

Power Supply Noise Control in Pseudo Functional Test

Tengteng Zhang Duncan M. (Hank) Walker

Department of Computer Science and Engineering

Texas A&M University

College Station, US

{tengflying, walker}@cse.tamu.edu

Abstract—Pseudo functional *K* Longest Path Per Gate (KLPG) test (PKLPG) is proposed to generate delay tests that test the longest paths while having power supply noise similar to that seen during normal functional operation. Our experimental results show that PKLPG is more vulnerable to under-testing than traditional two-cycle transition fault test. In this work, a simulation-based X-Filling method, Bit-Flip, 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 (WSA). Experimental results on both compacted and un-compacted test patterns are presented. The results demonstrate that our method can significantly increase effective WSA while limiting the fill rate.

Keywords: delay test, pseudo functional test, power supply noise, test generation

I. INTRODUCTION

In recent years, considerable research effort has been dedicated to at-speed test using the path delay fault model [1-4] since it has the advantage of capturing the accumulative effects of small delay defects [5]. However, the maximum operating frequency (FMAX) during at-speed scan test may not correlate well to functional and system tests due to the mismatch of power supply noise (PSN) [6, 7].

A. Over-testing and under-testing

One of the primary reasons for the FMAX mismatch between scan and functional test is that test patterns are generated utilizing only structural information and traditional two-cycle at-speed test uses only a short clock burst in the functional mode. Since the scanned-in pattern may contain illegal states [8], a short burst of functional cycles is insufficient to exclude all illegal states and bring the circuit to a functional state [9]. The illegal states can lead to increased switching activity, resulting in excessive power supply voltage drop. Once the test-mode PSN exceeds the design margin, a good chip may fail the test, which is called *over-testing* [10].

Several techniques have been proposed to handle over-testing in traditional two-cycle delay testing, such as low-power test generation [11], X-filling [12-15], and power-aware static compaction [10]. The main disadvantage of these methods is that they do not change the fact that illegal states may still be included in the test pattern. Pseudo functional test (PFT) [8, 16] has been proposed to explicitly constraint test generation to use functional or near-functional states.

A test must also propagate transitions on paths under the realistic worst-case PSN in order to minimize *under-testing*, where a test is too quiet and leads to a scan test FMAX higher than functional FMAX. Approaches using a genetic algorithm (GA) [1], automatic test pattern generation (ATPG) [3] and SAT [17] have been developed to maximize PSN for delay test.

In order to strike a balance between over-testing and under-testing, the approach in [2] proposes to test chips under the worst-case functional PSN condition. Delay test considering both functional constraints and power supply noise are reported in [2, 18].

B. Pseudo functional KLPG test

Pseudo functional KLPG (PKLPG) test is proposed to generate delay tests that test the 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 Figure 1. The initial functional clocks, termed the *preamble*, run at lower speed to ramp up off-chip power supply currents and minimize the noise due to the chip and package inductance [7]. 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 [7, 19].

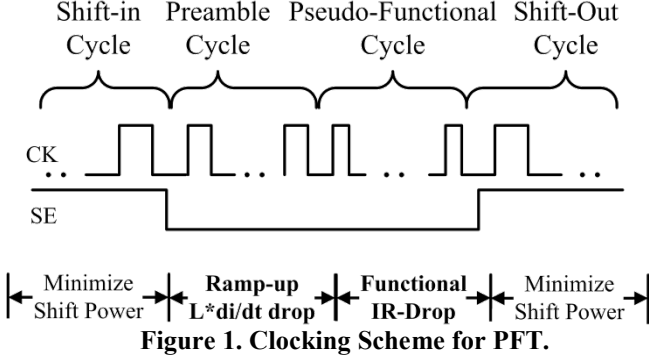
PKLPG can be viewed as short bursts of functional tests [9]. The primary advantage is that they apply tests in a more functional manner, so that power supply noise, signal coupling and power dissipation are more similar to functional operation. As for shift mode (both in and out), accumulative shift power dissipation can be minimized by either DFT, such as scan-chain partitioning [20] or test pattern manipulation, such as MT-filling [21]. Our in-house KLPG test generator, *CodGen* [4] 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.

Another key issue with PKLPG test is that functional PSN during the capture cycle can be much lower than the worst-case functional PSN and hence probably cause under-testing. 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. Therefore, the objective of our work is to develop a PSN control scheme for PKLPG.

In this work, we first investigate the PSN scenario for PKLPG test using random pattern simulation, which shows that PKLPG is more vulnerable to under-testing than over-testing. A simulation-based X-Filling procedure called *Bit-Flip* is proposed to maximize PSN during launch and capture for a partially-specified PKLPG pattern. Experimental results on un-compacted longest path patterns of ITC benchmark circuit b19 demonstrate our scheme is able to improve 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 is as high as 20%. The trade-off between CPU time and noise maximization is also discussed.

The remainder of this paper is organized as follows: Section II reviews previous work on PSN control for traditional two-cycle delay test. Section III investigates PSN in PFT test. Section IV presents our proposed X-Filling scheme, Section V describes experimental results, and Section VI concludes the paper and discusses future work.



II. BACKGROUND

This section reviews previous research on controlling PSN for traditional two-cycle delay test and discusses the space to extend this work to PKLPG test.

A. Related prior work

Several techniques [1-3, 17, 18] have been studied to maximize PSN during delay test. In [1], a GA-based iterative procedure is proposed. In each iteration, waveform simulations and fitness calculation are performed to guide selection, crossover and mutation to find patterns that induce larger PSN. In [17], PSN maximization is modeled as a MIN-ONE problem and a SAT solver is used to maximize the transition count. Two methods [2, 3] use justification to maximize transitions on the neighboring cells. While [2] proposes several techniques to speed up the justification process, [3] utilizes a commercial ATPG engine to sensitize neighboring signal lines by virtual test point insertion. Max-Fill [18] computes functional reachable states that induce maximal WSA using both logic simulation and justification. Later partially specified patterns are filled with these computed states.

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.

In our work, we investigate a simulation-based method, *Bit-Flip*, to fill X bits such that the effective WSA during the capture cycle is maximized. A partially-specified scan pattern set is first randomly filled and then an iterative procedure is invoked to flip a randomly-selected group of the filled bits. Effective WSA is measured after each flip and the inversion is retained if it improves the effective WSA. *Bit-Flip* requires no modification to the ATPG process or circuit and has no knowledge of the fault model used during ATPG. Incremental simulation is utilized to reduce CPU time.

Bit-Flip is similar to the hill-climbing procedure reported in [13], 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. The PSN in PKLPG focuses on IR-drop. We flip a group of bits per iteration rather than one at a time.

III. MULTIPLE-CYCLE PSN PROFILING

This section presents the PSN profiling results for PKLPG using random pattern simulation and analyzes the impact of random filling on WSA during the capture cycle. Primary inputs (PIs) are kept constant during simulation since low-cost 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 [9, 13] is used as a metric to evaluate the PSN during each cycle.

A. Pseudo-Functional PSN

Figures 2 and 3 depict the detailed results of PSN profiling. Chip1 and Chip2 are industrial designs, while b19 and s38417 are from the ITC99 and ISCAS89 benchmarks respectively.

Figure 2 shows the average WSA over the 16 cycles for the selected circuits. For convenience, WSA is normalized to cycle 2. 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 at roughly cycle 6. The reason is that the high WSA in the first two cycles is probably introduced by illegal states. After preamble cycles, the illegal states die out and the circuit approaches a functional state. Therefore, applying at least six preamble cycles (PKLPG) will produce PSN closer to functional [9, 18]. Traditional two-cycle test is applied at cycle 2 and sees a higher WSA. PKLPG sees a much lower WSA, and thus is more vulnerable to under-testing rather than over-testing.

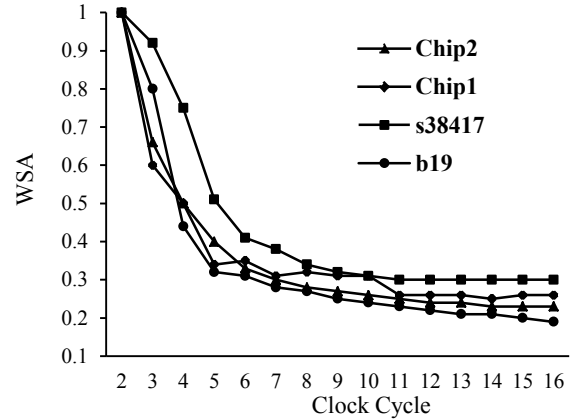


Figure 2. Average WSA Falls with Multiple Cycles.

Figure 3 illustrates the correlation between the WSA at cycle 2 and cycle 16 for b19. Similar results are observed for the other circuits. The PSN results 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-based methods [14] cannot guide filling in the presence of many preamble cycles. The large pseudo-functional PSN range at cycle 16 indicates the importance of X-filling. Given a partial specified pattern, the X bits should be assigned to sensitize the worst functional PSN condition. In the following section, we

will describe a simulation-based X-Filling method to control PSN.

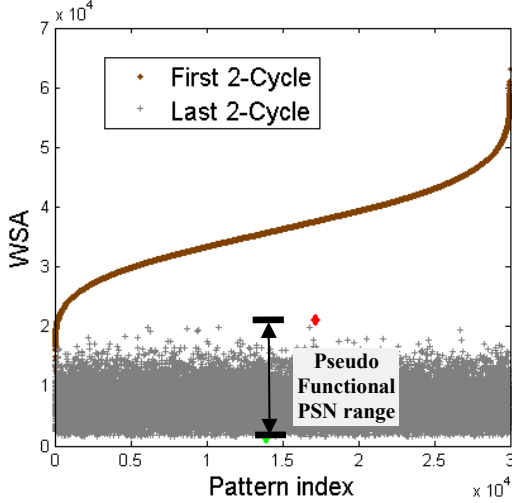


Figure 3. b19 WSA correlation of different cycles.

IV. NOISE CONTROL FOR PKLPG

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.

A. Overview of Proposed Framework

The Bit-Flip algorithm is shown in Table 1. It consists of a preprocessing step, an iterative step and a bit-relaxation step. Circuit net list, layout, critical path list and test pattern set are inputs to the algorithm.

In the preprocessing step, a pattern is fetched from the test set as well as the corresponding path(s). First procedure *Critical-Identify*($P, T, Circuit$) identifies the cells near on-path gates by physical position matching and stores them in *SCells*. We term these cells *Critical Cells*. It has been demonstrated that critical cells have considerable impact on the PSN of on-path gates [3]. Meanwhile, scan cells located in the fan-in cone of the critical cells are also marked and stored in *Sbits*. These scan cells are termed *critical bits*. Next, all critical bits (*Sbits*) are randomly filled and an event-driven logic simulator, *CodSim*, simulates the filled pattern. Then effective WSA (EWSA) is measured as initial *Max-WSA*. EWSA is defined as the sum of critical cell 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, *GetPotentialSCList*(*Circuit*) collects potential scan cells that have constant logic values at the launch/capture cycles. The Union, $L1$, of *Sbits* and the potential scan cells list serves as the final bit set for PSN control. Then Bit-Flip enters the iterative process. In each iteration, it randomly selects up to G scan cells from the set $L1$ and flips them. Then *CodSim* simulates the result, and EWSA is measured. If EWSA is increased, the flipped bits are kept; 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 or test compaction.

Table 1. *Bit-Flip* algorithm.

Bit-Flip(Circuit, Path, Pattern, Layout)

1. Layout-Preprocess (Circuit, Layout);
2. for each Pattern T and corresponding path P
3. ($SCells, Sbits$) = *Critical-Identify* ($P, T, Circuit$);
4. *Critical-Random-Fill*($Sbits, T$);
5. *CodSim*($T, Circuit$);
6. $Max-WSA = \text{Critical-WSA}(SCells)$;
7. while more Rounds
8. $L1 = \text{GetPotentialSCList}(Circuit) \cup Sbits$;
9. While ($L1 \neq \emptyset$)
10. Randomly select up to G bits, flip them;
11. *Event-Driven-Sim*($T, Circuit$);
12. $NewWSA = \text{Critical-WSA}(SCells)$;
13. if($NewWSA < Max-WSA$)
14. Fail, Restore flipped bits;
15. else
16. Succeed, Keep the flipped bits;
17. $Max-WSA = New-WSA$;
18. EndWhile
19. Update(G);
20. EndWhile
21. *Bit-Relaxation*(Circuit, T)
22. EndFor

B. Critical Cell Identification

In this work, we utilize the effective region to identify critical cells, as shown in Figure 4. This approach is motivated by the model used in [3]. Vertically, gates in the same and neighboring rows are critical cell candidates since they share either power or ground with 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 localization character of PSN. All gates within the effective region are critical cell candidates.

Critical-Identify ($P, T, Circuit$) performs 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).

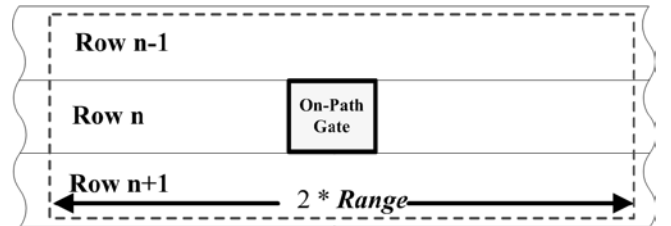


Figure 4. Effective Region.

C. 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 simulation time.

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 scan bits. The flipped bits spread sparsely along the scan chains.

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

$$B = \sum_{i=0}^{\lfloor \frac{G}{d} \rfloor - 1} I \cdot (G - d \cdot i)$$

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 be 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 must 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.

D. Critical Bit Fill and Bit Relaxation

In order to narrow 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, *SBits*. 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 *SBits*, we need to identify the bits which are insignificant to EWSA maximization and exclude them from *SBits*. That is, if we relax an insignificant bit to X, EWSA will not be reduced. Therefore, we apply a bitwise bit-relaxation procedure *Bit-Relaxation(Circuit, T)* 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 [22], 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 rates, 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.

E. Compacted Pattern Consideration

Test compaction [10, 23] is used to reduce pattern count and minimize test 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 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.

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.

V. EXPERIMENT 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 Figure 5, 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.

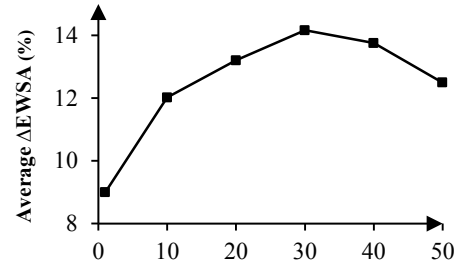


Figure 5. b19: Average ΔEWSA vs. Group Size.

Second, we investigate 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 2.

In Table 2, 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%. 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 Figure 6. 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, Figure 7 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.

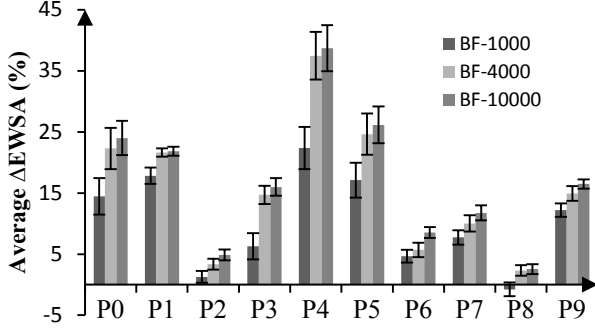


Figure 6. Average ΔEWSA with 95% C.I.

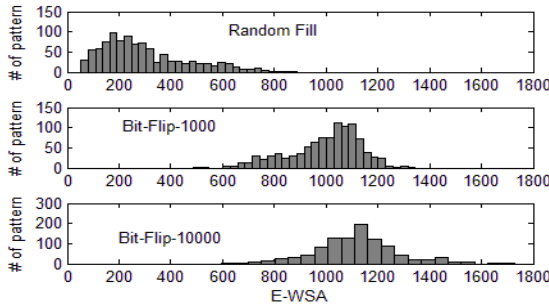


Figure 7. P0 EWSA Distribution vs. Fill Method.

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 3. 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 patterns are care bits. 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 [3]. 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 [3] is 14.08% with virtual test point insertion, with a CPU time of 40s for the ATPG step. As shown in Tables 2 and 3, BF-4000 averages 17.24% T.R. at 38s CPU time. So the two methods have similar performance.

Thirdly, 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 $\Delta\text{EWSA-R}$) and from best random (Average $\Delta\text{EWSA-BR}$) are shown in Figure 8. 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%.

Finally, we evaluate the ΔEWSA achieved on compacted patterns of benchmark circuits in Table 4. The patterns were dynamically compacted [23]. 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 pattern count (T.P.), critical pattern count (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.

Table 2 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

VI. CONCLUSIONS AND FUTURE WORK

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 method, *Bit-Flip*, to control PSN for PKLPG test. Experimental results on both un-compacted and compacted test patterns demonstrated the effectiveness of the method.

Future work will focus on improving algorithm efficiency by utilizing parallel pattern logic simulation. This should reduce the CPU time by more than an order of magnitude. The compaction algorithm will also be investigated to determine its interaction with PSN on PKLPG patterns. Prior work [10] has focused on vetoing compactations that would exceed a noise limit, but have not looked at compacting patterns to minimize critical patterns. Furthermore, we will investigate more accurate and efficient PSN estimation methods, such as [24] and apply the proposed scheme to multiple clock domains.

Table 3. 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

Table 4. PSN Control Result of Compacted Patterns.

Circuit	T.P.	C.P.	Avg. Δ WSA (%)	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

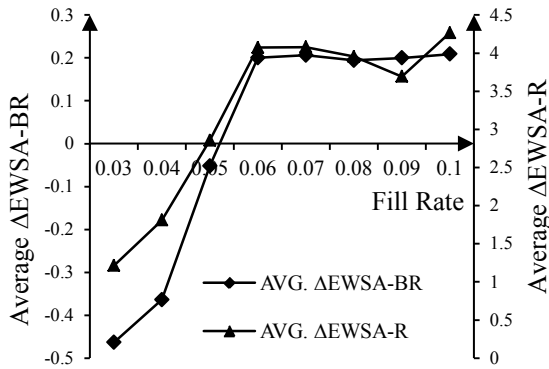


Figure 8. Average Δ EWSA vs. Fill Rate.

References

- [1] 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. Computer-Aided Design Int. Circ. Sys.*, vol. 20, pp. 416-425, 2001.
- [2] F. Yuan, X. Liu and Q. Xu, "Pseudo-functional testing for small delay defects considering power supply noise effects," *Int'l Conf. Computer-Aided Design*, 2010.
- [3] J. Ma and M. Tehranipoor, "Layout-Aware Critical Path Delay Test Under Maximum Power Supply Noise Effects," *IEEE Trans. Comp.-Aided Design Int. Circ. Sys.*, vol. 30, pp. 1923-1934, 2011.
- [4] W. Qiu, J. Wang, D. Walker, D. Reddy, X. Lu, Z. Li, W. Shi and H. Balachandran, "K longest paths per gate (KLPG) test generation for scan-based sequential circuits," *Int'l Test Conf.*, 2004, pp. 223-231.
- [5] L. T. Wang, C. W. Wu and X. Wen, *VLSI Test Principles and Architectures: Design for Testability*. Morgan Kaufmann Pub, 2006.
- [6] P. Pant and E. Skeels, "Hardware hooks for transition scan characterization," *IEEE Int'l Test Conf.*, 2011, pp. 1-8.
- [7] P. Pant and J. Zelman, "Understanding power supply droop during at-speed scan testing," *IEEE VLSI Test Symp.*, 2009, pp. 227-232.
- [8] Y. C. Lin, F. Lu and K. T. Cheng, "Pseudofunctional testing," *IEEE Trans. Comp.-Aided Design*, vol. 25, pp. 1535-1546, 2006.
- [9] I. Pomeranz, "On the Switching Activity and Static Test Compaction of Multicycle Scan-Based Tests," *IEEE Trans. Computers*, vol. 61, pp. 1179-1188, 2012.
- [10] J. Wang, W. Qiu, S. Fancier, D. Walker, X. Lu, Z. Yue and W. Shi, "Static compaction of delay tests considering power supply noise," *IEEE VLSI Test Symp.*, 2005, pp. 235-240.
- [11] X. Wen, Y. Yamashita, S. Morishima, S. Kajihara, L. T. Wang, K. K. Saluja and K. Kinoshita, "Low-capture-power test generation for scan-based at-speed testing," *IEEE Int'l Test Conf.*, 2005, pp. 1-10.
- [12] 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," *Design Auto. and Test in Europe*, 2008, pp. 1184-1189.
- [13] R. Sankaralingam and N. A. Touba, "Controlling peak power during scan testing," *IEEE VLSI Test Symp.*, 2002, pp. 153-159.
- [14] S. Remersaro, X. Lin, Z. Zhang, S. M. Reddy, I. Pomeranz and J. Rajski, "Preferred fill: A scalable method to reduce capture power for scan based designs," *IEEE Int'l Test Conf.*, 2006, pp. 1-10.
- [15] X. Wen, K. Miyase, S. Kajihara, T. Suzuki, Y. Yamato, P. Girard, Y. Ohsumi and L. T. Wang, "A novel scheme to reduce power supply noise for high-quality at-speed scan testing," *IEEE Int'l Test Conf.*, 2007, pp. 1-10.
- [16] F. Yuan and Q. Xu, "On systematic illegal state identification for pseudo-functional testing," *ACM/IEEE Design Auto. Conf.*, 2009.
- [17] L. Fang and M. S. Hsiao, "A fast approximation algorithm for MIN-ONE SAT," *Design Auto. and Test in Europe*, 2008, pp. 1087-1090.
- [18] X. Fan, S. Reddy and I. Pomeranz, "Max-fill: A method to generate high quality delay tests," *IEEE Symp. Design and Diagnostics of Electronic Circuits & Systems*, 2011, pp. 375-380.
- [19] B. Nadeau-Dostie, K. Takeshita and J. F. Cote, "Power-aware at-speed scan test methodology for circuits with synchronous clocks," *IEEE Int'l Test Conf.*, 2008.
- [20] L. Whetsel, "Adapting scan architectures for low power operation," *IEEE Test Conf.*, 2000, pp. 863-872.
- [21] N. Badereddine, P. Girard, S. Pravossoudovitch, C. Landrault, A. Virazel and H. J. Wunderlich, "Minimizing peak power consumption during scan testing: Test pattern modification with X filling heuristics," *Int'l Conf. Design and Test of Integrated Systems in Nanoscale Technology*, 2006, pp. 359-364.
- [22] A. El-Maleh and K. Al-Utaibi, "An efficient test relaxation technique for synchronous sequential circuits," *IEEE Trans. Comp.-Aided Design of Int. Circ. Sys.*, vol. 23, pp. 933-940, 2004.
- [23] Z. Wang and D. Walker, "Dynamic compaction for high quality delay test," *IEEE VLSI Test Symp.*, 2008, pp. 243-248.
- [24] M. F. Wu, H. C. Pan, T. H. Wang, J. L. Huang, K. H. Tsai and W. T. Cheng, "Improved weight assignment for logic switching activity during at-speed test pattern generation," *Asia and South Pacific Design Auto. Conf.* 2010.