Special Session: Survey of Test Point Insertion for Logic Built-in Self-test

Yang Sun, Spencer K. Millican, and Vishwani D. Agrawal
Department of Electrical and Computer Engineering, Auburn University
341 War Eagle Way, Auburn, AL 36849-5201
yzs0057@auburn.edu, millican@auburn.edu, agrawvd@auburn.edu

***Abstract*–This article surveys test point (TP) architectures and test point insertion (TPI) methods for increasing pseudo-random and logic built-in self-test (LBIST) fault coverage. We present a history of TPI approaches, including TPI for increasing stuck-at fault coverage, compressing test patterns, detecting path delay faults, and reducing test power. We discuss some known weaknesses of TPs and explore research directions to overcome them.**

Index Terms–survey, test points, test point insertion, built-in self-test, path delay test, pseudo-random test

I. Introduction

Modern electronics in critical and high-assurance applications (e.g. self-driving cars, aerospace, and medical devices) have strict reliability requirements. Since defective devices create economic loss or catastrophic loss-of-life, manufacturing tests must be credible in detecting and preventing faulty behavior. Tests are also required *in the field* after manufacturing to detect post-delivery defects, (i.e., *soft errors*).

Logic built-in self-test (*LBIST*) [1] is commonly used for both manufacturing tests and post-manufacturing reliability checks [2]. LBIST uses on-chip stimulus generators, i.e. *pseudo-random pattern generators* (*PRPGs*) [3], [4] to stimulate circuits inputs and set circuit states while circuit outputs and states are observed. When complementing conventional test methods (i.e., *ATPG*), LBIST can significantly increase fault coverage while decreasing test application time. LBIST is useful in field test. With embedded LBIST, devices become testable with minimal functional interruption by saving the circuit state, applying test enable/disable signals, and then reloading the circuit state to resume the normal function.

A major challenge for LBIST is detecting *random pattern resistant* (*RPR*) faults [5]. RPR faults manifest in logic with many inputs when few input combinations can excite certain logic paths, and therefore pseudo-random tests often fail to excite and observe RPR faults. The prototypical example of an RPR fault is the output of a large logic gate. For example, probability of the output of a 32-input AND gate being logic-1 and exciting a stuck-at-0 fault is 0.5^{32} (presuming all AND gate inputs are equally likely to be logic-0 or logic-1), which implies more than one billion pseudo-random patterns may be needed to excite the fault. Under the presence of RPR faults, applying LBIST becomes a time and power consuming process.

To detect RPR faults, many technologies have been proposed to improve the type of patterns applied, either through weighted random pattern testing [6]–[13] or PRPG reseeding [14]–[22]. Although useful, these methods require extra hardware to generate weights, require significant computation to calculate seeds, or require significant memory and hardware to store and apply seeds. Although still used in modern designs, some applications that have restricted hardware resources and nuanced environments cannot apply such methods, whilst others still need additional fault coverage with such methods.

An alternative method to improve LBIST performance is modifying circuits with *test points* (*TPs*). TPs change circuit values or observe values in a circuit, thus making the detection of RPR faults easier. *Test point insertion* (*TPI*) techniques find high-quality TPs locations which improve fault coverage or reduce the number of test patterns. Since the concept of TPI was proposed by Hayes and Friedman [23] in 1974, numerous algorithms have appeared to improve TPI performance. These methods can be placed into one of three categories based on how testability is measured: fault simulation, approximate testability measures, or multiple measurements. TPs can also be used in analog circuits, e.g., fault diagnosis and analog testing [24], [25], or to improve ATPG results [26], but present survey focuses on TPs for digital circuit LBIST.

II. TP Architectures

A. TP implementations

TPs are circuit modifications which change or observe circuit functions during test but do not change the circuit function when disabled [23], [27]. Conventional TPs are categorized into two types [28]: *control TPs* and *observe TPs* (as shown in Figure 1(a), (b), and (d)). Control TPs are typically implemented using OR gates for control-1 TPs or AND gates for control-0 TPs (and NAND/NOR gates can be used at the output of inverters). During test, a *test enable* pin forces lines to their controlled values [29]. While not under test, this test enable pin is disabled and the circuit function does not change. The goal of control TPs is to increase the probability of exciting faults in a circuit and to make faults easier to observe by creating propagation paths to circuit outputs. Observe TPs change circuit observability by inserting fan-outs to circuit outputs, which makes faulty values on lines easily observed [30].

The source of test enable and the output for observe points can either be a pin or a scan latch. Although test enable is most often modeled as a pin, implementing it as a circuit pin is impractical given the high cost of circuit pins. Instead, outputs and inputs of scan latches typically provide additional TP "pins". A large circuit with many TPs and (latch-implemented) TP pins may require significant area overhead. There are numerous articles on reducing TP pin/latch area overhead while using TP pins selectively to increase fault coverage. The following sub-section surveys these.

978-1-7281-5359-9/20/$31.00 ©2020 IEEE

IEEE computer society

Authorized licensed use limited to: Hiroshi Takahashi. Downloaded on June 14,2020 at 11:36:20 UTC from IEEE Xplore. Restrictions apply.

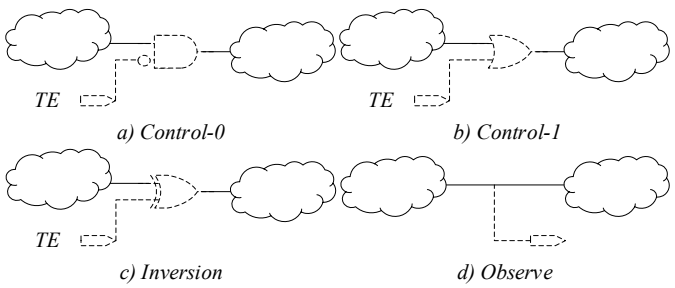

Figure 1: Illustrated here are the logic-level implementations of control, inversion, and observe TPs.

Although effective for increasing stuck-at fault coverage, both control and observe TPs have their detriments; hence, TPI must carefully select TP locations and types. Since a control TP forces the line to '0' or '1' when active, the controlled line can only be a single value when the TP is active. This prevents one stuck-at fault on the line from being excited. Additionally, active control TPs block the transmission of excited faults through the controlled line. Although observation TPs do not block faults like control TPs, observation TPs cannot detect RPR faults that are difficult to excite.

In contrast to control TPs, *inversion TPs* use inversions to change line values during test [31]–[35]. Inversion TPs are made with XOR gates and a test enable pin (shown in Figure 1(c)): when the test enable pin is active, the XOR gate becomes an inverter; otherwise the XOR gate acts as a buffer. In contrast to conventional TPs which force lines to values, inversion TPs invert signal probabilities, i.e., if a line has an 80% probability of being logic-1, the line will have a 20% probability of being logic-1 with an active inversion TP. Because active inversion TPs do not force a single value, both stuck-at-0 and stuck-at-1 faults can be excited on active TP locations. Additionally, faults can propagate through inversion TPs to circuit outputs (unlike control TPs which block faults from propagating through) [35]. However, inversion TPs add more propagation delay, power, and overhead compared to control TPs [35], [36]. Additionally, RPR faults may require values to be forced to optimally increase fault coverage [35], which inversion TPs cannot perform.

B. TP selection architectures

Although TPs can significantly improve fault coverage, they can create significant area overhead, which in turn increases production costs and reduces yields due to larger die areas and fewer dies per wafer [37]. One study found chip area increased by 2.68% when using logic BIST, and TPs constituted 43% of this area increase [38]. It is therefore important to reduce the area overhead of TPs while keeping LBIST fault coverage high.

To reduce TP area overhead, some methods proposed sharing flip-flops or other existing circuit signals to reduce TP-controlling hardware [28], [31], [38]–[42]. [28] and [39] proposed sharing a single flip-flop for multiple test enable signals, which reduced the number of flip-flops that were required to implement control points. [38], [41] found more than half of TPs inserted were control points, so replacing dedicated test enable flip-flops with existing functional flip-flops reduced area overhead: suitable functional flip-flops can be found in the fan-in region with the shortest distance from the control TPs. Additionally, the test enable signals were only active in the test mode since the test enable signal is generated based on latch value combinations that can never appear in the functional mode. [40] proposed a *self-drive TP*, which used test enable signals created from gate outputs already existing in the circuit, which eliminated the test enable signal generation. Similar to [40], [31] used pre-existing signals for test enable without the need for extra registers. [42] utilized *controllability don't-cares* to generate TPs activation signals instead of a global test enable signal, which generated test enable signals locally and allowed TPs to be randomly activated: these controllability don't-cares are constant values in the functional mode, such as circuit states accessible only through scan, and thus can only change values in the test mode.

Other studies proposed reducing the number of TPs needed through various means. [43], [44] partitioned circuit tests into multiple phases, and sub-sets of control TPs were activated during certain phases. This provided greater control over the interaction between control points and helped reduce the total number of TPs needed to obtain adequate fault coverage.

III. Test Point Insertion Algorithms

TPI algorithms iteratively select TPs from a list of candidates. In each iteration, they select a TP that increases the fault coverage the most without violating other constraints, such as, fault coverage, power, delay, etc. Optimal TP placement in circuits with reconvergent fanouts is a known *NP-hard* problem [45], [46] and, therefore, most TPI approaches use heuristics to select TP locations.

Many TPI algorithms proposed in the literature perform the following steps to insert a single TP. First, fault simulation or approximate testability measures identify RPR faults. Second, candidate TPs are evaluated for their impact on fault coverage. Third, the TP with the highest positive impact on fault coverage is inserted into the circuit. This process repeats until reaching the number of desired TPs or achieving some pre-designated threshold for estimated fault coverage.

A. TPI using simulation

Using fault simulation to find undetected faults and then inserting TPs to detect these faults is a straightforward method of TPI. [47] inserted control TPs on gate outputs where faults were not excited while inserting observe TPs at the input of gates which blocked propagation. [48] used *backward path tracing* [49]–[51] on undetected fault sites and used control TPs to sensitize a path to the fault.

Several methods [43], [44] used *probabilistic fault simulation* to guide TP placement combined with greedy heuristics. Probabilistic fault simulation performs regular logic simulation to find signal probabilities and faults which are propagated in the circuit, and then uses these probabilities to predict the probability any fault will be detected at a given location [44]. [44] used this method combined with a *divide-and-conquer* technique: probabilistic fault simulation was performed in phases, and at the end of each phase, TPs were inserted to target faults with the lowest detection probability. [43] improved memory usage and TPI CPU time whilst marginally sacrificing TPI accuracy: instead of using logic simulation to determine all faults which can be detected on each circuit line (and the probability of each fault being detected), a representative of all faults at each fan-out location is chosen in order to reduce the number of faults to consider during TPI.

Authorized licensed use limited to: Hiroshi Takahashi. Downloaded on June 14,2020 at 11:36:20 UTC from IEEE Xplore. Restrictions apply.

B. TPI using approximate testability measures

Fault simulation accurately quantifies fault coverage, but its computation complexity (in terms of CPU time and memory) is infeasible for modern circuits: to overcome this, numerous studies replace fault simulation with approximate *testability measures*, such as *SCOAP* [52] and *COP* [53]. SCOAP [52] is a linear complexity algorithm (relative to the number of logic gates in a circuit to analyze) which estimates the number of circuit inputs needed to force a logic-0/1 on a line, defined as *controllability*. Using these values, SCOAP can estimate the *observability* of a line, which is the number of inputs that must be set to propagate a faulty value on a line to an observable output. SCOAP also includes the depth of a line in a circuit in its controllability and observability estimations. Alternatively, COP [53] predicts the probability a line will be logic-0/1 and the probability of a line's value will be observed at a circuit output presuming random stimuli is applied to circuit inputs. COP values can directly predict the probability of fault detection: $controllability * observability$ for stuck-at-0 faults and $(1 - controllability) * observability$ for stuck-at-1 faults. Controllability and observability measures can therefore be used to identify hard-to-control and hard-to-observe locations in a circuit, and they can be used to predict the current fault coverage (with or without a TP) of a circuit without performing fault simulation: TPs can then be inserted based on this information.

Compared against exact fault simulation, testability measurements take substantially less time to calculate but loose accuracy for circuits with many reconvergent fanouts. However, experiments have suggested approximate testability measurements can be accurate enough for use in TPI for large designs [54]. Therefore, many TPI methods from literature [54]–[59] use a *cost function* to estimate a TP's quality, with a typical example [55] provided below: F is a set of faults, and P_{d_j} is the probability the fault j is detected (calculated using COP).

$$U = \frac{1}{|F|} \sum_{\forall j \in F} \frac{1}{P_{d_j}}$$

Cost functions such as U are used as indicators of circuit testability, and many TPI algorithms attempt to maximize such cost functions during TPI. In this example, the value of U changes when a TP is inserted, and the difference in U before and after a TP is inserted is called *actual cost reduction* (*ACR*) [55]. Gradient calculations [55] can select TPs with the largest ACR, but the computational complexity of finding the ACR for every TP is too high and unpractical for modern circuits [54]. Therefore, the concept of a *cost reduction factor* (*CRF*) was introduced to approximate ACRs [59]. The algorithms which use CRFs and ACRs typically perform as follows: first, controllability and observability are calculated using an approximate testability measure, e.g. COP or SCOAP; second, the CRF for each TP in a candidate set is calculated, discarding TPs with CRF below a given threshold; third, the ACR for remaining candidate TPs is calculated; lastly, the TP with the largest ACR is inserted.

For evaluating a TP by cost function, various studies use nuances to either select superior TPs or reduce TPI CPU time. [56] selected TPs whose impacts on timing slack and fault coverage were smaller and larger, respectively, than given thresholds. [54], [57] proposed *hybrid cost reduction*: after a TP was inserted, faults were divided into two subsets, and only those with a large change in $1/P_{d_f}$ had their $1/P_{d_f}$ value recalculated using fault simulation (with other faults being calculated by testability analysis); the rationale for this is that large changes in a CRF may be inaccurate. [58] proposed three strategies for accelerating CRF-based algorithms: remove TPs with redundant *TPI-effective regions* (i.e., regions where the same controllability (for control TPs) or observability (for observe TPs) are changed), choose the TP with the highest CRF (i.e., do not calculate ACR), and reduce candidate TPs by selecting the first TP found to reduce the cost function (instead of calculating the ACR or CRF for all candidate TPs).

Beyond COP and SCOAP, other methods use additional/alternative cost functions or introduce additional constraints. [28], [60] identified RPR faults using COP and created *fault sectors*: RPR faults were sorted by ascending logic levels, then control TPs targeted faults in ascending order and observation TPs targeted faults in descending order. This prevented the targeting of same fault by multiple TPs, which reduced the number of TPs required. [61] used *test counts* (*TCs*) to complement COP-based TPI: the TC of a line is the fewest number of tests that must pass through the line such that all faults in its fan-in cone will be tested, and TPs were selected in order of the most tests which must pass through the TP location. [26] proposed several cost functions using one or multiple test analysis measurements (COP, SCOAP, or TC): TPI was split into multiple stages, where each stage selected a cost function to target the currently the *hardest test problem*, namely, finding tests for RPR faults, reducing test vectors, or a combination of the two. [62], [63] used pre-TP COP controllability and observability as an input feature to an artificial neural network which predicted the quality of a TP. [64], [65] used an *efficiency equation* for TPs, which evaluated the size of a TP fan-out/fan-in cone-of-influence and the number of undetected faults in this cone. This information then helps select the TP with the highest efficiency. An estimation metric approximates the final area overhead and test coverage without TP insertion and synthesis. [66] proposed a new conditional testability measure to overcome COP's inability to account for reconvergent fan-outs, thus increasing the accuracy of calculated cost functions.

Some methods incorporate non-fault coverage information, such as, timing violations, into cost functions [56] or efficiency equations [65], as discussed in the following section.

C. TPI using multiple measures

Some approaches utilize both fault simulation and testability measures to increase TP quality [67]–[69]. [67] reduced test vector counts and test generation time by considering layout and timing information for observe TPs. The cost function (see Section III.B) of an observe TP was the product of the total number of independent faults (i.e., faults which cannot be simultaneously detected by any single pattern) in the fan-in cone of the observe TP (which was found through fault simulation) and the minimum number of controlled primary inputs needed to propagate the independent faults to the TP location (estimated using SCOAP). [68], [69] performed COP and fault simulation to calculate fault testability, propagated faults, and faults blocked by control TPs: the cost function of control TPs is based on the controllability of blocked faults and that of observation TPs is composed of the observability of unobserved faults.

Authorized licensed use limited to: Hiroshi Takahashi. Downloaded on June 14,2020 at 11:36:20 UTC from IEEE Xplore. Restrictions apply.

IV. Modern Targets for TPI

A. Path manipulation to increase path delay fault coverage

A path delay fault (PDF) [70] occurs when any path's delay exceeds a circuit's designed clock speed, and the PDF models defects which cause cumulative propagation delays along a circuit path that exceed the circuit's specifications. Unlike stuck-at faults, PDFs involve operational features as well. Thus, a PDF exists only within a certain range of operational clock periods. A PDF test uses a set-up vector to create the precondition for a transition and a second trigger vector to initiate the transition. Specialized test hardware, using a clock period greater than the operational clock period loads a set-up vector and then applies a trigger vector, but the operational clock period must follow the trigger vector in order to capture a transition along the path under test. If the target output has not changed from its value after the set-up vector, then the circuit is faulty.

There are three problems associated with PDFs, which are problems TPs have attempted to address. First, the number of paths (and number of PDFs) in practically-sized circuits is too large for test tools to handle [71]. Second, the number of tests needed to detect all PDFs is too large [72]. Third, many PDFs in practical circuits are not testable [73]. To remedy this, TPs can divide full paths into sub-paths, thus making paths easier to test and reducing the number of paths [74]. Additionally, it is easier to generate tests for shorter sub-paths [74].

TPI methods have incorporated these observations into cost functions, which represent the number of paths in a circuit, i.e., the TP that reduces the total number of paths in the circuit is iteratively chosen [74]–[76]. [75] selected TPs using the cost function above. [74] added an additional constraint to the above cost function, i.e., clock speed of the circuit under test: if a TP reduces the longest path in the circuit, the clock speed during test can be increased, thus decreasing test application time. [76] targeted *non-robust-dependent* faults of functionally sensitizable paths [77], thereby reducing the fault set and hence the number of TPs.

B. Power reduction during test

The power consumption of digital systems is considerably higher in a test mode compared to functional modes. This is because during normal circuit operation, a relatively small number of flip-flops change value each clock cycle, whilst in a test mode, a much larger number of flip-flops will change values, which results in excessive switching activity and current spikes [78]. Especially during self-test, power dissipation increases since random patterns can cause many nodes to switch [79]. If the peak power during test is too large, V_{dd} drop or ground bounce can cause false-failures or device damage.

Some studies [78], [79] inserted TPs to reduce power consumption during test, but TP placement was restricted to flip-flop outputs. [79] used modified shift registers which suppress activity at the output during shift operations: by adding NOR or NAND gates to the outputs of latches controlled by a test enable pin, latch outputs were forced to known values and thus did not cause circuit switching. [78] proposed inserting TPs into a conventional full-scan circuit to keep peak power during scan below a given limit without decreasing fault coverage (with TPs being inactive during the capture cycles): a subset of scan flip-flop outputs were forced to 0 and 1 during scan. First, cycle-by-cycle simulation identified which scan cycle's power consumption was greater than the specified limit. Second, an event-driven, selective trace simulation procedure [80] estimated the power reduction for every latch when its output was forced to 0 or 1, then latches were iteratively forced to reduce power consumption.

C. Considering the timing impacts of TPs

Inserted TPs may cause circuit timing violations which break proper circuit operation [37], and resolving these timing violations may require several tedious design iterations. Many attempts have been proposed [56], [81]–[83] to insert TPs to increase fault coverage without creating new timing violations. In [56], [83], timing analysis was performed before TPI to identify paths with small timing slacks, then TPI was performed after removing candidate TPs which reside on such paths. [81] performed TPI without any constraints, then timing analysis was performed to remove TPs which caused timing violations. [82] performed TPI at RTL-level (instead of the typical logical netlist level), which means TPs were inserted before logic synthesis to avoid later design iterations.

For timing or delay test, LBIST often involves at-speed application of pseudorandom patterns. Possible activation of non-functional false paths or multi-cycle paths may cause a good circuit to fail during test. We use timing analysis to ether suitably reduce the LBIST clock frequency [84] or identify circuit responses for masking during test [85]. The timing analysis must account for the TPI related logic modifications as well.

V. The Future of TPI

TPI has been explored extensively since 1974 [23]. Numerous proposed methods improved existing approaches or targeted nuanced issues. The testability of circuits and potential for using LBIST have been consistently improving.

However, as logic circuits become ever-more complex (despite the theoretical presence of "Moore's Wall") new problems will appear and the performance of TPI methods will need to improve further. First, few studies have touched the impact TPI has on power (or how to use TPI to reduce power without restricting TP placement to latch outputs), thus power will be an attracting topic in the future, especially since low-power applications are in high demand for consumer. Second, most TPI studies only target one or two problems, but many design issues (timing, area, power, and testability) directly conflict with each other, and addressing several of these issues simultaneously requires nuances to be addressed. A noteworthy challenge is the security issue created by TPs [86]. Third, with rapid technology developments, modern circuits have millions of components and the computational complexity of TPI is growing at a rate faster than the size of the circuits to be analyzed [62]. Additionally, more TPs need to be evaluated and inserted to increase fault coverage to acceptable levels in these large circuits, thus current TPI heuristics (and other DFT algorithms) will need significantly more time to meet fault coverage requirements. Given most TPI experiments in literature are done under old benchmark circuits, e.g., ISCAS'85 [87], ISCAS'89 [88], or ITC'99 [89], the ability of established heuristics to perform on larger, modern circuits need to be studied, as do new computing paradigms that can break the "heuristic wall".

Authorized licensed use limited to: Hiroshi Takahashi. Downloaded on June 14,2020 at 11:36:20 UTC from IEEE Xplore. Restrictions apply.

VI. REFERENCES

[1] B. L. Keller and T. J. Snethen, "Built-in Self-Test Support in the IBM Engineering Design System," *IBM J. Res. Dev.*, vol. 34, no. 2.3, pp. 406–415, Mar. 1990.

[2] P. H. Bardell and W. H. McAnney, "Self-Testing of Multichip Logic Modules," in *Proc. Intl. Test Conf. (ITC)*, Philadelphia, PA, USA, Nov. 1982, pp. 200–204.

[3] E. B. Eichelberger and E. Lindbloom, "Random-Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self-Test," *IBM J. Res. Dev.*, vol. 27, no. 3, pp. 265–272, May 1983.

[4] I. Pomeranz and S. M. Reddy, "3-Weight Pseudo-Random Test Generation Based on a Deterministic Test Set for Combinational and Sequential Circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 12, no. 7, pp. 1050–1058, Jul. 1993.

[5] E. J. Mccluskey, "Built-In Self-Test Techniques," *IEEE Des. Test Comput.*, vol. 2, no. 2, pp. 21–28, Apr. 1895.

[6] D. Xiang, X. Wen, and L.-T. Wang, "Low-Power Scan-Based Built-In Self-Test Based on Weighted Pseudorandom Test Pattern Generation and Reseeding," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 25, no. 3, pp. 942–953, Mar. 2017.

[7] H.-J. Wunderlich, "Multiple Distributions for Biased Random Test Patterns," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 9, no. 6, pp. 584–593, Jun. 1990.

[8] F. Brglez, C. Gloster, and G. Kedem, "Hardware-Based Weighted Random Pattern Generation," in *Proc. Intl. Test Conf. (ITC)*, Washington, DC, USA, Aug. 1989, pp. 264–274.

[9] M. Bershteyn, "Calculation of Multiple Sets of Weights for Weighted Random Testing," in *Proc. Intl. Test Conf. (ITC)*, Baltimore, MD, USA, Oct. 1993, pp. 1031–1040.

[10] A. Jas, C. V. Krishna, and N. A. Touba, "Hybrid BIST Based on Weighted Pseudo-Random Testing: A New Test Resource Partitioning Scheme," in *Proc. 19th IEEE VLSI Test Symp. (VTS)*, Marina Del Rey, CA, USA, Apr. 2001, pp. 2–8.

[11] S. Pateras and J. Rajski, "Cube-Contained Random Patterns and their Application to the Complete Testing of Synthesized Multi-level Circuits," in *Proc. Intl. Test Conf. (ITC)*, Nashville, TN, USA, Oct. 1991, pp. 473–482.

[12] R. Kapur, S. Patil, T. J. Snethen, and T. W. Williams, "Design of an Efficient Weighted Random Pattern Generation System," in *Proc. Intl. Test Conf. (ITC)*, Washington, DC, USA, Oct. 1994, pp. 491–500.

[13] L. Lai, J. H. Patel, T. Rinderknecht, and W.-T. Cheng, "Hardware Efficient LBIST With Complementary Weights," in *Proc. Intl. Conf. Computer Design (ICCD)*, San Jose, CA, USA, Oct. 2005.

[14] I. Pomeranz, "Computation of Seeds for LFSR-Based n-Detection Test Generation," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 22, no. 2, pp. 29:1–29:13, Jan. 2017.

[15] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Embedded Deterministic Test," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 23, no. 5, pp. 776–792, May 2004.

[16] G. K. Contreras, Y. Zhao, N. Ahmed, L. Winemberg, and M. Tehranipoor, "LBIST Pattern Reduction by Learning ATPG Test Cube Properties," in *Proc. 16th Intl. Symp. Quality Electronic Design (ISQED)*, Santa Clara, CA, USA, Mar. 2015, pp. 147–153.

[17] G. Contreras, N. Ahmed, L. Winemberg, and M. Tehranipoor, "Predictive LBIST Model and Partial ATPG for Seed Extraction," in *Proc. IEEE Intl. Symp. Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, Amherst, MA, USA, Oct. 2015, pp. 139–146.

[18] S. Hellebrand, S. Tarnick, J. Rajski, and B. Courtois, "Generation of Vector Patterns Through Reseeding of Multiple-Polynominal Linear Feedback Shift Registers," in *Proc. Intl. Test Conf. (ITC)*, Baltimore, MD, USA, Sep. 1992, pp. 120–129.

[19] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois, "Built-in Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers," *IEEE Trans. Comput.*, vol. 44, no. 2, pp. 223–233, Feb. 1995.

[20] B. Koenemann, C. Barnhart, B. Keller, T. Snethen, O. Farnsworth, and D. Wheater, "A SmartBIST Variant with Guaranteed Encoding," in *Proc. 10th Asian Test Symp. (ATS)*, Kyoto, Japan, 2001, pp. 325–330.

[21] C. V. Krishna, A. Jas, and N. A. Touba, "Test Vector Encoding Using Partial LFSR Reseeding," in *Proc. Intl. Test Conf. (ITC)*, Baltimore, MD, USA, Nov. 2001, pp. 885–893.

[22] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, N. Tamarapalli, and J. Qian, "Embedded Deterministic Test for Low-Cost Manufacturing," *IEEE Des. Test Comput.*, vol. 20, no. 5, pp. 58–66, Oct. 2003.

[23] J. P. Hayes and A. D. Friedman, "Test Point Placement to Simplify Fault Detection," *IEEE Trans. Comput.*, vol. C–23, no. 7, pp. 727–735, Jul. 1974.

[24] K. K. Pinjala and B. C. Kim, "An Approach for Selection of Test Points for Analog Fault diagnosis," in *Proc. 18th IEEE Intl. Symp. Defect and Fault Tolerance in VLSI Systems (DFT)*, Boston, MA, USA, Nov. 2003.

[25] C. Yang, S. Tian, and B. Long, "Application of Heuristic Graph Search to Test-Point Selection for Analog Fault Dictionary Techniques," *IEEE Trans. Instrum. Meas.*, vol. 58, no. 7, pp. 2145–2158, Jul. 2009.

[26] M. J. Geuzebroek, J. Th. van der Linden, and A. J. van de Goor, "Test Point Insertion That Facilitates ATPG in Reducing Test Time and Data Volume," in *Proc. Intl. Test Conf. (ITC)*, Baltimore, MD, USA, Oct. 2002, pp. 138–147.

[27] A. J. Briers and K. A. E. Totton, "Random Pattern Testability by Fast Fault Simulation," in *Proc. Intl. Test Conf. (ITC)*, Washington, DC, USA, Sep. 1986, pp. 274–281.

[28] M. Youssef, Y. Savaria, and B. Kaminska, "Methodology for Efficiently Inserting and Condensing Test Points," *IEE Proc E - Comput. Digit. Tech.*, vol. 140, no. 3, pp. 154–160, May 1993.

[29] J.-S. Yang, N. A. Touba, and B. Nadeau-Dostie, "Test Point Insertion with Control Points Driven by Existing Functional Flip-Flops," *IEEE Trans. Comput.*, vol. 61, no. 10, pp. 1473–1483, Oct. 2012.

[30] J. R. Fox, "Test-Point Condensation in the Diagnosis of Digital Circuits," *Proc Inst. Electr. Eng.*, vol. 124, no. 2, pp. 89–94, Feb. 1977.

[31] H. Ren, M. Kusko, V. Kravets, and R. Yaari, "Low Cost Test Point Insertion Without Using Extra Registers for High Performance Design," in *Proc. Intl. Test Conf. (ITC)*, Austin, TX, USA, Nov. 2009.

[32] D. V. Bakshi, "Techniques for Seed Computation and Testability Enhancement for Logic Built-In Self Test," M.S. thesis, Virginia Tech - Bradley Dept. Elec. Comp. Eng., Blacksburg, Virginia, 2012.

[33] Y. Fang and A. Albicki, "Efficient Testability Enhancement for Combinational Circuit," in *Proc. Intl. Conf. Computer Design (ICCD)*, Austin, TX, USA, Oct. 1995, pp. 168–172.

[34] E. M. Rudnick, V. Chickermane, and J. H. Patel, "An Observability Enhancement Approach for Improved Testability and at-Speed Test," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 13, no. 8, pp. 1051–1056, Aug. 1994.

[35] S. Roy, B. Stiene, S. K. Millican, and V. D. Agrawal, "Improved Random Pattern Delay Fault Coverage Using Inversion Test Points," in *Proc. IEEE 28th North Atlantic Test Workshop (NATW)*, Burlington, VT, May 2019.

[36] K. Juretus and I. Savidis, "Reducing Logic Encryption Overhead Through Gate Level Key Insertion," in *Proc. IEEE Intl. Symp. Circuits and Systems (ISCAS)*, Montreal, QC, Canada, May 2016, pp. 1714–1717.

[37] H. Vranken, F. S. Sapei, and H.-J. Wunderlich, "Impact of Test Point Insertion on Silicon Area and Timing During Layout," in *Proc. Design, Automation, and Test in Europe Conf. (DATE)*, Paris, France, Feb. 2004.

[38] J.-S. Yang, B. Nadeau-Dostie, and N. A. Touba, "Test Point Insertion Using Functional Flip-Flops to Drive Control Points," in *Proc. Intl. Test Conf. (ITC)*, Austin, TX, USA, Nov. 2009.

[39] M. Nakao, S. Kobayashi, K. Hatayama, K. Iijima, and S. Terada, "Low Overhead Test Point Insertion for Scan-based BIST," in *Proc. Intl. Test Conf. (ITC)*, Atlantic City, NJ, USA, Sep. 1999, pp. 348–357.

[40] F. Muradali and J. Rajski, "A Self-Driven Test Structiure for Pseudorandom Testing of Non-Scan Sequential Circuits," in *Proc. 14th IEEE VLSI Test Symp. (VTS)*, Princeton, NJ, USA, Apr. 1996, pp. 17–25.

[41] J. Yang, B. Nadeau-Dostie, and N. A. Touba, "Reducing Test Point Area for BIST through Greater Use of Functional Flip-Flops to Drive Control Points," in *Proc. 24th IEEE Intl. Symp. Defect and Fault Tolerance in VLSI Systems (DFT)*, Chicago, IL, USA, Oct. 2009, pp. 20–28.

[42] K.-H. Chang, C.-W. Chang, J.-H. R. Jiang, and C.-N. J. Liu, "Reducing Test Point Overhead with Don't-Cares," in *Proc. Intl. Midwest Symp. on Circuits and Systems (MWSCAS)*, Boise, ID, USA, Aug. 2012, pp. 534–537.

[43] N. Z. Basturkmen, S. M. Reddy, and J. Rajski, "Improved Algorithms for Constructive Multi-Phase Test Point Insertion for Scan Based BIST," in *Proc. Asia and South Pacific Design Automation Conf. (ASP-DAC)*, Bangalore, India, Jan. 2002.

Authorized licensed use limited to: Hiroshi Takahashi. Downloaded on June 14,2020 at 11:36:20 UTC from IEEE Xplore. Restrictions apply.

[44] N. Tamarapalli and J. Rajski, "Constructive Multi-phase Test Point Insertion for Scan-based BIST," in *Proc. Intl. Test Conf. (ITC)*, Washington, DC, USA, Oct. 1996, pp. 649–658.

[45] B. Krishnamurthy, "A Dynamic Programming Approach to the Test Point Insertion Problem," in *Proc. 24th ACM/IEEE Design Automation Conf. (DAC)*, Miami Beach, Florida, USA, Jun. 1987, pp. 695–705.

[46] J. Sziray, "Test Generation and Computational Complexity," in *Proc. IEEE 17th Pacific Rim Intl. Symp. Dependable Computing*, Pasadena, CA, USA, Dec. 2011, pp. 286–287.

[47] V. S. Iyengar and D. Brand, "Synthesis of Pseudo-random Pattern Testable Designs," in *Proc. Intl. Test Conf. (ITC)*, Washington, DC, USA, Aug. 1989, pp. 501–508.

[48] N. A. Touba and E. J. McCluskey, "Test Point Insertion Based on Path Tracing," in *Proc. 14th IEEE VLSI Test Symp. (VTS)*, Princeton, NJ, USA, Apr. 1996, pp. 2–8.

[49] T. Ramakrishnan and L. Kinney, "Extension of the Critical Path Tracing Algorithm," in *Proc. 27th ACM/IEEE Design Automation Conf. (DAC)*, Orlando, FL, USA, Jun. 1990, pp. 720–723.

[50] P. Menon, Y. Levendel, and M. Abramovici, "SCRIPT: A Critical Path Tracing Algorithm for Synchronous Sequential Circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 10, no. 6, pp. 738–747, Jun. 1991.

[51] M. Abramovici, P. R. Menon, and D. T. Miller, "Critical Path Tracing - an Alternative to Fault Simulation," in *Proc. 20th ACM/IEEE Design Automation Conf. (DAC)*, Miami Beach, FL, Jun. 1983, pp. 214–220.

[52] L. H. Goldstein and E. L. Thigpen, "SCOAP: Sandia Controllability/Observability Analysis Program," in *Proc. 17th ACM/IEEE Design Automation Conf. (DAC)*, Minneapolis, MN, USA, Jun. 1980, pp. 190–196.

[53] F. Brglez, "On Testability Analysis of Combinational Networks," in *Proc. IEEE Intl. Symp. Circuits and Systems (ISCAS)*, Montreal, Quebec, Canada, May 1984, vol. 1, pp. 221–225.

[54] H.-C. Tsai, K.-T. Chen, C.-J. Lin, and S. Bhawmik, "Efficient Test-Point Selection for Scan-Based BIST," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 6, no. 4, pp. 667–676, Dec. 1998.

[55] R. Lisanke, F. Brglez, A. J. de Geus, and D. Gregory, "Testability-Driven Random Test-Pattern Generation," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 6, no. 6, pp. 1082–1087, Nov. 1987.

[56] K.-T. Cheng and C.-J. Lin, "Timing-driven Test Point Insertion for Full-scan and Partial-scan BIST," in *Proc. Intl. Test Conf. (ITC)*, Washington, DC, USA, Oct. 1995, pp. 506–514.

[57] H.-C. Tsai, K.-T. Cheng, C.-J. Lin, and S. Bhawmik, "A Hybrid Algorithm for Test Point Selection for Scan-based BIST," in *Proc. 34th ACM/IEEE Design Automation Conf. (DAC)*, Anaheim, CA, Jun. 1997, pp. 478–483.

[58] M. Nakao, K. Hatayama, and I. Higashi, "Accelerated Test Points Selection Method for Scan-Based BIST," in *Proc. 6th Asian Test Symp. (ATS)*, Akita, Japan, 1997, pp. 359–364.

[59] B. H. Seiss, "Test Point Insertion for Scan-based BIST," in *Proc. 2nd European Test Conf.*, Munich, Germany, Apr. 1991, pp. 253–262.

[60] Y. Savaria, M. Youssef, B. Kaminska, and M. Koudil, "Automatic Test Point Insertion for Pseudo-random Testing," in *Proc. IEEE Intl. Symp. Circuits and Systems (ISCAS)*, Singapore, Jun. 1991, pp. 1960–1963.

[61] M. J. Geuzebroek, J. Th. van der Linden, and A. J. van de Goor, "Test Point Insertion for Compact Test Sets," in *Proc. Intl. Test Conf. (ITC)*, Atlantic City, NJ, USA, Oct. 2000, pp. 292–301.

[62] Y. Sun and S. K. Millican, "Test Point Insertion Using Artificial Neural Networks," in *Proc. IEEE Computer Society Annu. Symp. VLSI (ISVLSI)*, Miami, FL, USA, Jul. 2019, pp. 253–258.

[63] S. K. Millican, Y. Sun, S. Roy, and V. D. Agrawal, "Applying Neural Networks to Delay Fault Testing: Test Point Insertion and Random Circuit Training," in *Proc. 28th Asian Test Symp. (ATS)*, Kolkata, India, Dec. 2019, pp. 13–18.

[64] M. He, G. K. Contreras, M. Tehranipoor, D. Tran, and L. Winemberg, "Test-Point Insertion Efficiency Analysis for LBIST Applications," in *Proc. 34th IEEE VLSI Test Symp. (VTS)*, Las Vegas, NV, USA, Apr. 2016.

[65] M. He, G. K. Contreras, D. Tran, L. Winemberg, and M. Tehranipoor, "Test-Point Insertion Efficiency Analysis for LBIST in High-Assurance Applications," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 25, no. 9, pp. 2602–2615, Sep. 2017.

[66] M. Chen and D. Xiang, "Pseudorandom Scan BIST Using Improved Test Point Insertion Techniques," in *Proc. 7th Intl. Conf. Solid-State and Integrated Circuits Technology (ICSICT)*, Beijing, China, Oct. 2004, pp. 2043–2046.

[67] R. Sethuram, S. Wang, S. T. Chakradhar, and M. L. Bushnell, "Zero Cost Test Point Insertion Technique to Reduce Test Set Size and Test Generation Time for Structured ASICs," in *Proc. 15th Asian Test Symp. (ATS)*, Fukuoka, Japan, Nov. 2006.

[68] C. Acero *et al.*, "Embedded Deterministic Test Points," *IEEE Trans. Very Large Scale Integr. VLSI Syst.*, vol. 25, no. 10, pp. 2949–2961, Jul. 2017.

[69] E. Moghaddam, N. Mukherjee, J. Rajski, J. Solecki, J. Tyszer, and J. Zawada, "Logic BIST with Capture-per-Clock Hybrid Test Points," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 38, no. 6, pp. 1028–1041, Jun. 2019.

[70] G. L. Smith, "Model for Delay Faults Based Upon Paths," in *Proc. Intl. Test Conf. (ITC)*, Philadelphia, PA, USA, 1985, pp. 342–351.

[71] I. Pomeranz and S. M. Reddy, "An Efficient Non-Enumerative Method to Estimate Path Delay Fault Coverage," in *Proc. IEEE/ACM Intl. Conf. Computer-Aided Design (ICCAD)*, Santa Clara, CA, USA, Nov. 1992, pp. 560–567.

[72] I. Pomeranz and S. M. Reddy, "On the Number of Tests to Detect All Path Delay Faults in Combinational Logic Circuits," *IEEE Trans. Comput.*, vol. 45, no. 1, pp. 50–62, Jan. 1996.

[73] C. J. Lin and S. M. Reddy, "On Delay Fault Testing in Logic Circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 6, no. 5, pp. 694–703, Sep. 1987.

[74] S. Tragoudas and N. Denny, "Testing for Path Delay Faults Using Test Points," in *Proc. IEEE Intl. Symp. Defect and Fault Tolerance in VLSI Systems (DFT)*, Albuquerque, NM, USA, Nov. 1999.

[75] I. Pomeranz and S. M. Reddy, "Design-for-testability for Path Delay Faults in Large Combinational Circuits Using Test Points," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 17, no. 4, pp. 333–343, Apr. 1998.

[76] P. Uppduri, U. Sparmann, and I. Pomeranz, "On Minimizing the Number of Test Points Needed to Achieve Complete Robust Path Delay Fault Testability," in *Proc. 14th IEEE VLSI Test Symp. (VTS)*, Princeton, NJ, USA, Apr. 1996, pp. 288–295.

[77] U. Sparmann, D. Luxenburger, K.-T. Cheng, and S. M. Reddy, "Fast Identification of Robust Dependent Path Delay Faults," in *Proc. 32nd ACM/IEEE Design Automation Conf. (DAC)*, San Francisco, CA, 1995.

[78] R. Sankaralingam and N. A. Touba, "Inserting Test Points to Control Peak Power During Scan Testing," in *Proc. 17th IEEE Intl. Symp. Defect and Fault Tolerance in VLSI Systems (DFT)*, Vancouver, Canada, Nov. 2002.

[79] S. Gerstendörfer and H.-J. Wunderlich, "Minimized Power Consumption For Scan-Based BIST," in *Proc. Intl. Test Conf. (ITC)*, Atlantic City, NJ, USA, Sep. 1999, pp. 77–84.

[80] E. G. Ulrich, V. D. Agrawal, and J. H. Arabian, *Concurrent and Comparative Discrete Event Simulation*. Boston, MA: Springer, 1994.

[81] X. Gu, S. S. Chung, F. Tsang, J. A. Tofte, and H. Rahmanian, "An Effort-Minimized Logic BIST Implementation Method," in *Proc. Intl. Test Conf. (ITC)*, Baltimore, MD, USA, Nov. 2001, pp. 1002–1010.

[82] S. Roy, G. Guner, and K.-T. Cheng, "Efficient Test Mode Selection & Insertion for RTL-BIST," in *Proc. Intl. Test Conf. (ITC)*, Atlantic City, NJ, USA, Oct. 2000, pp. 263–272.

[83] H. Vranken, F. Meister, and H. Wunderlich, "Combining Deterministic Logic BIST with Test Point Insertion," in *Proc. 7th IEEE European Test Workshop (ETW)*, Corfu, Greece, May 2002.

[84] F. P. Higgins and R. Srinivasan, "BSM2: Next Generation Boundary-Scan Master," in *Proc. 18th IEEE VLSI Test Symposium (VTS)*, Montreal, Quebec, Canada, Apr. 2000, pp. 67–72.

[85] V. Vorisek, B. Swanson, K.-H. Tsai, and D. Goswami, "Improved Handling of False and Multicycle Paths in Atpg," in *Proc. 24th IEEE VLSI Test Symposium*, Berkeley, CA, USA, Apr. 2006, pp. 6 pp. – 165.

[86] E. Moghaddam, N. Mukherjee, J. Rajski, J. Tyszer, and J. Zawada, "On Test Points Enhancing Hardware Security," in *Proc. 25th Asian Test Symp. (ATS)*, Hiroshima, Japan, Nov. 2016, pp. 61–66.

[87] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Targeted Translator in FORTRAN," in *Proc. IEEE Intl. Symp. Circuits and Systems (ISCAS)*, Kyoto, Japan, Jun. 1985.

[88] F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," in *Proc. IEEE Intl. Symp. Circuits and Systems (ISCAS)*, Portland, OR, USA, May 1989, pp. 1929–1934.

[89] S. Davidson, "ITC'99 Benchmark Circuits - Preliminary Results," in *Proc. Intl. Test Conf. (ITC)*, Atlantic City, NJ, USA, Sep. 1999, pp. 1125–1125.

Authorized licensed use limited to: Hiroshi Takahashi. Downloaded on June 14,2020 at 11:36:20 UTC from IEEE Xplore. Restrictions apply.