Journal of Electronic Testing
https://doi.org/10.1007/s10836-020-05859-4

Improved Pseudo-Random Fault Coverage Through Inversions: a Study on Test Point Architectures

Soham Roy[1] · Brandon Stiene[2] · Spencer K. Millican[1] · Vishwani D. Agrawal[1]

Received: 31 August 2019 / Accepted: 14 January 2020

Abstract

This article analyzes and rationalizes the capabilities of inversion test points (TPs) when implemented in lieu of traditional test point architectures. With scaling transistor density, logic built-in self-test (LBIST) quality degrades and additional efforts must keep LBIST quality high. Additionally, delay faults must be targeted by LBIST, but delay faults can be masked when using control-0/1 (i.e., traditional) TP architectures. Although inversions as TPs have been proposed in literature, the effect inversion TPs have on fault coverage compared to traditional alternatives has not been explored. This study extends work previously presented in the North Atlantic Test Workshop (NATW'19) and finds both stuck-at and delay fault coverage improves under pseudo-random tests using inversion TPs, and extended data collection finds noteworthy trends on the effectiveness of TP architectures.

Keywords Design for test · Built-in self-test · Test points · Delay test

1 Introduction

Circuit test is a critical part of the integrated circuit (IC) manufacturing process which prevents the release of defective circuits: by applying stimulus to manufactured ICs, defects created during silicon lithography are excited and detected. The cost of testing circuits is a significant portion of IC manufacturing costs [25], and as transistor density continues to scale upwards, circuit test costs are increasing and efforts continue to keep these costs down. The challenge of circuit test is reducing test-related costs while **1)** preventing the release of defective circuits whilst **2)** not discarding good devices. To obtain these goals, small increases in fault coverage are worth investment to reduce manufacturing defect levels.

Pseudo-random testing is an effective circuit testing method, both during manufacturing and for post-delivery reliability checks. 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 require less computational effort to generate and can be applied with minimal on-chip hardware. Pseudo-random tests have several additional advantages: **1)** pseudo-random tests do not require expensive automatic test equipment (ATE) since they can be applied by logic built-in self-test (LBIST) hardware, **2)** pseudo-random tests can easily be applied "at-speed" to increase delay test quality, and **3)** pseudo-random tests can be applied "in-the-field" to confirm circuit reliability.

The utility of pseudo-random tests degrades for modern technologies due to random-pattern-resistant (RPR) faults, and approaches can modify pseudo-random tests to detect such faults. RPR faults naturally occur in complex logic circuits, and since fulfilling consumer demands requires ever-more complex circuits, RPR fault density will continue to increase in new technologies. Many methods improve RPR fault coverage and reduce test lengths of pseudo-random test patterns, with a common technique

Responsible Editor: T. Xia

✉ Spencer K. Millican
millican@auburn.edu

Soham Roy
szr0075@auburn.edu

Brandon Stiene
bstiene@nvidia.com

Vishwani D. Agrawal
agrawvd@auburn.edu

[1] Auburn University, Auburn, AL 36849, USA

[2] Nvidia, Madison, AL 35748, USA

Published online: 03 February 2020

being test point (TP) insertion (TPI) [15]. Other methods of improving pseudo-random test effectiveness include changing the nature of random stimuli, such as deterministic seeding [39] and pattern weighting [28], but modifying circuit logic during test using control and observe TPs is still widely used due to their ease-of-implementation on circuit netlists.

This study analyzes the effectiveness of atypical inversion TPs in lieu of traditional control and observe TPs [2] by analyzing the stuck-at and delay fault coverages achieved using these TP architectures, which has not been examined by earlier studies on inversion TPs [3, 12, 29, 31]. Inversion TPs are not the standard TP architecture in modern literature and industrial tools, and this study will motivate their use by the electronic design automation (EDA) industry for their design for test (DFT) tools. Although inversion TPs have been introduced in previous literature [3, 12, 31], their utility compared to conventional TP architectures was not analyzed.

This article extends work presented at the North Atlantic Test Workshop (NATW'19) [30]; it was the first article to explore the effectiveness of inversion TPs at improving stuck-at and delay fault coverages compared to conventional TP architectures. The original article made the following contributions, which are reiterated in this article.

- A rationale for the effectiveness of inversion TPs compared to conventional control TPs is provided.
- Experiments empirically demonstrate inversion TPs frequently improve stuck-at fault coverage compared to conventional control TPs.
- Experiments demonstrate inversion TPs frequently improve delay fault coverage compared to conventional control TPs while not degrading stuck-at fault coverage.

Through significantly expanded data collection and a wider exploration of TP architectures, this extended article adds the following contributions:

- An environment more representative of the industrial application of TPs is implemented: this demonstrates inversion TPs remain superior under typical DFT practices.
- The fault coverage impact of observation points is explored, further supporting the fault coverage-increasing qualities of inversion TPs.

The remainder of this article is organized as follows. Motivation and a literary history for this study is described in Section 2. Inversion TPs and their function are discussed in Section 3. The experimental setup to evaluate a TP architecture's abilities is given in Section 4. Results and discussions on these experiments are given in Section 5, and conclusions and future research directions are expressed in Section 6.

2 Motivation

Pseudo-random tests are repeatable circuit stimuli generated by a pseudo-random pattern generator (PRPG) in an LBIST environment. The typical hardware schema for applying pseudo-random tests is illustrated in Fig. 1. A PRPG is typically implemented with a linear-feedback shift register (LFSR) [4], although other architectures that generate predictable stimuli can be used. To apply tests, the PRPG is first loaded with a "seed" that 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 directly observed or compressed into a "signature" generated by signal compression hardware (e.g., a multiple-input signature register (MISR) [10]). This signature is compared against a simulated value, and if the hardware signature matches the simulated signature, the circuit is considered defect-free. Although a false defect-free signature is possible (i.e., "aliasing" [19]), this occurrence is practically impossible for significantly-sized compactors.

There are many motivations for using pseudo-random tests, the first being they do not require expensive ATE to apply and can substantially reduce testing costs. ATE is manufacturing equipment that applies stimuli and measures responses. ATE requires expensive signal application and measurement equipment that applies consistent stimuli and takes accurate measurements, thus the purchase, use, and maintenance of ATE significantly impacts circuit manufacturing costs [33]. Alternatively, PRPGs can apply tests with minimal on-circuit hardware and the cost of ATE

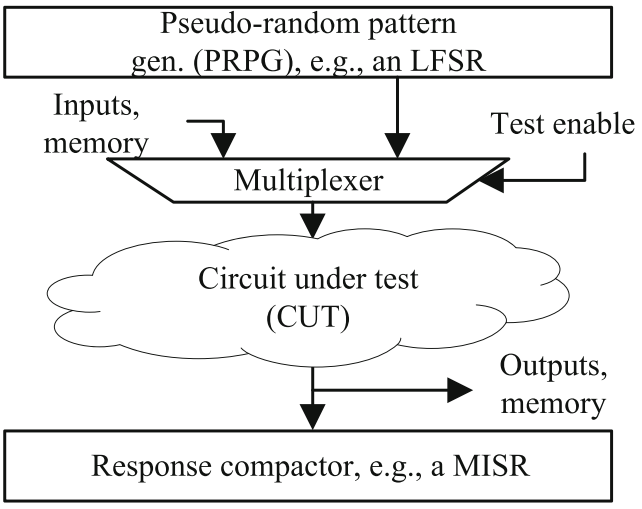

Fig. 1 The typical arrangement of pseudo-random testing hardware. While under test, memories are accessed as scannable flip-flops connected to one or more scan chains [4]. The scan chains receive inputs from a PRPG and memory states feed into a response compactor

can be eliminated (or minimal-cost ATE can be used during manufacturing).

A second advantage of pseudo-random tests is they can easily be applied "at-speed" to increase test quality. ATE applies (and observes) tests at limited speeds due to the latency of external circuit connections: this impacts test quality since tests must be applied at a circuit's designed-speed (found through delay simulation or static timing analysis) to detect delay-causing defects [21, 26]. Circuits can be modified to apply tests at-speed using external ATE [11], but these modifications degrade circuit performance and increase circuit power consumption. Since PRPGs are implemented on-circuit, they can more easily apply tests at-speed and detect delay-causing defects [27]. However, the use of PRPGs for at-speed test requires caution: PRPGs may excite non-functional false timing paths, and if the circuit runs faster than these non-functional paths allow, random patterns may cause all functionally-good circuits to fail the test [17].

A third advantage of using PRPGs for tests is since they do not require external ATE, they can be used "in-the-field" to check a circuit's reliability. Post-manufacturing degradation failures and temporary "soft errors" have been observed for modern technologies [14, 34, 35], which concerns life-critical applications (e.g., medical, transport, etc.). Therefore, high-quality tests must be applied periodically after manufacturing. Doing this with ATE is impossible due to its size and cost, but pseudo-random tests can be run in-the-field with ease: test hardware is built into the circuit and can be activated by programming the appropriate signals.

The effectiveness of pseudo-random tests is impaired by the presence of RPR faults, which naturally occur in complex logic circuits. RPR faults [13] are logical representations of defects unlikely to be detected using random stimuli, i.e., RPR faults are detected by a small set of test vectors among all possible stimuli. Although the probability of detecting RPR faults is improved by applying more test vectors, the number of vectors required may be infeasibly large. A typical example of such a fault is illustrated in Fig. 2: exciting and observing the indicated fault requires 32 logic-0s to be applied to the OR gate and 32 logic-1s to be applied to the AND gate, and presuming all inputs have an equal probability of being logic-0 or logic-1, the probability of this occurring is $2^{-64} \approx 5.4 \cdot 10^{-20}$, which is not feasible using random stimuli.

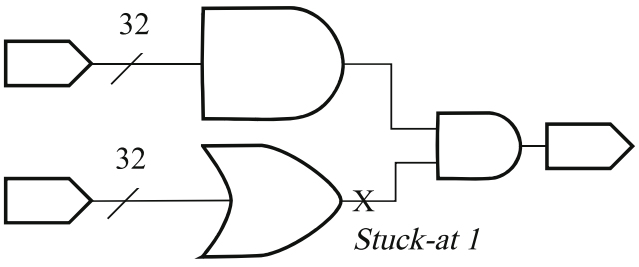

Fig. 2 An example of a fault requiring one specific vector to detect, which is unlikely to be generated using random stimuli

2.1 Test Point Insertion (TPI)

TPs are circuit modifications that increase fault detection probabilities, and the process of inserting TPs is named TPI. TPs modify circuits during test (but do not change the circuit function outside of test) and make RPR faults easier to excite and observe. Since TPs create undesirable design overheads (added circuit area, power, and delay), TPI methods attempt to select TP locations and TP types to increase fault coverage as much as possible while minimizing the number of TPs. This constraint can be expressed as, "maximize fault coverage given TP limits," or "minimize TPs given fault coverage limits," but an algorithm which performs one can easily be converted to perform the other. This study performs the former.

Many TPI algorithms have been proposed in literature, and TPI is typically performed with the following steps. First, fault simulation is performed on the circuit to identify RPR faults. This simulation replicates a sub-set of vectors to be applied to the circuit, but additional vectors will be applied to the circuit with TPs enabled. Second, TPs are iteratively selected for insertion using a TP evaluation method (see below). Third, selected TPs are inserted in the post-synthesis circuit netlist. Alternatively, TPs can be inserted during or before logic synthesis at the expense of synthesis complexity [37].

TPI algorithms targeting RPR faults can be divided into two general categories, the first being methods using fault simulation. After performing fault simulation, any faults not yet detected can be labeled as RPR, and several TPI algorithms take advantage of this to target such faults. These algorithms use heuristics to target not-yet-detected faults [18], with some generating and using additional statistics during simulation, e.g. signal probabilities [7]. Although such TPI methods were effective in previous generations of technologies, these methods have fallen out of favor due to the increasing complexity of fault simulation. However, it is common practice to perform fault simulation before TPI: TPI is not required if adequate fault coverage is obtained, and other TPI methods should not target faults easily detected through fault simulation.

A second category of TPI algorithms use circuit testability calculations, e.g. COP [5], to choose TP locations [32, 38, 41]. In lieu of performing time-consuming fault simulation, these TPI algorithms "calculate" each TP's impact on fault coverage and iteratively insert the "best" TP, i.e., the TP increasing fault coverage the most. Fault

coverage calculations are known to be less accurate than fault simulation, but performing these calculations requires orders of magnitude less time, and therefore calculations can be repeated for every candidate TP.

2.2 Conventional TP Architectures

Control TPs attempt to make the excitation and observation of faults more likely under random stimulus by forcing logic values during test. Signal-controlling TPs use an extra *test enable* (*TE*) pin (or scannable register [40]) to force logic in a circuit to a controlled value. While not under test, *TE* is disabled, which leaves the function of the circuit unchanged. When *TE* is asserted, hard-to-control signals are directly forced to pre-determined values. Typical control TPs use AND gates or OR gates (or other analogous logic-forcing structures) to force circuit lines to logic-0/1 (respectively) during test [28], as illustrated in Fig. 3a and b. The first input to control TPs is the circuit line to be forced during test, and the second input is the *TE* signal. *TE* may be partially enabled (i.e., a subset of TPs may be active during a subset of tests [28, 36]) in order to substantially reduce TP area overhead [16], but architectures and selection methods for controlling *TE* are outside the scope of this study.

Forcing circuit lines during test has two effects: **1)** RPR faults can be easier to excite under random stimulus, and **2)** RPR faults can be propagated to circuit outputs by activating circuit paths. Although control TPs increase circuit area and delay with their extra gates, these overheads are worthwhile given their positive impacts on fault coverage [37].

A second category of TPs, *observe TPs*, directly observe circuit lines as opposed to controlling them. To make a circuit line easier to observe, the line is diverted to circuit outputs (or scannable latches [22]) made specifically for test, as shown in Fig. 3c.

Observe TPs were not addressed by the authors' original article [30], but this extended article explores their impact along with the proposed TP architecture. The detriments described in the following section were initially limited to control TPs, thus the presence of observe points were viewed as a constant and their effect on fault coverage was not explored. This choice to forgo exploring observe points was also motivated by the lack of computational resources at the time of the original study. Additionally, future research directions will address other issues involving observe TPs (see Section 5).

2.3 Conventional TP Detriments

Although control TPs can excite "stuck values" (i.e., "stuck-at faults") in circuits, their use creates unintended consequences. Since active control TPs force lines to a single value, only one stuck-at value (stuck-at 0 or stuck-at 1) can be excited when a control TP is active. Also, active control TPs prevent logic on controlled signals from passing through the TP. This latter effect prevents faulty values on the controlled line from being observed. Although control TPs intend to increase stuck-at fault coverage, these qualities degrade their ability to excite and propagate such faults.

Another disadvantage of control TPs is they prevent signal transitions: this blocks the transmission and excitation of delay faults. In a circuit's functional mode, delay-causing defects can cause incorrect values to be observed/captured at circuit outputs/latches, as illustrated in Fig. 4. Unlike stuck-at faults, delay faults require two input vectors to detect: one to activate faults and the other to launch slow transitions. Control TPs prevent signal transitions when enabled (since they force signals to a value), and therefore control TPs 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 controlled lines less likely to be excited. This latter detriment is partially remedied by conditionally-active TPs [28, 36], but such architectures are outside the scope of this study and create other detriments (see Section 6).

Although observe TPs do not possess the fault-blocking effects of control TPs (and in fact, using only observe TPs can never decrease stuck-at fault coverage), observe TPs cannot detect RPR faults that are difficult to excite and may

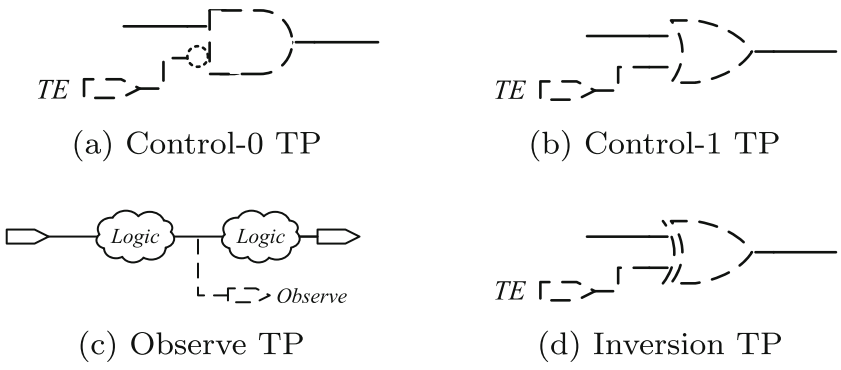

Fig. 3 Test points (TPs) force signals to logic-0 (control-0), force signals to "logic-1" (control-1), observe signals directly (observe), or invert signals (inversion). All TPs require an additional pin/latch for a *test enable* (*TE*) or *observe* signal. Circuit modifications are shown in dashed lines

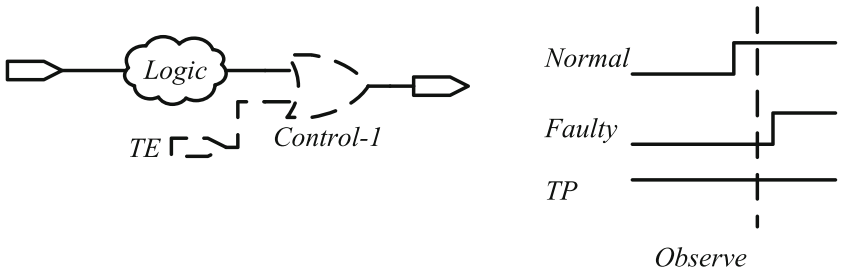

Fig. 4 Here, the effect a control TP has on the excitation and observation of a delay fault is illustrated. The output waveforms of a normal circuit, a faulty circuit, and a circuit with an active control TP are labeled, as is the point in time where an observation occurs

not provide the best TP solution. Additionally, control TPs placed at strategic locations can significantly increase fault observability by "unblocking" logic (as illustrated in Fig. 5) while simultaneously exciting hard-to-detect faults. Given that control TPs can block logic while observe TPs may fail to excite logic, a goal of the proposed architecture (see Section 3) is to simultaneously perform both functions.

Although typical test procedures will apply tests with TPs both enabled and disabled, removing disadvantages from 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.

3 Inversion-based TP Architecture

Unlike conventional control TP architectures discussed in Section 2.2, *inversion TPs* change signal value probabilities through inversions as opposed to forcing circuit lines to pre-determined 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%: this creates the detriments discussed in Section 2.3. Alternatively, inversion TPs invert signal probabilities. For example, if a circuit line has a 75% chance of being logic-1 without a TP present, activating an inversion TP on this line will change the probability of logic-1 to 25%.

The implementation of inversion TPs is illustrated in Fig. 3d. Inversion TPs are implemented using XOR gates

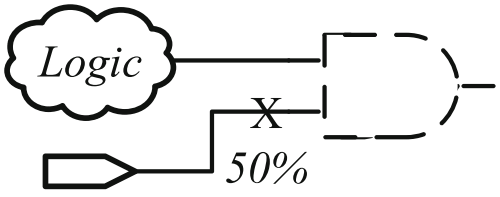

Fig. 5 In this example circuit, inversion TPs may not increase fault coverage: if the best TP solution is to force the circuit input to logic-1 during test, an inversion TP will fail to do so

connected to *TE* with the other input connected to the inverted signal. When the TP is disabled, the XOR gate functions as a buffer, thus allowing the original circuit value to pass unchanged. When the TP is enabled with *TE*, the XOR functionally becomes an inverter. Like control TPs, the source of *TE* signal can be a scanned register or a circuit-level pin.

The advantages of inversion TPs over conventional control TPs are **1)** their ability to excite both stuck-at 0 and stuck-at 1 faults when activated and **2)** their ability to allow stuck-at faults on inverted lines to propagate through TPs. Since these TPs do not force logic to set values, it is possible to excite both logic-0 and logic-1 when inversion TPs are activated. Additionally, faulty values on the inverted line are no longer forced to a given value but are instead allowed to pass through the TP (albeit inverted), which in turn allows faults to be observed.

An additional 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 allow transitions on TP outputs, and these transitions can excite delay faults on the TP output (or on logic driven by the TP). Additionally, if transitions (from delay faults) occur on the inverted line, these transitions can pass through the inversion TP (albeit inverted) and can be observed.

Inversion TPs may falter when tests need to be "less random", but finding if such occurrences degrade inversion TP performance requires empirical analysis (which was first explored in [30] and is significantly expanded in this article). To detect faults under random stimuli, relatively random signals (i.e., signals which are logic-1 50% of the time) may need to be forced to specific values. For example, as illustrated in Fig. 5, an AND gate may prevent faulty logic from passing based on another signal's value: forcing this signal to logic-1 allows values to pass through the gate. If the controlled signal is normally random, this cannot be accomplished with an inversion TP. Whether such conditions are likely to occur will be explored though empirical data (see Section 5).

Using inversions as TPs has been noted in previous studies [3, 12, 29, 31], but their advantages compared to conventional TPs and their effect on delay fault coverage have only been explored in the authors' original article [30].

The focus of [3] was broad and covered many topics in weighted random pattern generation with a single section devoted to TPI. [3] chose to implement control-0 and control-1 TPs, but these TPs were implemented as XOR gates assuming TP locations were almost always logic-0/1 under random stimuli. Beyond this assumption, the effect of implementing control-0/1 TPs in this manner was not explored, and neither was the effect this TP architecture had on delay fault coverage. Also, this assumption contradicts the observation made in the previous paragraph. Using XORs as TPs was allowed in [12, 29, 31], but these studies did not compare inversion TPs against conventional control TPs, nor did they analyze TPs' effect on delay fault coverage.

A possible detriment of using inversion TPs is they can add more delay compared to alternative TP architectures, but balancing the delay-causing impacts of inversion TPs with their fault coverage-increasing benefits is left to future studies. Previous studies noted inversion TPs may not be allowable if circuit performance is a concern to circuit designers [29]: this is because the XOR gates which implement inversions will add more delay compared to the AND/OR gate counterparts of control-0/1 TPs. However, the added delay of inversion (and control) TPs may be irrelevant if TPs are not added to critical paths, which TPI from literature account for [8]. The delay impact of inversion TPs is not the immediate concern of this article, but the authors intend to add inversion TPs while not impacting delay performance in future studies (see Section 6).

4 Experimental Setup

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

4.1 TPI for Architecture Comparison

This study does not propose a new TPI algorithm: instead, an established TPI algorithm from literature will compare TP architectures [38]. Tsai et al. [38] is chosen because it can insert any TP type without favoring one architecture over another: the algorithm does not require specific information on the TPs used and instead only knows the calculated effect a circuit modification will have on fault coverage. In this algorithm, the controllability and observability program (COP) [5] calculates fault coverage using probabilistic metrics. Many TPI algorithms use COP [29, 38, 40] to calculate the fault coverage of a circuit (FC) by calculating the probability that stuck-at 0 (SA0) or stuck-at 1 (SA1) faults will be detected (P_f):

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

$$FC = \frac{1}{|F|} \sum_{\forall f \in F} P_f$$

The implemented TPI algorithm starts by performing testability analysis [5] 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) values to all lines in a circuit. These controllability and observability values are used to predict the fault coverage of the circuit using the above equations. It should be noted that this calculated fault coverage is not the actual circuit fault coverage: controllability and observability calculations are not precise when re-convergent fan-outs are present in a circuit [5]. The TPI algorithm iteratively calculates the impact each candidate TP will have on fault coverage by re-calculating controllability and observability values when the TP is active. The TPI program then inserts the TP with the highest positive impact on the calculated fault coverage. This process is repeated until a given TP hardware limit is reached (expressed as a conservative TP limit, as correlating the number of inserted TPs to hardware overhead is difficult [16]), 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 from literature can insert inversion TPs, it is presumed that the TPI algorithm used will not impact this study's conclusions if the TPI method does not give an advantage to any TP architecture.

For this study, three separate categories of TPs are considered (control-0/1, inversion, and observe), and the impact of allowing most combinations of these TPs is studied. This is an extension to the original article [30] where only inversion-and-observe and control-and-observe combinations were considered, and noteworthy results are produced through this extension (see Section 5). Candidate TPs include control-0, control-1, inversion, and observe TPs on the input and output of every gate.

In this study, every TP has the same *TE* signal, but the use of this signal is altered from the original study [30]. In the original study [30], this signal was enabled for every vector applied, but this does not accurately reflect an industrial use of TPs: instead, this signal will be enabled for half of all vectors applied. In literature, different TPs can have different *TE* signals which are not uniformly on/off [28, 36], but the impacts of such *TE* controls are not the scope of this study.

To model delay faults, this study uses the transition delay fault (TDF) model and tests are applied using a launch-off-shift methodology (i.e., both circuit inputs and scannable latches are fed by a 64-bit LFSR). The TDF model is chosen because of its ease of implementation and its known ability to model circuit defects [20].

4.2 Execution

All programs (i.e., TPI and fault simulation) are executed on high-performance workstations representative of industrial development environments. Workstations are equipped with Intel i7-8700 processors and 8GB of RAM. Software was written in C++ and complied using the MSVC++14.15 compiler with maximum optimization parameters. Source code for executables is provided by the authors through an open-source license [24].

Original TPI and fault simulation programs were written for this study. This choice was made in lieu 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 favorable result towards conventional TPs. This contrasts with the implementation of [38], which does not have knowledge of TP types and instead only predicts a circuit modification's impact on fault coverage

Table 1 gives benchmark circuit information, including the number of logic gates ("Gates") in the ISCAS'85 [6] and ITC'99 [9] benchmarks. Benchmarks which achieved 100% fault coverage without performing TPI after simulating 10,000 vectors (b01, b02, b06, and c17) are omitted. Otherwise, the fault coverages given are after simulating 10,000 vectors, and TPs are active for half of these vectors.

5 Results and Discussion

Table 1 presents the number of TPs inserted ("Inserted TPs") by the described TPI algorithm (see Section 4.1). Columns "C,I,O", "C/O", "I/O", and "C/I/O" give the number of TPs inserted: the first column ("C,I,O") represents only a single TP type being allowed, while other columns represent multiple TP types allowed. When multiple TP types are allowed, the number of individual types of TPs inserted are separated by '/'. It is noteworthy that, with a single exception (c6288), when any TP type is available ("C/I/O"), the number of inversion TPs inserted is always greater than or equal to the number of control or observe TPs inserted: this implies that inversion TPs were more frequently calculated to increase fault coverage more than alternatives, which supports the architecture's qualities discussed in Section 3.

Columns "Orig.", "C", "O", "I", "C/O", I/O" and "CIO" under the headings "SAF/TDF Coverage (%)" give the stuck-at fault and transition delay fault coverage after simulating 10,000 random vectors (or vector pairs for TDF coverage) in the original circuit and with the corresponding TPs inserted. TP combinations having the highest SAF/TDF coverage are underlined. Half of applied vectors have TPs enabled, and the other half have TPs disabled, as is typical of industrial settings.

The first noteworthy result from Table 1 is inserting solely inversion TPs ("I") most frequently obtains the highest stuck-at and delay fault coverage. For twelve out of twenty-two benchmarks, inversion test points obtain the highest stuck-at fault coverage, and inversion TPs obtain the highest delay fault coverage for eleven out of twenty-two benchmarks. These trends support the qualities of inversion TPs expressed in Section 3.

A second observation from fault coverage results, which complements results of the original study [30], is although the delay fault coverage obtained by the various TPI methods increases, inversion TPs most frequently increase delay fault coverage the most. The original study [30] concluded that delay fault coverage decreases when control TPs are present: this was found by comparing the TP combinations "I/O" and "C/O". However, the original study presumed TPs were active through the entire 10,000 vector test, which is not representative of industrial environments. When the original results [30] are combined with these extended results, one can conclude that control TPs are detrimental to delay fault coverage when *active*, which supports the detriments of control TPs expressed in Section 2.3. In fact, adding only control TPs decreases stuck-at for one benchmark (b03) and decreases delay fault coverage for two (b03 and c6288).

An unexpected third observation not made in the original study [30] is the effects observation TPs have on fault coverage. First, when observe TPs are allowed by themselves ("O"), they rarely obtain the highest fault coverage (only for c2670 and c7553, with the latter only for delay fault coverage). Second, allowing observe TPs to "complement" control/inversion TPs infrequently increases fault coverage, but instead more frequently decreases it. This is best illustrated by plotting the change in fault coverage (Δ Stuck-at Fault Coverage) obtained when observe points are allowed, which is illustrated in Fig. 6: the plot of delay fault coverage is omitted since it follows a nearly identical trend. Allowing more TPs to decrease fault coverage is a counterintuitive result, but an explanation exists: observe TPs can only increase observability, but control and inversion TPs can increase both controllability and observability.

Table 1 Effect of test points on PRPG pattern fault coverage in benchmark circuits

Circuit details		Inserted TPs				Fault Coverage												
						SAF Coverage (%)							TDF Coverage (%)					
Bench.	Gates	C,I,O	C/O	I/O	C/I/O	Orig.	O	C	I	C/O	I/O	C/I/O	Orig.	O	C	I	C/O	I/O	C/I/O
b03	190	1	0/1	0/1	0/1/0	73.10	73.10	71.43	81.19	73.10	73.10	81.19	77.11	77.11	75.45	83.58	77.11	77.11	83.58
b04	803	8	4/4	0/8	0/4/4	48.40	64.90	56.79	66.50	69.52	64.90	70.66	49.71	66.00	58.15	67.38	69.93	66.00	70.91
b05	1,032	10	7/3	6/4	4/5/1	41.89	58.15	55.82	61.45	63.03	64.26	53.34	44.22	61.22	58.12	64.36	66.36	67.29	54.54
b07	490	4	0/4	0/4	2/2/0	65.96	70.90	73.02	77.87	70.90	70.90	72.31	70.89	75.37	76.37	81.26	75.37	75.37	75.37
b08	204	2	0/2	0/2	0/1/1	42.59	56.79	50.00	57.61	56.79	56.79	59.26	45.03	58.67	50.77	60.20	58.67	58.67	62.37
b09	198	1	0/1	0/1	0/1/0	53.05	56.34	62.91	62.91	56.34	56.34	62.91	53.68	57.65	62.32	62.32	57.65	57.65	62.32
b10	223	2	0/2	0/2	0/1/1	75.99	77.78	77.42	78.14	77.96	77.78	79.03	79.82	81.49	81.49	81.93	81.60	81.49	82.93
b11	801	8	3/5	3/5	1/4/3	42.78	57.44	56.06	71.55	66.98	62.57	71.06	45.71	61.67	59.95	75.81	71.65	66.99	74.95
b12	1,197	11	0/11	0/11	3/5/3	70.46	73.62	77.82	77.85	73.62	73.62	74.63	73.03	75.88	79.73	79.63	75.88	75.88	76.99
b13	415	4	0/4	0/4	2/2/0	69.23	73.98	77.60	83.26	73.98	73.98	74.55	72.44	76.95	80.10	84.88	76.95	76.95	76.68
b14	10,343	2	1/1	1/1	1/1/0	17.83	20.27	42.05	42.05	37.99	37.99	42.05	20.13	22.50	46.80	46.80	42.82	42.82	46.80
b15	9371	3	0/3	0/3	1/2/0	24.61	26.95	41.34	41.31	26.95	26.95	36.39	27.79	30.35	45.69	45.66	30.35	30.35	40.34
c432	203	2	0/2	0/2	1/1/0	91.73	92.10	92.46	93.01	91.73	91.73	92.46	93.29	93.63	93.87	94.56	93.29	93.29	93.87
c499	275	2	1/1	1/1	0/1/1	93.94	95.62	94.61	95.79	95.62	95.79	94.95	95.09	96.29	95.69	96.69	96.49	96.49	95.89
c880	469	4	0/4	0/4	0/2/2	58.25	59.36	60.66	60.87	59.36	59.36	60.06	65.74	66.88	68.07	68.18	66.88	66.88	67.56
c1355	619	6	0/6	0/6	2/3/1	88.75	90.30	93.14	95.67	90.30	90.30	92.58	90.89	92.18	94.54	96.61	92.18	92.18	94.13
c1908	938	9	0/9	0/9	0/5/4	69.65	73.15	86.33	69.55	73.01	73.10	72.18	77.52	80.82	89.54	77.49	80.74	80.79	79.95
c2670	1,566	15	0/15	0/15	0/8/7	66.22	68.92	66.66	67.26	68.86	68.89	67.40	66.89	70.04	67.32	68.05	70.04	70.02	68.39
c3540	1,741	17	1/16	0/17	2/9/6	46.45	49.44	50.56	48.77	50.11	49.36	52.49	49.44	52.73	53.55	51.86	53.35	52.70	55.59
c5315	2,608	26	21/5	21/5	2/13/11	63.93	74.52	71.99	75.62	76.50	76.50	77.34	68.49	79.98	76.18	80.38	81.14	81.45	82.31
c6288	2,480	7	4/3	4/3	7/0/0	69.83	69.83	69.83	70.78	69.86	70.56	69.83	74.19	74.19	74.16	74.97	74.19	74.78	74.16
c7552	3,827	13	0/13	0/13	0/7/6	77.66	79.62	79.10	79.01	79.62	79.62	79.64	81.52	83.30	82.87	82.79	83.30	83.30	83.27

Fig. 6 When observe points are allowed with control ("C+O") and inversion ("I+O") TPs, post-TPI fault coverage most frequently decreases significantly. Although not plotted, delay fault coverage follows a nearly identical trend

A final corollary result, plotted in Fig. 7, is the time to perform TPI using the combinations of TPs. The original study [30] concluded using inversion TPs significantly decreases TPI time (regardless of the TPI algorithm used) because there are fewer TPs to evaluate: each line in a circuit has two possible control TPs to evaluate (control-0 and control-1) while there is only one inversion TP to evaluate. Showing the time required to perform TPI (normalized to inversion-only TPI time, listed in Fig. 7) shows a similar trend, except using only observe points is most frequently the fastest method. The TPI time of observe TPs only ("O") being the fastest can be explained: inserting an observe TPs will only change circuit observability values, so the TPI algorithm does not need to recalculate controllabilities. Given inversion TPs only ("I") frequently obtain the highest fault coverage while also inserting TPs in nearly the smallest amount of time, this implies only using inversion TPs will yield the most favorable fault coverage results with minimal TPI effort.

6 Conclusion and Future Directions

This article presented the impacts of different TP architectures on stuck-at and delay fault coverage, most notably inversion TPs. Results showed inversion TPs frequently increase stuck-at fault coverage more than conventional TP alternatives while simultaneously increasing delay fault coverage. Additionally, it was shown that the time required to insert inversion TPs compared to alternative TP architectures is significantly less, which further encourages their use.

Although inversion TPs performed admirably in this study, some data points motivate the authors to further quantify circuit qualities to improve TP selection. This is best shown through the benchmark c1908: control TPs only ("C") provided delay and stuck-at fault coverage significantly higher than any other TP architecture. Although this can be viewed as an anomaly, quantifying this anomaly and using this measurement in TPI algorithms may significantly increase TP quality.

Observe and control TPs must be a focus of future studies, as their true impact on delay fault detection is not yet known. Observe TPs shorten circuit timing paths and will not capture small delay defects requiring full circuit paths to be observed. Likewise, control TPs with controlled *TE* signals [28, 36] (as opposed to always-active *TE* signals) will excite delay faults before typical signal arrivals times. In this study, delay faults were modeled as TDFs due to their ease of implementation and their ability to model known

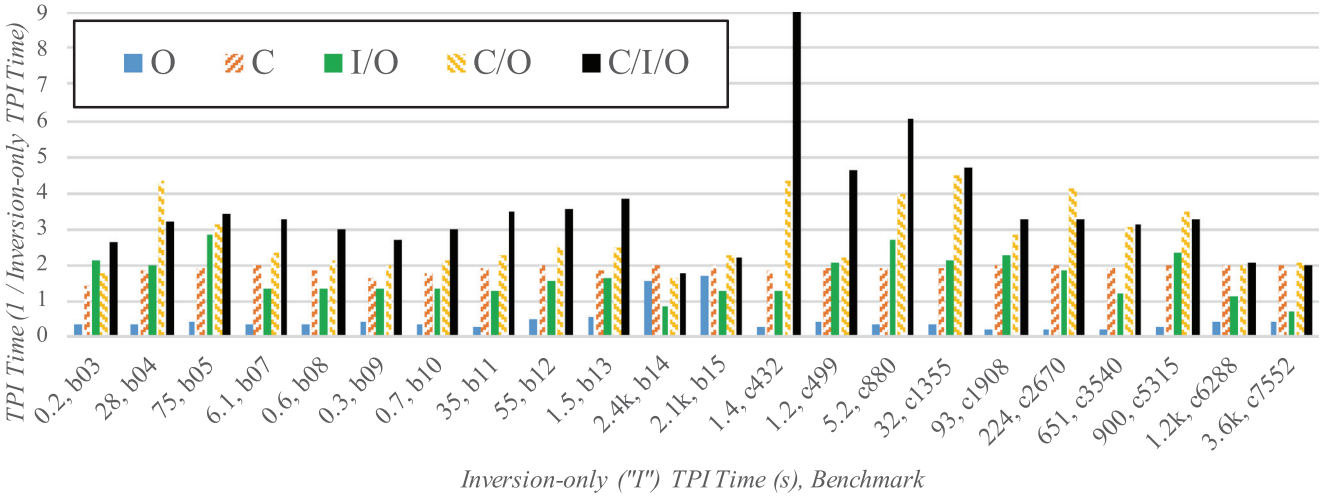

Fig. 7 When TPI time is normalized to inversion-only TPI time ("I"), inserting only observation points ("O") is the only TP combination which performs TPI faster

defects (i.e., stuck-open faults) [20], but if the fault model includes the magnitude of a delay (as is the case of small delay defects [23]), the presence of observe and control TPs may decrease delay fault coverage. Future endeavors will explore the impact conventional TP architectures have on these defects and will explore architectures which can remedy their detriments.

As was noted in Section 3, the use of inversion TPs may unfavorably impact circuit timing performance, and the trade-off between increased fault coverage and degraded performance must be explored. It may be possible to insert inversion TPs outside of critical timing paths while still obtaining favorable fault coverage, but this will be explored with future empirical data. Additionally, TPI algorithms that consider delay impacts [8] will be implemented to observe if the benefit of inversion TPs hold under timing constraints.

Another future avenue of research is the impact TPs have on detecting redundant faults and producing false failures. In this study, fault simulation did not remove redundant faults [1], 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 not detecting redundant faults.

References

1. Abramovici M, Breuer MA (1979) On redundancy and fault detection in sequential circuits. IEEE Transactions on Computers C-28(11):864–865
2. Acero C, Feltham D, Liu Y, Moghaddam E, Mukherjee N, Patyra M, Rajski J, Reddy SM, Tyszer J, Zawada J (2017) Embedded deterministic test points. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 25(10):2949–2961
3. Bakshi D (2012) Techniques for seed computation and testability enhancement for logic built-in self test. Master's thesis, Virginia Tech
4. Bardell PH, McAnney WH, Savir J (1987) Built-in test for VLSI: pseudorandom techniques. Wiley-Interscience, New York
5. Brglez F (1984) On testability analysis of combinational networks. In: Proceedings of the international symposium on circuits and systems (ISCAS), vol 1, pp 221–225
6. Brglez F, Fujiwara H (1985) A neutral netlist of 10 combinational benchmark circuits and a targeted translator in fortran. In: Proceedings of the IEEE Int. symposium on circuits and systems (ISCAS), pp 677–692
7. Briers AJ, Totton KAE (1986) Random pattern testability by fast fault simulation. In: Proceedings of IEEE international test conference (ITC)
8. Cheng K-T, Lin C-J (1995) Timing-driven test point insertion for full-scan and partial-scan BIST. In: Proceedings of the IEEE international test conference (ITC), pp 506–514
9. Corno F, Reorda MS, Squillero G (2000) RT-level ITC'99 benchmarks and first ATPG results. IEEE Design & Test of Computers 17(3):44–53
10. David R (1986) Signature analysis for multiple-output circuits. IEEE Trans Comput 35(9):830–837
11. Dervisoglu BI, Stong GE (1991) Design for testability using scanpath techniques for path-delay test and measurement. In: Proc IEEE International Test Conference, pp 365–374
12. Fang Y, Albicki A (1995) Efficient testability enhancement for combinational circuit. In: Proceedings of international conference on computer design (ICCD), pp 168–172
13. Geuzebroek MJ, van der Linden JT, van de Goor AJ (2002) Test point insertion that facilitates ATPG in reducing test time and data volume. In: Proceedings of the IEEE international test conference, Washington, DC, USA, pp 138–147
14. Ghani T, Mistry K, Packan P, Thompson S, Stettler M, Tyagi S, Bohr M (2000) Scaling challenges and device design requirements for high performance sub-50 nm gate length planar CMOS transistors. In: Proc Symposium on VLSI Technology, pp 174–175
15. Hayes JP, Friedman AD (1974) Test point placement to simplify fault detection. IEEE Trans Comput C-23(7):727–735
16. He M, Gustavo K, Contreas, Tran D, Winemberg L, Tehranipoor M (2017) Test-point insertion efficiency analysis for LBIST in high-assurance applications. IEEE Transactions on Very Large Scale Integration 25(9)
17. Higgins FP, Srinivasan R (2000) BSM2: Next generation boundary-scan master. In: Proc 18th IEEE VLSI Test Symposium (VTS), pp 67–72
18. Iyengar VS, Brand D (1989) Synthesis of pseudo-random pattern testable designs. In: Proceedings of the international test conference, pp 501–508
19. Karpovsky MG, Gupta SK, Pradhan DK (1991) Aliasing and diagnosis probability in misr and stumps using a general error model. In: Proceedings of the international test conference, Nashville, TN, pp 828–839
20. Mahmod J, Millican SK, Guin U, Agrawal VD (2019) Special session: delay fault testing - present and future. Proceedings of the 37th VLSI Test Symposium (VTS), Monterey, CA
21. Majhi AK, Agrawal VD (1998) Delay fault models and coverage. In: Proc 11th international conference on VLSI Design, Chennai, India, pp 364–369
22. Makar SR, McCluskey EJ (1995) Functional tests for scan chain latches. In: Proceedings of international test conference (ITC), pp 606–615
23. Mattiuzzo R, Appello D, Allsup C (2009) Small-delay-defect testing. EDN (Electrical Design News) 54(13):28
24. Millican SK (2019) OpenEDA. Online. Available: https://github.com/vlsi-test-lab/OpenEDA
25. Nag PK, Gattiker A, Wei S, Blanton RD, Maly W (2002) Modeling the economics of testing: a DFT perspective. IEEE Design & Test of Computers 19(1):29–41
26. Nigh P, Needham W, Butler K, Maxwell P, Aitken R (1997) An experimental study comparing the relative effectiveness of functional, scan, IDDq and delay-fault testing. In: Proc 15th IEEE VLSI Test Symposium, pp 459–464
27. Pateras S (2003) Achieving at-speed structural test. IEEE Design and Test of Computers 20(5):26–33
28. Rajski J, Tyszer J (1998) Arithmetic built-in self-test for embedded systems. Prentice-Hall Inc., Upper Saddle River

29. Ren H, Kusko M, Kravets V, Yaari R (2009) Low cost test point insertion without using extra registers for high performance design. In: Proceedings of the International Test Conference (ITC), Austin, TX
30. Roy S, Stiene B, Millican SK, Agrawal VD (2019) Improved random pattern delay fault coverage using inversion test points. In: IEEE 28th North Atlantic Test Workshop (NATW), pp 206–211
31. Rudnick EM, Chickermane V, Patel JH (1994) An observability enhancement approach for improved testability and at-speed test. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 13(8):1051–1056
32. Savaria Y, Youssef M, Kaminska B, Koudil M (1991) Automatic test point insertion for pseudo-random testing. In: Proceedings of the IEEE international sympoisum on circuits and systems (ISCAS), vol 4, pp 1960–1963
33. Sayil S (2018) Conventional test methods. In: Contactless VLSI measurement and testing techniques. Springer, pp 1–7
34. Sootkaneung W, Howimanporn S, Chookaew S (2018) Temperature effects on BTI and soft errors in modern logic circuits. Microelectronics Reliability 87:259–270
35. Takeda E, Suzuki N (1983) An empirical model for device degradation due to hot-carrier injection. IEEE Electron Device Letters 4(4):111–113
36. Tamarapalli N, Rajski J (1996) Constructive multi-phase test point insertion for scan-based bist. In: Proceedings of the International Test Conference (ITC), pp 649–658
37. Touba NA, McCluskey EJ (1994) Automated logic synthesis of random pattern testable circuits. In: Proceedings of the IEEE international test conference (ITC), pp 174–183
38. Tsai HC, Cheng K-T, Lin CJ, Bhawmik S (1997) A hybrid algorithm for test point selection for scan-based BIST. In: Proceedings of the 34th design automation conference (DAC), pp 478–483
39. Xiang D, Wen X, Wang L (2017) 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 25(3):942–953
40. Yang J, Touba NA, Nadeau-Dostie B (2012) Test point insertion with control points driven by existing functional flip-flops. IEEE Trans Comput 61(10):1473–1483
41. Youssef M, Savaria Y, Kaminska B (1993) Methodology for efficiently inserting and condensing test points (cmos ics testing). IEE Proceedings-E (Computers and Digital Techniques) 140(3):154–160

Soham Roy received his B.Tech. degree in Electronics and Instrumentation from West Bengal University of Technology, Kolkata, India, in 2011. He was with Wipro Ltd., VLSI Division, Bangalore, India, as a design for test engineer from 2011–2015. He received his MS degree from the Department of Electrical and Computer Engineering, Technical University of Dresden, Dresden, Germany, in 2018. He is currently a PhD candidate at Auburn University, and his research interest includes VLSI design and test, artificial intelligence, and quantum computing.

Brandon Stiene received his BS degree from Auburn University in 2018 in Electrical and Computer Engineering. He is currently a Verification Engineering with Nvidia in Madison, AL, USA.

Spencer Millican received his PhD, MS, and BS degrees from the University of Wisconsin - Madison in 2015, 2013, and 2011, respectively. He was with IBM in Rochester, MN, USA as a design for test engineer for 2 years and is currently an assistant professor at Auburn University. He has published several articles, received a best paper award at the 2014 IEEE International Conference on VLSI Design, and has received patents in the field of encrypted circuit simulation. His research interests include logic built-in self-test for modern technologies and encrypted circuit implementation and simulation.

Vishwani Agrawal is an Emeritus Professor in the Department of Electrical and Computer Engineering at Auburn University, Alabama, USA. Prior to retirement in 2016, he was the James J. Danaher Professor in the same department. He has over 45 years of industry and university experience, working at Auburn University, Bell Labs, Murray Hill, NJ, USA; Rutgers University, New Brunswick, NJ, USA; TRW, Redondo Beach, CA, USA; Indian Institute of Technology Delhi (IITD), New Delhi, India; EG&G, Albuquerque, NM, USA; and ATI, Champaign, IL, USA. His areas of expertise include VLSI testing, low-power design, and microwave antennas. He obtained a BE (1964) degree from the Indian Institute of Technology Roorkee (IITR), Roorkee, India; ME (1966) from the Indian Institute of Science, Bangalore, India; and PhD (1971) from the University of Illinois at Urbana-Champaign (UIUC), IL, USA. He has coauthored over 400 papers and 5 books. He holds 13 United States patents. He is the Editor-in-Chief of the Journal of Electronic Testing: Theory and Applications, and a past Editor-in-Chief (1985–87) of the IEEE Design & Test of Computers magazine. His many invited talks include the plenary (1998) at the International Test Conference, Washington, DC, USA and the keynote (2012) at the 25th International Conference on VLSI Design, Hyderabad, India. He served on the Board of Governors (1989–90) of the IEEE Computer Society, and in 1994 chaired the Fellow Selection Committee of that Society. He has received ten Best Paper Awards, two Lifetime Achievement Awards, and two Distinguished Alumnus Awards. Agrawal is a Fellow of the ACM, IEEE and IETE-India. He has served on the Advisory Boards of the ECE Departments at UIUC, New Jersey Institute of Technology (NJIT), and the City College of the City University of New York (CCNY). See his website: http://www.eng.auburn.edu/vagrawal.