Testing Signal Integrity Faults in VLSI Circuits

ZHANG, Yubin

A Thesis Submitted in Partial Fulfilment

of the Requirements for the Degree of

Doctor of Philosophy

in

Computer Science and Engineering

The Chinese University of Hong Kong

June 2011

Abstract

As the ever-advancing fabrication technologies in semiconductor industry enable the VLSI circuits with increasing integration and decreasing cost, the circuits suffer from much severer Signal Integrity (SI) faults, where SI is the capability of signals generating correct responses in their downstream circuits. SI faults are complex problems to tackle since SI may be damaged by numerous kinds of causes and SI faults may impact multiple aspects of circuits' performance. Such SI problems can seriously reduce product yield, result in function error or even permanently damage the chip. Therefore, effective testing methodologies are essential to alleviate SI problems by verifying the SI satisfaction of VLSI circuits efficiently.

Hereby the thesis has examined the SI problems systematically and proposed effective test methods corresponding to the specific feature of SI faults. Firstly, considering that SI on inter-core interconnects of SOCs is under severe danger, new test wrapper design has been proposed to achieve accurate SI test on interconnects. Secondly, test architecture has been optimized for cost reduction considering SI test and logic test simultaneously. Thirdly, the impact of power distribution network (PDN) defects on SI has been analyzed and efficient computation method has been proposed to identify those potentially harmful PDN defects. Effective test pattern manipulation method has also been proposed to improve test coverage of PDN defects. Fourthly, considering the increasing impact of process variation and aging effect on SI, an innovative online test architecture has been proposed, which can accurately measure the delay of critical paths when the circuit is working in function mode, where such valuable information is of great help for a variety of applications.

摘要

不斷進步的半導體製造工藝使得超大型積體電路的集成度不斷提高而單位成本不斷降低，然而電路同時卻遭受越來越嚴重的信號完整性錯誤，在此信號完整性指的是信號在其下游電路中激發正確反應的能力 因為多種因素皆可能導致信號完整性錯誤以及電路的多種性能皆會受到其影響，所以信號完整性錯誤是難以處理的問題 信號完整性錯誤會顯著地降低成品率、導致功能錯誤甚至永久地損壞晶片，所以有效的測試方案成為必須

此論文系統地研究了信號完整性問題，並針對其特性而提出了相應的高效解決方案 首先考慮到長連接線上的信號完整性遭受嚴重的危險，提出了新的測試包裝設計從而達到精確的測試結果；其二，提出了晶片測試架構的優化方案，可以顯著地降低片上系統中片內及片間測試的總體成本；其三，分析了電源傳輸網路缺陷對信號完整性的影響，提出了高效的演算法以分辨出有害缺陷，並設計了測試向量選擇方法以提高測試覆蓋率；其四，針對日益嚴重的電路不確定性及老化效應所導致的信號完整性錯誤，提出了創新性的線上測試架構，能夠在電路工作時即時測試關鍵路徑的延遲，所得到的寶貴資訊對廣泛的應用皆有重大幫助

Acknowledgement

At the very beginning, I am deeply indebted to my supervisor, Professor Qiang Xu, who patiently motivated me to conceive and develop the main ideas in the thesis. I would like to express to him my sincere gratitude for his seasoned guidance from the very early stage of this research work as well as providing constructive advices throughout the entire study.

My research partners Feng Yuan, Lin Huang, Xiao Liu, Li Jiang and Rong Ye in the The CUhk REliable Computing Laboratory (CURE), thank you for your insightful comments on my research work. I am also grateful to all the colleagues in the EDA lab located at HSB 506, Linfu Xiao, Xiaoqing Yang, Lin zhou, Liang Li, Minqi Jiang, Zaichen Qian, and Yan Jiang, it is you who bring me laugher and make my post-graduate study life colorful.

Last but not the least, all my family members, without your love and support, I cannot achieve anything. I would like to give my greatest appreciation to you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction to Signal Integrity Testing

The preliminaries of Signal Integrity (SI) testing are firstly introduced in this chapter, including the symptom and causes of SI faults, the prior works and the new challenges in this domain, which motivates the thesis.

1.1 Signal Integrity Problems

As the size of VLSI circuit devices keeps shrinking while the working frequency keeps increasing for today's Integrated Circuit (IC) chips, signal integrity (SI), i.e., the capability of a signal to generate correct responses in its downstream circuit [34], is becoming a major concern [39]. SI problems, typically caused by parasitic capacitance and inductance between interconnects, include overshoots, undershoots, glitches, oscillations, excessive signal delay, and even signal speedup [40], which is demonstrated in Figure 1.1. If the noise-induced voltage swing and timing skew depart from the noise-immune region, functional error may occur.

Traditionally, SI problems have been treated as design errors, and a number of

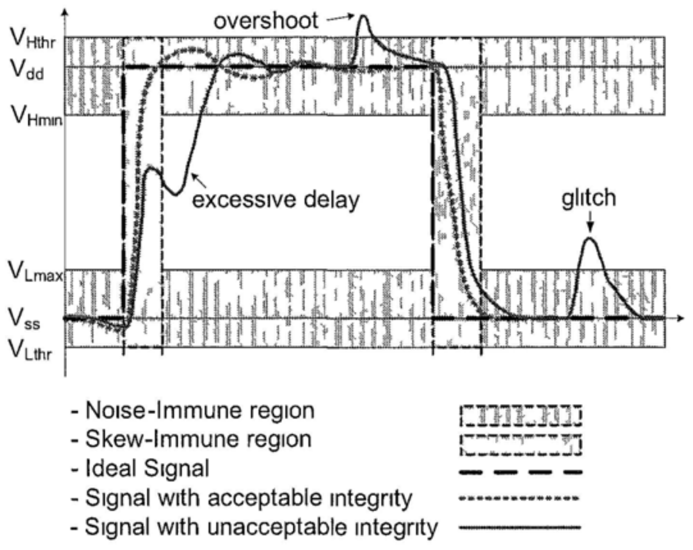


Figure 1 1 Multiple types of signal integrity problems

physical design and fabrication solutions, e g , [11, 19, 53, 93], have been proposed in the literature to tackle them The above design techniques rely on accurate simulation of SI effects, which are affected by many parameters (e g , characteristics of interconnects and transistors, input data and environmental noise) Unfortunately, these parameters are interdependent and the lack of complete knowledge of this interdependence leads to uncertainty and inaccuracies in the simulation of SI loss [85] Moreover, process variations and manufacturing defects may aggravate the SI-related problems [58] Since it is unacceptable to over-design the circuit to tolerate SI loss in all cases and it is impossible to predict the occurrence of defects, manufacturing test strategies are essential for detecting SI-related errors [21, 77, 80]

1.2 Causes of Signal Integrity Faults

The main reasons causing SI faults can be categorized into the following three domains: I) crosstalk between neighboring devices; II) defects on Power Distribution Network (PDN); III) Process variation and aging effect.

1.2.1 Crosstalk

Crosstalk, or coupling, between neighboring devices can significantly impact the SI status of each other, where both mutual capacitance and inductance between them can play the role. Especially for long wires neighboring with each other, they can demonstrate strong crosstalk because their mutual capacitance and inductance are considerably large. In the case of crosstalk, the device whose SI status has been changed is called victim while the devices that has changed the SI status of others is called aggressors.

Crosstalk induced SI faults can be clearly seen from the example shown in Figure 1.2. It demonstrates that the victim wire travels in such a path that it has five neighboring wires close to it, where the parasitic capacitance between the victim and its neighbors is designedly shown for clarity.

Figure 1.3 shows the corresponding waveform when the victim tries to making a rising transition "$0 \rightarrow 1$" while the aggressors transiting oppositely "$1 \rightarrow 0$". As it can be seen, the voltage level of the victim is forced to lower at first and rise in a slow rate, which results in much longer transition time compared to that without aggressors. In this case the crosstalk induces the SI problem of excessive delay.

1.2.2 Power Distribution Network Defects

The correct functioning of circuit devices relies on the accurate power supply that is delivered to every device on chip via the PDN. It is typical that 1% change

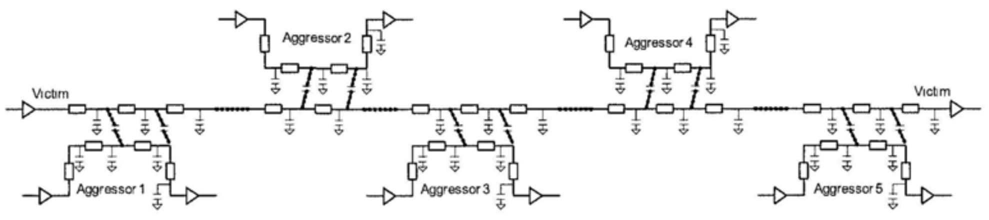


Figure 1.2: An example of crosstalk on a victim with five aggressors.

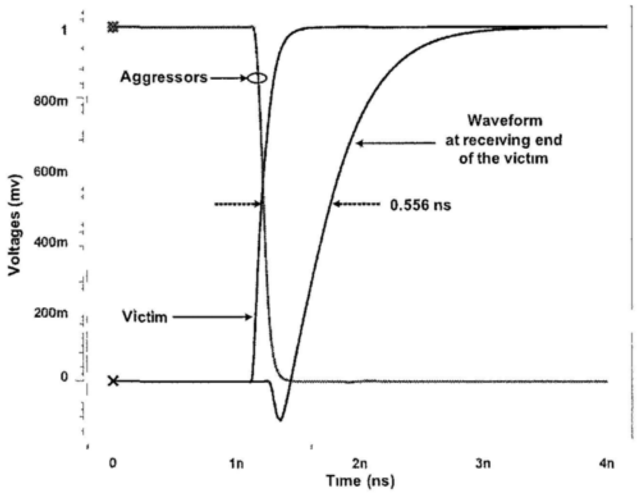


Figure 1.3: Timing diagram of a victim with aggressors transiting oppositely.

on the supply voltage can result in nearly 4% change in circuit delay with 90nm technology [82]. On the other hand, such PDN defects as open wire can change the topology of PDN significantly so that a part of devices cannot receive enough power supply. Consequently, such degraded power supply results in extra delay of circuit devices, which then demonstrates the SI problem of excessive delay.

An example of such SI problem resulting from PDN defect is illustrated in Figure 1.4. The open defect in Figure 1.4 forces the device of SC_3 drawing power supply from the remote Via_2 instead of the nearby Via_1. Such deviation from expectation brings on reduced power supply for SC_3 resulting from the excessively

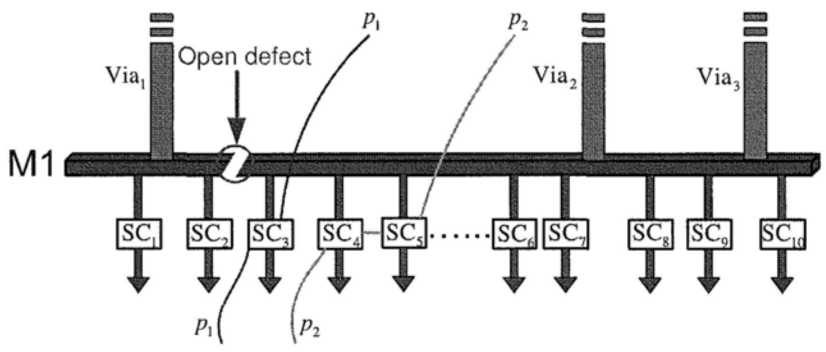


Figure 1.4: An example of PDN defect inducing SI faults.

large power drop along the longer path from Via_2. The SI problem of excessive delay then manifests itself on SC_3 corresponding to the reduced power supply, which can result in delay error on the path p_1 propagating through SC_3.

1.2.3 Process Variation and Aging Effect

Nowadays process variations have an increasing impact on the timing behavior of ICs, because the parameters of fabricated circuit devices can deviate significantly from the expectation in design stage, which makes it hard to guarantee the SI correctness. In addition, circuits fabricated with latest technology suffer from ever-increasing aging effects (e.g., Negative Bias Temperature Instability (NBTI)), which can gradually degrade the performance of circuits, e.g., extend the delay of circuit devices [12, 72]. Traditional test methods cannot handle such kind of SI problem because of uncertainty and gradual change, which highlights the need of online monitoring and testing with details discussed in Chapter 5.

1.3 Challenges in Signal Integrity Testing

As the SI problems keep worsening, new challenges emerge in VLSI testing, which demands new test methodologies that can take such challenges into account.

1.3.1 SI Problems on Long Interconnect Wires

Long interconnect wires travel a long way and may be close to a lot of devices in the VLSI circuits, which induces strong crosstalk among one another. Consequently, the crosstalk can significantly degrade the SI status and demonstrate such symptom as extra delay, voltage overshoot/undershoot/swing, etc. On the other hand, previously only short/open faults have been taken into account for interconnects testing, which raises the need for new Design For Testability (DFT) to facilitate arising SI test.

What is worse, the SI problems on interconnects displays two characteristic features that are different from traditional VLSI testing.

I) As can be seen from Figure 1.1, severe voltage overshoot/undershoot can result from strong crosstalk. From logic point of view, voltage overshoot/undershoot are still logically correct, but they can gradually or instantly damage the chip, which requires new test mechanism.

II) The crosstalk effect on victim induced by its aggressors is dependent on the time skew between the transition time of victim and that of aggressors. Prior test structures always launch the transition simultaneously for victim and aggressors in test mode, while they may transit at different time in function mode (details shown in Chapter 2), which raises new challenges for test structures.

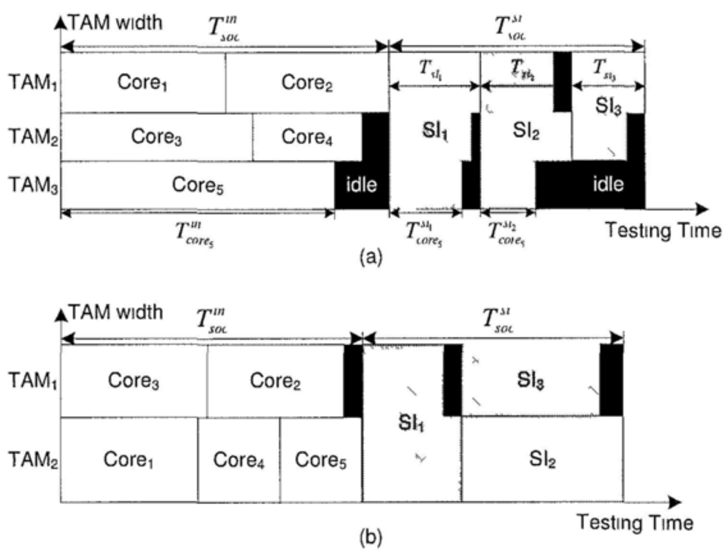


Figure 1.5: An example of SOC TAM design and test scheduling.

1.3.2 Test Architecture Optimization

As mentioned before, previously only short/open faults have been taken into account for interconnects test, which is easy to handle and costs low test effort. However, as the SI test for interconnects is a must and the corresponding test effort is comparable to traditional logic test, the previous test architectures, which are optimized for logic test only, are no longer efficient. Prohibitively high test time is needed if no effective SI test pattern compaction scheme is employed and the SOC test architecture is not optimized for both core-internal logic test and interconnect SI test.

An example showing the effect of different test architectures is demonstrated in Figure 1.5. There are five cores in this SOC and the total available Test Access Mechanism (TAM) width is fixed, while the grouping of cores and partitioning of TAM wires is different in part (a) & (b) of Figure 1.5. For example, the overall TAM wires have been partitioned into three groups in part (a) while two groups in part (b). The SI test in part (a) has been placed in three groups, where the SI_1 group involves all the five embedded cores, SI_2 group involves $Core_1$, $Core_4$ and $Core_5$, and SI_3 group involves $Core_2$ and $Core_3$. As can be seen from Figure 1.5, there is less idle time wasted in part (b) so that the overall test time of part (b) is less than that of (a), which demonstrates the effect of test architecture optimization on test cost reduction.

In addition to the high volume of SI test data, SI test involves multiple cores at a time, instead of one core only in traditional logic test, which makes the test architecture optimization more complex.

1.3.3 Analysis of Power Distribution Network Defects

PDN is typically designed with a hierarchical structure, which traverses multiple metal layers in the VLSI circuits. Figure 4.1 depicts a typical PDN structure, which involves three metal layers (M1, M4, and M5 from bottom up). The circuit devices are arranged below M1 layer in rows, and obtain power supply from power wires on M1 layer.

With the ever increasing circuit complexity, PDN designs for high-performance ICs typically occupy a considerable part of metal resources in the circuit [2]. Therefore, it is very likely that defects are introduced on PDN during the manufacturing process [47, 76]. The total number of possible PDN defect locations is enormous and it is impossible to generate test for every one of them. On the other hand, typical PDN is essentially a defect-tolerant structure and a large number of PDN defects are in fact harmless.

For those harmful PDN defects that cannot be tolerated, designers mainly rely on test patterns for stuck-at faults and delay faults to identify them implicitly. While the above implicit test strategy works quite well for relatively simple circuits previously, with increasing circuit complexity, PDN defects may cause substantial

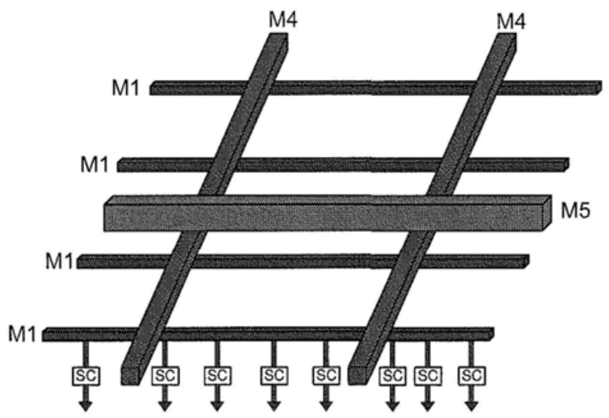


Figure 1.6: An example power distribution network.

timing errors for high-performance ICs, which cannot be detected by regular delay tests. This is because: (i). the power supply noise margin of IC devices gets smaller and smaller with decreasing supply voltage in each new technology generation [14, 23], and circuit performance suffers to more extent from the power supply degradation [82]; (ii). the worst-case power supply degradation caused by PDN defects occurs when the local switching activities surrounding the defective position are maximized, which leads to severe power droop [6, 7, 79]; On the other hand, conventional test patterns that target for delay faults try to sensitize the logic paths so that the effect of target delay error can be observed [46, 73], without considering to activate intensive transitions near the PDN defect locations.

It is therefore essential to identify those potentially harmful defects on the PDN and apply test patterns to detect them, ensuring the quality of the shipped products. Due to the huge number of possible PDN defect locations, this is an interesting and challenging problem.

1.3.4 Uncertainty and Aging effect Worsening SI

Nowadays advanced fabrication technology results in the timing performance of VLSI circuits suffering increasing disturbance from such uncertainty as process variation. Consequently, it is increasingly difficult to ensure circuits' timing correctness solely by off-line manufacturing test. What is worse, circuits also suffer from increasing aging effects, gradually reducing their performance. While there have been some attempts to conduct on-chip delay measurement to tackle the above problems (e.g., [64, 86, 87]), they require a dedicated test mode. Such non-concurrent solutions are hence inherently inaccurate due to the discrepancy between circuits' timing behavior in functional mode and that in test mode.

The requirements on test accuracy and long term effectiveness challenge the traditional test methods and demand the online counterpart, which can generate the test result corresponding to the circuits's performance in function mode. However, multiple difficulties arise for this online task, where main of them are listed as follows.

I) Considering the online test is performed as the circuits working in function mode, there should be negligible disturbance introduced into the circuits' performance by the online test task.

II) Since uncertainty can also happen on the test circuit itself, most of the impact induced by the uncertainty must be removed. Otherwise, the test accuracy will be much degraded.

III) The online test architecture must be with acceptable overhead, including low hardware overhead and short response time. Otherwise, the application of such online test architecture is significantly limited.

1.4 Contributions of the thesis

Hereby the thesis examines the specific features of SI problems and their arising challenges induced to VLSI circuit testing, and correspondingly proposes efficient solutions. The main contributions of the thesis are summarized as follows.

I) Because the inter-core interconnect wires of SOC are under serious SI problem while the accuracy of previous test methods is far from enough, new SI detector and test wrapper designs have then been proposed to achieve accurate SI test on the basis of examining two important factors that disturb the SI status on interconnects.

II) It is noticed that, considering the test architecture design of SOC, attention has been previously paid to core-internal logic test. However, nowadays the SI test is with increasing test cost and must be taken into account in test architecture design in order to reduce the overall test cost of IC chips. Therefore, test architecture design is optimized here for efficient SOC testing taking both SI test and logic test into account simultaneously.

III) Considering the high complexity in analyzing the impact of PDN defects on SI, efficient computation algorithm has been motivated so that potentially harmful PDN defects can be efficiently identified from high volume of candidates, which is of great help for further PDN testing and strengthening. Correspondingly, test pattern manipulation method is proposed to improve the test coverage of those identified harmful PDN defect.

IV) For the increasing difficulty in tackling uncertainty and aging effect induced SI problems, accurate online test methodology is of great need, which can not only perform accurate SI test in function mode but also continuously monitor the gradual change of SI status resulting from aging effect. Correspondingly, an innovative concurrent online delay measurement architecture is proposed here, which can accurately measure the delay of critical paths when the circuit is working in function mode. Such valuable information is crucial for SI testing and a variety of important applications.

In a word, this thesis has been motivated to examine the important factors that impact the SI status and to propose efficient test methodologies achieving high accuracy of SI test.

□ **End of chapter.**

Chapter 2

Detect SI Faults on Interconnects

To effectively test SI faults on core-external interconnects, core test wrappers need to be able to generate appropriate transitions at a wrapper output cell (WOC) on the driving side and detect the signal integrity loss at a wrapper input cell on the receiving side. In current wrapper designs, the WOCs for a victim interconnect and its aggressors make transitions at the same time with a common test clock signal in test mode, which is different from the functional mode. This is not adequate for SI test because the time elapsed between the transition of the victim and the transitions of its aggressors significantly affects the behavior of SI-related errors. To address this problem, this chapter proposes new IEEE Std. 1500-compliant wrapper designs that are able to apply SI test at functional mode or make transitions with various pre-defined skews between a victim line and its aggressors. A novel overshoot detector inside the proposed wrapper has also been introduced. Experimental results show that the proposed wrapper designs are more effective for detecting SI-related errors when compared to existing techniques, with a moderate amount of DFT overhead.

2.1 Introduction

Due to continuing advances in VLSI technology, integrated circuits (ICs) nowadays integrate hundreds of millions of transistors, and they can operate at Gigahertz frequencies. However, test problems are greatly exacerbated by the increasing complexity of ICs. For example, signal-integrity (SI) loss due to cross-coupling capacitance and inductance among interconnects, IR drop, ground bounce, and environmental variations may lead to overshoot, undershoot, glitches, ringing, inter-symbol interference, excessive signal delay or even signal speedup. These SI-related problems are aggravated in core-based system-on-a-chip (SOC) designs [56, 60], as interconnects carrying signals between embedded cores tend to be long and hence suffer more from parasitic effects. In addition, due to the complexity of the SOC interconnect topology and close proximity of long interconnects between multiple cores, signal integrity loss may involve several cores at the same time in core-based SOCs. If the noise-induced voltage swing and/or timing skew depart from the immune region, functional error may occur. A number of physical design and fabrication solutions (e.g., [11, 19]) have been proposed in the literature to tackle signal integrity problems during design and manufacture, but none of them guarantee that the problem can be completely solved. In addition, process variations and manufacturing defects aggravate the coupling problem between interconnects and render the above techniques less effective [8]. Since it is unacceptable to over-design the circuit to tolerate all possible process variations and it is impossible to predict the occurrence of physical defects, manufacturing test strategies are essential for the detection of SI-related errors [21, 77, 80].

Some SI-related errors can be detected by applying functional patterns at rated speed, but such a functional test strategy is often inadequate because it is dificult to determine appropriate functional test sequences that target all possible SI faults. The IEEE 1500 standard wrapper [35] can potentially support SI interconnect test if two-pattern at-speed signal transition capability is provided at the core test wrapper of the interconnect driving side (e.g., every wrapper output cell is equipped with two flip-flops), while at the same time integrity loss sensor (ILS) cells (e.g., [81]) are added at the wrapper of the interconnect receiving side. Unfortunately, such a design is unlikely to provide an acceptable quality level for interconnect SI test. Compared to the test for interconnect opens/shorts, SI test is very sensitive to how the test patterns are applied. The time between the signal transition of the victim lines and the transitions of its aggressor lines significantly affects the behavior of SI-related errors, as stressed in [60, 59]. Therefore, care must be taken during test application so that the dedicated SI test is applied as close to the functional mode of operation as possible. If this goal cannot be achieved, core-external SI test must be carried out to cover the worst-case scenario so as to ensure that low-quality parts are not shipped to customers.

For effective SI testing, we need to activate the worst-case crosstalk effect by aligning the switching times for all the involved interconnects, and address the worst-case effects on the victim line due to IR drop and process/environmental variations. To the best of our knowledge, prior work (e.g., [9, 16, 44, 81]) has not taken these issues into consideration. The approach employed in prior work is to simply let all involved core wrapper cells transit concurrently with the same test clock signal [17].

To address the above problem, we present two IEEE Std. 1500-compliant wrapper designs in this chapter. These wrappers are able to either apply the core-external interconnect SI test in functional mode, or address the worst-case SI loss scenario, respectively. Note that the proposed wrappers facilitate the test application scheme to be more effective for SI-related errors, which is independent of the test patterns provided by the SoC integrators.

While overshoots lead to functional errors if only if they occur during latching windows, their adverse impact on chip reliability is a major concern [42]. In this chapter, at the interconnect receiving side, we also design a new overshoot detector, which, compared to the one introduced in [62], is able to detect overshoots in all possible situations.

2.2 Related Prior Work

The *aggressor alignment problem* refers to the alignment of the switching times for multiple aggressors that results in the worst-case delay (WCD) and/or worst-case noise (WCN) on a victim interconnect. This problem has been well studied in recent years in the context of static timing analysis. It has been shown that the WCD or WCN on the victim line usually does not occur when all its aggressor lines make transitions at the same time with it, especially when there are timing-window constraints for these transitions [33, 17, 15]. Unfortunately, this problem has not been taken into account in manufacturing test solutions targeting core-external interconnects.

Built-in self-test (BIST) has been advocated to detect SI-related errors on core-external interconnects. At the driver side, test generators are embedded to generate transitions on the victim and its aggressors. Bai et al. [9] and Tehranipour et al. [81] introduced on-chip test generators based on the maximum aggressor (MA) fault model [21] and the multiple transition (MT) fault model, respectively. Li et al. [44] presented an oscillation-ring-based test scheme for SOC interconnects. None of the above techniques, however, considers the aggressor alignment issue. They apply all transitions with the same test-clock signal in test mode, which may be significantly different from the victim's functional behavior. As a result, the test outcomes can be misleading, and result in either test escapes or yield loss. Therefore, there is a need for wrapper designs that can target core-external interconnect SI test.

On the receiver side, various types of ILS cells have been proposed to detect SI-related errors. An XOR-network-based error-detector was described in [9]. However, this design is not area-efficient because it contains a local test generator to compare with test responses. Zhao et al. [95] presented an on-line testing technique that captures noise-induced logic failures by sampling the input data into two flip-flops during a given time interval and checking whether they are consistent with each other. Tehranipour et al. [80] proposed an ILS cell design to detect timing violations. However, these ILS cells cannot detect signal overshoots. Nourani and Attarha presented an ILS cell design to address this problem [62]. Their overshoot detector, however, cannot detect overshoots that occur in all situations, as described in Section 3.

The test architecture for interconnect SI faults should not invalidate the existing SOC test methodologies, i.e., it should be compatible with IEEE Std. 1500, which defines module-level test wrappers for embedded cores and allows inter-core and intra-core tests to be carried out via test access mechanisms that link the test source/sink (e.g., tester) with the test wrapper. The wrapper has three main modes [50]: (*i*) functional operation, in which the wrapper is transparent; (*ii*) an *inward-facing* test mode (also called *INTEST* mode), in which test access is provided to the core itself; and (*iii*) an *outward facing* test modes (also called *EXTEST* mode), in which test access is provided to the circuitry outside the core. Wrapper cells are introduced to the wrapper to provide controllability and observability for all core functional terminals. In the INTEST mode, used for testing the core's internal logic, the wrapper input cells (WICs) act as primary inputs to the core under test (CUT), while the wrapper output cells (WOCs) act as primary outputs. In the EXTEST mode, when all the embedded cores are wrapped, the goal is to test the interconnect wires or logic between the cores. Thus the wrapper output cells provide stimuli and the wrapper input cells capture responses from the interconnect

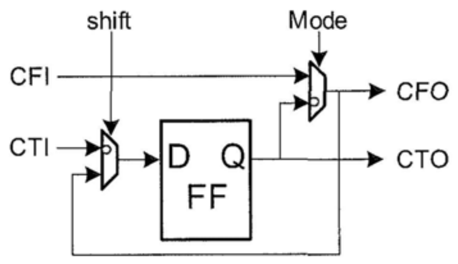


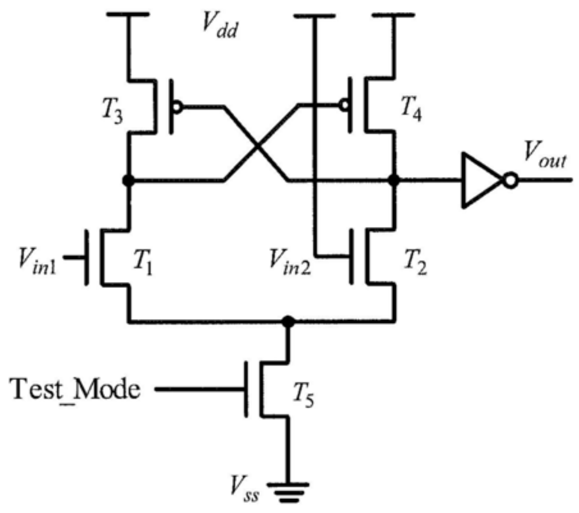
Figure 2.1: Regular wrapper cell design [50].

that are blocked in them and do not get propagated to the core's internal logic.

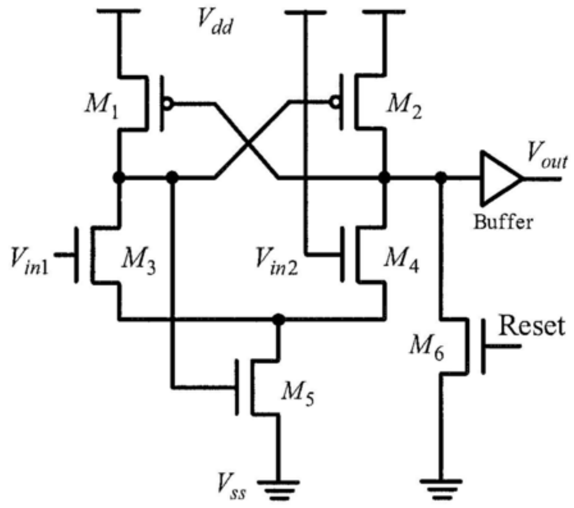
A typical wrapper cell implementation is shown in Figure 2.1, which has four standard terminals: the cell functional input (CFI), the cell functional output (CFO), the cell test input (CTI), and the cell test output (CTO) [35]. The meaning of the dark circle in each multiplexer is to indicate that the corresponding path will be selected when the control signal is '1'. Note that the IEEE Std. 1500 wrapper targets only the interface, i.e., the manner in which the core communicates with its surroundings in various modes of operation. Hence, the internal structure of an IEEE Std. 1500-compliant wrapper cell can be adapted to the specific SOC test requirements, e.g., the detection of SI errors.

2.3 Proposed Overshoot Detector

Overshoot, the physical phenomenon in which a signal exceeds V_{dd} momentarily, seldom results in any logic error to the circuit. A logic error is likely only if the overshoot occurs during a latching window. Repeated overshoots, however, are known to be able to cause hot-carrier damage in MOS transistors [38]. These hot-carriers might penetrate the gate oxide, leading to permanent changes in the oxide charge distribution, and creating serious reliability concerns for the circuit over time [18, 42]. It is therefore important to detect the occurrences of overshoots in all possible situations as part of the manufacturing test flow.



(a) Overshoot detector in [62]



(b) Proposed overshoot detector

Figure 2.2: Comparison of the proposed overshoot detectors with [62].

Figure 2.2 compares the overshoot detector presented in [62] with the one proposed in this chapter. As shown in Figure 2.2(a), the overshoot detector in [62] is composed of a cross-coupled differential amplifier (T_1–T_5) and an inverter (used to stabilize the output voltage). The input V_{in1} takes the signal from the victim interconnect and compares it with the other input V_{in2}, which is connected to power supply V_{dd}. Transistor T_5 that connects to the source terminal of transistors T_1 and T_2 serves as a current source to the differential amplifier. It is controlled by the SI test mode signal, so that it can be bypassed in other test modes. If the transistors are sized properly, the detector exhibits the hysteresis (Schmitt-trigger) property [68]. The output V_{out} takes logic value '0' when the voltage level of the input signal exceeds the pre-defined positive threshold voltage V_+ (say 1.1 V for a technology with supply voltage 1 V), and changes back to logic value '1' only when the voltage level of the input signal is lower than a pre-defined negative threshold voltage V_- (say 0.8 V for a technology with supply voltage 1 V). This temporary storage behavior is very useful for capturing overshoots that are of a short duration. The design in [62] is therefore very effective for detecting overshoots that occur when there is low-to-high transition for the input signal. However, due to hysteresis, once the output signal V_{out} detects an overshoot and stays at logic '0', it cannot detect the overshoots that happen when the input signal stays at logic '1' or has a high-to-low transition. This is unfortunate because overshoots may often occur in such situations.

To address the above problem, we introduce an extra transistor M_6 in our overshoot detector, as shown in Fig. 2.2(b). Whenever an SI error on the victim interconnect is captured, the *Reset* signal is asserted (controlled locally in the wrapper cell; see Figure 2.3) and the output signal is forced to return to the error-free state. Therefore, we are now able to detect overshoots that occur in all situations. In addition, we also modify the amplifier to be self-biased. The current source is provided by transistor M_5, whose gate terminal is controlled by a signal changing in the opposite direction of V_{out}, so that less current is supplied when V_{out} is logic '1', and larger current is supplied for logic '0' output. This self-biased amplifier needs no external signal for the current source and it has stronger feed-

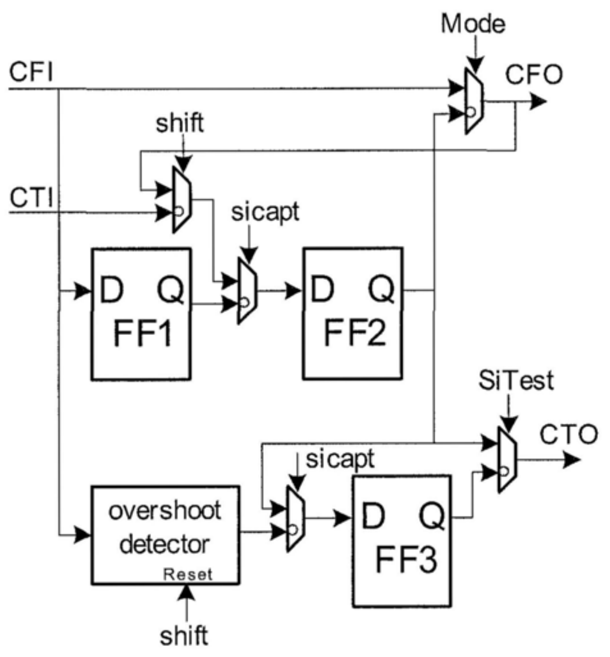


Figure 2.3: Wrapper input cell design with the proposed overshoot detector.

back; as a result, it has a larger voltage gain and higher resolution. Finally, the buffer in the detector enables the output signal to be at the standard voltage level, corresponding to the logic value, thereby improving the driving capability for the following stages.

Figure 2.3 shows the IEEE Std. 1500-compliant wrapper input cell, which has been enhanced with the proposed overshoot detector. To detect timing errors and other voltage violations out of the noise-immune region, we simply use a flip-flop (FF1 in Fig. 2.3) in our design, but it can be replaced by other integrity-loss sensors (e.g., [80, 95]). The proposed WIC functions as the standard WIC in [50] when signals "SiTest" and "sicapt" are de-asserted. When the wrapper is in core-external interconnect SI test mode (*SiTest* ='1'), during the capture phase (*sicapt* ='1'), the test response is captured in the detector and then saved in flip-flops FF2 and FF3. After the test responses are obtained, signal *sicapt* is de-asserted while signal

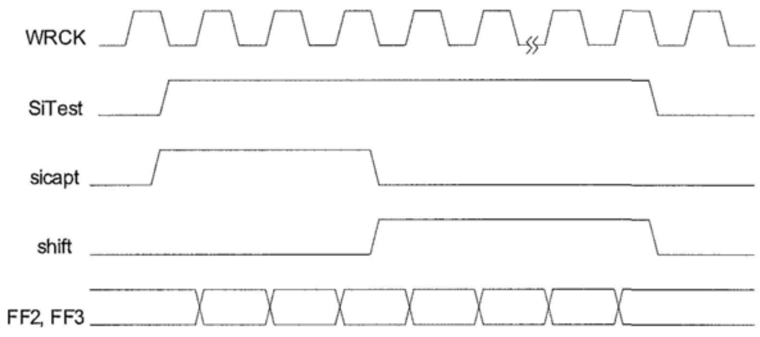


Figure 2.4: Timing diagram for the wrapper input cell in SI test mode.

shift is asserted, and the test responses are shifted out, as shown in the timing diagram of Figure 2.4.

2.4 Wrapper Designs for Core-External Interconnect SI Tests

As discussed in Section 2.2, the differences in transition times between a victim and its aggressors determine the magnitude and impact of the SI error. Previous work does not take this issue into consideration; the SI test is applied concurrently at all the involved wrapper cells with the same test-clock signal. This approach may result in over-testing or under-testing of the signal integrity loss on core-external interconnects. In this section, we present two methods and the associated wrapper designs to tackle this problem for different types of cores. We assume that the test stimuli are loaded from an external tester. BIST pattern generators are not considered in this work.

For interconnect SI tests involving only "soft cores", for which we have detailed structural information, we propose to apply the SI test patterns in normal functional mode during the capture phase. In other words, we load patterns into the wrapper input cells and the internal scan chains of the cores, and apply the patterns in functional mode in consecutive functional clock cycles to generate the test stimuli on the victim interconnect and its aggressors. There are two main advantages of this SI test strategy: (i) since the SI test is conducted in functional mode, it captures accurate SI-related errors and does not result in over-testing or under-testing; (ii) we can simply use a standard wrapper output cell as shown in Fig. 2.1 to apply SI test with small design-for-test (DFT) area overhead. A difficulty with this test application strategy however lies in the fact that we have to justify the test patterns on the core outputs through its inputs and internal memory elements, i.e., we need to run automatic test pattern generation (ATPG) to generate the *indirect* patterns to be loaded into the core's wrapper input cells and the internal scan chains, instead of directly loading patterns into the core's wrapper output cells and applying onto interconnects. This approach is not feasible for cores for which we do not have knowledge about their detailed internal structures (e.g., "hard cores"). Moreover, it requires relatively long test application time since we need to load the SI test patterns into core wrapper input cells and core internal scan chains, instead of directly loading them into the wrapper output cells. Finally, a new wrapper instruction needs to be introduced to apply the proposed interconnect SI test methodology.

For interconnect SI tests involving cores whose internal structure is not known or if system integrators are not willing to incur the complexity of applying the SI test in functional mode, we propose a new WOC design that is able to enforce different transition times on victims and aggressors.

It is hard to accurately predict crosstalk between SOC interconnect wires early in the design phase and it is even more difficult to accurately calculate the (temporal) skews between a victim and its aggressors that result in the maximum SI effects. We propose to realize the skewed transitions by adding buffers with different delay values in the WOC, and apply them iteratively or selectively to

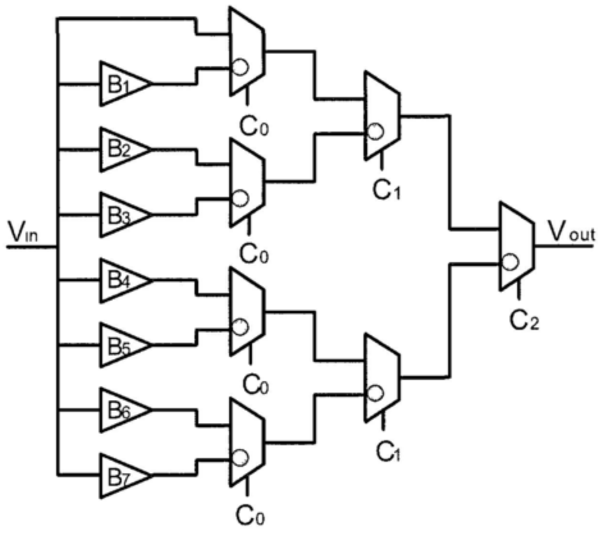


Figure 2.5: Controlled-delay element obtained by adding buffers with various delays.

target the worst-case scenario during test application.

Figure 2.5 shows an example that adds eight different kinds of transition delays. The signal V_{out} will make a transition after V_{in} passes a buffer controlled by C_2,C_1 and C_0 and some multiplexers. The delays through the multiplexers do not affect the test procedure because all paths between V_{in} and V_{out} go through the same number and type of multiplexers. The wires connecting the components in Figure 2.5 are very short and their delay is neglectable. Therefore, the transition skews between the different V_{out} signals are determined only by the buffers that they pass through.

The proposed WOC with skewed transition capability is shown in Figure 2.6. Flip-flops FF1 and FF2 are used to store the transition to be applied to the core-external interconnect, while FF3, FF4 and FF5 are used to control the delay unit. In SI test mode, both the test vectors and the control signals are shifted into the wrapper cell first. The test patterns are then applied with skewed transition

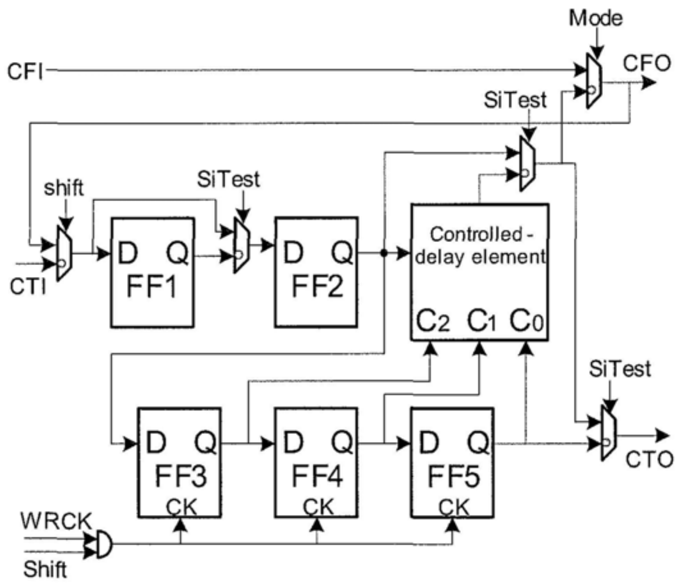


Figure 2.6: Wrapper output cell design with skewed-transition capability.

times on the victim and the aggressor interconnects. In order to prevent the control signals for the delay element to change during the capture phase, their clock signals are generated by 'ANDing' the test clock signal with the 'Shift' signal. The timing diagram of the WOC in operation is shown in Figure 2.7.

In practice, we expect the designers to estimate the time windows between functional mode and test mode, and the possible worst-case noise from IR drop, environmental fluctuations, and process variations. These compound noise effects can be mapped into possible skewed transitions between the victim and its aggressors, which can then be used to determine the number and the sizes of the buffers that cover worst-case scenarios to avoid the under-testing of devices. More buffers imply better and more fine-grained resolution and hence more accurate test results, but they result in larger DFT area overhead, longer testing time, and possibly some yield loss as a side-effect. For example, consider a victim interconnect with four aggressors and the maximum skew between the victim and any aggressor of 300

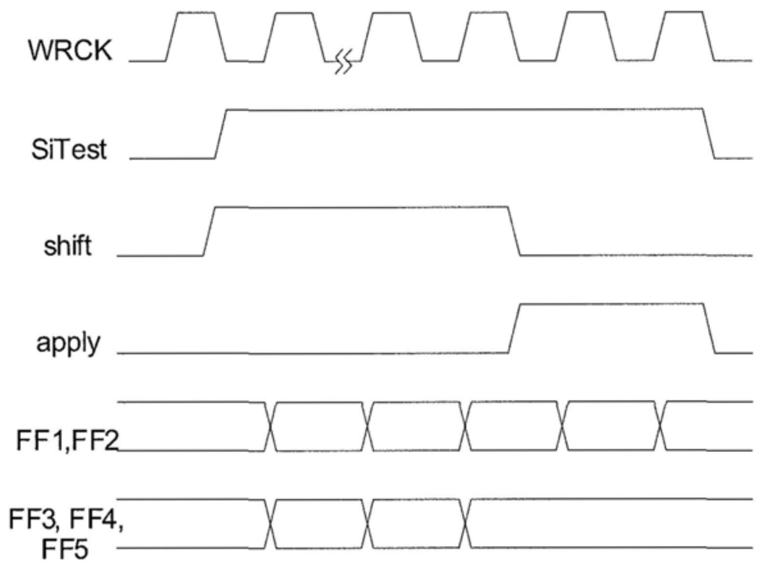


Figure 2.7: Timing diagram for the wrapper output cell in SI test mode.

ps. If we introduce three types of buffers with delays 100 ps, 200 ps and 300 ps, respectively, the skew resolution is 100 ps and the test time for one SI pattern is at most $4^4 = 256$ cycles. If, however, only one type of buffer with delay 300 ps is introduced, the test time for one test pattern is at most $2^4 = 16$ cycles; however, the skew resolution for the test is significantly reduced. As a result, designers need to trade-off test quality, testing time and DFT area overhead in determining the number of buffers to be introduced in the wrapper cells.

It should be also noted that, because the proposed wrapper output cells add delay values that might not be coincident with the ones in functional mode, it is likely that the interconnect SI faults are over-tested. We rely on the designers to carefully select delay elements added into the wrapper cells to solve this problem. After all, the proposed methodology gives the designers the flexibility to achieve quality tests in line with their guidelines, which is not possible with traditional wrapper cells and is particularly important for SoCs with high reliability requirements.

2.5 Experimental Results

In this section, we report SPICE simulation results for a 90 nm process technology with nine metal layers. The results demonstrate the effectiveness of the proposed methods for interconnect SI testing. We use Synopsys HSPICE tool for the simulations.

2.5.1 Results for the Overshoot Detector

The power-supply voltage V_{dd} for the 90 nm technology is $1V$, and we set the overshoot limit as 10% more than V_{dd}, i.e., $V_+ = 1.1$ V. Figure 2.8 compares the simulation waveforms for the detector in [62] with those for the proposed overshoot detector. As can be observed from Figure 2.8(a), the input signal to the wrapper input cell, CFI, has a low-to-high transition in the first SI test and stays at logic '1' in the next two SI tests, followed by a high-to-low transition after the third SI test and transits back to logic '1' for the last two SI tests. Overshoots occur in the first, the third, the fourth, and the fifth SI tests, in which the amplitude and breadth of the overshoot happened in the fourth SI test (i.e., VO_3) is smaller than the others. We carefully fine-tuned the design of the proposed detector and the detector proposed in [62] so that they are of similar size and activate detection when the overshoot signal exceeds V_+.

As shown in Figure 2.8(d), the overshoot detector of [62] successfully identifies the first overshoot, but its output subsequently remains in the error state (logic '0' in this detector) until CFI changes to logic '0' before the fourth SI test. This response is clearly incorrect for the second SI test. For the proposed overshoot detector, however, every time before the SI test pattern is actually applied to the interconnects (i.e., during the shift phase), the 'Shift' signal inside the wrapper input cell stays at logic '1' and resets the detector back to the error-free state. During the capture phase, as shown in Fig. 2.8(c), whenever the input signal has a

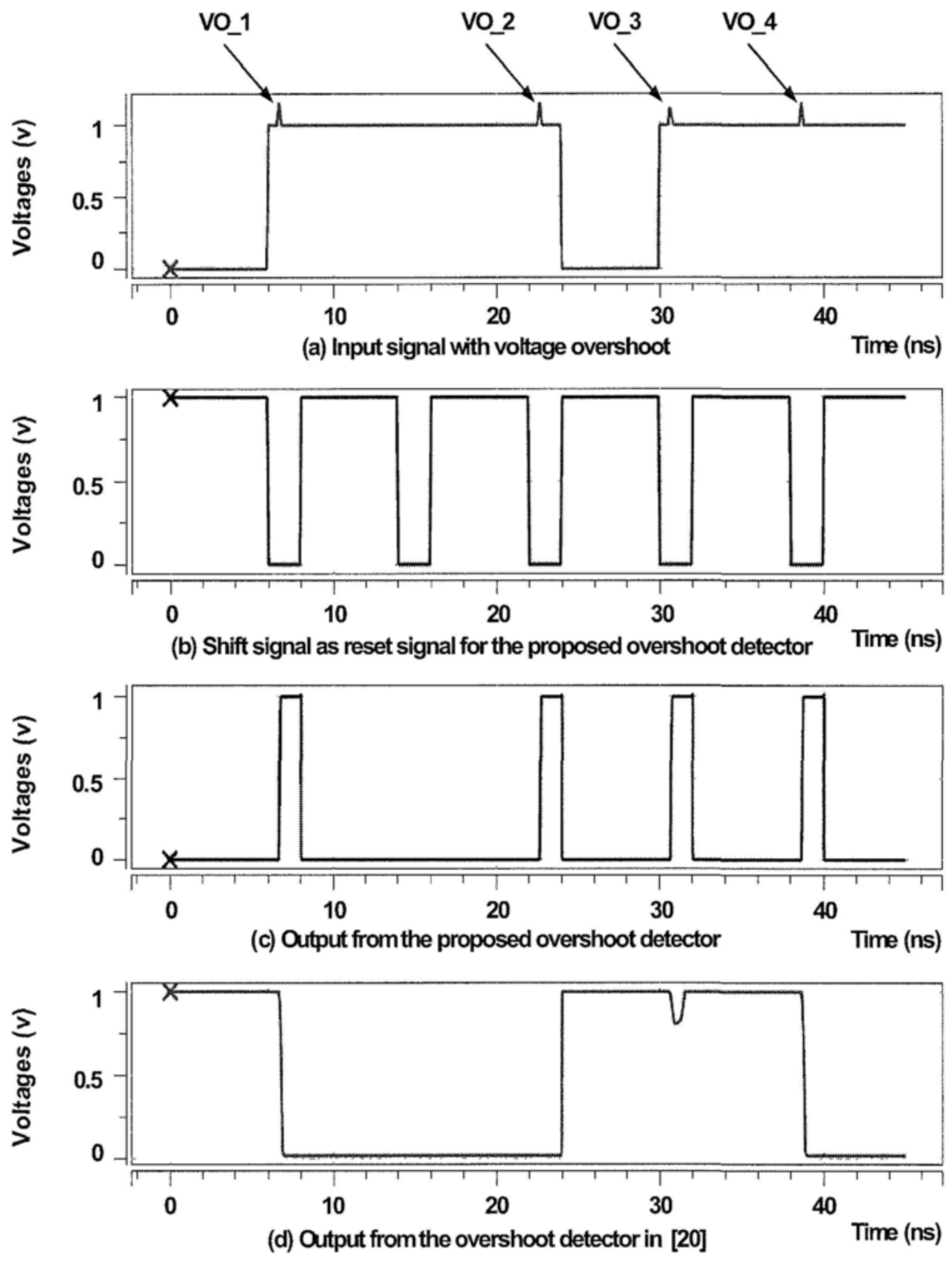


Figure 2.8: Comparison of the overshoot detection results.

overshoot violation, our wrapper input cell is able to capture it (logic '1' in this detector). That is, with the property that the proposed overshoot detector is able to reset itself in the beginning of every SI test cycle, we can correctly identify all overshoots at the cost of only one transistor. In addition, for the overshoot happened in the fourth SI test, the breadth of the overshoot is not large enough for the overshoot detector of [62] to catch. Because our overshoot detector has a larger voltage gain and higher resolution with the self-biased amplification property resulting from the circuit topological advantage, however, it successfully detects this overshoot using similar size as the one in [62].

2.5.2 Results for the Proposed Wrapper Output Cell

To evaluate the effectiveness of the proposed wrapper output cell for detecting core-external SI errors, we set up the simulation environment shown in Figure 2.9. The victim interconnect is assumed to be 5 mm long and we also assume that there are five aggressors coupled with it. These five aggressors couple with the victim line at different sites with 0.5 mm length on the eighth metal layer of the technology and the distance between every aggressor and the victim is 0.28 μm. In addition, all the wires are assumed to be 0.84 μm wide. The resistances, wire self capacitances and the coupling capacitances are calculated based on the parameters from the technology provider. For SPICE simulation, we use a distributed RC model with each segment of length 0.05 mm long. At the driving side, the WOC connects to a buffer composed of two inverters (with the first one minimum-sized and the second one four times larger) to drive the interconnects. At the receiving side, the interconnect directly connects to the wrapper input cell.

Figure 2.10 presents the timing diagram for the case that the regular IEEE 1500 Std. wrapper output cells in [50] are used to apply the signal integrity test. As can be observed from the figure, all the five aggressors make the high to low transitions at the same time with the victim's transition, and the propagation delay from the driving end of the victim interconnect to its receiving end is 0.556 ns. Figure 2.11 shows the timing diagrams for the case when the proposed wrapper output

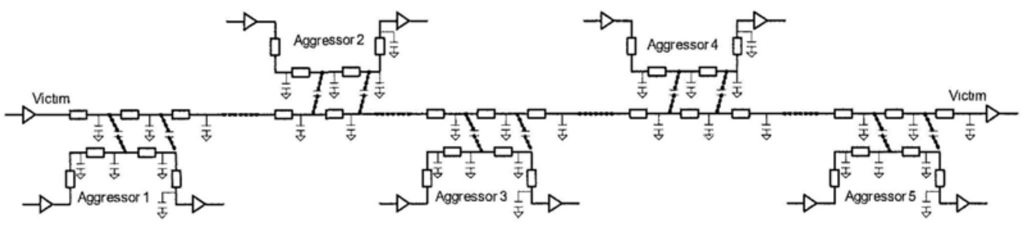


Figure 2.9: Experimental setup for SPICE simulations.

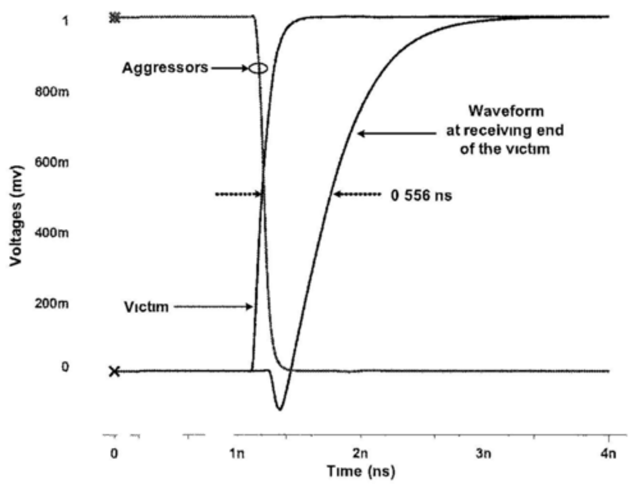
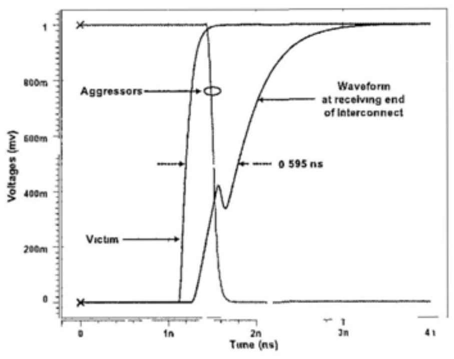
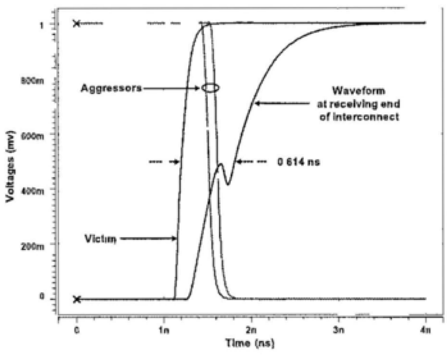


Figure 2.10: Timing diagram for SI tests with the regular IEEE Std. 1500 WOC.

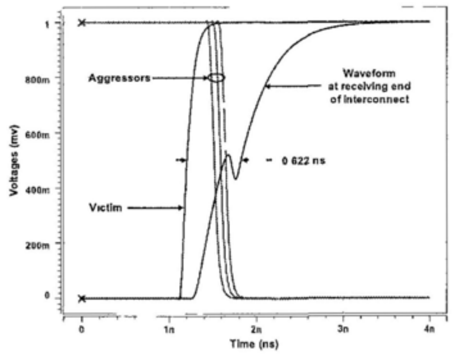
cells are utilized, with the number of buffers to be 2, 4, 6, and 8, respectively. When the aggressors make transitions at different time with 8 buffers, as shown in Fig. 2.11(d) (three lines are shown because aggressors 1 and 2, and aggressors 3 and 4 go through the same buffer), the propagation delay from the driving end of the victim interconnect to its receiving end is 0.627 ns, which is close to 13% larger than the case with regular wrapper output cells. Clearly, without skewed-transition capability, the regular wrapper output cells is not able to detect timing errors that may happen in functional mode on the victim interconnect, thus leading to under-testing of the device. The proposed WOC design is able to detect such errors and hence allows us to ship high-quality products to the customer.



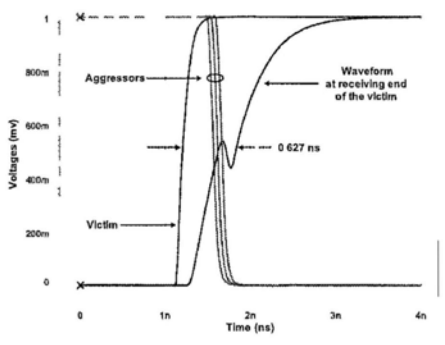
(a) SI test with 2 buffers in the WOC



(b) SI test with 4 buffers in the WOC



(c) SI test with 6 buffers in the WOC



(d) SI test with 8 buffers in the WOC

Figure 2.11: Timing diagrams for SI tests using the proposed WOC with different buffer counts.

Figure 2.11 also compares the effectiveness of the SI tests when the number of buffers inside the proposed WOC is varied. As can be observed from the figure, when only 2 buffers are used in the WOC, the propagation delay is 0.595 ns, which is 5 percent less than the case when 8 buffers are used in the WOC. When 4 or 6 buffers are employed, however, the propagation delay is quite similar to the one with 8 buffers. More buffers in the WOC implies larger DFT area overhead and longer testing time; therefore, although the test resolution is higher for 8 buffers, the DFT engineers may opt to use 4 or 6 buffers based on their test requirements.

2.5.3 DFT Area Overhead for the Proposed Wrapper Output Cell

Finally, we discuss the area overhead of the proposed wrapper designs. Compared to a IEEE Std. 1500 wrapper cell, the new wrapper input cell contains two more flip-flops, three additional multiplexers and a new overshoot detector and hence is about 30∼40 two-input NAND-equivalent gates larger than the standard wrapper input cell. The new wrapper output cell contains four more flip-flops, three extra multiplexers and a new controlled-delay element. The size of the delay unit is determined by the number and the sizes of the buffers. With the example design in Fig. 2.5, the new WOC has 60∼80 additional two-input NAND gates compared to the regular wrapper output cell. The DFT area overhead is hence potentially quite high if all the core outputs are supplied with the proposed wrapper cells. For example, for a large industrial benchmark SoC s34392 with 997 core output terminals [52], the DFT area overhead is around 60k∼80k two-input NAND gates. In practice, however, the proposed wrapper cells can be used only for those interconnects that are estimated to have larger coupling effects with other wires (typically long wires and those not adequately protected/shielded for SI-related errors). Suppose 20 percent of these core outputs are equipped with the proposed WOC. In this case, the DFT area is only 15k gates, which is acceptable considering the improved test quality.

2.6 Conclusion

Signal integrity is a major concern for today's complex system-on-a-chip integrated circuits. In this chapter, we have presented two IEEE 1500-compatible wrapper designs to effectively test SI-related faults on core-external interconnects. These wrappers can apply signal integrity tests in functional mode or enforce transitions on interconnects with various pre-defined skews between a victim line and its aggressors. We have also introduced a novel overshoot detector inside the proposed wrapper, which is able to detect the occurrences of overshoots in all possible situations. SPICE Simulation results for a 90 nm technology show that the proposed wrapper design is more effective for detecting SI-related errors when compared to existing techniques, with a moderate amount of DFT overhead.

□ **End of chapter.**

Chapter 3

SOC Test Architecture Optimization

The test time for core-external interconnect shorts and opens is typically much less than that for core-internal logic in core-based system-on-a-chip (SOC). Therefore, prior work on test-infrastructure design has mainly focused on minimizing the test time for core-internal logic. However, as feature sizes shrink for newer process technologies, the test time for signal integrity (SI) faults on interconnects cannot be neglected. The test time for SI faults can be comparable to, or even larger than, the test time for the embedded cores. It is investigated in this chapter the impact of interconnect SI tests on SOC test-architecture design and optimization. A compaction method for SI faults and algorithms for test-architecture optimization are also presented. Experimental results for the ITC'02 benchmarks show that the proposed approach can significantly reduce the overall testing time for core-internal logic and core-external interconnects.

3.1 Introduction

As feature sizes shrink and clock frequencies increase for high-performance system-on-a-chip (SOC) designs, signal integrity (SI) is becoming a major concern for the interconnects between embedded cores [39]. SI-related problems are aggravated in core-based SOC designs because interconnects transporting signals between embedded cores tend to be long, hence they suffer more from crosstalk effects [60].

Traditionally, SI problems have been treated as design errors, and a number of physical design and fabrication solutions, e.g., [11, 19, 53, 93], have been proposed in the literature to tackle them. The above design techniques rely on accurate simulation of SI effects, which are affected by many parameters (e.g., characteristics of interconnects and transistors, input data and environmental noise). Unfortunately, these parameters are interdependent and our lack of complete knowledge of this interdependence leads to uncertainty and inaccuracies in the simulation of SI loss [85]. Moreover, process variations and manufacturing defects may aggravate the SI-related problems [58]. Since it is unacceptable to over-design the circuit to tolerate signal integrity loss in all cases and it is impossible to predict the occurrence of defects, manufacturing test strategies are essential for detecting SI-related errors [21, 77, 80].

Various SI fault models (e.g., [21, 40, 81]) and associated test methodologies (e.g., [10, 80]) have been proposed in the literature. SI-related problems are aggravated in core-based SOC designs because interconnects carrying signals between embedded cores tend to be long and hence they suffer more from parasitic effects [60]. Despite this problem, most prior work in SOC test-architecture optimization has focused on core-internal test (InTest) only (e.g., [25, 31, 36, 26, 41, 57, 89, 94, 98]) and neglected the problem posed by core-external interconnect SI faults. The test time for SI faults is long because of the need to exercise a large number of signal-state combinations for the interconnects [77, 80]. For nanometer SOCs running at speeds of several hundred MHz and higher, the test time for SI faults can be as high as or even exceed the test time for the embedded cores. Therefore, the goal of this chapter is to study, for the first time, the impact of SI faults on SOC test-architecture design. The main contributions of this chapter are as follows:

- We present a two-dimensional SI test pattern compaction strategy to reduce the interconnect SI test data volume.
- We develop algorithms for SOC test-architecture optimization to minimize the overall SOC testing time for both interconnect SI faults and core-internal faults.
- We show that for the ITC 2002 SOC Test benchmarks, the test time obtained using the proposed method is significantly less than that using two baseline methods: (i) a test access architecture optimized for core-internal test and then used for core-external SI fault testing; (ii) a test-access architecture optimized for both core-internal test and core-external test, but where only pattern count reduction is employed for the tests for SI faults.

3.2 Related Work and Motivation

Early attempts for testing SI-related problems modeled crosstalk at the circuit level [5, 20]. Although more accurate than gate-level models, the complexity of the associated test-pattern generation procedures limits its usefulness for SOC interconnects. Cuviello et al. [21] proposed a behavioral-level SI fault model, called the *maximal aggressor (MA)* model. As defined in Chapter 1, an interconnect on which the error effect takes place is defined as the *victim*, while the affecting interconnects are referred to as its *aggressors*. The MA model assumes that all aggressors make the same simultaneous transition (in the same direction) and act collectively to generate a glitch when the victim is quiescent or a delay error when the victim makes an opposite transition. Therefore, $6N$ test-vector pairs are needed to detect

SI faults for a set of *N* interconnects. Since in reality, inter-core interconnects in SOC can be of any arbitrary topology, to reduce test pattern count, Sirisaengtaksin and Gupta [77] extended the MA fault model to the so-called maximum-affecting-line (MAL) fault model by taking the physical layout information into account. If all the physical defects are capacitive or resistive, all MA/MAL faults can be targeted using a pattern count that is linear in the number of interconnects. When inductance is considered, however, such test patterns may not be able to generate maximum noise/delay on the victim line [20, 55]; hence, Tehranipour et al. [81] presented a *multiple transition (MT)* fault model that covers all transitions on victim and multiple transitions on aggressors. The number of test patterns for this MT fault model, however, is exponential in the number of interconnects under test. To address this problem, an empirically-determined locality factor *k* showing how far the effect of aggressors remains significant, was introduced. For a set of *N* interconnects, the number of test patterns for the reduced-MT fault model is approximately $N \cdot 2^{2k+2}$.

Built-In Self-Test (BIST) has been a popular test method used to detect SI-related errors [74, 81]. In this approach, driver side of interconnects are equipped with test generators to generate transitions on the aggressors and victims, while at the receiver side, various types of integrity-loss sensor (ILS) cells are embedded to detect SI-related errors. Bai et al. [10] introduced on-chip test generators and error detectors at the core boundaries, based on the MA fault model [21]. Nourani and Attarha [61] presented two ILS cell designs to detect voltage distortions and timing violations, respectively. Later, several other ILS designs [13, 78] were introduced, which are more accurate in measuring voltage and/or timing violations, at the cost of large area overheads. Assuming the existence of logic BIST structures in an SOC, Sekhar and Dey [19] presented a self-test solution, called LI-BIST, for both the core internal logic and the SOC interconnects. Zhao et al. presented an on-

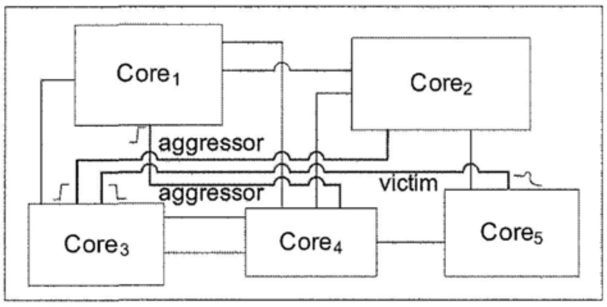


Figure 3.1: An arbitrary interconnect topology in an SOC, with a victim and the corresponding aggressors.

line testing technique to capture noise-induced logic failures in functional buses [96]. Yang et al. [92] used boundary scan and IDDT to test functional buses. Finally, [16] discussed how to test SI defects on the data bus and the address bus by executing a test program on the microprocessor.

A test method that relies on hardware-based test generators may cause over-testing and/or under-testing since not all test patterns generated in the test mode are valid in the normal functional mode of the SOC. In addition, since the SOC interconnect topology can be arbitrary (see Fig. 3.1) and it is hard to predict it during test-hardware insertion, the interconnects between several cores may be close enough to result in SI errors [77]. It is very difficult, if not impossible, to take interconnect proximity into account for these hardware-based test techniques. Therefore, in this work, we assume that the test stimuli are loaded from an external tester to the core-test wrapper. To apply SI test at the core-level, as shown in Fig. 3.2 [80], the wrapper-output cell (WOC) should be able to provide the necessary consecutive transitions to interconnects; the wrapper-input cell (WIC) needs to be equipped with a signal integrity loss sensor (e.g., [10, 80]), to capture the signal with noise and/or delay error.

Most prior work in SOC test-architecture optimization [90] only takes core in-

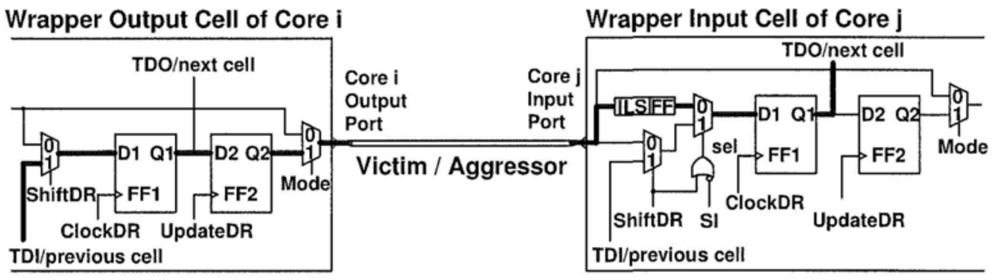


Figure 3.2: Wrapper cell design for SI test (redrawn from [Tehranipour et al. 2003]).

ternal testing into account, which is mainly because testing interconnect shorts/opens requires little time and therefore core-external (ExTest) testing can be ignored in the test-architecture optimization process. However, when high SI fault coverage is desired for today's SOCs, the testing time for SOC interconnects can be comparable to or even higher than the testing time for the core-internal logic. To understand this issue, let us estimate the interconnect SI testing time for a representative video-processing SOC [24, 32], which contains two 32-bit programmable interconnect (PI) buses, each connecting to a number of embedded cores (e.g., MIPS/TriMedia processor, mpeg-2 decoder, transport stream processor, and IEEE 1394 controller). Without loss of generality, suppose ten cores connect to each PI buses and assume that each core on average sends data to two other cores on the bus. Hence, the number of victim interconnects under test on each PI bus is $N = 2 \times 10 \times 32 = 640$. Based on the above discussion, without test set compaction, $6N \times 2 = 7680$ test vector pairs are needed for the MA fault model; while roughly $N \cdot 2^{2k+2} \times 2 = 327680$ test vector pairs are needed for the reduced-MT fault model with the locality factor $k = 3$. Since the total numbers of all the core I/Os for a typical SOC is in the range of several thousand, the test time for MA faults is in the range of millions of clock cycles for serial ExTest, while the test time for reduced-MT faults is two orders of magnitude higher. On the other hand, as reported in [32], the SOC test time for core-internal logic is less than two million clock cycles when the total number of test access mechanism (TAM) wires is 140, which in turn is less than the testing time for the above SI faults. Moreover, with shrinking feature sizes of deep-submicron technology, short interconnects may also suffer from SI problems [60]. Therefore, it is likely that we need to test for SI faults on hundreds or even thousands of interconnects in the SOC. Prohibitively high test time is needed if an effective test-pattern compaction scheme is not employed and the SOC test-architecture is not optimized for both core-internal logic test and interconnect SI test.

Three important conclusions can be drawn from the above discussion:

- Effective test set compaction strategy should be utilized to reduce the volume of test data for interconnect SI faults;
- Parallel external testing is required in order to reduce the test time for interconnect SI faults;
- The SOC test-architecture needs to minimize the overall testing time for both core-internal logic and core-external interconnects.

The above observations motivate the work presented in this chapter.

3.3 Two-Dimensional SI Test-Set Compaction

We assume that the test stimuli for SI faults are given to us *a priori*; these stimuli can take the form of functional patterns, pseudorandom patterns, and/or patterns generated for various SI fault models [21, 77, 81]. Since a victim interconnect is mainly affected by its neighboring aggressors [40], the signal integrity test patterns typically feature a large number of don't-care bits. The format of the SI test vector

Table 3.1: Format of the SI test patterns.

	$core_1$ *WOC*	...	$core_i$ *WOC*	...	$core_n$ *WOC*	**Bus**
p_0	... x ↑ x x ↓	...	x 1 ↑ ... x	...	x x x ...	x x 1 ...
p_1	... ↑ x ↓ x x	...	x x x ... ↑	...	x x ↑ ...	x x 1 ...
...		...		...		
p_x	... x x x x x	...	0 x ↓ ... x	...	↓ x x ...	1 x x ...

pairs applied at the wrapper output cells of the embedded cores is shown in Table 3.1. The entry 'x' represents a don't-care bit; '0/1' indicates that the corresponding core output terminal stays at 0/1 in consecutive cycles, and ↑ (↓) represents a positive (negative) transition. For each test pattern, we also add a postfix to denote whether this test pattern utilizes a shared bus line (as discussed in the following paragraph)—a '1' indicates that the specific bus line is utilized while 'x' implies that it is a "don't-care".

Test-Pattern-Count Reduction. Because of the large number of don't-care bits in each test pattern, it is possible to reduce the volume of test data by compacting multiple test vectors into one vector when they are compatible (i.e., their intersection is non-empty). Note that since bus lines are shared by the cores and they may connect many cores at the same time, several SI test patterns may trigger the same bus line from different core boundaries; these patterns cannot be compacted into one test pattern. The postfix that we add to each SI test pattern is used to identify such situations. If the bit values for a specific position in the postfix of two SI test patterns are both '1', they are marked as incompatible (e.g., p_0 and p_1 in Table I). The problem of finding a compacted test set of minimum size for a given test set is very similar to the traditional test compaction problem and can be formulated as a maximal clique-partitioning problem [37]. The pattern com-

Algorithm 1 - TestCompaction

INPUT: P_o

OUTPUT: P_c

```
1. initialize $P_c = \emptyset$; $P_u = P_o$;      /* $P_u = \{P_u[1], P_u[2], \ldots\}$ */
2. while ($|P_u| > 0$) {
3.    set $p_c = P_u[1]$ ; $P_m = \{P_u[1]\}$;
4.    for ($i = 2$ to $|P_u|$) {
5.       if($p_c$ and $P_u[i]$ are compatible) {
6.          merge $p_c$ and $P_u[i]$ to $p_c$;
7.          $P_m = P_m \cup \{P_u[i]\}$; }
.    }
8.    $P_u = P_u \setminus P_m$; $P_c = P_c \cup \{p_c\}$;
. }
9. return $P_c$.
```

Figure 3.3: Procedure for SI test pattern count reduction.

paction problem is mapped to a graph, where each vertex corresponds to a test pattern and an edge is added between two vertices if the corresponding test patterns are mutually compatible. A set of compatible SI test patterns form a clique in this graph; our objective to find a minimum number of disjoint cliques that cover all the vertices in the graph. The clique partitioning problem, however, is known to be NP-complete [29], and approximation algorithms, i.e., those with bounded approximation error, suffer from high computational complexity [3].

To reduce computation time, we use a simple greedy heuristic as shown in

Fig. 3.3. The algorithm takes the original test set P_o as input. A compacted test pattern is generated in each inner loop (Lines 4-7) by merging the first pattern p_1 in the un-compacted test set P_u with the compatible patterns that follow in one pass. The algorithm terminates when all test patterns are compacted, and it outputs the compacted test set P_c. Obviously this greedy strategy is not optimal, and the quality of the resulting P_c depends on the order of the test patterns. To address this problem, we randomize the order of test patterns several times, apply our greedy heuristic, and select the best result, i.e., the ordering that leads to the smallest compacted set P_c. Suppose the number of original test patterns is n and the test pattern width is m. In the worst case, no test pattern is compatible with any other pattern. The time complexity of the above heuristic is $O(mn^2)$.

The above compaction scheme to reduce test pattern count can be viewed as reducing the volume of the test data in a vertical manner.

Test-Pattern-Length Reduction. If we compact all the test patterns together, the length of every compacted pattern will be very large—it will be equal to the sum of the number of WOCs for the different cores. As each SI test pattern involves only a few cores' terminals (referred to as *care cores* of the SI test pattern), we can bypass the boundaries of the remaining *don't-care cores* (e.g., Core 1 for p_x in Table 3.1) and reduce the length of this test pattern. The above strategy can be viewed as compacting the test pattern in a horizonal manner.

That is, instead of compacting all the test patterns together, we first partition the set of cores into several smaller groups of cores (say, N_g groups). Next we classify the SI test patterns in such way that the test patterns, whose care cores are all within the same core group, form an SI test group. The length of each test pattern is now reduced to the sum of the number of WOCs of this core group, instead of the WOCs of all cores. Let woc_i denote the number of WOCs for core group i. For the remaining test patterns whose care cores fall into multiple core

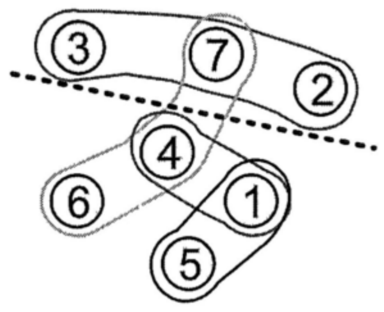


Figure 3.4: Hypergraph partitioning for SI test pattern length reduction.

groups, we simply group them as a whole and their length remains the sum of the lengths of the WOCs for all the cores, denoted as woc_{SOC}. The test data volume V_c after two-dimensional compaction is as follows:

$$V_c = (\sum_{i=1}^{N_g} p_i \times woc_i + p_r \times woc_{SOC}) \times 2, \tag{3.1}$$

where p_i and p_r represent the number of compacted test patterns in SI test group i and the number of compacted remaining test patterns, respectively. The value '2' in the above equation is added because each SI test pattern contains two vectors.

To achieve better compression, we should minimize the number of remaining patterns, and at the same time, balance the test-pattern lengths for the partitions. This problem can be formulated as a hypergraph partitioning problem, with each vertex in the hypergraph corresponding to a core. The weight of each vertex is the number of WOCs of the core corresponding to the vertex and it is used to balance the partitions. A hyperedge is added for each test pattern that connects all its care cores (vertices). Since there might be multiple test patterns having the same care cores, we use the weight of each hyperedge to represent this information. The hypergraph partitioning problem has been well-researched in the literature and we use the *hMetis* package [75] to solve this problem. As shown in Fig. 3.4, for the horizontal SI test pattern compaction of a hypothetical SOC containing seven cores, the patterns corresponding to the cut hyperedge 7-4-6 need to load the WOCs for all the cores, while the other patterns can be applied with shorter pattern lengths. For simplicity, the vertex and edge weights are not shown in the figure.

The main objective of the *hMetis* package is to minimize the number of hyperedges that are cut (i.e., the remaining patterns in our problem), while our objective is to achieve the minimum test data volume as shown in Equation (3.1). To obtain better results, we use the partitioning result obtained from *hMetis* as an initial solution, and on top of it employ the FM partitioning algorithm [28] with the cost function shown in Equation (3.1) to refine the original solution. That is, we try to move one core at a time between partitions, and check whether the test data volume is reduced. If this is indeed the case, we fix this movement; otherwise, we try an alternative movement. Our experiments show that the above refinement step is able to further reduce test data volume by $3 \sim 5$ percent when compared to the solution obtained from *hMetis*.

3.4 Test-Access Architecture Design and Optimization

We consider, as a starting point, that every core in the SOC uses wrapper cells as shown in Fig. 3.2 [80]. These wrappers are compatible with the IEEE 1500 standard [35] with some additional hardware added to the wrappers for signal integrity test, including a new wrapper instruction to enter the signal-integrity test (*SITest*) mode. In addition, the user-defined logic (e.g., the glue logic between embedded cores) is also treated as a wrapped core. In other words, the SOC is assumed to contain only wrapped logic blocks and interconnect wires that are affected by signal integrity faults.

In addition, we use the TestRail TAM architecture in this work [49]. While it is possible to use the alternative Test Bus architecture [84] to support parallel external testing [88], the TestRail architecture is more amenable for core-external testing [31].

3.4.1 SI Test-Architecture Optimization: Problem Formulation

As discussed in Section 2.2, the testing time for interconnect SI faults can be comparable to or even higher than the testing time for core-internal logic. Therefore, it is necessary for system integrators to optimize the SOC test-architecture for both kinds of tests in order to reduce the overall testing time. The optimization problem addressed in this section can be formulated as follows:

Problem P_{SI_opt}: Given the maximum TAM width W_{max} for the SOC, and

- the test set parameters for each embedded core, including the number of input and output terminals, the number of test patterns for core internal logic, the number of scan chains and the length of each scan chain;
- the test set parameters for each group of compacted interconnect SI tests obtained using the method proposed in Section 3.3, including the set of cores involved and the number of SI test patterns;

Determine the wrapper design for each core, the TAM resources assigned to each core and a test schedule for the entire SOC such that: (i) the sum of the TAM width used at any time does not exceed W_{max}; (ii) the total SOC testing time T_{SOC} is minimized.

One of the subproblems of P_{SI_opt} is to design and optimize the test wrapper for each core. Since the test application time of a core is dependent on the length of the maximum wrapper scan chain[1], the main objective in wrapper design and optimization is to build balanced wrapper scan chains. This is a well-researched problem [51, 36], and we use the *Combine* procedure from [51] for solving it in *InTest* mode. For a core wrapper in SI test mode, wrapper scan chains contains wrapper cells only and we can therefore assume that balanced wrapper input/output scan chains are achieved. Based on the TestRail architecture, we propose to solve Problem P_{SI_opt} in two steps. First, we describe how to schedule SI tests for a given TAM design, as shown in Section 3.4.2. Next, we describe our solution for the general problem of how to design and optimize the SOC test-architecture from scratch by adapting an existing method [31]; this approach is presented in Section 3.4.3.

3.4.2 SI Test Scheduling for a Given TAM Design

Wrapper cells are used for both core-external interconnect SI test and core-internal logic test at the same time. Hence, to avoid test-resource conflicts, we schedule the two types of tests at different times. Therefore, $T_{SOC} = T_{SOC}^{in} + T_{SOC}^{si}$, where T_{SOC}^{in} and T_{SOC}^{si} denote the test time for the core-internal logic and the test time of the core-external interconnects, respectively.

The need for combining interconnect SI test with core-internal test makes test-architecture optimization more difficult compared to the case when only core-internal test is considered. This difficulty results from the fact that interconnect SI test patterns may involve multiple TAMs at the same time, as victim and aggressors in a crosstalk environment may link cores connected to different TAMs. To highlight this problem, we examine how T_{SOC} can be calculated for a given TAM design.

[1]Wrapper scan chains are constructed by concatenating core-internal scan chains and WBR cells for InTest.

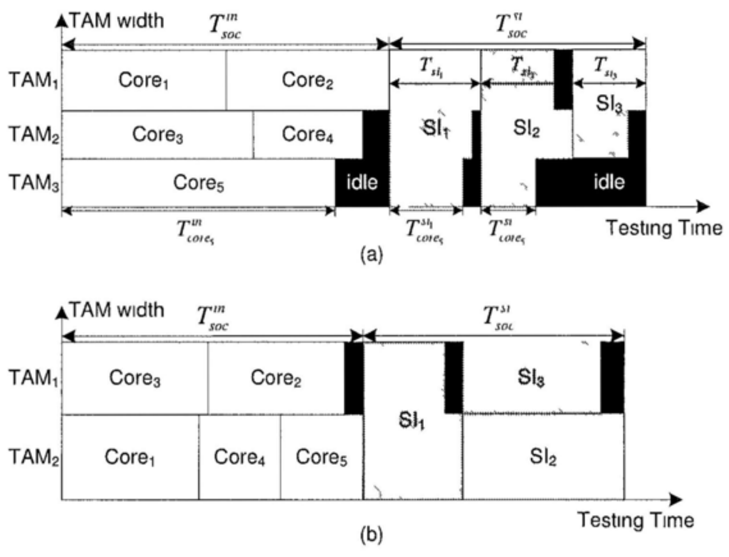


Figure 3.5: Example TAM designs and their corresponding test schedules.

Consider the hypothetical SOC shown in Fig. 3.1. Suppose that after two-dimensional test compaction, the SI test has been placed in three groups, where the SI_1 group involves all the five embedded cores (these are the remaining test patterns after partitioning), SI_2 group involves $Core_1$, $Core_4$ and $Core_5$, and SI_3 group involves $Core_2$ and $Core_3$. Two possible TAM designs and their corresponding test schedules are shown in Fig. 3.5. Let $T^{in}_{core_i}$ denote the core-internal test time for of Core i, and let $T^{SI_j}_{core_i}$ denote the interconnect test time for SI test group j contributed by core i. For the schedule shown in Fig. 3.5(a),

$$\begin{aligned}
T^{in}_{SOC} &= T^{in}_{tam_1} = T^{in}_{core_1} + T^{in}_{core_2} \\
T^{si}_{SOC} &= T_{si_1} + T_{si_2} + T_{si_3} \\
T_{si_1} &= \max\{T^{si_1}_{core_1} + T^{si_1}_{core_2}, T^{si_1}_{core_3} + T^{si_1}_{core_4}, T^{si_1}_{core_5}\} \\
&= T^{si_1}_{core_1} + T^{si_1}_{core_2} \\
T_{si_2} &= \max\{T^{si_2}_{core_1}, T^{si_2}_{core_4}, T^{si_2}_{core_5}\} \\
&= T^{si_2}_{core_4} \\
T_{si_3} &= \max\{T^{si_3}_{core_2}, T^{si_3}_{core_3}\} \\
&= T^{si_3}_{core_2}
\end{aligned}$$

For the second test schedule shown in Fig. 3.5(b),

$$\begin{aligned}
T^{in}_{SOC} &= T^{in}_{tam_2} = T^{in}_{core_3} + T^{in}_{core_4} + T^{in}_{core_5} \\
T^{si}_{SOC} &= T_{si_1} + \max\{T_{si_2}, T_{si_3}\} = T_{si_1} + T_{si_2} \\
T_{si_1} &= \max\{T^{si_1}_{core_1} + T^{si_1}_{core_4} + T^{si_1}_{core_5}, T^{si_1}_{core_2} + T^{si_1}_{core_3}\} \\
&= T^{si_1}_{core_1} + T^{si_1}_{core_4} + T^{si_1}_{core_5} \\
T_{si_2} &= T^{si_2}_{core_1} + T^{si_2}_{core_4} + T^{si_2}_{core_5} \\
T_{si_3} &= T^{si_3}_{core_2} + T^{si_3}_{core_3}
\end{aligned}$$

In the above example, T^{in}_{SOC} is the maximum core-internal test time of the individual TAMs for a given TAM architecture. The sequence of the core internal tests does not affect the value of T^{in}_{SOC} hence the test time can easily calculated from the TAM architecture. The calculation of T^{si}_{SOC}, however, is less straightforward because multiple TAMs may be involved. First, we need to calculate T_{si_j} for each SI test group j, which is determined by a single TAM denoted as the *bottleneck TAM* for this SI test (e.g., TAM_2 for SI test SI_2). Second, we need to schedule the SI tests to minimize T^{si}_{SOC}. We next elaborate on these two steps .

```
Data structure SI Test s

1. $C(s)$;              /* The set of cores involved */
2. $pattern(s)$;        /* Number of patterns */
3. $begin(s)$;          /* Schedule begin time */
4. $end(s)$;            /* Schedule end time */
5. $time_{si}(s)$;      /* SI Testing time */
6. $R_{tam}(s)$;        /* The set of TAMs involved */
7. $r_{btn}(s)$;        /* The bottleneck TAM for this SI test */
```

```
Data structure TestRail r

1. $C(r)$;              /* The set of cores on TestRail r */
2. $width(r)$;          /* TAM width of r */
3. $time_{in}(r)$;      /* Internal testing time */
3. $time_{si}(r)$;      /* Utilized SI testing time */
4. $time_{used}(r)$;    /* Utilized total testing time */
```

Figure 3.6: Data structures for SI test and TestRail.

Data Structure. The data structures that we use to store the SI test group information and the TestRail configuration are presented in Fig. 3.6. The two data structures are updated whenever the SOC TAM design is changed. In particular, in data structure for *TestRail r*, we use $time_{in}(r)$, $time_{si}(r)$ and $time_{used}(r)$ to denote the internal testing time, the SI testing time and the utilized testing time on TAM *r*, respectively. For example, for TAM_3 shown in Fig. 3.5(a),

Algorithm 2 - CalculateSITestTime

INPUT: R_{SOC}, S_{SOC}

OUTPUT: S'_{SOC}

```
1. for all $s_i \in S_{SOC}$ {
2.   initialize $time_{si}(s_i) = 0$;
3.   for all $r_j \in R_{tam}(s_i)$ {
4.     $C_{involved} = C(r_j) \cap C(s_i)$;
5.     $time_{si}(r_j) = \sum_{c_i \in C_{involved}} time_{si}(c_i)$;
6.     if($time_{si}(r_j) > time_{si}(s_i)$) {
7.       $time_{si}(s_i) = time_{si}(r_j)$;
8.       $r_{bin}(s_i) = r_j$; }
.    }
. }
9. return $S'_{SOC}$.
```

Figure 3.7: Procedure for calculate SI testing time.

- $time_{in}(r) = T^{in}_{core_5}$;
- $time_{si}(r) = T^{si_1}_{core_5} + T^{si_2}_{core_5}$;
- $time_{used}(r) = time_{in}(r) + time_{si}(r) = T^{in}_{core_5} + T^{si_1}_{core_5} + T^{si_2}_{core_5}$;

We use $time_{used}(r)$ to compare the actual utilization of TAM resources for different TAMs.

Calculation of Test Time for Individual SI Test. The pseudocode for the procedure to calculate the testing time for each SI test group is shown in Fig. 3.7.

The procedure takes the TestRail architecture R_{SOC} and all the SI test groups S_{SOC} as inputs and calculates $time_{si}(s_i)$ for each SI test group s_i. In the inner loop (Lines 3-8), we check all the TAMs that are involved in SI test s_i to identify the TAM that determines $time_{si}(s_i)$. Line 4 finds out $C_{involved}$, i.e., all the cores on TAM r_j that are involved in SI test s_i. Line 5 then calculates $time_{si}(r_j)$, the signal integrity testing time contributed by TAM r_j. We record the SI testing time $time_{si}(s_i)$ (Line 7) and the bottleneck TAM $r_{btn}(s_i)$ for SI test s_i (Line 8). Finally the procedure returns the SI tests with updated SI testing time (Line 9).

To calculate the time of an SI test group, the test time of each of the cores and each of the TAMs involved in this SI test needs to be computed. Therefore, the complexity of the *CalculateSITestTime* algorithm is $O(N_{SI}N_cN_{TAM})$, with N_{SI}, N_c and N_{TAM} corresponding to the number of SI test groups, the number of cores and the number of TAMs, respectively.

Scheduling of SI Tests. Once $time_{si}(s_i)$ for each individual SI test s_i is known, we can schedule the SI tests to acquire the SOC signal integrity testing time T_{SOC}^{si}. This procedure is shown in Fig. 3.8. Line 1 performs procedure *CalculateSITestTime* to calculate the testing time for each SI test. Line 2 initializes *unSchedSI*, the unscheduled SI tests and *currSchedTAMs*, the TAMs that are utilized by the SI tests currently under schedule. Line 3 initializes *currTime*, i.e., the begin time for to-be-scheduled SI test. After the initialization, the following loop schedule SI test one by one (Lines 4-17). Inside the loop, we first try to find a SI test s^* that can be scheduled with begin time *currTime*, that is, s^* does not utilize any TAM in *currSchedTAMs* (Line 5). If such s^* can be found, we schedule it by updating $begin(s^*)$, $end(s^*)$, *currSchedTAMs*, and *unSchedSI* (Lines 7-10). If s^* is the last scheduled SI test, we shall update T_{SOC}^{si} as the end time of TAM $end(s^*)$. If all the unscheduled SI tests utilize the TAM resources in *currSchedTAMs* and hence cannot be scheduled with begin time *currTime*, we find *nextTime*, i.e., the time for

Algorithm 3 - ScheduleSITest

INPUT: R_{SOC}, S_{SOC}

OUTPUT: S'_{SOC}, T^{si}_{SOC}

```
1. $S'_{SOC} = CalculateSITestTime(R_{SOC}, S_{SOC})$;
2. initialize $unSchedSI = S'_{SOC}$; $currSchedTAMs = \emptyset$;
3. initialize $currTime = 0$;
4. while $unSchedSI \neq \emptyset$ {
5.    find $s^* \in unSchedSI$ for which $R_{tam}(s^*) \cap currSchedTAMs = \emptyset$;
6.    if found {
7.       set $begin(s^*) = currTime$;
8.       set $end(s^*) = begin(s^*) + time_{si}(s^*)$;
9.       $currSchedTAMs = currSchedTAMs \cup R_{tam}(s^*)$;
10.      $unSchedSI = unSchedSI \setminus \{s^*\}$;
11.      if($unSchedSI == \emptyset$) {
12.         $T^{si}_{SOC} = end(s^*)$; }
13.   } else {
14.      calculate $nextTime = end(s')$ such that
.           $end(s') > currTime$ and $end(s')$ is the minimum;
15.      set $currTime = nextTime$;
16.      update $currSchedTAMs$;
.     }
. }
17. return $S'_{SOC}$, $T^{si}_{SOC}$.
```

Figure 3.8: Procedure for scheduling SI tests.

the first SI test that is expected to end after $currTime$ (Line 14). We then update the begin time of the to-be-scheduled SI tests (Line 15) and $currSchedTAMs$ based on the SI tests still under schedule (Line 16). Finally the procedure returns the SOC SI testing time T_{SOC}^{si} and the SI tests with updated schedule information.

Let us take the test architecture and test schedule shown in Fig. 3.5(b) as an example to demonstrate the *ScheduleSITest* algorithm. After the internal tests have been scheduled, all the TAM resources are available and we choose to schedule SI test SI_1. However, before SI_1 is completed, there is no TAM available because SI_1 uses them to shift test patterns to all cores' wrapper cells. Therefore, the other SI tests have to wait for the completion of SI_1. Afterwards, the TAM resources occupied by SI_1 are released and SI_2 and SI_3 can be scheduled with TAM_2 and TAM_3 in parallel.

During the initialization process of the *ScheduleSITest* procedure, we need to conduct procedure $CalculateSITestTime$ with complexity $O(N_{SI}N_cN_{TAM})$. Then in the inner loop, we schedule every SI test and the complexity is $O(N_{SI}N_{TAM})$ because the complexity to check the TAM availability and to update the occupancy information of a single SI test is proportional to N_{TAM}. Therefore, the overall complexity of the *ScheduleSITest* procedure is $O(N_{SI}N_cN_{TAM})$.

3.4.3 TAM Design and Optimization

The above discussion for calculating T_{SOC}^{si} is based on a given TAM architecture. What makes Problem P_{SI_opt} more difficult is that the testing time for a SI test $time_{si}(s)$ is not known until the SOC test-architecture is determined This makes P_{SI_opt} fundamentally different from the problem of designing and optimizing an SOC test-architecture for core internal-logic only. In the latter case, the testing time for each core can be pre-determined for a given TAM width [90]. Unlike many test scheduling algorithms that schedule cores one after another and terminate after all cores are scheduled, the $TR-Architect$ algorithm proposed in [31] generates an initial test-architecture with all cores assigned to TAMs in the beginning and then optimizes this architecture in an iterative manner. This strategy is particularly attractive for interconnect SI test as we are able to calculate the SI testing time in each optimization step. Therefore, we propose to adapt the $TR-Architect$ algorithm for solving Problem P_{SI_opt} in this chapter. At the same time, this adaptation is not straightforward, as described in the following paragraphs.

Identifying Bottleneck TAMs. The basic idea of the $TR-Architect$ algorithm is to optimize T_{SOC}^{in} at the TAM level by merging TAMs and/or distributing free TAM wires to the bottleneck TAM, i.e., the TAM with the longest T_{tam}^{in}. As a result, we define the *bottleneck TAMs* of the SOC (in contrast to the single bottleneck TAM for an SI test) to be those which are critical to the test time; T_{SOC} is reduced if extra wires are assigned to them; the remaining TAMs are referred to as *non-bottleneck TAMs* of the SOC. In $TR-Architect$, there exists only a single bottleneck TAM at a time during the optimization process. Either two non-bottleneck TAMs are merged with less TAM width to release freed TAM resources to the bottleneck TAM, or the bottleneck TAMs is merged with another TAM to decrease T_{SOC}^{in} [31].

In our problem, as we try to minimize $T_{SOC} = T_{SOC}^{in} + T_{SOC}^{si}$, it is possible that multiple bottleneck TAMs exist at the same time. That is, in addition to the bottleneck TAM for core-internal logic test, each SI test has its own bottleneck TAM, which may affect the total SOC testing time T_{SOC}. For example, for the schedule shown in Fig. 3.5(a), the bottleneck TAM for SI_2 (i.e., TAM_2) is a bottleneck TAM for the SOC; on the other hand, for the schedule shown in Fig. 3.5(b), the bottleneck TAM for SI_3 (i.e., TAM_1) is not a bottleneck TAM for the SOC. For the schedule shown in Fig. 3.5(a), TAM_1 and TAM_2 are bottleneck TAMs and TAM_3

Algorithm 4 -IdentifyBtnTAMs

INPUT: R_{SOC}, S_{SOC}

OUTPUT: R_{btn}

```
1. find $r'$ with maximum internal testing time;
2. initialize $R_{btn} = \{r'\}$;
3. find $r^*$ with maximum SI testing time;
4. for every core $c \in C(r^*)$ {
5.    for all $s \in S_{SOC}$ that satisfies $c \in C(s)$ {
6.       if $r_{btn}(s) \notin R_{btn}$ {
7.          $R_{btn} = R_{btn} \cup \{r_{btn}(s)\}$; }
.    }
. }
8. return $R_{btn}$.
```

Figure 3.9: Procedure for identifying bottleneck TAMs of the SOC.

is a non-bottleneck TAM, while for the schedule shown in Fig. 3.5(b), TAM_2 is a bottleneck TAM and TAM_1 is a non-bottleneck TAM.

The procedure to identify SOC bottleneck TAMs is shown in Fig. 3.9. The bottleneck TAM for core-internal logic test is guaranteed to affect T_{SOC}. Therefore, in Line 1 and Line 2, we find this bottleneck TAM r' and it is identified as a SOC bottleneck TAM (e.g., TAM_1 in Fig. 3.5(a)). Next, in Line 3, we find the TAM r^* with the longest SI test time. Each core on r^* might be involved in several SI tests. For every one of these SI tests, we identify its bottleneck TAM r_{btn}. Since the SI test bottleneck TAMs identified in this way must affect the total SOC testing time

T_{SOC}, we treat each of them as the bottleneck TAM of the SOC (Lines 4-7). From the above procedure, it can be seen that, the bottleneck TAM for those SI tests that are not involved with any core on r^* can be ignored, e.g., the bottleneck TAM for SI_3 shown in Fig. 3.5(b). For the test architecture and test schedule in the example of 3.5(a), TAM_1 is the bottleneck TAM for internal test. On the other hand, both TAM_1 and TAM_2 are bottleneck TAMs for SI tests. Therefore, the bottleneck TAM set for this schedule is composed of TAM_1 and TAM_2.

In the $IdentifyBtnTAMs$ algorithm, every SI test involving a core on the TAM with the longest SI test time is checked for its bottleneck TAM . Therefore, the complexity of this algorithm is $O(N_{SI}N_c)$.

Algorithm for Problem P_{SI_opt}. Next we introduce our algorithm for Problem P_{SI_opt}. Similar to the $TR-Architect$ algorithm, we first create an initial TestRail architecture and optimize it by merging TAMs and distributing free TAM wires afterwards. There are two key questions during the optimization process, namely, "How to find out the merging candidate and merge them?" and "How to distribute free TAM wires?". Because there may exist multiple bottleneck TAMs at the same time in our problem, the answers to these two questions highlight the main differences between our algorithm and the $TR-Architect$ algorithm proposed in [31].

The procedure for distributing free TAM wires is shown in Fig. 3.10. The procedure takes the given TestRail architecture R_{SOC}, all the SI tests S_{SOC} and the number of free TAM wires *numFreeWires* as inputs. The free TAM wires are distributed iteratively to the bottleneck TAMs (Lines 2-6). Since we may have multiple bottleneck TAMs at the same time, we select one of them based on the criteria that T_{SOC} is the minimum after obtaining the extra TAM wire (Line 4). Because R_{SOC} is changed whenever a free TAM wire is assigned (Line 5), $time_{si}(r)$ and $time_{used}(r)$ for every $r \in R_{SOC}$ are updated (Line 6). Finally the procedure outputs the new TestRail architecture R'_{SOC} with all free TAM wires assigned.

Algorithm 5 - distributeFreeWires

```
INPUT: $R_{SOC}, S_{SOC}, numFreeWires$
OUTPUT: $R'_{SOC}$

1. initialize $R'_{SOC} = R_{SOC}$;
2. for i=1 to $numFreeWires$ {
3.     $R'_{btn} = IdentifyBtnTAMs(R'_{SOC}, S_{SOC})$;
4.     find $r' \in R'_{btn}$ such that
.              $T'_{SOC}$ is the minimum when $width(r') = width(r') + 1$;
5.     $width(r') = width(r') + 1$; update $time_{in}(r')$;
6.     update $time_{si}(r)$ and $time_{used}(r)$ for all $r \in R'_{SOC}$;
. }
7. return $R'_{SOC}$.
```

Figure 3.10: Procedure for distributing free TAM wires.

Let us take the test architecture and test schedule shown in 3.5(a) as an example to explain the *distributeFreeWires* algorithm. Suppose there is one more free wire to be distributed to one of the three TAMs. The algorithm tries to distribute it to either TAM_1 or TAM_2 (they are the bottleneck TAMs), compares the overall SOC testing time for the two choices, and finally selects the one with smaller testing time.

To distribute a free wire, we need to identify all bottleneck TAMs at the current time and schedule the SI tests with one of the bottleneck TAMs added with one more wire, which means that we need to run SI test scheduling $O(N_{TAM})$ times in the worst case. Therefore, if the number of free wires is N_{FW}, the overall complexity of the *distributeFreeWires* procedure is $O(N_{FW}N_{SI}N_cN_{TAM}^2)$.

The procedure for merging TAMs is shown in Fig. 3.11. In this procedure, with the given TestRail architecture R_{SOC}, all the SI tests S_{SOC} and one of the merging candidate r_1 as inputs, we look for another TAM candidate in $R_{find} = R_{SOC} \setminus \{r_1\}$, which leads to the lowest testing time after merging with r_1. After initialization (Lines 1 and 2), we enumeratively try every TAM r_i in R_{find} as the other merging candidate (Lines 3-14). What's more, we also try to merge r_i and r_1 with different TAM width in the range of $width_{min} = \max\{width(r_i), width(r_1)\}$ (Line 4) and $width_{max} = width(r_i) + width(r_1)$ (Line 5). The intuition behind this is that we may be able to merge two TAMs with less TAM width and the extra free TAM wires can be assigned to other bottleneck TAMs to reduce T_{SOC}. The procedure outputs the TestRail architecture R'_{SOC} with the lowest testing time after merging (Line 15). It is also possible that we cannot find a merging plan to reduce T_{SOC}. In such case, the original TestRail architecture is returned.

Again, let us take the test architecture and test schedule shown in 3.5(a) as an example to explain the *mergeTAMs* procedure. Consider the case that TAM_1 is the candidate TAM and we need to find another TAM to be merged with TAM_1 and re-distribute the TAM wires. We try each and every one of the other TAMs (TAM_2 and TAM_3 in this case) to be merged with TAM_1. If TAM_1 is merged with TAM_2, the newly merged TAM, TAM_{12}, will be assigned $\max\{width(r_1), width(r_2)\}$ wires initially. By now there are two TAMs (TAM_{12} and TAM_3) and $width(r_1) + width(r_2) - \max\{width(r_1), width(r_2)\}$ free wires. We then distribute these free wires to the two TAMs one wire at a time to achieve maximum total test time reduction. The merging of TAM_1 and TAM_3 is similar to the above procedure. We will then select the merging with smaller test time and the corresponding TAM architecture will be generated from the *mergeTAMs* algorithm.

In the *mergeTAM* procedure, to find a TAM to be merged with the candidate

Algorithm 6 - mergeTAMs

INPUT: R_{SOC}, S_{SOC}, r_1

OUTPUT: R'_{SOC}

```
1. initialize $R'_{SOC} = R_{SOC}$; $R_{find} = R_{SOC} \setminus \{r_1\}$;
2. initialize $T_{min} = T_{SOC}$
3. for i=1 to $|R_{find}|$ {
4.     set $width_{min} = \max\{width(r_i), width(r_1)\}$;
5.     set $width_{max} = width(r_i) + width(r_1)$;
6.     set $maxFreeWires = width_{max} - width_{min}$;
7.     set $R_{find} = R_{find} \setminus \{r_i\}$;
8.     for numFreeWires=0 to $maxFreeWires$ {
9.         set $r_{temp} = r_i \cup r_1$; $width(r_{temp}) = maxFreeWires - numFreeWires$;
10.        set $R_{temp} = R_{find} \cup \{r_{temp}\}$;
11.        $R_{temp} = distributeFreeWires(R_{temp}, S_{SOC}, numFreeWires)$;
12.        if($T_{temp} \leq T_{min}$) {
13.            $T_{min} = T_{temp}$;
14.            $R'_{SOC} = R_{temp}$;}
.      }
. }
15. return $R'_{SOC}$.
```

Figure 3.11: Procedure for merging TAMs.

TAM for maximum test time reduction, we need to try all other TAMs (i.e., $N_{TAM} - 1$ times). For each of these TAMs, we need to call the *distributeFreeWires* procedure multiple times (the worst case complexity is $O(W_{max})$ times). Since the worst case complexity for the utilized *distributeFreeWires* procedure is $O(W_{max}N_{SI}N_cN_{TAM}^2)$, the overall complexity of the *mergeTAM* procedure is $O(W_{max}^2N_{SI}N_cN_{TAM}^3)$.

The pseudocode for our top-level algorithm *TAM_Optimization* for Problem P_{SI_opt} is presented in Fig. 3.12, which is adapted from the $TR-Architect$ algorithm [31]. First, we create a start solution (Lines 1-16). This mainly consists of three steps. In Step 1 (Lines 2-5), we assign each core to a one-bit wide TAM and we calculate the testing time of core internal logic $time_{in}(r)$, the testing time of interconnects $time_{si}(r)$ and the actual utilized testing time $time_{used}(r)$ for every $r \in R_{SOC}$. In case $W_{max} < |R_{SOC}|$, we do not have enough TAM wires and hence we need to merge TAMs together (Lines 7-13). We first sort R_{SOC} based on the total utilized testing time in each TAM (Line 9), then $r_{W_{max}+1}$ is merged iteratively with another TAM r_i. We select this merging candidate r_i based on the criteria that T_{SOC} is the minimum after merging with $r_{W_{max}+1}$ (Line 10). Since R_{SOC} is changed after merging, $time_{si}(r)$ and $time_{used}(r)$ for every $r \in R_{SOC}$ are updated (Line 13). In the case $W_{max} > |R_{SOC}|$, we have extra free TAM wires left and procedure *distributeFreeWires* is called to distribute them.

Next, we optimize the TAM architecture by merging the TAM with the lowest $time_{used}$ with another TAM (Lines 17-23). We first sort R_{SOC} in non-increasing order and we select $r_{|R_{SOC}|}$ as one of the merging candidate r_1, then we call procedure *mergeTAMs* to search for another TAM to merge with r_1 and possibly redistribute TAM resources to reduce T_{SOC}. This is an iterative procedure and it stops when no reduction in T_{SOC} can be achieved (Lines 22-23). Afterwards, we try to further optimize the TAM architecture by trying to merge the TAM with the longest $time_{used}$ with another TAM (Lines 25-30) and merging other TAMs (Lines 31-36). Finally, *TAM_Optimization* tries to minimize T_{SOC} by iteratively moving one core from bottleneck TAMs of the SOC to another TAM, if possible (Line 37).

Algorithm 7 - TAM_Optimization

INPUT: C_{SOC}, W_{max}, S_{SOC}

OUTPUT: R_{SOC}

```
1. initialize $R_{SOC} = \emptyset$;
.  /* Create a start solution */
2. for all $c_i \in C_{SOC}$ {
3.     create TAM $r_i$ such that $C(r_i) = \{c_i\}$;
4.     $width(r_i) = 1; time_{in}(r_i) = time_{in}(c_i)$;
5.     $R_{SOC} = R_{SOC} \cup \{r_i\}$;
.  }
6. calculate $time_{si}(r)$ and $time_{used}(r)$ for all $r \in R_{SOC}$;
7. if ($W_{max} < |R_{SOC}|$) {
8.     for($i = W_{max} + 1$ to $|R_{SOC}|$) {
9.         sort $R_{SOC}$ such that $time_{used}(r_1) \geq$
.              $time_{used}(r_2) \geq ... \geq time_{used}(r_{|R_{SOC}|})$;
10.        find $r_i (1 \leq i \leq W_{max})$ such that
.              $T_{SOC}$ is the minimum when merging with $r_{W_{max}+1}$;
11.        $r_i = r_i \cup r_{W_{max}+1}$; update $time_{in}(r_i)$;
12.        $R_{SOC} = R_{SOC} \setminus r_{W_{max}+1}$;
13.        update $time_{si}(r)$ and $time_{used}(r)$ for all $r \in R_{SOC}$;
.      }
14. } else if ($|R_{SOC}| < W_{max}$) {
15.    set $numFreeWires = W_{max} - |R_{SOC}|$;
16.    $R_{SOC} = distributeFreeWires(R_{SOC}, S_{SOC}, numFreeWires)$;
.  }
.  /* Optimize the TestRail architecture from bottom-up */
17. set $isImproved = true$;
18. while($isImproved$ AND $|R_{SOC}| > 1$) {
19.   set $initTestingTime = T_{SOC}$;
20.   sort $R_{SOC}$ such that $time_{used}(r_1) \geq time_{used}(r_2) \geq ... \geq time_{used}(r_{|R_{SOC}|})$;
21.   $R_{SOC} = mergeTAMs(R_{SOC}, S_{SOC}, r_{|R_{SOC}|})$;
22.   if($T_{SOC} == initTestingTime$) {
23.      $isImproved = false$; }
.  }
.  /* Optimize the TestRail architecture from top-down*/
24. set $isImproved = true$;
25. while($isImproved$ AND $|R_{SOC}| > 1$) {
26.   set $initTestingTime = T_{SOC}$;
27.   sort $R_{SOC}$ such that $time_{used}(r_1) \geq time_{used}(r_2) \geq ... \geq time_{used}(r_{|R_{SOC}|})$;
28.   $R_{SOC} = mergeTAMs(R_{SOC}, S_{SOC}, r_1)$;
29.   if($T_{SOC} == initTestingTime$) {
30.      $isImproved = false; R_{skip} = \{r_1\}$; }
.  }
31. while($R_{skip} \neq R_{SOC}$) {
32.   set $R_{temp} = R_{SOC} \setminus R_{skip}$; $initTestingTime = T_{SOC}$;
33.   find $r^* \in R_{temp}$ such that $time_{used}(r^*)$ is the maximum;
34.   $R_{SOC} = mergeTAMs(R_{SOC}, S_{SOC}, r^*)$;
35.   if($T_{SOC} = initTestingTime$) {
36.      $R_{skip} = R_{skip} \cup \{r_1\}$; }
.  }
37. $coreReshuffle(R_{SOC}, S_{SOC})$;
38. return $R_{SOC}$, $T_{SOC}$.
```

Figure 3.12: Algorithm for solving P_{SI_opt}.

As can be observed in Fig. 3.12, the computational complexity for the algorithm of *TAM_Optimization* is mainly determined by the bottom-up and top-down optimization procedures, which require the *mergeTAM* procedure to be carried out $O(W_{max})$ times in the worst case. Therefore, the complexity of the overall algorithm is $O(W_{max}^3 N_{SI} N_c N_{TAM}^3)$.

3.5 Experimental Results

To evaluate the effectiveness of the proposed solution, experiments were carried out for three ITC'02 benchmark SOCs from [52], namely, g1023, p34392 and p93791. Without loss of generality, we do not consider hierarchy in the testing of core-internal logic. Since the topology of these benchmark SOCs and the connection between embedded cores are not available, we cannot obtain the test patterns for core-external interconnect SI faults for these benchmark SOCs. Therefore, we generate random test patterns for our experiments in the following manner. For the smaller SOC g1023, we generate $1,000$ and $5,000$ random patterns, respectively. For p34392 and p93791 we generate $10,000$ and $50,000$ random patterns, respectively. Each test pattern targets one victim and N_a $(2 \leq N_a \leq 6)$ random aggressors. Suppose the victim wire connects two cores $Core_a$ and $Core_b$. Then at least $N_a - 2$ aggressor lines are between these two cores. In addition, we assume that a 32-bit bus is utilized in all the three SOCs. The probability that the bus is used by a test pattern is set to 50%. If the bus is used for a particular pattern, we randomly generate $1 \sim N_a$ specified bits in the postfix of the pattern (see Section 3.3).

Table 3.2 shows the results for our two-dimensional test compaction scheme. We partition the SOCs in N_g parts using the *hMetis* package [75]. Therefore, the row with $N_g = 1$ is for the case when the test set is compressed without partitioning. The parameters N_c and D_s denote the compacted test pattern count and test data

Table 3.2: Comparison of compressed test data volume with different SI test pattern counts and different core groups

		N_r = 1,000			N_r = 5,000		
	N_g	N_c	D_s	ΔD_s (%)	N_c	D_s	ΔD_s (%)
	1	35	383950	/	156	1711320	/
SOC g1023	2	39	324230	15.55	161	1347898	21.24
	4	47	314018	18.21	170	1235982	27.78
	8	58	**311198**	18.95	192	**1233058**	27.95
		N_r = 10,000			N_r = 50,000		
	N_g	N_c	D_s	ΔD_s (%)	N_c	D_s	ΔD_s (%)
	1	258	1486596	/	1247	7185214	/
SOC p34392	2	285	1196228	19.53	1294	5748670	19.99
	4	316	1221490	17.83	1375	5394738	24.92
	8	376	**1185938**	20.22	1555	**5371424**	25.24
		N_r = 10,000			N_r = 50,000		
	N_g	N_c	D_s	ΔD_s (%)	N_c	D_s	ΔD_s (%)
	1	270	5750460	/	1266	26963268	/
SOC p93791	2	285	4715026	18.01	1304	21219284	21.30
	4	308	4355544	24.26	1355	19734208	26.81
	8	317	**3944650**	31.40	1401	**19067892**	29.28

N_r: Initial interconnect test pattern count; N_g: Number of partitions;

N_c: Number of compacted patterns; D_s: Test data volume;

$\Delta D_s = \frac{D_s(N_g=1)-D_s}{D_s(N_g=1)} \times 100\%$.

volume (calculated as the sum of the test pattern length times the test pattern count in each SI test group), respectively. ΔD_s is the percentage reduction in test data volume compared to the case when $N_g = 1$. It can be observed from the table that, with test pattern merging only, the compaction over the original test set is only $\Delta V = 3\%$ (i.e., $\frac{N_c}{N_r} \times 100\%$ when $N_g = 1$). With test-pattern-length reduction by SI test grouping, we are able to further reduce the test data volume by more than 20 % on top of ΔV.

Tables 3.3, 3.4 and 3.5 present results for the SOC test application time, measured in terms of the number of clock cycles. We compare between the following cases: (i) optimizing T_{SOC}^{in} using only the $TR-Architect$ algorithm [31] ($T_{TR-Arch}$); (ii) optimizing T_{SOC} using our proposed algorithm $TAM\_Optimization$ for several SI test pattern counts N_r and the SI test grouping strategy. Note that $T_{[31]}$ is determined by optimizing the SOC TAM architecture in terms of core-internal test time T_{SOC}^{in} only, and then computing the total test time T_{SOC} by adding to T_{SOC}^{in} the time needed for SI test. The parameter T_{g_i} denotes the SOC test time obtained using the proposed $TAM\_Optimization$ algorithm when the SI tests are partitioned into i parts; $T_{min} = \min_i\{T_{g_i}\}$, which corresponds to the test-architecture that we choose; $\Delta T_{TR-Arch}$ and ΔT_g are computed as $\Delta T_{TR-Arch} = \frac{T_{TR-Arch} - T_{min}}{T_{TR-Arch}} \times 100\%$ and $\Delta T_g = \frac{T_{g_1} - T_{min}}{T_{g_1}} \times 100\%$, respectively. Note that ΔT_g quantifies the benefit derived from our two-dimensional compaction strategy over the one-dimensional compaction scheme that reduces only the test-pattern count. We can see that more than 20% test-time reduction can be achieved in some cases (e.g., for SOC g1023, when $W_{max} = 64$ and $N_r = 5,000$). The magnitude of this reduction depends on the initial SI test set and the core configurations. It should be noted that the maximum value of N_g does not necessarily lead to minimum testing time. This is mainly because, when we have a large value of N_g, we have more SI tests to schedule and conflicts are more likely to arise during the scheduling process, thus leading

Table 3.3: Test application time comparison for SOC g1023

SOC g1023								
	$N_r = 1,000$							
W_{max}	$T_{TR-Arch}$ (cc)	T_{g_1} (cc)	T_{g_2} (cc)	T_{g_4} (cc)	T_{g_8} (cc)	T_{min} (cc)	$\Delta T_{TR-Arch}$ (%)	ΔT_g (%)
8	119791	93913	92723	94519	**92708**	92708	22.61	1.28
16	70161	**50349**	52073	50842	50840	50349	28.24	0.00
24	44603	**33652**	35038	34998	33900	33652	24.55	0.00
32	35145	26196	27290	26045	**24895**	24895	29.16	4.97
40	28619	22697	22672	**21163**	21619	21163	26.05	6.76
48	28619	20989	20618	18920	**18184**	18184	36.46	13.36
56	28619	20986	20618	19255	**19213**	19213	32.87	8.45
64	28619	20941	20618	**18998**	19139	18998	33.62	9.28
	$N_r = 5,000$							
W_{max}	$T_{TR-Arch}$ (cc)	T_{g_1} (cc)	T_{g_2} (cc)	T_{g_4} (cc)	T_{g_8} (cc)	T_{min} (cc)	$\Delta T_{TR-Arch}$ (%)	ΔT_g (%)
8	296814	182173	175354	160032	**155091**	155091	47.75	14.87
16	197211	92026	103685	84900	**80199**	80199	59.33	12.85
24	111395	67172	60975	59042	**56124**	56124	49.62	16.45
32	100249	51278	43629	43750	**43261**	43261	56.85	15.63
40	98098	40350	39222	38528	**35343**	35343	63.97	12.41
48	98098	35835	32088	31921	**30212**	30212	69.20	15.69
56	98098	32780	31435	**28758**	29737	28758	70.68	12.27
64	98098	32260	29947	**25770**	27246	25770	73.73	20.12

N_r: Initial interconnect test pattern count; W_{max}: Given SOC TAM width;

$T_{TR-Arch}$: Test time obtained by optimizing the SOC TAM architecture for InTest only;

T_{g_i}: Test time obtained using the proposed *TAM_Optimization* algorithm

with the SOC cores partitioned into i groups;

$T_{min} = \min_i\{T_{g_i}\}$; $\Delta T_{TR-Arch} = \frac{T_{TR-Arch}-T_{min}}{T_{TR-Arch}} \times 100\%$; $\Delta T_g = \frac{T_{g_1}-T_{min}}{T_{g_1}} \times 100\%$.

Table 3.4: Test application time comparison for SOC p34392

SOC p34392								
	$N_r = 10,000$							
W_{max}	$T_{TR-Arch}$ (cc)	T_{g_1} (cc)	T_{g_2} (cc)	T_{g_4} (cc)	T_{g_8} (cc)	T_{min} (cc)	$\Delta T_{TR-Arch}$ (%)	ΔT_g (%)
8	2499024	2460860	2189687	2154845	**2153117**	2153117	13.84	12.51
16	1215631	1100901	1157715	**1094507**	1097405	1094507	9.96	0.58
24	863107	801474	828945	816120	**793520**	793520	8.06	0.99
32	644122	612926	636092	674518	**609820**	609820	5.33	0.51
40	604177	583720	564348	**563295**	601969	563295	6.77	3.50
48	604177	563253	**558339**	559762	558829	558339	7.59	0.87
56	604177	561607	**556440**	556910	556972	556440	7.90	0.92
64	604177	561607	556404	556430	**555728**	555728	8.02	1.05
	$N_r = 50,000$							
W_{max}	$T_{TR-Arch}$ (cc)	T_{g_1} (cc)	T_{g_2} (cc)	T_{g_4} (cc)	T_{g_8} (cc)	T_{min} (cc)	$\Delta T_{TR-Arch}$ (%)	ΔT_g (%)
8	2862976	2582663	2685154	2523919	**2477706**	2477706	13.46	4.06
16	1436474	1323698	1313744	**1312050**	1385386	1312050	8.66	0.88
24	1120118	1035895	950950	**910648**	955818	910648	18.70	12.09
32	872581	720887	714779	**700121**	704810	700121	19.76	2.88
40	832636	657092	652525	**633428**	641855	633428	23.92	3.60
48	832636	638850	624037	**607619**	610755	607619	27.02	4.89
56	832636	619399	607829	**599433**	600075	599433	28.01	3.22
64	832636	619399	601601	**593223**	600177	593223	28.75	4.23

Table 3.5: Test application time comparison for SOC p93791

SOC p93791								
	$N_r = 10,000$							
W_{max}	$T_{TR-Arch}$ (cc)	T_{g_1} (cc)	T_{g_2} (cc)	T_{g_4} (cc)	T_{g_8} (cc)	T_{min} (cc)	$\Delta T_{TR-Arch}$ (%)	ΔT_g (%)
8	4695241	4064748	4144176	**4034628**	4075377	4034628	14.07	0.74
16	2298848	2072745	2029390	**2018862**	2037524	2018862	12.18	2.60
24	1622729	1412641	**1392753**	1434277	1392762	1392753	14.17	1.41
32	1217985	1039646	1030373	1057084	**1026253**	1026253	15.74	1.29
40	1041244	841096	**829314**	856368	832855	829314	20.35	1.40
48	959503	708385	713458	710631	**697217**	697217	27.34	1.58
56	808771	606411	600039	611748	**590176**	590176	27.03	2.68
64	784023	528642	527789	521335	**497329**	497329	36.57	5.92
	$N_r = 50,000$							
W_{max}	$T_{TR-Arch}$ (cc)	T_{g_1} (cc)	T_{g_2} (cc)	T_{g_4} (cc)	T_{g_8} (cc)	T_{min} (cc)	$\Delta T_{TR-Arch}$ (%)	ΔT_g (%)
8	6037849	5407578	5106050	5124448	**5044333**	5044333	16.45	6.72
16	3286880	3134458	2579717	2588326	**2506449**	2506449	23.74	20.04
24	2374697	1825802	1782340	1759197	**1700872**	1700872	28.38	6.84
32	1943073	1432611	1405361	1389043	**1344381**	1344381	30.81	6.16
40	1485965	1124129	1138165	**1075325**	1080395	1075325	27.63	4.34
48	1522634	945505	923041	906744	**883350**	883350	41.99	6.57
56	1665028	836821	814316	824757	**751578**	751578	54.86	10.19
64	1902274	708727	675423	707474	**654393**	654393	65.60	7.67

to longer testing time.

From Tables 3.3, 3.4 and 3.5, we note that obliviously optimizing SOC test-architectures, without considering interconnect SI faults, leads to much higher test time. This gap grows with an increase in the pattern count for the SI faults and the associated percentage of SI testing time in T_{SOC}. We can also see that when W_{max} is small, there is no significant advantage in using proposed algorithm; in a few cases, worse results are obtained compared to SI-oblivious TAM optimization (e.g., for SOC p34392, when $W_{max} = 8$ and $N_r = 10,000$). This is mainly because the TAM design solution space is small for smaller values of W_{max}, therefore similar TAM architectures are obtained with different optimization criterion. When W_{max} is higher, we have more freedom during the TAM design process and hence the improvement offered by the new optimization procedure is more noticeable. We can also observe that, for SOC p34392, when $W_{max} > 32$, T_{min} remains nearly the same. This is because the testing time for $Core_{18}$, the largest embedded core, dominates T_{SOC}.

We attribute the few exceptions to the nature of the heuristics that explore a limited part of the solution space.

When the number of SI test pattern grows, it is more important to optimize the test-architecture for both core-internal faults and interconnect SI faults. In Figure 3.13, we vary the original SI test-pattern count while keeping the TAM width at 32 bits. We compare the test time obtained using the proposed method with the test time for the baseline method based on [31]. The number of (given) SI test patterns is increased from 100 to 5,000 for SOC g1023, and from 1,000 to 50,000 for SOC p34392 and p93791, respectively. It can be observed that the gap between the two solutions becomes larger when the number of SI test patterns increases, which highlights the importance of optimizing the SOC test-architecture for interconnect SI faults for newer technology generations.

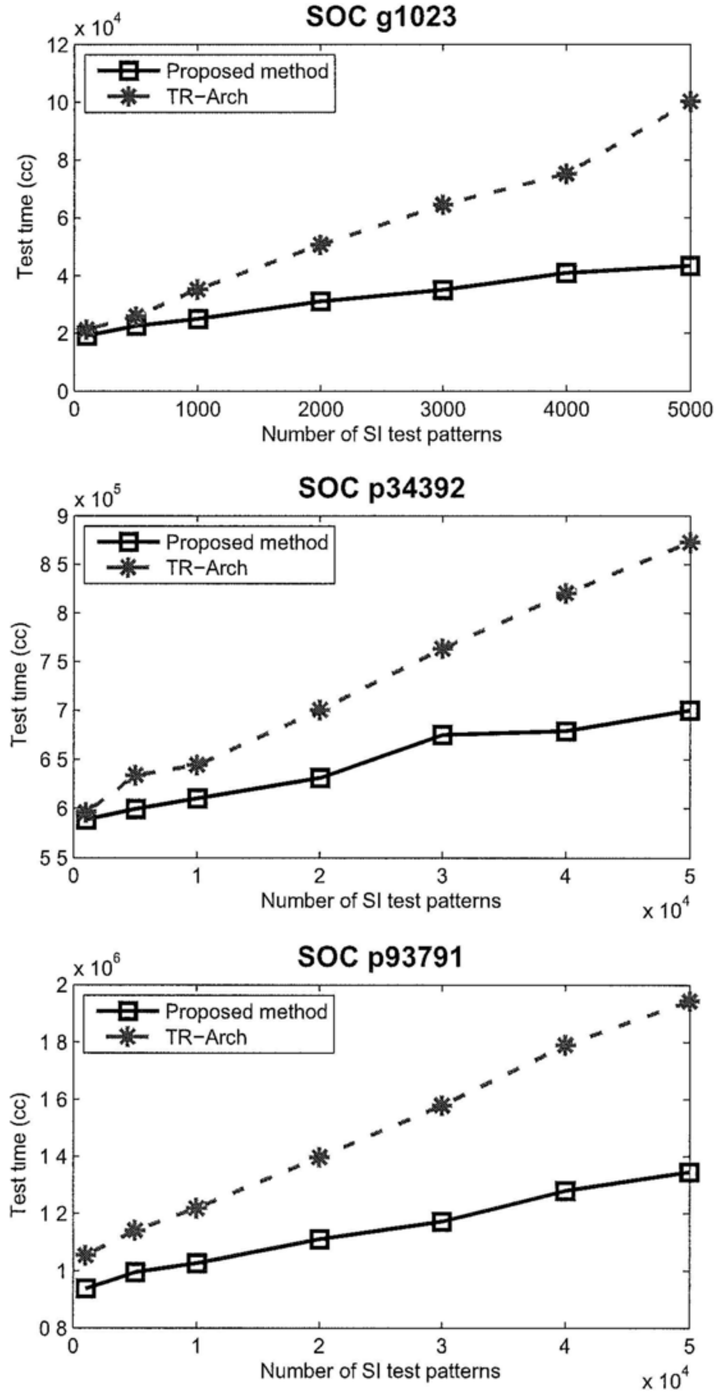


Figure 3.13: Test time comparison with different SI test pattern counts.

3.6 Conclusion

As feature sizes shrink with newer process technologies and clock frequencies increase, the test cost due to interconnect signal integrity faults can be considerable. To cope with this problem, we have presented a new TAM optimization flow for core-based SOCs, which considers test times for both core-internal logic and core-external signal integrity faults on interconnects. This is in contrast to prior work on test infrastructure design for core-based system-on-a-chip, which has focused on minimizing only the test time for core-internal logic. We have investigated the impact of interconnect SI tests on SOC test-architecture design and optimization. We have also presented a compaction method for SI test sets such that the test data volume is reduced. Experimental results for the ITC'02 benchmarks show that the proposed approach can significantly reduce the overall testing time for core-internal logic and core-external interconnects. The test times obtained using this approach are noticeably less than that obtained by a baseline based on the $TR-Architect$ algorithm, which only considers the core-internal test time during optimization. As part of future work, we are considering the role of different core frequencies for reducing the test time [91]. We are also investigating how interconnect layout information can be used for more effective test-infrastructure optimization.

□ **End of chapter.**

Chapter 4

Testing of Harmful PDN Defects

Power distribution network (PDN) designs for today's high performance integrated circuits (ICs) typically occupy a significant share of metal resources in the circuit, and hence defects may be introduced on PDNs during the manufacturing process. Since we cannot afford to over-design the PDNs to tolerate all possible defects, it is necessary to conduct manufacturing test for them. This chapter proposes novel methodologies to identify those potentially harmful open defects in PDNs and we show how to select a set of patterns that initially target transition faults to achieve high fault coverage for the PDN defects. Experimental results on benchmark circuits demonstrate the effectiveness of the proposed technique.

4.1 Introduction

For integrated circuits (ICs) fabricated with advanced semiconductor technology, the on-chip power distribution network (PDN) is responsible for providing accurate-enough power and ground voltages to every logic standard cell (SC) [97]. With the ever increasing circuit complexity, PDN designs for high-performance ICs have become increasingly challenging [45, 54] and they typically occupy a considerable part of metal resources in the circuit, e.g., 28% for a recent microprocessor [2].

Because of the above, it is very likely that defects are introduced on PDN during the manufacturing process [47, 76]. The total number of possible PDN defect locations is enormous and it is impossible to generate test for every one of them. Fortunately, typical PDN is essentially a defect-tolerant structure and a large number of PDN defects are in fact harmless, especially when the PDN is over-designed. However, it is extremely costly, if not impossible, to strengthen the PDN tolerating all possible defects, since such PDN designs would waste lots of crucial routing resources that can be used to improve the circuit performance.

Today, for those harmful PDN defects that cannot be tolerated, designers mainly rely on test patterns for stuck-at faults and delay faults to identify them implicitly. While the above implicit test strategy works quite well for relatively simple designs previously, with technology advances, PDN defects may cause substantial timing errors for high-performance ICs, which cannot be detected by regular delay tests. This is because: (i). the power supply noise margin of IC devices gets smaller and smaller with decreasing supply voltage in each new technology generation [14, 23], and circuit performance suffer to more extent from the power supply degradation. For example, it was shown that a 1% change for the supply voltage can result in nearly 4% change in circuit delay with 90nm technology [82]; (ii). the worst-case power supply caused by PDN defects occurs when the local switching activities surrounding the defective position are the maximum, which leads to severe power droop [6, 7, 79]; On the other hand, conventional test patterns that target for delay faults try to sensitize the logic paths so that the targeted delay error effects can be observed [46, 73], which may not activate intensive transitions near the PDN defect locations.

It is therefore essential to identify those potentially harmful defects on the PDN and apply test patterns to detect them, ensuring the quality of the shipped products.

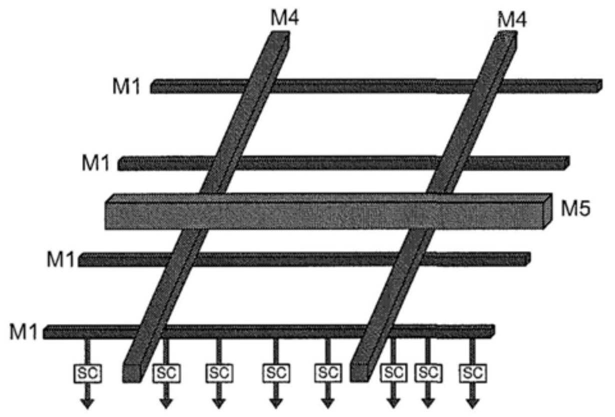


Figure 4.1: An example power distribution network.

Due to the huge number of possible PDN defect locations, this is an interesting and challenging problem. In this work, we first present novel techniques to efficiently extract a small set of vulnerable PDN locations, which contains all the potentially harmful defects. Secondly, we propose to select a set of 1-detect transition fault patterns out of the N-detect patterns [65] so that we are able to achieve high PDN defect coverage without increasing the test pattern count.

4.2 Preliminaries and Motivation

4.2.1 PDN Structure and Modeling

Power distribution network is typically designed with a hierarchical structure, which traverses multiple metal layers in the circuit. Fig. 4.1 depicts a typical power network structure, which involves three metal layers (M1, M4, and M5 from bottom up). The wires on different metal layers are linked together through vias, and they are eventually connected to the power pads at the uppermost layer. The devices are arranged below M1 layer in rows, and obtain power supply from power wires on M1 layer. It can be easily observed that, PDN with such infrastructure is to a great extent defect-tolerant since the device can acquire power supply through multiple paths from the power pad. The ground network of the PDN, which is the counterpart of the power network, is with similar structure and therefore is not shown in Fig. 4.1. Here in this chapter, for clarity only the analysis for power network will be presented, while ground network is not mentioned because of the similarity unless it is with necessary to show the difference.

As can be observed from Fig. 4.1, the PDN structure on M1 metal layer typically consists of a group of parallel tracks while each track can be viewed as a series of wire segments, wherein a segment refers to a part of PDN wire on M1 between two vias, called *M1 segment* hereafter. An example of such M1 segments is shown in Fig. 4.2, which are marked as S_{i-1}, S_i and S_{i+1}. The open defects may occur on a via or a wire segment and an example open defect is demonstrated on Fig. 4.2. With the existence of PDN open defects, those SCs that normally draw current from the nearby wire/via would be forced to get power supply through longer paths. This can change the power supply topology and may induce insufficient supply voltage for certain SCs.

To model the PDN for analysis, let $\mathbf{G} = \{G_{i,j}\}$ be the conductance matrix of the PDN, namely *PDN matrix*, wherein $G_{i,j}$ represents the conductance between node i and j on PDN. In addition, $\mathbf{V} = \{v_j\}$ and $\mathbf{I} = \{i_j\}$ represent the voltage vector and the current vector of all the nodes, respectively. The analytical model for the PDN can then be expressed as $\mathbf{GV} = \mathbf{I}$. Here, $\mathbf{G}$ is extracted from layout information, and $\mathbf{I}$ corresponds to switching activity. The power supply voltage distribution on PDN can then be achieved by solving these linear equations [97], called PDN simulation in this chapter.

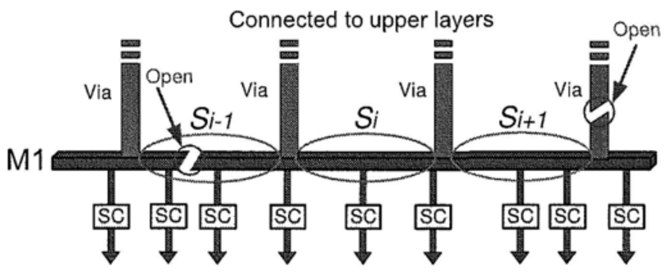


Figure 4.2: Segmented structure for PDN tracks on M1 layer and possible open defects.

4.2.2 Prior Work and Motivation

There are only limited works on PDN testing in the literature. Sato *et al.* [70] proposed a so-called box-scan method to identify weak positions in power distribution networks. Their method, however, is not applicable for industrial designs due to the high computational complexity. Recently, Ma *et al.* [47] proposed an automatic test pattern generation (ATPG) method for open defects on PDN. To reduce the problem complexity, this work proposes to divide the power distribution network into a number of regions and tries to activate as many switching activities in a region as possible for test generation. This strategy can be very effective for testing defects on high metal layers as they affect a relatively larger region at the circuit level. For defects on low metal layers (especially M1), however, their impacts are localized and activating switching activities for a sizable region may not be able to identify the error. As there are significantly more PDN defects on the denser low metal layers than that on higher ones, it is essential to develop new test methodologies for such defects, which motivates this work. In particular, we focus on open defects on the lowest PDN metal layer (say, M1) and vias that connect it to its neighboring PDN metal layer (say, M2) in this work, which usually covers more than 95% of the total possible PDN defects.

As discussed earlier, a great share of PDN defects are harmless for the design due to the defect-tolerance nature of PDN structure. On the other hand, because of the huge number of possible PDN defects, it is very difficult, if not impossible, to evaluate their effects by simulating them one by one. As a result, it is essential to identify those potentially harmful PDN defects in an effective and efficient manner. Then, to detect these harmful defects, we can either generate dedicated test patterns for them or select a set of delay test patterns to cover them implicitly. In this work, we take the latter approach and try to select a set of 1-detect transition fault patterns out of N-detect patterns that are able to achieve high PDN defect coverage.

It should be highlighted that simply increasing the switching activities of the test patterns (e.g., by X-filling), although effective for the detect of PDN defects, is not a good solution. This is because: test patterns with prohibitively-large switching activities can result in serious over-testing problem [43, 71], since it increases the power supply noise of the circuit to an extent that cannot occur in functional mode and may reject some good chips that would work in application, thus leading to yield loss due to test overkills.

4.3 Potentially Harmful PDN Defects Identification

In today's large ICs, there are millions of wire segments in the PDN and hence the number of possible open defects is enormous. It is very difficult, if not impossible, to evaluate the impact of all these defects by PDN simulation because of the extremely high time complexity. Based on the observation that most of these defects can be tolerated by the PDN infrastructure itself, we propose an efficient and effective approach to identify those PDN defects that *might* result in errors, which are referred to as *potentially harmful defects* in this chapter.

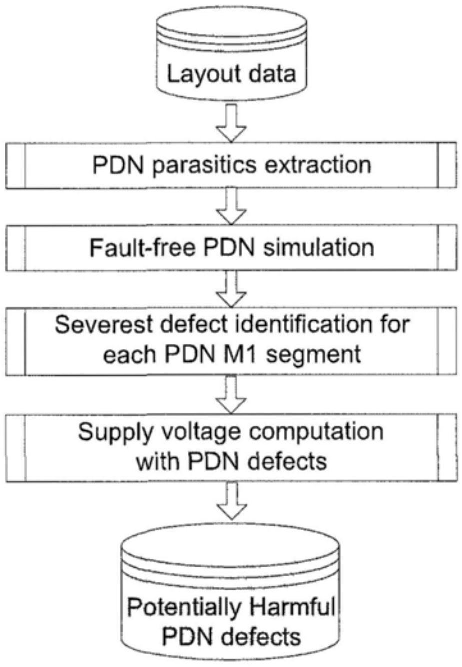

Layout data
PDN parasitics extraction
Fault-free PDN simulation
Severest defect identification for each PDN M1 segment
Supply voltage computation with PDN defects
Potentially Harmful PDN defects

Figure 4.3: Flowchart of the proposed method for identifying potentially harmful PDN defects.

4.3.1 Overall Flow

The main concern for the harmful PDN defect identification process is its time complexity. To handle this problem, we should perform the time-consuming PDN simulation process as few times as possible. The flowchart of the proposed harmful defect identification method is shown in Fig. 4.3 while the details are shown as follows.

PDN parasitics extraction: In this step, the parasitic information of the whole PDN is extracted from the circuit layout files. The extracted parasitic information is then transformed to feed the linear equation solver for PDN simulation. With N nodes in the PDN, a $N \times N$ matrix $\mathbf{G}$ need to be built. In addition, the mapping between the gates at logic level and the PDN nodes extracted from the layout information should also be constructed to build the current vector $\mathbf{I}$. The PDN analysis expressed as $\mathbf{GV} = \mathbf{I}$ can then be solved to produce the supply voltage distribution result $\mathbf{V}$.

Defect-free PDN simulation: To analyze the PDN system, we build a PDN simulator based on a linear equation solver [66, 67] to obtain the voltage of every node according to the relationship $\mathbf{GV} = \mathbf{I}$. The defect-free PDN can then be analyzed by the PDN simulator while the result will be further used in the following steps.

Severest defect identification for each PDN M1 segment: As the currents go through vias to M1 tracks of PDN and then to SCs, the severest open defect within each M1 segment, which usually contains tens of SCs and corresponding wires, must be on the wires next to the vias that are denoted as edge wires, since power supply to the SCs must be through the edge wires. The open defects on the edge wires are correspondingly denoted as *edge defects* while the defects on the internal wires of M1 segments are denoted as *internal defects*. The two edge defects in each M1 segment will be evaluated by PDN simulation and the worse one will be the severest defect in this segment. If the worst open defect in an M1 segment cannot induce any function error, all the defects within this segment will not be considered as harmful defects any more since they are in fact tolerable defects. Such technique can greatly reduce the time complexity for identifying harmful defects, or excluding tolerable defects, since the defect tolerance characteristic and the specific power supply structure of PDN have been taken into account.

The PDN simulation times can be further reduced by simultaneous injection of multiple independent PDN defects into the circuit at one time for simulation, instead of injecting and simulating them one by one. Here, independence indicates the property that the effective regions of any two defects in this set do not overlap with each other while the effective region of a defect includes all the PDN nodes that may be influenced by such defect. As the effect of a defect on M1 layer is typically localized in a small region, high compaction ratio in defect simulations can be achieved because each defect is independent with most of other defects except its neighboring ones.

Besides identifying the severest defect in each M1 segment, the accurate voltage result from PDN simulation will be further used to calibrate the computation of the internal defects, which is detailed in the following.

Supply voltage computation with PDN defects: Recall that we cannot afford to evaluate all the possible open defects by PDN simulation and we need to pick out the most destructive ones in an efficient manner. Instead of running the time-consuming PDN simulation to obtain the supply voltage disturbance induced by PDN defects, we propose a fast computation method for the internal defects, which requires much less computational time. The details of the proposed computation algorithm is shown in section 4.3.2.

Finally, through either simulation or computation, the power supply voltage distribution induced by each PDN defect can be obtained, with which the defects that can result in severe supply voltage droop and function error are identified as potentially harmful PDN defects.

4.3.2 Computation of Voltage Drop Induced by PDN Defects

A fast yet accurate power supply distribution computation method is proposed in this section, which is composed of two steps: calculation and calibration. Calculation step aims to quickly obtain the voltage distribution while the calibration step adjusts the calculation results segment by segment for higher accuracy. By means of the proposed computation method, only the two edge defects in each segment need to be simulated while the effects of all the internal defects can be accurately computed. As the average quantity of SCs in each segment is typically around one hundred, this strategy can significantly cut down the time consumption.

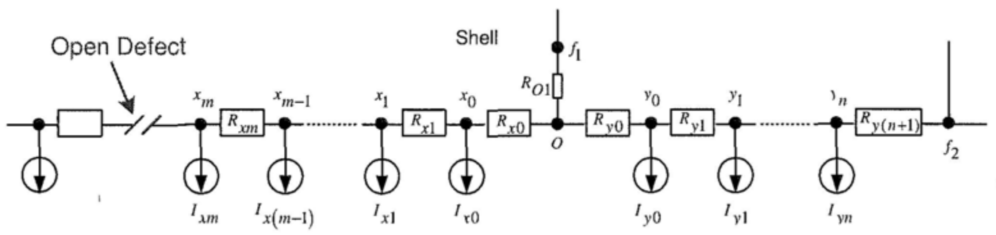


Figure 4.4: Distributed circuit model for the computation of PDN open defects induced power supply drop.

The effective region of PDN defects on M1 layer is localized in a small domain, which indicates that, compared with that of the defect-free situation, only a small part of the PDN nodes will have voltage variation after the injection of such a defect. With a shell surrounding the whole effective region, we can then perform voltage calculation only on the nodes inside the shell while all other nodes are left alone with defect-free voltage value, which can greatly reduce the number of elements involved into the calculation.

We model SCs as current sources, and the wires between two PDN nodes as resistors. The distributed circuit model is constructed accordingly for supply voltage drop calculation, and an example is shown in Fig. 4.4. Setting V_{f1} and V_{f2} at the same values as that in defect-free situation, the shell for localized calculation is actually set as the node set including node f_1, f_2 and x_m in this example. The voltage calculation for all the SCs in the two segments (denoted as segment x and y, respectively) can then be localized in the region surrounded by these three nodes.

We denote by I_L and I_R the currents flowing out of node O to the left and right directions, respectively. Since all the standard cells within segment x are fed with current from node O, we have $I_L = \sum_{i=1}^{m} I_{xi}$. The SCs within segment y, on the other hand, are fed with currents from two vias while the current from the right via is given by $I_R = \frac{V_{f2}-V_{yn}}{R_{y(n+1)}}$. As the total current drawn by the SCs in segment y is $\sum_{j=1}^{n} I_{yj}$, the current drawn from the right side of node O is therefore $I_R = \sum_{j=1}^{n} I_{yj} - \frac{V_{f2}-V_{yn}}{R_{y(n+1)}}$. With these notations, we first build relation considering the current of node O by the following equation.

$$\frac{V_{f1}-V_O}{R_{O1}} = I_L + I_R = \sum_{i=1}^{m} I_{xi} + (\sum_{j=1}^{n} I_{yj} - \frac{V_{f2}-V_{yn}}{R_{y(n+1)}}) \tag{4.1}$$

After some algebra, we express the voltage of node O as

$$V_O = V_{f1} - R_{O1}(\sum_{i=1}^{m} I_{xi} + \sum_{j=1}^{n} I_{yj} - \frac{V_{f2}-V_{yn}}{R_{y(n+1)}}) \tag{4.2}$$

With V_O, we thereafter calculate the voltage for any node within these two segments. For example, the voltage of the k^{th} node in segment x, V_{xk}, can then be expressed as

$$V_{xk} = V_O - \sum_{i=0}^{k}(R_{xi}\sum_{j=i}^{m} I_{xj}) \tag{4.3}$$

Fig. 4.5 depicts the error between the calculated and simulated values of the minimum voltage caused all the defects in one segment. We observe that the errors follow approximately segmented linear distribution. We therefore further calibrate the results obtained by the fast calculation method as follows.

(I) Simulate the two edge defects in each segment;

(II) Calculate the supply voltage drop induced by all the defects through the localized calculation method proposed in the above;

(III) Calculate the errors between the simulated and the calculated voltage values for the two edge defect;

(IV) Calculate the errors for all the internal defects with the error value of the two edge defects and the segmented linear distribution, segment by segment;

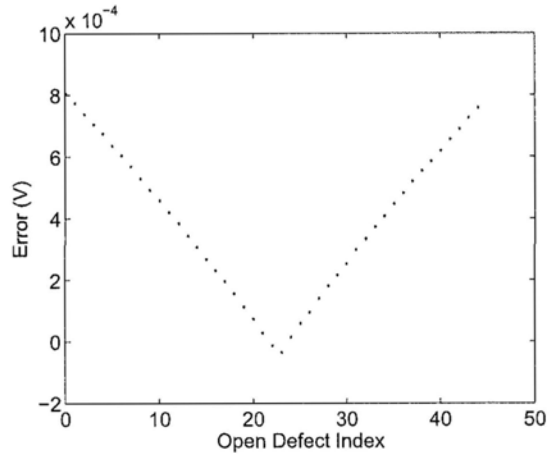


Figure 4.5: Errors between the simulated and computed values.

(V) Calibrate the initial calculation results with the errors obtained in (IV) for each internal defect.

Apparently, the proposed voltage computation method, composed of fast calculation and accurate calibration, can be finished in neglectful time compared with defect simulation because such computation is localized and can be completed straightforward while the defect simulation is performed globally with multiple iterations.

4.4 Test Pattern Sorting and Selection Methodology

To detect potential harmful PDN open defects, we need to visualize these defects with test patterns so that the chips with such defects can be screened out. In this section, we propose to select test patterns for PDN defects from the transition fault test patterns while the generation of dedicated PDN test patterns is out of the scope of this chapter and will be addressed in the future.

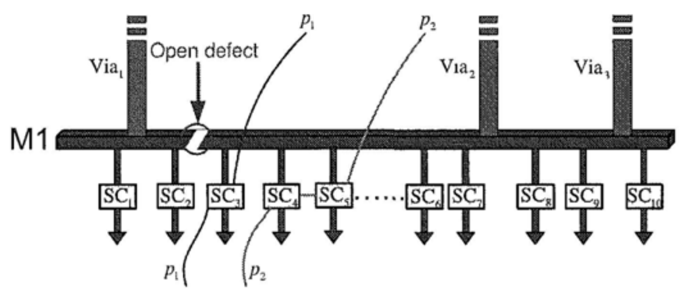


Figure 4.6: PDN defect detection via exercising transition faults in the circuit.

4.4.1 PDN Defect Detection

The existence of PDN open defects change the topology of power supply so that certain SCs cannot be fed with adequate supply voltage. Therefore, in order to detect PDN defects, it is essential for the test patterns to activate dense switching activity in the corresponding effective region to burden the power supply during testing. In addition, since we expect delay errors at the observing points if harmful PDN defects occur, test patterns should also be able to activate paths through the SCs with extra delay due to insufficient power supply induced by PDN defects. The combination of power activation and path activation at the same time can then demonstrate the existence of harmful PDN defects.

An example of PDN defect detection is illustrated in Fig. 4.6. Because of the open defect, some standard cells (e.g., SC_3 in Fig. 4.6) have to draw currents from the faraway via_2 instead of from the nearby via_1 as in the defect-free situation, which results in more power supply drop on these SCs compared with the defect-free situation. To detect the defect, we need to activate heavy current drawn from the SCs at the right side of the PDN defect with test patterns. In addition, the paths through the standard cells inside the effective region with severe power supply (i.e., p_1 and p_2) should be activated to propagate the long delay of those affected SCs out to primary outputs (POs) or pseudo-primary outputs (PPOs).

4.4.2 Overall Flow of Pattern Sorting and Selection Procedure

The flowchart of the proposed test pattern sorting and selection methodology is shown in Fig. 4.7, and the details are explained as follows.

Transition fault simulation: In this step we run transition fault simulation to identify the transition faults that can be detected by each test pattern.

PDN defect simulation: The goal of PDN defect simulation is to identify the PDN defects that can be detected by each pattern while three main tasks are needed to be performed on all the patterns one by one to achieve such purpose.

(I) Logic simulation: With a given pattern stating the logic values of Primary Inputs (PIs) and scan cells, the values of all the logic gates can be obtained by logic simulation. The transition information of all the SCs inside the circuit can then be collected. Thereafter, the exact current activated by each pattern can be known and such information can then be used to build the current vector **I**. As a result, each test pattern is with a corresponding vector **I**.

(II) Power supply distribution computation: In order to identify the PDN defects covered by a pattern, we need to compute the power supply distribution induced by each defect under the application of such pattern. That is, computation for every combination of one pattern and one defect is needed while the straightforward method of circuit simulating for each of such combinations will cost prohibitive time. The proposed voltage calculation/calibration method and the simulation of multiple independent defects, shown in section 4.3, will also be applied here, which greatly reduces the run time. That is, defect-free PDN simulation is performed for each pattern at first; Only the edge defects in the segments with potentially harmful PDN defects will be simulated with the injection of multiple independent defects in each time; For other defects, the corresponding induced supply voltage distribution will be obtained by calculation and calibration instead of PDN simulation.

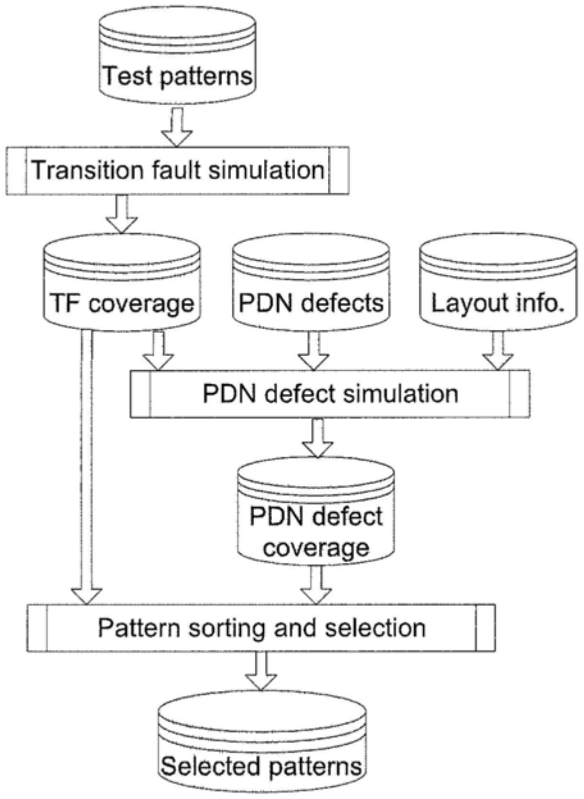


Figure 4.7: Flowchart of the pattern sorting and selection methodology.

(III) PDN defect identification: With the voltage distribution for each pattern with the existence of every defect, the PDN defects that can be detected by each pattern can then be identified. If a test pattern p can detect the transition fault of a standard cell whose power supply is lower than a pre-defined threshold in the existence of PDN defect d, it is then collected that pattern p can detect defect d.

Pattern sorting and selection: Given a set of n-detect transition fault patterns, our objective is select a subset to cover all the transition faults and detect as many PDN defects as possible. We propose a greedy heuristic to tackle this problem, as detailed in Section 4.4.3.

pattern	TF detected	R_{tf}	PDN defects detected	R_{pdn}
p_0	tf_0, tf_5, tf_{16}	6%	d_1, d_7, d_9	15%
p_1	tf_5, tf_9	4%	d_3, d_{12}	10%
...	...	...	...	...
p_n	$tf_1, tf_3, tf_{11}, tf_{19}$	8%	$d_6, d_{14}, d18$	15%

Table 4.1: Transition faults and PDN defects coverage table.

4.4.3 Pattern Sorting and Selection Algorithm

In this section, we propose to sort and select a list of test patterns from a group of candidates. We first find out the transition faults and PDN defects that can be detected by each pattern. Thereafter, the transition fault coverage, R_{tf}, and the PDN defect coverage, R_{pdn}, of each pattern can be computed. These data are summarized into a *Fault and Defect Coverage Table* (FDCT) while an example is shown in Table 4.1.

To evaluate the effectiveness of test patterns, we need to consider both transition fault and PDN defect coverage. A metric is therefore proposed, denoted as weighted coverage R_t.

$$R_t = a \times R_{tf} + b \times R_{pdn} \tag{4.4}$$

where a and b are weight parameters that reflect the relative importance of different kinds of fault coverage. Instead of using fixed a and b during the pattern selection procedure, we propose to adjust them at each step to achieve better pattern selection effect in the following manner.

$$\frac{a}{b} = \frac{1 - C_{tf}}{1 - C_{pdn}} \tag{4.5}$$

C_{tf} and C_{pdn} are the fractions of transition fault covered by the presently selected pattern set and that of PDN defect respectively. The parameters of a and b dynamically changing in such a way can reflect the updated coverage information, which can therefore achieve balanced coverage between transition fault and PDN defect. At the same time, such technique enables the pattern that can improve the total coverage most with high priority to be selected. As long as $1 - C_{pdn} \neq 0$, we always set b to 1, and compute a by Eq. 4.5. Specially we set $a = 1$ and $b = 0$ when $1 - C_{pdn} = 0$. For example, suppose $C_{tf} = 30\%$ and $C_{pdn} = 50\%$, which means that there are 70% of transition faults and 50% of PDN defects uncovered by the presently selected pattern set. We have $a = 1.4$, the patterns with high transition fault coverage are more preferable to be selected so that the transition fault coverage can be improved faster than that of the PDN defect.

The procedure to sort the n-detect patterns, which can test each of the faults by n times, and select a subset of them to achieve high PDN defect coverage is shown in Fig. 4.8. This algorithm takes the fault and defect coverage table T of the n-detect patterns as input and produces a sorted subset of the patterns, P_L. At the beginning, P_L is initialized to be empty while the number of left patterns, n_{lp}, is initialized as the total number of available test patterns, N_p; the uncovered transition fault list, L_{tf}, initially contains all the transition faults F and the uncovered PDN defect list, L_{pdn}, contains all the PDN defects D (Line 1). One pattern will be selected into P_L during each iteration (Lines 2-14). At the beginning of each iteration, parameters a and b are computed according to Eq. 4.5 (Line 3). All the patterns are evaluated with respect to transition fault and PDN defect coverage (Lines 6-7) and the pattern with maximum R_t is selected into P_L (Line 9). After the selection of one pattern, all the related information should be updated (Lines 11-14): the selected pattern, p_{sel}, is appended to the selected pattern list (Line 11); all the transition faults that can be detected by the selected pattern, $T_{tf}[sel]$, are

Pattern Selection Algorithm

INPUT: Table T

OUTPUT: Selected Patterns P_L

```
1. initialize $P_L \leftarrow \emptyset$; $n_{lp} \leftarrow N_p$; $L_{tf} \leftarrow F$; $L_{pdn} \leftarrow D$;
2. while $n_{lp} > 0$
3.     $\frac{a}{b} \leftarrow \frac{|L_{tf}|/|F|}{|L_{pdn}|/|D|}$;
4.     $R_{max} \leftarrow 0$; $sel \leftarrow -1$
5.     for $i = 0$ to $|T| - 1$
6.         compute $R_{tf}[i]$ and $R_{pdn}[i]$;
7.         $R_t[i] \leftarrow aR_{tf}[i] + bR_{pdn}[i]$;
8.         if $R_t[i] > R_{max}$
9.             $R_{max} = R_t[i]$; $sel = i$;
10.    if($R_{max} \leq 0$) break;
11.    $P_L = P_L \cup \{p_{sel}\}$;
12.    $L_{tf} = L_{tf} \setminus T_{tf}[sel]$;
13.    $L_{pdn} = L_{pdn} \setminus T_{pdn}[sel]$;
14.    $T = T \setminus T[sel]$;
15. return $P_L$;
```

Figure 4.8: Procedure of the proposed pattern sorting and selection algorithm.

removed (denoted by the symbol $\setminus$) from the uncovered transition fault list (Line 12) and all PDN defects that are covered by the selected pattern, $T_{pdn}[sel]$, are removed from the uncovered PDN defect list (Line 13); the table T should also be updated by removing the row corresponding to the selected pattern (Line 14). The procedure will terminate if the maximum weighted coverage $R_{max} = max(R_t)$ is no greater than zero or there is no unselected pattern left. Finally, the selected and sorted test patterns are returned (Line 15).

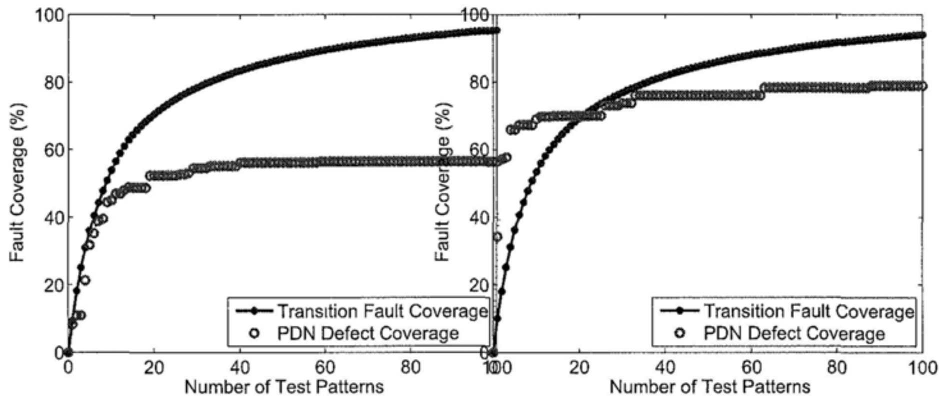
4.5 Experimental Results

To demonstrate the effectiveness of the proposed methodology, we have applied it on ISCAS'89 benchmark circuit s38417, which has 134 I/O ports and 23815 DFF/gates. Firstly, we generate the layout of this circuit using a commercial tool and the corresponding parasitic information is then extracted from such layout. The power supply voltage V_{dd} is set at 1.2V corresponding to the layout library.

Given no PDN open defects, all SCs can get power supply voltage higher than 94% V_{dd} with transition probability of 37.5%, according to the simulation results. The threshold voltage V_{th} for transition fault is at 90% V_{dd}, implying that there will be a slow-to-rise transition fault happening to the SCs that intend to make a rising transition but are provided with power supply voltage lower than V_{th}. In addition, we generate a 5-detect transition fault test pattern set by a commercial tool, and we intend to select a 1-detect pattern set out of it and compare the selected set against the 1-detect set directly generated from the same tool.

In the potentially harmful defects identification process, we set the transition probability as 37.5% to compute the current vector **I**. By using the proposed defect simulation and computation method presented in section 4.3, 840 open defects are identified as potentially harmful, which is about 3.5% of the total number of possible defects inside the whole PDN (around 25K). The existence of any of these defects may potentially induce insufficient power supply on one or more SCs (i.e., lower than the threshold voltage V_{th}).

Then, we evaluate the PDN defect coverage by the original 1-detect test pattern set, which is composed of 416 patterns and have a transition fault coverage of



(a) 1-detect pattern set. (b) Selected pattern set.

Figure 4.9: Transition fault and PDN defect coverage by different sets of patterns.

97.79%. The PDN defect coverage by up to 100 patterns are depicted in Fig. 5.3(a). With this set, totally 57.02% of the potentially harmful PDN defects can be detected.

Next, we select a set of 1-detect test patterns from the 5-detect patterns (including 1999 test patterns) to cover all the testable transition faults and at the same time we intend to cover as many PDN defects as possible, using the proposed strategy described in section 4.4. Fig. 5.3(b) shows the transition fault and PDN defect coverage variation with respect to the selected pattern count. With the proposed methodology, 366 patterns are selected to cover all the testable transition faults, which is even smaller than the original 1-detect test set. More importantly, the total PDN defect coverage is increased to 79.40%, which is 1.39 times of that of the original 1-detect pattern set.

The remaining 20.6% (173 out of 840) of the potentially harmful defects cannot be detected by all the patterns. We attribute them to: (i). some of the identified potentially harmful PDN defects can indeed be tolerated by the PDN structure; (ii). some PDN defects need more intensive switching activities to induce severer power supply droop so that they can be detectable, but the transition patterns are generated without such consideration and hence even the 5-detect test set may not be able to visualize certain PDN defects.

Considering the computation complexity, the run time needed for pattern sorting and selection is much less than that of potentially harmful open defects identification because other kinds of computation need neglectable time compared the time-consuming circuit simulation. For the similar reason, the time for defect identification is mainly dependent on the number of SCs in each segment since the proposed methodology needs to perform circuit simulation three times for each segment while the proposed voltage computation method consumes neglectable time compared to circuit simulation. In this experiment the average number of SCs inside each segment is 76, wherein the proposed methodology can reduce the computation time by 96%, compared to the case by circuit simulation only.

4.6 Conclusion and Future Work

In this chapter, we propose novel techniques to identify potentially harmful open defects in the power distribution networks. To cover these defects, we also present an effective test pattern selection algorithm. Both methods can greatly reduce the time complexity. Experimental results show that our proposed techniques are able to result in high PDN defect coverage with the 1-detect transition fault patterns selected out of N-detect patterns, when compared to conventional 1-detect patterns.

Since a considerable part of the PDN defects cannot be detected by normal delay test patterns even after applying the test pattern selection technique presented in this work, we plan to work on ATPG techniques that are able to generate dedicated test patterns for PDN defects.

□ **End of chapter.**

Chapter 5

Concurrent Online Delay Measurement

With technology scaling, integrated circuits behave more unpredictably due to process variation, environmental changes and aging effects. Various variation-aware and adaptive design methodologies have been proposed to tackle this problem. Clearly, more effective solutions can be obtained if we are able to collect real-time information such as the actual propagation delay of critical paths when the circuit is running in normal function mode. Motivated by the above, in this chapter, we propose a novel concurrent online delay measurement architecture for critical paths, namely *CODA*, to facilitate this task. Experimental results demonstrate high accuracy and practicality of the proposed technique.

5.1 Introduction

With technology scaling, process, voltage and temperature variations have a high impact on the timing behavior of integrated circuits (ICs), and hence it is increasingly difficult to ensure ICs' timing correctness solely by off-line manufacturing test. What's worse, circuits fabricated with latest technology suffer from ever-increasing aging effects (e.g., negative bias temperature instability (NBTI)), gradually reducing their performance [12, 72]. While there have been some attempts to conduct on-chip delay measurement to tackle the above problems (e.g., [64, 86, 87]), they require a dedicated test mode. Such non-concurrent solutions are hence inherently inaccurate due to the discrepancy between circuits' timing behavior in functional mode and that in test mode.

The ever-increasing non-predictability of IC performance has also motivated a number of research efforts in variation-aware and adaptive design methodologies. Various types of sensors (e.g., ring oscillator for process variation characterization [48, 63] and aging sensor [1]) are introduced on-chip to characterize the circuits' timing behavior, which can then be used for post-silicon tuning. Although helpful, these sensors can only provide some rough timing estimation without directly measuring critical paths in function mode.

Suppose the propagation delay of critical paths can be acquired with high accuracy as the circuit is working in function mode, such on-site information will be of great help for process variation characterization, dynamic management policy design and aging monitoring. For example, in dynamic voltage scaling (DVS), reducing supply voltage too much may result in prolonged delay outside of timing constraint while leaving a large margin wastes the timing slack for energy savings. If real-time path delay can be obtained, we are able to set proper values for the parameters of various dynamic management policies.

The above motivates us to develop a concurrent online delay measurement architecture, namely CODA, which is able to accurately measure the path delay while the circuit is working in normal function mode. As far as we know, this is the first accurate online delay measurement architecture. The innovative CODA holds the following outstanding benefits:

- High accuracy of measurement can be achieved by eliminating the interference from process variation and routing uncertainty.
- CODA measures the actual path delay as the circuit is working, without the need of switching to specific test mode.
- CODA completes delay measurement in a few clock cycles, which enables concurrent response corresponding to the real time changes of circuit status.
- The equipment of CODA requires acceptable hardware overhead and introduces negligible interference to the circuits' normal working, enabling CODA with high practicality.
- CODA is beneficial to a variety of tasks, including dynamic scaling, aging monitoring, speed binning, process variation characterization, etc. CODA itself can work for so multiple purposes and there is no need to equip the dedicated devices for each purpose.

5.2 Preliminaries and Motivations

Traditionally, the timing correctness of ICs is guaranteed by manufacturing test with external automatic test equipment (ATE). With technology scaling, however, this task has become increasingly difficult because: (i). it is quite difficult and costly to generate and apply delay tests for a large number of critical paths; (ii). ATE itself introduces inaccuracy into delay measurement. To address this problem, various on-chip delay measurement techniques and online delay error detection mechanisms have been proposed.

On-chip delay measurement: The key issue in on-chip delay measurement is to implement a time-to-digit converter. Vernier Delay Line (VDL) is a popular technique to achieve this objective [22, 64, 83], where two signals propagate through respective delay chain, working as data and clock inputs for a series of

FFs, respectively. The measurement result is then the sum of the delay difference in all the stages latching value '1'. In [30, 69], Ghosh *et al.* introduced another technique, wherein a capacitor is discharged linearly during measurement, while different delay intervals are converted into different voltage levels and then translated to digital signals. Ring oscillator-based methods have also been applied in path delay measurement by including the target path into an oscillation ring [48, 63].

In the above works, the delay introduced by the measurement circuit, including interconnect wires and logic gates, is assumed to be known or negligible, which severely impacts the measurement accuracy or even disables the practicality, especially considering the routing uncertainty at design stage and the ever-increasing effects of process variation. To tackle this problem, Wang *et al.* proposed a novel on-chip path delay measurement architecture, namely *Path-RO*, which builds an oscillation ring for a target path. Delay of the path and its returning loop is then translated into oscillation frequency while the delay of returning path is firstly set close to one clock cycle.

Path-RO needs a specific test mode for path delay measurement, where all the side-inputs of the gates along the target path are required to be non-controlling values in consecutive clock cycles to enable oscillation. Not all true paths in the circuit can satisfy such stringent requirement. Secondly, the operation condition in this test mode (e.g., power supply voltage, crosstalk and temperature) can be quite different from that in function mode. Therefore, significant deviation can occur between the measurement result from path-RO and the actual path delay in functional mode. Thirdly, the time needed to measure a path via *Path-RO* is in the order of k clock cycles, which is considerably long and costly.

Online delay error detection: As discussed earlier, it is increasingly difficult to predict and ensure the circuits' performance at design and manufacturing test stage. To accommodate this problem, various adaptive design methodologies have been presented, by detecting/predicting errors on-chip and compensate their effects.

One representative method is the so-called *Razor* technique used for error-tolerant microarchitecture design [27]. In this technique, a shadow latch is introduced to the receiving FF of each critical path. The value difference between the shadow latch and the flip-flop indicates the corresponding path delay is outside of one clock cycle, and such errors can be corrected by flushing the pipeline. Power supply voltage can then be adjusted according to the timing violation rate to achieve better balance between energy consumption and performance.

As circuit aging has the unique feature that circuit delay increases slowly and steadily, Agarwal *et al.* proposed to conduct circuit failure prediction for this particular failure mechanism so that circuit can take proactive actions before errors actually appear [1]. To achieve this, an aging sensor is integrated inside each target flip-flop for guardband checking, i.e., to check whether there are transitions close to the end of the clock period. Adjustment can then be applied for timing safety, e.g., prolonging the clock cycle or reducing the path delay by increasing supply voltage.

The above techniques either work in a try-and-error manner or target on a particular kind of purpose. Clearly, if we are able to acquire the delay of critical paths with high accuracy as the circuit is working in function mode, such information will be of great help for process variation characterization, dynamic management and aging monitoring, which motivates a novel concurrent online delay measurement architecture, namely *CODA*.

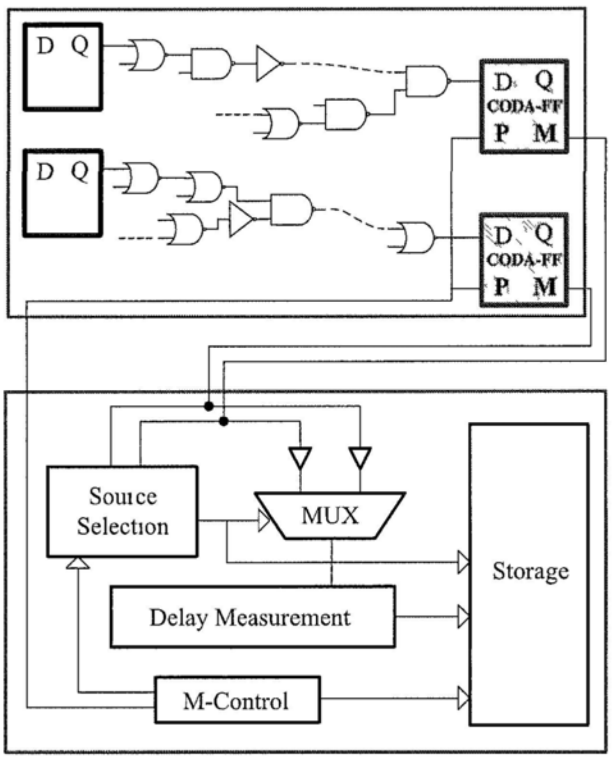


Figure 5.1: Structure of the proposed online delay measurement architecture.

5.3 Overview of CODA

We firstly overview CODA here, including the functionality of the main components and the delay measurement procedure in CODA.

CODA Infrastructure: The schematic structure of a circuit equipped with CODA is shown in Fig. 5.1. The upper rectangle encircles the circuit under measurement (CUM), where two FFs are selected as targets for delay measurement, while inside the lower rectangle is the measurement circuit. CODA is mainly composed of the following components.

1). *CODA Flip-Flop* (*CODA-FF*) at the receiving ends of critical paths. Two more ports (i.e., *P* and *M*) and extra circuitries are introduced into CODA-FF, compared with normal FF, to facilitate online delay measurement.

2). *MUX*, which is used to select one signal at a time for delay measurement among multiple sources sharing the same delay measurement circuit.

3). *Source selection block*, which generates codes controlling the MUX, ensuring that at most one signal with transition in a pre-defined time window is selected during each measurement.

4). *Delay measurement unit* (*DMU*). While various types of on-chip delay measurement scheme can be utilized, in this work, we adopt the VDL-based technique, where the target signal and the clock signal are fed into DMU to measure the delay between them.

5). *M-Control block*, which is the main controller to ensure consistent operation of CODA.

6). *Storage module*, which stores the measurement results and outputs the corresponding value when requested.

Delay Measurement Procedure: In CODA, the delay measurement procedure is made up of two stages.

1) Probe route delay measurement. We name the route connecting a CODA-FF with DMU as *probe route*. Correspondingly, its delay is denoted as *probe route delay* or *probe delay*. As the IC starts to work in function mode, M-Control block firstly initializes CODA into probe delay measurement mode, wherein port P of each CODA-FF is fed with logic '1' and port M outputs signal for probe delay measurement. The source selection block selects the sources, one by one and orderly, so that their probe delays are measured, respectively. The measurement results of probe delay are then stored into the storage module.

2) Online delay measurement. After finishing the probe delay measurement, CODA conducts online delay measurement. Port P of each CODA-FF is fed with logic '0' and the target signals travel through port M to DMU along respective probe route so that their delay is measured. The measurement result is then the

total delay that includes the target path delay in the CUM and the probe delay, which is also saved in the storage module. Deducting the probe delay from the total delay can then obtain the target path delay.

Characteristics of CODA: Firstly, CODA does not disturb the function of CUM, and in fact CUM does not realize the existence of CODA. The only interference induced to CUM is the negligible delay overhead of CODA-FF compared to normal FF, with the detailed analysis presented in Section 5.5.3.

Secondly, in CODA, only the receiving end of a critical path is connected with DMU, while the traditional techniques need to connect two ends of a path with DMU, which greatly reduces routing overhead.

Thirdly, critical FFs are selected as targets for delay measurement in CODA, instead of critical paths. Considering that frequently multiple critical paths converge at the same FF, CODA significantly reduces the complexity of measurement.

Fourthly, CODA acquires the path delay by deducting the probe delay from the total delay. Such measurement mechanism can tolerate routing uncertainty and most of the variations, guaranteeing high accuracy.

5.4 Detailed Design of CODA

In this section, we present the detailed design of the major components in CODA.

5.4.1 CODA-FF Design

In a FF there are a master latch (ML) and a slave latch (SL). The timing constraint requires that the signal should propagate through the combinational circuit path, including the SL at the driving end and the ML in the receiving end, within one clock cycle. Such propagation delay determines the performance of VLSI circuits and it is just the target delay to be measured in this work.

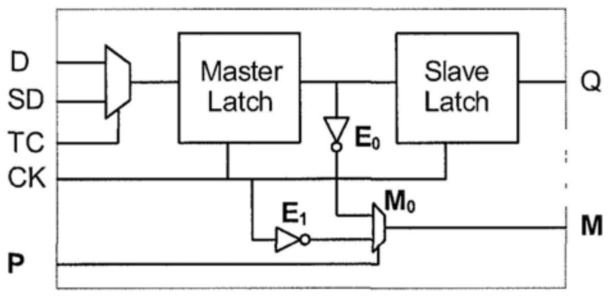


Figure 5.2: Design of the proposed CODA-FF.

In CODA, the normal FF at a target location is replaced by the proposed CODA-FF whose design is shown in Fig. 5.2, which is namely target FF. Compared with normal FF, CODA-FF is with two additional ports (P and M) and three addition gates (inverter E_0, E_1 and multiplexer M_0).

Port M is connected with DMU through probe route to transfer signal for delay measurement. Port P is fed with mode signal from the M-Control block to select measuring the probe delay ($P = 1$) or total delay ($P = 0$), as mentioned in section 5.3.

1) When $P = 1$, clock signal is hence transferred to port M. The transition of clock then propagates to DMU through the probe route while the other input of DMU is directly connected with clock. In such circumstance, the transition skew between the two inputs of DMU is just the probe route delay.

2) When $P = 0$, the output of ML is connected with port M so that the signal arriving at CODA-FF through the target path in CUM is further transferred to DMU. Now the skew between the two inputs of DMU is therefore the total delay, which equals to the sum of the path delay in CUM and the probe delay.

Consequently, the target path delay can be obtained by deducting the probe delay from the total delay.

The two inverters E_0 and E_1 are to isolate the CUM from the measurement circuit so that the only interference, induced to CUM by CODA, is that the output of ML in CODA-FF is with extra capacitance load resulting from the input of inverter E_0. Such extra capacitance results in prolonged delay, which in fact is negligible with detailed experimental results shown in section 5.5.3. On the logic level, CUM does not realize the existence of CODA. That is, CUM and CODA work independently without interference.

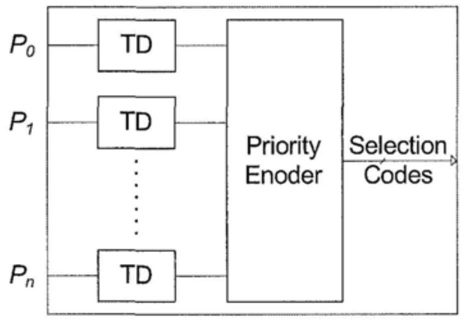
About the target FFs selection, those FFs whose delay is crucial for circuits' performance and dynamic methodologies, should be selected as targets. Because of limited space, the selection details are out of the scope of this chapter.

The proposed delay measurement mechanism and the corresponding CODA-FF design demonstrate two outstanding advantages. 1) The way of obtaining the path delay in CUM by deducting the probe delay from the total delay can eliminate most of the disturbance happening on the probe route, including process variation, routing uncertainty and aging effect. This enables high measurement accuracy and the sharing of one CODA module among multiple target FFs. 2) CODA introduces no interference to the circuit's function, which simplifies the design and operation of circuits equipped with CODA.

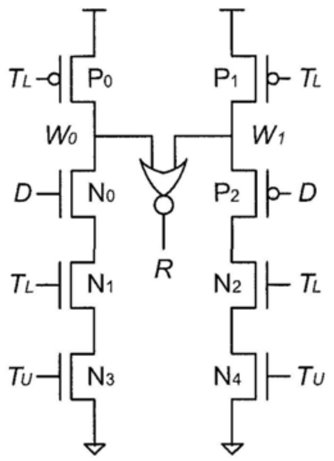
5.4.2 Source Selection

Sharing one DMU among multiple target FFs can significantly reduce hardware overhead. Correspondingly, it must be guaranteed that at most one signal is selected at any time. Here we adopt transition detection and priority based selection mechanism, which firstly checks if there is transition in pre-defined time window for each signal and then select the one with highest priority among all the signals with transition.

Fig. 5.3(a) shows the design of the proposed signal selection block, which is mainly composed of Transition Detectors (TDs) and a priority encoder. The sig-



(a) Source selection block.



(b) TD.

Figure 5.3: Source selection block and transition detector (TD).

nals from CODA-FFs go into respective TD for transition checking. The output of a TD becomes active once it detects a transition in the pre-define time window, which indicates that the corresponding path is activated and waiting for delay measurement. When there are multiple signals ready for measurement simultaneously, further selection algorithm is required to ensure only one signal is finally selected. Here we adopt priority encoder to ensure the uniqueness. Target FFs whose delay information is more crucial are with higher priority to be selected. The outputs from all TDs are further fed into the priority encoder that then generates selection codes controlling the MUX block.

We adopt the TD with design shown in Fig. 5.3(b), which is based on the stability checker in [1]. The TD detects whether transition happens on the input signal *D* during the time window with lower bound T_L and upper bound T_U. Particularly, the rising edge of T_L indicates the start point of the time window while the falling edge of T_U indicates the end point. Only when *D* transits between T_L and T_U, the output *R* becomes logic '1'. The working mechanism of the proposed TD is as follows.

1): Before the time window, $T_L = 0$ and $T_U = 1$. Consequently, PMOS transistors P_0 and P_1 are on while NMOS transistors N_1 and N_2 are off. Therefore, nodes W_0 and W_1 are charged to logic '1' while output $R = 0$.

2): During the time window, $T_L = 1$ and $T_U = 1$. P_0 and P_1 are off while N_1, N_2 N_3 and N_4 are on. If there is a transition on signal D, both N_0 and P_2 will be on during a period of time so that W_0 and W_1 will both be discharged, resulting in $R = 1$. If there is no transition, only one of W_0 and W_1 will be discharged with R staying at 0.

3): After the time window, $T_L = 1$ and $T_U = 0$. Both the pull-up and pull-down networks are off so that the output stays stable.

In a word, only the transition of D, no matter rising or falling transition, happening in the time window defined by $[T_L, T_U]$ can generate $R = 1$.

One thing should be noted is the timing between the source selection and the delay measurement. Extra delay is added before the MUX, denoted as buffers in Fig. 5.1, insuring that selection codes are ready before the signals arrive at MUX. Since in CODA the path delay is generated by subtracting probe delay from the total delay while the latter two delays both include such extra delay, the interference of extra delay devices can be eliminated in fact.

5.4.3 Delay Measurement Circuit

The delay measurement circuit in CODA is to quantify the time interval between two input signals. There have been multiple kinds of methods targeting at measuring delay on chip that can be applied in CODA. Here we adopt the VDL based measurement method which can achieve high resolution with low hardware overhead.

Structure of the proposed DMU: Fig. 5.4 shows the proposed circuit design of DMU, which digitalizes how late the rising transition of signal T happens after

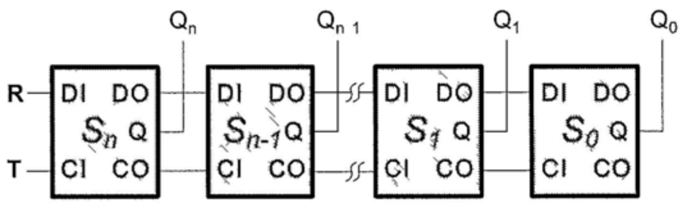


Figure 5.4: The proposed delay measurement circuit.

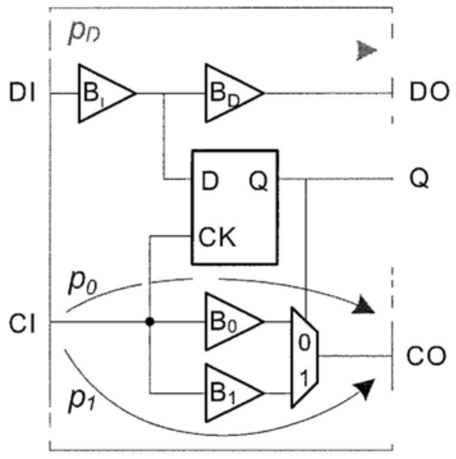


Figure 5.5: A stage of the proposed delay measurement circuit.

that of signal R. That is, R is connected with reference signal while the to be measured signal is fed into T. The falling transition of R and T will be transformed into rising one before measurement [83]. The measurement circuit shown in Fig. 5.4 is made up of a series of $n+1$ stages, S_n to S_0, while the combination of the output at each stage, Q_n to Q_0, shows the measurement result.

In each stage there are two input ports (DI and CI) and three output ports (DO, CO and Q). DO from the previous stage is fed into DI of the next stage and CO is connected with CI. The proposed design of a stage module is shown in Fig. 5.5, where it can be seen that the signal DI always travels through path p_D to DO, while CI travels through path p_0 or p_1 corresponding to the output of FF $Q = 0$ or $Q = 1$, respectively.

Working mechanism of DMU: Considering the timing of the signals in a stage, $t_{DO} = t_{DI} + d_{p_D}$ while $t_{CO} = t_{CI} + d_{p_0}$ or $t_{CO} = t_{CI} + d_{p_1}$; $t_D = t_{DI} + d_{B_t}$ while $t_{CK} = t_{CI}$. A stage will work in one of the following two ways corresponding to different timing relationship with t_{setup} representing the setup time of the FF.

1) $t_D < t_{CK} - t_{setup}$. This timing relationship means that $s = t_{CI} - t_{DI} > d_{B_t} + t_{setup}$, indicating that the skew between CI and DI is longer than $d_{B_t} + t_{setup}$. Hereafter, $d_{B_t} + t_{setup}$ is denoted as d_i, representing the character delay of stage i. As the result of $t_D < t_{CK} - t_{setup}$, logic '1' will be latched in the FF and the output Q changes to '1' from the initialized value '0', which consequently controls CI to traverse path p_1 with buffer B_1. Path p_1 is so design that $d_{p_1} = d_{p_D} - d_i$, meaning the delay of path p_1 is shorter by d_i than that of path p_D, which results in the skew between CO and DO is shorter by d_i than that between CI and DI.

2) $t_D \geq t_{CK} - t_{setup}$. Value '0' will be latched in the FF while Q stays at '0'. Consequently, CI travels through path p_0 that is designed with $d_{p_0} = d_{p_D}$, which results in the skew between CO and DO equal to that between CI and DI.

In conclusion of cases 1 and 2, if the skew between two input signals of stage i is longer than its character delay d_i, the output Q_i becomes '1' while the skew is reduced by d_i and fed into next stage; If the skew is shorter than d_i, Q_i stays '0' and the skew stays unchanged for next stage. Therefore, the measurement range of such measurement circuit equals to the sum of all character delays, while summing the character delay of all the stages outputting '1' generates the measurement result for the skew between the two input signals.

The character delay of each stage in the proposed design can be with any desired value. Here we adopt exponential distribution of character delay along the series of stages where $d_i = d_0 2^i$, with motivation from [64] [87]. Such assignment is with the advantage of low hardware overhead and the measurement result is

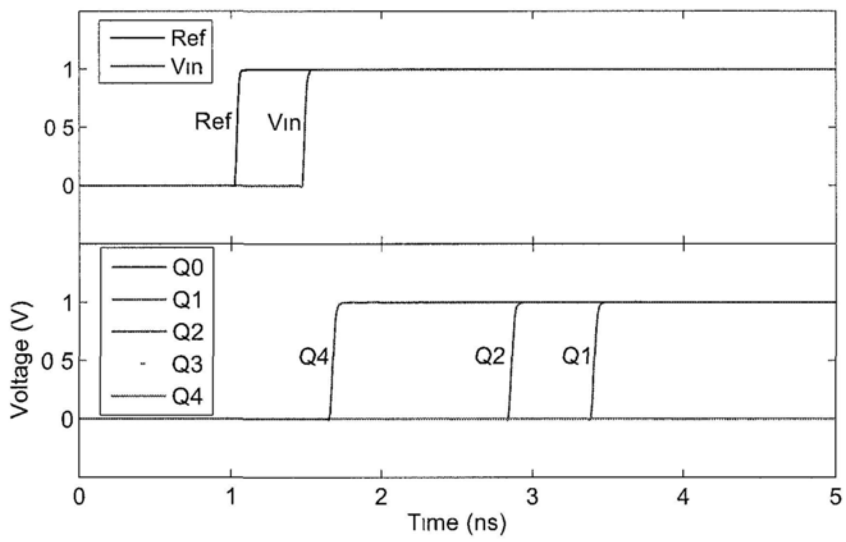


Figure 5.6: Operation waveform of the delay measurement circuit.

directly expressed as binary number, easing the following processing.

The waveform from an example of delay measurement is shown in Fig. 5.6, where the measurement circuit consists of five stages with character delay of 0.32, 0.16, 0.08, 0.04 and 0.02 *ns* from stage S_4 to S_0, respectively. Compared with R_{ef}, the other input signal V_{in} is with 0.45 *ns* delay as shown in the upper part of Fig. 5.6, while the measurement output is demonstrated in the lower part as $Q_4Q_3Q_2Q_1Q_0 = 10110$. Correspondingly, the measurement result is $d_{VDL} = (10110)_2 \times d_0 = 22 \times 0.02 = 0.44$ *ns*, which is with -0.01 *ns* deviation from the real delay.

From the working mechanism it can be seen that the measurement result will always be no greater than the real delay because the skew between the output signals DO and CO of each stage will always be no less than zero. That is, the output skew of the last stage S_0 will be thrown out of measurement and this uncovered part of the skew will result in measurement result less than real delay. Consequently, the measurement resolution is dependent on the character delay of the last stage d_0. With exponential distribution of character delay along the stages, the last stage can be designed with small character delay to achieve high measurement resolution while the previous stages with exponentially increasing character delay can provide long measurement range without the need of large number of stages.

Influence of clock skew: In CODA, one input of DMU is always connected with clock, which means that the measured delay in DMU is in fact the skew between the other input signal and the clock at DMU. In addition, we use the clock signal inside CODA-FF for probe route delay measurement. Let t_s represent the timing of the signal arriving at CODA-FF whose delay is to be measured, while t_{C_M} and t_{C_FF} represent the timing of the clock at the measurement circuit and that at CODA-FF, respectively. In probe delay measurement mode, the delay d_p measured by CODA can then be expressed as follows with d_{probe} denoting the probe route delay.

$$d_p = t_{C\_FF} + d_{probe} - t_{C\_M} \tag{5.1}$$

While in online delay measurement mode, CODA measures the total delay d_t.

$$d_t = t_s + d_{probe} - t_{C\_M} \tag{5.2}$$

Consequently, the final result from CODA d_{CODA} is calculated by deducting the probe delay from the total delay.

$$d_{CODA} = d_t - d_p = t_s - t_{C\_FF} \tag{5.3}$$

That is, the final result from CODA for represents the skew between the function signal arriving at CODA-FF and the clock at the same CODA-FF. Such comparison is in fact to check if the timing constraint is satisfied. For example, a result

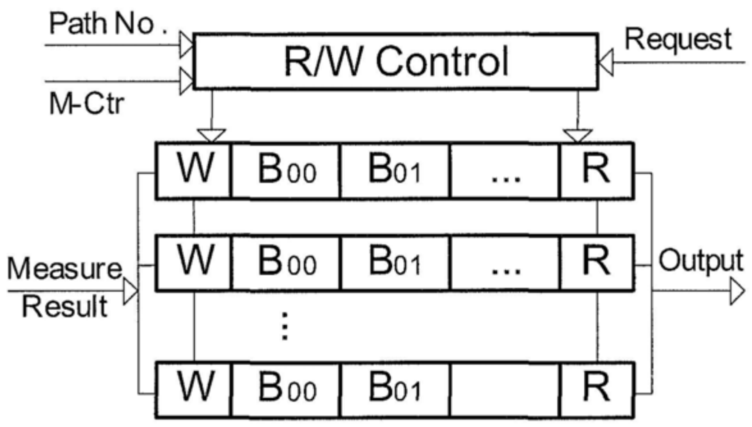


Figure 5.7: Storage module for delay measurement results.

with $d_{CODA} = 1.1T$ indicates that the timing constraint has been violated at the corresponding CODA-FF. Therefore, the final result from CODA is exactly the information needed for timing constraint checking. In other words, CODA can check to what extent the timing constraint can be satisfied at the locations equipped with CODA-FFs, no matter how much clock skew exits.

5.4.4 Storage of Measurement Results

The measurement results from CODA include source No., delay value and delay type (probe delay or total delay). If there is no dedicated storage attached to CODA, such measurement results need to be collected each time by such circuits as the master processor on chip. Here we propose an effective storage design with moderate overhead that can significantly ease the processing of measurement results, which is demonstrated in Fig. 5.7.

In each row of the storage module, there are two storage blocks for one source, one storing the probe delay while the other storing the longest total delay. In probe delay measurement mode, the measurement result is stored in the first storage block. In online delay measurement mode, the R/W control block first compare a new measurement result with the content in the second block of the corresponding source. Only if the new measurement result is with larger value, the second block is refreshed with the new value. Once there comes the request for delay reading, the desired source is selected and the corresponding values are outputted.

5.5 Experimental Results

In this section, we report SPICE simulation results for CODA, based on a 90 *nm* IC fabrication technology with nine metal layers and $1V$ power supply voltage.

5.5.1 Measurement Accuracy

In the experiments we have implemented CODA with seven stages of delay measurement circuit, where the character delay of stage i $d_i = 0.02 \times 2^i$ ns. Table 5.1 shows the measurement results, with time unit of ns, for eight target FFs in ISCAS'89 benchmark circuit S38417, as a proof of concept. The FFs are selected to cover paths with delay in the range of $[0.8L, L]$ with L as the longest path delay. After the source No. listed in the first column of Table 5.1, the path delay from Spice simulation is shown in the second column as d_{simu}. The following three columns specify the results from CODA, where d_p, d_t and d_{CODA} denote probe delay, total delay and path delay from CODA, respectively, with $d_{CODA} = d_t - d_p$. The last two columns then shows the comparison between the path delay from Spice simulation and that from CODA, where $\Delta d = d_{CODA} - d_{simu}$ and $Err = |\Delta d / d_{simu}| \times 100\%$.

The measurement inaccuracy in CODA results from (I) the delay variation of the probe route, and (II) the resolution of the delay measurement unit.

Considering part I, firstly, CODA generate path delay by measuring the probe delay and deducting it from the total delay, which can tolerate probe delay varia-

Table 5.1: Delay measurement results for benchmark circuit S38417

No.	d_{simu}	d_p	d_t	d_{CODA}	Δd	Err (%)
0	1.076	0.40	1.48	1.08	0.004	0.36
1	1.062	0.12	1.18	1.06	-0.002	0.20
2	1.034	0.36	1.40	1.04	0.006	0.53
3	0.990	0.08	1.06	0.98	-0.010	1.02
4	0.966	0.64	1.62	0.98	0.014	1.48
5	0.942	0.24	1.18	0.94	-0.002	0.24
6	0.922	0.56	1.48	0.92	-0.002	0.19
7	0.865	0.72	1.58	0.86	-0.005	0.55

tion resulting from process variation and routing uncertainty at design stage. The uncovered measurement inaccuracy is from the operation condition variation. That is, the delay of a probe route can be different during each measurement because of variation in temperature, power degradation and crosstalk. This kind of variation is hard to be thoroughly eliminated, requiring that there is no probe route needed or the probe delay and total delay are measured at the same time, which is a limitation of CODA. However, CODA can mitigate such variation by suffering less probe delay variation benefiting from the short probe routes needed in CODA, while the detailed analysis of the routing in CODA is explained in section 5.5.3.

For part II, the resolution of VDL based delay measurement circuit with exponential distribution of character delay is determined by the smallest character delay d_0, as explained in section 5.4.3. That is, the measurement result d_{VDL} is always in the range of $[d - d_0, d]$, where d is the practical delay value. Consequently, the path delay that is obtained by deducting the probe delay from the total delay is with deviation from the practical value in the range of $[-d_0, d_0]$. The measurement error in our experiments is from -0.005 to 0.014 ns corresponding to $d_0 = 0.02ns$, which certifies the effectiveness of the proposed delay measurement circuit and CODA.

5.5.2 Impact of Process Variation

In CODA, process variation on probe route can be eliminated while the process variation on target paths are directly measured, which will not degrade the measurement accuracy. Process variation, however, can also occur on the delay measurement circuit itself, i.e., the pre-defined buffer delay used in VDL circuit may be inaccurate and hence induces measurement inaccuracy. We therefore conduct experiments to show its impact on CODA.

As shown in [4], the variation of multiple types of semiconductor device factors can be unified into changes of the threshold voltage V_{th}. In other words, variation of V_{th} can reflect multiple types of variation. Therefore, we introduce V_{th} variation into the proposed circuit and conduct Monte-Carlo simulation to analyze how robust CODA is with the existence of process variation. The V_{th} variation of a transistor is assumed to be inversely linear to the sqaure root of its size [4], as shown in formula 5.4.

$$\Delta V_{th} \propto \frac{C}{\sqrt{W_{eff}L_{eff}}} \tag{5.4}$$

In formula 5.4, C is a constant value which is associated with manufacturing technology. W_{eff} and L_{eff} are the effective channel width and length of a transistor, respectively. For transistor with minimum size, we set V_{th} to follow normal distribution with sigma as 5% of its mean value. Different ΔV_{th} is then applied onto each transistor according to its size. Nine delay values from 0.8 ns to 1.2

Table 5.2: Error of delay measurement under process variation

delay	Measurement error					
	-60*ps*	-40*ps*	-20*ps*	0*ps*	+20*ps*	+40*ps*
0.80 *ns*		13%	44%	27%	16%	
0.90 *ns*	1%	12%	42%	45%		
1.00 *ns*		15%	33%	42%	9%	1%
1.10 *ns*		1%	51%	35%	13%	
1.20 *ns*		9%	45%	35%	11%	
	-50*ps*	-30*ps*	-10*ps*	0*ps*	+10*ps*	+30*ps*
0.85 *ns*	7%	23%	43%		22%	5%
0.95 *ns*		30%	44%		21%	5%
1.05 *ns*	3%	24%	40%		29%	
1.15 *ns*	3%	29%	46%		22%	3%

ns are measured by the proposed circuits in the presence of process variation and Monte-Carlo simulation with 1000 iterations is performed for each delay value. We have not enlarged the size of the transistors in the proposed circuit on purpose, i.e., all transistors are with the minimum size that can realize the functionality.

The measurement results under process variation is shown in Table 5.2, where the first column shows the delay to be measured while the second and eighth row lists the classification of measurement error. The other entries of the table show the percentage of measurement with the corresponding error. For example, considering the measurement results for 0.8 *ns*, 13%, 44%, 27% and 16% of the 1000 iterations is with error of -40, -20, 0, 20 *ps*, respectively. The distribution of the

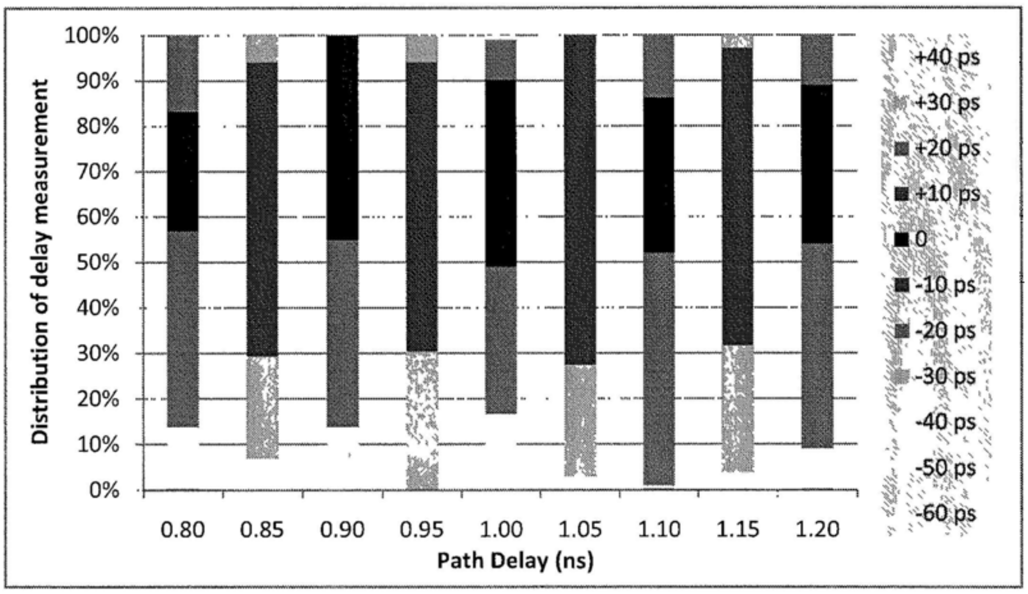


Figure 5.8: Distribution of measurement error under process variation

measurement result is also shown in Fig. 5.8 by histogram for clarity, where the one with less shadow is with larger error. Only 0.1% of the total measurement is with the maximum error of 60 *ps*, and 15% is with error larger than 20 *ps*. The remaining can achieve accuracy within 20 *ps*, which accounts for nearly 85% of the total measurements. Such measurement error distribution demonstrates that CODA is insensitive to process variation and therefore can achieve fine measurement resolution in the presence of process variation. Enlargement of the transistor sizes can further improve the robustness of the delay measurement circuit.

5.5.3 Overhead of CODA

CODA introduces two kinds of overhead into the original circuit: (i). the hardware used to implement it; and (ii). the extra delay due to the replacement of normal FF by CODA-FF.

Hardware Overhead

The hardware overhead includes the logic gates used to implement the functionality of CODA and the wires needed to connect the target FFs in CUM with the delay measurement unit.

Transistor overhead: The transistor overhead required for CODA mainly exists in four parts.

1) Compared to traditional SFF, the proposed CODA-FF needs two more inverters and one more MUX2 gate, which costs extra hardware overhead roughly equivalent to four NAND2 gates of minimum size.

2) One stage in the delay measurement circuit needs hardware overhead equivalent to about 14 NAND2 gates. The total hardware overhead there can then be expressed as $14N_m$ with N_m representing the number of stages in the measurement circuit.

3) For each target, hardware overhead of about 15 NAND2 gates are needed to build the transition detector, encoder, decoder, MUX, etc.

4) To store the measurement result, 28 NAND2 gates equivalent overhead are required to construct the memory part for each target.

Therefore, in the case with N_t target FFs and N_m stages of DMU, the hardware overhead is equivalent to $A = 4N_t + 14N_m + 15N_t + 28N_t = 47N_t + 14N_m$ NAND2 gates. For example, in a design with 64 targets and eight stages of delay measurement, the total transistor overhead is equivalent to 3k NAND2 gates. Such amount of transistor overhead is acceptible for nowaday IC, especially considering that at present IC is mostly routing limited.

Routing overhead: Compared to transistor overhead, wire routing overhead is more critical nowadays since the performance of VLSI circuits is generally more dependent on wire routing. Especially, those long wires demonstrate long delay and cost lots of crucial routing resources. For the oscillation ring based delay measurement methods, a returning loop is needed for each target path, traversing backward from the receiving end to measurement unit and then finally to the driving end of the path. Therefor, the returning loop is with length similar to critical paths. What's worse, it is very possible that a considerable part of returning loops is even much longer than critical paths, considering that multiple paths share one measurement unit which cannot be close to all the paths. Consequently, so long returning loops occupy a lot of routing resources. In CODA, only the receiving ends of the paths are required to be connected with the measurement circuit while there is no routing requirement at the driving ends. Such character can greatly reduce the routing overhead, especially considering that the measurement circuit can be placed near those target FFs.

Delay overhead of CODA-FF

To equip CODA, CUM needS to replace the traditional SFFs at the target locations by the proposed CODA-FFs. The only interference induced to CUM is the extra delay caused by the extra capacitance load of the extra gates in CODA-FFs.

Here we show the worst case delay overhead resulting from the proposed CODA-FF. First, we build the traditional SFF where all the internal logic gates are with driving capability equivalent to minimum size inverter. Thereafter, we add the extra gates to build the proposed CODA-FF. We compare the path delay with traditional SFF and the proposed CODA-FF attached to the receiving end, respectively, which can then show the extra delay caused by the proposed CODA-FF. In the comparison experiments the last gate before the FF is designedly assigned with minimum size inverter. Such arrangement is to create worst case condition, where all the related gates are with minimum driving capability and maximum extra delay can then result from the extra load in the proposed CODA-FF.

Spice simulation results show that the maximum delay is 4.8 *ps*, which is negligible compared to path delay in the order of ns. Such delay value is similar to that of [87], which is much smaller than that in the initial version [86]. What's more, such delay can be greatly reduced by increasing the size of the related gates moderately. Therefore, the proposed CODA-FF introduces negligible interference onto the circuits' operation.

5.5.4 Measurement Time

The time needed to complete a delay measurement in CODA is dependent on three factors: I. The probe route delay; II. The desired time window within which the delay is to be measured; III. The response time of DMU.

I) It is reasonable that probe route delay is considerably smaller than one clock cycle, especially considering the low routing overhead in CODA.

II) Time window with the length of one clock cycle is enough since the center of the timing window can be set as desired. For example, we can set it at the end of clock cycle (T), meaning that the time window covers delay from $0.5T$ to $1.5T$, which is enough to cover critical paths with delay close to $1T$.

III) The required measurement time is dependent on the measurement range, which is the sum of the delay in part I and II. When the probe delay and the time window are both within 1T, it means that the signals arriving at DMU are with time variation of $2T$. Consequently, DMU is required for a measurement range of $2T$, where the longer chain delay in DMU is within $4T$. Therefore, the measurement result can be ready in $1T+1T+4T=6T$.

In fact time window of $0.5T$ and maximum limit of $0.5T$ for probe route delay are enough in most cases, where the measurement result will be ready in $4T$. For the oscillation ring based delay measurement, the time needed for each measurement is in the order of 10^3 clock cycles, which is longer by three orders than that of CODA.

5.6 Conclusion

Currently process variation collection and dynamic management can help IC chips to achieve efficiency of performance, cost and life. An online delay measurement architecture, CODA, is therefore proposed here to support multiple applications, which can measure the online path delay concurrently as the circuit is functioning. CODA is with low cost and can achieve high measurement accuracy by eliminating the influence of probe route, even in the presence of process variation. The accurate real-time delay information from CODA can serve for a variety of purposes.

□ **End of chapter.**

Bibliography

[1] M. Agarwal, B. Paul, M. Zhang, and S. Mitra. Circuit Failure Prediction and Its Application to Transistor Aging. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 277–286, 2006.

[2] C. J. Anderson et al. Physical Design of a Fourth-Generation POWER GHz Microprocessor. In *Proceedings International Solid State Circuits Conference (ISSCC)*, pages 232–233, 2001.

[3] S. Arora. The approximability of NP-hard problems. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 337–348, 1998.

[4] A. Asenov, A. Brown, J. Davies, S. Kaya, and G. Slavcheva. Simulation of intrinsic parameter fluctuations in decananometer and nanometer-scale mosfets. *Electron Devices, IEEE Transactions on*, 50(9):1837–1852, Sept. 2003.

[5] A. Attarha and M. Nourani. Test Pattern Generation for Signal Integrity Faults on Long Interconnects. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 336–341, April 2002.

[6] G. Bai, S. Bobba, and I. Hajj. Static Timing Analysis Including Power Supply Noise Effect on Propagation Delay in VLSI Circuits. In *Proceedings ACM/IEEE Design Automation Conference (DAC)*, pages 295–300, 2001.

[7] G. Bai, S. Bobba, and I. N. Hajj. RC Power Bus Maximum Voltage Drop in VLSI Circuits. In *Proceedings International Symposium on Quality of Electronic Design (ISQED)*, pages 257–258, 2001.

[8] X. Bai, R. Chandra, S. Dey, and P. Srinivas. Interconnect Coupling-Aware Driver Modeling in Static Noise Analysis for Nanometer Circuits. *IEEE Transactions on Computer-Aided Design*, 23(8):1256–1263, August 2004.

[9] X. Bai, S. Dey, and A. Krstid. Hyac: A Hybrid Structural SAT Based ATPG for Crosstalk. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 112–121, September 2003.

[10] X. Bai, S. Dey, and J. Rajski. Self-Test Methodology for At-Speed Test of Crosstalk in Chip Interconnects. In *Proceedings ACM/IEEE Design Automation Conference (DAC)*, pages 619–624, 2000.

[11] M. Becer, R. Vaidyanathan, C. Oh, and R. Panda. Crosstalk Noise Control in an SoC Physical Design Flow. *IEEE Transactions on Computer-Aided Design*, 23(4):488–497, April 2004.

[12] S. Borkar. Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. *IEEE Micro*, 25(6):10–16, 2005.

[13] F. Caignet, S. Delmas-Bendhia, and E. Sicard. The Challenge of Signal Integrity in Deep-Submicrometer CMOS Technology. *Proceedings of the IEEE*, 89(4):556–573, April 2001.

[14] H. H. Chen. Power Supply Noise Analysis Methodology for Deep-Submicron VLSI Chip Design. In *Proceedings ACM/IEEE Design Automation Conference (DAC)*, pages 638–643, 1997.

[15] J. Chen and L. He. Worst Case Crosstalk Noise for Nonswitching Victims in High-Speed Buses. *IEEE Transactions on Computer-Aided Design*, 24(8):1275–1283, August 2005.

[16] L. Chen, X. Bai, and S. Dey. Testing for Interconnect Crosstalk Defects Using On-Chip Embedded Processor Cores. In *Proceedings ACM/IEEE Design Automation Conference (DAC)*, pages 317–322, 2001.

[17] L. H. Chen and M. Marek-Sadowska. Aggressor Alignment for Worst-Case Crosstalk Noise. *IEEE Transactions on Computer-Aided Design*, 20(5):612–621, May 2001.

[18] P. Chen, G. Zhang, and Z. Liu. A Unified Compact Scalable ΔI_d model for Hot Carrier Reliability Simulation. In *IEEE 37th Annu. Int. Reliability Phys. Symp.*, pages 243–248, 1999.

[19] T.-S. Chen, C.-Y. Lee, and C.-H. Kao. An Efficient Noise Isolation Technique for SOC Application. *IEEE Transactions on Electron Devices*, 51(2):255–260, February 2004.

[20] W.-Y. Chen, S. K. Gupta, and M. A. Breuer. Test Generation for Crosstalk-Induced Delay in Integrated Circuits. In *Proceedings IEEE International Test Conference (ITC)*, pages 191–200, October 1999.

[21] M. Cuviello, S. Dey, X. Bai, and Y. Zhao. Fault Modeling and Simulation for Crosstalk in System-on-Chip Interconnects. In *Proceedings International Conference on Computer-Aided Design (ICCAD)*, 1999.

[22] R. Datta, G. Carpenter, K. Nowka, and J. Abraham. A scheme for on-chip timing characterization. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 24–29, 2006.

[23] A. Devgan and S. Nassif. Power Variability and its Impact on Design. In *Proceedings International Conference on VLSI Design*, pages 679–682, 2005.

[24] S. Dutta, R. Jensen, and A. Rieckmann. Viper: A Multiprocessor SOC for Advanced Set-Top Box and Digital TV Systems. *IEEE Design & Test of Computers*, 18(5):21–31, Sept.-Oct. 2001.

[25] Z. S. Ebadi and A. Ivanov. Time Domain Multiplexed TAM: Implementation and Comparison. In *Proceedings Design, Automation, and Test in Europe (DATE)*, pages 732–737, 2003.

[26] Z. P. Erik Larsson. An Integrated Framework for the Design and Optimization of SOC Test Solutions. *Journal of Electronic Testing: Theory and Applications*, 18(4/5):385–400, Aug. 2002.

[27] D. Ernst, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, N. S. Kim, and K. Flautner. Razor: Circuit-level Correction of Timing Errors for Low-power Operation. *IEEE Micro*, 24(6):10–20, Nov.-Dec 2004.

[28] C. M. Fiduccia and R. M. Mattheyses. A Linear-Time Heuristic for Improving Network Partitions. In *Proceedings ACM/IEEE Design Automation Conference (DAC)*, pages 175–181, 1982.

[29] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman Publishers, 1979.

[30] S. Ghosh, S. Bhunia, A. Raychowdhury, and K. Roy. A novel delay fault testing methodology using low-overhead built-in delay sensor. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25(12):2934–2943, Dec. 2006.

[31] S. Goel and E. Marinissen. Effective and efficient test architecture design for SOCs. In *Proceedings IEEE International Test Conference (ITC)*, pages 529–538, 2002.

[32] S. K. Goel, K. Chiu, E. J. Marinissen, T. Nguyen, and S. Oostdijk. Test Infrastructure Design for the Nexperia^{TM} Home Platform PNX8550 System Chip. In *Proceedings Design, Automation, and Test in Europe (DATE) Designers Forum*, pages 108–113, Paris, France, Feb. 2004.

[33] P. D. Gross, R. Arunachalam, K. Rajagopal, and L. T. Pileggi. Determination of Worst-Case Aggressor Alignment for Delay Calculation. In *Proceedings International Conference on Computer-Aided Design (ICCAD)*, pages 212–219, 1998.

[34] M. Guler and H. Kilic. Understanding the Importance of Signal Integrity. *IEEE Circuits and Devices Magazine*, 15(6):7–10, November 1999.

[35] IEEE Std. 1500. *IEEE Standard for Embedded Core Test - IEEE Std. 1500-2004*. IEEE, New York, 2004.

[36] V. Iyengar, K. Chakrabarty, and E. J. Marinissen. Co-Optimization of Test Wrapper and Test Access Architecture for Embedded Cores. *Journal of Electronic Testing: Theory and Applications*, 18(2):213–230, Apr. 2002.

[37] N. Jha and S. Gupta. *Testing of Digital Systems*. Cambridge University Press, 2003.

[38] K. Mistry, et al. Voltage Overshoots and N-MOSFET Hot Carrier Robustness in VLSI Circuits. In *IEEE Reliability Phys. Symp.*, pages 65–71, 1994.

[39] W. H. Kao, C.-Y. Lo, M. Basel, and R. Singh. Parasitic Extraction: Current State of the Art and Future Trends. *Proceedings of the IEEE*, 89(5):729–739, May 2001.

[40] S. Kundu, S. T. Zachariah, Y.-S. Chang, and C. Tirumurti. On Modeling Crosstalk Faults. *IEEE Transactions on Computer-Aided Design*, 24(12):1909–1915, December 2005.

[41] E. Larsson and H. Fujiwara. Test Resource Partitioning and Optimization for SOC Designs. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 319–324, Napa, CA, Apr. 2003.

[42] Y. Leblebici. Design Considerations for CMOS Digital Circuits with Improved Hot-Carrier Reliability. *IEEE Journal of Solid-State Circuits*, 31(7):1014–1024, July 1996.

[43] J. Li, Q. Xu, Y. Hu, and X. Li. iFill: An Impact-Oriented X-Filling Method for Shift- and Capture-Power Reduction in At-Speed Scan-Based Testing. In *Proceedings IEEE/ACM Design, Automation, and Test in Europe (DATE)*, pages 1184–1189, 2008.

[44] K. S.-M. Li, C. L. Lee, C. Su, and J. E. Chen. Oscillation Ring Based Interconnect Test Scheme for SoC. In *Proceedings IEEE Asia South Pacific Design Automation Conference (ASP-DAC)*, pages 184–187, 2005.

[45] S. Lin and N. Chang. Challenges in Power-Ground Integrity. In *Proceedings International Conference on Computer-Aided Design (ICCAD)*, pages 651–654, 2001.

[46] X. Liu, M. S. Hsiao, S. Chakravarty, and P. J. Thadikaran. Efficient transition fault atpg algorithms based on stuck-at test vectors. *Journal of Electronic Testing: Theory and Applications*, 19(4):2003, August 2003.

[47] J. Ma, J. Lee, M. Tehranipoor, and A. Crouch. Test Pattern Generation for Open Defects in Power Distribution Networks. In *IEEE North Atlantic Test Workshop*, pages 31–36, 2008.

[48] M. B. K. Manjul Bhushan, Anne Gattiker and K. K. Das. Ring Oscillators for CMOS Process Tuning and Variability Control. *IEEE Transactions on Semiconductor Manufacturing*, 19(1):10–18, February 2006.

[49] E. J. Marinissen et al. A Structured And Scalable Mechanism for Test Access to Embedded Reusable Cores. In *Proceedings IEEE International Test Conference (ITC)*, pages 284–293, Washington, DC, Oct. 1998.

[50] E. J. Marinissen et al. On IEEE P1500's Standard for Embedded Core Test. *Journal of Electronic Testing: Theory and Applications*, 18(4/5):365–383, Aug. 2002.

[51] E. J. Marinissen, S. K. Goel, and M. Lousberg. Wrapper Design for Embedded Core Test. In *Proceedings IEEE International Test Conference (ITC)*, pages 911–920, Atlantic City, NJ, Oct. 2000.

[52] E. J. Marinissen, V. Iyengar, and K. Chakrabarty. A Set of Benchmarks for Modular Testing of SOCs. In *Proceedings IEEE International Test Conference (ITC)*, pages 519–528, Baltimore, MD, Oct. 2002.

[53] Y. Massoud, S. Majors, J. Kawa, T. Bustami, D. MacMillen, and J. White. Managing On-Chip Inductive Effects. 10(6):789–798, December 2002.

[54] A. V. Mezhiba and E. G. Friedman. Scaling Trends of On-Chip Power Distribution Noise. *IEEE Transactions on Computer-Aided Design*, 12(4):386–394, April 2004.

[55] S. Naffziger. Design Methodologies for Interconnects in GHz+ICs. In *Proceedings International Solid State Circuits Conference (ISSCC)*, 1999.

[56] N. S. Nagaraj, W. R. Hunter, P. R. Chidambaram, T. Y. Garibay, U. Narasimha, A. Hill, and H. Shichijo. Impact of Interconnect Technology Scaling on SOC Design Methodologies. In *Proceedings IEEE International Test Conference (ITC)*, pages 71–73, 2005.

[57] M. Nahvi and A. Ivanov. Indirect Test Architecture for SoC Testing. *IEEE Transactions on Computer-Aided Design*, 23(7):1128–1142, 2004.

[58] S. Natarajan, M. A. Breuer, and S. K. Gupta. Process Variations and Their Impact on Circuit Operation. In *Proceedings IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, pages 73–81, 1998.

[59] S. Nazarian, A. Iranli, and M. Pedram. Crosstalk Analysis in Nanometer Technologies. In *Proceedings ACM Great Lakes Symposium on VLSI (GLSVLSI)*, pages 253–258, 2006.

[60] P. Nordholz, D. Treytnar, J. Otterstedt, H. Grabinski, D. Niggemeyer, and T. W. Williams. Signal Integrity Problems in Deep Submicron Arising from Interconnects between Cores. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 28–33, April 1998.

[61] M. Nourani and A. Attarha. Built-In Self-Test for Signal Integrity. In *Proceedings ACM/IEEE Design Automation Conference (DAC)*, pages 613–618, 2001.

[62] M. Nourani and A. Attarha. Detecting Signal-Overshoots for Reliability Analysis in High-speed System-on-Chips. *IEEE Transactions on Reliability*, 51(4):494–504, December 2002.

[63] M. Nourani and A. Radhakrishnan. Testing On-Die Process Variation in Nanometer VLSI. *IEEE Design & Test of Computers*, 23(6):438–451, November 2006.

[64] S. Pei, H. Li, and X. Li. A low overhead on-chip path delay measurement circuit. In *Proceedings IEEE Asian Test Symposium (ATS)*, pages 149–154, 2009.

[65] I. Pomeranz and S. M. Reddy. On n-Detection Test Sets and Variable n-Detection Test Sets for Transition Faults. *IEEE Transactions on Computer-Aided Design*, 19(3):372–383, March 2000.

[66] H. Qian, S. R. Nassif, and S. S. Sapatnekar. Random walks in a supply network. In *Proceedings ACM/IEEE Design Automation Conference (DAC)*, pages 93–98, 2003.

[67] H. Qian and S. S. Sapatnekar. A hybrid linear equation solver and its application in quadratic placement. In *Proceedings International Conference on Computer-Aided Design (ICCAD)*, pages 905–909, 2005.

[68] J. Rabaey, A. Chandrakasan, and B. Nikolic. *Digital Integrated Circuits: A Design Perspective*. Prentice-Hall, 2nd edition, 2003.

[69] A. Raychowdhury, S. Ghosh, and K. Roy. A novel on-chip delay measurement hardware for efficient speed-binning. In *IOLTS '05: Proceedings of the 11th IEEE International On-Line Testing Symposium*, pages 287–292, Washington, DC, USA, 2005. IEEE Computer Society.

[70] T. Sato, S. Hagiwara, T. Uezonon, and K. Masu. Weakness Identification for Effective Repair of Power Distribution Network. In *International Workshop on Integrated Circuit Design, Power and Timing Modeling, Optimization and Simulation*, pages 222–231, Oct. 2007.

[71] J. Saxena, K. Butler, V. Jayaram, and S. Kundu. A Case Study of IR-Drop in Structured At-Speed Testing. In *Proceedings IEEE International Test Conference (ITC)*, 2003.

[72] D. Schroder and J. F. Babcock. Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufacturing. *Journal of Appllied Physics*, 94(1):1–18, July 2003.

[73] M. H. Schulz and F. Brglez. Accelerated Transition Fault Simulation. In *Proceedings ACM/IEEE Design Automation Conference (DAC)*, pages 237–243, 1987.

[74] K. Sekar and S. Dey. LI-BIST: A Low-Cost Self-Test Scheme for SoC Logic Cores and Interconnects. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 417–422, 2002.

[75] N. Selvakkumaran and G. Karypis. Multi-Objective Hypergraph Partitioning Algorithms for Cut and Maximum Subdomain Degree Minimization. In *Proceedings International Conference on Computer-Aided Design (ICCAD)*, pages 726–733, 2003.

[76] J. P. Shen, W. Maly, and F. J. Ferguson. Inductive fault analysis of mos integrated circuits. *IEEE Design and Test*, 2(6):13–26, November 1985.

[77] W. Sirisaengtaksin and S. K. Gupta. Enhanced Crosstalk Fault Model and Methodology to Generate Tests for Arbitrary Inter-Core Interconnect Topology. In *Proceedings IEEE Asian Test Symposium (ATS)*, pages 163–169, November 2002.

[78] S. Tabatabaei and A. Ivanov. An Embedded Core for Sub-Picosecond Timing Measurements. In *Proceedings IEEE International Test Conference (ITC)*, pages 129–137, October 2002.

[79] K. T. Tang and E. G. Friedman. Simultaneous switching noise in onchip cmos power distribution networks. *IEEE Transactions on VLSI Systems*, 10(4):487–493, August 2002.

[80] M. H. Tehranipour, N. Ahmed, and M. Nourani. Testing SoC Interconnects for Signal Integrity Using Bounary Scan. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 158–163, April 2003.

[81] M. H. Tehranipour, N. Ahmed, and M. Nourani. Testing SoC Interconnects for Signal Integrity Using Extended JTAG Architecture. *IEEE Transactions on Computer-Aided Design*, 23(5):800–811, May 2004.

[82] C. Tirumurti, S. Kundu, S. Sur-Kolay, and Y.-S. Chang. A Modeling Approach for Addressing Power Supply Switching Noise Related Failures of Integrated Circuits. In *Proceedings IEEE/ACM Design, Automation, and Test in Europe (DATE)*, pages 1078–1083, 2004.

[83] M.-C. Tsai, C.-H. Cheng, and C.-M. Yang. An all-digital high-precision built-in delay time measurement circuit. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 249–254, 2008.

[84] P. Varma and S. Bhatia. A Structured Test Re-Use Methodology for Core-Based System Chips. In *Proceedings IEEE International Test Conference (ITC)*, pages 294–302, Washington, DC, Oct. 1998.

[85] L.-T. Wang, C. E. Stroud, and N. A. Touba. *System-on-Chip Test Architectures: Nanometer Design for Testability*. Morgan Kaufmann Pub., 2007.

[86] X. Wang, M. Tehranipoor, and R. Datta. Path-RO: A Novel On-Chip Critical Path Delay Measurement Under Process Variations. In *Proceedings International Conference on Computer-Aided Design (ICCAD)*, pages 640–646, 2008.

[87] X. Wang, M. Tehranipoor, and R. Datta. A Novel Architecture for On-chip Path Delay Measurement. In *Proceedings IEEE International Test Conference (ITC)*, pages 1–10, 2009.

[88] Q. Xu and N. Nicolici. On Reducing Wrapper Boundary Register Cells in Modular SOC Testing. In *Proceedings IEEE International Test Conference (ITC)*, pages 622–631, Charlotte, NC, Sept. 2003.

[89] Q. Xu and N. Nicolici. Multi-Frequency Test Access Mechanism Design for Modular SOC Testing. In *Proceedings IEEE Asian Test Symposium (ATS)*, pages 2–7, Kenting, Taiwan, Nov. 2004.

[90] Q. Xu and N. Nicolici. Resource-Constrained System-on-a-Chip Test: A Survey. *IEE Proceedings, Computers and Digital Techniques*, 152(1):67–81, January 2005.

[91] Q. Xu and N. Nicolici. Multi-Frequency TAM Design for Hierarchical SOCs. *IEEE Transactions on Computer-Aided Design*, 25(1):181–196, Jan. 2006.

[92] S.-Y. Yang, C. A. Papachristou, and M. Taib-Azar. Improving Bus Test via I_{DDT} and Boundary Scan. In *Proceedings ACM/IEEE Design Automation Conference (DAC)*, pages 307–312, 2001.

[93] T. Zhang and S. S. Sapatnekar. Simultaneous Shield and Buffer Insertion for Crosstalk Noise Reduction in Global Routing. In *Proceedings International Conference on Computer Design (ICCD)*, pages 93–98, 2004.

[94] D. Zhao and S. Upadhyaya. Dynamically Partitioned Test Scheduling with Adaptive TAM Configuration for Power-Constrained SoC Testing. *IEEE Transactions on Computer-Aided Design*, 24(6):956–965, June 2005.

[95] Y. Zhao, L. Chen, and S. Dey. On-Line Testing of Multi-Source Noise-Induced Errors on the Interconnects and Buses of System-on-Chips. In *Proceedings IEEE International Test Conference (ITC)*, pages 491–499, October 2002.

[96] Y. Zhao, S. Dey, and L. Chen. Double Sampling Data Checking Technique: An Online Testing Solution for Multisource Noise-Induced Errors on On-Chip Interconnects and Buses. *IEEE Transactions on VLSI Systems*, 12(7):746–755, July 2004.

[97] Q. K. Zhu. *Power Distribution Network Design for VLSI.* John Wiley & Sons, 2004.

[98] W. Zou, S. M. Reddy, I. Pomeranz, and Y. Huang. SOC Test Scheduling Using Simulated Annealing. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 325–330, Napa, CA, Apr. 2003.