HIGH QUALITY TEST GENERATION TARGETING POWER SUPPLY NOISE

A Dissertation

by

TENGTENG ZHANG

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

DOCTOR OF PHILOSOPHY

Chair of Committee,	Duncan M. Walker
Committee Members,	Weiping Shi
	Jyh-Charn Liu
	Rabi Mahapatra
Head of Department,	Dilma Da Silva

December 2015

Major Subject: Computer Engineering

ABSTRACT

Delay test is an essential structural manufacturing test used to determine the maximal frequency at which a chip can run without incurring any functional failures. The central unsolved challenge is achieving high delay correlation with the functional test, which is dominated by power supply noise (PSN). Differences in PSN between functional and structural tests can lead to differences in chip operating frequencies of 30% or more. Pseudo functional test (PFT), based on a multiple-cycle clocking scheme, has better PSN correlation with functional test compared with traditional two-cycle at-speed test. However, PFT is vulnerable to under-testing when applied to delay test. This work aims to generate high quality PFT patterns, achieving high PSN correlation with functional test.

First, a simulation-based don't-care filling algorithm, Bit-Flip, is proposed to improve the PSN for PFT. It relies on randomly flipping a group of bits in the test pattern to explore the search space and find patterns that stress the circuits with the worst-case, but close to functional PSN. Experimental results on un-compacted patterns show Bit-Flip is able to improve PSN as much as 38.7% compared with the best random fill.

Second, techniques are developed to improve the efficiency of Bit-Flip. A set of partial patterns, which sensitize transitions on critical cells, are pre-computed and later used to guide the selection of bits to flip. Combining random and deterministic flipping, we achieve similar PSN control as Bit-Flip but with much less simulation time.

Third, we address the problem of automatic test pattern generation for extracting circuit timing sensitivity to power supply noise during post-silicon validation. A layout-aware path selection algorithm selects long paths to fully span the power delivery network. The selected patterns are intelligently filled to bring the PSN to a desired level. These patterns can be used to understand timing sensitivity in post-silicon validation by repeatedly applying the path delay test while sweeping the PSN experienced by the path from low to high.

Finally, the impacts of compression on power supply noise control are studied. Illinois Scan and embedded deterministic test (EDT) patterns are generated. Then Bit-Flip is extended to incorporate the compression constraints and applied to compressible patterns. The experimental results show that EDT lowers the maximal PSN by 24.15% and Illinois Scan lowers it by 2.77% on un-compacted patterns.

DEDICATION

To my parents:

Without their support, this would not have been possible.

ACKNOWLEDGEMENTS

I would like to thank my advisor and committee chair, Dr. Duncan M. (Hank) Walker, for his advice and support throughout my Ph.D. studies at Texas A&M University. His insights in this particular research area, his technical guidance and spiritual support were invaluable to this work. This dissertation would never have been completed without his advice and encouragement. I owe him lots of gratitude for making my research life enjoyable and rewarding. What I learned from him will benefit my whole life.

I am also grateful to my committee members, Dr. Steve Liu, Dr. Rabinarayan Mahapatra and Dr. Weiping Shi, for their valuable suggestions and personal encouragement.

My gratitude also goes to my lab mates and friends: Punj Pokharel, Yukun Gao, Kun Bian, and Swati Chakraborty.

Also, I want to acknowledge the Semiconductor Research Corporation (SRC) and National Science Foundation (NSF) for their financial support of this research.

Finally, I would like to thank my parents and my girlfriend, Cong Feng, for giving me company during my study. Their love enriched my life. Without their constant support, this would not have been a wonderful journey.

TABLE OF CONTENTS

Page

ABSTRACT ii

DEDICATION iv

ACKNOWLEDGEMENTS v

TABLE OF CONTENTS vi

LIST OF FIGURES ix

LIST OF TABLES xii

CHAPTER I INTRODUCTION AND LITERATURE REVIEW 1

1.1 Scan-Based Test 1
1.2 Delay Test 4
1.3 Power Supply Noise 6
1.4 Power Supply Noise Control 8
1.5 Pseudo Functional Path Delay Test Generation 10
1.6 Organization of the Dissertation 12

CHAPTER II POWER SUPPLY NOISE CONTROL 14

2.1 Introduction 14
2.2 Pseudo-Functional PSN 15
2.3 Overview of Proposed Framework 18
2.3.1 Critical Cell Identification 21
2.3.2 Task Granularity 21
2.3.3 Critical Bit Fill and Bit Relaxation 23
2.3.4 Compacted Pattern Consideration 23
2.4 Experimental Results 24
2.5 Summary 31

CHAPTER III IMPROVED POWER SUPPLY NOISE CONTROL 32

3.1 Introduction 32

3.2 Background and Motivation 32
3.3 Background Pattern 36
3.3.1 Background Pattern Generation 36
3.3.2 Background Pattern Pool 38
3.3.3 Evaluation of Background Patterns 39
3.4 Improved Power Supply Noise Control 41
3.4.1 Two-pass Bit-Flip w/ Single-bit Group 41
3.4.2 BPs-Based Flip 42
3.4.3 Rank of BPs 42
3.4.4 Handling of Conflicts 43
3.4.5 Dynamic Bit Weighting 44
3.4.6 Compression Consideration 45
3.5 Experiment Results 45
3.6 Summary 52

CHAPTER IV PATTERN GENERATION FOR POST-SILICON TIMING VALIDATION 53

4.1 Introduction 53
4.2 Pseudo Functional Test Generation 55
4.3 Layout-Aware Path Selection 56
4.4 Post-ATPG Processing for PSN 58
4.5 PSN Estimation 60
4.6 Experimental Results 66
4.7 Summary 73

CHAPTER V IMPACT OF TEST COMPRESSION ON PSN CONTROL 74

5.1 Introduction 74
5.2 Background 77
5.2.1 Pseudo Functional Test with Dynamic Compaction 77
5.2.2 PSN Control for PFT 78
5.2.3 Test Compression 79
5.3 Pattern Generation with PSN Control 81
5.3.1 Compressible Pattern Generation 81
5.3.2 Compression Aware Supply Noise Control 82
5.4 Experimental Results 86
5.4.1 Results of EDT 89
5.4.2 Results of Illinois Scan 92
5.4.3 Results on Other Circuits 93
5.5 Summary 95

CHAPTER VI SUMMARY AND FUTURE WORK 96

REFERENCES 100

LIST OF FIGURES

Page

Fig. 1 Flip-Flop and scan cell 2

Fig. 2 Launch on shift scheme 5

Fig. 3 Launch on capture scheme 6

Fig. 4 Over-testing and under-testing 8

Fig. 5 Clocking scheme for pseudo functional test 10

Fig. 6 Average WSA falls with multiple cycles 17

Fig. 7 b19 WSA correlation of different cycles 18

Fig. 8 Bit-Flip flow 19

Fig. 9 Effective region 21

Fig. 10 b19: Average ΔEWSA vs. group size 25

Fig. 11 Average ΔEWSA with 95% C.I. 27

Fig. 12 P0 EWSA distribution vs. fill method 28

Fig. 13 Average ΔEWSA vs. fill rate 30

Fig. 14 Original Bit-Flip flow 33

Fig. 15 Bit-Flip PSN control scenarios 33

Fig. 16 Critical cell sharing 34

Fig. 17 Background pattern generation algorithm 37

Fig. 18 Background pattern pool 38

Fig. 19 iBF flow 40

Fig. 20 Classification of conflicts 43

Fig. 21 iBF background patterns 44

Fig. 22 Normalized EWSA at each stage (un-compacted) 50

Fig. 23 Normalized EWSA at each stage (compacted) 51

Fig. 24 Pattern generation and post-silicon validation 55

Fig. 25 An example of 3x3 layout regions 57

Fig. 26 Path selection algorithm 58

Fig. 27 Post-ATPG processing flow 59

Fig. 28 Circuit to study IR-drop 61

Fig. 29 Voltage at Cell9 and Cell10 with different transition time 63

Fig. 30 Minimum voltage seen at Cell10 64

Fig. 31 Alignment of transitions on critical cell and on-path gate 65

Fig. 32 Critical cell identification 66

Fig. 33 Number of paths covering each region in s35932 (top 30%) 68

Fig. 34 Number of paths covering each region in s38417 (top 30%) 68

Fig. 35 Number of paths covering each region in s38417 (top 60%) 69

Fig. 36 s35932: Test pattern with different PSN level 69

Fig. 37 PSN level of each on-path gate 70

Fig. 38 Average number of critical cells per on-path gate 73

Fig. 39 Pattern compaction example [3] 78

Fig. 40 EDT scheme 80

Fig. 41 Illinois Scan: parallel mode and serial mode 80

Fig. 42 PSN control with Illinois Scan constraints 83

Fig. 43 EDT PSN control 85

Fig. 44 Avg. number of bits flipped un-compacted EDT_CR 91

Fig. 45 PSN control without EDT constraints 91

Fig. 46 PSN control with EDT constraints 92

LIST OF TABLES

Page

Table 1 PSN control result of un-compacted patterns....26

Table 2 No. critical cells with transition output....29

Table 3 PSN control result of compacted patterns....30

Table 4 Critical cell sharing summary....35

Table 5 Profiles for each circuit and BPs....46

Table 6 BP profiles....46

Table 7 Bit-Flip on un-compacted patterns....48

Table 8 iBF on un-compacted patterns....48

Table 9 Bit-Flip on compacted patterns....49

Table 10 iBF on compacted patterns....49

Table 11 Regions and critical cells per path....67

Table 12 PSN post processing results w/ filtering....71

Table 13 PSN post processing results w/o filtering....73

Table 14 Pattern statistics....88

Table 15 WSA and WSA improvement....90

Table 16 EWSA result of Illinois Scan....93

Table 17 Statistics for ISCAS circuits....94

Table 18 Results of ISCAS circuits (EDT) 94

Table 19 Results of ISCAS circuits (Illinois Scan) 94

CHAPTER I

INTRODUCTION AND LITERATURE REVIEW

1.1 Scan-Based Test

The difficulty of testing very large scale integrated (VLSI) circuit is the limited controllability and observability after chips are fabricated [1]. This is worsening with the increasing size of modern designs, typically millions of transistors. Various design-for-test (DFT) techniques, such as ad-hoc test point insertion, scan design, and logic build-in self-test, have to be deployed in order to achieve high test quality and low defect level.

Mux-D full scan is one of the most popular techniques and features for its simple structure and small hardware area overhead. Fig. 1 shows the structure of a Mux-D scan cell (SC), which consists of a multiplexer and a D Flip-Flop (DFF). The data input of the DFF has two sources: functional input (D) and scan input (SI). These two signals are selected by the scan enable (SE) signal: when SE is 1, the data presented at SI is selected to DFF at the effective clock edge (CK); when SE is 0, the functional input D is selected. In full scan design, all the DFFs are replaced by SCs and SCs are connected into scan chains, i.e. the output Q is connected to the SI input of next SC.

In the test mode (SE=1), a test pattern can be applied at the SI port of the first SC in each scan chain and then is shifted into the SCs cycle by cycle. After all SCs are initialized, the circuit under test (CUT) switches to functional mode and the response is captured into SCs through DI ports. The contents of the SCs can be shifted out for observation. While the captured data is shifted out, the next test pattern can be shifted

into the SCs. The response is compared against the expected values. If they mismatch, the CUT fails the test. Otherwise, it passes. With SC, the internal chip signals can be controlled and observed. A sequential circuit is transformed into a combinational circuit and the testability is highly improved. Test patterns to test sequential circuits can be developed based on this model. Test patterns generated based on the scan model are *structural tests* as functionality is not considered during pattern generation.

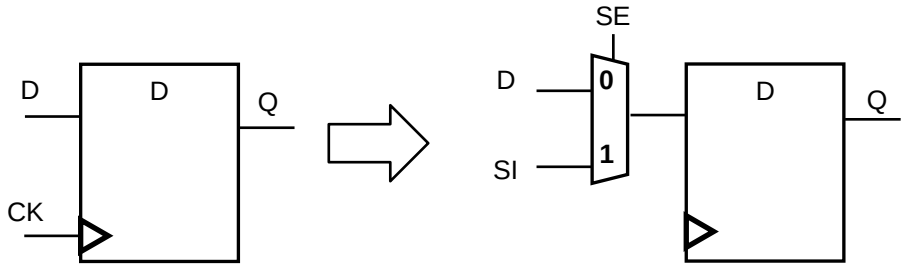


Fig. 1 Flip-Flop and scan cell

As the number of possible test patterns is exponential to the number of scan cells and primary inputs (PI), it is impossible to numerate all the patterns and also not necessary. In practice, physical defects induced by the imperfection of the manufacturing process are modeled as logic faults based on their functional behavior, such as stuck-at-0/stuck-at-1 (SA0/SA1), or bridging faults. So test patterns can be developed based on these faults models. This makes the test generation easier, as the test generation tool does not consider the mechanism behind the behavior, which is complex and infeasible to cover all the possible mechanisms. For example, there are various scenarios that a signal line cannot be changed but is stuck at logic 0 (SA0). But for test generation, the

pattern generation tool only needs to cover the logic fault. Therefore, the complexity of test generation is highly reduced. The quality of the test set is evaluated by fault coverage, which is calculated as the number of faults tested over the total number of faults.

Automatic test pattern generation (ATPG) consists of two steps: sensitizing a fault and propagating the faulty value for observation [2]. For instance, to test a SA0 fault, the test must control the signal line to logic 1, and observe the faulty value 0 at the primary output or a scan cell. Several classic ATPG algorithms, such as D-algorithm, PODEM, and FAN [2], have been developed. They are optimized to reduce the search space and test generation time cost. Test points, including control point and observation point, are usually used to improve the fault coverage. With a good DFT design, stuck-at fault coverage can be as high as 100%.

Test pattern compaction and compression are two popular techniques developed to reduce the test data volume for modern large designs, such as System-on-Chip (SoC), and thus reduce the test application time. Test compaction highly reduces the number of patterns by testing as many faults as possible in one single test pattern. It can be done either during the ATPG stage, i.e. dynamic compaction [3], or post ATPG stage, i.e. static compaction [4]. Dynamic compaction usually achieves a better compaction ratio but with higher CPU time cost. Test compression, such as Illinois Scan [5] and embedded deterministic test (EDT) [6], reduces the length of each test pattern and restores all the test bits using an on-chip decompressor.

1.2 Delay Test

Delay test is one of the essential structural tests and verifies the speed of the chips against the design specification. There are two main delay faults models: transition delay fault model (TDF) and path delay fault model (PDF) [7]. TDF targets individual signal lines and assumes gross delay defects, which is sufficient to cause a signal slow to rise or slow to fall. The benefit of TDF is that the fault set size is linear in the circuit size. Moreover, a stuck-at fault ATPG engine can be reused for TDF as the test of TDF can be viewed as a test of two stuck-at faults. Therefore, existing industry approaches to delay test are built on the TDF model. The drawback of TDF is that it tends to propagate transitions along short paths [8], as the ATPG chooses paths with high observability. As a result, the generated patterns are not effective in covering small delay defects (SDD), which have become increasingly critical with technology scaling to 45nm and below. SDD introduces only a small amount of extra delay to the chip and a single SDD may not be enough to cause failures but only degrade the performance. ATPG engines considering timing information are proposed to propagate transitions along long paths [9] [10]. However, the size of the test set increases and CPU time cost for such ATPG is usually very high [11].

PDF has the advantage of better coverage for SDD as it targets the accumulated delay along a path. False paths [12] and multiple-cycle paths must be excluded from the paths tested as these path might cause false rejection. One of the challenges for PDF is the number of paths increases exponentially with the size of the circuit. Another challenge is that the timing is sensitive to process variation [13] and

power/voltage/temperature conditions. So it is proposed to select a subset of paths for test or timing validation [14] [15]. To select the real speed-limiting paths, a well-established timing model, which should be simple but relatively accurate [16] [17], should be used. One potential solution is to target the *K* longest paths through each gate (KLPG) [18]. In this way, the number of paths is linearly in the size of the circuit.

A delay test pattern consists of two vectors. The first vector initializes the CUT, and the second vector generates a transition and propagates the transition to the observation point. Delay test patterns are applied based on two clocking schemes as shown in Fig. 2 and Fig. 3: launch-on-shift (LOS) [19] and launch-on-capture (LOC) [20]. In LOS, the second vector is one-bit shifted from the first vector. LOS has better fault coverage and pattern generation is usually easier. But it requires the SE signal to switch at functional speed. In LOC test, the second vector is derived from the first vector. It is a two cycle sequential test so ATPG is more complex. The fault coverage is also typically lower.

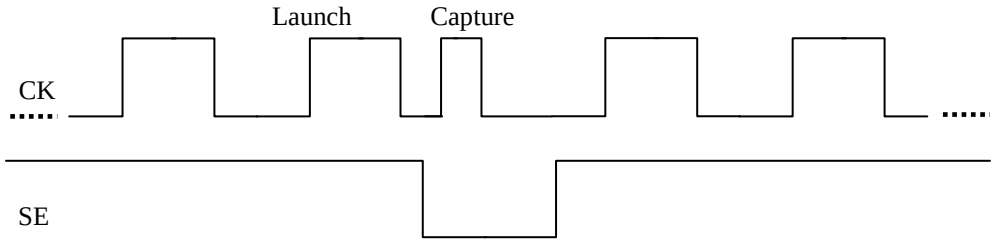


Fig. 2 Launch on shift scheme

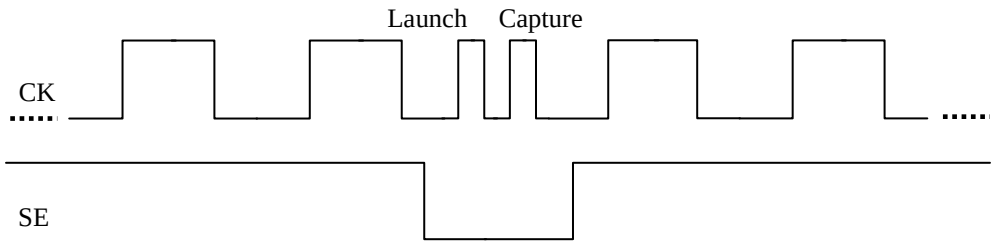


Fig. 3 Launch on capture scheme

1.3 Power Supply Noise

The central unsolved challenge in structural delay test is ***achieving high delay correlation with the functional test***, which is the de-facto standard in determining the chip maximal operating frequency (FMAX). The correlation problem is dominated by power supply noise since it significantly impacts the delay of the selected paths [21] [22] [23] [24]. Differences in PSN between functional and structural tests can lead to differences in chip operating frequencies of 30% or more. Worse, it is becoming very difficult for the test engineer to know the supply noise environment on the chip, due to the use of SoC designs and 3D packaging.

There are two main reasons causing PSN discrepancy between the structural and functional mode. First, scan-based delay test uses slow shift and fast capture clock cycles. In LOC test, the hold cycle between the last shift-in cycle and the first functional cycle must be long enough to allow the scan enable signal to switch. In the meantime, the off-chip inductances settle down to their quiescent currents. When the test is applied, on-chip switching currents must be supplied from on-chip capacitances, causing the supply voltage to drop (*dI/dt* noise). Second, scan patterns are structurally generated for

low cost, randomly filled for fortuitous fault detection and highly compacted for small data volume. Illegal states, which are never visited in functional mode, reside in the test pattern [25] [26]. The combination of these effects results in higher simultaneous switching activities in the test mode. Considering the power grid is not optimized for delivering such a large amount of switching current, excessive supply voltage drop (IR-drop) will occur in test mode.

As shown in Fig. 4, the PSN mismatch between structural and functional mode leads to either over-testing or under-testing. For simplicity, let us assume the delay of a path consists of two parts. One is PSN-induced delay and the other is the normal delay, which counts for manufacturing process variation. In order to pass the delay test, the total delay should be smaller than the test cycle time (CLK). In case (a), the PSN-induced delay is larger in test mode than that in worst-case functional mode. As a result, the total delay exceeds the cycle time and the chip fails the delay test. This is called over-testing. In case (b), the PSN-induced delay is smaller in test mode and the test does not test the worst-case delay of the circuit. So the measured FMAX is too optimistic. This is called under-testing. Over-testing causes yield loss while under-testing may later test to fail (higher test cost) or cause field failure after shipping to the customer. Both of the cases should be avoided in manufacturing test.

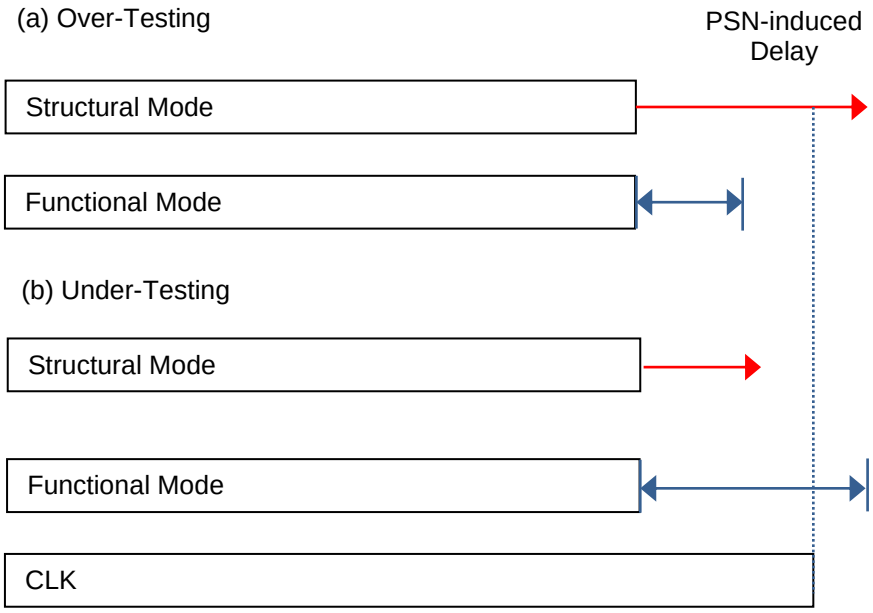


Fig. 4 Over-testing and under-testing

1.4 Power Supply Noise Control

The philosophy of handling PSN in test mode can be divided into three categories: (1) minimize PSN; (2) maximize PSN and (3) maximize the realistic PSN.

The first approach spares no effort to minimize power or PSN to avoid over-testing. Plenty of techniques have been developed, such as scan chain reordering [27], low-power test generation [28] [29] [30], low-power X-filling [31] [32] [33] [34] [35], and power-aware compaction [4] [36]. Here "X" stands for don't-care bits in the test pattern. The main disadvantage of these methods is that they do not change the fact that illegal states may still reside in the test pattern. The illegal states mean circuit states that are never reached in the functional mode. Pseudo functional test [25] [26] is another technique falling into this category but explicitly places logic constraints on ATPG to

avoid illegal states. However, it is hard to enumerate all possible illegal states and also potentially causes under-testing.

The second approach attempts to maximize PSN in order to achieve high SDD coverage and avoid under-testing [37] [38] [39]. In reference [37], an iterative procedure based on genetic algorithm (GA) is reported. In each iteration, waveform simulations and fitness calculation are performed to guide selection, crossover and mutation to find patterns that induce larger PSN. The problems in this method are the dependency on initial population and the high simulation cost. In reference [38], PSN maximization is modeled as a MIN-ONE problem and a SAT solver is used to maximize the transition count. A SAT-based method can find a near-optimal solution. However, CPU time cost is usually high. The work in [39] utilizes a commercial ATPG engine to sensitize as many neighboring signal lines as possible by virtual test point insertion. This approach features good compatibility with commercial tools, but the optimality highly depends on the implementation of the industrial tool.

The third approach argues that chips should be tested under the maximal functional PSN and attempts to strike a balance between over-testing and under-testing by considering functional constraints and PSN at the same time. These approaches usually consist of two steps - pseudo functional test generation and PSN maximization, such as MAX-fill [40] and backward justification based approach [41]. Max-Fill computes functional reachable states that induce maximal switching activities using both logic simulation and backward justification. Later partially specified patterns are filled with these computed states. The approach proposed in reference [41] extracts logic

constraints, which constraint ATPG to avoid illegal states, and then justify transitions on neighboring signals.

1.5 Pseudo Functional Path Delay Test Generation

The focus of previous work is the on-chip IR-drop noise. However, silicon measured data reveals the necessity of considering both on-chip IR-drop and off-chip *dI/dt* noise in order to improve the delay correlation of structural and functional test [22] [42]. Pseudo functional KLPG (PKLPG) [18] [43] test is proposed to generate delay tests that test the *K* longest paths through each line in the circuit while having PSN similar to that seen during normal functional operation. Rather than scanning in a test pattern, applying it with a few functional test clocks and scanning out the results, PKLPG is applied by scanning in a test pattern, running multiple functional clocks, and then scanning out the result, as shown in Fig. 5.

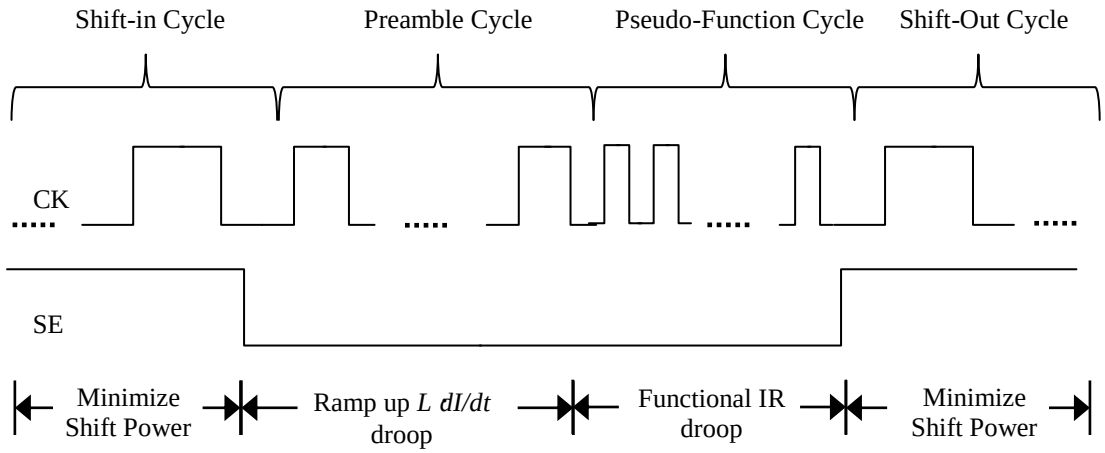


Fig. 5 Clocking scheme for pseudo functional test

The initial functional clocks, termed the preamble, run at lower speed to ramp up the off-chip power supply currents and minimize the noise due to the chip and package inductance (*dI/dt*). The preamble cycle time must be much less than the off-chip inductor time constant, but should be as large as possible to minimize the number of preamble cycles needed to stabilize off-chip inductor currents.

PKLPG can be viewed as a short burst of functional tests. The primary advantage is that it applies tests in a more functional manner, so power supply noise, signal coupling and power dissipation are more similar to functional operation. As for shift mode (both in and out), accumulative shift power dissipation can be minimized by DFT techniques, such as scan-chain partitioning [27], or test pattern manipulation, such as MT-filling [31]. Our in-house KLPG test generator, *CodGen* [18], was modified to support PKLPG pattern generation by introducing additional cycles prior to the launch-capture cycles. Currently, *CodGen* supports up to 32 preamble cycles followed by at-speed launch and capture.

As discussed above, PKLPG is effective in reducing the impact of *dI/dt* noise and produces an on-chip IR-drop environment similar to functional test. Now the major concern is ***whether the test pattern fully exercises the circuit and induces the maximal possible simultaneous switching activities, which in turn produce the worst-case delay on the path. Moreover, it is interesting to explore the functional PSN range between the max and min to verify timing robustness during post-silicon validation.*** Our PSN profiling results (detailed in Chapter II) indicates that functional PSN during the capture cycle is much lower and hence probably causes under-testing. Therefore, ***the goal of this***

work is to generate high quality test patterns and achieve high correlation with functional test.

1.6 Organization of the Dissertation

The dissertation is organized as described below.

In Chapter II, a simulation-based PSN control algorithm, Bit-Flip, is proposed to maximize the power supply noise during PKLPG test [44]. Given a set of partially-specified scan patterns, random filling is done and then an iterative procedure is invoked to flip some of the filled bits, to increase the effective weighted switching activity (EWSA). The experimental results on the b19 benchmark circuit show that the algorithm improved EWSA as much as 38% compared with best random fill. Also the results on both compacted and un-compacted test patterns demonstrate that Bit-Flip can significantly increase effective WSA while limiting the fill rate.

Chapter III proposes several techniques to improve the efficiency of Bit-Flip (iBF) [45]. iBF combines random flipping with deterministic modification to efficiently fill the don't-care bits. Background patterns, which sensitize transitions on critical cells, are pre-computed and later used to select the bits to flip. Dynamic bit weighting permits intelligent selection of background patterns. Experimental results on benchmark circuits shows iBF achieves worst-case realistic PSN in significantly less CPU time than Bit-Flip.

In Chapter IV, we address the problem of automatic test pattern generation for understanding circuit timing sensitivity to power supply noise during post-silicon timing validation [46]. Long paths are selected from a PKLPG path set to span the power

delivery network. To determine the sensitivity of timing to on-chip noise, the patterns are intelligently filled to achieve different PSN levels. The PSN control algorithm, Bit-Flip, is enhanced to consider both spatial and temporal information for better correlation with functional PSN. These patterns can be used to understand timing sensitivity in post-silicon validation by repeatedly applying the path delay test while sweeping the PSN experienced by the path from low to high.

Chapter V studies how compression affects PSN control. We first generate pseudo functional Illinois Scan and EDT patterns that target the longest paths through each gate. Then a compression-aware PSN control scheme is implemented to maximize the PSN while obeying the compression constraints. For each path, four different patterns are used for the experiment: un-compacted Illinois Scan, compacted Illinois Scan, un-compacted EDT, and compacted EDT. With these 4 patterns, we are able to study the PSN impact of both compaction and compression. The experimental results show that our PSN algorithm achieves significantly higher PSN compared to random or best random fill on both un-compacted and compacted patterns. Our constrained random (CR) algorithm for EDT compression reduces CPU time, while achieving slightly better results than best random fill.

Chapter VI summarizes the dissertation and discusses future work.

CHAPTER II

POWER SUPPLY NOISE CONTROL

2.1 Introduction

In recent years, considerable research effort has been dedicated to at-speed test using the path delay fault model since it has the advantage of capturing the accumulative effects of small delay defects. However, the maximum operating frequency during at-speed scan test may not correlate well to functional and system tests due to the mismatch of power supply noise. While PKLPG produces a functionally similar PSN environment for at-speed test, it potentially causes under-testing without PSN control. Moreover, it is interesting to explore the range of functional PSN during launch and capture. This can be used to verify timing robustness during post-silicon validation and to set the PSN margin. The central challenge of applying prior work to PKLPG is that PKLPG is a multiple-cycle sequential test. Computational cost increases dramatically with the number of preamble cycles, making it difficult to apply GA, SAT or justification-based methods. The objective of this chapter is to develop a PSN control scheme for PKLPG.

In this chapter, we first investigate the PSN scenario for PKLPG using random pattern simulation, which shows that PKLPG is more vulnerable to under-testing rather than over-testing. Then a simulation-based PSN control algorithm called Bit-Flip is proposed to maximize PSN during launch and capture cycles for partially-specified PKLPG patterns. Experimental results on un-compacted longest path patterns of ITC99 benchmark circuit b19 demonstrate our scheme is able to improve the effective WSA as

much as 38.7% compared with the best random fill. Results on compacted patterns show that Bit-Flip can perform effectively even if the care bit density (fraction of test pattern bits that are 0/1) is as high as 20%. The trade-off between CPU time and noise maximization is also discussed.

Bit-Flip is similar to the hill-climbing procedure reported in reference [35], but with a few differences. We explore Bit-Flip to maximize effective WSA for PKLPG test rather than minimize Hamming distance of scan cell states for traditional two-cycle delay test. We flip a group of bits per iteration rather than one at a time.

2.2 Pseudo-Functional PSN

This section presents the PSN profiling results for PKLPG using random pattern simulation and analyzes the impact of random filling on weighted switching activities during the capture cycle. Primary inputs are kept constant during simulation considering low-cost automatic test equipment has few high-speed pins. For each circuit, 30,000 random patterns are applied with a burst of 16 functional mode cycles. WSA is used as a simple metric to evaluate the PSN during each cycle. WSA at cycle *k* can be calculated using formula (1) - (3).

$$WSA_k = \sum_i WSA_i * S_{i,k} \quad (1)$$

$$WSA_i = \begin{cases} 1, & if\ \#FO == 1 \\ \#FO + 1, & Otherwise \end{cases} \quad (2)$$

$$S_{i,k} = V_{i,k-1} \oplus V_{i,k}, K \geq 2 \quad (3)$$

Where #FO is the number of fan-out of gate i; $V_{i,k-1}$ is the logic value of gate i at cycle k.

Fig. 6 and Fig. 7 depict the detailed results of PSN profiling. Chip1 and Chip2 are two industrial designs, while b19 and s38417 are two large circuits from the ITC99 and ISCAS89 benchmarks respectively. Fig. 6 shows the average WSA over the 16 cycles for the selected circuits. For convenience, WSA is normalized to the first capture cycle. As expected, for all the circuits the WSA falls rapidly during the first several cycles, and then stabilizes at 20-30% of the cycle 2 level after roughly cycle 6. The reason is that the high WSA in the first two cycles is probably introduced by illegal states. After applying preamble cycles, the illegal states die out gradually and the circuit approaches a near-functional state. Therefore, applying at least six preamble cycles (PKLPG) will produce a PSN environment closer to functional. Traditional two-cycle at-speed test is applied at cycle 2 and sees a higher WSA, while PKLPG sees a much lower WSA. Therefore, PKLPG is more vulnerable to under-testing rather than over-testing.

Fig. 7 illustrates the correlation between the WSA at cycle 2 and cycle 16 for b19. Similar results are observed for the other circuits. The WSA of the 30,000 random patterns at cycle 2 are sorted in increasing order. The corresponding WSA at cycle 16 is also plotted. The minimum and maximum WSA observed at cycle 16 is denoted as the "*Pseudo Functional PSN Range*". We can see that the WSA at cycle 16 (last capture cycle) is independent of that at cycle 2 (first capture cycle). This indicates that probability calculation based methods [32] [34] cannot guide filling in the presence of many preamble cycles. The large pseudo functional PSN range at cycle 16 indicates the importance of don't-care filling. Given a partial specified pattern, the X bits should be

assigned wisely to sensitize the worst functional PSN condition. In the following section, we will describe a simulation-based X-filling method to control PSN.

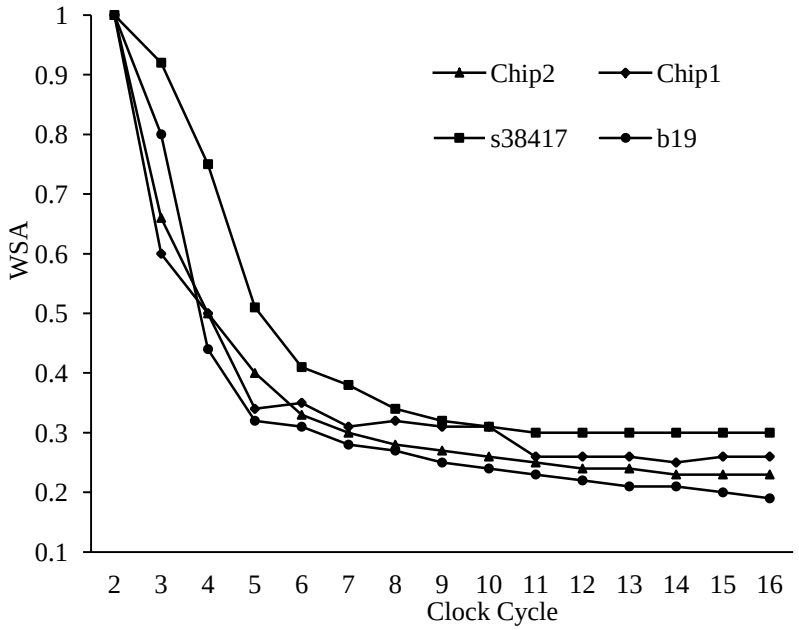


Fig. 6 Average WSA falls with multiple cycles

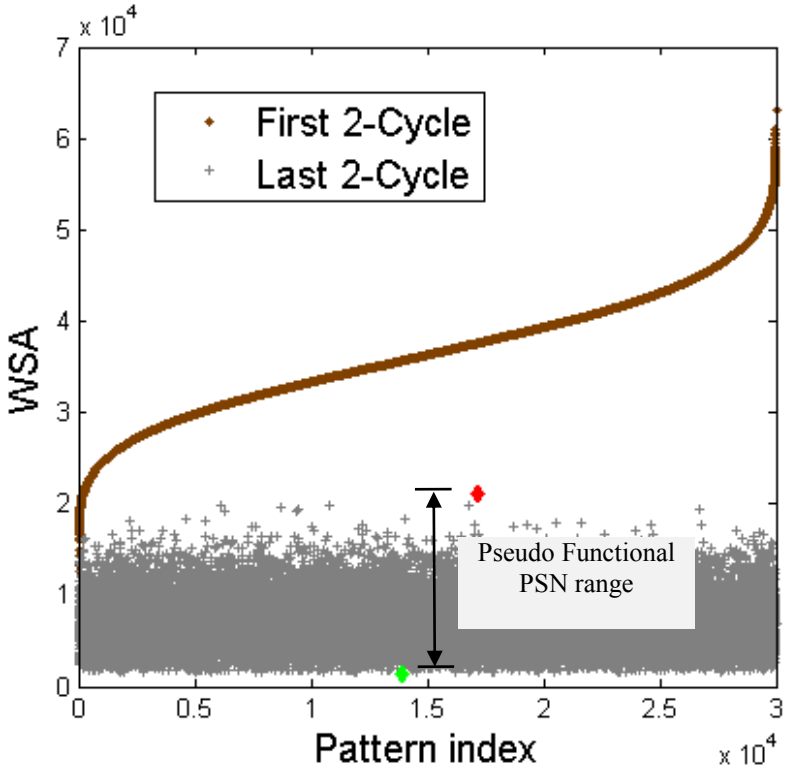


Fig. 7 b19 WSA correlation of different cycles

2.3 Overview of Proposed Framework

In this section, a simulation-based PSN maximization scheme, Bit-Flip, is described. For a given partially-specified test pattern, it attempts to incrementally improve the effective WSA by flipping a group of randomly-selected X-bits. The Bit-Flip procedure is detailed in Fig. 8. It consists of a preprocessing step, an iterative step and a bit-relaxation step. Circuit netlist, layout, critical path list and test pattern set are the inputs to the algorithm.

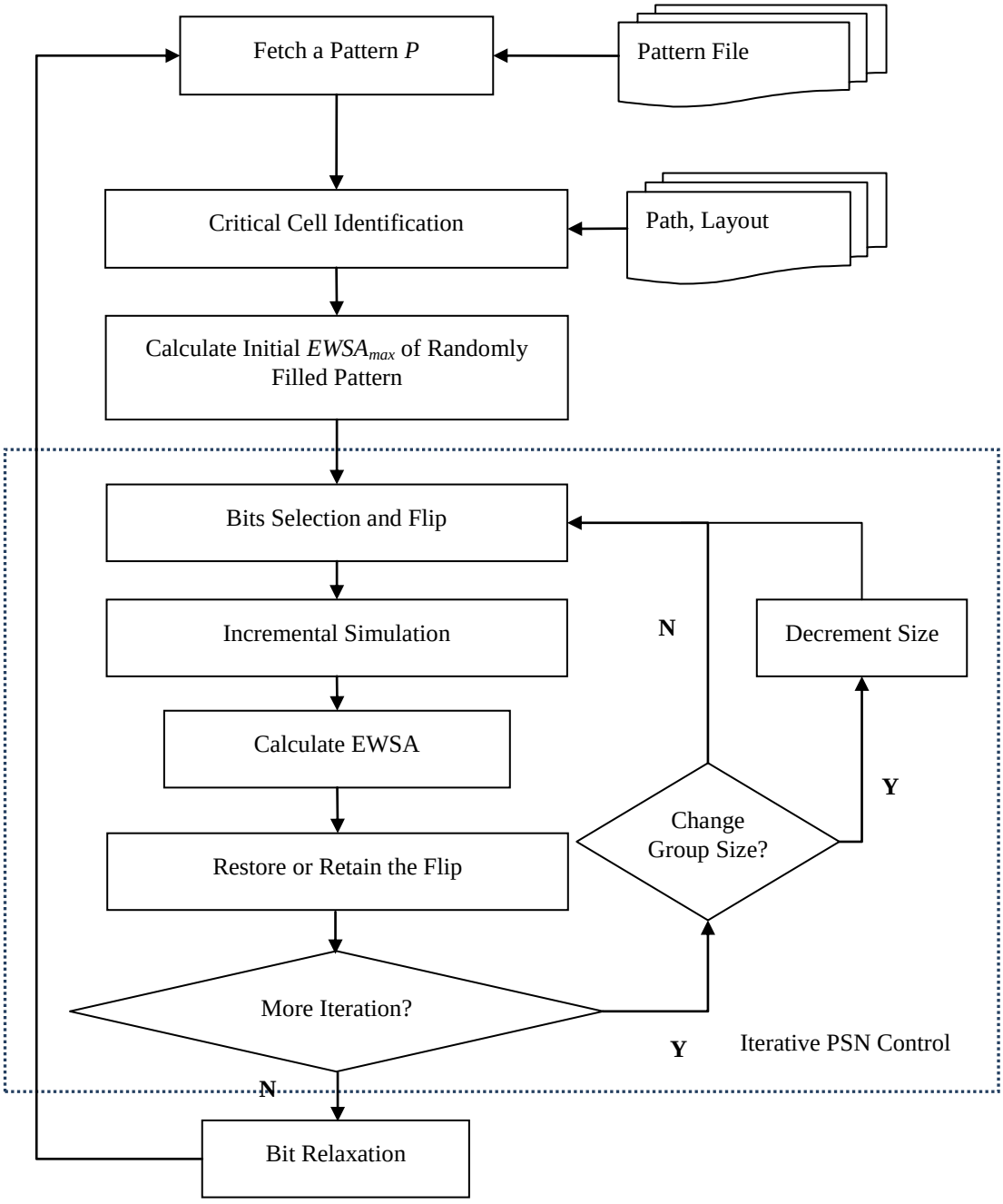


Fig. 8 Bit-Flip flow

In the preprocessing step, a pattern is fetched from the pattern file as well as the corresponding path(s). First the cells near on-path gates are identified by physical

position matching and are stored in a list. We term these cells as *critical cells*. It has been demonstrated that critical cells have considerable impact on the PSN of on-path gates [39]. Meanwhile, scan cells located in the fan-in cone of the critical cells are also marked and stored in another list. These scan cells are termed as *critical bits*. Next, all critical bits are randomly filled and an event-driven logic simulator, *CodSim* [47], simulates the filled pattern. Then EWSA is measured as the initial $EWSA_{max}$. EWSA is defined as the sum of transitioning critical cells' WSA.

In the second step, the number of rounds and initial group size (initial value of *G*) are chosen based on the CPU time budget. At the beginning of a round, the scan cells that have constant logic values at the last launch/capture cycles are collected as potential scan cells. The union, *U*, of the critical bits and the potential scan cells serves as the final bit set for PSN control. Then Bit-Flip enters the iterative PSN control process. In each iteration, it randomly selects up to *G* scan cells from the set *U* and flips their logic value. Then *CodSim* simulates the modification incrementally, and a new EWSA is measured. If EWSA is increased, the flipped bits are retained and $EWSA_{max}$ is updated; otherwise, the flipped bits are restored. At the end of each round, the group size is reduced by a constant. The flipping process is terminated when the maximum number of rounds or enough failures in a row is reached. After the iterative procedure concludes, bit-wise relaxation is performed to maximize the number of X-bits, for the benefit of MT-filling [31] or test compaction [3] [4].

2.3.1 Critical Cell Identification

In this work, we utilize the effective region to identify critical cells, as shown in Fig. 9. This approach is motivated by the model used in reference [39]. Vertically, gates in the same and neighboring rows are critical cell candidates since they share either power or ground lines with the on-path gates. Horizontally, each row is divided into segments by power/ground strips. Based on power grid analysis, an effective region can be set around on-path gates in order to capture the localized nature of PSN. All gates within the effective region are critical cell candidates.

We perform a 3-value (logic 1, 0, X) simulation on the partially specified pattern. All candidates that have undetermined values (XX, 1/0X, X1/0) at launch/capture cycles are denoted as critical cells. After critical cells are identified, Bit-Flip attempts to maximize the sum of the WSA of critical cells (EWSA).

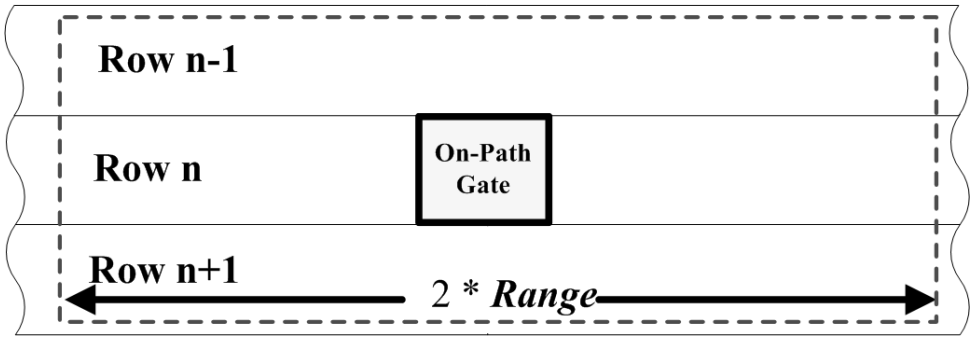


Fig. 9 Effective region

2.3.2 Task Granularity

Bit-Flip flips a group of bits each time. To select an appropriate group size, we need to consider the potential EWSA improvement as well as the simulation time cost.

Assuming there is no overlap among the fan-out cones of the flipped bits, simulation time increases linearly with the group size and the total iterations. This is usually true if the group size is much smaller than the number of test pattern bits. The flipped bits spread sparsely along the scan chains.

The total number of bits that are covered by Bit-Flip is:

$$B = \sum_{i=0}^{\left|\frac{G}{d}\right|-1} I \cdot (G - d \cdot i)$$

where I is the total iterations of each round, G is the initial group size, and d is the decrement of the group size. The time cost of Bit-Flip can formulated as:

$$T = C \cdot B$$

where C is the simulation time cost of flipping one bit. In order to reach the maximal PSN, two conditions much be satisfied: 1) $B \gg S$, where S is the number of scan bits. This guarantees that each bit is flipped enough times; 2) I is large enough to adequately explore the exponential search space.

In practice, the time budget T is fixed. Therefore, the total number of bits B that can be flipped is fixed. Here we assume that condition 1) is satisfied. With a fixed B, in order to make I large enough, we need to make G as small as possible. However, too small a group size causes the transitions (flipped bits) to die out over the preamble cycles, and so not improve EWSA.

Therefore, in Bit-Flip we first try a large group size to search across the exponential space and approach some local optima PSN. By decreasing the group size round by round, we gradually achieve the optimal result as well as limiting the execution time.

2.3.3 Critical Bit Fill and Bit Relaxation

In order to narrow down the search space, structural information can be used to identify critical bits that are highly correlated to the logic value of the critical cells. Bits in the fan-in cone of critical cells are most likely critical. After critical cells are identified, a multi-cycle back-trace procedure is called to collect critical bits. However, multi-cycle back-trace may cause too many bits to become critical, which will increase the fill rate and degrade pattern compaction performance.

To limit the number of critical bits, we need to identify the bits which are insignificant to EWSA maximization and exclude them from the critical bits set. That is, if we relax an insignificant bit to X, EWSA will not be reduced. Therefore, we apply a bitwise bit-relaxation procedure to turn insignificant bits into X bits. The procedure relaxes each bit to X, simulates the circuit, and keeps the relaxation if EWSA is not decreased. Otherwise the bit is restored. An efficient relaxation method can be found in reference [48], although their focus is fault coverage, not PSN.

If the fill rate of the test patterns is limited, such as to enable high test compression ratio, a trade-off must be made between EWSA maximization and X-bit utilization. This is done by adding a significance ranking to X-bits during the relaxation process. We use the change in EWSA to rank the bits. This can then be used to select which bits are relaxed.

2.3.4 Compacted Pattern Consideration

Test compaction is used to reduce pattern count and minimize the test application time. Compacted patterns typically have higher care-bit density, which reduces the

search space for PSN control. Bit-Flip can be applied to compacted patterns with slight modification.

First, the paths tested by a given pattern can be searched in a breath-first manner. If the pattern tests a critical path, we term this a critical pattern and Bit-Flip is applied to it. Critical paths can be obtained from static timing analysis (STA) tools or by setting a threshold on path length. In practice, it can be selected based on path length distribution and CPU time budget. If the compaction algorithm attempts to pack critical paths together, the number of critical patterns may be small. In our future work, we will enhance the compaction algorithm to compact long paths together, so the critical pattern count is reduced.

Second, critical cells are identified for each critical path tested by the critical patterns, and its EWSA weighted based on the path length. Since a longer path is more sensitive to PSN induced delay, a larger weight is assigned to its critical cells. The weight is the ratio of path length to the longest path length (or clock cycle time). If there is an overlap of critical cells on different paths, the WSA is weighted by the longest path. Bit-Flip attempts to maximize the weighted EWSA of all critical cells.

2.4 Experimental Results

We implemented Bit-Flip in *C++* running on a 3.16 GHz processor with 4 GB of memory. Robust paths and patterns are generated using the in-house PKLPG tool, *CodGen*, with *K=1* (one longest rising and falling path per line) and 6 preamble cycles. Physical layouts were generated using commercial tools. In the following, Bit-Flip with

N iterations will be termed BF-N. The 10 longest paths from b19 that do not share gates were selected for experiments. These paths/patterns are termed P0 to P9.

First, we investigate how group size affects Bit-Flip performance for a fixed CPU time budget. We ran Bit-Flip on path P0 while limiting CPU time to 10s. This is a generous amount of CPU time for one path. For each group size, we filled the pattern 1000 times and the average EWSA is compared with the best of 10,000 randomly-filled patterns (ΔEWSA). As shown in Fig. 10, the average ΔEWSA peaks for an initial group size of 30, which is about 0.5% of the total bits. Similar results are observed on ISCAS89 circuits S38417, S38584, and S35932, which peak at group size 5. A larger group size can discover the logic correlation among bits. However, too large a group cannot maximize the average ΔEWSA within the time budget.

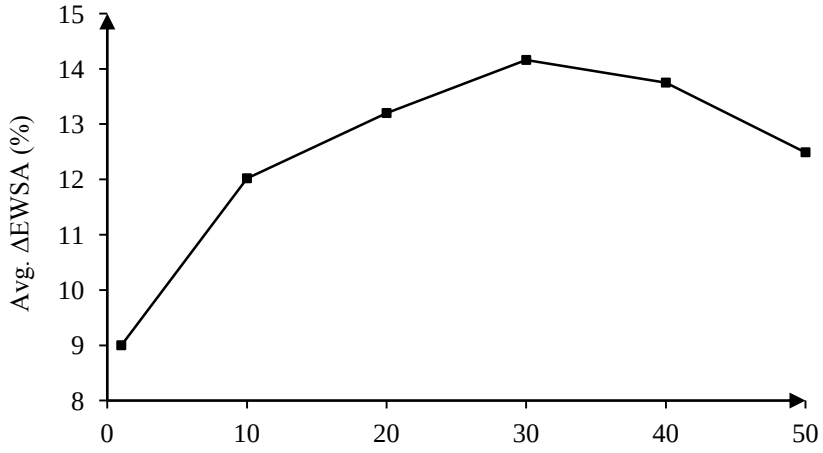


Fig. 10 b19: Average ΔEWSA vs. group size

Second, we investigated how the number of iterations affects performance. We ran Bit-Flip on P0 to P9 for 1000, 4000 and 10,000 iterations (BF-1000, BF-4000, and

BF-10000) with an initial group size of 30. To validate Bit-Flip effectiveness, each pattern is filled 100 times for each configuration and the results are compared with the best random patterns as shown in Table 1.

Table 1 PSN control result of un-compacted patterns

Path No.	Path Length (gates)	Initial Care Bits	Bit-Flip (30-bit initial group) Compared With Best Random Fill								
			BF-1000			BF-4000			BF-10000		
			Avg. ΔEWSA (%)	Final Care Bits	CPU Time (s)	Avg. ΔEWSA (%)	Final Care Bits	CPU Time (s)	Avg. ΔEWSA (%)	Final Care Bits	CPU Time (s)
P0	59	160	14.45	475	11	22.03	472	41	24.00	466	100
P1	43	205	17.83	492	11	21.63	500	41	21.86	506	97
P2	55	178	1.28	376	11	3.35	388	38	4.88	389	93
P3	49	160	6.29	412	11	14.70	425	39	16.00	425	98
P4	36	189	22.35	393	10	37.46	404	35	38.70	404	89
P5	32	185	17.12	385	10	24.64	392	36	26.16	392	89
P6	32	142	4.65	320	10	5.70	319	35	8.54	334	90
P7	40	220	7.73	471	11	10.02	476	36	11.74	484	95
P8	48	158	-0.76	336	10	2.33	337	36	2.55	342	90
P9	39	210	12.19	449	10	14.92	449	41	16.48	460	94
Avg.		181	10.31	410	11	15.71	416	38	17.09	420	94

In Table 1, the initial and final care-bit count, average ΔEWSA and CPU time are shown for each path. The average ΔEWSA of BF-1000, BF-4000, and BF-10000 are 10.31%, 15.71% and 17.09% respectively. The best performance is observed on P4 using BF-10000, which has a ΔEWSA of 38.7%. Most paths have a 10%-25%

improvement using BF-4000. The rate of EWSA improvement levels off with more iterations. For most paths, BF-4000 provides the best trade-off between PSN maximization and CPU time.

The 95% confidence interval for average ΔEWSA is shown in Fig. 11. There is a relatively large range of pseudo functional EWSA for a given path. Quiet and noisy patterns can be binned and used to characterize the noise sensitivity of the paths. For example, Fig. 12 illustrates the EWSA distribution for P0 of 1000 randomly filled patterns and 1000 patterns filled using BF-1000 and BF-10000. By applying patterns from left (quiet) to right (noisy) and computing FMAX for each bin, the sensitivity of delay to PSN can be understood.

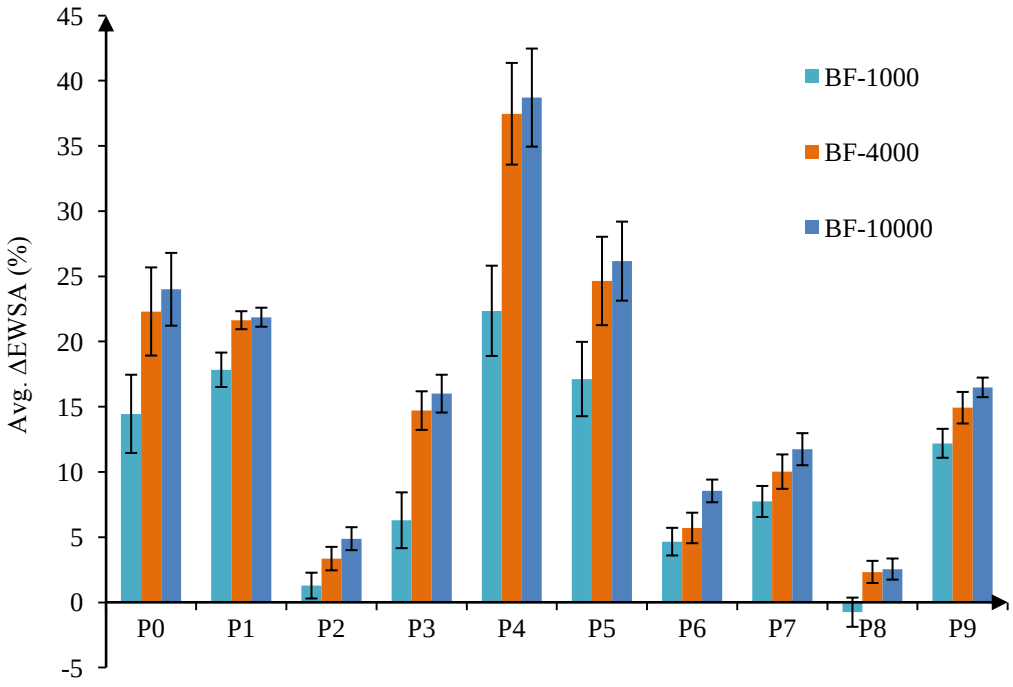


Fig. 11 Average ΔEWSA with 95% C.I.

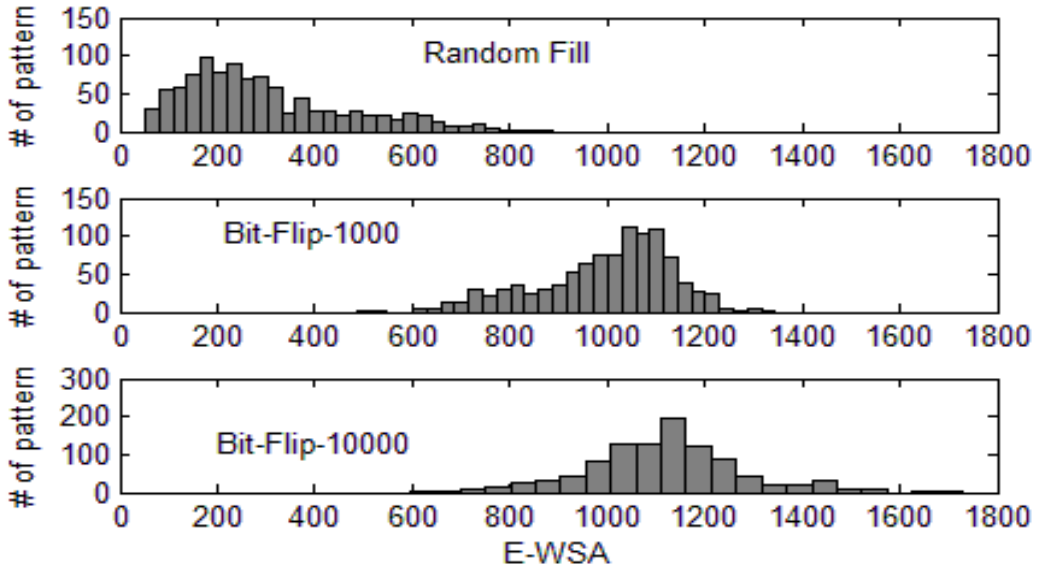


Fig. 12 P0 EWSA distribution vs. fill method

Bit-Flip provides the least improvement on paths P2, P6 and P8 compared to the best random pattern. To understand these three cases, the total number of critical cells (T.C.), transitioning critical cells count (T.O.) and transition rate (T.R.) are shown in Table 2. It can be seen that the transition rate of these three paths is relatively higher than other paths. The noise on these three paths is relatively high and there is not much room for improvement.

The number of X bits used for PSN control is about 40% more than the original care bits. In aggregate less than 10% of the pattern bits are specified. Since logic simulation time dominates the algorithm, the CPU time is nearly linear in the number of iterations. BF-4000 takes about 40s on b19 while simulating 10,000 random patterns takes more than 2 hours.

We compare the Bit-Flip approach to the ATPG-based PSN maximization approach in [39]. Based on their published data, we can only compare the average

transition rates (T.R.) (total aggressor transitions divided by total aggressors). The average T.R. in [39] is 14.08% with virtual test point insertion, with a CPU time of 40s for the ATPG step. As shown in Table 1and Table 2, BF-4000 averages 17.24% T.R. at 38s CPU time. So the two methods have similar performance.

Table 2 No. critical cells with transition output

Path No.	T.C.	BF-1000		BF-4000		BF-10000	
		T.O.	T.R. (%)	T.O.	T.R. (%)	T.O.	T.R. (%)
P0	5818	494	8.49	521	8.96	525	9.02
P1	5544	335	6.04	343	6.19	344	6.21
P2	5181	1282	24.74	1301	25.11	1313	25.33
P3	6827	695	10.18	753	11.03	756	11.07
P4	2148	285	13.27	318	14.80	320	14.92
P5	2206	296	13.42	316	14.31	320	14.49
P6	2492	536	21.51	544	21.84	560	22.46
P7	5501	714	12.98	735	13.35	747	13.58
P8	3737	1147	30.69	1179	31.55	1181	31.60
P9	3645	902	24.75	921	25.28	935	25.65
Avg.	4309	668	16.61	693	17.24	700	17.43

We investigated how fill rate constraints limits the performance of Bit-Flip. We run BF-4000 with group size 30 and the fill rate is varied from 3% to 10%, compared to the original fill rate of 2.4%. For each case, we run BF-4000 100 times on P0. After BF-4000 completed, the remaining X bits in the filled pattern were randomly filled for a fair comparison. The average ΔEWSA from random fill (Average ΔEWSA-R) and from best

random (Average ΔEWSA-BR) are shown in Fig. 13. BF-4000 always performs better than random fill and always performs better than best random once the fill rate is more than 5%. The improvement for P0 levels off when the fill rate is above 7%.

Table 3 PSN control result of compacted patterns

Circuit	T.P.	C.P.	Avg. ΔEWSA (%)	T.R. (%)	Avg. Care Bits		CPU Time (s)
					Original	Post-fill	
b19	283	37	263	11	5%	9%	1003
s35932	235	100	53	67	21%	56%	174
s38417	519	98	59	29	24%	38%	100
s38584	198	63	74	28	19%	34%	67

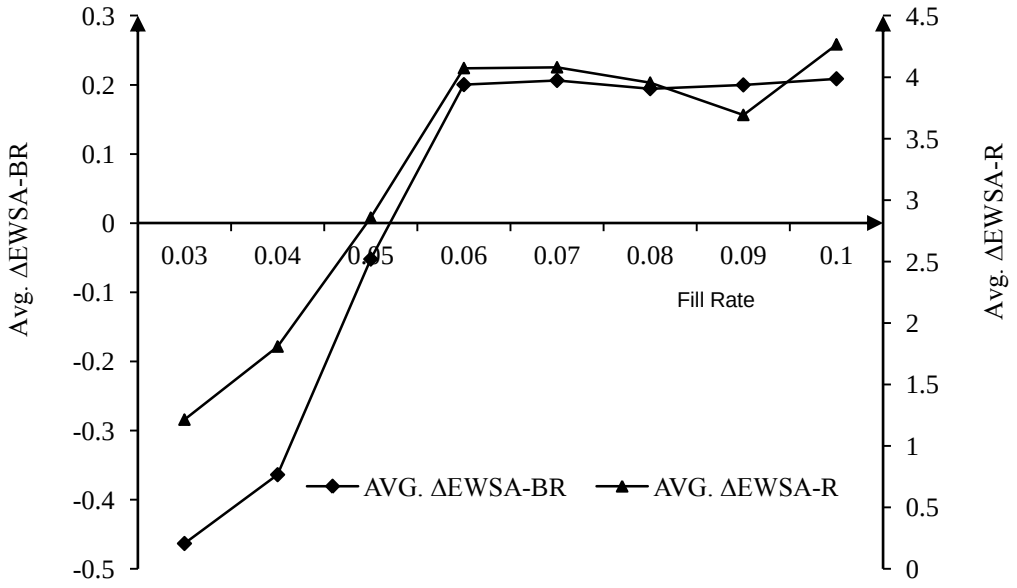


Fig. 13 Average ΔEWSA vs. fill rate

Finally, we evaluate the ΔEWSA achieved on compacted patterns of benchmark circuits and the results are listed in Table 3. The patterns were dynamically compacted. Paths longer than 70% of the longest path are considered critical, and any patterns containing them are subject to the Bit-Flip procedure. We chose 70% as the threshold since STA errors of 30% have been reported in the literature. The total number of patterns (T.P.), the number of critical pattern (C.P.) and transition rate (T.R.), are also shown. On average 24% of patterns are critical and require PSN control. b19 is less compacted than the other three circuits and a large ΔEWSA is obtained. Although the other three circuits have about a 20% care bit density, Bit-Flip still performs significantly better than random fill.

2.5 Summary

The noise scenario for pseudo functional test using KLPG patterns is quite different from traditional two-cycle launch-on-capture delay test and it is vulnerable to under-testing. We proposed a simulation-based algorithm, Bit-Flip, to control PSN for PKLPG test. Experimental results on both un-compacted and compacted test patterns demonstrated the effectiveness of the method.

CHAPTER III

IMPROVED POWER SUPPLY NOISE CONTROL

3.1 Introduction

In this chapter, the improved Bit-Flip (iBF) is presented. It combines random flipping with background patterns based (BGs-based) modification for higher PSN control efficiency. Initially, Bit-Flip is applied to sensitize transitions on the outputs of high-controllability critical cells. As this approach saturates, iBF switches to a BGs-based approach to identify additional transitions. This is similar to a standard ATPG process. Dynamic bit weighting permits intelligent selection of background patterns for PSN improvement. Experimental results on benchmark circuits show that iBF achieves results similar to Bit-Flip in much less CPU time.

3.2 Background and Motivation

The original flow of Bit-Flip is shown in Fig. 14. Bit-Flip starts by fetching a test pattern and the corresponding path(s). Critical cells, which are neighboring-row gates located within a certain distance (critical range) of on-path gates, are identified by layout analysis. Don't-care bits are initially randomly filled and the EWSA is computed as the initial $EWSA_{max}$. EWSA is the sum of the WSA of transitioning critical cells. Then Bit-Flip enters the iterative PSN control step and randomly flips a group of bits to maximize the EWSA. In each iteration, up to G don't-care bits are randomly selected and their values are flipped. Incremental logic simulation is used to simulate the change in test

pattern. The flip is retained if EWSA does not decrease. To reduce CPU time and direct the search, Bit-Flip shrinks the group size *G* during the iterations.

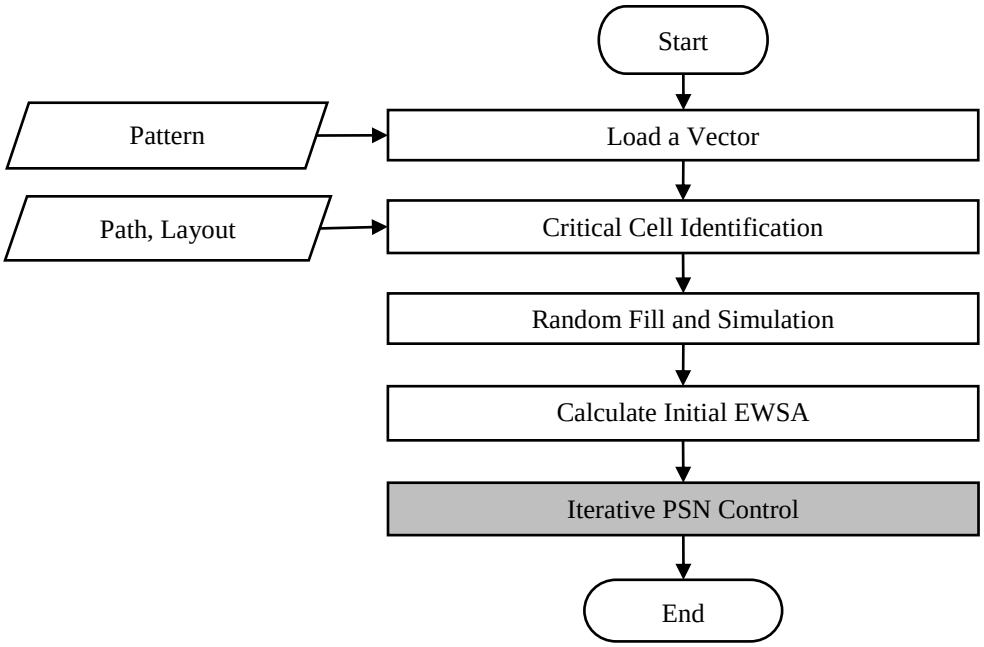


Fig. 14 Original Bit-Flip flow

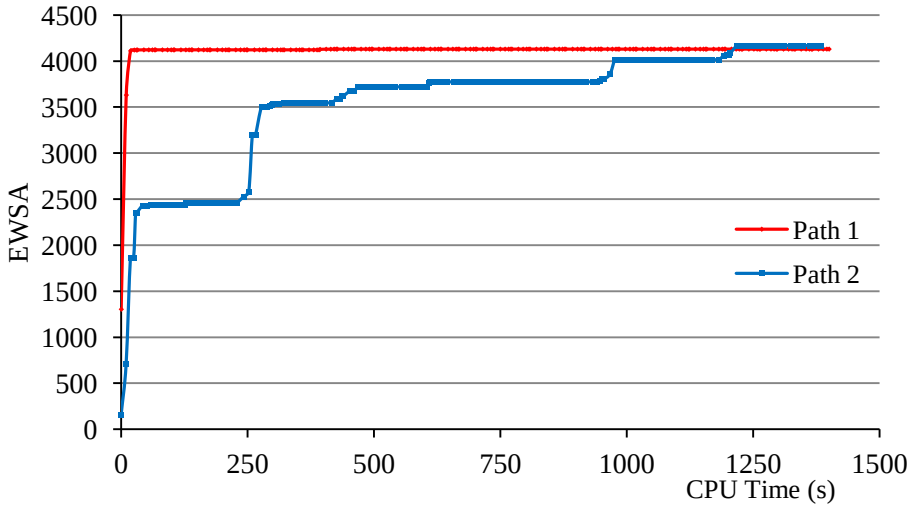


Fig. 15 Bit-Flip PSN control scenarios

Bit-Flip devotes the same effort to each critical pattern (a pattern containing critical paths) regardless of the potential EWSA improvement. For some patterns, EWSA is maximized after a few iterations and the remaining flips provide little benefit. Some patterns require many iterations. In Fig. 15, Path1 quickly reaches its maximal EWSA while Path2 requires a lot more iteration to find each EWSA "jump". Therefore, a flexible iteration control will reduce CPU time while maximizing EWSA. Another drawback is that Bit-Flip strongly relies on the random selection of bit groups to discover the logic dependency among bits. Many iterations are required for random-pattern-resistant (RPR) critical cells. Logic dependence refers to situations where two or more bits should be flipped simultaneously to increase EWSA.

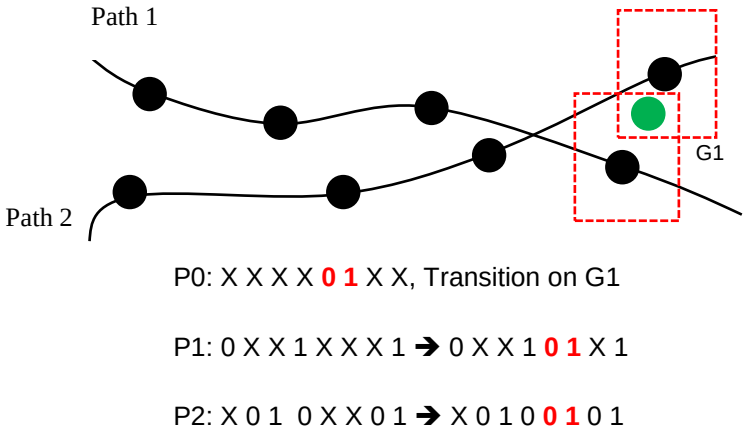


Fig. 16 Critical cell sharing

The basic principle of iBF is to take advantage of both random flipping and deterministic modification. Random flipping works well, except for RPR critical cells. While deterministic justification is adequate in covering RPR critical cells, the cost is

usually high due to multi-cycle backtracking across the preamble cycles in PKLPG tests. Similar to ATPG [1], random flipping can be applied first to generate transitions on the critical cells that are not RPR. When the EWSA levels off, the algorithm switches to the deterministic method for further EWSA improvement.

The observation that critical cells are shared by multiple paths permits further CPU time reduction. As shown in Fig. 16, Path1 and Path2 are two critical paths and G1 is located within the critical ranges of the two paths, which is the overlap of the two dashed rectangles. Therefore, G1 is a shared critical cell. Considering these two paths independently requires separately justifying a transition on G1 for both paths. Using the approach in Fig. 16, one pass is sufficient. Assume pattern P0 sensitizes a transition on G1, and P1 and P2 test the two paths. Since P0 is compactable with P1 and P2, we can simply merge P0 into P1 and P2 to generate the desired transitions. Therefore, the CPU cost is halved.

Table 4 Critical cell sharing summary

Circuit	s35932	s38417	s38584	b19
#Path	3488	2660	1484	1030
#Path/CC	250	201	83	75

In Table 4, we listed the critical cell sharing in several benchmark circuits. The second row showed the number of PKLPG paths in the top 30% in length. The third row listed the average number of paths that share each critical cell. Critical cell sharing is a common phenomenon. For example, in s38417 each critical cell is shared by an average

of 201 paths, which potentially offers as much as 200x speedup compared with justifying the critical cells for each path separately. Based on the above discussion, we can pre-compute the patterns sensitizing the critical cells and use these patterns to modify the path test patterns.

3.3 Background Pattern

Background pattern (BP) refers to a pre-computed partially specified pattern, which generates transitions on the outputs of critical cells. Background pattern generation, the structure of the background pattern pool and the weighting of background patterns will be discussed in this section.

3.3.1 Background Pattern Generation

The following three requirements should be taken into account when generating the background patterns. First, the number of BPs should be minimized to reduce CPU time. iBF iterates over the BPs and modifies the test pattern based on BPs. Fewer BPs means less logic simulation. Second, each BP should have as few bits specified as possible. Only the bits necessary for the targeted critical cell transitions are specified. Unnecessary specified bit values increase the possibility of bit assignment conflicts, resulting in higher CPU time. Third, it is necessary to include diverse BPs for each critical cell. Although a critical cell can be shared by multiple paths, different test patterns may have different care bits and thus require different BPs for the critical cell. If possible, we should include every BP input assignment that generates the desired critical cell transitions.

The three requirements for BPs are in conflict. Dynamic compaction [3] is effective in reducing pattern count but violates the second requirement. Critical cell dropping, similar to fault-dropping in fault simulation, can be used to reduce pattern count by removing a critical cell once it is covered by a generated BP. However, this conflicts with the third requirement. To strike a balance between the first and the second requirements, compaction with a care-bit density constraint can be used to reduce BP count. A compromise can be reached between the second and third constraints by allowing multiple sensitizations for each critical cell. If the size of the critical cell list is too large and the justification time is unacceptable, we may only target critical cells on which transitions are hard to generate. Controllability [1] can be used to select such critical cells.

```
Algorithm: BP_Generation (CC_list)
For each critical cell CC_i
   If (more_rising (CC_i))
      If  (justification(rising, CC_i);)
          Record pattern and update coverage;
   If  (more_falling(CC_i))
      If  (justification(falling, CC_i))
          Record pattern and update coverage;
Endfor
```

Fig. 17 Background pattern generation algorithm

Fig. 17 shows the BP generation process. Critical cells are identified for all critical paths before BP generation and are stored in the critical cell list (*CC_list*). For each critical cell, background patterns that produce rising and falling output transitions

are generated using a PODEM-like justification algorithm [2]. The other critical cells are checked to see if the new BP causes an output transition on them. Once a critical cell is sensitized for *C* rising and *C* falling transitions, it is dropped from the critical cell list. *C* is set in accordance to the number of critical paths that share the critical cell. A critical cell shared by more critical paths typically requires more BPs to mitigate the conflicts among the critical patterns. Critical cells are considered in reverse rank order, since this increases the fortuitous drop rate for critical cells. If a BP for one critical cell matches an existing BP, the BPs are merged.

3.3.2 Background Pattern Pool

The background pattern pool (BPP) is a bipartite graph that stores BPs and the critical cells covered by each BP. BPP can be built easily by logic simulation. For each BP, the set of transitioning critical cells are associated with the BP by adding edges between them. Each critical cell maintains a list of BPs that causes transitions on the cell. Each edge is weighted using the WSA of the critical cell.

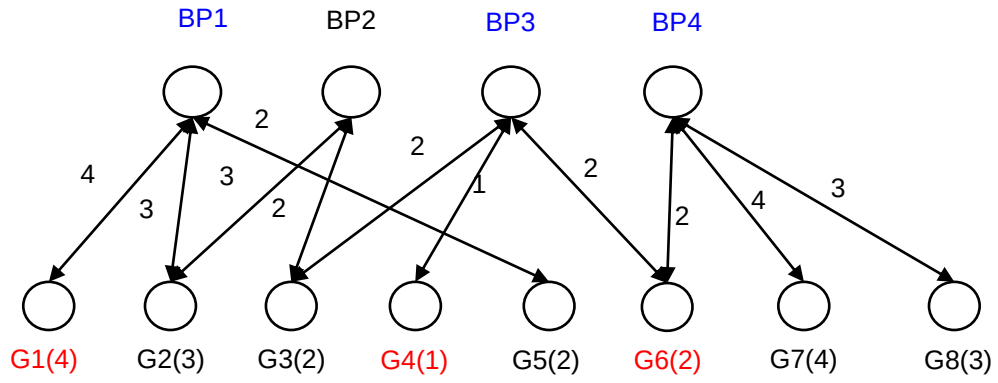


Fig. 18 Background pattern pool

Consider a BPP example in Fig. 18. It consists of 4 BPs (BP1 to BP4) and 8 critical cells (G1 to G8) with WSA labeled in parentheses. Edges are added between BPs and critical cells to indicate the coverage relationship. For example, since G1 is covered by BP1, an edge is added and the weight of the edge equals the WSA of G1, i.e. 4. The weights of other edges are assigned similarly.

3.3.3 Evaluation of Background Patterns

Potential EWSA (PEWSA) is a static measurement for evaluating the potential of a BP to increase EWSA. PEWSA is the sum of the weights of edges connecting the BP to critical cells. For a given critical path, a set of critical cells are labeled and the PEWSA is calculated by traversing the BP list of each critical cell.

As illustrated in Fig. 18, G1, G4, and G6 are three critical cells. From the graph, we know that BP1 appears in G1's BP list, thus $PEWSA(BP1) = WSA(G1) = 4$; BP3 covers both G4 and G6, thus $PEWSA(BP3) = WSA(G4) + WSA(G6) = 3$; BP4 has a connection with G6, thus $PEWSA(BP4) = WSA(G6) = 2$.

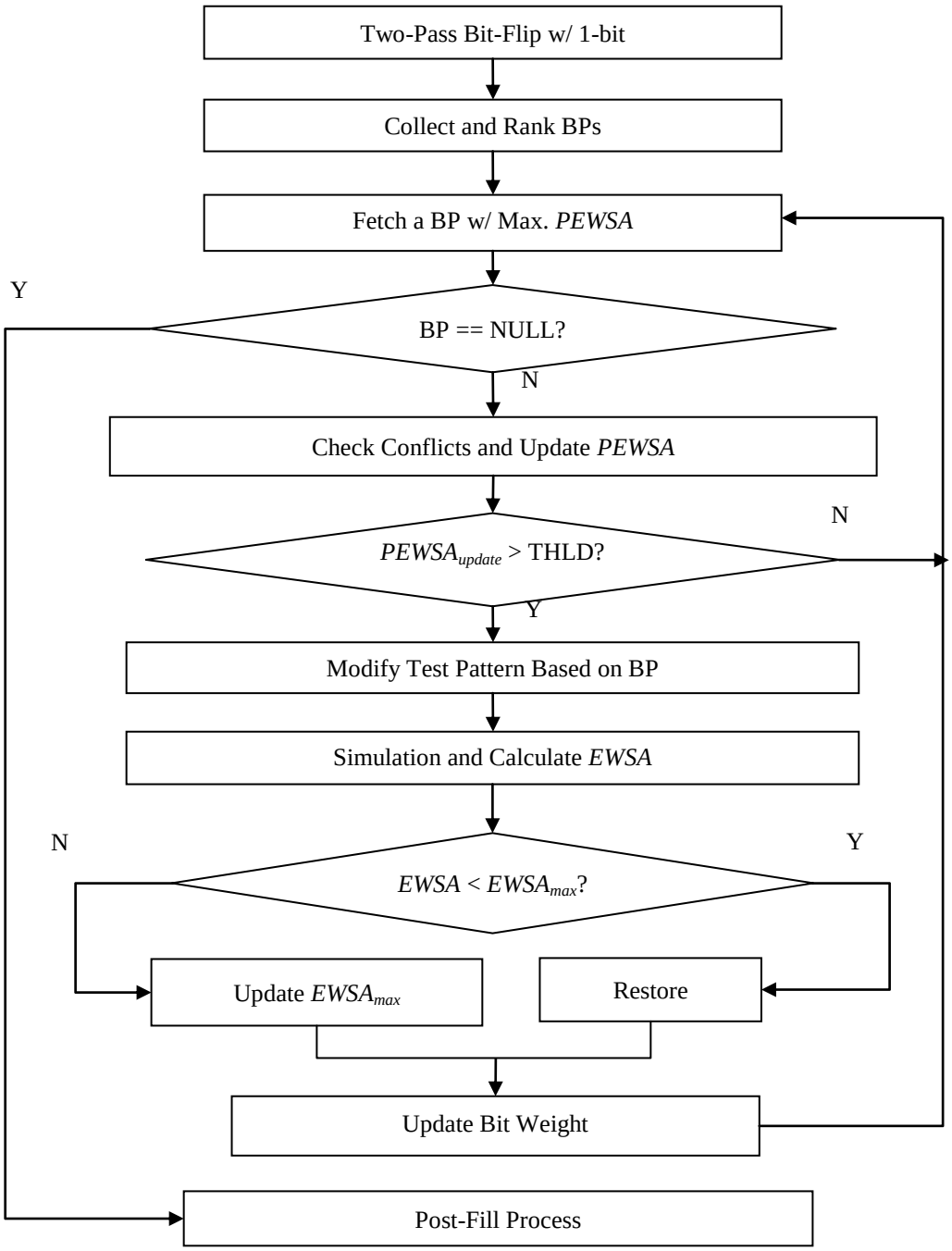


Fig. 19 iBF flow

3.4 Improved Power Supply Noise Control

The details of iBF are discussed in this section. As with the Bit-Flip algorithm, the goal of iBF is to maximize EWSA on the long paths in each test pattern. iBF has the same preprocessing phase as Bit-Flip, as discussed in Chapter II. We will focus on the PSN control as shown in Fig. 19. The iBF core consists of two parts. Initially, a two-pass Bit-Flip with single-bit group size is applied. This stage will cover most of the easily controlled critical cells and bring the EWSA to the level-off point. Then the BPs-based approach is applied to cover the remaining critical cells by modifying the test pattern using the background patterns. Since BPs provide a directed basis to modify the test pattern, the BPs-based approach reduces CPU time while maintaining the same test quality as the original Bit-Flip algorithm.

3.4.1 Two-pass Bit-Flip w/ Single-bit Group

Two-pass Bit-Flip with single-bit group size is applied to the randomly filled pattern. The first pass Bit-Flip flips each bit in the same order as it occurs in the scan chain (referred to as BF.1-Bit), and the second pass in the reverse order (referred to as BF.1-Bit-R). After two-pass Bit-Flip, most of the critical cells will have transitions on them. The remaining critical cells are relatively hard to control and require the knowledge of bit correlation, i.e. flip some combination of bits to launch the transition. We assume the EWSA after two-pass single-bit flip has leveled off. If this is not the case, additional random Bit-Flip iterations can be applied, using the critical cell toggling rate to identify the leveling-off point. In this work, we use the two-pass Bit-Flip.

3.4.2 BPs-Based Flip

iBF considers BPs in decreasing order of PEWSA. The BP is fetched, and its PEWSA updated, to account for the fact that some critical cells already have transitions, and conflicts between the BP and the test pattern filled by the two-pass Bit-Flip. These conflicts are likely to cause some potential PEWSA loss. A BP will not be used unless its updated PEWSA is larger than a threshold THLD.

iBF modifies the test pattern based on the fetched BP, incrementally simulates the modification, and computes the EWSA. Any modification that reduces the EWSA will be restored. Handling of conflicts and dynamic bit weighting will be detailed in the following sections. This process is repeated until there is no untried BP.

3.4.3 Rank of BPs

The PEWSA of each BP was originally computed without any constraints, so that BP generation could be done once for the entire circuit. When considering a BP, the PEWSA is updated using dynamic bit weighting and potential weight updating using the following formula:

$$PEWSA_{update} = PEWSA - \sum_{i} WSA_i - \sum_{j} NW_j$$

where NW_j is the negative weight of conflicting bit j and WSA_i is the WSA of critical cell i that is sensitized by the BP, but already has a transition. The NW is the reduction in PEWSA that results when flipping a bit. The WSA of the already transitioning critical cells must be subtracted, to avoid double credit for them. The updated PEWSA is viewed as the maximal EWSA the BP may achieve.

3.4.4 Handling of Conflicts

There may be conflicts between the intermediate test pattern (ITP) and the BP. The ITP is the original test pattern (OTP) with any subsequent assignments to the don't-care bits. A conflict is classified as either a hard conflict or soft conflict. A hard conflict occurs when the BP has some care bits that differ from the OTP care bits. A soft conflict is a conflict between the BP and the current assignment of the ITP don't care bits.

An example of hard and soft conflicts is shown in Fig. 20. The first bit is 0 in the BP and 1 in the OTP. This is a *hard conflict*. We must keep the value in the OTP to preserve the path test. Therefore, Rule1 is applied for hard conflicts. The fourth and the eighth bits are *soft conflicts* since they are don't care bits in the OTP, but have current assignments that conflict with the BP. Rule2 is applied for soft conflicts. Bit positive weight will be discussed in the next section.

- *Rule1: Any flip to hard conflict bit is **rejected**.*
- *Rule2: Flips to soft conflicts are **accepted** if the $PEWSA_{update}$ is ≥ the positive weight of the bit*

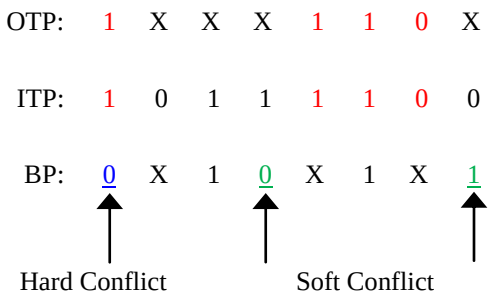


Fig. 20 Classification of conflicts

3.4.5 Dynamic Bit Weighting

Conflicts with the ITP may require flipping a care bit in the BP, reducing its PEWSA. For each bit in the ITP, positive weight and negative weight are maintained. Positive weight tracks how much a bit contributes to the current EWSA. It is used to determine if iBF flips an ITP bit due to a soft conflict. Negative weight is the possible loss in EWSA from flipping the ITP bit.

Fig. 21 illustrates the dynamic bit weighting mechanism. Set *U* contains all possible critical cell transitions for the test pattern. *S1* contains all transitioning critical cells in the ITP before adding a BP, while *S2* contains all transitioning critical cells after the modification. Set *S3* is the critical cells covered by BP. The goal is to include the most critical cell transitions in *S2*.

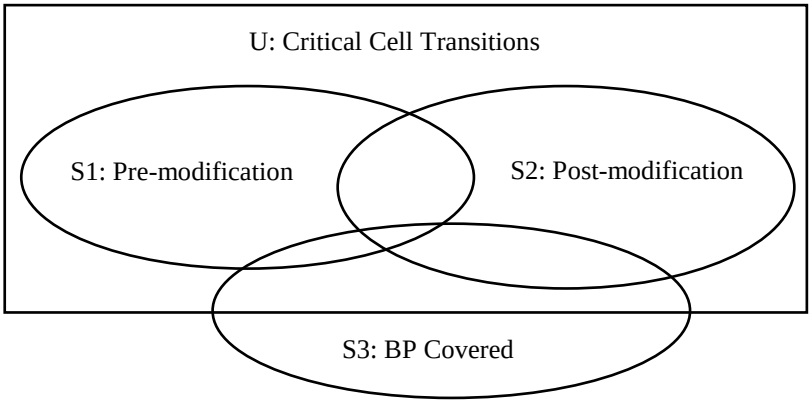


Fig. 21 iBF background patterns

Critical cell transitions present in *S1* but not in *S2* are caused by ITP flipped bits. The WSA of all such flips is termed soft loss, since they result from soft conflicts.

Transitions in *S3* but not *S2* are caused by hard or soft conflicts that prevent a flip, and are termed hard loss. Both soft and hard loss is negative weight. After each ITP update and EWSA calculation, negative weights of the conflicting bits are updated with the maximum of the old and new negative weight. Transitions included in *S2* but not in *S1* result from ITP bit flips by the BP. The WSA of this region is positive weight. The positive weight of each ITP bit is updated with the maximum of the old and new positive weight, if the ΔEWSA of the ITP update was non-negative.

3.4.6 Compression Consideration

The proposed method is compatible with test compression, such as linear-decompression-based schemes [6]. One approach is to apply Bit-relaxation and minimize the post-fill care-bit density. The impact on compression can be minimized if post-fill care-bit density is low enough. Another approach handles compression during pattern filling. The compressed pattern can be computed first which will determine the values of a certain fraction of variables. The remaining variables can be used for PSN control. Randomly fill the remaining variables and randomly flip a group of variables as in Bit-Flip. BGs-based modification can also be used, where BGs are in the form of variables. Since we flip the variables, rather than the original test pattern bits, the resulting patterns are still compressible. This will be investigated in Chapter V.

3.5 Experiment Results

The proposed BPs-based PSN control, iBF, is validated on several representative benchmarks circuits: s35932, s38417, s38584 and b19. Robust paths and patterns are generated using an in-house PKLPG tool, *CodGen*, with *K*=1 (one longest rising and

falling path per line) and 6 preamble cycles. Physical layouts are generated using a commercial placement tool. The paths that are in the top 30% in length are selected as critical paths and any test pattern that tests a critical path is considered a critical pattern. iBF is implemented in *C++* and runs on a 3.16 GHz Dell Optiplex 960 with 4 GB of memory.

Table 5 Profiles for each circuit and BPs

Circuit	Un-compacted				Compacted			
	#P	#CP	CD (%)	CC/P	#P	#CP	CD (%)	CC/P
s35932	9442	3488	0.64	1102	261	242	8.76	3863
s38417	8555	2660	4.12	1670	477	322	22.47	3646
s38584	5016	1484	2.16	1022	198	194	12.11	4066
b19	2607	1030	2.35	4096	283	232	4.48	11625

Table 6 BP profiles

Circuit	BPs				
	# CC	#BP	Avg. C	Coverage (%)	Time (s)
s35932	15381	5915	5	99.79	98
s38417	22140	3521	3	74.88	306
s38584	18218	4800	3	81.68	451
b19	56198	7191	3	63.55	4464

The profiles of both un-compacted and compacted pattern sets are summarized in Table 5, including the total number of patterns (#P), number of critical patterns (#CP), average care-bit density of critical patterns (CD (%)), and the number of critical cells per

pattern (CC/P). In Table 6, the total number of critical cells (#CC), the number of BPs (#BP), average C (Avg. C), percentage of critical cells that the BPs cause to have a rising or falling transition (Coverage (%)), and the CPU time for BP generation (Time (s)) are presented for each circuit. THLD=0 in these experiments and C is 2, 4, 8 or 16 according to the number of paths sharing the critical cell (C=2 for cells sharing up to 25% of the maximum level of sharing in the circuit, 4 for up to 50%, 8 for up to 75% and 16 for up to 100%). THLD is a PEWSA threshold to select background patterns. It is notable that the BG count is significantly smaller than the critical cell count. In order to make each BG as specific as possible, compaction is disabled during BG generation. The impact of BP compaction will be investigated in future work. Here we concentrate on comparing the performance of Bit-Flip and iBF.

For comparison, Bit-Flip and iBF with and without BG-pruning (dynamic bit weighting) was performed for each circuit. The configuration parameters for Bit-Flip are initial group size *G*, decrementing constant *D*, iterations per round *I* and rounds count *R*. For the three small circuits, the parameters are set as *G=5, D=1, I=1200,* and *R=5*. A total of 18K bits are flipped. The configuration for b19 is *G=30, D=6, I=1200,* and *R=5*. In total 108K bits are flipped.

Table 7 and Table 8 show the results on un-compacted patterns. The tables list average improvement in EWSA over random fill (ΔEWSA), transition rate (TR), average number of flips per bit (Flips/Bit) and CPU time per pattern (Time (s)). The CPU time excludes the BP generation time in Table 6. The transition rate is the average fraction of critical cells that are transitioning after filling.

Table 7 Bit-Flip on un-compacted patterns

Circuit	Bit-Flip			
	ΔEWSA (%)	TR (%)	Flips/Bit	Time (s)
s35932	70.17	72.5	10	2.16
s38417	129.78	34.35	11	1.56
s38584	166.90	28.19	12	1.24
b19	200.88	25.84	16	31.69

Table 8 iBF on un-compacted patterns

Circuit	iBF w/o BP Pruning				iBF w/ BP Pruning			
	ΔEWSA (%)	TR (%)	Flips/Bit	Time (s)	ΔEWSA (%)	TR (%)	Flips/Bit	Time (s)
s35932	70.64	72.54	5	0.68	70.43	72.46	4	0.61
s38417	125.24	33.71	7	1.4	121.30	33.06	4	0.76
s38584	198.19	30.34	7	0.73	195.68	30.05	4	0.53
b19	229.43	28.07	4	10.38	226.26	27.69	3	7.34

For s35932, iBF achieves similar ΔEWSA as Bit-Flip in less CPU time. Although the average number of flips per bit in iBF is 36% less than Bit-Flip, the CPU time cost doesn't reduce accordingly to obtain similar ΔEWSA on s38417. This is due to the fact that some BG conflicts have a large impact on PEWSA, and the high fan-out of these bits increases the incremental simulation cost of updating the PEWSA. This demonstrates the necessity for dynamic bit weighting and BG-pruning. With dynamic bit weighting, the important bits will get a high negative weight, which reduces the

possibility of using a BG that flips such bits. For the other two circuits, iBF achieves significantly better ΔEWSA in much less CPU time.

The results in Table 8 demonstrate the effectiveness of dynamic bit weighting. The technique reduced the CPU time while maintaining a higher ΔEWSA than Bit-Flip on circuit s35932, s38584 and b19. The CPU time for circuit s38417 is reduced by 51% with a 7% ΔEWSA loss, compared to Bit-Flip. The slight ΔEWSA loss is due to the difference between potential and actual EWSA change with each BP application.

Table 9 Bit-Flip on compacted patterns

Circuit	Bit-Flip			
	ΔEWSA (%)	TR (%)	Flips/Bit	Time (s)
s35932	63.14	70.55	10	2.68
s38417	75.25	35.22	11	1.42
s38584	115.36	25.25	12	1.72
b19	233.45	23.09	16	40.81

Table 10 iBF on compacted patterns

Circuit	iBF w/o BP Pruning				iBF w/ BP Pruning			
	ΔEWSA (%)	TR (%)	Flips/Bit	Time (s)	ΔEWSA (%)	TR (%)	Flips/Bit	Time (s)
s35932	63.57	70.39	7	1.22	63.51	70.36	5	0.91
s38417	74.76	35.12	5	0.84	73.71	34.95	3	0.58
s38584	126.37	25.95	8	1.14	123.48	25.67	5	0.79
b19	256.57	24.59	7	25.68	253.54	24.26	4	17.88

Paths tested by a compacted pattern are identified using breadth-first search. The goal of iBF is to have as many transitioning critical cells as possible for all the critical paths in the pattern. Experimental results for compacted patterns are shown in Table 9 and Table 10. Compared with un-compacted patterns in terms of transition rate, three of the four circuits, except s38417, have a smaller transition rate due to the higher care-bit density, which places more constraints on PSN control. In most cases, iBF can outperform Bit-Flip in less CPU time.

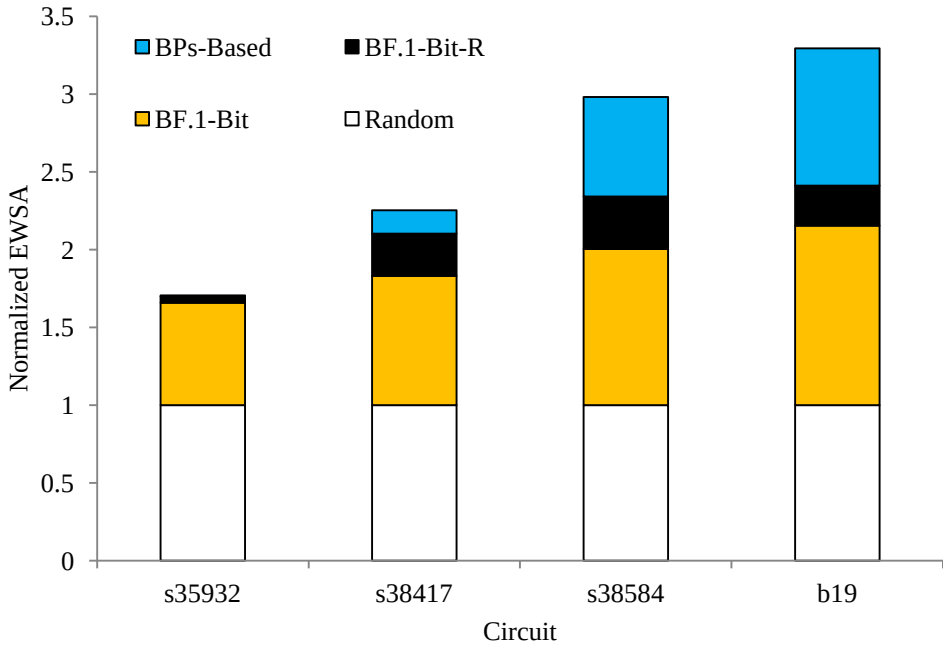


Fig. 22 Normalized EWSA at each stage (un-compacted)

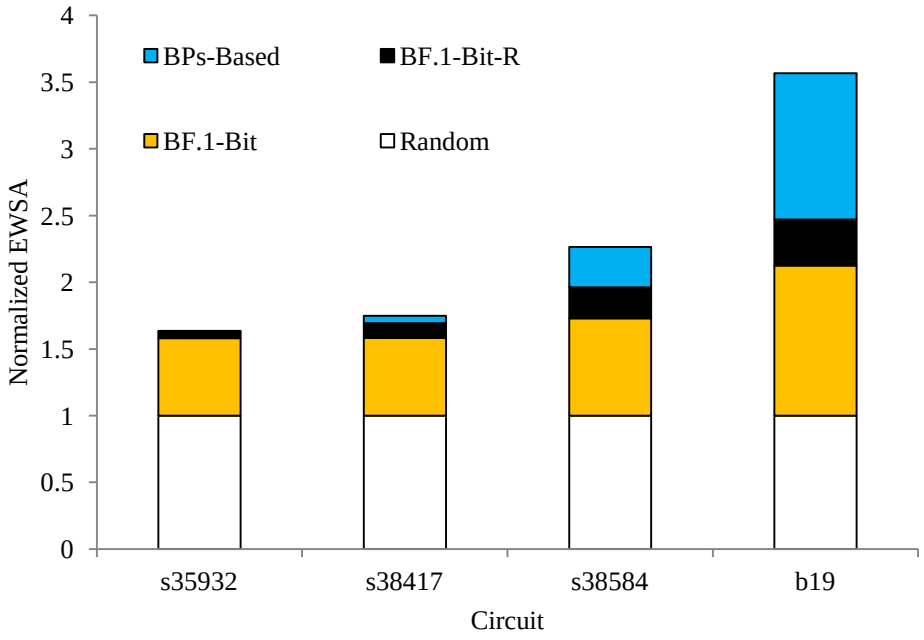


Fig. 23 Normalized EWSA at each stage (compacted)

In Fig. 22 and Fig. 23, the normalized EWSA for both un-compacted and compacted patterns at each stage of the iBF algorithm are shown. The first pass of Bit-Flip with single bit group size (BF.1-Bit) can dramatically improve the EWSA by sensitizing most of the easily-controllable critical cells. The second pass (BF.1-Bit-R) is not as effective and very limited EWSA improvements are observed. This indicates that EWSA improvement for Bit-Flip has leveled off. Instead of applying a large number of random flips, as in Bit-Flip, BP-based modification is utilized in iBF and effectively sensitizes the remaining possible transitions for s38584 and b19. BPs does not provide much improvement on the other two circuits. This is because most of the critical cells are relatively easily controlled and covered by the two-pass Bit-Flip.

The CPU times in Table 8 and Table 10 do not include the BP generation time in Table 6. If the BP generation time is included in the analysis, then iBF in aggregate takes 61% less time than Bit-Flip on un-compacted patterns, and 8% less on compacted patterns. Bit-Flip time mostly depends on the number of critical patterns, while iBF depends on the number of critical cells, so test sets with many critical patterns, such as s35932, will favor iBF.

3.6 Summary

We presented an improved Bit-Flip algorithm, iBF, to maximize functionally realistic supply noise in path delay test. It combines random flipping with background patterns to achieve cost-effective PSN control. Dynamic bit weighting permits intelligent BP selection. Experimental results show that iBF achieves worst-case realistic PSN in significantly less CPU time than previous techniques.

CHAPTER IV

PATTERN GENERATION FOR POST-SILICON TIMING VALIDATION*

4.1 Introduction

Power supply noise significantly impacts the timing performance of integrated circuits, and it may cause maximum operating frequency FMAX mismatch between structural at-speed test and functional test [49]. Much research has been done to generate test patterns with minimized [31] [33] [34], maximized [38] [39]or worst-case realistic PSN [40] [44]. These approaches can potentially improve delay test quality and achieve high FMAX accuracy. However, they lack the ability to provide knowledge to the design engineer for power supply noise prediction, such as the sensitivity of timing to PSN. Commercial tools [50] can support dynamic IR-drop analysis and the voltage at each node can be back-annotated for accurate circuit simulation. However, this is not an ideal solution considering the long simulation time but shortened product development cycle.

Combining the pre-silicon delay model with post-silicon timing measurements has the potential to improve the accuracy of timing analysis since the impact of real silicon variations, such as process variation and power supply noise, is naturally considered. Two typical measurement techniques are critical path monitors (CPM) [51]

---

*Reprinted with kind permission from "Pattern Generation for Understanding Timing Sensitivity to Power Supply Noise" by T. Zhang, Y. Gao and D. M. H. Walker, 2015. Journal of Electronic Testing: Theory and Applications, vol 31, P.P. 99-106, Copyright [2014] by Springer Science and Business Media. The final publication is available at Springer via http://dx.doi.org/10.1007/s10836-014-5502-4

and ring oscillators (ROs) [52]. They are able to capture the post-silicon variations, including process variation, temperature and voltage. These techniques can have high area overhead, especially for large circuits where large numbers of CPMs and ROs are needed. Another technique to capture the post-silicon process variation is gate-level timing extraction [53]. A novel path selection algorithm is used to generate paths and obtain an accurate variation distribution with no hardware overhead. However, power supply noise is not considered.

In this chapter, we address the problem of automatic test pattern generation for extracting circuit timing sensitivity to power supply noise during post-silicon validation. Test patterns targeting the *K* longest paths through each gate are first generated. Then a layout-aware path selection algorithm is implemented to select long paths, which fully span the power delivery network. Finally, the selected patterns are intelligently filled to bring the PSN to a desired level. These patterns can be used to understand timing sensitivity in post-silicon validation by repeatedly applying the path delay test while sweeping the PSN experienced by the path from low to high. This work extends our Bit-Flip algorithm by including layout-aware path selection and detailed PSN control analysis at the on-path gate level.

The flow of pattern generation for post-silicon validation is shown in Fig. 24. The validation consists of four parts: pseudo functional test generation, layout-ware path selection, post-ATPG processing for PSN, and sensitivity measurement. In the following sections, we will focus on the second and third parts.

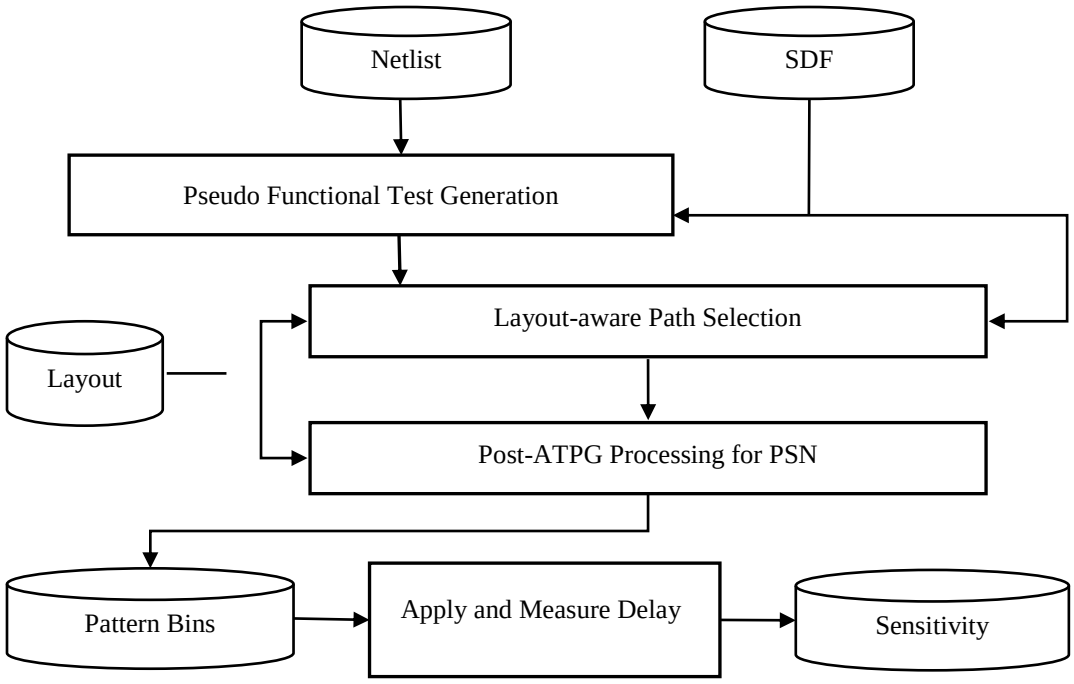


Fig. 24 Pattern generation and post-silicon validation

4.2 Pseudo Functional Test Generation

To understand timing sensitivity during functional operation, the structurally generated test patterns should mimic the functional PSN environment. The optimal solution is using functional patterns, but automatic generation of functional path delay tests is currently infeasible. Here we use PKLPG pattern generation engine *CodGen*.

PKLPG is applied by scanning in a pattern, clocking the circuit with several medium-speed functional cycles (termed preamble cycles), launching the at-speed test and then scanning out the response. The preamble cycles ramp up off-chip inductor currents and minimize *dI/dt* noise. They also filter out most non-functional state

transitions, so the at-speed test can be viewed as functional or close-to-functional. Correspondingly, the on-chip IR-drop is similar to functional operation.

4.3 Layout-Aware Path Selection

In order to obtain the sensitivity information, layout should be considered during the path generation process. The goal is to select paths so that the power delivery network can be characterized. Actually PKLPG can be viewed as a layout-aware path generation engine since it regards each gate as a fault site, i.e. it targets every node connected to the power delivery network.

However, it is not necessary to select every path of the circuit for sensitivity analysis. If all paths through a gate have large timing slack, PSN-induced delay can never cause these paths to fail. For that small region where the gate is located, we do not need to extract the timing sensitivity information. In contrast, in regions where low-slack (long) paths are clustered, enough paths must be generated to characterize the power delivery network.

Therefore, we can divide the circuit into small regions and weight each region using static delay information. For regions containing gates with less timing slack, we assign large K to the gates. For other regions, we may have smaller K or do not target those gates. Here, a region is a small area in the layout, inside which the sensitivity information of each gate is identical. Gates inside a region will have the same K since any one of the gates can be used to characterize the region. In this paper, we assume $K = 1$ (one rising and one falling path through the fault site) is good enough for PSN characterization.

After the paths are generated, a group of paths are selected for characterization. There are two basic rules to select a path: (1) the length of the path is among the top 30%; (2) the path covers regions not covered by previously chosen paths. Each region spans only one row in the layout and it has a much smaller range compared to the critical range in. The finer granularity guarantees the assumption that gates inside each region have similar sensitivity characteristic. In Fig. 25, an example layout is given and divided into 3x3 grids. Each region is indexed with X and Y axis value. Y value is simply the row number in the layout. X value stands for the index of the power/ground segment.

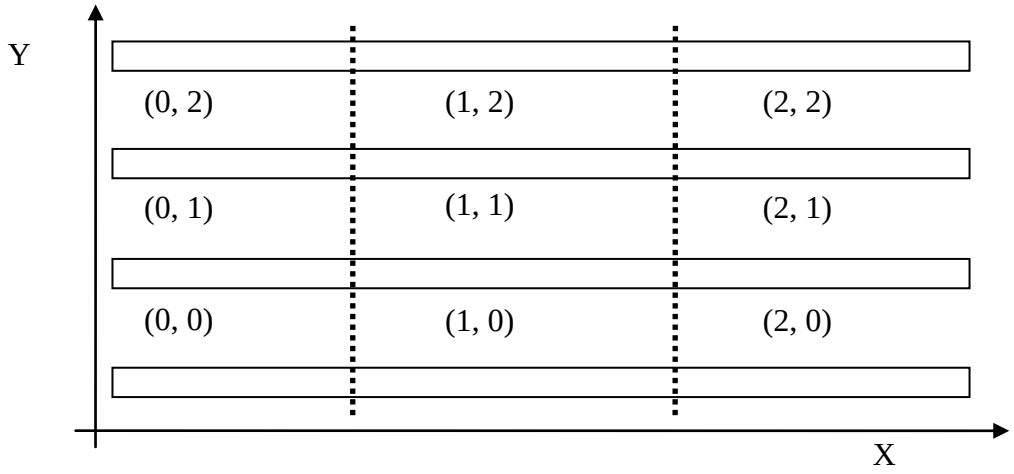


Fig. 25 An example of 3x3 layout regions

Theoretically, we only need one path to cover each region we care about. In the experiments, each region is allowed to be covered up to 10 times to achieve higher accuracy. The path selection algorithm is shown in Fig. 26.

The complexity of the path selection algorithm is linear in the size of the circuit. Since we look at the *K* longest path through each gate, the number of paths in top 30% is

linear in the circuit size. For each long path, the algorithm iterates on each on-path gates, the number of which is usually a constant. Therefore, the complexity is *O(K L N)*, where *L* is the length of the longest testable path and *N* is the circuit size.

Algorithm: Path-Selection

1) Read in circuit and layout
2) Divide layout to regions and assign region ID to gates
3) Read in all PKLPG paths and rank them by length
4) ***For*** each region (X_i, Y_i)
   a) Initialize number of paths covering it as Cover (X_i, Y_i) = 0
5) ***For*** each path P_n among the top 30%
   a) *Cover_New_Region* = 0
   b) ***For*** each on-path gate G_i with region ID (X_i, Y_i)
      i. ***If*** Cover (X_i, Y_i) < 10 ***and*** not covered by P_n
         Label the region covered by P_n
         Cover_New_Region ++;
   c) ***If*** *Cover_New_Region* > 0
      Add P_n to the selected path list

Fig. 26 Path selection algorithm

4.4 Post-ATPG Processing for PSN

Bit-Flip is a simulation-based PSN control algorithm [44]. It starts by fetching a test pattern and the corresponding path. Critical cells, which are neighboring-row gates located within a certain distance (critical range) of on-path gates, are identified by layout analysis. X-bits are initially randomly filled and the effective weighted switching activity computed as the initial $EWSA_{max}$. EWSA is the sum of the weighted switching activity of transitioning critical cells. Then Bit-Flip enters an iterative PSN control step and randomly flips a group of bits to improve the EWSA. In each iteration, up to *G*

don’t-care bits are randomly selected and their values flipped. Incremental logic simulation is used to compute the change in EWSA. The flip is retained if EWSA does not decrease. To reduce CPU time and direct the search, the algorithm starts with large group size *G* and then gradually shrinks *G* during the iterations.

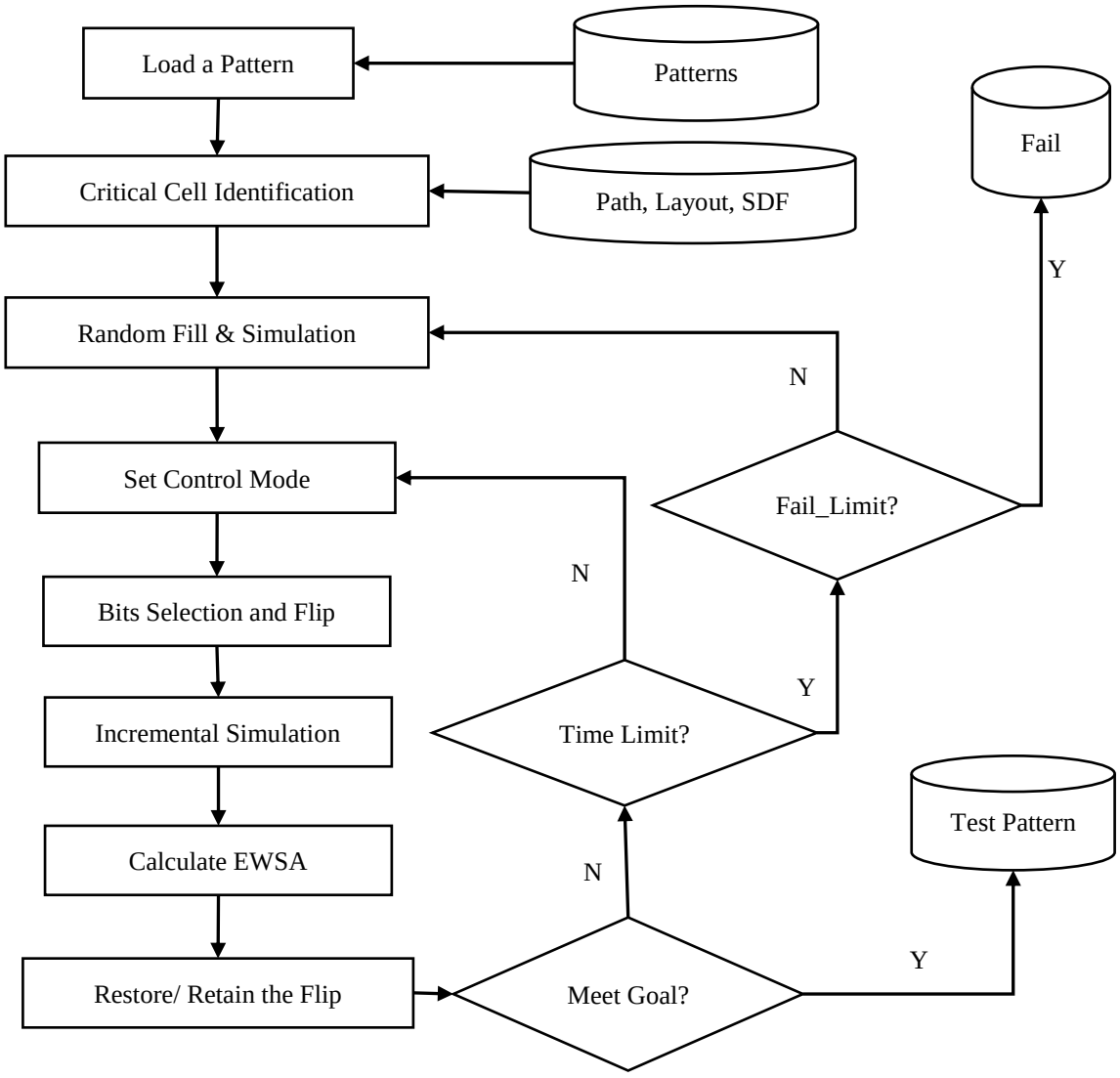


Fig. 27 Post-ATPG processing flow

iBF [45] combines random flipping with deterministic modification to achieve similar PSN level in less CPU time. It first uses random flipping to cover most easily controlled critical cells. Then it uses background patterns as a guide to modify the test pattern in order to cover random-resistant cells. The idea is similar to a typical ATPG flow. Experimental results on benchmark circuits have demonstrated the effectiveness of these two methods.

In this chapter, we incorporate Bit-Flip and iBF to support pattern generation for post-silicon validation as shown in Fig. 27. The scheme can be used to increase or reduce EWSA in order to meet a specified PSN goal. This permits generating a family of patterns with different PSN levels. The control mode is set based on the current PSN level vs. the goal. If smaller, MAX mode is used to increase PSN. Otherwise, MIN mode reduces switching activity. The tool switches between Bit-Flip and iBF whenever 50 iterations in a row fail to increase PSN. If it fails to bring the PSN to the goal level within 10,000 total iterations, it will restart with a new randomly-filled pattern. The number of restarts is also limited. If the target PSN cannot be achieved after enough attempts, a failure will be reported.

4.5 PSN Estimation

Dynamic IR-drop analysis is accurate but usually requires time-consuming power-grid simulation, making it too expensive for integration into an ATPG flow. Although EWSA lacks accuracy in PSN estimation, it is widely used to approximate PSN due to its low computation complexity. Many approaches have been proposed to

enhance the accuracy of WSA-based PSN estimation by incorporating either spatial or temporal information or both [54].

Simulation results in [39] demonstrate that local transitions have a relatively large impact on the voltage level of the on-path gate. They proposed to generate as many transitions on neighboring signal lines as possible in order to maximize PSN for a given path. Bit-Flip and iBF also used a similar model to select critical cells. Transition time differences between the critical cells and on-path gate is another important factor that should be considered. In the following, we will investigate the PSN impact of transition timing difference (T_d) and physical distance from the on-path gate to the critical cell.

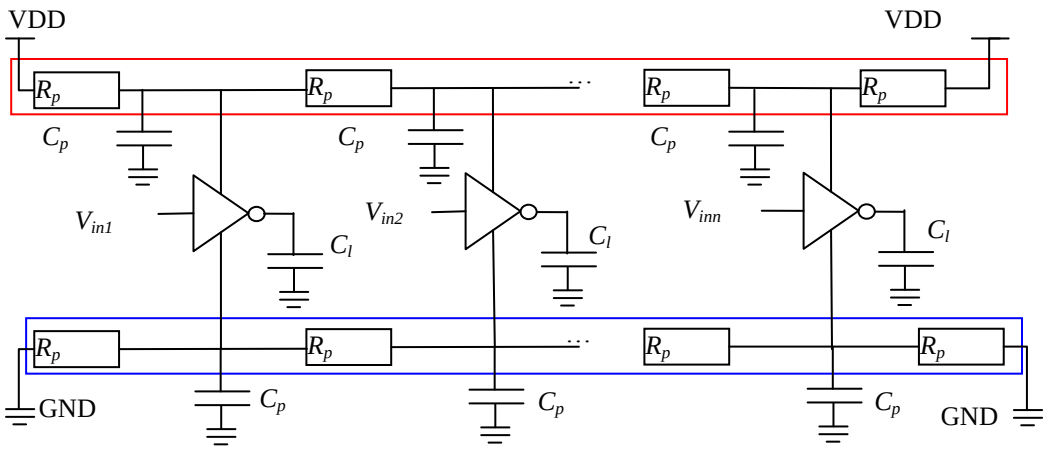
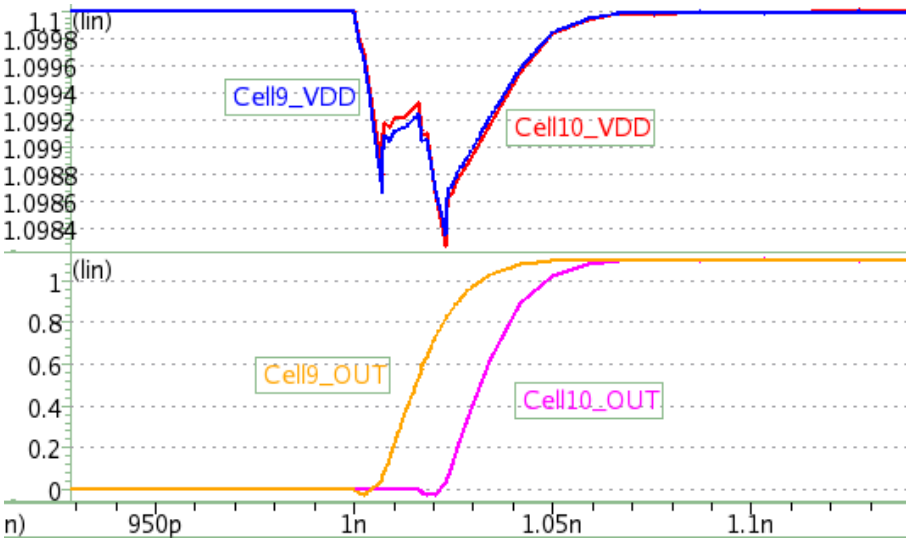


Fig. 28 Circuit to study IR-drop

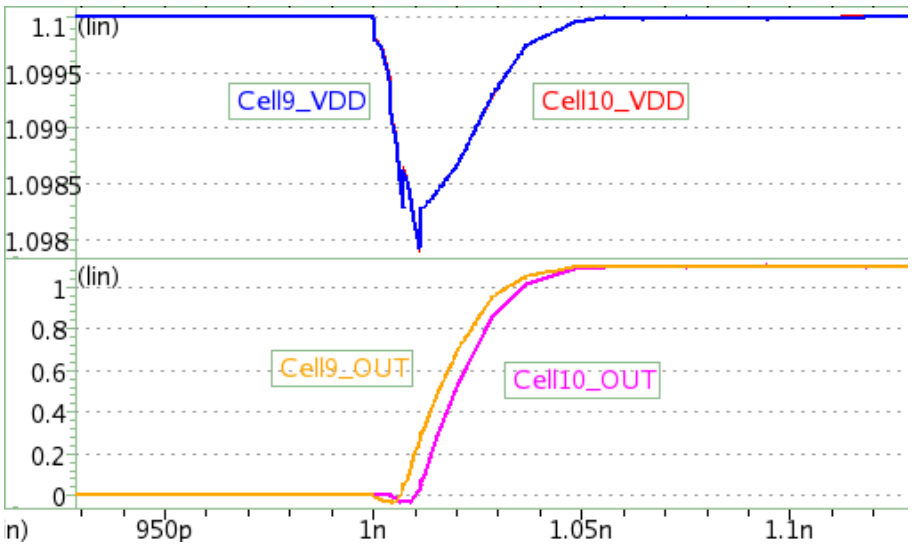
The circuit in Fig. 28 is used to study the impact of timing and location on power supply noise. The power grid line is modeled as a RC tree (R_p and C_p are parasitic resistance and capacitance respectively). In total, N independent inverters are placed in the row (N=19). Each inverter has its own input voltage source (V_{ini}) and output load (C_l).

In our experiment, ground line parasitic R_p and C_p are not used, since the focus is IR-drop on the supply. The parameters are set as C_p = 0.06fF, C_l = 0.5fF, and R_p =4Ω, based on the 45nm NanGate OpenCell Library. With this circuit, we can align the transition time of different cells and study how T_d can affect the voltage level. For simplicity, we only consider how transitions on Cell1/Cell9 can affect the voltage at Cell10 which is located in the middle of the row. $T_d = T_i - T_{10}$, where i = 1 or 9, and T_i is the transition time.

In Fig. 29(a) and Fig. 29(b), we depicted the VDD level (upper half of each figure) and the output transitions (lower half of each figure) of Cell9 and Cell10 when the Cell9 transition arrives 16ps earlier (T_d = -16ps) and 4ps earlier (T_d = -4ps) than Cell10. The difference of the minimum voltage at Cell10 is 0.4mV. In Fig. 29(a), the transition of Cell9 causes a drop at Cell10 and reaches the first trough after several picoseconds. Then the voltage starts to recover. When Cell10 begins to transition, the voltage seen by Cell10 has already ramped up. Therefore, the impact resulting from Cell9's transition is reduced. With larger T_d, the IR drop caused by the two transitions will be separated from each other (non-overlap). In Fig. 29(b), Cell10 switches while the drop caused by Cell1 has not recovered. So Cell10 will see a larger drop.



(a) IR-Drop seen at Cell9 and Cell10 when T_d = -16ps



(b) IR-Drop seen at Cell9 and Cell 10 when T_d = -4ps

Fig. 29 Voltage at Cell9 and Cell10 with different transition time

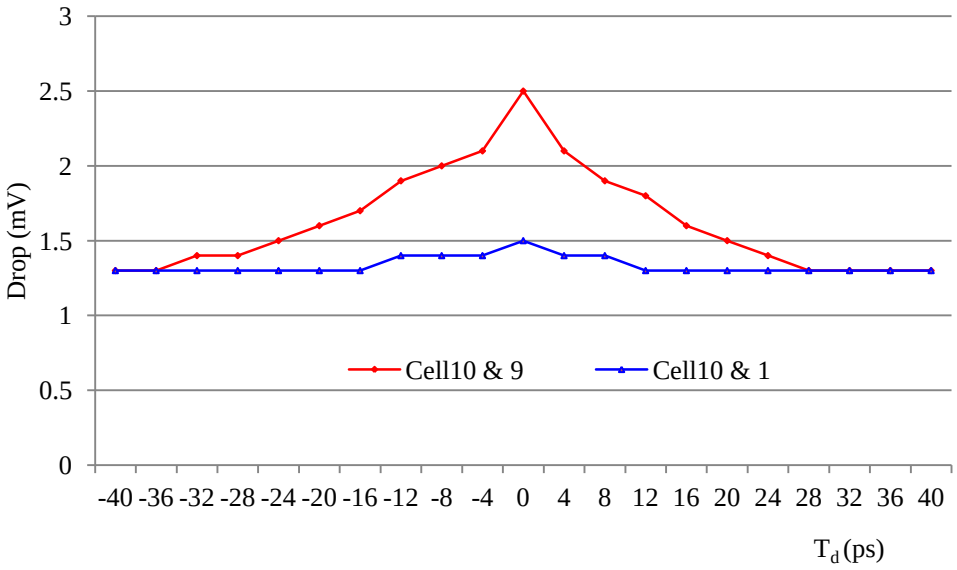


Fig. 30 Minimum voltage seen at Cell10

In Fig. 30, we swept T_d from -40ps to 40ps and plot the minimal voltage at Cell10. There are two groups of experiments: (1) Cell10 and Cell9; (2) Cell10 and Cell1. It clearly shows that for small $|T_d|$ the voltage drop is high. The difference of the MAX and MIN T_d, at which the other gate can affect the voltage of the on-path gate, is called the effective timing window (ETW). Gates located far away have a small ETW and relatively smaller impact on the IR-drop amplitude with the same T_d. In this experiment, Cell9 caused 1mV higher IR-drop than Cell1. These observations demonstrate the necessity of considering timing information in PSN control.

As in Bit-Flip, critical range is used to identify the small region around the on-path gate. Gates located in this region are critical cell candidates. The next step is to filter out the gates located outside the effective timing window. As shown in Fig. 31, we

only consider critical cell transitions arriving within the timing window as in (b). If a transition of a cell arrives too early (a) or too late (c), it is not considered a critical cell.

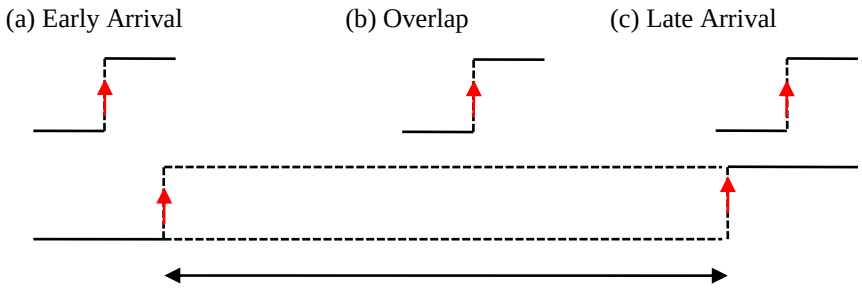


Fig. 31 Alignment of transitions on critical cell and on-path gate

For cells with transitions that fall within the timing window, a weight based on its distance to the on-path gate is assigned to quantize the significance of its impact on the on-path gate. The weight is calculated using the formulas below, termed critical WSA (CWSA). If a critical cell is critical to multiple on-path gates, the maximum weight will be assigned.

$$WSA_i = \begin{cases} 1, & if\ FO = 1 \\ FO + 1, & otherwise \end{cases}$$

$$W_i = \frac{1}{1 + D_i/W}$$

$$CWSA_i = W_i * WSA_i$$

where FO is the number of fan-outs, D_i is the distance between critical cells and on-path gate, and W is the critical range.

As discussed above, the critical cell identification and CWSA update can be done in Fig. 32. In this algorithm, the *R* is a distance from the current on-path gate horizontally (Critical range in Chapter II).

Algorithm: Critical-Cell-Identification

1) ***For*** each on-path gate G_i located at (X_i, Y_i) with delay d_i
   a) ***For*** each Gate G_j located in row Y_i
      i. ***If*** G_j located within [X_i – R, X_i + R) ***and*** d_i *fall within* [d_i – ETW/2, d_i + ETW/2)
         Calculate the new $CWSA_i$
         If new $CWSA_i$ is larger than current
            Update to new $CWSA_i$
   b) ***If*** $Y_i - 1 >= 0$
      Repeat step a) for row $Y_i - 1$
   c) ***If*** $Y_i + 1 <=$ MAX_ROW_INDEX
      Repeat step a) for row $Y_i + 1$

Fig. 32 Critical cell identification

4.6 Experimental Results

The proposed pattern generation scheme is validated on the three largest ISCAS89 benchmark circuits (s35932, s38417 and s38584), running on a 3.18 GHz Dell Optiplex 960 with 4 GB memory. The paths are generated using *CodGen* with *K =1*, launch-on-capture, one path per pattern. The number of preamble cycles is 6, i.e. 8 cycles including the launch and capture cycles. The ETW is set to ±1 gate delay, based on the results in the previous section.

First of all, in Table 11, the number of regions of each circuit, number of covered regions and the average number of critical cells without/with timing filtering are listed.

For s35932, 87.69% of the regions are covered by at least one critical path in the top 30%, while the other two only have around 25% regions covered. More than 75% of critical cells are filter out for each circuit.

Table 11 Regions and critical cells per path

Circuit	Regions (XxY)	# Covered Regions	Avg. # CC/Path		
			w/o Filtering	w/ Filtering	Filtered (%)
s35932	23x118	2380	2132	508	76.17
s38417	24x123	811	4509	910	79.82
s38584	24x122	735	2299	490	78.69

In Fig. 33 and Fig. 34, the number of paths in top 30% covering each region of s35932 and s38417 are shown, respectively. For s38417, we can see some "hot" regions which are covered by relatively large number of critical paths, while most of regions in s35932 see similar number of critical path. These two circuits can represent two different scenarios of path length distribution. In s38417, the long paths are clustered to some regions while in s35932, long paths are distributed evenly. Thus s38417 does not have enough long paths to cover each region. When the paths are selected from the top 60%, the coverage for regions can be improved as shown in Fig. 35. However, as discussed in previous section, for the regions which don't have many critical paths going through it, there is no need to characterizing the timing sensitivity since short path never fail due the delay caused by PSN. Also notice that, although we allow each region to be covered by up to 10 paths, there are regions with more than 10 paths. The reason is that paths that

cover regions that is lower than 10 go through regions that already covered by 10 times or more.

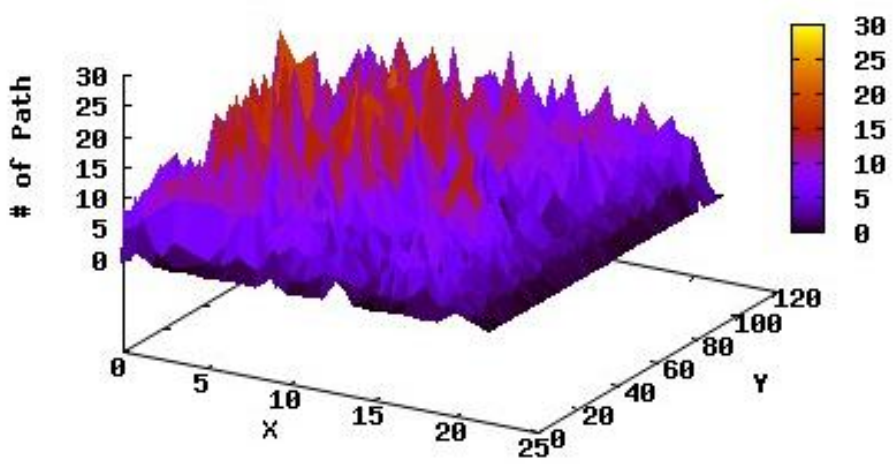


Fig. 33 Number of paths covering each region in s35932 (top 30%)

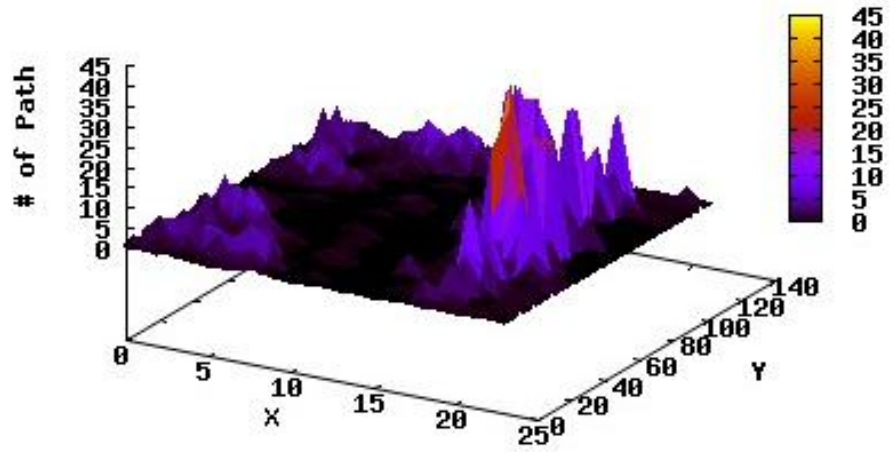


Fig. 34 Number of paths covering each region in s38417 (top 30%)

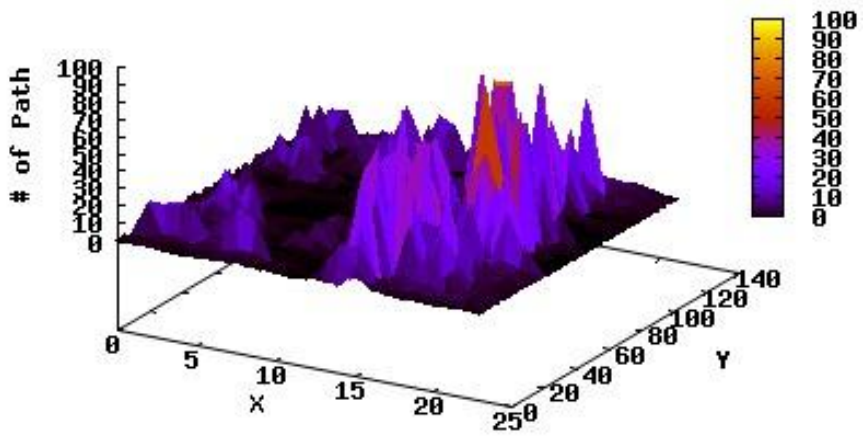


Fig. 35 Number of paths covering each region in s38417 (top 60%)

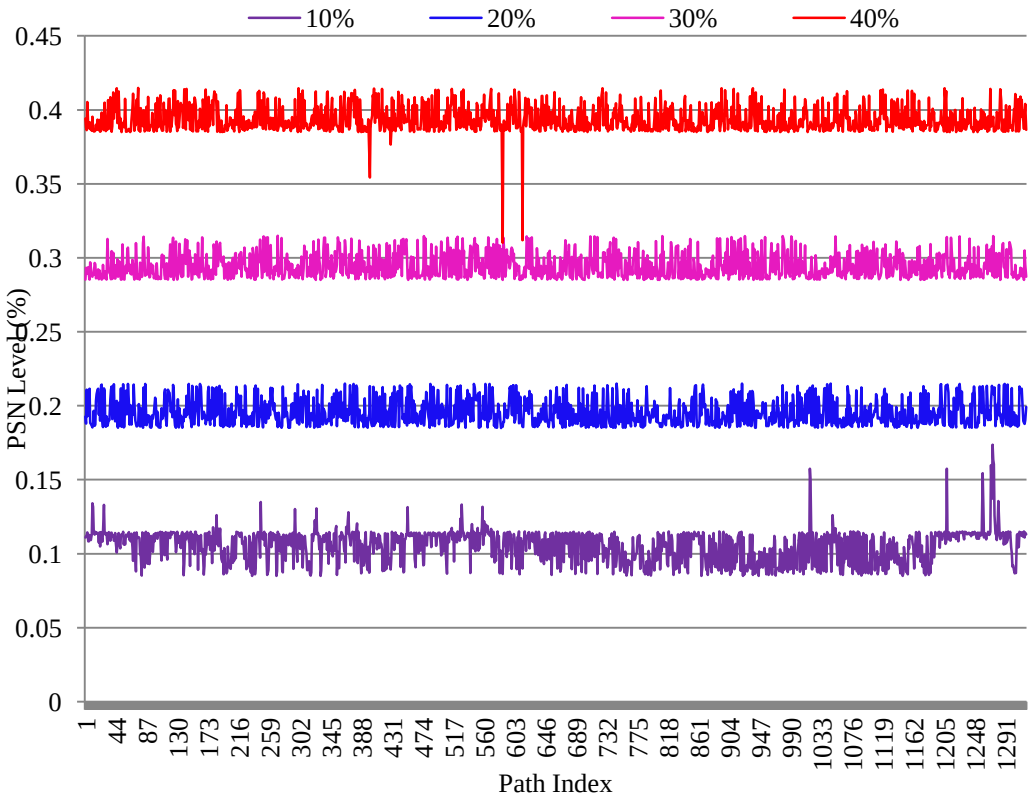


Fig. 36 s35932: Test pattern with different PSN level

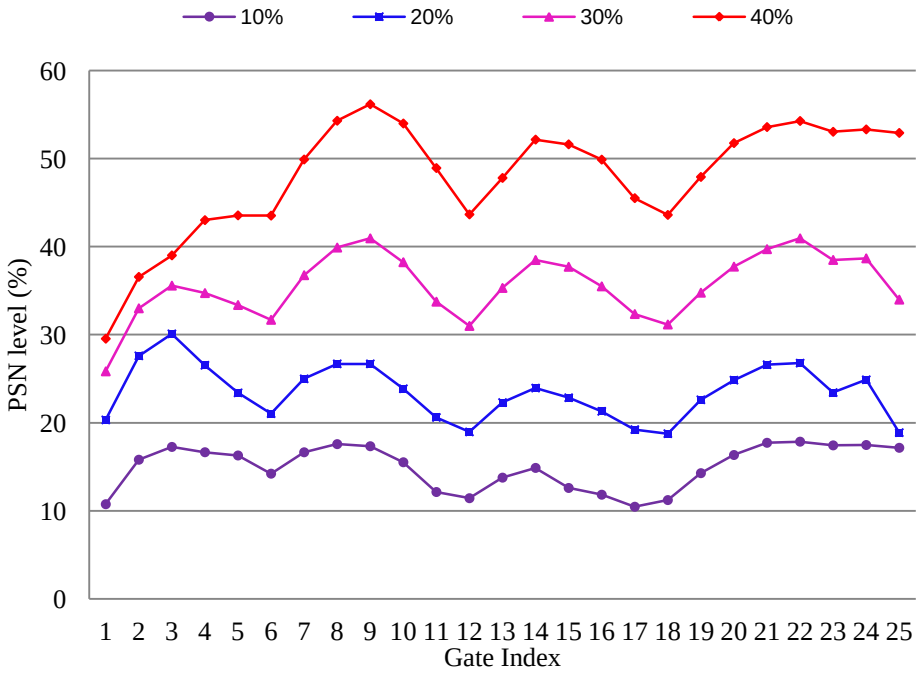


Fig. 37 PSN level of each on-path gate

The PSN level seen by each path is computed as:

$$P = \sum_i CWSA_i \Big/ \sum_j CWSA_j$$

where i is a transitioning critical cell and j is a critical cell. In Fig. 36, we present the PSN control results for s35932. The target PSN level is set to 10%, 20%, 30% and 40%. For timing sensitivity analysis, we can apply these patterns from low to high noise level and measure the FMAX. In Fig. 37, we give the average PSN level of each on-path gate, indexed by its location on the path. The on-path gate PSN level is computed similar to the critical path PSN level. For each on-path gate, we have a list of its critical cells. Therefore, after the PSN control commits, the PSN level per gate can be computed as CWSA sum of its transitioning critical cell over the CWSA sum of all of its critical cells.

We can see that for this circuit, the PSN level of each on-path gate is higher than the overall PSN level.

If the PSN achieved by the algorithm falls within ± 1.5% of the target level, we will report success. Otherwise, a fail is reported. Some of the patterns are intrinsically noisy, which means the pattern itself causes PSN higher than the upper bound. It is impossible to bring down the PSN level. The intrinsic PSN can be calculated by simulating the partially specified pattern. Such patterns are termed a noisy pattern (NP).

Table 12 PSN post processing results w/ filtering

Circuit	#P	PSN	#NP	#InitFail (#Res/#Fail)	#Suc	Time (s)
s35932	1320	10%	9	62 (25/37)	1274	1037
		20%	0	0 (0/0)	1320	143
		30%	0	0 (0/0)	1320	152
		40%	0	5(0/5)	1315	264
s38417	183	10%	178	2(0/0)	3	39
		20%	46	19(4/15)	122	212
		30%	1	12(1/11)	171	216
		40%	0	100(7/93)	90	2011
s38584	245	10%	72	22(4/18)	155	282
		20%	6	7(1/6)	233	123
		30%	0	42(6/36)	209	617
		40%	0	139(13/126)	119	2103

The detailed pattern generation results are listed in Table 12. In columns 1 and 2, we list the circuit and number of paths selected (#P). Column 3 lists the target PSN.

Column 4 lists the number of NP. In column 5, we list the number of initial failures (after the first try), the number of patterns that succeeded after restart (#Res), and the number of failed patterns (#Fail). Column 6 lists the number of successfully filled patterns (#Suc). Column 7 lists the CPU time cost for post-silicon PSN control.

The number of paths selected for characterization at most 10 times of the number of regions in the layout since we allow up to 10 covering of each regions. Here we can see that the number of selected path is much smaller, even for s35932 where up to 87.69% of its regions are covered. The possible reason is that each path will typically cover multiple regions. In Table 13, we listed the experiment results of targeting 20% PSN level without critical cells filtering. It clearly shows the benefit of critical cell filtering - reducing the CPU time cost.

As discussed in the previous section, the timing information will filter out gates that do not affect the delay of the path. To further understand how this affects the number of critical cells per gate, we depicted the average number of critical cells per on-path gate in Fig. 38. The on-path gates are indexed in the order they appear along the path. The results clearly show that the first ~20 gates have a relatively large number of critical cells. The possible reasons are that most of the transitions in the circuit happen too early to affect the voltage level. Therefore, the algorithm is dominated by the first ~20 gates. Without the filtering, the number of critical cells per gate is dependent on the critical range size. Since each gate has a similar number of critical gates, filtering prevents simulation time from being spent on critical cells that do not affect timing.

Table 13 PSN post processing results w/o filtering

Circuit	PSN	#NP	#InitFail (#Res/#Fail)	#Suc	Time (s)
s35932	20%	0	0 (0/0)	1320	364
s38417	20%	0	5(1/4)	179	190
s38584	20%	0	8(4/4)	241	316

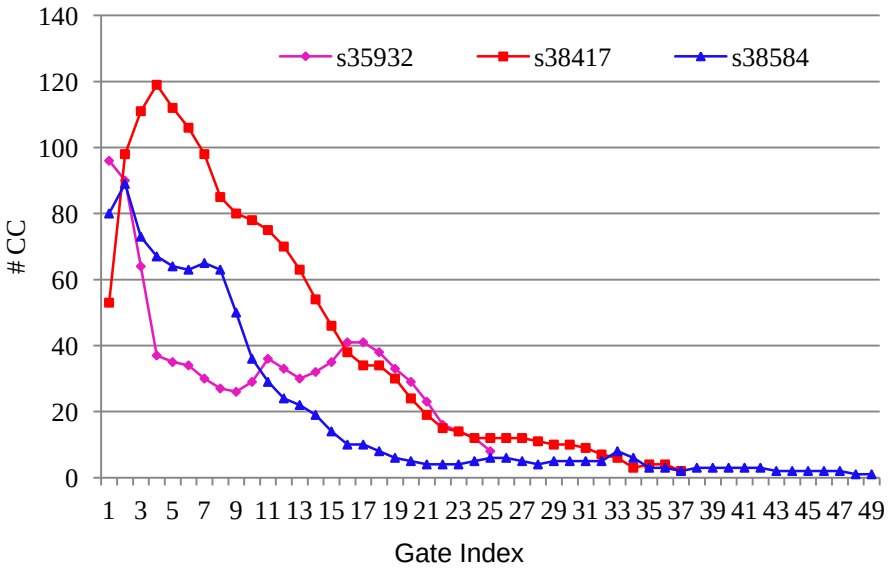


Fig. 38 Average number of critical cells per on-path gate

4.7 Summary

In this work, we addressed the problem of automatic test pattern generation for understanding the timing sensitivity to power supply noise during post-silicon validation. Experimental results demonstrate that patterns with different PSN levels can be generated.

CHAPTER V

IMPACT OF TEST COMPRESSION ON PSN CONTROL

5.1 Introduction

Reducing test data volume (TDV) is critical to reducing test time and test cost. TDV is estimated as $L\ N$, where L is the total length of scan chains and N is the number of test patterns. Usually, a large number of test patterns are required to achieve acceptable fault coverage so N is large. L is design size dependent. Two techniques, test compaction and test compression, were invented to reduce TDV [1]. Test compaction minimizes the pattern count, i.e. N. The key idea is to test as many faults in one pattern as possible. Compaction can be done after automatic test pattern generation (ATPG), such as static compaction [4], or during ATPG, such as dynamic compaction [3].

Compression minimizes the TDV transferred between automatic test equipment (ATE) and the chip under test. Various compression techniques have been reported in the literature, including test pattern encoding, Illinois Scan, and embedded deterministic test (EDT) [1] [6]. These techniques generate the test bits on chip from many fewer bits on the ATE.

These two techniques take advantage of the fact that many test pattern bits are don’t-cares (DC) [6]. Instead of randomly filling the DC bits, compression assigns the bits using compression constraints, and compaction techniques specify the bits to target new faults. A tradeoff must be made between compression and compaction in order to achieve the largest TDV reduction.

Compaction and compression create undesirable side effects. Compacted patterns usually generate much more switching activity than functional patterns, potentially causing test overkill or chip damage due to excessive power dissipation [17] [21] [22] [30] [36] [37]. Several techniques, such as low-power compaction [4] [36], have been proposed to maintain the power supply noise at an acceptable level during compaction. Pseudo functional test (PFT) [25] [41] [43] attempts to apply only tests that are functionally-reachable, so that the tests mimic the functional PSN.

For at-speed PFT delay test, the PSN control scenario is different, as PSN great affects the timing performance. In order to exercise the worst-case delay, maximization of realistic PSN is crucial [39] [41] [44] [45]. This is especially true for frequency binning and speed-limiting path debugging on high performance chips. So compaction might help find the worst-case delay as it can generate more switching activity on the chip. However, compaction also introduces constraints into the test pattern, as testing more faults usually requires more care bits, and as a result, the freedom to maximize the PSN is reduced. With PSN maximization, it is possible higher PSN can be achieved on an un-compacted pattern than on a compacted pattern, while both test a long path in the design. It is interesting to experimentally study this case. While the capture PSN is maximized, the shift power can be reduced by varying the shift frequency [42].

Differing from compaction, compression potentially reduces the PSN as it places constraints on filling the DC bits. For example, in Illinois Scan, multiple scan chains share one ATE channel and these chains have the same logic value. Compared to random fill, it likely lowers the PSN for the test pattern. Compression-aware PSN

reduction [54] [55] has been proposed and experimentally validated using two-cycle transition fault model. However, little work has been reported on maximizing PSN considering compression

One type of PSN control is intelligently filling the DC bits to maximize or minimize PSN, or generate patterns with different PSN levels for understanding the timing sensitivity to PSN [32] [33] [34] [35] [40] [55]. There are several advantages to using DC filling for PSN control. The first is its compatibility with industrial ATPG tools. DC filling comes after ATPG, so incurs no fault coverage loss. Even if all bits are specified by ATPG, bit-relaxation can be used to turn bits that are not related to coverage into DC bits, which can be used for PSN control. Moreover, these approaches do not require hardware overhead and are easy to implement with simplified low-cost PSN estimation metrics. As test compaction, compression, and PSN control make use of the fact that a large portion of test pattern bits are DC, a tradeoff usually has to be made to balance TDV against PSN. However, to the best of our knowledge, no work has been published on how compression and compaction affect PSN control. So far, we only have some intuition on how they affect each other.

In this work, we run experiments to show to what level compaction/compression can affect PSN control. We first generate pseudo functional EDT and Illinois Scan patterns that target the longest paths. Then a compression-aware PSN control scheme is implemented to maximize the PSN while obeying the compression constraints. For each path, four different patterns are used for the experiment: un-compacted Illinois Scan, compacted Illinois Scan, un-compacted EDT, and compacted EDT. With these 4 patterns,

we are able to study the PSN impact of both compaction and compression. The experimental results show that with our PSN control algorithm, EDT lowers the maximal PSN by 24.15% and Illinois Scan lowers it by 2.77% on un-compacted patterns. The maximal PSN is 22.32% and 6.94% lower on compacted patterns.

5.2 Background

In this section, we will briefly describe the related work and background.

5.2.1 Pseudo Functional Test with Dynamic Compaction

PFT [43] is a multi-cycle test to improve the PSN correlation between structural and functional test. It works by scanning in a test pattern, executing several slow functional cycles (preamble) and then applying the delay test in the last two cycles at full functional speed. A PFT test can be viewed as a short burst of functional cycles and the IR-drop that the at-speed test sees is close to functional mode. The preamble cycles also minimize the off-chip dI/dt noise.

Dynamic compaction [3] is implemented during the pattern generation process. The logic values necessary to sensitize the target path, termed necessary assignments (NAs) are justified back to the primary inputs (PIs) or pseudo primary inputs (PPIs), which are scan cell outputs, to ensure the path is a true path. Whenever a new path is generated, we check whether it can be compacted together with another path by first checking the compatibility of the NAs. In Fig. 39, Path 1 has NAs denoted by circles, and Path 2 has NAs denoted by X's. If these values do not conflict, we can attempt to justify all of the NAs together, and if a test pattern can be found, these two paths can be placed in the same test pattern. The approach in [3] achieves high levels of compaction.

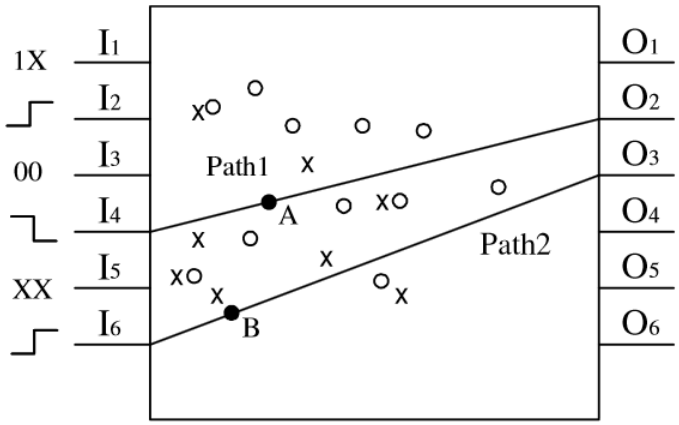


Fig. 39 Pattern compaction example [3]

5.2.2 PSN Control for PFT

While While PFT achieves PSN similar to functional mode, it is vulnerable to under-testing as switching activity dies out during the first a few functional cycles [33] [44]. PSN control algorithms, Bit-Flip [44] and improved Bit-Flip (iBF) [45] were developed to maximize PSN by setting DC bits. We make the key assumption that this maximized PSN is functionally realistic.

Bit-Flip is a simulation-based algorithm that iteratively flips DC bits. It starts by fetching a test pattern and the corresponding paths. Critical cells, which are the neighboring-row gates located within a certain distance of on-path gates, are identified by layout analysis. Switching activities on these cells have relatively large impact on the delay of the target paths. DC bits are randomly filled and the effective weighted switching activity is computed as the initial $EWSA_{max}$. EWSA is the sum of the weighted switching activity of the transitioning critical cells.

Then Bit-Flip iteratively flips groups of DC bits to maximize the EWSA. In each iteration, up to G DC bits are randomly selected and flipped. Incremental logic simulation is used to compute the change in EWSA. The flip is retained if EWSA does not decrease. To reduce CPU time and direct the search, after a certain number of iterations, group size G is reduced. A set of iterations with a fixed group size is termed a round. iBF combines random flipping with deterministic modification to achieve PSN similar to the Bit-Flip algorithm, but in less CPU time. Since Bit-Flip achieves better results than iBF (but at higher CPU cost), we extend it here to handle compression constraints.

5.2.3 Test Compression

EDT is a widely-used compression scheme shown in Fig. 40. An on-chip LFSR-style ring generator decompressor is used to decompress the test data from ATE into the test pattern. On each scan clock test variables are fed into the decompressor and produce one bit for each scan chain. A compressed vector is computed by solving a group of linear equations. The decompressor is designed to minimize the impact on fault coverage. With ring generator and phase shifter, the input variables are distributed to each state variable quickly and the logical dependency is reduced. For a decompressor with *m* ATE channels, and *n* output channels, the compression ratio can be estimated as *n/m*.

Illinois Scan, also termed broadcast scan, is another widely practiced technique for compression. The key idea is sharing the tester channels among the scan chains, i.e., one ATE channel drives multiple chains. The TDV is reduced as a group of chains share the same value. An example is shown in Fig. 41, in which the ATE data is only 1/3 of

the original test data. The scan chain constraints might reduce fault coverage. This can be avoided by connecting the chains in serial mode to target the remaining faults. Rather than a fixed connection between ATE channel and scan chains, the connection can be programmable using multiplexers, which provides potentially higher compression ratios. EDT and Illinois Scan can be mixed, with one output channel of the decompressor drives several short chains, to further increase compression.

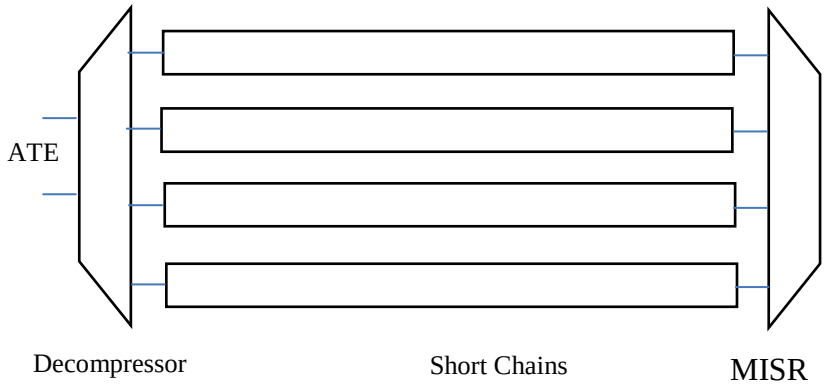


Fig. 40 EDT scheme

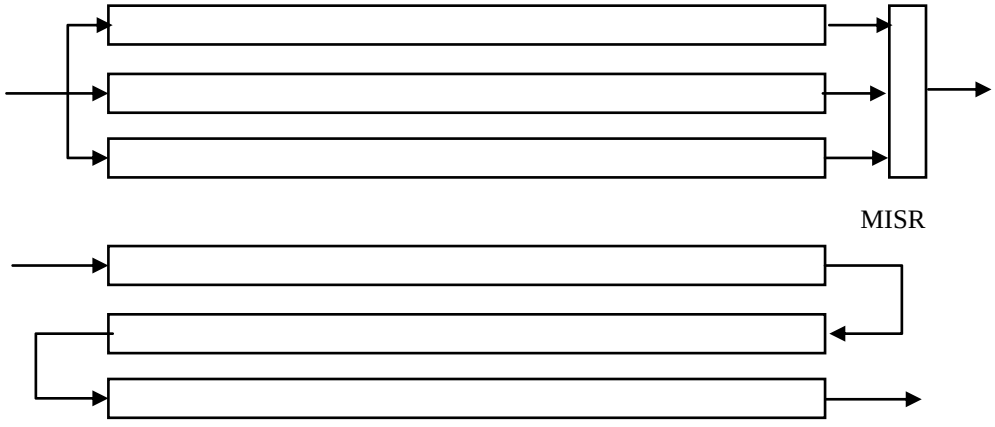


Fig. 41 Illinois Scan: parallel mode and serial mode

5.3 Pattern Generation with PSN Control

5.3.1 Compressible Pattern Generation

Our Our in-house ATPG tool, CodGen [18], is used to generate path delay test patterns and targets the K longest paths through each gate using PFT [43]. Dynamic compaction is enabled during ATPG, which can effectively reduce pattern count by a factor of 2 or more compared with static compaction. The ATPG also reports in which compacted pattern a path is tested. So we can select both un-compacted and compacted patterns for each path. If this information is unavailable, logic simulation and path searching, such as breadth-first-search, can be used to find which compacted pattern tests a path.

We enhanced CodGen to generate compressible patterns for Illinois Scan and EDT. For Illinois Scan, we need to consider the compression constraints when backward/forward implication and justification are performed. In our ATPG, we first generate a structural path, and keep the NAs to sensitize the path. After a complete path is generated, the NAs are justified recursively until all NAs can be derived from either PIs or PPIs. Implication is used for conflict detection after each assignment. When an assignment is made on a scan cell output (PPI), we need to check if the bits sharing the ATE channel have the same value. If any of the bits has a different value, a conflict is reported and we need to invert the assignment or try the next partial path. If all other bits sharing the channel are DCs, they are assigned to the same value. During justification, similar constraints are applied to the bits driven by the same channel in the same cycle to make the pattern compressible.

For EDT, the compressibility check is done by solving a group of linear equations after the path is justified. After passing justification, a partially-specified test pattern is available to test the path. For each care bit, the corresponding row in the compression matrix is used to form the linear equations. The checker performs Gaussian Elimination on the equations and checks if the equations are solvable [1] [6]. A compressibility check is also used during the compaction stage. When the NAs for a new path are compatible with an existing pattern and pass justification, the resulting new pattern is further checked for compressibility.

5.3.2 Compression Aware Supply Noise Control

Our Our previous PSN control scheme, Bit-Flip, is extended to consider Illinois Scan and EDT constraints.

In Illinois Scan, vector bits that are shifted from the same ATE channel at the same scan cycle are grouped together. These bits must always have the same logic value. If any bit in the group is specified, all the remaining bits are also care bits. A bit is DC only if all other bits sharing the ATE channel are DC. We call this a DC group. So in Illinois Scan, Bit-Flip manipulates DC groups. To achieve high PSN, we could use metrics to rank the DC groups and select the DC group with the highest potential to increase PSN. We considered fan-out tracing and switching probability calculation, but these approaches did not outperform random selection with the same simulation time cost.

As detailed in Fig. 42, we load a pattern and randomly fill the DC groups. The pattern here is already the compressed pattern. We run logic simulation on the filled test

pattern and calculate the initial EWSA. During the PSN control procedure, a DC group is randomly selected and flipped. Incremental simulation is performed to calculate an updated EWSA. We will restore the flipped bits if the flip reduces PSN. Otherwise, the flip is retained and maximal EWSA is updated. This process continues until the PSN improvement levels off.

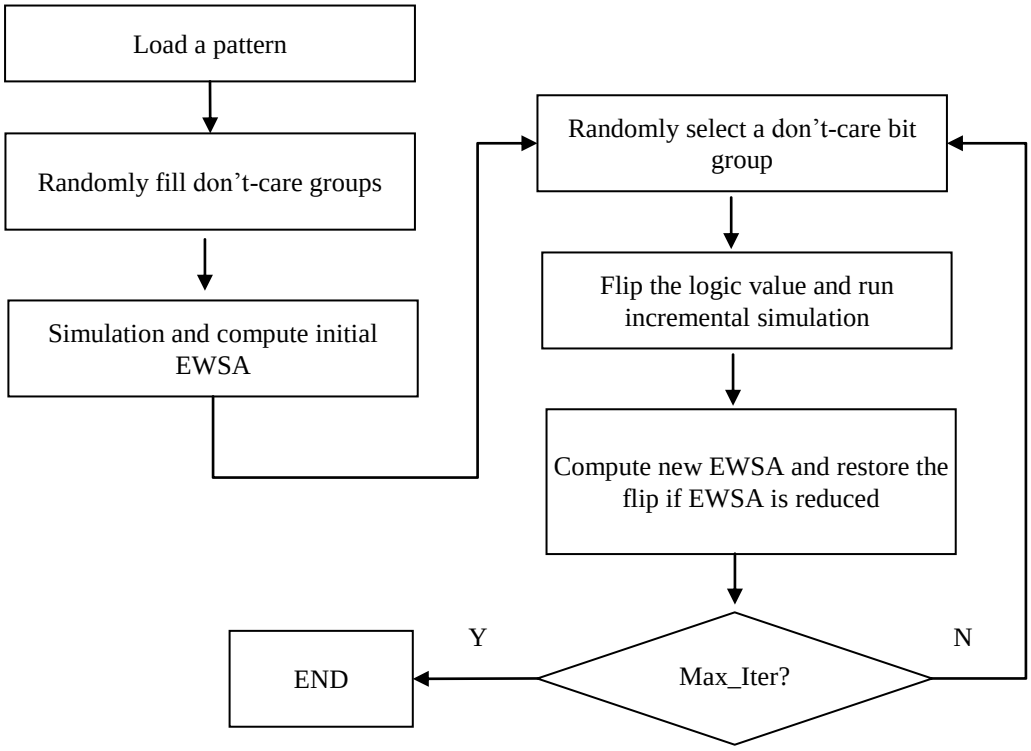


Fig. 42 PSN control with Illinois Scan constraints

PSN control with EDT constraints works differently than Illinois Scan, with the flow detailed in Fig. 43. The EDT patterns are not in compressed form, but are compressible. Therefore, we need to calculate the initial compressed vector and

randomly assign the variables that are not implied by others. From the compressed vector, an initial test pattern can be calculated. We run logic simulation and calculate the initial EWSA. Then the PSN control starts on this initial compressed vector. We first store the current compressed vector as a background vector and then randomly select one variable that is not implied by the others. We invert that variable's logic value and a new compressed vector is calculated. During the calculation, if one variable is not constrained by the flipped variable, it is assigned with the same value as the background vector. We calculate a new test pattern from the compressed vector and the difference between the new and original pattern is simulated and PSN calculated. If the flip does not reduce the PSN, the flip is retained. Otherwise, the flip is restored. If the flip is retained, the background vector and $EWSA_{max}$ is updated. The flipping process is repeated until $EWSA_{max}$ levels off. Depending on the selection of the decompressor, the flipping of one variable in the compressed pattern is likely to cause other variables to flip, and further causes hundreds of test pattern bits to flip, resulting in high simulation time.

As reported in [44], simulation time can be reduced by gradually reduced the group size G with the solution becoming more optimal. In Illinois Scan, the number of bits in each DC group is fixed and *G* represents the number of DC groups selected. G is used to trade optimality with simulation time – a larger reduction in G per round saves CPU time, but reduces the search space of the algorithm and the optimality of the result. However, in EDT, it is difficult to control the number of bits to flip, as the test vector bits are highly correlated. We have developed a constrained random (CR) approach to attempt to control the number of bit flips, reducing CPU time, at the cost of optimality.

We attempt to control the number of bit flips by tracing the history of each variable – how many bits flip when flipping that variable. This is only an approximation as the number of bits changes from pattern to pattern. This information is only used as a reference when selecting the variable.

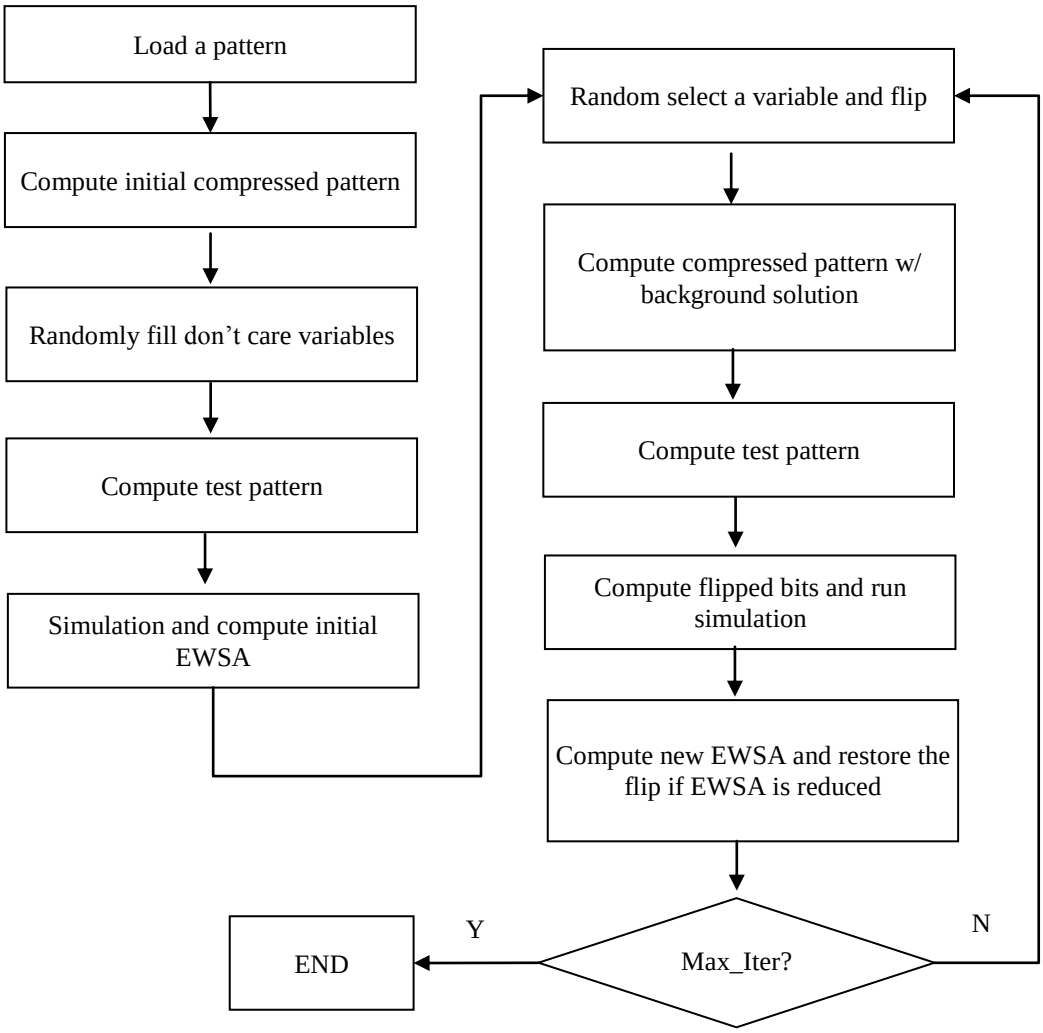


Fig. 43 EDT PSN control

The total number of flips is divided into several rounds, and each round targets a different group size G. The first round targets the largest group size while the last targets the smallest group size. Before starting the PSN control algorithm, the range between maximal and minimal number of bits flipped per variable from previous patterns is divided with respect to the number of rounds. If we have 6 rounds with the largest group size 100 and the smallest group size 40, we will target the group size for each round starting with 100-90, then 90-80, until 50-40. A variable is first randomly selected and its history checked to see if the number of bits to be flipped falls in the targeted range. This helps control the optimization process and reduce simulation time. It is possible no variable meets the requirement. In this case, random selection is used. For the first pattern processed, we initialize all variables to a history of 0 bits flipped, i.e. we use random selection for the first pattern.

5.4 Experimental Results

The largest ITC99 benchmark circuit, b19, is selected for experiments. Un-compacted and compacted PFT test patterns are generated with 6 preamble cycles and 2 at-speed cycles. Both un-compacted and compacted patterns are selected for paths that are in the top 30% in length. For Illinois Scan, the scan cells are formed into 68 chains, and placed into 9 groups (9 ATE channels), for a compression ratio of about 7.6. For EDT, 6 ATE channels and 68 scan chains are used, for a compression ratio of 11.3. The machine used to run the experiments is a 3.16 GHz Dell Optiplex 960 with 4GB memory. As we use a different PSN model and delay fault models than previous work [18] [19], it is difficult to make a direct comparison here.

We produce four sets of test patterns: T1 (no compaction + EDT); T2 (compaction + EDT); T3 (no compaction + Illinois Scan); T4 (compaction + Illinois Scan). T1 and T2 test the same set of paths, and the only difference is whether compaction is enabled. T3 and T4 test the same set of paths, but not necessarily the same as those in T1 and T2, due to compression constraints. Our goal is to study how compression can affect the PSN control. Since simulation time is not a focus, we run the PSN control until $EWSA_{max}$ levels off, approximating the maximum PSN achievable for each pattern set. For each pattern set, two runs of PSN control are performed. In the first run, we do not consider the compression constraints, while in the second run we consider the compression constraints. These runs will be denoted as T1_1, T1_2, T2_1, T2_2, T3_1, T3_2, T4_1, and T4_2. Moreover, the results of CR PSN control for EDT patterns are also included here for comparison (denoted as T1_2_CR and T2_2_CR).

Table 14 summarizes the statistics of the paths for each pattern set. Columns 2 and 3 list the total number of paths (TP) and total number of compacted patterns (CP) testing those paths. The number of tested paths is different as there two schemes place different constraints on ATPG. The compaction ratios for EDT and Illinois Scan are 7.0 and 3.7 respectively. So this indicates the Illinois Scan constraint is not as flexible as EDT in terms of compaction. For EDT, 507 paths longest paths are selected for PSN while for Illinois Scan, 602 paths are selected (SP). The average length (Avg. L) in gates of the paths is listed in column 4. The average length of Illinois Scan is longer than EDT. This indicates some long paths are not testable due to the EDT constraints. The average care bit density (Avg. CD) for each of the test sets is also listed for un-compacted (UNC)

and compacted (COM) patterns. Note that for EDT patterns, the CD is for the patterns before compression constraints are applied. The reason is that to get the care bits of patterns after compression, all the variables must be specified. Therefore, all the test pattern bits are specified correspondingly. The CD for Illinois Scan patterns are counted after compression. We can see that the number of care bits for Illinois Scan is much higher. It could be much lower before compression. We can see that compacted patterns have higher CD.

Table 14 Pattern statistics

_	T.P./ C.P.	#S.P.	Avg. L.	Avg. C.D. (%)	
				UNC	COM
EDT	6054/867	507	37	2.68 (T1)	5.23 (T2)
Illinois	5937/1611	602	48	13.17 (T3)	15.46 (T4)

For b19, the algorithm configuration for T1_1, T2_1, T3_1, and T4_1 are 6 rounds, with initial group size G of 30 and group decrement of 5. Each round performs 1000 flips. For T1_2 and T2_2, the total number of variables flipped is set to 350. For T1_2_CR and T2_2_CR, the configuration is also 350 variables, which is further divided evenly into 7 sub-groups. For T3_2 and T4_2, the algorithm flips one DC group each time and the CPU time per path is set as 50s. These values are sufficient for EWSA to level off.

5.4.1 Results of EDT

This section presents the experimental results for EDT patterns as detailed in Table 15.

We list the average EWSA of random fill (Avg. R), the average best random (Avg. BR) and the average EWSA after PSN control (Avg. EWSA). The CPU time per pattern is also listed in the last row. The best random PSN is the maximal of the four random filled patterns in the two runs on T1 and T2. From random fill, we can see that compacted patterns have higher PSN than un-compacted patterns, which is expected. Also there is no large difference in the EWSA for random fill whether the EDT compression constraints are considered or not, as seen by comparing T1_1 vs. T1_2 or T2_1 vs. T2_2. However, a large difference exists in the optimized EWSA between compression and no compression. When the CR algorithm is used, its results are only slightly better than the best random fill (vs. BR). The noisiest pattern set is T2_1 (compaction without EDT). This has 31.7% higher noise than T2_2 (compaction and EDT). For un-compacted patterns, the difference is 36.1%. Compared with random fill (vs. R), the optimization algorithm with EDT can achieve a 20% higher PSN. However, the CR algorithm achieves results similar to best random fill (vs. BR).

We compute the impact of compression using the following formula, in which EWSAw/ and EWSAw/o are final EWSA with and without compression respectively. The *Impact* of compression on un-compacted patterns is -26.81% and the Impact on compacted patterns is -24.18%.

$$Impact = \left(EWSA_w - EWSA_{w/o}\right)/EWSA_{w/o}$$

As discussed in previous sections, in EDT each variable flip may cause hundreds of bits to flip. In the CR approach, we use history to estimate the number of bits that might be flipped for each variable and select the one that meets the desired bit flip count for that round. The average number of bits flipped per variable (Avg. FB) during the optimization process is shown in Fig. 44. We can see that the number of bits flipped gradually falls, while inside each round the number of bits flipped is similar. The number of bits flipped at a time (group size *G*) is still very large compared to the initial 30 used in our previous work. In future work, alternative decompressor designs will be considered that enable smaller group sizes. By comparing the simulation time of T1_2 vs. T1_2_CR or T2_2 vs. T2_2_CR in Table 15, we can see that the EDT CR group size control reduces simulation time, but achieves lower EWSA.

Table 15 WSA and WSA improvement

-	T1_1	T2_1	T1_2	T2_2	T1_2_CR	T2_2_CR
Avg.R	1773	1927	1757	1921	1781	1900
Avg. BR	2078	2078	2078	2078	2078	2078
EWSA	3007	3032	2209	2302	2060	2249
vs R (%)	71.97	63.42	27.06	21.50	15.44	20.14
vs BR (%)	47.78	49.52	6.86	11.33	-0.01	9.30
Avg. Time (s)	49.79	49.77	152.42	174.00	123.36	134.35

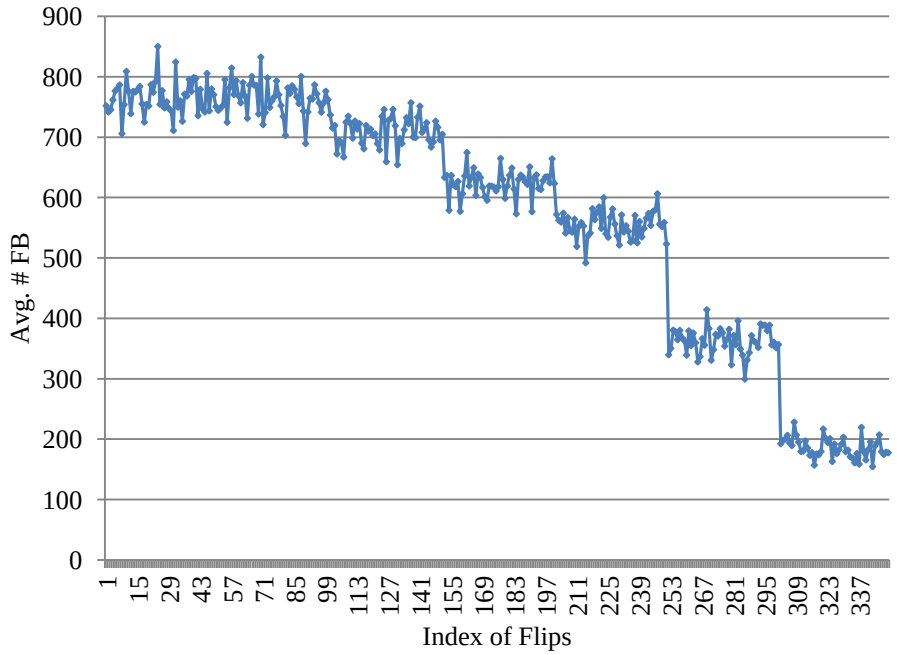


Fig. 44 Avg. number of bits flipped un-compacted EDT_CR

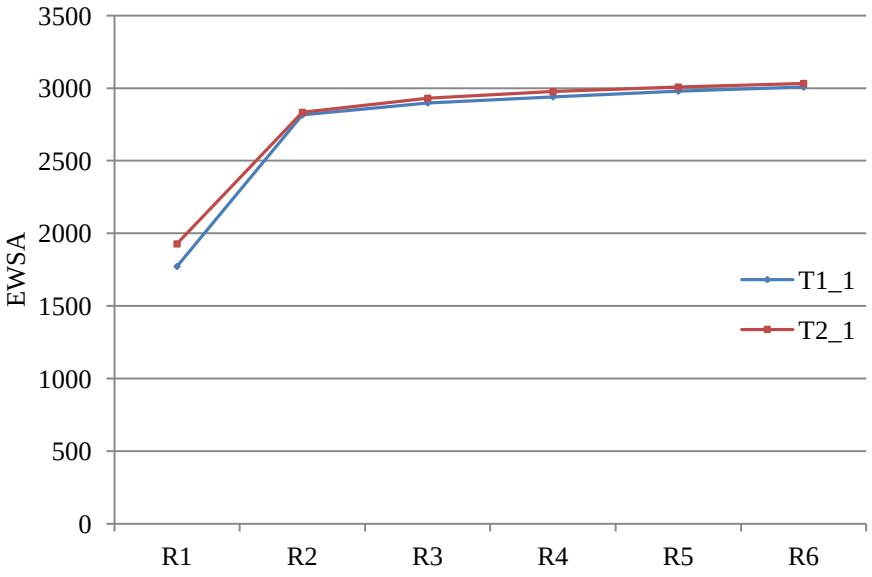


Fig. 45 PSN control without EDT constraints

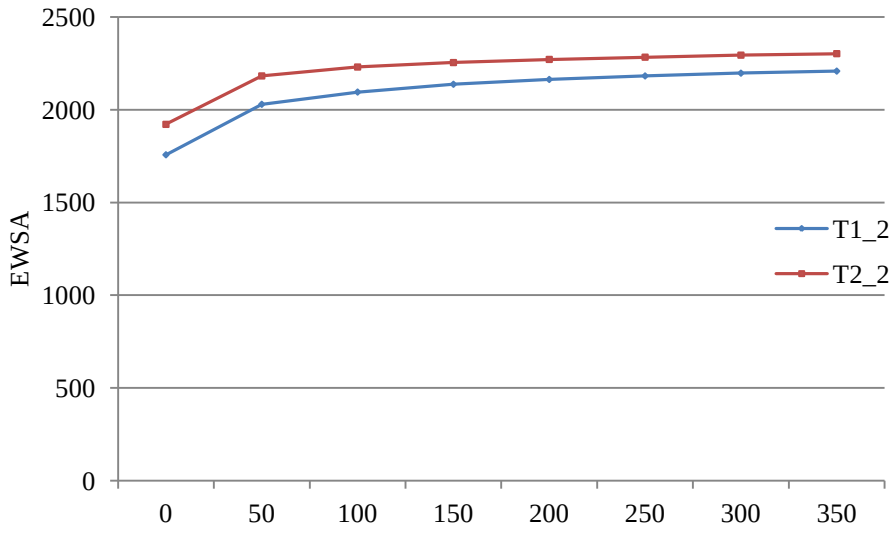


Fig. 46 PSN control with EDT constraints

Fig. 45 and Fig. 46 show the trend of EWSA increase without and with EDT constraints. Both runs show that in later rounds or when more variables have been flipped, EWSA saturates. Compaction has little impact on EWSA when EDT constraints are not present, due to the low care bit density of un-compacted and compacted patterns. With EDT constraints, compacted patterns consistently have higher EWSA, although the gap shrinks with more simulation rounds.

5.4.2 Results of Illinois Scan

The results for Illinois Scan are detailed in Table 16. Similar to EDT patterns, for T3_1 and T4_1, the PSN control does not consider compression constraints. In T3_2 and T4_2, the compression constraints are considered. For un-compacted patterns, uncompressed patterns have 3.8% more EWSA than compressed patterns. For compacted patterns, the difference is 3.4%. Under compression, un-compacted patterns

have 0.9% more EWSA than compacted patterns. This demonstrates that compaction introduces constraints that limit the PSN improvement. We did not see this in EDT patterns due to the lower care bit density of EDT patterns. The Impact on un-compacted patterns is -3.15% and the Impact on the compacted is -3.25%.

Table 16 EWSA result of Illinois Scan

-	T3_1	T4_1	T3_2	T4_2
Avg. R	3278	3392	3266	3395
Avg. BR	3626	3626	3626	3626
Avg. EWSA	4960	4902	4780	4739
vs R(%)	52.46	46.07	48.05	41.00
vs BR (%)	37.86	35.86	33.44	31.36
Avg. Time	49.75	49.74	50.03	50.03

5.4.3 Results on Other Circuits

Experiments are also performed on the three largest ISCAS89 circuits. For EDT, a 2-20-decompressor is used and for Illinois Scan, there are 2 ATE channels and 20 short chains. Each ATE channel drives 10 chains. The patterns are generated in the same way as b19 and so is the path selection. The path statistics are detailed in Table 17. The algorithm configuration for T1_1, T2_1, T3_1, and T4_1 are 6 rounds, with initial group size G of 6 and group decrement of 1. Each round performs 1000 flips. For T1_2 and T2_2, the total number of variables flipped is set to 350. For T3_2 and T4_2, the algorithm flips one DC group each time and the CPU time per path is set as 5s.

Table 17 Statistics for ISCAS circuits

Circuit	#FF	Scheme	TP/CP	SP	Avg. L
s35932	1728	EDT (20x90)	9253/635	3520	23
		Illinois Scan	9442/143	3520	22
s38417	1636	EDT (20x85)	4604/486	275	30
		Illinois Scan	7807/1075	1868	29
s38584	1426	EDT (20x75)	5013/319	41	33
		Illinois Scan	4758/244	80	26

Table 18 Results of ISCAS circuits (EDT)

Circuit	Avg. CD (%)		vs. BR (%)				Impact(%)	
	UNC	COM	T1_1	T2_1	T1_2	T2_2	UNC	COM
s35932	0.63	6.61	47.72	44.87	12.48	7.37	-25.76	-23.78
s38417	3.14	9.38	35.21	32.39	8.65	5.77	-19.65	-19.11
s38584	2.19	10.23	37.89	31.43	15.69	-1.08	-24.39	-22.2

Table 19 Results of ISCAS circuits (Illinois Scan)

Circuit	Avg. CD (%)		vs. BR (%)				Impact(%)	
	UNC	COM	T3_1	T4_1	T3_2	T4_2	UNC	COM
s35932	4.40	87.67	45.65	4.27	32.75	1.86	-2.02	-8.72
s38417	31.26	71.21	31.63	14.14	18.9	8.64	-4.3	-9
s38584	10.32	63.45	56.88	10.59	45.24	8.76	-1.6	-6.8

Results are shown in Table 18 and Table 19. The un-compacted patterns have a slightly higher EWSA than compacted patterns, as expected. Including b19, the average

Impact of EDT on un-compacted patterns is -24.15% and that on compacted patterns is -22.32%. The average Impact of Illinois Scan is -2.77% and -6.94%.

5.5 Summary

In this work, we ran experiments to show how compaction and compression can affect PSN control. We generated pseudo functional path delay test patterns and implemented a compression-aware PSN control scheme to maximize the PSN, while obeying EDT and Illinois Scan compression constraints.

Our experimental results show that our PSN control algorithm achieves significantly better results than random fill or best random fill for both Illinois Scan and EDT compression. EDT lowers maximal PSN by 24.15% and Illinois Scan lowers it by 2.77% on un-compacted patterns. The achieved maximal PSN on compacted patterns is 22.32% and 6.94% lower.

CHAPTER VI

SUMMARY AND FUTURE WORK

Testing integrated circuits to verify their operating frequency, known as delay testing, is essential to achieve high product quality. In order to limit test development costs, industry relies heavily on automatically-generated structural tests, applied by low-cost testers taking advantage of design-for-test circuits on the chip, such as scan chains, on-chip test pattern compression/decompression, built-in self-test and test access mechanisms. The ***central unsolved challenge*** in structural delay test is achieving high delay correlation with functional test. The correlation problem is dominated by ***power supply noise***. Differences in power supply noise between functional and structural tests can lead to differences in chip operating frequencies of 30% or more. Worse, it is becoming very difficult for the test engineer to know the supply noise environment on the chip, due to the use of system-on-chip designs and 3-D packaging.

The focus of this dissertation is ***achieving high correlation between structural and functional delay tests on high-performance chips***. Pseudo-functional KLPG tests create supply noise similar to functional operation by introducing preamble cycles (medium speed functional cycles before applying delay test). The preamble cycles ramp up the off-chip *dI/dt* noise and exclude illegal states in the test pattern. However, PSN profiling results from our experiments have shown that ***PKLPG is vulnerable to under-testing rather than over-testing***. Therefore, it is critical to develop PSN control algorithms to maximize functional power supply noise for PKLPG and generate test

patterns for understanding delay sensitivity to power supply noise. To achieve these goals, we proposed several test pattern generation algorithms as below.

- ***A simulation based PSN control scheme, Bit-Flip, is developed***. It is proposed to maximize the power supply noise during PKLPG test. Given a set of partially-specified scan patterns, random filling is done and then an iterative procedure is invoked to flip some of the filled bits, to increase the effective weighted switching activity. Experimental results on both compacted and un-compacted test patterns demonstrate that our method can significantly increase effective WSA while limiting the fill rate.
- Based on the observation of critical cell sharing, a scheme ***combining random flipping with deterministic modification to fill the don't-care bits is proposed***. Deterministic modification is guided by pre-computed background patterns, which sensitize transitions on critical cells. Dynamic bit weighting permits intelligent selection of background patterns. Experimental results on benchmark circuits validate the effectiveness of the techniques as worst-case realistic PSN is achieved in significantly less CPU time.
- The ***problem of automatic test pattern generation for understanding circuit timing sensitivity to power supply noise during post-silicon validation is addressed.*** . Long paths are selected from a pseudo functional test set to span the power delivery network. To determine the sensitivity of timing to on-chip noise, the patterns are intelligently filled to achieve the desired PSN level. PSN control algorithms are enhanced to consider both spatial and temporal information for

better correlation with functional PSN. These patterns can be used to understand timing sensitivity in post-silicon validation by repeatedly applying the path delay test while sweeping the PSN experienced by the path from low to high.

- The ***impact of compaction/compression on PSN control is studied***. We ran experiments to show to what level compaction/compression can affect PSN control. Pseudo functional Illinois Scan and EDT patterns that target the longest paths are generated. Then a compression-aware PSN control scheme was implemented to maximize the PSN while obeying the compression constraints. For each path, four different patterns were used for the experiment: un-compacted Illinois Scan, compacted Illinois Scan, un-compacted EDT, and compacted EDT. With these 4 patterns, we were able to study the PSN impact of both compaction and compression. The experimental results showed that our PSN algorithm achieved significantly higher PSN compared to random or best random fill on both un-compacted and compacted patterns. Our constrained random (CR) algorithm for EDT compression reduced CPU time, while achieving slightly better results than best random fill.

In future work, there are several interesting topics to explore. First, the algorithm considering compression is not CPU time cost optimized. We could further optimize the algorithm by considering the correlations between variables, so that we control group size as well as maintain the PSN level. This requires carefully selection of variables. Another efficiency improvement is by utilizing parallel pattern logic simulation. This should reduce the CPU time by more than an order of magnitude. Third, we will

incorporate additional power grid hierarchy and validate the correlation between CWSA and measured IR drop. We also plan to apply the generated patterns to real chips and measure the delay of the paths. Then we can extract sensitivity information.

REFERENCES

[1] L. T. Wang, C. W. Wu, and X. Wen, *VLSI Test Principles and Architectures: Design for Testability*.: Morgan Kaufmann Pub, 2006.

[2] T. Kirkland and M.R. Mercer, "Algorithms for automatic test pattern generation," in IEEE Design and Test of Computers, 1988, pp. 43-55.

[3] Z. Wang and D. M. H. Walker, "Dynamic compaction for high quality delay test," in IEEE VLSI Test Symp., 2008, pp. 243-248.

[4] J. Wang et al., "Static compaction of delay tests considering power supply noise," in IEEE VLSI Test Symp., 2005, pp. 235-240.

[5] I. Hamzaoglu and J. H. Patel, "Reducing Test Application Time for Full Scan Embedded Cores," in Int'l Symp. Fault-Tolerant Computing, 1999, pp. 260-267.

[6] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Embedded deterministic test," IEEE Trans. on CAD, vol. 23, no. 5, pp. 776-792, 2004.

[7] K. Angela and K.-T. Cheng, Delay Fault Testing for VLSI Circuits.: Springer Science & Business Media, 1998.

[8] M. Amodeo and B. Cory, "Defining faster-than-at-speed delay test," Nanometer Test Article, 2005.

[9] X. Lin et al., "Timing-Aware ATPG for High Quality At-speed Testing of Small Delay Defects," in Asian Test Symp., 2006, pp. 139-146.

[10] R. Kapur, J. Zejda, and T. W. Williams, "Fundamentals of timing information for test: How simple can we get?," in Int'l Test Conf., 2007, pp. 1-7.

[11] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Test-Pattern Grading and Pattern Selection for Small-Delay Defects," in IEEE VLSI Test Symp., 2008, pp. 233-239.

[12] T. J. Chakraborty and V. D. Agrawal, "Effective Path Selection for Delay Fault Testing of Sequential Circuits," in Int'l Test Conf., 1997, pp. 998-1003.

[13] H. Chang and S. S. Sapatnekar, "Statistical Timing Analysis Considering Spatial Correlations using a Single Pert-Like Traversal," in Int'l Conf.on CAD, 2003.

[14] J. J. Liou , L.-C. Wang, and K.-T. Cheng, "On theoretical and practical

considerations of path selection for delay fault testing," in Int'l Conf.on CAD, 2002, pp. 94-100.

[15] J. J. Liou, L-C Wang, A. Krstic, and K.-T. Cheng, "Experience in critical path selection for deep sub-micron delay test and timing validation," in Asia and South Pacific Design Auto. Conf., 2003, pp. 751-756.

[16] G. Bai, S. Bobba, and I. N. Hajj, "Static timing analysis including power supply noise effect on propagation delay in VLSI circuits," in Design Auto. Conf., 2001, pp. 295-300.

[17] J. Wang et al., "Modeling Power Supply Noise in Delay Testing," IEEE Design & Test of Computers, vol. 24, no. 3, pp. 226-234, 2007.

[18] W. Qiu et al., "K Longest Paths Per Gate (KLPG) Test Generation for Scan-Based Sequential Circuits," in IEEE Int'l Test Conf., 2004, pp. 223-231.

[19] J. Savir and S. Patil, "Scan-based transition test," IEEE. Trans. on CAD, vol. 12, no. 8, pp. 1232-1241, 1993.

[20] J. Savir and S. Patil, "Broad-side delay test," IEEE Trans. on CAD, vol. 13, no. 8, pp. 1057-1064, 1994.

[21] J. Saxena et al., "A case study of IR-drop in structured at-speed testing," in IEEE Int'l Test Conf., 2003, pp. 1098-1104.

[22] P. Pant and J. Zelman, "Understanding power supply droop during at-speed scan testing," in IEEE VLSI Test Symp., 2009, pp. 227-232.

[23] P. Pant and E. Skeels, "Hardware hooks for transition scan characterization," in IEEE Int'l Test Conf., 2011, pp. 1-8.

[24] M. Chen and A. Orailoglu, "Examining Timing Path Robustness Under Wide-Bandwidth Power Supply Noise Through Multi-Functional-Cycle Delay Test," IEEE Trans. on VLSI, vol. 22, no. 4, pp. 734-746, 2014.

[25] Y. C. Lin, F. Lu, and K. T. Cheng, "Pseudo functional testing," IEEE Trans. on CAD, vol. 25, pp. 1535-1546., 2006.

[26] F. Yuan and Q. Xu, "On systematic illegal state identification for pseudo-functional testing," in Design Auto. Conf., 2009, pp. 702-707.

[27] L. Whetsel, "Adapting scan architectures for low power operation," in IEEE Int'l Test Conf., 2000, pp. 863-872.

[28] X. Wen et al., "Low-capture-power test generation for scan-based at-speed testing," in IEEE Int'l Test Conf., 2005, pp. 1019-1028.

[29] I. Pomeranz , "On the Switching Activity and Static Test Compaction of Multicycle Scan-Based Tests," IEEE Trans. on Computer , vol. 61, no. 8, pp. 1179-1188, 2012.

[30] H. Liu, H. Li, Y. Hu, and X. Li, "A scan-based delay test method for reduction of overtesting," in Int'l Symp. on Electronic Design, Test and Applications, 2008, pp. 521-526.

[31] N. Badereddine et al., "Minimizing peak power consumption during scan testing: Test pattern modification with X filling heuristics," in Int'l Conf. Design and Test of Integrated Systems in Nano. Technology, 2006, pp. 359-364.

[32] S. Remersaro et al., "Preferred fill: A scalable method to reduce capture power for scan based designs," in IEEE Int'l Test Conf., 2006, pp. 1-10.

[33] E. K. Moghaddam, J.S. Rajski, M. Reddy, and M. Kassab, "At-speed scan test with low switching activity," in IEEE VLSI Test Symp., 2010, pp. 177-182.

[34] 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 Design Auto. and Test in Europe, 2008, pp. 1184-1189.

[35] R. Sankaralingam and N. A. Touba, "Controlling peak power during scan testing," in IEEE VLSI Test Symp., 2002, pp. 153-159.

[36] Z. Jiang, Z. Wang, and D. M. H. Walker, "Levelized low cost delay test compaction considering IR-drop induced power supply noise," in VLSI Test Symp., 2011, pp. 52-57.

[37] A. Krstic, Y. M. Jiang, and K. T. Cheng, "Pattern generation for delay testing and dynamic timing analysis considering power-supply noise effects," IEEE Trans. on CAD, vol. 20, pp. 416-425, 2001.

[38] L. Fang and M. S. Hsiao, "A fast approximation algorithm for MIN-ONE SAT," in Design Auto. and Test in Europe, 2008, pp. 1087-1090.

[39] J. Ma and M. Tehranipoor, "Layout-Aware Critical Path Delay Test Under Maximum Power Supply Noise Effects," IEEE Trans. on CAD, vol. 30, no. 12, pp.

1923-1934, 2011.

[40] X. Fan, S. Reddy, and I. Pomeranz, "Max-fill: A method to generate high quality delay tests," in Int'l Symp. Design and Diagnosis of Electronic Circuits and Systems, 2011, pp. 375-380.

[41] F. Yuan, X. Liu, and Q. Xu, "Pseudo-functional testing for small delay defects considering power supply noise effects," in IEEE Int'l Conf. on CAD, 2010.

[42] J. Schulze and R. Tally, "Mitigating Voltage Droop during Scan with Variable Shift Frequency," in IEEE Int'l Test Conf., 2014, pp. 1-8.

[43] K. Bian, D. M. H. Walker, S. Khatri, and S. Lahiri, "Mixed structural-functional path delay test generation and compaction," in IEEE Int'l Symp. Defect and Fault Tolerance in VLSI and Nano. Systems, 2013.

[44] T. Zhang and D. M. H. Walker, "Power supply noise control in pseudo functional test," in IEEE VLSI Test Symp., 2013, pp. 1-6.

[45] T. Zhang and D. M. H. Walker, "Improved power supply noise control for pseudo functional test," in IEEE VLSI Test Symp., 2014, pp. 1-6.

[46] T. Zhang, Y. Gao, and D. M. H. Walker, "Pattern Generation for Understanding Timing Sensitivity to Power Supply Noise," Journal of Electronic Testing, vol. 31, no. 1, pp. 99-106, Feb 2015.

[47] W. Qiu, X. Lu, Z. Li, and D. M. H. Walker, "CodSim -- A Combined Delay Fault Simulator," in IEEE Int'l Symp. on Defect and Fault-Tolerance in VLSI Systems, 2003.

[48] A. El-Maleh and K. Al-Utaibi, "An efficient test relaxation technique for synchronous sequential circuits," IEEE. Trans.on CAD, vol. 23, pp. 933-940, 2004.

[49] S. Sde-Paz and E. Salomon , "Frequency and power correlation between at-speed scan and functional tests," in Int'l Test Conf., 2008, pp. 1-9.

[50] Cadence Design Systems, "Voltage Storm Power Verification Datasheet," 2014.

[51] S. Pei, H. Li, and X. Li, "A high-precision on-chip path delay measurement architecture," IEEE Trans. on VLSI, vol. 20, no. 9, pp. 1565-1577, 2012.

[52] X. Li, R. R. Rutenbar , and R. D. Blanton , "Virtual probe: a statistically optimal

framework for minimum-cost silicon characterization of nanoscale integrated circuits," in Int'l Conf.on CAD, 2009, pp. 433-440.

[53] X. Zhang, J. Ye, Y. Hu, and X. Li, "Capturing post-silicon variation by layout-aware path-delay testing," in Design Auto. and Test in Europe, 2013, pp. 288-291.

[54] M. F. Wu et al., "Improved weight assignment for logic switching activity during at-speed test pattern generation," in Asia and South Pacific Design Auto. Conf., 2010.

[55] J. L. Huang, X. Wen, K. Miyase M. F. Wu, "Reducing Power Supply Noise in Linear-Decompressor-Based Test Data Compression," in Int'l Test Conf., 2008, pp. 1-10.

[56] X. Liu and Q. Xu, "On X-Variable Filling and Flipping for Capture Power Reduction in Linear decompressor based compression environment," IEEE Trans. on CAD, vol. 31, no. 11, pp. 1743-1753, 2012.