Improved Random Pattern Delay Fault Coverage Using Inversion Test Points

Soham Roy, Brandon Stiene, Spencer K. Millican and Vishwani D. Agrawal
Auburn University, Department of Electrical and Computer Engineering
Auburn, AL 36849, USA
szr0075@auburn.edu; bks0015@auburn.edu; millican@auburn.edu; agrawvd@auburn.edu

***Abstract*—This article analyzes and rationalizes the capabilities of inversion-based test points (TPs) when implemented in lieu of control-0/1 TPs. With upward scaling of transistor density, delay faults can be masked when using pseudo-random tests with control-0/1 ("conventional") TP architectures. This study finds delay fault coverage can be improved using inversion TPs in logic circuits using pseudo-random tests without negatively impacting stuck-at fault coverage.**

***Keywords*—Design for test, random test, built-in self-test, test points, delay test**

I. Introduction

Circuit test is a critical part of the integrated circuit (IC) manufacturing process which confirms circuit reliability. Circuit test applies stimuli to a manufactured IC to excite defects created as a natural consequence of the silicon manufacturing process. The cost of testing circuits is a significant portion of IC manufacturing costs [1], [2], and as transistor density continues to scale upwards, circuit test costs are increasing and efforts continue to keep circuit test costs down. The primary challenge of IC test is to reduce test-related costs while preventing the release of defective circuits while not discarding good devices. Small increases in fault coverage is worth investment in order to reduce manufacturing defect levels.

Pseudo-random testing has been demonstrated to be effective at detecting defects in previous generations of technology, but its utility is degraded for complex circuits. Although pseudo-random tests detect fewer defects per test compared to deterministic, circuit-specific tests, i.e. those generated by automatic test pattern generators (ATPGs), they are more economical since they require less computational effort to generate and can be applied with minimal on-chip hardware. However, the utility of pseudo-random tests is degraded for modern technologies due to random-pattern-resistant (RPR) faults [1] (see Section II). These faults are a natural consequence of increasing circuit complexity and will continue to be present in new technologies since fulfilling consumer demands requires ever-more complex circuits. Many methods have been developed to improve the fault coverage of pseudo-random test patterns, with a common technique being test point insertion (TPI). Other methods of improving pseudo-random test effectiveness include changing the nature of random stimuli, such as deterministic seeding [3] and pattern weighting [4], but modifying the logic of a circuit during test using circuit controlling and circuit observing "test points" (TPs) has been of particular interest due to its ease-of-implementation on circuit netlists.

This study demonstrates the effectiveness of unconventional inversion TPs in lieu of traditional control TPs [5] by analyzing the delay and stuck-at fault coverage achieved by implementing these two TP architectures, which has not been examined by earlier studies on inversion TPs. Using inversion TPs is not standard practice today, and this study will hopefully motivate the electronic design automation industry to support inversion TPs in their DFT tools. Although inversion TPs have been introduced in earlier literature [6], [7], [8], no studies have compared their impact compared to conventional TP architectures. Their effectiveness at improving fault coverage, especially delay fault coverage compared to conventional TPs has yet to be explored. The specific contributions of this study are as follows:

- A rationale for the effectiveness of inversion TPs compared to conventional control TPs is provided.
- Experiments are performed which demonstrate inversion TPs frequently have higher stuck-at fault coverage compared to conventional TPs.
- Experiments are performed which demonstrate inversion TPs ability to increase delay fault coverage while conventional TPs frequently degrade stuck-at fault coverage.

The remainder of this article is organized as follows. The motivation for this study is described in Section II. The inversion TP architecture and its functioning are discussed in Section III. The experimental setup which evaluates inversion TP abilities is given in Section IV. Results and discussion on these experiments are given in Section V, and conclusions and future research directions are expressed in Section VI.

II. Motivation

Pseudo-random tests are variable and predictable circuit stimuli generated by a pseudo-random pattern generator (PRPG) in a built-in self-test (BIST) environment [1]. The typical hardware schema for applying pseudo-random tests is illustrated in Figure 1. A PRPG is typically implemented with a linear feedback shift register (LFSR) [9], although other architectures which generate variable, but predictable, stimuli can be used. To apply a test, the PRPG is first loaded

9978-1-7281-3382-9/19/$31.00

***Proceedings of 28th IEEE North Atlantic Test Workshop* (NATW), Essex, VT, May 13-15, 2019 (6 pages)**

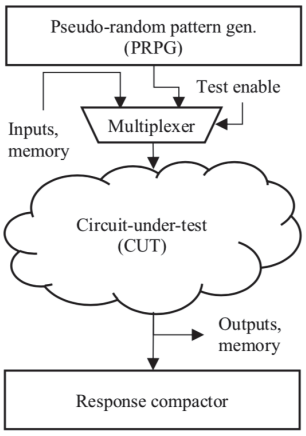


Fig. 1: The typical arrangement of pseudo-random testing hardware. In the test mode, inputs, memory, and outputs may be scan flip-flops connected in one or more scan chains [9]. The chains receive inputs from PRPG and feed into the response compactor.

with a "seed" which determines future stimuli. The circuit is then programmed to take inputs from the PRPG as opposed to normal circuit inputs. With each stimulus applied, circuit outputs are either directly observed or compressed into a "signature" generated by signal compression hardware (e.g., a multiple-input signature register (MISR) [1], [10]). This signature is compared against a simulated value, and if the hardware signature matches the simulated signature, the circuit is considered free of defects.

The effectiveness of pseudo-random tests is impaired by the presence of RPR faults, which are a natural occurrence in complex logic circuits. An RPR fault [1], [11] is a logical representation of a defect which is unlikely to be detected using random stimulus, since RPR faults can only be detected by a small set of test vectors among all possible stimulus. Although the probability of detecting these types of faults can be improved by applying many test vectors, the number of vectors needed to do so may be infeasibly large. A typical example of such a fault is illustrated in Figure 2. Exciting and observing the indicated fault requires 32 logic-0's to be applied to the OR gate and 32 logic-1's to be applied to the AND gate. Presuming all inputs have an equal probability of being logic-0 or logic-1, the probability of this stimulus occurring is $2^{-64} \approx 5.4 * 10^{-20}$, which makes its application highly unlikely using random stimuli.

A. Conventional TP Architectures

The purpose of a TP is to make the excitation of faults and the observation of faults more likely under random stimulus. Signal-controlling TPs function using an extra *test enable* pin (or scannable register) which forces logic in a circuit to a controlled value. While not under test, *test enable* is disabled, which leaves the function of the circuit unchanged. During test, *test enable* is enabled, which allows hard-to-control signals to be directly controlled. Typical TP architectures use AND gates

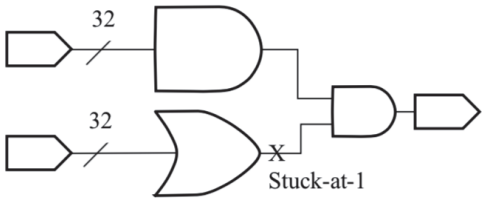


Fig. 2: An example of a fault requiring one specific vector to detect, which is unlikely to be generated using random stimuli.

or OR gates (or other analogous logic-forcing structures) to force a circuit line to logic-0/1 (respectively) during test [4], as illustrated in Figure 3(a) and 3(b). The first input to these *control TPs* is the circuit line to be forced during test, and the second input is the *test enable* signal (which can come from a circuit pin, a dedicated scannable latch, or a functional latch [12]).

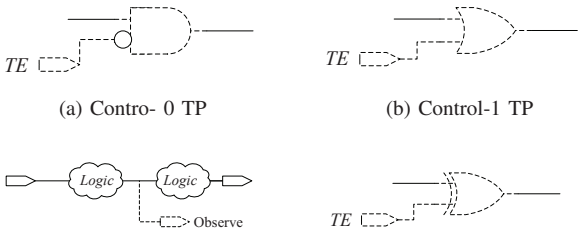
Forcing circuit lines to known values has two effects: defects can be excited which are unlikely to be excited under random stimulus, and circuit paths can be activated to allow defects to be propagated to circuit outputs under random stimuli. Control TPs are known to increase circuit area with extra gates, but this extra circuit area is deemed worthwhile given fault coverage improvements made by the control TPs [13].

Another category of TPs, *observe TPs*, directly observes circuit logic as opposed to controlling it, but observe TPs are not the immediate subject of this study. To make a circuit line easier to observe, a circuit line can be diverted to drive circuit outputs (or scannable latches [14]) made specifically for test, as shown in Figure 3(c). Observe TPs are not addressed by this study as the detriments described in the following sub-section do not immediately apply to them, but issues involving observe TPs will be addressed in future studies (see Section V).

B. Conventional TP Detriments

Although the intention of control TPs is to excite "stuck values" (i.e., "stuck-at faults") in a circuit, their use can have unintended consequences. Since an active control TP forces a line to a set value, only one stuck-at value (stuck-at 0 or stuck-at 1) can be excited when a control TP is active. Also, an active control TP will prevent logic on the controlled signal from passing through the TP. This later effect will prevent any faults present on the controlled line from being observed. Although the intention of control TPs is to increase stuck-at fault coverage, these qualities can hinder their ability to excite and propagate such faults.

Another disadvantage of control TPs is they prevent signal transitions, which blocks the transmission and excitation of delay faults. In a circuit's functional mode, delay-causing defects can cause an incorrect value to be observed at circuits output when the correct circuit value can not reach the output in time, as is illustrated in Figure 4. Unlike stuck-at faults, delay faults require two circuit input vectors to detect: one to activate the fault and one to launch the slow transition in the circuit. Control TPs prevent signal transitions when enabled



(a) Contro- 0 TP (b) Control-1 TP

(c) Observe TP (d) XOR based TP

Fig. 3: Test Points (TPs) force a signal to logic-0 (control-0), force a signal to "logic-1" (control-1), observe a signal directly (observe), or invert signal from the previous logic stage. TPs may require an extra pin to force a circuit value. Circuit changes are shown in dashed lines.

(since they force signals to a value), and therefore control TPs will block all delay faults from passing through the TP. They also prevent delay faults on the output of the TP from being excited, and they can make other delay faults driven by the controlled line less likely to be excited.

Although typical test procedures will apply tests with TPs both on and off, removing the disadvantages form active TPs will make tests more effective. This is needed when the time to apply random stimuli is limited (such as for in-field tests) and when few "test modes" can be applied.

III. Inversion-based TP architecture

Unlike conventional control TP architectures discussed in Section II-A, *inversion TPs* change signal value probabilities through inversions as opposed to forcing circuit lines to predetermined values. Conventional control TPs force signal value probabilities by forcing constant values on lines, i.e., when a control TP is enabled, the probability of logic-0/1 on a line is 100% while the probability of the opposite value occurring is 0%, which creates the detriments discussed in Section II-B. Inversion TPs, on the other hand, invert signal probabilities. For example, if a signal has a 75% chance of being logic-1, activating an invert TP will make the probability of being logic-1 25%.

The implementation of inversion TPs is illustrated in the Figure 3(d). An inversion TP is implemented using an XOR gate connected to *test enable* and the other input is the inverted signal. When the TP is disabled, the XOR gate functionally becomes a buffer, allowing the original circuit value to pass-though. When the TP is enabled with *test enable*, the XOR functionally becomes an inverter. Like control TPs, the source of *test enable* signal can be a scanned register or a circuit-level pin.

A potential advantage of inversion TPs over conventional control TPs is their ability to excite both stuck-at 0 and stuck-at 1 faults when activated, as well as to allow stuck-at faults on the inverted line to propagate through. Since these TPs do not force logic to set values, it is possible to excite both logic-0 and logic-1 with these TPs when activated. Also, when an

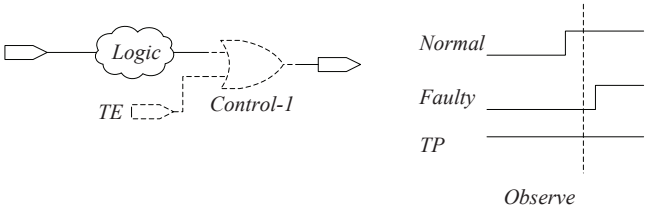


Fig. 4: Here, the effect a control TP has on the excitation and observation of a delay fault is illustrated. The output waveforms of a normal circuit, a faulty circuit, and a circuit with an active control TP are labeled, as is the point in time where an observation occurs.

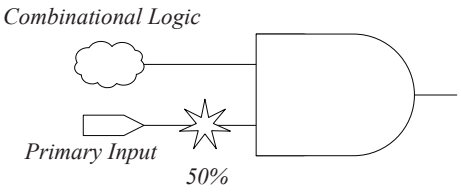


Fig. 5: This figure illustrates an instances where inversion TPs may not increase fault coverage: if the best TP solution is to force the circuit input to logic-1 during test, an inversion TP will fail to do so.

inversion TP is activated, a fault's effect on the inverted line is no longer forced to a given value but is instead allowed to pass though the TP (albeit inverted), which in turn gives a chance for the fault's effect to be observed at a circuit output.

A second potential advantage of inversion TPs is they can excite and propagate delay faults when active. Since inversion TPs do not force circuit lines to set values, they provide an opportunity for transitions to occur on the TP output, and these transitions can excite delay faults on the TP output (or on logic driven by the TP). Additionally, if a transition (from a delay fault) occurs on the inverted line, this transition can pass through the inversion TP (albeit inverted) and has the opportunity to be observed at a circuit output.

Inversion TPs may falter when a random signal needs to be less random, but whether such occurrences will degrade inversion TPs performance has yet to be studied. To detect faults under random stimuli, it may be required to force a relatively random signal to a set value. For example, as illustrated in the Figure 5, an AND gate may prevent faulty logic from passing though it based on another signals value, and thus forcing that signal to logic-1 allows values to pass though the gate. If that controlled signal is normally random, this cannot be accomplished with an inversion TP. Whether such conditions are likely to occur, however, has yet to be determined and will be explored though experiments (see Section V).

To the author's knowledge, using XORs as TPs has been noted in previous studies [6], [7], [8], but their advantages compared to conventional control TPs and their effect on

delay fault coverage has yet to be examined or leveraged. The focus of [6] was broad and covered many topics in weighted random pattern generation, with a single section devoted to TPI. The study chose to implement control-0 and control-1 points in a circuit but implemented these control TPs as XOR gates under the assumption that such TPs would only be placed on lines where signals were almost always be logic-0/1 under random stimuli. Beyond this assumption, the effect of implementing control-0/1 TPs in such a manner was not explored, and neither was the effect such a TP architecture has on the detection of delay faults. It must also be noted that the assumption made by the study contradicts the observation made by the previous paragraph, since if a control-0/1 TP on the previously described location was replaced by an inversion point, its effect would be removed. XORs as a TP was also noted in [7], [8], but the study did not compare inversion TPs against conventional control TPs, nor did the study analyze the effectiveness of inversion TPs at detecting delay faults.

IV. Experimental Setup

This section provides the experimental setup used to evaluate conventional control TPs and inversion TPs. Experiments under this setup will provide a fair comparison between the two architectures and demonstrate their ability to detect faults under random stimuli.

A. Test Point Insertion (TPI)

This study does not propose a new TPI algorithm, and instead will use an established TPI algorithm from literature to compare the different TP architectures [15]. This method was chosen because it can insert inversion TPs or control TPs without favoring one architecture over another since the algorithm itself does not require specific information on the TPs used and instead only knows the calculated effect a TP will have on fault coverage. The controllability and observability program (COP) [16] is used to quantify a circuit based on probabilistic metrics. Many algorithms use COP [15], [12] to find the TP which maximizes the calculated fault coverage of the circuit (FC) by calculating the probability that each fault will be detected (P_f), depending on stuck-at-0 (*SA0*) or stuck-at-1 (*SA1*) fault model:

$$P_f = \begin{cases} controllability \times observability & \text{for } SA0 \\ (1 - controllability) \times observability & \text{for } SA1 \end{cases}$$

$$FC = (1/|F|) \sum_{\forall_f \in F} P_f \quad (1)$$

Heuristic TPI algorithm starts by performing testability analysis [16] on a given circuit, which assigns "controllability" (the probability a circuit line will be logic-1) and "observability" (the probability a circuit line will be observed at a circuit output pin) values to all lines in a circuit. These controllability and observability values can then be used to predict if a fault will be detected, i.e., if it will be excited and observed, which in turn can be used to predict the fault coverage if a given number of random vectors are applied. It should be noted that this calculated fault coverage is not the actual circuit fault coverage since the controllability and observability calculations are not precise when re-convergent fan-outs are present in the circuit [16]. The TPI algorithm iteratively calculates the impact each candidate TP will have on circuit fault coverage by re-calculating controllability and observability values when the TP is active. The TPI program then inserts the TP which provides the highest positive impact on the calculated fault coverage. This process is repeated until the number desired TPs is inserted, the predicted fault coverage is reached, no more TPs which increase the fault coverage can be inserted, or a computation time limit is reached.

Although other TPI implementations from literature can be chosen for this study, it is presumed the TPI algorithm used will have not impact this study's conclusions as long as the TPI method does not give an advantage to any TPI architecture.

When performing TPI with the two architectures, observe points will also be allowed on each line to mimic an industrial application of the inversion-based TP architecture, as control and observe TPs are often used together. Candidate TPs for the conventional architecture will include control-0, control-1, and observe TPs on the input and output of every gate, while candidate TPs for the inversion-based TP architecture will include inversion TPs and an observe TPs on every such line.

In this study, every TP has the same *test enable* signal. Alternatively, different TPs can have different *test enable* signals which are not uniformly on/off [17], [4], but the impacts of such TP architectures is not the scope of this study.

To model delay faults, this study uses the transition delay fault (TDF) model. This choice is made due to it's ease of implementation and its known ability to model circuit defects [18].

B. Execution

All programs (TPI and fault simulation) are run on a high-performance workstation representative of an industrial development environment. The workstation is equipped with an Intel i7-8700 processor and 8GB of RAM. Software programs were written in C++ and complied using the MSVC++14.15 compiler with maximum optimization parameters.

Original TPI and fault simulation programs were written for this project. This choice was made in lieu of using of industrial tools for ease-of-integration of different parts of the TPI flow (circuit testability analysis, circuit modification with TPs, fault simulation, etc.). Using industrial tools was also infeasible since such tools are not programmed to give optimal inversion TP placement and would give favourable result towards conventional control TPs. This is in contrast to the implementation of [15], which does not have knowledge of TP types and instead only predicts their impact on fault coverage given circuit modifications.

Table I provides information of benchmark circuits used in this study. These circuits include ISCAS'85 [19] and ITC'99

TABLE I: Experimental Results on Benchmark Circuits

Circuit Details					#TP (Cont./Obs.)		SAF Coverage (%)			TDF Coverage (%)			TPI CPU seconds	
Name	#Gates	#PI	#PO	#Faults	Conv.	Invert	No TP	Conv.	Invert	No TP	Conv.	Invert	Conv.	Invert
b04	652	77	74	1832	6/6	12/0	98.75	99.12	99.67	98.41	95.64	97.61	222.29	106.15
b05	927	35	60	2520	1/1	2/0	77.06	77.50	77.50	72.57	69.04	69.03	1323.91	39.21
b07	383	50	57	1112	2/3	5/0	97.93	98.38	100	96.94	89.74	100	34.82	16.79
b11	726	38	37	1900	2/2	4/0	95.63	95.63	96.63	95.15	91.960	95.168	88.62	38.39
b12	944	126	125	2825	4/1	4/1	97.34	98.30	99.89	94.30	84.61	97.10	247.18	75.79
b13	289	63	63	851	1/0	1/0	96.00	97.06	100	96.00	96.94	99.53	3.88	1.02
c432	160	36	7	573	1/1	2/0	99.00	99.30	99.47	99.30	98.94	97.56	3.41	1.61
c1355	546	41	32	1566	2/5	7/0	99.48	99.48	100	99.48	93.37	100	109.04	28.54
c2670	1193	233	140	3481	7/7	12/2	82.90	95.68	95.63	81.55	85.39	90.43	55542.43	402.99
c3540	1669	50	22	4527	4/2	6/0	96.00	96.17	96.04	95.22	54.62	95.22	1006.89	352.33

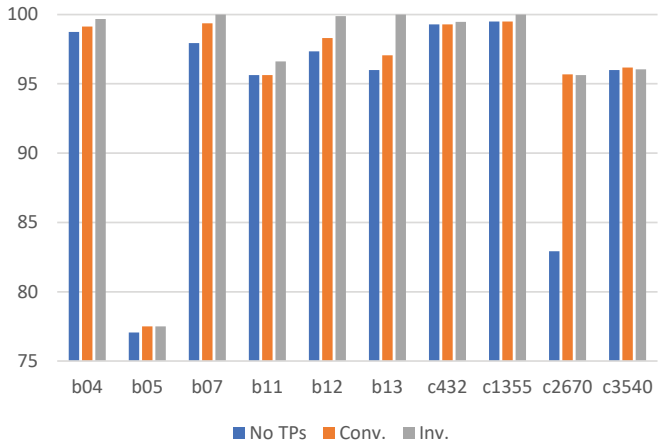


Fig. 6: Stuck-at fault coverage for ITC'99 and ISCAS'85 benchmarks

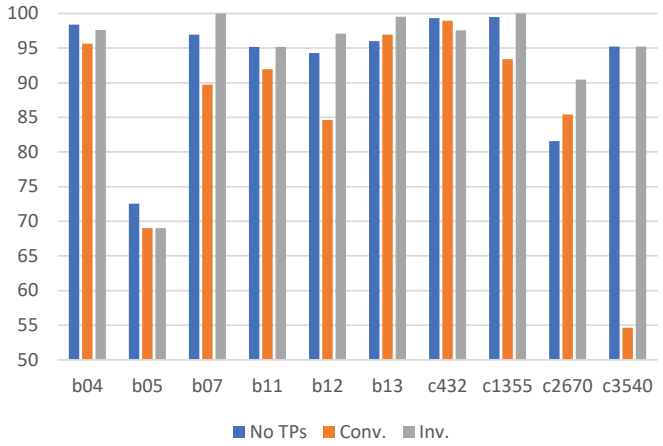


Fig. 7: Transition delay fault coverage for ITC'99 and ISCAS'85 benchmarks

[20] benchmarks. For each circuit, the table provides the number of gates ("#Gates"), number of primary inputs ("#PI"), the number of primary outputs ("#PO"), and the number of stuck-at or transition delay faults without fault collapsing ("#Faults"). The sequential cells in benchmark circuits are converted into pseudo-inputs and pseudo-outputs in order to emulate a full-scan environment (i.e., every sequential cell is fully controllable and observable during test by a PRPG and compactor, e.g., the STUMPs architecture [9]), and these additional circuit pins are included in "#PI" and "#PO". Table I omits ISCAS'85 and ITC'99 benchmarks which achieved greater than 99.5% fault coverage without TPI or where TPs (conventional or proposed) do not change fault coverage after simulating 65,535 vectors (b01, b02, b06, b08, b09, b10, c17, c499, c880, and c1908), as these benchmarks do not require TPs to be tested using random stimuli.

V. Results and Discussion

Table I presents the results of performing TPI using control TPs and inversion TPs. Each row gives the results obtained from a given benchmark circuit. Columns "Conv." and "Invert" under the heading "#TP (Cont./Obs.)" give the number of TPs inserted under the conventional and the inversion TP architecture, respectively. The number left of '/' gives the number of control/inversion TPs inserted and the number right of '/' gives the number of observe points inserted. It is noteworthy that under the inversion-based TP architecture, fewer observe points are chosen to be inserted despite the same observe points being candidates for insertion under both architectures. This implies inversion TPs have a positive impact on observability that reduces the need for observe points. The columns "No TP", "Conv." and "Invert" under the heading "SAF/TDF Coverage (%)" give the SAF/TDF fault coverages after simulating 65,535 random vectors (or vector pairs for TDF coverage) with no TP, using coventional TPs, and using inversion TPs. For the two TP architectures, half of the vectors are with TPs enabled and the other half are with TPs disabled, which is typical in industrial settings.

Stuck-at fault coverage results from Table I, which are comparatively plotted in Figure 6 show the proposed architecture does not negatively impact stuck-at fault coverage, but instead appears more likely to increase it. With the exception of circuits c2670 and c3540, the inversion-based TP architecture always obtains higher stuck-at fault coverage with the same number of TPs.

TDF coverage results from Table I, which are comparatively plotted in Figure 7, demonstrate the positive impact that inversion TPs have on delay fault coverage. For every benchmark except b05 and c432, the delay fault coverage obtained by inversion TPs is significantly greater than the

delay fault coverage obtained by conventional TPs. This result was predicted in Section III, with anomalies being acceptable due to the nature of random stimuli. A second noteworthy result from Figure 7 is, more often than not, the conventional architecture decreases delay fault coverage significantly. This can be attributed to factors discussed in Section II-B, which proved to be worthwhile motivation for this study.

A corollary result from Table I is the time required to perform TPI using inversion TPs is significantly less compared to using conventional control TPs since fewer TPs need to be evaluated. Results under the heading "TPI Time (s)" show the time required to insert TPs is decreased never less than by half. The likely reason for this significant decrease in TPI time is fewer TPs need to be considered during TPI, and thus finding the best TP to insert requires less computation time. This is because the conventional TP architecture has 3 possible TPs on a given line (control-0, control-1, and observe) while the inversion-based TP architecture has only 2 (invert and observe).

VI. Conclusion and Future Directions

In this article, improving delay fault coverage through alternative TP implementations has been studied. The results show that inversion TPs do not negatively impact stuck-at fault coverage while simultaneously increasing delay fault coverage compared to conventional control TPs.

Observe TPs must be a focus of future studies, as their impact on delay fault detection is not yet known. Observe TPs change circuit timing paths when present, and therefore may capture a line's value after the effect of a delay fault has passed. This study chose to model delay faults as TDFs due to their ease of implementation and their ability to model known defects (i.e. stuck-open faults), but if the fault model includes the magnitude of a delay (as is the case of small delay defects [21]), the presence of observe points may decrease delay fault coverage. Future endeavors will explore the impact observe points have on such fault coverages as well as explore architectures which can remedy observe TP detriments.

Another future avenue of research is the impact TPs have on the detection of redundant faults and producing false failures. In this study, fault simulation did not remove redundant faults [22], and therefore some undetected faults are truly undetectable. However, the addition of TPs may make faults which are normally impossible to detect (and therefore have no impact on the function of the circuit) detectable. If such faults are detected when TPs are active, this will create a "false failure", which in turn unnecessarily reduces circuit manufacturing yield. An avenue of future studies is to select TP locations which increase fault coverage while simultaneously preventing redundant faults from being excited.

References

[1] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Springer, 2000.

[2] P. K. Nag, A. Gattiker, S. Wei, R. D. Blanton, and W. Maly, "Modeling the economics of testing: a DFT perspective," *IEEE Design & Test of Computers*, vol. 19, no. 1, pp. 29–41, Jan 2002.

[3] D. Xiang, X. Wen, and L. Wang, "Low-power scan-based built-in self-test based on weighted pseudorandom test pattern generation and reseeding," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 3, pp. 942–953, Mar. 2017.

[4] J. Rajski and J. Tyszer, *Arithmetic Built-In Self-Test for Embedded Systems*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1998.

[5] C. Acero, D. Feltham, Y. Liu, E. Moghaddam, N. Mukherjee, M. Patyra, J. Rajski, S. M. Reddy, J. Tyszer, and J. Zawada, "Embedded deterministic test points," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2949–2961, Oct 2017.

[6] D. Bakshi, "Techniques for Seed Computation and Testability Enhancement for Logic Built-In Self Test." Master's thesis, Virginia Tech., 2012.

[7] Y. Fang and A. Albicki, "Efficient testability enhancement for combinational circuit," in *Proceedings of International Conference on Computer Design (ICCD)*, Oct 1995, pp. 168–172.

[8] E. M. Rudnick, V. Chickermane, and J. H. Patel, "An observability enhancement approach for improved testability and at-speed test," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 8, pp. 1051–1056, Aug 1994.

[9] P. H. Bardell, W. H. McAnney, and J. Savir, *Built-in Test for VLSI: Pseudorandom Techniques*. New York: Wiley-Interscience, 1987.

[10] R. David, "Signature analysis for multiple-output circuits," *IEEE Trans. Comput.*, vol. 35, no. 9, pp. 830–837, Sep. 1986.

[11] M. J. Geuzebroek, J. T. van der Linden, and A. J. van de Goor, "Test point insertion that facilitates ATPG in reducing test time and data volume," in *Proceedings of the IEEE International Test Conference*, Washington, DC, USA, 2002, pp. 138–147.

[12] J. Yang, N. A. Touba, and B. Nadeau-Dostie, "Test point insertion with control points driven by existing functional flip-flops," *IEEE Transactions on Computers*, vol. 61, no. 10, pp. 1473–1483, Oct 2012.

[13] N. A. Touba and E. J. McCluskey, "Automated logic synthesis of random pattern testable circuits," in *Proceedings International Test Conference*, Oct 1994, pp. 174–183.

[14] S. R. Makar and E. J. McCluskey, "Functional tests for scan chain latches," in *Proceedings of International Test Conference (ITC)*, Oct 1995, pp. 606–615.

[15] H. C. Tsai, K.-T. Cheng, C. J. Lin, and S. Bhawmik, "A hybrid algorithm for test point selection for scan-based BIST," in *Proceedings of the 34th Design Automation Conference*, June 1997, pp. 478–483.

[16] F. Brglez, "On testability analysis of combinational networks," in *Proceedings International Symposium on Circuits and Systems*, vol. 1, May 1984, pp. 221–225.

[17] N. Tamarapalli and J. Rajski, "Constructive multi-phase test point insertion for scan-based BIST," in *Proceedings International Test Conference*, Oct 1996, pp. 649–658.

[18] J. Mahmod, S. Millican, U. Guin, and V. D. Agrawal, "Delay fault testing: Present and future," in *Proc. IEEE 37th VLSI Test Symposium (VTS)*, Apr. 2019.

[19] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a targeted translator in fortran," in *IEEE Int. Symposium on Circuits and Systems*, Jun. 1985, pp. 677–692.

[20] F. Corno, M. S. Reorda, and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results," *IEEE Design & Test of Computers*, vol. 17, no. 3, pp. 44–53, July 2000.

[21] R. Mattiuzzo, D. Appello, and C. Allsup, "Small-delay-defect testing," *EDN (Electrical Design News)*, vol. 54, no. 13, p. 28, 2009.

[22] M. Abramovici and M. A. Breuer, "On redundancy and fault detection in sequential circuits," *IEEE Transactions on Computers*, vol. C-28, no. 11, pp. 864–865, Nov 1979.