. Hao and E. J. McCluskey, "Resistive Shorts Within CMOS Gates", in *Proc. IEEE International Test Conference*, Nashville, Oct. 1991, pp. 292-301.

[6] M. Renovell, P. Huc, and Y. Bertrand, "The Concept of Resistance Interval: A New Parametric Model for Realistic Resistive Bridging Fault", in *Proc. VLSI Test Symposium*, Princeton, NJ, Apr. 1995, pp. 184-189.

[7] R. Degraeve, B.Kaczer, A. de Keersgieter, G. Groeseneken, "Relation Between Breakdown Mode and Location in Short-channel NMOSFETs and Its Impact on Reliability Specifications", *IEEE Trans. Dev. Mat. Rel.*, vol. 1, no.3, pp.163-169, Sept. 2001.

[8] D. Gaitonde and D. M. H. Walker, "Circuit-level Modeling of Spot Defects", in *Proc. IEEE International Workshop on Defect and Fault Tolerance in VLSI Systems*, Pittsburgh, Nov. 1991, pp. 63-66.

[9] J. Segura, C. D. Benito, A. Bubio and C. F. Hawkins, "A Detailed Analysis of GOS Defects in MOS Transistors: Testing Implications at Circuit Level", in *Proc. IEEE International Test Conference*, Washington DC, Oct. 1995, pp. 544-551.

[10] H. Hao and E. J. McCluskey, “On the Modeling and Testing of Gate Oxide Shorts in CMOS Logic Gates”, in *Proc. IEEE International Workshop on Defect and Fault Tolerance in VLSI Systems*, Pittsburgh, Nov. 1991, pp. 161-174.

[11] V. R. Sar-Dessal and D. M. H. Walker, “Resistive Bridge Fault Modeling, Simulation and Test Generation”, in *Proc. IEEE International Test Conference*, Atlantic City, NJ, Sept. 1999, pp. 596-605.

[12] W. Moore, G. Gronthoud, K. Baker and M. Lousberg, “Delay-fault Testing and Defects in Sub-micron ICs – Does Critical Resistance Really Mean Anything?”, in *Proc. IEEE International Test Conference*, Atlantic City, NJ, Oct. 2000, pp. 95-104.

[13] W. Chuang and I N.Hajj, “Fast Mixed-mode Simulation for Accurate MOS Bridging Fault Detection”, in *Proc. IEEE International Symposium on Circuits and Systems*, Chicago, May 1993, pp. 1503-1506.

[14] D. Shaw, D. Al-Khalili and C. Rozon, “Accurate CMOS Bridge Fault Modeling with Neural Network-based VHDL Saboteurs”, in *Proc. International Conference on Computer-Aided Design*, San Jose, CA, Nov. 2001, pp. 531-536.

[15] Z. Li, X. Lu, W. Qiu, W. Shi and D. M. H. Walker, “A Circuit Level Fault Model for Resistive Opens and Bridges”, *ACM Trans. on Design Automation of Electronic Systems*, vol. 8, no. 4, pp. 546-559, 2003.

[16] S. R. Nassif, “Modeling and Analysis of Manufacturing Variations”, in *Proc. IEEE Custom Integrated Circuits Conference*, San Diego, CA, May 2001, pp. 223-228.

[17] Y. Liu, S. R. Nassif, L. T. Pileggi and A. J. Strojwas, “Impact of Interconnect Variations on the Clock Skew of A Gigahertz Microprocessor”, in *Proc. ACM/IEEE Design Automation Conference*, Los Angeles, Jun. 2000, pp. 168-171.

[18] V. Mehrotra, S. L. Sam, D. Boning, A. Chandrakasan, R. Vallishayee and S. Nassif, “A Methodology for Modeling the Effects of Systematic Within-die Interconnect and Device Variation on Circuit Performance”, in *Proc. ACM/IEEE Design Automation Conference*, Los Angeles, Jun. 2000, pp. 172-175.

[19] E. Malavasi, S. Zanella, C. Min J. Uschersohn, M. Misheloff and C. Guardiani, "Impact Analysis of Process Variability on Clock Skew", in *Proc. International Symposium on Quality Electronic Design*, San Jose, CA, Mar. 2002, pp. 129-132.

[20] R. B. Brawhear, N. Menezes, C. Oh, L. T. Pillage and M. R. Mercer, "Predicting Circuit Performance Using Circuit-level Statistical Timing Analysis", in *Proc. European Conference on Design Automation*, Paris, Mar. 1994, pp. 332-337.

[21] H. Chang and S. S. Sapatnekar, "Statistical Timing Analysis Considering Spatial Correlations Using a Single PERT-like Traversal", in *Proc. International Conference on Computer-Aided Design*, San Jose, CA, Nov. 2003, pp. 621-625.

[22] A. Agarwal, D. Blaauw and V. Zolotov, "Statistical Timing Analysis for Intra-die Process Variations with Spatial Correlations", in *Proc. International Conference on Computer-Aided Design*, San Jose, CA, Nov. 2003, pp. 271-276.

[23] M. Orshansky, L. Milor, P. Chen, K. Keutzer and C. Hu, "Impact of Systematic Spatial Intra-chip Gate Length Variability on Performance of High-speed Digital Circuits", in *Proc. International Conference on Computer-Aided Design*, San Jose, CA, Nov. 2000, pp. 62-67.

[24] E. Acar, S. N. Nassif, L. Ying and L. T. Pileggi, "Assessment of True Worst Case Circuit Performance Under Interconnect Parameter Variations", in *Proc. International Symposium on Quality Electronic Design*, San Jose, CA, Mar. 2001, pp. 431-436.

[25] A. Gattiker, S. Nassif, R. Dinakar and C. Long, "Timing Yield Estimation from Static Timing Analysis", in *Proc. International Symposium on Quality Electronic Design*, San Jose, CA, Mar. 2001, pp. 437-442.

[26] S. Borkar, T. Kamik, S. Narendra, J. Tschanz, A. Keshavarzi and V. De, "Parameter Variations and Impact on Circuits and Microarchitecture", in *Proc. ACM/IEEE Design Automation Conference*, Chicago, Jun. 2003, pp. 338-342.

[27] G. M. Luong and D. M. H. Walker, "Test Generation for Global Delay Faults", in *Proc. International Test Conference*, Washington DC, Oct. 1996, pp. 433-442.

[28] J. J. Liou, A. Krstic, L. C. Wang and K. T. Cheng, “False-path-aware Statistical Timing Analysis and Efficient Path Selection for Delay Testing and Timing Validation”, in *Proc. ACM/IEEE Design Automation Conference*, New Orleans, Jun. 2002, pp. 566-569.

[29] A. Krstic, L. C. Wang, K. T. Cheng and J. J. Liou, “Diagnosis of Delay Defects Using Statistical Timing Models”, in *Proc. IEEE VLSI Test Symposium*, Napa, CA, Apr. 2003, pp. 339-344.

[30] X. Lu, Z. Li, W. Qiu, D. M. H. Walker and W. Shi, “Longest Path Selection for Delay Test Under Process Variation”, in *Proc. IEEE Asia South Pacific Design Automation Conference*, Yokohama, Japan, Jan. 2004, pp. 98-103.

[31] A. D. Fabbro, B. Franzini, L. Croce and C. Guardiani, “An Assigned Probability Technique to Derive Realistic Worst-case Timing Models of Digital Standard Cells”, in *Proc. ACM/IEEE Design Automation Conference*, San Francisco, Jun. 1995, pp. 702-706.

[32] R. R. Montanes, J. P. de Gyvez and P. Volf, “Resistance Characterization for Weak Open Defects”, *IEEE Design and Test of Computers*, vol. 19, no. 5, pp. 18-25, Sept. 2002.

[33] W. N. Li, S. M. Reddy and S. K. Sahni, “On Path Selection in Combinational Logic Circuits”, *IEEE Trans. Computer-Aided Design*, vol. 8, no. 1, pp. 56-63, Jan. 1989.

[34] Y. Shao, S. M. Reddy, I. Pomeranz and S. Kajihara, “On Selecting Testable Paths in Scan Designs”, in *Proc. IEEE European Test Workshop*, Corfu, Greece, May 2002, pp. 53-58.

[35] M. Sharma and J. H. Patel, “Finding a Small Set of Longest Testable Paths that Cover Every Gate”, in *Proc. IEEE International Test Conference*, Baltimore, Oct. 2002, pp. 974-982.

[36] M. Sivaraman and A. J. Strojwas, “Path Delay Fault Diagnosis and Coverage – A Metric and an Estimation Technique”, *IEEE Trans. Computer-Aided Design*, vol. 20, no. 3, pp. 440-457, Mar. 2001.

[37] T. W. Williams, B. Underwood and M. R. Mercer, "The Interdependence Between Delay Optimization of Synthesized Networks and Testing", in *Proc. ACM/IEEE Design Automation Conference*, San Francisco, Jun. 1991, pp. 87-92.

[38] S. Tani, M. Teramoto, T. Fukazawa and K. Matsuhiro, "Efficient Path Selection for Delay Testing Based on Partial Path Evaluation", in *Proc. IEEE VLSI Test Symposium*, Princeton, NJ, Apr. 1998, pp. 188-193.

[39] B. Stine, D. Boning and J. Chung, "Analysis and Decomposition of Spatial Variation in Integrated Circuit Process and Devices", *IEEE Trans. on Semiconductor Manufacturing*, vol. 10, no. 1, pp. 24-41, 1997.

[40] L. Pillage and R. Rohrer, "Asymptotic Waveform Evaluation for Timing Analysis", *IEEE Trans. Computer-Aided Design*, vol. 9, no. 4, pp. 352-366, Apr. 1990.

[41] P. R. O'Brien and T. L. Savarino, "Modeling the Driving-point Characteristic of Resistive Interconnect for Accurate Delay Estimation", in *Proc. International Conference on Computer-Aided Design*, Santa Clara, CA, Nov. 1989, pp. 512-515.

[42] S. Irajpour, S. Nazarian, L. Wang, S. K. Gupta and M. A. Breuer "Analyzing Crosstalk in the Presence of Weak Bridge Defects", in *Proc. VLSI Test Symposium*, Napa, CA, Apr. 2003, pp. 385-393.

[43] N. H. E. Weste and K. Eshraghiaghian, "MOS Transistro Theory" in *Principles of CMOS VLSI Design—A Systems Perspective*, 2nd edition, Addison-Wesley Publishing Company: Boston,1992, pp. 61-63..

[44] W. Qiu, X. Lu, Z. Li, D. M. H. Walker and W. Shi, "CodSim: A Combined Delay Fault Simulator", in *Proc. International Symposium on Defect and Fault Tolerance in VLSI Systems*, Boston, Nov. 2003, pp. 79-88.

[45] M. Spica, M. Tripp and R. Roeder, "A New Understanding of Bridge Defect Resistances and Process Interactions from Correlating Inductive Fault Analysis Predictions to Empirical Test Results", in *Proc. International Workshop on Defect Based Testing*, Los Angeles, Apr. 2001, pp. 11-16.

[46] J. Qian, S. Pullela and L. Pillage, "Modeling the 'Effective Capacitance' for the RC Interconnect of CMOS Gates", *IEEE Trans. on Computer-Aided Design*, vol. 13, no. 12, pp. 1526-1535, 2001.

[47] K. A. Bowman, B. L. Austin, J. C. Eble, X. Tang, and J. D. Meindl, "A Physical Alpha-Power Law MOSFET Model", *IEEE Journal of Solid-State Circuits*, vol. 34, no. 10, pp. 1410-1414, 1999.

[48] G. L. Smith, "Model the Delay Faults Based Upon Path", in *Proc. IEEE International Test Conference*, Philadelphia, Nov. 1985, pp. 342-349.

[49] W. Qiu and D. M. H. Walker, "An Efficient Algortihm for Finding the K Longest Testable Paths Through Each Gate in a Combinational Circuit", in *Proc. IEEE International Test Conference*, Charlotte, NC, Oct. 2003, pp. 592-601.

[50] W. Qiu, J. Wang, D. M. H. Walker, D. Reddy, X. Lu, Z. Li, W. Shi and H. Balachandran, "K Longest Paths per Gate (KLPG) Test Generation for Scan-Based Sequential Circuits", in *Proc. IEEE International Test Conference*, Charlotte, NC, Oct. 2004, pp. 223-231.

[51] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran", in *Proc. IEEE International Symposium on Circuits and Systems*, Newport Beach, CA, Jun. 1985, pp. 663-698.

[52] D. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits", in *Proc. IEEE International Symposium on Circuits and Systems*, Portland, OR, May1989, pp. 1929-1934,

[53] D. Blaauw, V. Zolotov and S. Sundareswaran, "Slope Propagation in Static Timing Analysis", *IEEE Trans. on Computer-Aided Design*, vol. 21, no. 10, pp. 1180-1195, 2002.

[54] A. B. Kahng and S. Muddu, "Improved Effective Capacitance Computations for Use in Logic and Layout Optimizations", in *Proc. International Conference on VLSI Design*, Goa, India, Jan. 1999, pp. 578-582.

[55] C. V. Kashyap, C. J. Alpert and A. Devgan, “An ‘Effective’ Capacitance Based Delay Metric for RC Interconnect”, in *Proc. International Conference on Computer-Aided Design*, San Jose, CA, Nov. 2000, pp. 229-235.

[56] N. Megiddo, “Linear Programming in Linear Time When the Dimension Is Fixed”, *J. ACM*, vol. 31, no. 1, pp. 114-127, Jan. 1984.

VITA

Xiang Lu was born in Shi Yan City, Hu Bei, China. He completed his Bachelor and Master’s degree at Xi’an Jiaotong University, Xi’an, China in July 1997 and June 2000, respectively. He then attended Texas A&M University, College Station, TX as a graduate student in Computer Engineering and graduated with a Ph.D. degree in December 2005. His research interests are process variation effects on nano-scale VLSI circuits, delay test under process variation, and static/statistical timing analysis. Now he is working with P. A. Semi, Inc., a chip design company in Santa Clara, CA. He can be reached by email at lu.shawn@gmail.com, or by mail care of Dr. Weiping Shi, Dept. of Electrical Engineering, Texas A&M University, College Station, TX 77843.