國立臺灣大學電機資訊學院電子工程學研究所

碩士論文

Graduate Institute of Electronics Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master's Thesis

於邏輯掃描測試培養中除錯及預防異常高的最小工作電壓

Debugging and Preventing Abnormally High V_{min} during Logic Scan Test Bring-up

劉旻鑫

Min-Hsin Liu

指導教授: 李建模 博士

Advisor: James Chien-Mo Li, Ph.D.

中華民國 114 年 7 月

July, 2025

doi:10.6342/NTU202502538

Acknowledgements

兩年的碩士生活隨著論文的完成即將畫下句點，這段期間充滿了挑戰與挫折，也伴隨著許多人的鼓勵與幫助。在此，我衷心感謝這一路上所有支持與陪伴我的師長、朋友與家人，因為有你們的支持，我才能順利完成這篇論文，也為我的研究生涯留下豐富而深刻的回憶。

首先，我要特別感謝我的指導老師。在升碩一之前，老師鼓勵我們前往業界實習，這段經驗讓我對產業運作有更實際的了解，也為後來的研究與職涯奠定了基礎。碩士期間，老師即使事務繁忙，仍固定與我們開會、提供研究建議，從文獻閱讀、題目發想到實驗設計與論文撰寫，都給予我非常細緻的指導。當實驗遇到瓶頸時，老師總是能幫我從不同角度思考問題，引導我找到新的方向，讓我明白研究的過程本就伴隨著挑戰，而失敗也能成為前進的養分。此外，老師經常提醒我們保持規律運動與正面心態，這讓我在研究過程中學會調節壓力，也更有動力面對困難。謝謝老師在這兩年中全方位的提攜與鼓勵，讓我受益良多。

接著，我要感謝高通所有同仁在研究上的支持與幫助。特別感謝 Chris 在整個論文過程中提供大量資源與實作建議，每週兩次的 meeting 提供寶貴意見與技術指導，甚至利用休息時間測試所需資料；在研究之外，也常給我心理上的鼓勵與關心，使我在挫折時能勇敢前行。感謝 Szu Huat Goh、Mason Chern、Khee Sang Cheah 及秉翰學長在會議中給我建議與技術支援，Subhadip Kundu 提供研究方向與技術協助，讓我的研究能夠順利進行。

doi:10.6342/NTU202502538

感謝實驗室的每一位夥伴。謝謝昀昇學長在我加入 diagnosis 組後不吝給我建議，解答我對 paper 的疑問，在 meeting 時也時常幫我解釋我說不清楚的概念，並在我思考研究方向時給予很多幫助。也謝謝定緯一路上的討論與陪伴，希望你之後的研究一路順利。另外，感謝承運、明膀、旭釭、雲丰、詠珍、政毓等學長姐提供的研究建議，以及筑云、柏傑、濰紳、官欣、力揚、孟宸等同學一起度過討論與趕進度的時光，還有緦綺、銘澤、紹宇、秉宸等學弟們在會議中的回饋與提問，都讓我學到很多。

接下來，我想感謝研究所期間對我非常重要的朋友們。謝謝楷崴，儘管你遠在美國，卻始終不缺席地陪我聊天、分享彼此的經歷與低潮，也總是給我很多鼓勵與支持。從碩零到現在，你的陪伴讓這段旅程多了很多溫暖和勇氣，真心祝福你在美國的事業與感情一切順利。謝謝虹伶和耕頡，每週一起午餐的時光，總能讓我短暫抽離繁忙的研究節奏，聊聊進度、感情和生活，讓人感到安心又踏實，相信你們也都快完成碩士學業了，我們一起加油！

最後，我想感謝家人在我身旁不斷給予我鼓勵。感謝我的哥哥們在我鬱悶時和我聊天，分享彼此的困難並釋放壓力，在我對於未來的選擇感到困惑時，能夠理性的和我分析所有選擇的利弊並給予我支持。感謝我的父母在學業上從來不會給我壓力，總是默默支持我所選擇的路，也讓我能夠安心專注於自己的步調與理想，希望我今天的成果，能讓你們感到驕傲。

doi:10.6342/NTU202502538

摘要

在行動晶片的品質驗證中，at-speed 邏輯掃描測試是一項重要工具。在初期測試樣本導入階段，若測試展現出異常偏高的 V_{min}，可能導致過度測試，進而影響量產良率，這種情況尤其在測試的 V_{min} 顯著高於實際系統工作負載的 V_{min} 時更為嚴重，在這類情況下，工程師會對 at-speed 邏輯掃描測試進行除錯，以找出並解決造成高 V_{min} 的根本原因。本論文介紹一個 V_{min} 除錯的案例研究，透過一系列實驗，找出問題根源為某些測試樣本擷取了未受約束路徑的響應。我們提出了晶片前期（pre-silicon）與後期（post-silicon）的方法，藉由預防問題樣本並減少測試導入期間的除錯負擔，以改善 V_{min}，這些方法已在 ATE（自動測試設備）上驗證，能有效改善 V_{min}，提升幅度為 28.83mV 至 39.33mV，且測試向量僅增加 0% 至 0.5%。

關鍵字：測試診斷、延遲測試、最低操作電壓、未受約束路徑

doi:10.6342/NTU202502538

Abstract

At-speed logic scan tests are an important tool to ensure desired quality in mobile chips. During initial test pattern bring-up, tests that exhibit an unexpectedly high V_{min} pose a risk of over-testing and production yield loss. This is particularly problematic if the V_{min} of the test is significantly higher than that of the functional system workloads. In such situations, the at-speed logic scan test is debugged to find and resolve the source of the high V_{min}. This thesis describes an example case study of V_{min} debug, in which a series of experiments are performed to identify the root cause as individual test patterns that capture the responses of unconstrained paths. We propose pre-silicon and post-silicon methods to improve V_{min} by preventing problematic patterns and reducing the debug effort during test bring-up. Our methods have been verified on ATE to effectively improve V_{min} by 28.83mV to 39.33mV with 0% to 0.5% pattern count inflation.

Keywords: Diagnosis, Delay Test, V_{min}, Unconstrained Path

doi:10.6342/NTU202502538

Contents

doi:10.6342/NTU202502538

doi:10.6342/NTU202502538

List of Figures

doi:10.6342/NTU202502538

List of Tables

doi:10.6342/NTU202502538

Denotation

Term	**Definition**
Test bring-up	A phase before production to ensure that test specifications are met and test patterns will not induce significant yield loss
Unconstrained path	Paths whose timing behavior would not be checked in STA, including false paths and multi-cycle paths
V_{spec}	The voltage requirement that chips can operate within
V_{min}	The minimum voltage that chips can operate
Pattern-set V_{min}	V_{min} of a pattern set
Per-pattern V_{min}	V_{min} of a pattern in a pattern set
Outlier Pattern	Patterns that have abnormally high V_{min} in a pattern set
Majority Pattern	Patterns that have normal V_{min} in a pattern set
PS	Pattern set without any power constraints
$PS_{10} - PS_{15}$	Pattern set with WSA threshold set to 10%-15%
PS', PS'_{13}	Debugged PS, PS_{13} with the pre-silicon method
PS'', PS''_{13}	Debugged PS, PS_{13} with the post-silicon method and the first masking technique
PS''', PS'''_{13}	Debugged PS, PS_{13} with the post-silicon method and the second masking technique

doi:10.6342/NTU202502538

Chapter 1 Introduction

1.1 Motivation

When validating a new test pattern set for the first time, a key performance metric is V_{min}, defined as the minimum operating voltage at which a chip passes the test. A test pattern set that causes an excessively high V_{min} could lead to over-testing, which in turn creates yield loss in volume production. As part of the test bring-up process, we must debug such pattern sets to achieve a lower, acceptable V_{min} and prevent these negative consequences.

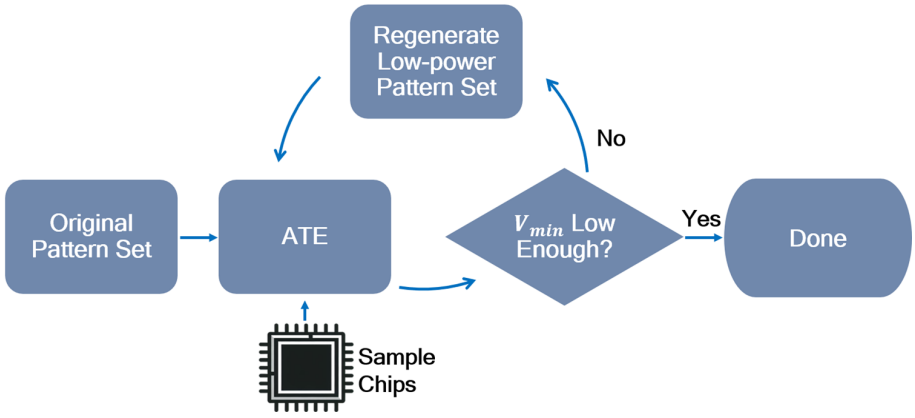
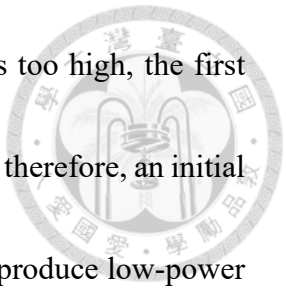


Figure 1.1: Traditional flow to debug and improve V_{min} during test-bring

doi:10.6342/NTU202502538



Traditionally, if the V_{min} of an at-speed logic scan pattern set is too high, the first suspected root cause is excessive power consumption [1,2]. To debug, therefore, an initial step is often to regenerate the pattern set with power-aware ATPG to produce low-power patterns. Figure 1.1 depicts this traditional flow to improve V_{min} of a pattern set. Given a test pattern set, test engineers test some sample chips on *Automated Test Equipment* (ATE) and collect the V_{min}. If the V_{min} is too high, *Design-for-Test* (DFT) engineers regenerate a low-power pattern set, and the sample chips are tested again. This process is repeated with stricter power constraints until the pattern set has sufficiently low V_{min}.

However, there are some drawbacks to the traditional flow. First of all, without an understanding of the root causes of high V_{min} pattern sets, this flow may not be suitable for all cases. Second, the iterative back and forth between pre-silicon and post-silicon to ensure satisfactory V_{min} of the regenerated pattern set is extremely time-consuming and requires significant human effort. What's worse, if power is not the root cause of high V_{min}, this flow may be ineffective. Last but not least, regenerating a pattern set with stricter power constraints is likely to produce serious pattern count inflation, increasing the test time. With all these disadvantages, we need a new method to efficiently improve V_{min}.

Conventionally, a high V_{min} of a pattern set is considered to be associated with power

doi:10.6342/NTU202502538

supply noise like IR drop and power supply droop [1–3]. Since scan test patterns would typically induce much higher switching activity compared to functional mode, circuits could have higher power consumption. When it comes to at-speed tests, the high frequency would lead to even more critical power consumption. Excessive power consumption would require a higher voltage so that circuits could operate correctly, which increases V_{min}. Nonetheless, in the case presented in this paper, no correlations have been found between power supply noise and V_{min}, suggesting an alternate cause for increased V_{min}.

Apart from power, timing is another factor that could result in the high V_{min} of a pattern set. Before sign-off, engineers would check all paths meet the timing requirements with the STA tool for some corners. However, actual operating voltages may shift due to process variation, which makes it hard to estimate timing accurately. As a result, some paths may have worse slacks than those calculated by the STA tool, which may lead to higher V_{min}. In addition to process variation, unconstrained paths are another possible root cause of the high V_{min} of a pattern set. The timing of unconstrained paths, which includes false paths and multi-cycle paths, would not be checked when engineers apply the STA tool. If the responses of some unconstrained paths are captured, they would possibly fail at higher voltages, which thus increases V_{min}. In our debugging attempts, we apply path delay fault simulation and one-hot patterns to confirm that the responses of unconstrained

doi:10.6342/NTU202502538

paths are captured by the outlier patterns. To conclude, we find that the root cause of high V_{min} pattern sets is unconstrained paths in our case.

1.2 Debug Attempts and Proposed Methods

In our pattern sets, we find that some outlier patterns have higher per-pattern V_{min}, limiting the V_{min} of pattern sets. Figure 1.2 shows the V_{min} distribution of a pattern set. We can see that some patterns have significantly higher per-pattern V_{min} than the others. By debugging the outlier patterns, we may potentially improve V_{min} without excessive pattern count inflation.

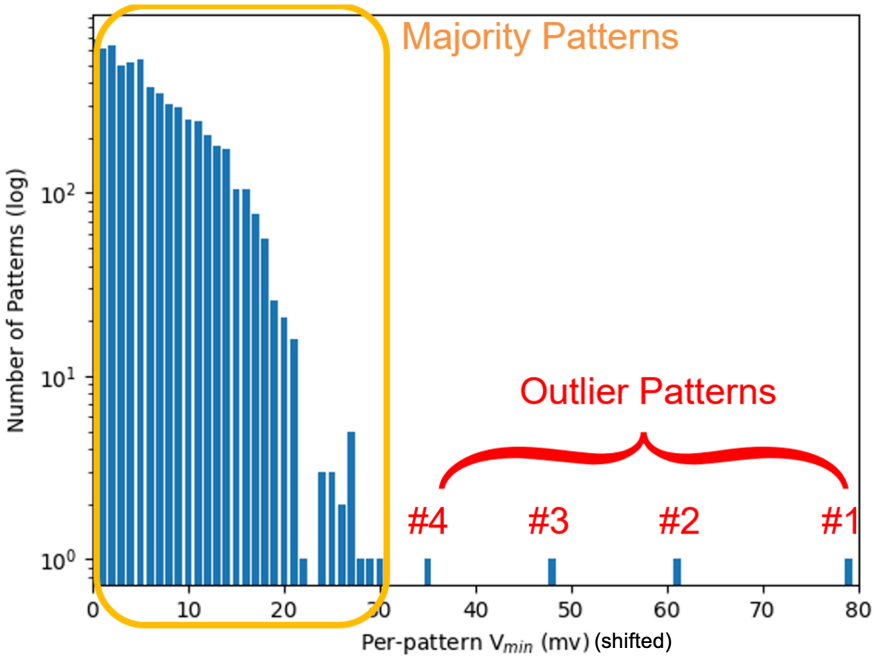


Figure 1.2: Shifted V_{min} distribution of a pattern set

In this thesis, we first perform five debugging attempts to find out the root cause

doi:10.6342/NTU202502538

inducing abnormally high V_{min} patterns. The first debugging attempt aims to find the effectiveness of power-aware ATPG. We apply power-aware ATPG to generate pattern sets with different power constraints. The results shows that directly apply power-aware ATPG with stricter power constraints may be ineffective to reduce V_{min}. The second debugging attempt aims to find the correlation between global dynamic power and V_{min}. We calculate the global power metrics for each pattern and analyze the correlation between them and per-pattern V_{min}. The results shows that no correlation between global dynamic power and per-pattern V_{min}. Besides, the outlier patterns don't have extremely high values of global dynamic power. Figure 1.3 shows a figure of the correlation between one global power metric and per-pattern V_{min}. The third debugging attempt aims to find the correlation

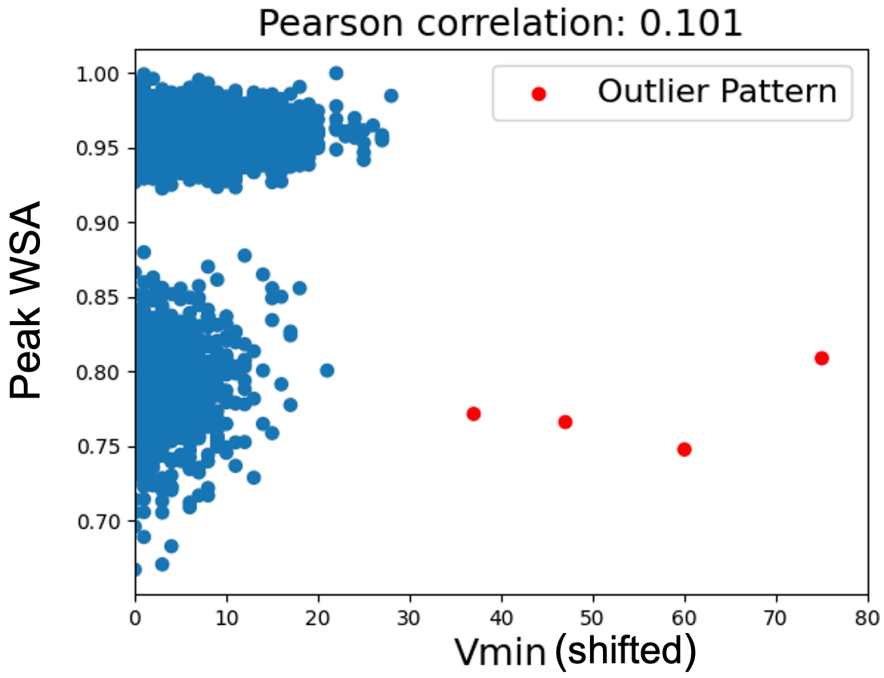
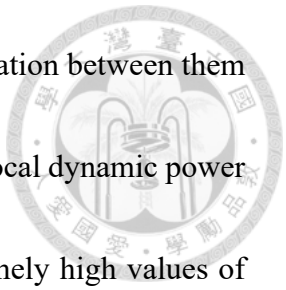


Figure 1.3: Correlation between global peak WSA and per-pattern V_{min} (shifted)

between local dynamic power and V_{min}. We calculate the local power metrics for each

doi:10.6342/NTU202502538



grid, extract the highest value for each pattern, and analyze the correlation between them and per-pattern V_{min}. The results shows that no correlation between local dynamic power and per-pattern V_{min}. Besides, the outlier patterns don't have extremely high values of local dynamic power. The fourth debugging attempt aims to find the correlation between dynamic power around the longest path and V_{min}. We calculate the local power metrics around the longest path and analyze the correlation between them and per-pattern V_{min}. The results shows that no correlation between dynamic power around the longest path and per-pattern V_{min}. Besides, the outlier patterns don't have extremely high values of local dynamic power around the longest path. The fifth debugging attempt aims to find the correlation between unconstrained paths and V_{min}. We apply path delay fault simulation to confirm that whether the outlier patterns fail at higher voltages because of unconstrained paths. The results shows that unconstrained paths are the root cause of the outlier patterns. Apart from path delay fault simulation, we also generate one-hot patterns to validate the results.

To deal with the root cause of the outlier patterns, we propose the pre-silicon and post-silicon methods, which are shown in Figure 1.4 and Figure 1.5. They can prevent pattern sets with high V_{min} caused by unconstrained paths. The pre-silicon method can be applied before we apply test pattern sets on ATE. It masks the capture flip-flops of uncon-

doi:10.6342/NTU202502538

strained paths that could potentially lead to outlier patterns. The pre-silicon method can significantly improve the V_{min} of pattern sets. However, it requires path definition files, which can be extracted from Synopsys Design Constraint (SDC) files, and path delay fault simulation, which takes more time. Besides, it leads to little pattern count inflation. The post-silicon method can only be applied after we apply test pattern sets on ATE and obtain the fail logs. We have two masking techniques for post-silicon method. The first one masks the capture flip-flops of unconstrained paths that could potentially lead to outlier patterns like the pre-silicon method. The second one directly masks the failing flip-flops in the fail logs. Even though the post-silicon method requires fail logs, it can achieve similar V_{min} improvements as the pre-silicon method with barely any pattern count inflation.

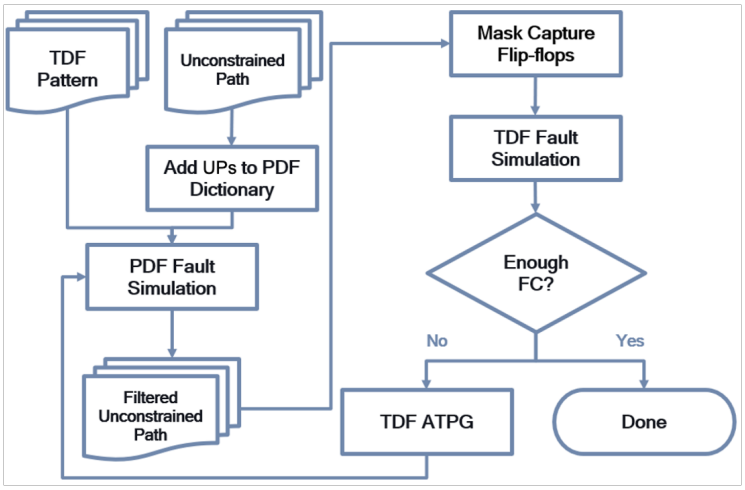


Figure 1.4: Pre-silicon method

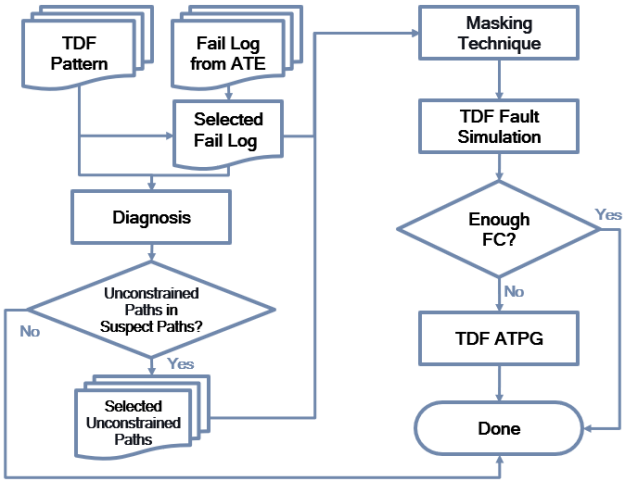


Figure 1.5: Post-silicon method

doi:10.6342/NTU202502538

1.3 Contributions

In this paper, we conduct several debug experiments to identify the root cause of a high V_{min} pattern set. In our case study, we find that the correlations between power and per-pattern V_{min} are very low. The root cause in our case is the unconstrained paths. To deal with the unconstrained paths, we propose two methods, pre-silicon and post-silicon, to prevent and efficiently improve the high V_{min} of a pattern set. The pre-silicon method improves V_{min} of a pattern set without ATE iterations at the cost of pattern count inflation. The post-silicon method requires failing information from ATE and can improve V_{min} with less pattern count inflation. Our proposed methods are verified with real industrial data to reduce 28.83mV to 39.33mV with 0% to 0.5% pattern count inflation. With our methods, we can improve V_{min} at early stages and thus reduce the iterative V_{min} debugging loop in Figure 1.1 to solve over-testing issues caused by high V_{min}.

1.4 Organization

The rest of this thesis is organized as follows. In Chapter 2, we introduce the background and related works of this thesis. In Chapter 3, we present a case study demonstrating our debug results of an industrial core with a V_{min} issue, step by step. In Chapter 4,

doi:10.6342/NTU202502538

we propose two prevention flows, pre-silicon and post-silicon, to improve V_{min} by solving the root cause found in Chapter 3. Then, Chapter 5 describes the experimental results on industrial silicon data, and finally Chapter 7 concludes this thesis.

doi:10.6342/NTU202502538

Chapter 2

Background and Preliminaries

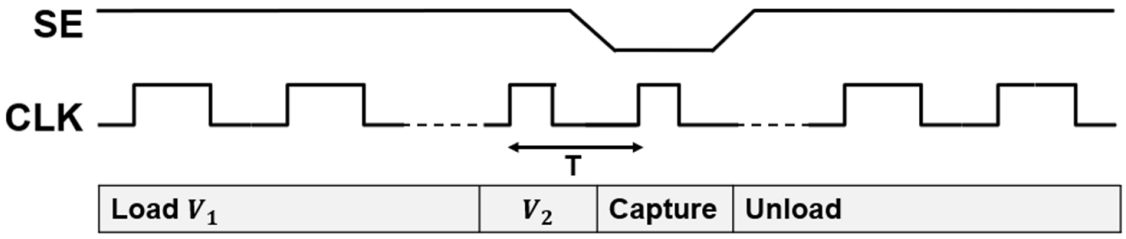
2.1 At-speed Testing

At-speed testing is a critical technique in industry [4]. It is used to verify the performance and reliability of integrated circuits under their actual operating frequency. Unlike traditional methods that test at reduced speeds, at-speed testing runs at operational frequency to better detect timing-related defects. Certain faults that only occur under operational frequency can be detected through at-speed testing. At-speed testing could improve fault coverage and product reliability, making it indispensable in the industry.

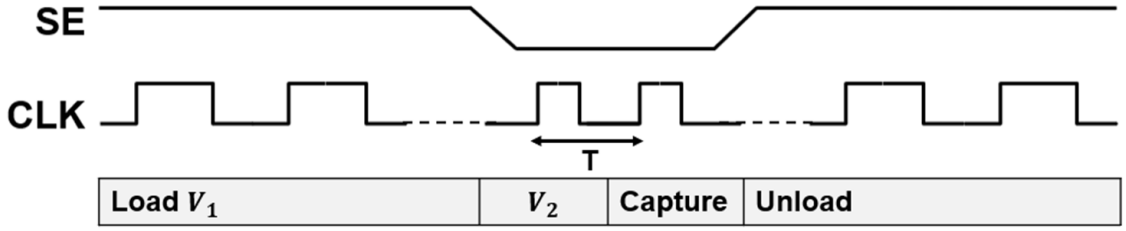
One of the most common timing-related fault model that requires at-speed testing is the transition delay fault. Transition delay faults require a two-pattern test—the first pattern initializes the circuit state, and the second pattern launches transitions and propagates the fault effect to an observable output. ATPG tools typically use two ways to generate

doi:10.6342/NTU202502538

TDF patterns: Launch-off-shift (LOS) [5,6] and launch-off-capture (LOC) [7]. Figure 2.1 shows how the scan enables signal and the clock signal work in LOS and LOC, where SE and CLK represent the scan enable signal and clock signal, respectively. The SE signal is high during test mode (shift) and low when in functional mode.



(a) Launch-off-shift



(b) Launch-off-capture

Figure 2.1: Transition delay test

LOS is an easier way to generate TDF patterns, but it loses some testability due to the circuit's structure. Furthermore, due to the limitation of the fall time of the scan enable signal shown in Figure 2.1 (a), LOS cannot achieve real at-speed testing. LOC, on the other hand, has higher testability but requires more time for the ATPG tool to generate patterns. To achieve even better testability, users can increase the number of capture cycles at the cost of longer runtime. In addition to testability, LOC also allows at-speed testing because the fall time of SE does not restrict the clock speed. Since LOC, compared with LOS, has

doi:10.6342/NTU202502538

more advantages, it is more commonly used in the industry. As a result, in our work, we would only discuss LOC TDF patterns.

2.2 Test Bring-up

Test bring-up is a phase to ensure that test specifications are met and test patterns will not induce significant yield loss when enabled in volume production. In this phase, only a small number of fault-free chips would be tested on ATE. Based on the test results, test engineers check that each specification is met. One of the performance specifications checked during this process is V_{min}. The validation V_{min} for a test must be lower than the product specification voltage, usually by some margin. If validation V_{min} is too high and there is insufficient margin, process variation could lead to yield loss in volume production.

The general test bring-up flow is shown as Figure 2.2. DFT engineers first generate initial pattern sets for each core of a design and provide them for testing. Test engineers apply the pattern set on ATE with a few sample chips. ATE application generates fail logs for each chip. Fail logs contain crucial information for diagnosis, such as failing cycles, failing pattern IDs, and failing scan chains. Test engineers and DFT engineers work together to identify the potential root causes and regenerate new pattern sets for

doi:10.6342/NTU202502538

resolution or further debugging. This process may be repeated several times to ensure the quality of the test. If root causes cannot be found efficiently, test bring-up becomes very time-consuming. The test bring-up process is essential for reducing time-to-market and ensuring the reliability of the final product. It helps in identifying potential issues early, allowing for corrections before volume production.

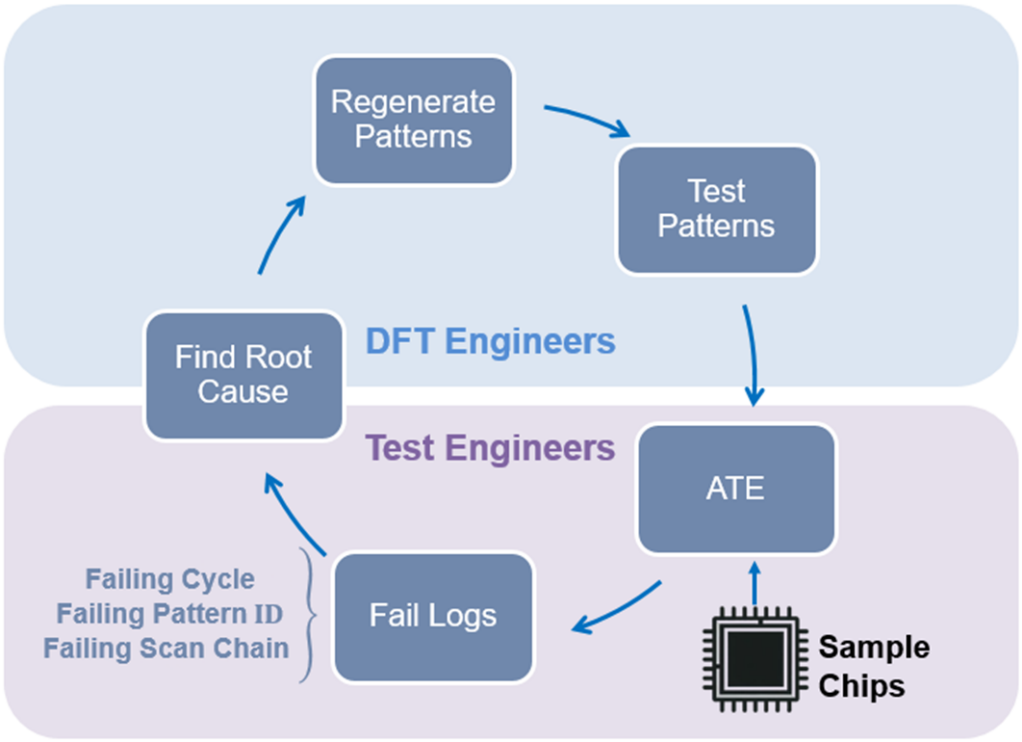


Figure 2.2: Example of test bring-up

2.3 V_{min} Test

V_{min} is a critical parameter in IC testing since it is a representative indicator of a chip's performance, power consumption, and reliability. It refers to the lowest supply voltage at which a chip or a unit can function correctly. When V_{min} binning is utilized,

doi:10.6342/NTU202502538

a chip with high V_{min} will consume more power than a typical chip, particularly in low-power mode [8]. The purpose of V_{min} tests is to perform this binning and screen out bad chips with excessively high V_{min}.

As technology advances, the impact of process variations becomes increasingly significant [9], underscoring the critical importance of V_{min} test in ensuring chips' power performance. During testing, if V_{min} test results are different from functional V_{min}, the supply voltage would be overestimated or underestimated, which leads to power inefficiency or failures. Therefore, accurately testing V_{min} is critical, especially under process variation.

V_{min} is also switching activity dependent. Compared to functional mode, the switching activity of test mode tends to be larger, which results in higher V_{min} test results than functional V_{min}. Even though functional V_{min} and V_{min} test results have a similar trend, it is hard to correlate them [10]. During testing, if V_{min} test results are much higher than functional V_{min}, it would lead to the over-testing problem. Therefore, test pattern sets play an important role in V_{min} testing.

doi:10.6342/NTU202502538

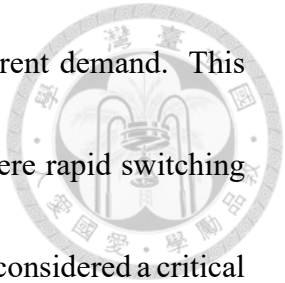
2.4 Suspected Root Causes for High V_{min}

Both power supply noise and timing behavior would significantly influence the V_{min} test results of integrated circuits (ICs). During testing, V_{min} may increase artificially. Both power supply noise and timing behavior could be the root cause of high V_{min} pattern sets. Power supply noise includes IR drop and power supply droop. Timing behavior would be affected by voltages, process, and unconstrained paths.

IR drop is the first suspected root cause of the high V_{min} problem. It is the voltage drop that occurs due to the resistance in the power delivery network as current flows through it. It reduces the effective voltage available to the IC components, which potentially causes timing violations and functional failures. IR drop can be categorized into static and dynamic IR drop. Static IR drop occurs when the circuit is in a steady state, while dynamic IR drop happens during active switching. Both types of IR drop can increase V_{min} test results due to voltage loss. During at-speed testing, IR drop has been recognized as a significant issue [3, 11, 12]. Local IR drop problems, in particular, are considered critical during testing. [13, 14] split a core into several grids to consider local IR drop.

Power supply droop is the second suspected root cause of the high V_{min} problem.

doi:10.6342/NTU202502538



It is a reduction in supply voltage caused by sudden changes in current demand. This phenomenon is particularly critical during high-speed operations where rapid switching can lead to significant current fluctuations [15]. Power droop has been considered a critical factor that degrades circuits' performance during at-speed test [1, 2, 12, 16].

Switching activity is an indicator of the severity of power supply noise. It refers to the transitions of digital signals within ICs. High switching activity leads to increased demand for current spikes, which causes rising temperatures and timing degradation. The consequences worsen the influences of power supply noise and increase V_{min} as chips require higher voltage to maintain reliable operation in strict environments. *Weighted switching activity* (WSA) [17] considers the load capacitance of each gate by weighting switching activity with the number of gates' fan-out. WSA is effective in conveying the severity of power supply noise. The definition of WSA is shown as Equation (2.1), where $Switch$ is 1 if a gate is switching and $\#gatefanout$ is the number of fan-out of a gate. WSA is used as a parameter of power-aware ATPG. WSA in test mode is typically higher than that in functional mode. That means, power supply noise in test mode is higher than that in functional mode. It would finally cause higher V_{min} and potential yield loss in the post-silicon phase if not properly handled.

doi:10.6342/NTU202502538

$$WSA = \Sigma_{\forall gate} Switch \times (1 + \#gate fanout) \quad (2.1)$$

Timing differences among various voltages are the third suspected root cause of the high V_{min} problem. In pre-silicon, all timing requirements must be met to ensure flip-flops capture the correct values when switching transitions occur in the circuit. Since the timing performance of each instance varies with voltages and only some corner cases are considered, the slack of each path cannot be accurately calculated across all voltages [18]. Besides, the actual operating voltages may shift due to process variation, which leads to difficulties of accurate timing estimation. The inaccuracy of timing estimation at an unconsidered voltage mode may lead to some unexpected failing paths. For example, at the specification voltage, suppose $path_1$ has a larger slack than $path_2$. However, when the voltage is lowered, the slack of $path_1$ may drop drastically due to transistors' characteristics and lead to failure at a higher voltage than $path_2$.

The last suspected root cause of high V_{min} is unconstrained paths. *Unconstrained paths* include false paths and multi-cycle paths. Before sign-off, engineers would use the STA tool to confirm that all paths meet the setup and hold time requirements at corner cases. However, the timing of unconstrained paths would not be fixed since unconstrained paths should not be activated in functional mode. If some unconstrained paths are

doi:10.6342/NTU202502538

not properly handled during pattern generation, the patterns that capture the responses of unconstrained paths would lead to unexpected failures. Finally, it could cause high V_{min} or even an over-testing problem [19–21].

doi:10.6342/NTU202502538

Chapter 3 Per-pattern V_{min} Debug

In this section, we present a case study with five V_{min} debugging attempts on an industrial core containing approximately 11 million gates. Eight sample chips have been tested thoroughly. These sample chips have abnormally high V_{min} during test bring-up. In Section 3.1, we first describe how to obtain per-pattern V_{min} and identify outlier patterns. Next, we present debugging results based on power metrics or timing paths, as detailed from Section 3.2 to Section 3.6. All pattern sets discussed in this section are logic scan transition delay fault (TDF) test patterns. Tests are applied at-speed (*i.e.*, at a frequency consistent with their target functional performance mode).

3.1 Per-pattern V_{min} and Outlier Patterns

During test bring-up, we find that the V_{min} of each test pattern within a pattern set is not the same. Some test patterns have abnormally high V_{min}. Therefore, we carry out experiments that measure V_{min} for each test pattern. The *per-pattern* V_{min} is defined as

doi:10.6342/NTU202502538

the lowest voltage at which a pattern does not fail, while the *pattern-set* V_{min} is defined as the lowest voltage at which the entire pattern set does not fail. The relationship between pattern-set V_{min} and per-pattern V_{min} of a chip c is shown in Equation (3.1), where P represents a pattern set, p represents a pattern of the pattern set.

$$V_{min}(c, P) = max_{\forall p \in P} V_{min}(c, p) \tag{3.1}$$

Figure 3.1 illustrates an example of determining the per-pattern V_{min} value within a pattern set. In this example, the pattern set is composed of five patterns. We test this pattern set at the initial voltage, which is set to the specification supply voltage (V_{spec}). After collecting the fail logs at this voltage, we reduce the supply voltage by 1% of V_{spec} and test the whole pattern set again. We iteratively go through the process until enough fail logs are collected. Figure 3.1 (a) shows the test results of a pattern set, where green boxes indicate that the tested chips pass the pattern at the voltage. Red boxes, on the contrary, indicate failures of the pattern at the voltage. Figure 3.1 (b) presents the corresponding per-pattern and pattern-set V_{min} based on the test results.

Identifying outlier patterns in a pattern set helps us debug cores with abnormally high V_{min}. Outlier patterns are those with significantly higher per-pattern V_{min} compared to the majority of patterns during test bring-up. Figure 3.2 gives an example of how to identify

doi:10.6342/NTU202502538

Pattern ID	$0.96V_{spec}$	$0.97V_{spec}$	$0.98V_{spec}$	$0.99V_{spec}$	V_{spec}
$Pattern_A$					
$Pattern_B$					
$Pattern_C$					
$Pattern_D$					
$Pattern_E$					

(a) Pass/Fail of 5 patterns

Pattern ID	Per–pattern V_{min}(mV)	Pattern-set V_{min}(mV)
$Pattern_A$	V_{spec}	V_{spec}
$Pattern_B$	$0.99V_{spec}$	
$Pattern_C$	$0.97V_{spec}$	
$Pattern_D$	$0.98V_{spec}$	
$Pattern_E$	$0.97V_{spec}$	

(b) Pattern-set V_{min} and per-pattern V_{min}

Figure 3.1: Example of pattern-set V_{min} and per-pattern V_{min}

the outlier patterns in a pattern set with only one sample chip. Given a pattern set during test bring-up, we collect the per-pattern V_{min} of the pattern set using ATE. After collecting all per-pattern V_{min} values, we draw a bar chart like Figure 3.2. In the bar chart, the x-axis represents the measured per-pattern V_{min} with offset, and the y-axis represents the number of patterns on the log scale. Each bar represents the number of patterns that have the same per-pattern V_{min}. We see that the patterns enclosed by the orange line are the majority patterns. Most patterns belong to the majority patterns and have similar per-pattern V_{min}. Outlier patterns, colored in red, are those patterns with high per-pattern V_{min} compared to the majority patterns. In this case, we can improve pattern-set V_{min} by about $40mV$ if we

doi:10.6342/NTU202502538

successfully debug all four outlier patterns, which is a significant improvement with little coverage loss.

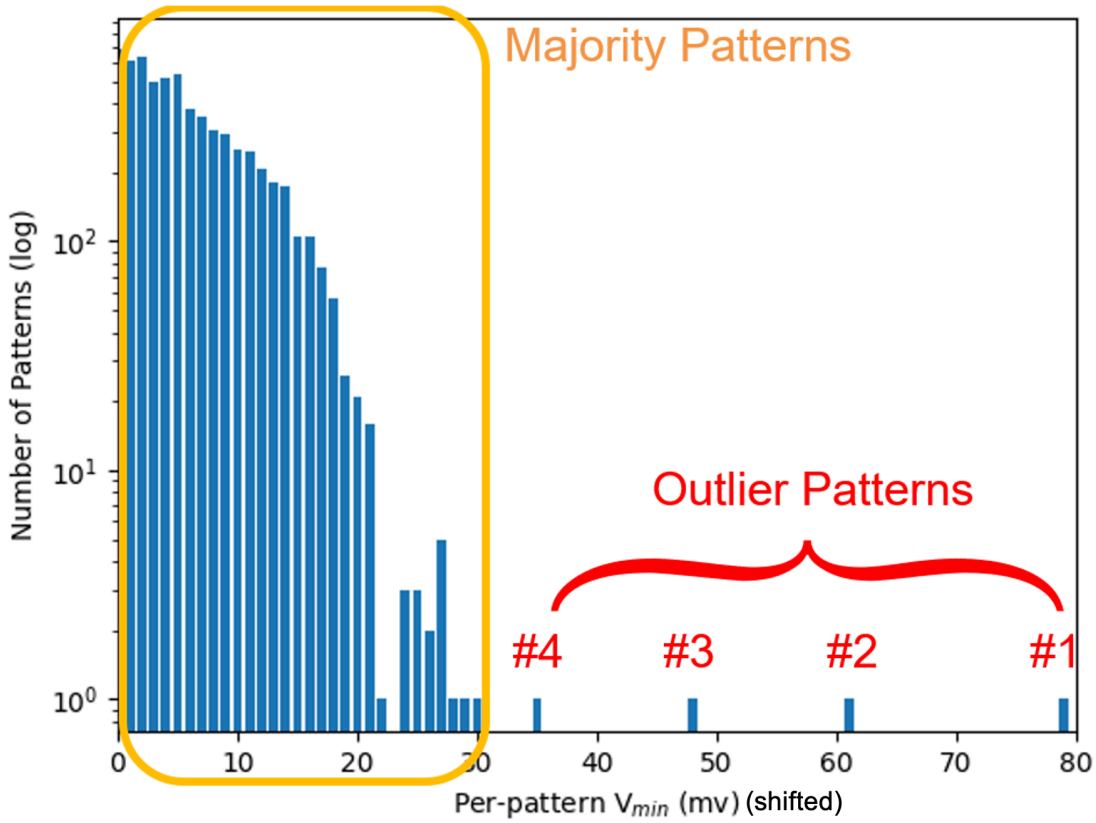


Figure 3.2: Shifted V_{min} distribution of a pattern set

Generally, multiple sample chips are used during test bring-up. The outlier patterns of a pattern set have to be identified as outlier patterns for most sample chips. The outlier patterns, from patterns #1 to #4, in Figure 3.2 have been verified to be outlier patterns for the other seven sample chips as well.

We focus on five debugging attempts for the following subsections. These attempts are based on either power metrics or timing paths. First, we apply power-aware ATPG to generate patterns with different power constraints. Second, we attempt to correlate global

doi:10.6342/NTU202502538

power metrics with per-pattern V_{min}. Third, we try to correlate local power metrics with per-pattern V_{min}. Fourth, we consider long paths when correlating local power metrics with per-pattern V_{min}. Lastly, we will debug high V_{min} patterns with unconstrained paths.

3.2 Debugging Attempt No.1: Power-aware ATPG

To debug whether high power is the root cause for high V_{min}, we generate power-aware ATPG test pattern sets with different power constraints. In commercial power-aware ATPG, we can set various power constraints to restrict the tool, including the WSA threshold and state element transition threshold for both capture cycles and shift cycles. Since the V_{min} issue caused by power supply noise mentioned in Section 2.4 is more serious during capture cycles, we generate seven pattern sets for a core with various WSA thresholds for capture cycles with power-aware ATPG. The experimental results are shown in Table 3.1. PS is the pattern set without power-aware ATPG. $PS_{10} - PS_{15}$ are pattern sets generated by power-aware ATPG with WSA thresholds for capture cycles. In Table 3.1, capture WSA_{th} is the value of the WSA threshold for capture cycles.

From Table 3.1, we observe that pattern-set V_{min} does not decrease with WSA_{th}. However, we find that there are some abnormally high pattern-set V_{min}. PS and PS_{13} have the two highest pattern-set V_{min} compared to the others. We can conclude that stricter

doi:10.6342/NTU202502538

power constraints do not necessarily lead to lower V_{min}. Although other pattern sets have lower V_{min} than PS, all of them still have outlier patterns, which means that we cannot necessarily solve this high V_{min} problem by low-power ATPG. Furthermore, pattern count increases as WSA_{th} decreases. PS_{10} has a 35% higher pattern count compared to PS. In the rest of this paper, we will focus on PS and PS_{13} pattern sets because they have the highest V_{min}.

Table 3.1: V_{min} of seven pattern sets with different capture WSA_{th}

Pattern Set	**Capture** WSA_{th}	**Pattern-set** V_{min} **(Shifted)**	**Outlier Pattern Exist?**
PS	None	79	Yes
PS_{15}	15%	46	Yes
PS_{14}	14%	49	Yes
PS_{13}	13%	63	Yes
PS_{12}	12%	47	Yes
PS_{11}	11%	50	Yes
PS_{10}	10%	48	Yes

We also try to generate low-power pattern sets with other power constraints, including different WSA_{th} for shift cycles, different state element transition thresholds for both capture and shift cycles. All of them come to the same conclusion that no significant correlation between pattern-set V_{min} and power constraints.

doi:10.6342/NTU202502538

3.3 Debugging Attempt No.2: Global Dynamic Power

To debug if excessive whole chip power is the root cause for per-pattern high V_{min}, we attempt to correlate global power metrics to per-pattern V_{min}. We select PS, one of the pattern sets with the highest pattern-set V_{min}, to carry out the debugging. The V_{min} distribution of PS is shown in Figure 3.2.

To determine if high per-pattern V_{min} is caused by excessive power consumption, we calculate power metrics for each pattern with commercial tools and correlate them to per-pattern V_{min}. There are six power metrics, regarded as important features to estimate dynamic power consumption. In this experiment, we calculate them for the whole core to obtain global dynamic power consumption.

(a) Scan cell transitions during loading patterns

(b) Scan cell transitions during unloading patterns

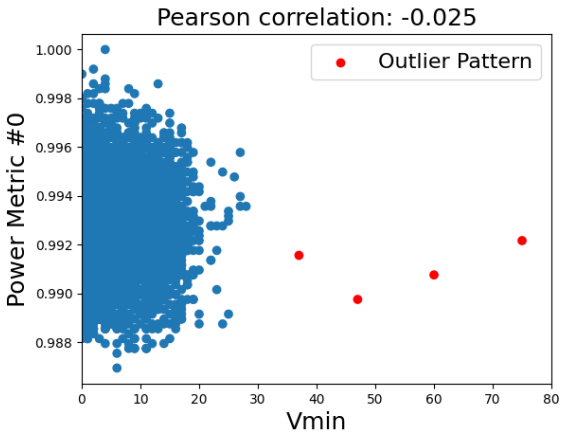
(c) Average of WSA per capture cycle

(d) WSA in the peak capture cycle

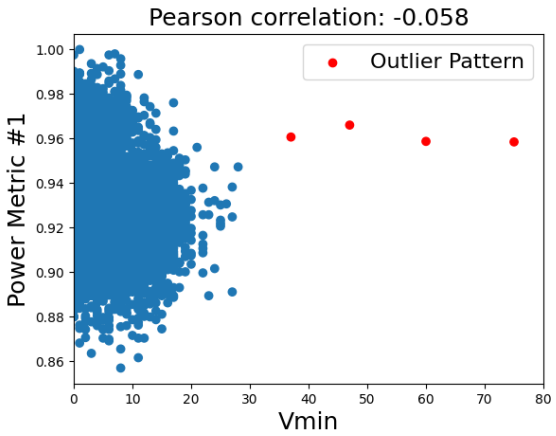
(e) Average of state element transitions per capture cycle

(f) State element transitions in the peak capture cycle

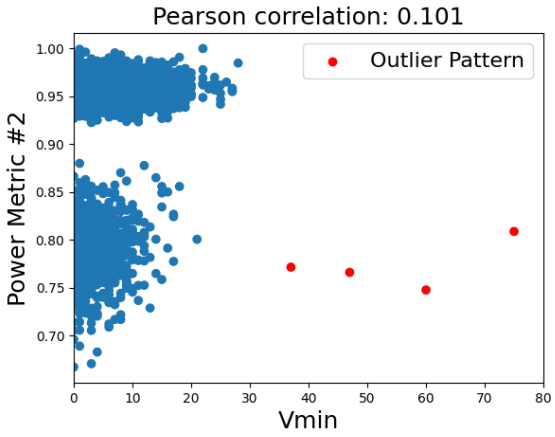
doi:10.6342/NTU202502538



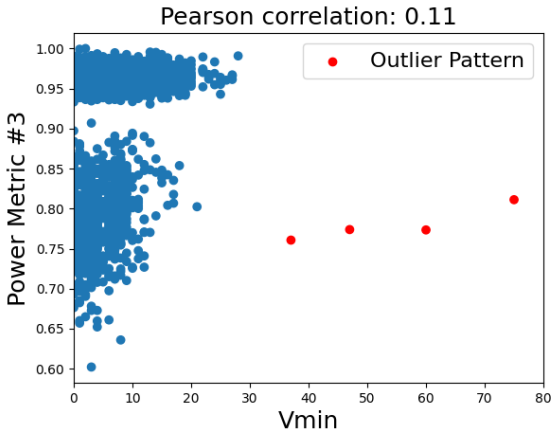
(a) Scan cell transition during loading



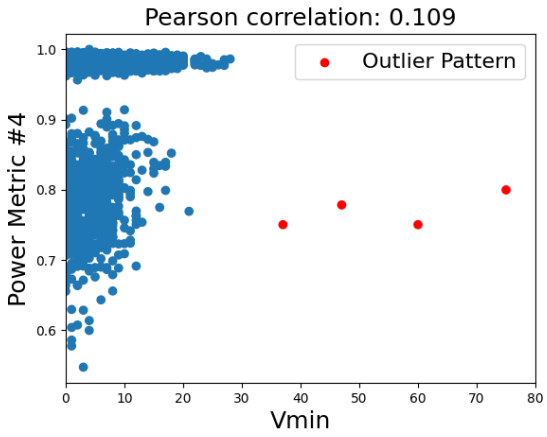
(b) Scan cell transition during unloading



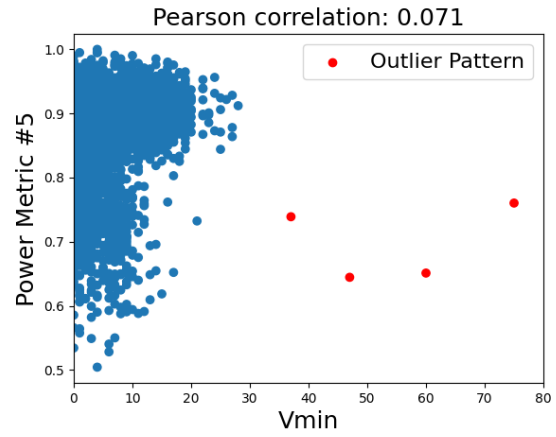
(c) Avg. WSA



(d) Peak WSA



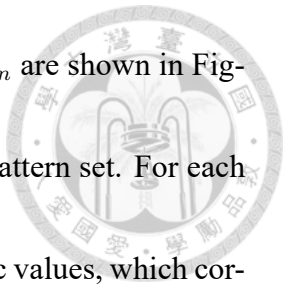
(e)Avg. state element transition



(f)Peak state element transition

Figure 3.3: Correlation between global dynamic power metrics and per-pattern V_{min} (shifted)

doi:10.6342/NTU202502538



The correlations between each power metric and per-pattern V_{min} are shown in Figure 3.3. Each dot in the plots represents an individual pattern in the pattern set. For each pattern, there are shifted per-pattern V_{min} and normalized power metric values, which correspond to the x-axis and y-axis, respectively. On the top of these figures, the Pearson correlations are also shown. We performed similar experiments on the other seven chips and observed similar results. From the figures, we can see that there is hardly any correlation (-0.058 to 0.110) between any of the power metrics and per-pattern V_{min}. Furthermore, outlier patterns do not demonstrate high values for power metrics. Additionally, multivariate correlation is also performed. However, we still cannot find any correlation in our case. These experiments indicate that global dynamic power does not play a significant role in the V_{min} problem in this case.

3.4 Debugging Attempt No.3: Local Dynamic Power

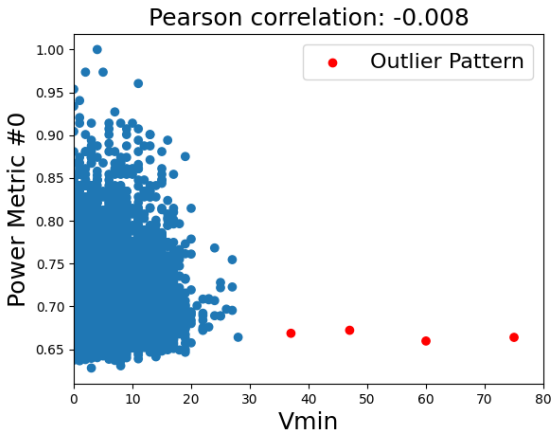
Some local regions in the core may have extremely high power consumption caused by local IR drop and power droop, which would not be considered by global power metrics. We split the core into grids [13, 14], calculate power metrics for each grid, and record the one with the highest value of power metrics for each pattern. Then, we draw similar plots like Figure 3.3 in Figure 3.4 and calculate the Pearson correlation. The results are

doi:10.6342/NTU202502538

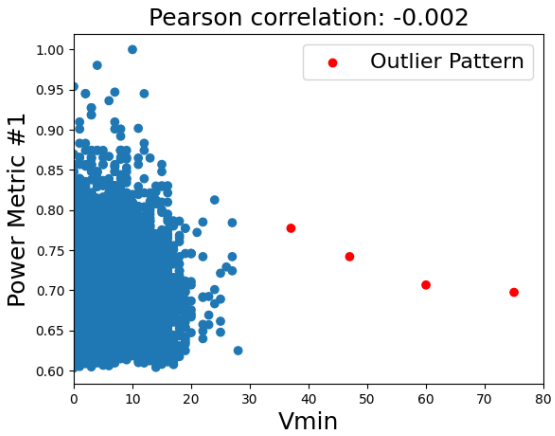
very similar to Figure 3.3, showing hardly any correlations (-0.008 to 0.094) between local power metrics and per-pattern V_{min}. Besides, no extremely high values of the power metrics are found for outlier patterns.

Based on the experimental results, both global and local dynamic power metrics have no impact on per-pattern V_{min}. It may indicate that local or global power is not the root cause for the high per-pattern V_{min} in this case.

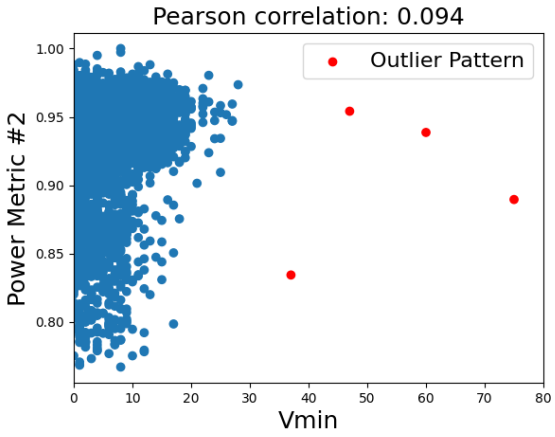
doi:10.6342/NTU202502538



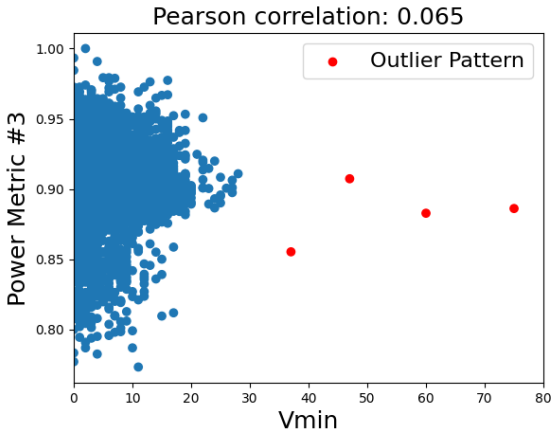
(a) Scan cell transition during loading



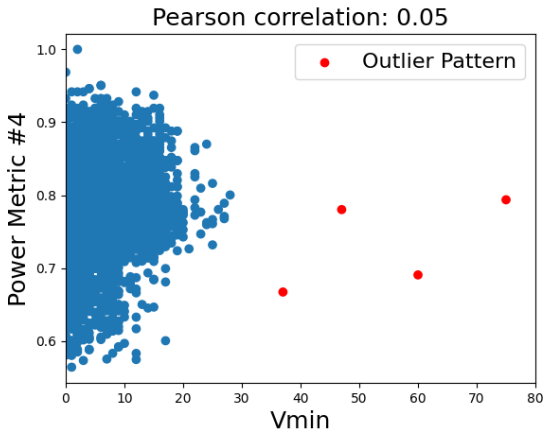
(b) Scan cell transition during unloading



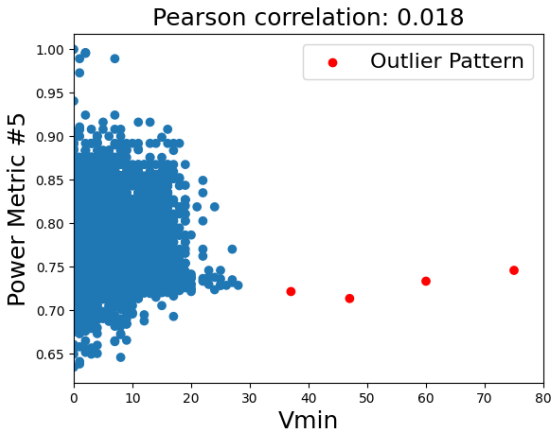
(c) Avg. WSA



(d) Peak WSA



(e)Avg. state element transition



(f)Peak state element transition

Figure 3.4: Correlation between local dynamic power metrics and per-pattern V_{min} (shifted)

doi:10.6342/NTU202502538

3.5 Debugging Attempt No.4: Dynamic Power around the Long Path

To debug whether long paths with high surrounding power are the root cause for high V_{min}, we calculate local power metrics around the longest failing paths. We choose the longest path from the diagnosis report of the outlier pattern #1. We calculate power metrics around the longest path for each pattern. If the calculated power metric of the #1 outlier pattern is the highest among all patterns, it is possible that *dynamic power around long paths* can be the root cause.

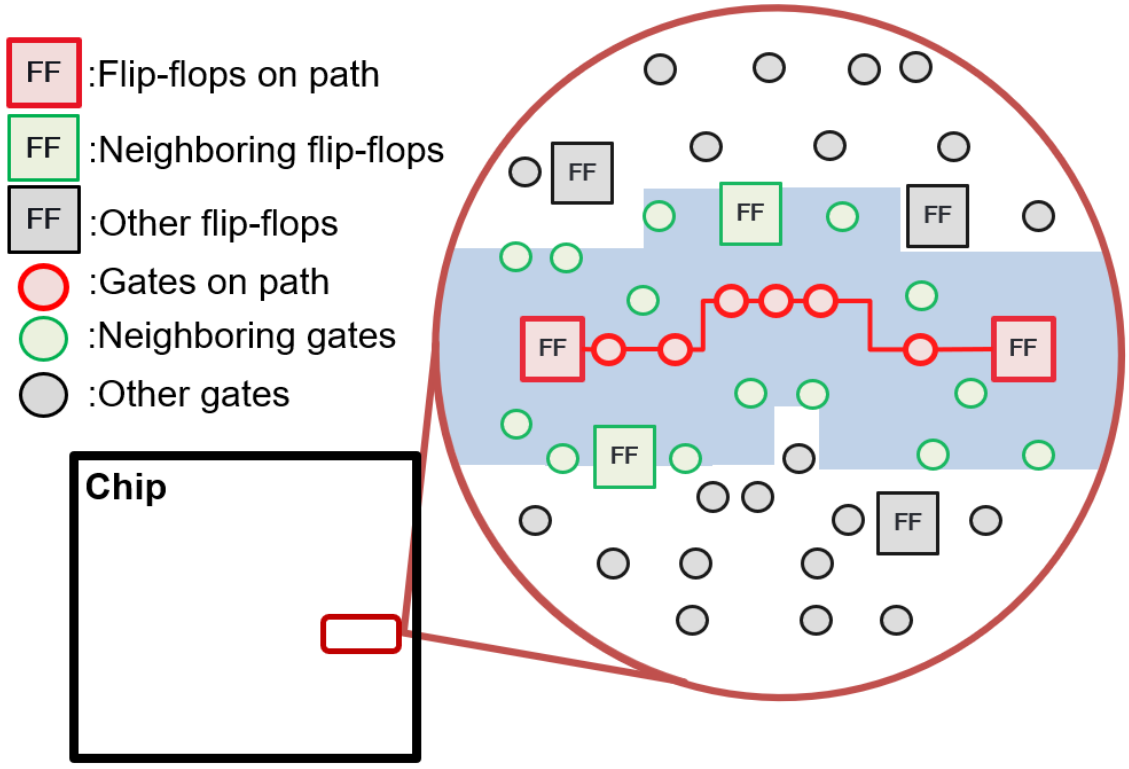
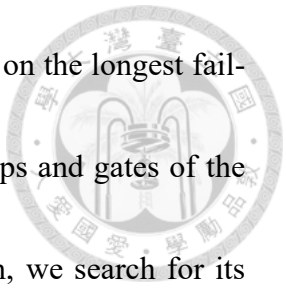


Figure 3.5: Power metrics around the longest path

Figure 3.5 illustrates how we consider the local power metrics around the longest

doi:10.6342/NTU202502538



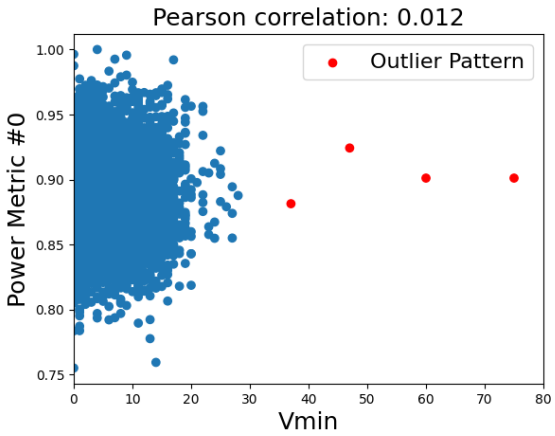
failing path. Red squares and circles indicate the flip-flops and gates on the longest failing path. Green squares and circles indicate the neighboring flip-flops and gates of the longest failing path. For each gate and flip-flop on the longest path, we search for its physically neighboring flip-flops and gates within a certain distance. The blue region in Figure 3.5 encloses all neighboring flip-flops and gates within a certain distance. In our experiment, we consider the neighboring flip-flops and gates within one micron and those on the longest failing path. Then, we calculate their power metrics for each pattern and draw plots like Figure 3.3 in Figure 3.6 and calculate the Pearson correlation. Still, we can barely find any correlation (-0.008 to 0.090) or extremely high values of power metrics for outlier patterns in this case.

3.6 Debugging Attempt No.5: Unconstrained Paths

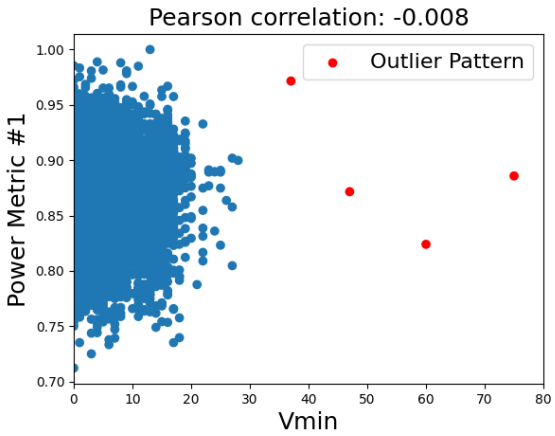
To debug if unconstrained paths are the root cause for high V_{min}, we check if any unconstrained path delay fault is detected by outlier patterns. During testing, unconstrained path delay faults should not be detected because they will not be activated in functional mode. If a pattern detects an unconstrained path delay fault, it may lead to abnormally high V_{min} since the timing of the unconstrained path would not be fixed before sign-off.

First of all, we want to check if all outlier patterns have at least one unconstrained

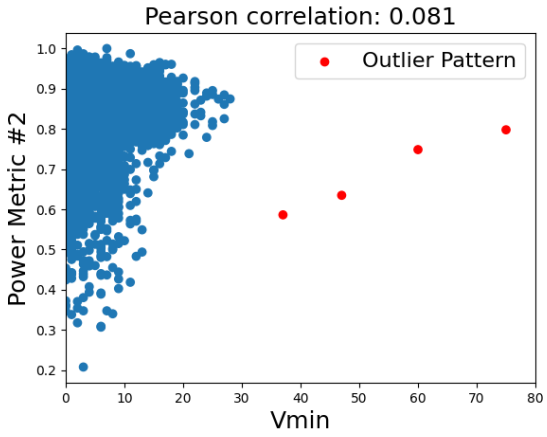
doi:10.6342/NTU202502538



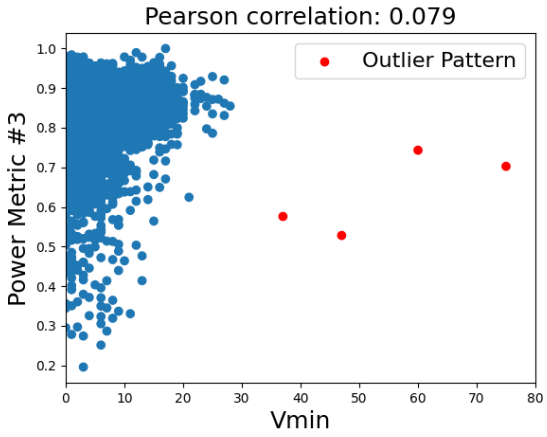
(a) Scan cell transition during loading



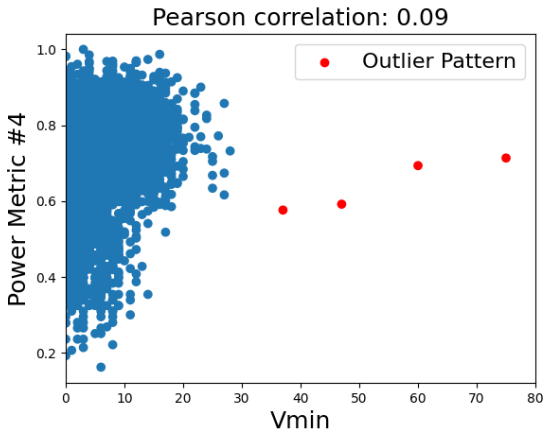
(b) Scan cell transition during unloading



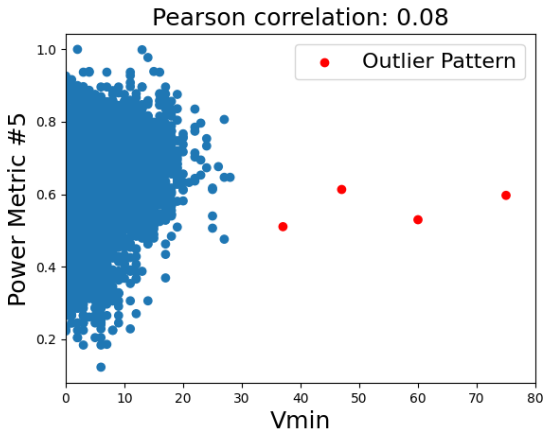
(c) Avg. WSA



(d) Peak WSA



(e)Avg. state element transition



(f)Peak state element transition

Figure 3.6: Correlation between dynamic power metrics around the longest path and per-pattern V_{min} (shifted)

doi:10.6342/NTU202502538

path in their diagnosis report. If all diagnosis reports of outlier patterns contain at least one unconstrained path, high V_{min} is likely caused by these unconstrained paths. To start with, we diagnose the failures of #1 outlier pattern to report the suspected failing paths. Then, we extract all the suspected failing paths in the diagnosis report of #1 outlier pattern. The diagnosis report contains 21 paths. Using a commercial STA tool, we find that one of them is an unconstrained path. We repeat this process for the other outlier patterns (#2-#4) in PS and notice that each of them has at least one unconstrained path in the diagnosis reports. Besides PS, we select PS_{13} for debugging. In PS_{13}, ten patterns are identified as outliers, and each of them had at least one unconstrained path reported.

Second, we want to confirm that the reported unconstrained paths are indeed activated by the outlier patterns. To accomplish this, we leverage fault simulation with the *path delay fault (PDF)* model. Path delay faults representing the suspect unconstrained paths are created, and then simulated for the outlier patterns. Ideally, no pattern should detect any unconstrained path delay fault, as they should have been masked during ATPG. However, in this case, each outlier pattern is found to detect at least one unconstrained path delay fault.

Lastly, we apply one-hot patterns to confirm that the flip-flops capturing the failure align with the unconstrained paths. The examined design uses an XOR compressor, which

doi:10.6342/NTU202502538

leaves potential uncertainty that the flip-flop identified by diagnosis is in fact the culprit. One-hot pattern expansion is a commonly used targeted debug method for identifying flip-flops capturing failing data for a given pattern. These patterns are copies of the original failing pattern that use the same stimulus, but leverage the selective compactor scheme of [22] to observe a single scan chain through the compactor at a time. When all copies are applied, the failing one-hot pattern points to the scan chain containing the failing flip-flop. We expand the outlier patterns of PS and PS_{13} into one-hot patterns, collect fail logs, and apply diagnosis. The results confirm that the culprit failing flip-flops are consistent with the identified unconstrained paths, further supporting that these paths are the root cause of outlier patterns with abnormally high V_{min}.

Based on our debugging results, we believe the root cause in our case is unconstrained paths that were not considered when generating TDF patterns. Capturing the responses of unconstrained paths leads to a pattern to have extremely high per-pattern V_{min}. We should exclude the outlier patterns caused by unconstrained paths to determine if the pattern set contains other outlier patterns due to different root causes. Prevention methods are needed to prevent patterns from detecting faults on unconstrained paths.

doi:10.6342/NTU202502538

3.7 Runtime of Debugging Attempts

Table 3.2 shows the runtime for each task used by debugging attempts. All recorded runtime was measured using only one processor. Usually, people can require more processors to speed up the tasks. Pattern generation is the process for ATPG to generate PS in Table 3.1, which takes about twenty-eight hours. For PS_{10} - PS_{15}, the pattern generation would take more time since there are power constraints during generation process. Analysis of global dynamic power is the whole process of Section 3.3. It includes the runtime of reporting six power metrics for each pattern in PS, which takes about 5 hours. Analysis of local dynamic power is the whole process of Section 3.4. It includes the runtime of partitioning a core into a hundred grids and reporting the six power metrics for each grids and each pattern in PS, which takes about 7 days. Since we have a hundred grids, the runtime is at least a hundred times longer than that of the global dynamic power analysis. Analysis of dynamic power around the long path is the whole process of Section 3.5. It includes the runtime of searching the flip-flops and gates around the path and reporting

Table 3.2: Runtime of debugging attempts

Task	**Runtime**
Pattern Generation	$\approx$ 28 hours
Analysis of Global Dynamic Power	$\approx$ 5 hours
Analysis of Local Dynamic Power	$\approx$ 7 days
Analysis of Dynamic Power around the Long Path	$\approx$ 7 days

doi:10.6342/NTU202502538

the six power metrics of the enclosed flip-flops and gates for each pattern in PS, which takes about 7 days. Since we have to consider the neighboring flip-flops and gates, the runtime much more longer than that of the global dynamic power analysis.

doi:10.6342/NTU202502538

Chapter 4

Proposed Prevention Methods

In this section, we propose two methods to prevent unconstrained paths from causing high V_{min} problems. Unlike traditional solutions, both methods do not need power-aware ATPG. First, we propose a *pre-silicon method*, which can be used before we bring the pattern set to ATE. It helps us to debug the patterns that could potentially lead to high V_{min} before we apply them on ATE. However, it may mask more flip-flops, which results in lower fault coverage. Second, we propose a *post-silicon method*, which can be used after we have fail logs from ATE. It reduces the iterative debug efforts (like Figure 1.1) when the high V_{min} problem is caused by unconstrained paths. Additionally, it masks fewer flip-flops, allowing fault coverage to remain nearly the same.

The pre-silicon and the post-silicon methods are independent. We could either use the pre-silicon method before testing chips on ATE, or use the post-silicon method after testing chips on ATE. However, it is possible that the pre-silicon method leaves some

doi:10.6342/NTU202502538

unconstrained path unconsidered due to incomplete Synopsys Design Constraint (SDC) files. If it happens, we can also choose to use both methods, such that the post-silicon method can be used to prevent any unconstrained paths missed by the pre-silicon method.

4.1 Pre-silicon Method

A pre-silicon method is proposed to prevent the responses of unconstrained paths from being captured by our generated pattern sets before ATE data is available. The goal of the pre-silicon method is to debug the patterns that detect any unconstrained path delay faults and mask their corresponding capture flip-flops before the pattern sets are applied on ATE.

The overall method is shown in Figure 4.1. At the start of our method, we would have a list of unconstrained paths that can be extracted from SDC files and a pattern set from DFT engineers. We apply the path delay fault model and add all the unconstrained paths to our fault dictionary. Then, we fault simulate each pattern in the pattern set to see if any of them activate some unconstrained path.

If a pattern detects an unconstrained path delay fault, we add the unconstrained path to the filtered unconstrained path list. We then mask capture flip-flops based on the filtered unconstrained path list, which would avoid the path being observed and cause high V_{min}

doi:10.6342/NTU202502538

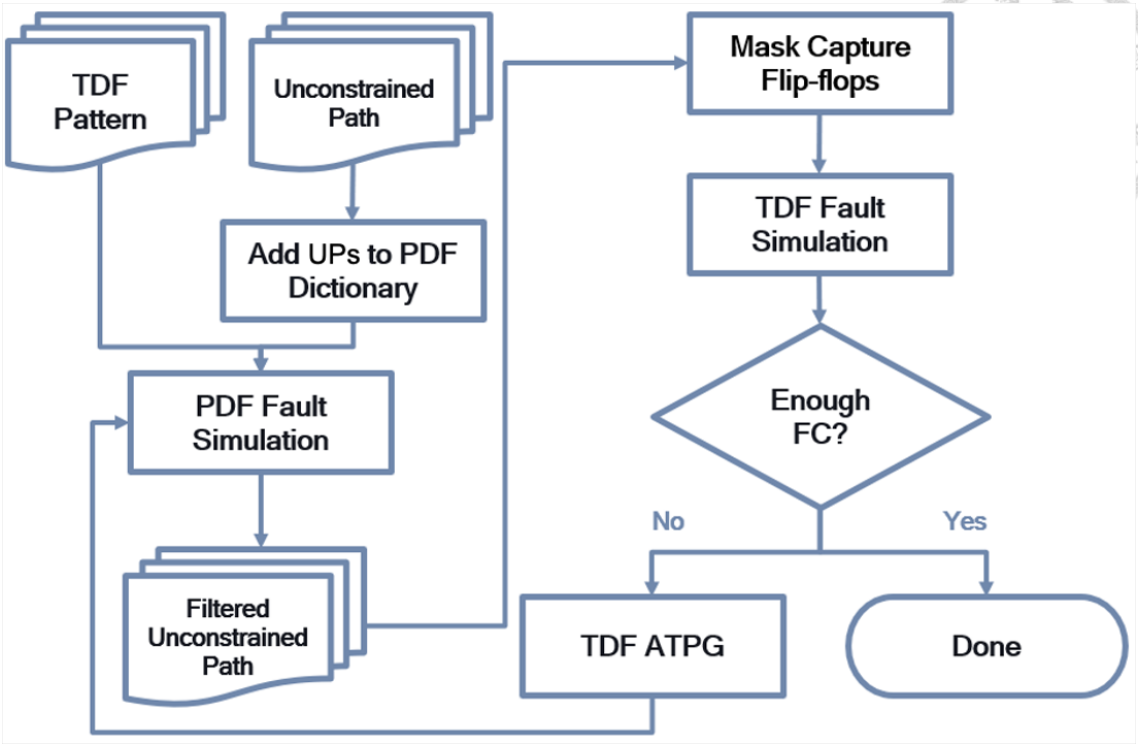


Figure 4.1: Pre-silicon method

failures. After masking, TDF coverage would be degraded. To recover the TDF coverage, we first fault simulate all patterns for TDF with capture flip-flops masked and record the fault coverage. If the fault coverage does not meet the target, we generate new TDF patterns with ATPG to recover the fault coverage. For those newly generated patterns, we apply the PDF fault simulation again to make sure these patterns don't capture responses from any unconstrained paths. If a new pattern activates any unconstrained path, we must mask the capture flip-flop again. We continue this loop until the fault coverage is high enough and no responses of unconstrained paths are captured.

Compared to the traditional method in Figure 1.1, the pre-silicon method can improve V_{min} with less pattern count inflation. Since our pre-silicon method does not replace all

doi:10.6342/NTU202502538

patterns with high V_{min}, we can have less pattern count inflation. However, it still causes some unnecessary pattern count inflation since not all unconstrained paths lead to abnormally high V_{min}. This issue can be further improved by the post-silicon method.

4.2 Post-silicon Method

To reduce the potential pattern count inflation of the pre-silicon method, we propose a post-silicon method. The post-silicon method requires the pattern sets with high V_{min} to be applied, and fail logs collected from ATE.

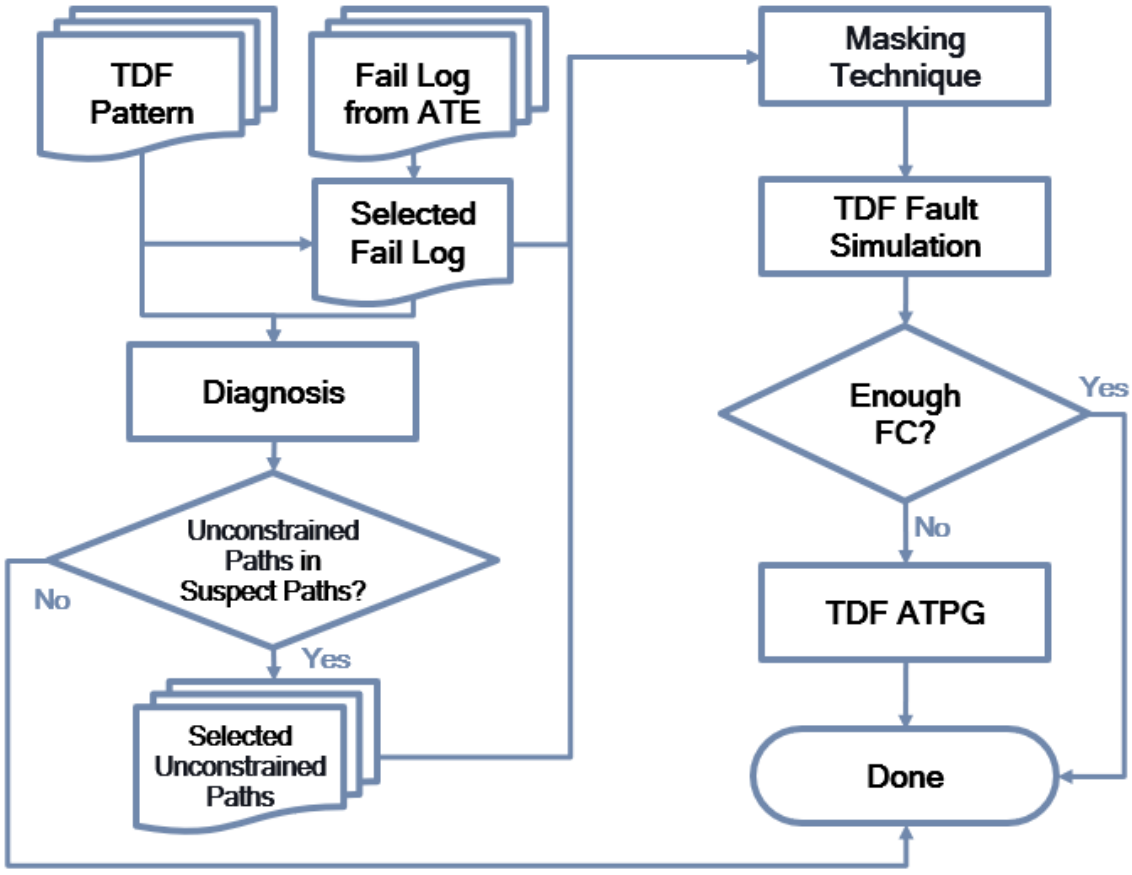


Figure 4.2: Post-silicon method

doi:10.6342/NTU202502538

The post-silicon method is shown in Figure 4.2. Given the fail logs from ATE, we first compute the per-pattern V_{min} of a pattern set. Then, we decide which patterns are outlier patterns by checking per-pattern V_{min} across all the tested sample chips. The detailed process to decide outlier patterns across all sample chips for a pattern set is introduced in Section 3.1. After determining outlier patterns, we identify the outlier pattern that has the lowest per-pattern V_{min} and pick out the fail log at that voltage. With the fail log and the pattern set, we apply the diagnosis tool to report the suspects and the potential failing paths. We inspect the potential failing paths and check if they are unconstrained paths or not. If a potential failing path is an unconstrained path, we add it to the diagnosed unconstrained path list. There are two ways to perform the masking in the next step.

1. Mask capture flip-flops for all patterns

2. Mask failing cycles for outlier pattern

The first masking technique masks the capture flip-flops of the diagnosed unconstrained path list for all patterns. This masking technique would affect all patterns (outlier as well as non-outlier patterns) and thus degrade fault coverages. The second masking technique masks the failing cycles, also called failing slices, for outlier patterns only. It would not affect fault coverage of non-outlier patterns, resulting in very little decrease in fault coverage. The first technique targets diagnosed unconstrained paths but results in a

doi:10.6342/NTU202502538

higher reduction in fault coverage, whereas the second technique disregards the diagnosed results and achieves a smaller reduction in fault coverage.

Table 4.1 presents an example of two masking techniques. Suppose there are five patterns in a pattern sets and three flip-flops we care, which are in first and second columns. The third column shows the responses for each flip-flop and each pattern. The fourth column indicates which whether a flip-flop pass(P) or fail(F) the pattern. The fifth and sixth columns presents the responses after we apply the first and second masking techniques. In the table, H, L, and X represent high, low ,and unknown respectively. For pattern E, the flip-flop #1 and flip-flop #3 both fail the test. As a result, the output of the flip-flops would be masked (set to X) by either the first or second masking techniques. However, with masking technique #1, we can only mask the flip-flops for all patterns. Thus, the responses of the flip-flop #1 and flip-flop #3 are all set to unknown for all five patterns. As for masking technique #2, we only ignore the failing cycles. Therefore, the flip-flops would be masked only when the flip-flops fail the pattern.

To maintain the fault coverage after masking, we fault simulate the patterns and check the TDF fault coverage. If the fault coverage is not enough, top-off ATPG is performed to generate additional patterns to make up the coverage. When the fault coverage is high enough, we finish our method. When generating additional patterns, the capture flip-flops

doi:10.6342/NTU202502538

Table 4.1: Comparison of two masking techniques

Pattern ID	Flip-Flop ID	Original String	Pass or Fail	Masking Tech. 1	Masking Tech. 2
Pattern A	1	H	F	X	X
	2	L	P	L	L
	3	H	P	X	H
Pattern B	1	L	P	X	L
	2	L	P	L	L
	3	H	P	X	H
Pattern C	1	L	P	X	L
	2	H	P	H	H
	3	L	F	X	X
Pattern D	1	L	P	X	L
	2	H	P	H	H
	3	H	P	X	H
Pattern E	1	H	F	X	X
	2	H	P	H	H
	3	L	F	X	X

H: High **L**: Low **X**: Unknown **P**: Pass **F**: Fail

should be masked as well to prevent the new patterns from activating the same unconstrained paths.

After the post-silicon method, the debugged patterns should have no outlier patterns remaining in the pattern set if all outlier patterns are caused by unconstrained paths. Nonetheless, there is a very small chance that the newly generated patterns activate other unconstrained paths that are not considered when we generate additional patterns. Therefore, we test the whole pattern set (including additionally generated patterns) on ATE again in the test bring-up to ensure that pattern-set V_{min} improves as expected.

doi:10.6342/NTU202502538

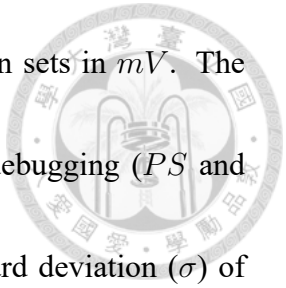
Chapter 5

Prevention Experimental Results

In this section, we demonstrate the experimental results of our pre-silicon and post-silicon prevention methods. The prevention experiments are performed on the same core as Chapter 3. We choose PS and PS_{13} to conduct the prevention experiments. We compare the pattern-set V_{min} and pattern count inflation with and without these two methods. The pattern sets are applied to twelve chips to demonstrate the V_{min} improvement.

Table 5.1 and Table 5.2 show the results of the pre-silicon method and the post-silicon method, respectively. PS and PS_{13} are the pattern sets before debugging. PS' and PS'_{13} are the pattern sets after debugging with our pre-silicon method. PS'' and PS''_{13} are the pattern sets after debugging with our post-silicon method and the first masking technique. PS''' and PS'''_{13} are the pattern sets after debugging with our post-silicon method and the second masking technique. Pattern count inflation indicates the increase in pattern count compared to PS or PS_{13} to maintain the same fault coverage. The column of *pattern-set*

doi:10.6342/NTU202502538



V_{min} *Improvement* shows the V_{min} improvement of debugged pattern sets in mV. The numbers are compared with the corresponding pattern sets before debugging (PS and PS_{13}). In the final two columns, we show the mean (μ) and standard deviation (σ) of V_{min} improvement on these twelve chips.

5.1 Pre-silicon Method

The experimental results of the pre-silicon method are shown in Table 5.1. Compared to PS and PS_{13}, PS' and PS'_{13} improve pattern-set V_{min} by 30.17mV and 39.33mV. The pattern count inflation is 0.33% and 0.50% for PS' and PS'_{13}. The pre-silicon method could improve V_{min} for PS and PS_{13} with low pattern count inflation.

Table 5.1: Pre-silicon method results

Pattern Set	**Pattern Count Inflation (%)**	**Pattern-set V_{min} Improvement (mV)**	
		μ	σ
PS'	0.33	30.17	12.25
PS'_{13}	0.50	39.33	8.02

We retest the same twelve chips on ATE with debugged pattern sets, PS' and PS'_{13}. For PS, there are four outlier patterns across most of the sample chips. For PS_{13}, there are ten outlier patterns across most of the sample chips. The fail logs of PS' and PS'_{13} show no outlier patterns, demonstrating that we successfully eliminated all the outlier patterns that capture the responses of unconstrained paths. After we remove outlier patterns, there

doi:10.6342/NTU202502538

could still be room for V_{min} improvements. If switching activity is a contributing factor, combining the proposed method with power-aware ATPG can further reduce V_{min} more efficiently than the commonly-used method of Figure 1.1.

5.2 Post-silicon Method

The experimental results of the post-silicon method are shown in Table 5.2. For both masking techniques, the post-silicon method achieves similar V_{min} improvements as the pre-silicon method. We describe the results of two masking methods as follows.

Table 5.2: Post-silicon method results

Pattern Set	**Pattern Count Inflation (%)**	**Pattern-set V_{min} Improvement (mV)**	
		μ	σ
PS''	0.04	28.83	11.74
PS'''	0	29.17	11.58
PS''_{13}	0.24	35.50	5.33
PS'''_{13}	0	35.17	5.42

The first masking technique improves V_{min} for PS and PS_{13} by 28.83mV and 35.50mV. The pattern count inflation is 0.04% and 0.24% for PS'' and PS''_{13}. The pattern count inflation for PS'' and PS''_{13} is lower than that of PS' and PS'_{13} because we only mask the unconstrained paths activated by outlier patterns. We retest the same twelve chips on ATE with debugged pattern sets, PS'' and PS''_{13}. We have the same results as PS' and PS'_{13}, that all the outlier patterns capturing the responses of unconstrained paths are eliminated.

doi:10.6342/NTU202502538

The second masking technique improves V_{min} for PS and PS_{13} by 29.17mV and 35.17mV respectively. After masking failing cycles, the fault coverage still meets the target. Therefore, no additional pattern generation is needed for $PS^{'''}$ and $PS_{13}^{'''}$, and the pattern count inflation is 0% for both pattern sets. This masking technique improves V_{min} by masking all failing cycles regardless of the root cause. We retest the same twelve chips on ATE with debugged pattern sets, $PS^{'''}$ and $PS_{13}^{'''}$. They could achieve similar V_{min} improvements to $PS^{''}$ and $PS_{13}^{''}$ with no pattern count inflation. It implies that the second masking technique is also effective in removing the outlier patterns caused by unconstrained paths without additional patterns to recover the coverage loss.

5.3 Methods Comparison

Table 5.3 presents a comparison of the different methods. In our experiments, the pre-silicon method is the best prevention method to improve V_{min}. It achieves the best V_{min} improvements for PS and PS_{13} without ATE. However, the pre-silicon method induces greater pattern count inflation than the post-silicon method. Furthermore, it requires additional effort to extract the unconstrained paths from SDC files and apply PDF fault simulation. The post-silicon method with either the first masking technique or the second masking technique could achieve similar V_{min} improvements with much less pattern

doi:10.6342/NTU202502538

count inflation and efforts. However, the post-silicon method requires the test results from ATE. The first masking technique is specified to remove the outlier patterns caused by unconstrained paths. If some outlier patterns are not caused by unconstrained paths, we can keep the failing information of those outlier patterns, which leads to better diagnostic information. However, it could only be applied on all patterns, which leads to more pattern count inflation than the second masking technique. The second masking technique is not specified to remove the outlier patterns caused by unconstrained paths. It directly masks the failing cycles regardless of the root causes. If there are some outlier patterns not caused by unconstrained paths, we would lose the failing information of the outlier pattern. As a result, it gives less diagnostic information. However, if we don't need to know the root causes of outlier patterns, the second masking technique may be better than the first masking technique since there is no pattern count inflation in our case.

Table 5.3: Comparison of pre-silicon and post-silicon methods

	Need ATE	Need PDF Fault Sim.	V_{min} Improvement	Pattern Count Inflation	Diagnostic Info.
Pre-silicon	No	Yes	1^{st}	3^{rd}	-
Post-silicon w/ Technique #1	Yes	No	2^{nd}	2^{nd}	1^{st}
Post-silicon w/ Technique #2	Yes	No	2^{nd}	1^{st}	2^{nd}

doi:10.6342/NTU202502538

5.4 Runtime of Proposed Prevention Methods

Table 5.4 shows the runtime for each proposed method. The pre-silicon method takes the longest time to complete, which is about eight days. There are two reasons why the pre-silicon takes so much time.

1. Conversion from SDC files to path definition files

2. Path delay fault simulation

Path definition files from SDC files are essential for path delay fault simulation, and path delay fault simulation is required to extract the detected unconstrained paths. These additional steps makes the pre-silicon method the most time-consuming method.

The post-silicon methods with either masking techniques require much less time to complete. However, the runtime is still different between the post-silicon methods with different masking techniques. The first masking technique masks the captured flip-flops of detected unconstrained paths. To know which unconstrained paths are captured, we have to diagnose the fail logs of outlier patterns. With diagnosis, the first masking technique requires more time than the second masking technique, which is about twenty-five hours. The second masking technique masks the failing cycles of outlier patterns. It only required

doi:10.6342/NTU202502538

the fail logs of outlier patterns, which is much more easier than the first masking technique.

As a result, it takes the shortest time, which is about thirteen hours.

Table 5.4: Runtime of proposed method

Method	**Runtime**
Pre-silicon Method	$\approx$ 8 days
Post-silicon Method with the First Masking Technique	$\approx$ 25 hours
Post-silicon Method with the Second Masking Technique	$\approx$ 13 hours

doi:10.6342/NTU202502538

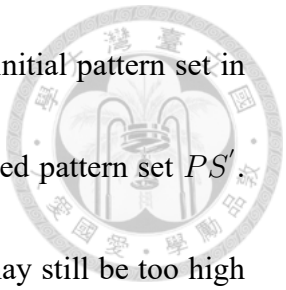
Chapter 6 Discussion

6.1 Power-aware ATPG and Prevention Methods

The improved V_{min} of a pattern set with our prevention methods may still be too high for mass production, which means the problem of over-testing may still occur. If V_{min} is still too high, we have to find another way to further reduce V_{min}. In the experiments of 3.3, we generate several pattern sets with different power constraints. We observe that power constraints cannot effectively reduce V_{min} of a pattern set, which means that power is not directly correlated to V_{min}. However, the reason why power is not correlated to V_{min} may be that there are outlier patterns caused by unconstrained paths in all the pattern sets. By removing all the outlier patterns for all the pattern sets, we find that the average of per-pattern V_{min} is reduced as the power constraints get stricter. It implies that power still has effects on V_{min} of a pattern set if no outlier patterns exist.

Based on our experimental results, we think that combining traditional V_{min} debug method, which apply power-aware ATPG, and our proposed prevention methods helps us

doi:10.6342/NTU202502538



to improve V_{min} more efficiently. For example, we take PS as our initial pattern set in test bring-up. We apply our pre-silicon method and obtain a debugged pattern set PS'. Even though PS' has much lower V_{min} than PS, the V_{min} of PS' may still be too high for production. If V_{min} of PS' is still too high for production, we regenerate a pattern set, PS_{13}, which has stricter power constraints. We apply our pre-silicon method again and obtain a debugged pattern set PS'_{13}. Compared to PS', PS'_{13} has about $12mV$ lower V_{min} in average. As a result, PS'_{13} can achieve lower V_{min} than PS' so that V_{min} over-testing problem would less likely to happen in production.

6.2 Capturing Responses of Unconstrained Paths

In Section 3.6, we mention that unconstrained paths should not be activated. Under normal circumstances, the capture flip-flop of a path would be masked if the path is set as an unconstrained path. However, in our case, we found that some unconstrained paths were activated and their responses were captured. This implies that something went wrong before we generated our pattern sets. There are several possible reasons that could have led to this issue.

The first possible reason is a tool bug. Before we generate patterns, we use a script derived from SDC files, which contains all the unconstrained paths that should be set in

doi:10.6342/NTU202502538

the tool. When the tool reads this script, it is supposed to mask all the capture flip-flops of the unconstrained paths so that these flip-flops are not considered valid capture points during pattern generation. However, since the script content can be very large, the tool might miss some unconstrained paths and fail to mask their capture flip-flops. This could result in outlier patterns that capture the responses of unconstrained paths. However, we don't have any direct evidence that a tool bug leads to the consequences so far.

The second possible reason is human error. Before pattern generation, engineers execute several commands with the tool. Due to the complexity of the tool and the number of commands involved, the interactions between commands may not be fully understood by engineers. As a result, some commands might unintentionally unmask flip-flops that should have remained masked after the script was read. This could also lead to unexpected outlier patterns that activate some unconstrained paths.

Besides the above reasons, there may be other factors contributing to the issue. We have only discussed the two most likely causes. To identify the actual root cause of activated unconstrained paths, we need to break down each step before pattern generation, which requires significant time and effort. What we do know is that some capture flip-flops of unconstrained paths were not masked, even though those paths were correctly set as unconstrained in the tool. Identifying the exact cause is left for future work.

doi:10.6342/NTU202502538

Chapter 7 Conclusion

This thesis presents a debug case study on identifying the root cause of abnormally high V_{min} during test bring-up for an industrial core. The correlation between power metrics and V_{min} has been observed to be very low in our case. We have successfully debugged the root cause of high V_{min} to unmasked unconstrained paths in this case. We propose pre-silicon and post-silicon methods to prevent high V_{min} caused by unconstrained paths. The experimental results show that both pre-silicon and post-silicon methods can effectively prevent test patterns from capturing the responses of unconstrained paths with extremely low pattern count inflation. For the pre-silicon method, we achieve 30.17mV to 39.33mV V_{min} improvements with 0.33% to 0.50% pattern count inflation. For the post-silicon method, we achieve 28.83mV to 35.50mV V_{min} improvements with 0% to 0.24% pattern count inflation. Our methods have been shown to be effective in improving V_{min} for real data if outlier patterns with high V_{min} are caused by unconstrained paths.

doi:10.6342/NTU202502538

References

[1] P. Pant and J. Zelman, “Understanding Power Supply Droop during At-Speed Scan Testing,” in *2009 27th IEEE VLSI Test Symposium (VTS)*, pp. 227–232, 2009.

[2] A. Srivastava and J. Abraham, “Low Capture Power At-Speed Test with Local Hot Spot Analysis to Reduce Over-Test,” in *2022 IEEE International Test Conference (ITC)*, pp. 446–455, 2022.

[3] J. Saxena, K. Butler, V. Jayaram, S. Kundu, N. Arvind, P. Sreeprakash, and M. Hachinger, “A Case Study of IR-drop in Structured At-speed Testing,” in *2003 IEEE International Test Conference (ITC)*, vol. 1, pp. 1098–1104, 2003.

[4] X. Lin, R. Press, J. Rajski, P. Reuter, T. Rinderknecht, B. Swanson, and N. Tamarapalli, “High-frequency, At-speed Scan Testing,” *IEEE Design & Test of Computers*, vol. 20, no. 5, pp. 17–25, 2003.

[5] J. Savir, “Skewed-Load Transition Test: Part I, Calculus,” in *1992 IEEE International Test Conference (ITC)*, pp. 705–, 1992.

doi:10.6342/NTU202502538

[6] S. Patil and J. Savir, "Skewed-Load Transition Test: Part II, Coverage," in *1992 IEEE International Test Conference (ITC)*, pp. 714–, 1992.

[7] J. Savir and S. Patil, "On Broad-side Delay Test," in *1994 IEEE VLSI Test Symposium (VTS)*, pp. 284–290, 1994.

[8] W.-C. Lin, C. Chen, C.-H. Hsieh, J. C.-M. Li, E. J.-W. Fang, and S. S.-Y. Hsueh, "ML-Assisted VminBinning with Multiple Guard Bands for Low Power Consumption," in *2022 IEEE International Test Conference (ITC)*, pp. 213–218, 2022.

[9] S. Borkar, "Designing Reliable Systems from Unreliable Components: the Challenges of Transistor Variability and Degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10–16, 2005.

[10] H. H. Chen, "Analysis of Vmin Variability in Complex Digital Logic via Post-Silicon Profiling," in *2023 International VLSI Symposium on Technology, Systems and Applications (VLSI-TSA/VLSI-DAT)*, pp. 1–4, 2023.

[11] V. Sontakke and J. Dickhoff, "A Survey of Scan-capture Power Reduction Techniques," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 13, no. 6, pp. 6118–6130, 2023.

doi:10.6342/NTU202502538

[12] R. Jajodia, J. Kurien, K. Zhou, T. Raja, P. Manikandan, K. Joshi, P. Singh, V. Srinath, J. Colburn, and S. Sharma, "Applications of Test Techniques for Improving Silicon to Pre-Silicon Timing Correlation," in *2019 IEEE International Test Conference India (ITC India)*, pp. 1–8, 2019.

[13] J.-X. Chen, S.-T. Liu, Y.-T. Wu, M.-T. Wu, J. C.-M. Li, N. Chang, Y.-S. Li, and W.-T. Chuang, "Vector-based Dynamic IR-drop Prediction Using Machine Learning," in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 202–207, 2022.

[14] Z.-J. Liang, Y.-T. Wu, Y.-F. Yang, J. C.-M. Li, N. Chang, A. Kumar, and Y.-S. Li, "High-Speed, Low-Storage Power and Thermal Predictions for ATPG Test Patterns," in *2023 IEEE International Test Conference (ITC)*, pp. 206–215, 2023.

[15] A. D. Singh, "Understanding Vmin Failures for Improved Testing of Timing Marginalities," in *2022 IEEE International Test Conference (ITC)*, pp. 372–381, 2022.

[16] S.-C. Hung, Y.-C. Lu, S. K. Lim, and K. Chakrabarty, "Power Supply Noise-Aware Scan Test Pattern Reshaping for At-Speed Delay Fault Testing of Monolithic 3D ICs," in *2020 IEEE 29th Asian Test Symposium (ATS)*, pp. 1–6, 2020.

doi:10.6342/NTU202502538

[17] P. Girard, “Survey of Low-power Testing of VLSI Circuits,” *IEEE Design & Test of Computers*, vol. 19, no. 3, pp. 82–92, 2002.

[18] J. Li, C.-W. Tseng, and E. McCluskey, “Testing for Resistive Opens and Stuck Opens,” in *2001 IEEE International Test Conference (ITC)*, pp. 1049–1058, 2001.

[19] K. S. Kim, S. Mitra, and P. Ryan, “Delay Defect Characteristics and Testing Strategies,” *IEEE Design & Test of Computers*, vol. 20, no. 5, pp. 8–16, 2003.

[20] D. Goswami, K.-h. Tsai, M. Kassab, T. Kobayashi, J. Rajski, B. Swanson, D. Walters, Y. Sato, T. Asaka, and T. Aikyo, “At-Speed Testing with Timing Exceptions and Constraints-Case Studies,” in *2006 15th Asian Test Symposium (ATS)*, pp. 153–162, 2006.

[21] D. Goswami, K.-H. Tsai, M. Kassab, and J. Rajski, “Test Generation in the Presence of Timing Exceptions and Constraints,” in *2007 44th ACM/IEEE Design Automation Conference (DAC)*, pp. 688–693, 2007.

[22] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, Kun-Han Tsai, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eide, and Jun Qian, “Embedded Deterministic Test for Low Cost Manufacturing Test,” in *2002 IEEE International Test Conference (ITC)*, pp. 301–310, 2002.

doi:10.6342/NTU202502538