Test-Point Insertion Efficiency Analysis for LBIST in High-Assurance Applications

Miao He, *Student Member, IEEE*, Gustavo K. Contreras, *Student Member, IEEE*, Dat Tran, LeRoy Winemberg, and Mark Tehranipoor, *Senior Member, IEEE*

***Abstract*— Test points are inserted into integrated circuits to increase fault coverage especially in logic built-in self-test schemes. Commercial tools have been developed over the past decade to insert test points in circuits under test, but they are often inefficient and incur unacceptably large area overhead. Our analysis shows that many test points have little or no impact on test coverage. Furthermore, depending on where test points are inserted, they can create a significant area overhead unnecessarily. Therefore, we propose a novel timing-aware framework to evaluate test points' impact on a design, rank them based on their efficiency, and obtain an optimal configuration of the most efficient test points accurately and rapidly. Specifically, the proposed framework considers not only individual test coverage improvement but also area penalty, path timing, and region in which each test point is inserted. Within this framework, we have two metrics, namely, efficient test point insertion (ETPI) and test point removal estimation (TPRE). The ETPI metric is developed to remove the most inefficient test points inserted in the circuit by commercial tools, thereby minimizing area penalty with very limited test coverage loss. The TPRE metric is introduced to estimate area overhead and test coverage for designs with different percentages (number) of test points removed without the actual insertion of test points and without the need for lengthy circuit simulation, thereby quickly selecting the most effective test point removal scheme and saving significant amount of processing time especially for large circuits. Experimental results, collected by applying the metrics to NXP Semiconductors circuits and academic benchmark circuits, indicate that ETPI can reduce area overhead by up to 95% with test coverage loss as low as 0.57%. In addition, results by applying the TPRE metric indicate that the difference between estimation and actual simulation/synthesis results for area overhead is less than 0.20% for most cases, and the difference between them for test coverage is less than 1% for most cases.**

***Index Terms*— Area overhead, logic built-in self-test (LBIST), test coverage, test point insertion (TPI), testability.**

I. Introduction

CRITICAL and high-assurance applications, e.g., automotive applications, have strict time and test coverage requirements. Logic built-in self-test (LBIST) is commonly used for in-field test of these applications [1]. However, a major challenge is the presence of random pattern resistant faults, which limit test coverage, especially for large designs. Over the past couple of decades, various methods have been proposed to overcome this problem. One approach is to improve the type of patterns applied or adjust pattern stimulus. For example, high test coverage can be achieved by applying deterministic patterns [2]–[4], using weighted random patterns [5]–[12], and linear feedback shift register reseeding [13]–[18]. However, these methods require either hardware components to generate deterministic patterns, weights, seeds, or extra memory to store them. These requirements cannot be met in many critical and high-assurance applications, which have strict hardware specifications.

Another approach to improve LBIST coverage is to modify the circuit by inserting test points. Test points are inserted to help propagate random patterns to hard-to-test areas in a circuit. Test point insertion (TPI) techniques have been studied extensively, since it was originally proposed by Hayes and Friedman [19] in 1973. Methods for test point placement mainly include exact fault simulation [20]–[22], approximate testability measures [23]–[27], and path tracing [22].

Although test points are generally very effective at increasing test coverage, they incur a significant area penalty, which increases production cost, test cost, and reduces yield due to more transistors on a chip and less dies per wafer [28]. Over the past decade or so, various works have focused on reducing the number of test points. Reference [29] proposes to reuse the existing flip-flops (FFs) to drive control points instead of using dedicated FFs. In [30], a method is presented to avoid adding extra registers by using preexisting logic as test points. In [31] and [32], a TPI technique is developed to utilize unused FFs existing in application specific integrated circuit designs to reduce test volume and automatic test pattern generation time. A divide and conquer technique in [33] is presented to partition the entire test into different phases. Within each phase, a specific set of control points is enabled and inserted to achieve high fault coverage thus reducing the number of control points and avoiding conflicts between them. In [34] and [35], approaches are introduced to reduce the area overhead of test point selection for scan-based BIST, such as restricted cell replacement and test point FF sharing.

Commercial tools are commonly used to insert test points [36]. However, a large portion of test points generated by these tools do not significantly improve test coverage but can have a considerable impact on area overhead. Furthermore, some test points become much less effective after synthesis in

Manuscript received January 5, 2017; revised April 4, 2017; accepted May 9, 2017. Date of publication June 13, 2017; date of current version August 23, 2017. *(Corresponding author: Miao He.)*

M. He, G. K. Contreras, and M. Tehranipoor are with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611 USA (e-mail: tonyhe@ufl.edu; gustavokcm@ufl.edu; tehranipoor@ufl.edu).

D. Tran and L. Winemberg are with NXP Semiconductors, Austin, TX 78735 USA (e-mail: dat.tran@nxp.com; leroy.winemberg@nxp.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2017.2704104

the final netlist. To be specific, a design with significant timing constraints could have a large number of failing paths (i.e., paths with negative slack values) before timing optimization. The algorithms used for TPI might overlook the impact of the synthesis process on the area incurred by test points if they are inserted in failing paths or timing critical paths. The added area overhead is a result of synthesis tools optimizing paths to meet timing constraints, which in turn can reduce test coverage improvement provided by these test points. The results from implementation of such analysis on industry circuits indicate that a large number of test points could negatively affect synthesis, thereby having considerable effect on area overhead when they are inserted in such paths. In our experiments, the area overhead for the same number of test points inserted into an industry circuit before timing optimization and after timing optimization is 1.81% and 6.11%, respectively. Therefore, the number of test points is not the main indicator of area overhead. Rather, a better indicator is the impact test points have on the design which requires synthesis tools to compensate for it, in order to meet design constraints, i.e., timing and/or power.

In [37], we proposed an evaluation framework to aid TPI, i.e., to evaluate test point's impact on a design and to obtain an optimal configuration of efficient test points. However, the methodology in [37] suffered from a few drawbacks. First, the framework did not consider that the impact test points have on the design when they are being placed in failing paths. Second, instead of being applied to industrial designs with rigid timing constraints to show its effectiveness in practice, it was only applied to various academic benchmark circuits without significant timing constraints.

These issues have motivated us to design a new methodology to address the above-mentioned challenges in [37] and other existing techniques to make the framework more effective for TPI. To be precise, in this paper, we propose a timing-aware framework to evaluate each test point's impact on the circuit and to optimize test point area overhead and test coverage. Furthermore, in order to meet significant timing constraints in industrial designs used for critical and high-assurance applications, the timing-aware framework also considers path timing, critical paths, and the effect of test points on these paths. Within this framework, we present two metrics, namely, test point removal estimation (TPRE) and efficient TPI (ETPI). First, the TPRE metric is proposed to quickly obtain the optimal configuration of efficient test points to insert by estimating test point area overhead and test coverage improvement (i.e., to avoid the lengthy simulation and synthesis) for different percentages (number) of test points inserted. Then, the configuration of test points generated by the TPRE metric, i.e., the removal percentage of control points and observation points, is used to guide the ETPI metric to remove test points. Specifically, the ETPI metric is developed using test point's characteristics obtained from static timing analysis (STA) and synthesis tools. This metric is used to remove the TPRE specified percentage of test points by ranking test points with respect to their "efficiency values," thereby minimizing test point area overhead while achieving desired test coverage.

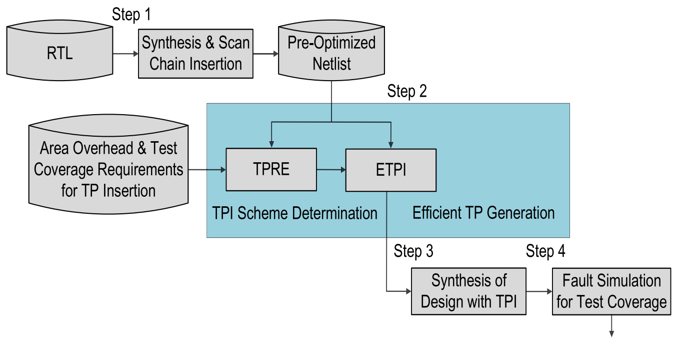

Step 1
RTL
Synthesis & Scan Chain Insertion
Pre-Optimized Netlist
Step 2
Area Overhead & Test Coverage Requirements for TP Insertion
TPRE
ETPI
TPI Scheme Determination
Efficient TP Generation
Step 3
Step 4
Synthesis of Design with TPI
Fault Simulation for Test Coverage

Fig. 1. Timing-aware evaluation framework for TPRE and efficient TPI (TP: test point).

The underlying philosophy of this methodology is to guide the designer to choose the optimal TPI scheme depending on the requirements of area overhead and test coverage without the need of performing multiple syntheses and fault simulations, which can be time-consuming for large designs.

The rest of this paper is organized as follows. In Section II, the evaluation framework of ETPI is presented. Section III presents the ETPI metric. Section IV describes the TPRE metric. Section V provides and fully analyzes the results to demonstrate the performance of the methodologies proposed in this paper. Finally, the concluding remarks are offered in Section VI.

II. Timing-Aware Evaluation Framework for Efficient Test Point Insertion

Fig. 1 outlines the timing-aware evaluation framework of TPI based on the proposed metrics.

In order to calculate test point's efficiency and estimate results of TPI, this framework considers not only individual test coverage improvement but also area overhead, path timing, and region, where each test point is inserted. In Step 1, the circuit is first synthesized with full scan chain insertion. In Step 2, the framework is divided into two parts: the TPRE metric and the ETPI metric. The path timing is analyzed based on the preoptimized netlist when applying both metrics. TPRE takes care of estimating the area overhead and test coverage, and decides the appropriate TPI scheme by considering the estimation results and the requirements for various applications. The TPI scheme created by the TPRE metric is used to guide the ETPI metric to remove test points with the specified removal percentage. ETPI is responsible for calculating the efficiency of test points as well as ranking them based on their efficiency. Then, ETPI removes the TPRE specified percentage of inefficient test points according to the evaluated efficiency of each test point, thus generating the set of most efficient test points to insert. Following this, the framework enters into Step 3 where the actual TPI takes place. By comparing the area overhead results shown in the synthesis reports between the original design and the design with test points inserted, the area overhead of test point logic can be obtained. In Step 4, fault simulation is implemented on the final netlist and test coverage is obtained. The metrics within this framework are explained in detail in Sections III and IV.

III. ETPI: Efficient Test Point Insertion Metric

TPI techniques generally select signal nodes in a combinational circuit to add control and observation points to minimize the number of test patterns required for detecting stuck-at faults in that circuit. Control points inserted to help with random patterns not only provoke faults but propagate faults to scan FFs or primary outputs. A control point is inserted at a node to improve the controllability of its fanout cone, where the inserted signal propagates toward outputs. It also affects the observability of the region where inputs propagate toward the inserted signal, since it can change the propagation path of the nodes in the region. On the other hand, observation points are added at nodes to which provoked faults can be propagated. An observation point is inserted at a node to improve the observability of the fanin cone, where inputs propagate toward the inserted signal.

From the tool's point of view, test point locations are the nodes in the design where the tool inserts logic to improve the testability of the design. These tools use iterative static analysis with random patterns to improve test coverage and reduce pattern count [38]. However, TPI tools perform this analysis and insert test points before the design is completely synthesized and optimized. Some test points, generated by TPI tools, might be inserted without considering the effect of these test points on critical and long paths, which will affect path delay and slack. After synthesis and optimization, these test points could have a significant impact on area overhead while still only detect few faults. We define efficient test points as those that target a large number of not-detected (ND) faults yet have a relatively small effect on area overhead. Contrarily, inefficient test points are those which target few faults but are inserted in areas that can significantly affect area overhead after synthesis and optimization. These are unacceptable in many high-assurance applications, especially those used in LBIST deployed in automotive applications. Therefore, it is important to reduce the number of test points so as to minimize area penalty while achieving the desired test coverage and test time.

In [37], we proposed the original ETPI metric to remove the most inefficient test points inserted in the circuits by the commercial tools. However, the originally proposed ETPI metric has not considered the impact on area overhead and test coverage incurred by test points inserted in failing paths. Therefore, to consider the impact those test points have on the design, the timing-aware ETPI metric is developed. The details of the originally proposed ETPI metric and the timing-aware ETPI metric are described in the following sections.

A. Proposed ETPI Metric

Control points allow signals within logic cones to be actively controlled during test mode to improve the controllability of the logic [38]. Thus, for a given control test point, the larger the fanout cone is, the more pins and cells it could control. Similarly, observation points passively capture the value of selected hard-to-observe signals to improve the observability of the logic [38]. Accordingly, for a given observation point, a larger fanin cone indicates a greater number of hard-to-observe signals it could observe.

However, only considering the size of fanout cone and fanin cone is not necessarily the most efficient methods for TPI. An efficient TPI algorithm aims to target as many ND faults as possible to aid LBIST patterns and improve testability. Thus, a node is selected to insert a control point, such that its fanout cone could contain as many not-controlled (NC) faults as possible. Likewise, a node is selected to insert an observation point, such that its fanin cone could contain as many not-observed (NO) faults as possible. Therefore, a test point's efficiency depends on the size of logic cones as well as the intersection between logic cones and ND faults. The size of fanout cones and fanin cones refers to the number of pins and cells in the fanout or the fanin of specified sources or sinks, respectively. The intersection of logic cones and ND faults refers to NC faults contained within the corresponding fanout cone and NO faults contained within the corresponding fanin cone.

In the calculation of test point's efficiency, the size of the logic cones and the number of ND faults in these regions are considered to be equally important. However, ND faults are a small subset of all the nodes in the logic cones; thus, the number of ND faults is far smaller than the number of nodes in the logic cones. To balance the magnitude of these two variables, the square root of the size of the logic cone is used when calculating a test point's efficiency. The square root used in the formula is empirical observation, which is based on the results of different benchmark circuits from Gaisler, OPENCORE, ITC99, and ISCAS85. We first look into the control point's efficiency (E_{ctl}), which is defined as

$$E_{\text{ctl}} = \sqrt{L_{\text{fanout}}} \times N_{\text{NC\_faults, fanout\_cone}} \tag{1}$$

where L_{fanout} is the size of fanout cone and calculated by the number of pins and cells in that fanout cone and $N_{\text{NC\_faults, fanout\_cone}}$ is the number of NC faults within that fanout cone.

Similar to the efficiency of a control point, the efficiency of an observation point (E_{obs}) can be defined as

$$E_{\text{obs}} = \sqrt{L_{\text{fanin}}} \times N_{\text{NO\_faults, fanin\_cone}} \tag{2}$$

where L_{fanin} is the size of fanin cone and calculated by the number of pins and cells in that fanin cone and $N_{\text{NO\_faults, fanin\_cone}}$ is the number of NO faults corresponding to that fanin cone.

The framework of calculating ETPI metric is shown in Fig. 2. The framework is divided into two parts, logic cone analysis and ND faults analysis. In the logic cone analysis, the entire set of test point candidates is generated first; then, test point characteristics, such as the size of logic cones, can be obtained through STA. In the ND faults analysis, fault simulation is performed to obtain a list of ND faults. The number of ND faults in each logic cone can then be calculated. Then, the efficiency of each test point is calculated based on the results of the two parts mentioned earlier. Then, all the test points are ranked with respect to their efficiency. Finally, the set of efficient test points is generated based

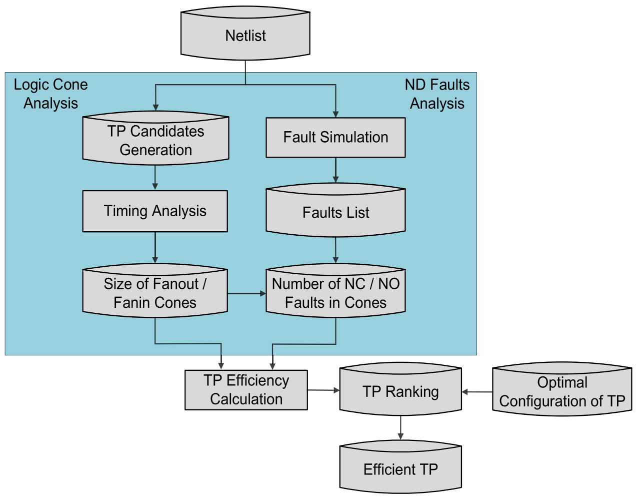


Fig. 2. ETPI metric for TPI.

on the optimized configuration of test points created by the TPRE metric and the evaluated efficiency of each test point.

B. Timing-Aware ETPI Metric

Path slack is measured as the difference between clock period and path delay. A positive slack is desired as it indicates the signal at the endpoint of the path arrives before the required time. Failing paths are the paths which fail to meet timing requirements before timing optimization. If path timing is not considered, a large number of test points could be inserted into failing paths, because test points inserted into these paths would drive large logic cones. Furthermore, high-assurance applications often have strict timing and power constraints. Therefore, synthesis tools work harder to meet design constraints affected by such test points. To be precise, extra gates, registers, and clocking buffers are added during TPI. In addition, gates are often resized during optimization. All of these could increase load on paths and have a large impact on physical synthesis and, hence, area overhead. Also, test coverage is impacted because of the additional gates, buffers, and resized gates. Thus, to consider the impact on area overhead and test coverage incurred by such test points inserted in paths with negative slack values, the timing-aware ETPI metric is introduced.

Before introducing the timing-aware ETPI metric, we analyze the case when test points are inserted in the design with a large number of failing paths. In the original ETPI metric, the size of the fanout/fanin cone was considered, because it is related to the number of ND faults covered by the test point. However, in the case of paths with strict timing, test points inserted in these locations can have a negative effect on area overhead and thus negatively impact the efficiency value. Therefore, the size of the fanout/fanin cone is removed from the efficiency value calculation in the original ETPI metric.

Based on a test point's impact on area overhead and test coverage when being inserted into a critical path, control points can be classified to four different groups: 1) control points with high impact on fault coverage and positive slack values; 2) control points with high impact on fault coverage and negative slack values; 3) control points with low impact on fault coverage and positive slack values; and 4) control points with low impact on fault coverage and negative slack values. The timing-aware ETPI metric aims to target the most faults with the least impact on timing and area. To be exact, it prioritizes control points in group 1 because of their high added testability and low area impact. It also minimizes control points in group 4, since they could have a negative effect on fault coverage due to the added area. Control points in groups 2 and 3 are used to create a tradeoff between fault coverage and area overhead. Therefore, in the timing-aware ETPI metric, a weighting factor based on slack values replaces the size of fanout cone in the original ETPI metric, because it considers the effect of test points on path delay. The efficiency of a control point (E_{ctl}) is defined as

$$E_{\text{ctl}} = N_{\text{NC\_faults, fanout\_cone}} \times W_{\text{slack}} \tag{3}$$

where $N_{\text{NC\_faults, fanout\_cone}}$ is the number of NC faults within the fanout cone and W_{slack} is the weighting factor based on slack values. As analyzed before, the adverse effects control points have on a design arise from the optimization required to meet timing constraints. Thus, the more negative the slack values of fanout paths are, the more gates need to be added or resized in order to meet design constraints, the smaller the weighting factor is introduced to calculate their efficiency.

Compared with control points, observation points are single FFs added to scan chains to capture responses from unobservable nets and inserted at the end of fanin paths. As a result, the timing impact observation points have on the design that is much less than control points. Therefore, the weighting factor is not considered and involved to calculate the efficiency of an observation point (E_{obs}), which is defined as

$$E_{\text{obs}} = N_{\text{NO\_faults, fanin\_cone}} \tag{4}$$

where $N_{\text{NO\_faults, fanin\_cone}}$ is the number of NO faults corresponding to the fanin cone.

The framework for the proposed timing-aware ETPI metric is shown in Fig. 3. It is divided into two parts, logic cone and slack analysis and ND faults analysis. In logic cone and slack analysis, the entire set of test point candidates is generated first. Then, the slack values of the paths in which control points are inserted are obtained through STA. The timing analysis can only be applied on the preoptimized netlist in order to see whether there exists any failing path because all the failing paths are fixed after timing optimization. The size of logic cones is obtained through synthesis tools. In ND faults analysis, fault simulation is performed to obtain a list of ND faults after timing optimization. The number of ND faults in each test point's logic cone is then calculated. Next, the efficiency of each test point is calculated based on the metrics presented earlier and using results of these two analyses. Specifically, if there exist failing paths, the weighting factor created by slack values is involved into the efficiency calculation [see (3) and (4)]; otherwise, the size of logic cones is considered as a factor to calculate efficiency

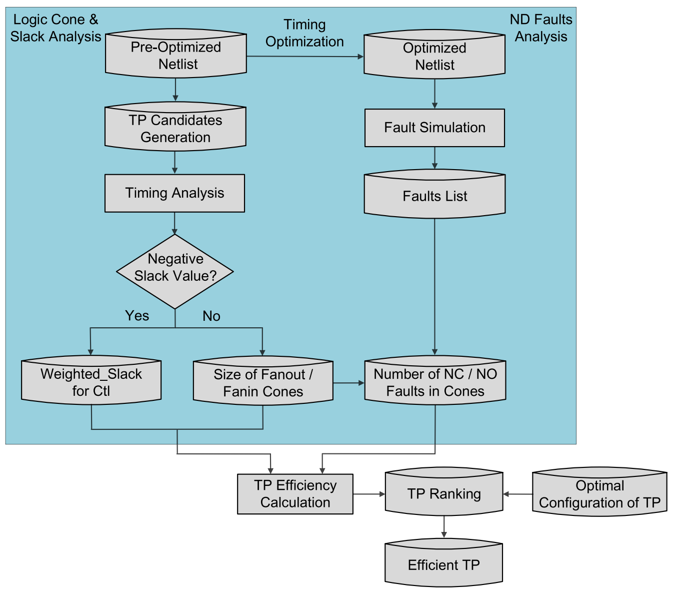


Fig. 3. Timing-aware ETPI metric for TPI.

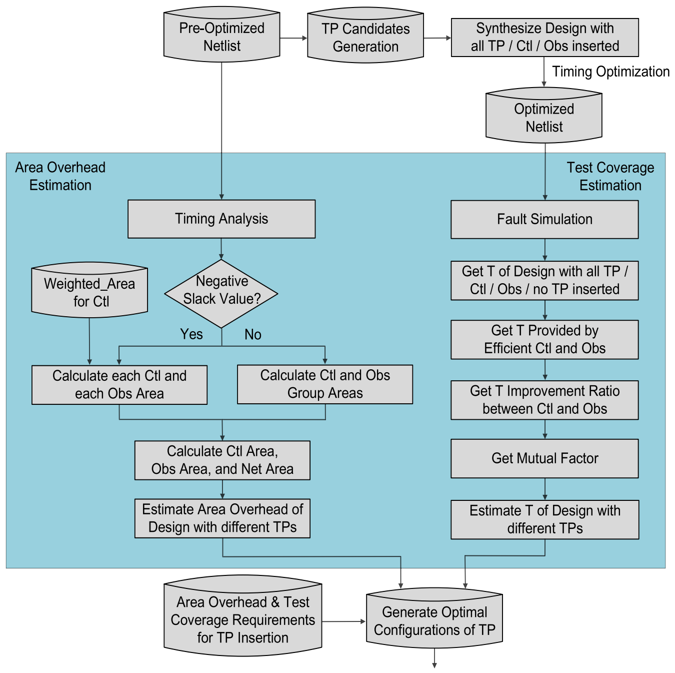

Pre-Optimized Netlist
TP Candidates Generation
Synthesize Design with all TP / Ctl / Obs inserted
Timing Optimization
Optimized Netlist
Area Overhead Estimation
Test Coverage Estimation
Timing Analysis
Fault Simulation
Weighted_Area for Ctl
Negative Slack Value?
Get T of Design with all TP / Ctl / Obs / no TP inserted
Yes
No
Get T Provided by Efficient Ctl and Obs
Calculate each Ctl and each Obs Area
Calculate Ctl and Obs Group Areas
Get T Improvement Ratio between Ctl and Obs
Calculate Ctl Area, Obs Area, and Net Area
Get Mutual Factor
Estimate Area Overhead of Design with different TPs
Estimate T of Design with different TPs
Area Overhead & Test Coverage Requirements for TP Insertion
Generate Optimal Configurations of TP

Fig. 4. Timing-aware TPRE metric for TPRE (T: test coverage).

[see (1) and (2)]. Then, all the test points are ranked with respect to their efficiency. Finally, the set of efficient test points is generated based on the optimized configuration of test points created by the TPRE metric and the evaluated efficiency of each test point.

IV. TPRE: Test Point Removal Estimation Metric

As the complexity of integrated circuits is rapidly growing, testing cost has become an important part of the overall chip manufacturing cost. Different applications have different needs; in some situations, they focus mainly on the silicon area instead of LBIST pattern count. In other cases, they put more emphasis on testability criteria than silicon area. Automotive applications are limited to test time of a few milliseconds during key-ON and key-OFF tests. LBIST is commonly used for the in-field test of these applications, because it can meet the strict test time requirement. Hence, meeting the test time budget while optimizing other parameters becomes an important issue for LBIST and in-field test. Generating an optimal configuration of control and observation points, which can improve testability, allows LBIST patterns to better test the design and achieve the target test coverage rapidly while meeting test time constraints.

ETPI will remove the number of test points specified by the DFT designer. However, the actual effect of test points on test coverage and area overhead cannot be measured until after the design has been optimized and fully synthesized. If the designer desires a higher area overhead reduction or lower test coverage loss, an entirely new synthesis of the design with a different set of parameters for TPI and ETPI calculations would have to be performed. Obtaining an optimal test coverage and area overhead tradeoff might require multiple lengthy syntheses and fault simulations. To mitigate this, TPRE provides an estimation metric, which allows the designer to approximate the final area overhead and test coverage without the actual insertion of test points and synthesis. Therefore, not only do we need the ETPI metric to remove the inefficient test points to minimize area penalty with limited test coverage loss, but also we need the TPRE metric to help produce flexible TPI schemes that can meet a range of different requirements that satisfy area and test coverage targets accurately and rapidly. Since large amount of simulation time is saved, significant acceleration in the selection of test point removal scheme can be achieved.

The framework of TPRE metric is shown in Fig. 4. First, the entire set of test point candidates is generated. We then synthesize the design under five scenarios to obtain the parameters for area overhead and test coverage estimation: with all the test points inserted, with only control points inserted, with only observation points inserted, and with 50% of control points and 50% of observation points inserted. These 50% of test points are obtained based on the ETPI metric. Following this, timing optimization is performed on the synthesized netlist. Then, the framework is divided into two parts: area overhead estimation and test coverage estimation. As their names indicate, the former is responsible for estimating the area overhead of design with different test points inserted, while the latter takes care of test coverage estimation. Specifically, if there exist failing paths in the design, the area overhead estimation will be based on the unit area of test points, since path timing has to be considered for each of them; otherwise, it will be based on the group area of test points. Finally, we get the optimized configuration of test points to be inserted based on the estimation results and the requirements for different applications. The details of the estimation process are illustrated as follows.

A. Proposed Metric for Area Overhead Estimation

To calculate area, we first introduce and analyze the structure of control and observation point groups. To reduce the area penalty of test point logic, the tools allow to share test point enable signal and scan registers with multiple test points. The number of test points of the same type that can share one scan cell is one to eight [38]. A higher number of test points grouped to share one scan cell incur a lower area overhead. For example, if the tools specify eight control or observation points share one scan cell, those test points are defined as a test point group. Each test point group has its own structure. Based on the structure and the library, we can find the area of test point groups. For example, a control point group contains AND gates, multiplexers, and one source register, while an observation point group contains exclusive-OR gates and one sink register. Therefore, the structure of a test point group can be expressed as

$$G_{\text{ctl}} = N_{\text{and}} + N_{\text{mux}} + 1_{\text{mux\_reg}} \tag{5}$$

$$G_{\text{obs}} = N_{\text{xor}} + 1_{\text{mux\_reg}} \tag{6}$$

where G_{ctl} and G_{obs} denote the number of elements in a control point and observation point group, respectively. N_{and}, N_{mux}, and N_{xor} indicate the number of gates per group and $1_{\text{mux\_reg}}$ is the scan register shared by each group.

With technology scaling, the net interconnect area has become a significant part of the overall chip area. To estimate the net interconnect area introduced by TPI, we first analyze the synthesized design with all the test points inserted, and find the average net area caused by each cell. Then, we can estimate the net area for the design with efficient test points inserted. The steps can be expressed as

$$A_{\text{net\_per\_cell}} = A_{\text{all\_net}}/N_{\text{cell, all\_tp}} \tag{7}$$

$$A_{\text{net}} = A_{\text{net\_per\_cell}} \times N_{\text{cell, eff\_tp}} \tag{8}$$

where $A_{\text{net\_per\_cell}}$ is the average net area caused by each cell, $A_{\text{all\_net}}$ is the net interconnect area caused by all the test points, $N_{\text{cell, all\_tp}}$ is the number of cells caused by all the test points, A_{net} is the net interconnect area caused only by efficient test points, and $N_{\text{cell, eff\_tp}}$ is the number of cells caused only by efficient test points.

Once cell area and net interconnect area are obtained, the area overhead can be estimated as

$$\begin{aligned} A_{\text{overhead}} &= A_{\text{ctl\_cell}} + A_{\text{obs\_cell}} + A_{\text{net}} \\ &= A_{\text{ctl\_group}} \times N_{\text{ctl\_group}} \\ &\quad + A_{\text{obs\_group}} \times N_{\text{obs\_group}} \\ &\quad + A_{\text{net\_per\_cell}} \times N_{\text{cell, eff\_tp}} \end{aligned} \tag{9}$$

where $A_{\text{ctl\_cell}}$ and $A_{\text{obs\_cell}}$ are the cell area caused by control and observation points, respectively, $A_{\text{ctl\_group}}$ and $A_{\text{obs\_group}}$ are the area of each control and observation point group, respectively, and $N_{\text{ctl\_group}}$ and $N_{\text{obs\_group}}$ are the number of control and observation point groups, respectively.

B. Timing-Aware Metric for Area Overhead Estimation

To consider the impact on area overhead when test points are being inserted into paths with negative slack values, the timing-aware metric for area overhead estimation is developed.

Since the slack value for each control point is different, the area incurred by control points needs to be estimated individually instead of estimating the area by group.

Therefore, the area caused by each control point is estimated as

$$A_{\text{ctl}, i} = (A_{\text{all\_ctl}} - A_{\text{org}})/N_{\text{all\_ctl}} \times (1 - W_{\text{area\_slack}, i}) \tag{10}$$

$$W_{\text{area\_slack}, i} = \begin{cases} \text{slack}_i/\text{factor}_{\text{pos\_slack}}, & \text{if slack}_i > 0 \\ \text{slack}_i/\text{factor}_{\text{neg\_slack}}, & \text{otherwise} \end{cases} \tag{11}$$

where $A_{\text{ctl}, i}$ is the area caused by the ith control point, $A_{\text{all\_ctl}}$ is the area of the design with all the control points inserted, A_{org} is the area of the design without any test points inserted, $N_{\text{all\_ctl}}$ is the number of control points, $W_{\text{area\_slack}, i}$ is the weighted area based on the slack value of each control point, slack_i is the slack value of each control point, and $\text{factor}_{\text{pos\_slack}}$ and $\text{factor}_{\text{neg\_slack}}$ are the factors for positive and negative slack values, respectively. The calculation of these two factors is shown as follows.

First of all, $\text{factor}_{\text{pos\_slack}}$ is initialized as one, which can be expressed as

$$\text{factor}_{\text{pos\_slack}} = 1. \tag{12}$$

Furthermore, the area of the design with all the control points inserted ($A_{\text{all\_ctl}}$) can be expressed as

$$A_{\text{all\_ctl}} = A_{\text{pos\_ctl}} + A_{\text{neg\_ctl}} \tag{13}$$

$$A_{\text{pos\_ctl}} = \sum_{i=1}^{N_{\text{pos\_ctl}}} \frac{A_{\text{all\_ctl}} - A_{\text{org}}}{N_{\text{all\_ctl}}} \times \left(1 - \frac{\text{slack}_i}{\text{factor}_{\text{pos\_slack}}}\right) \tag{14}$$

$$A_{\text{neg\_ctl}} = \sum_{i=1}^{N_{\text{neg\_ctl}}} \frac{A_{\text{all\_ctl}} - A_{\text{org}}}{N_{\text{all\_ctl}}} \times \left(1 - \frac{\text{slack}_i}{\text{factor}_{\text{neg\_slack}}}\right) \tag{15}$$

where $A_{\text{pos\_ctl}}$ and $A_{\text{neg\_ctl}}$ are the area of the design with control points inserted with positive and negative slack values, respectively and $N_{\text{pos\_ctl}}$ and $N_{\text{neg\_ctl}}$ are the number of control points with positive and negative slack values, respectively.

Then, $\text{factor}_{\text{neg\_slack}}$ can be calculated as

$$\text{factor}_{\text{neg\_slack}} = \frac{\sum_{i=1}^{N_{\text{neg\_ctl}}} \text{slack}_i}{N_{\text{neg\_ctl}} - \frac{A_{\text{neg\_ctl}} \times N_{\text{all\_ctl}}}{A_{\text{all\_ctl}} - A_{\text{org}}}}. \tag{16}$$

Therefore, the area caused by efficient control points can be estimated as

$$A_{\text{eff\_ctl}} = \sum_{i=1}^{N_{\text{eff\_ctl}}} A_{\text{ctl}, i} \tag{17}$$

where $A_{\text{eff\_ctl}}$ is the area caused by efficient control points and $N_{\text{eff\_ctl}}$ is the number of efficient control points.

The area caused by each observation point is estimated as

$$A_{\text{unit\_obs}} = (A_{\text{all\_obs}} - A_{\text{org}})/N_{\text{all\_obs}} \tag{18}$$

where $A_{\text{unit\_obs}}$ is the average area caused by each observation point, $A_{\text{all\_obs}}$ is the area caused by all the observation points and $N_{\text{all\_obs}}$ is the number of observation points.

Then, the area caused by efficient observation point is estimated as

$$A_{\text{eff\_obs}} = A_{\text{unit\_obs}} \times N_{\text{eff\_obs}} \tag{19}$$

where $A_{\text{eff\_obs}}$ is the area caused by efficient observation points and $N_{\text{eff\_obs}}$ is the number of efficient observation points.

The estimation of the net interconnect area is similar as earlier. Once these values are obtained, the area overhead caused by efficient test points can be estimated as

$$A_{\text{overhead}} = A_{\text{eff\_ctl}} + A_{\text{eff\_obs}} + A_{\text{net}} \tag{20}$$

where A_{net} is the net interconnect area caused only by efficient tests points.

C. Timing-Aware Metric for Test Coverage Estimation

To consider the impact on test coverage when test points are being inserted into paths with negative slack values, the original proposed metric for test coverage estimation has been extended and developed as the timing-aware metric for test coverage estimation, which is explained in the following paragraphs.

To obtain the metric for test coverage estimation, we analyze the contributions to test coverage in designs with test points inserted. The contributions come mainly from four aspects: test coverage of original circuit without any test points inserted, test coverage gain provided by control and observation points separately, and test coverage gain provided by the interaction of control and observation points. It takes five steps to estimate test coverage with test points inserted.

Step 1: Calculate test coverage for designs with all the test points, without any test point, with only control points, with only observation points, and with 50% of control points and 50% of observation points inserted.

Step 2: Calculate test coverage improved by the efficient control and observation points separately. Based on the rules of the ETPI metric, we consider the ratio of the size of logic cones between efficient test points and all the test points, as well as the ratio of the number of ND faults within those cones between efficient test points and all the test points. Therefore, test coverage improved by efficient control points ($T_{\text{eff\_ctl}}$) is estimated as

$$T_{\text{eff\_ctl}} = R_{\text{fanout\_len}}{}^{k_{\text{ctl}}} \times R_{\text{NC\_faults}} \times T_{\text{all\_ctl}} \tag{21}$$

$$R_{\text{fanout\_len}} = \frac{\sum_{i=1}^{N_{\text{eff\_ctl}}} L_{\text{fanout\_len},\ i}}{\sum_{i=1}^{N_{\text{all\_ctl}}} L_{\text{fanout\_len},\ i}} \tag{22}$$

$$R_{\text{NC\_faults}} = \frac{\sum_{i=1}^{N_{\text{eff\_ctl}}} N_{\text{NC\_faults},\ i}}{\sum_{i=1}^{N_{\text{all\_ctl}}} N_{\text{NC\_faults},\ i}} \tag{23}$$

where $R_{\text{fanout\_len}}$ is calculated by the ratio of the size of fanout cones between efficient control points and all the control points, $R_{\text{NC\_faults}}$ is calculated by the ratio of the number of NC faults within those cones between efficient control points and all the control points, $T_{\text{all\_ctl}}$ is test coverage improved by all the control points, $L_{\text{fanout\_len},\ i}$ is the size of fanout cone of the ith control point, $N_{\text{NC\_faults},\ i}$ is the number of NC faults within that cone of the ith control point, $N_{\text{eff\_ctl}}$ is the number of efficient control points, and $N_{\text{all\_ctl}}$ is the number of the entire set of control points, k_{ctl} is the parameter for the ratio of the size of fanout cones between efficient control points and all the control points. The calculation of k_{ctl} will be discussed later.

Test coverage improved by efficient observation points is calculated in a similar way as control points. However, in this case, we look at fanin cones of observation points and NO faults within these cones, rather than fanout cones and NC faults used for control points. To be precise, test coverage improved by efficient observation points ($T_{\text{eff\_obs}}$) is estimated as

$$T_{\text{eff\_obs}} = R_{\text{fanin\_len}}{}^{k_{\text{obs}}} \times R_{\text{NO\_faults}} \times T_{\text{all\_obs}} \tag{24}$$

$$R_{\text{fanin\_len}} = \frac{\sum_{i=1}^{N_{\text{eff\_obs}}} L_{\text{fanin\_len},\ i}}{\sum_{i=1}^{N_{\text{all\_obs}}} L_{\text{fanin\_len},\ i}} \tag{25}$$

$$R_{\text{NO\_faults}} = \frac{\sum_{i=1}^{N_{\text{eff\_obs}}} N_{\text{NO\_faults},\ i}}{\sum_{i=1}^{N_{\text{all\_obs}}} N_{\text{NO\_faults},\ i}} \tag{26}$$

where $R_{\text{fanin\_len}}$ is calculated by fanin cones of observation points, $R_{\text{NO\_faults}}$ is calculated by not observed faults within those cones, $T_{\text{all\_obs}}$ is test coverage improved by all the observation points, $L_{\text{fanin\_len},\ i}$ is the size of fanin cone of the i_{th} observation point, $N_{\text{NO\_faults},\ i}$ is the number of not observed faults within that cone of the i_{th} observation point, $N_{\text{eff\_obs}}$ is the number of efficient observation points, and $N_{\text{all\_obs}}$ is the number of the entire set of observation points, k_{obs} is the parameter for the ratio of the size of fanin cones between efficient observation points and all the observation points. The calculation of k_{obs} will be discussed later.

The problem of calculating k_{ctl} and k_{obs} can be summarized to find the minimum of an objective function in the presence of bound constraints.

First of all, the objective function diff_func can be defined as

$$\text{diff\_func} = T_{\text{eff\_tp\_50\_actual}} - T_{\text{eff\_tp\_50\_est}} \tag{27}$$

where $T_{\text{eff\_tp\_50\_actual}}$ is the actual test coverage improved by 50% of control points and 50% of observation points obtained based on the ETPI metric, $T_{\text{eff\_tp\_50\_est}}$ is the estimated test coverage improved by those 50% of test points, and k_{ctl} and k_{obs} are two variables in the objective function.

Furthermore, since the ratio of test coverage improvement between 50% of control/observation points and all the control/observation points is less than 1 and higher than 0.5, (21) and (31) can be expressed as

$$0.5 < \frac{T_{\text{eff\_ctl\_50}}}{T_{\text{all\_ctl}}} = R_{\text{fanout\_len\_50}}{}^{k_{\text{ctl}}} \times R_{\text{NC\_faults\_50}} < 1 \tag{28}$$

$$0.5 < \frac{T_{\text{eff\_obs\_50}}}{T_{\text{all\_obs}}} = R_{\text{fanin\_len\_50}}{}^{k_{\text{obs}}} \times R_{\text{NO\_faults\_50}} < 1 \tag{29}$$

where $T_{\text{eff\_ctl\_50}}$ is the estimated test coverage improved by 50% of control points, $R_{\text{fanout\_len\_50}}$ is the ratio of the size of fanout cones between 50% of control points and all the control points, $R_{\text{NC\_faults\_50}}$ is the ratio of the number of NC faults within those cones between 50% of control points and all the control points, $T_{\text{eff\_obs\_50}}$ is the estimated test coverage improved by 50% of observation points, $R_{\text{fanin\_len\_50}}$ is the ratio of the size of fanin cones between 50% of observation points and all the observation points, and $R_{\text{NO\_faults\_50}}$ is the ratio of the number of NO faults within those cones between 50% of observation points and all the observation points.

Then, the bound constraints for k_{ctl} and k_{obs} can be calculated based on the above inequalities, and can be expressed as

$$\frac{\log_{10} 0.5/R_{\text{NC\_faults\_50}}}{\log_{10} R_{\text{fanout\_len\_50}}} < k_{\text{ctl}} < 0 \tag{30}$$

$$\frac{\log_{10} 0.5/R_{\text{NO\_faults\_50}}}{\log_{10} R_{\text{fanin\_len\_50}}} < k_{\text{obs}} < 0. \tag{31}$$

Therefore, the problem of calculating k_{ctl} and k_{obs} can be formulated as

$$\begin{aligned} &\min \text{ diff\_func} \\ &\text{s.t. } k_{\text{ctl}-\min} < k_{\text{ctl}} < k_{\text{ctl}-\max} \\ &\qquad k_{\text{obs}-\min} < k_{\text{obs}} < k_{\text{obs}-\max} \end{aligned} \tag{32}$$

where $k_{\text{ctl}-\min}$ and $k_{\text{ctl}-\max}$ are the lower and upper bounds of k_{ctl}, respectively, and $k_{\text{obs}-\min}$ and $k_{\text{obs}-\max}$ are the lower and upper bounds of k_{obs}, respectively, which are obtained from (30) and (31).

Step 3: To estimate test coverage, it is important to combine test coverage obtained from control points and from observation points. For instance, if test coverage improvement contributed by observation points is less than control points, test coverage improved by observation points will be weighed by the test coverage improvement ratio $R_{\text{ctl,obs}}$, and vice versa. The reason to estimate test coverage improvement in this way, rather than to sum test coverage improvement obtained from control and observation points, is when control and observation points work together, they will influence each other, e.g., not only does a control point improve the controllability of its fanout cone, but also it affects the observability of its fanin cone. Therefore, test coverage improvement ratio between control and observation points is considered and calculated as

$$R_{\text{ctl, obs}} = \begin{cases} T_{\text{all\_obs}}/T_{\text{all\_ctl}}, & \text{if } T_{\text{all\_obs}}/T_{\text{all\_ctl}} < 1 \\ T_{\text{all\_ctl}}/T_{\text{all\_obs}}, & \text{otherwise.} \end{cases} \tag{33}$$

Step 4: Since test coverage is improved by the interaction of control and observation points as well, the mutual factor is considered and calculated as

$$\alpha = T_{\text{all\_tp}} - T_{\text{org}} - T_{\text{ctl\_obs}} \tag{34}$$

where $T_{\text{all\_tp}}$ is test coverage obtained from the design with all the test point candidates inserted, T_{org} is test coverage obtained from the original design, and $T_{\text{ctl\_obs}}$ is test coverage improvement obtained from control and observation points.

Step 5: Test coverage of a design with efficient test points inserted is estimated as

$$T_{\text{eff\_tp}} = \begin{cases} T_{\text{org}} + T_{\text{eff\_ctl}} + R_{\text{ctl, obs}} \times T_{\text{eff\_obs}} + \alpha \\ \quad \text{if } T_{\text{all\_obs}}/T_{\text{all\_ctl}} < 1 \\ T_{\text{org}} + T_{\text{eff\_obs}} + R_{\text{ctl, obs}} \times T_{\text{eff\_ctl}} + \alpha, & \text{otherwise.} \end{cases} \tag{35}$$

TABLE I

BENCHMARK CHARACTERISTICS

Benchmarks	Number of Components			
	FFs	Combinational	Buffer/Inverter	Transistor
s38417	1,564	3,746	573	34,813
mem_ctrl	1,065	3,127	483	26,218
usb_funct	1,744	5,453	659	45,013
dma	1,838	5,862	1,013	47,115
vga_lcd	17,057	30,354	6,042	362,137
b19	6,042	51,936	8,095	310,964
leon_proc	1,067	6,048	1,071	39,425
leon3s	17,495	37,640	6,207	377,796

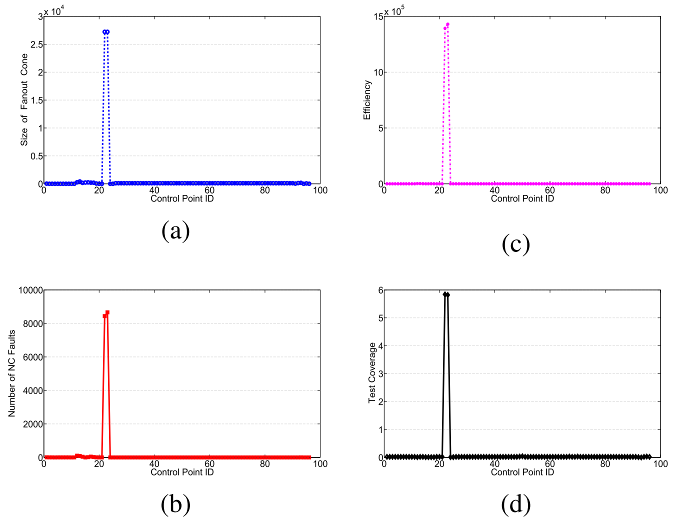


Fig. 5. Efficiency and test coverage of control points in *vga_lcd*. (a) Size of fanout cone for each ctl. (b) Number of NC faults within fanout cone for each ctl. (c) Efficiency of each ctl (E_{ctl}). (d) Test coverage of each ctl (T_{ctl}).

If $T_{\text{all\_obs}}/T_{\text{all\_ctl}} < 1$, $R_{\text{ctl, obs}}$ estimates test coverage improved by observation points with control points, and vice versa. α is adjusted by $R_{\text{fanout\_len}}$ and $R_{\text{fanin\_len}}$ based on the TPI scheme.

V. SIMULATION RESULTS AND ANALYSIS

The proposed techniques have been implemented and verified on NXP Semiconductors designs and using 32-nm technology node on several benchmark circuits from Gaisler, OPENCORE, ITC'99, and ISCAS85. The number of cells for each benchmark with scan chain inserted is listed in Table I, e.g., transistors, FFs, combinational cells, inverters, and buffers.

The tools used to generate test points are commercially available tools, including Synopsys Tetramax and Cadence Encounter RTL Compiler, which are standard in the DFT flow commercially used in the industry. Moreover, we follow the same flow used in state-of-the-art DFT insertion process when generating the initial set of test points in the experiment.

Control points play a major role in improving testability of vga_lcd benchmark circuit. Fig. 5 shows the efficiency and the actual test coverage of control points in vga_lcd. The efficiency of control points (E_{ctl}) is mainly determined by the size of their logic cones and the number of NC faults within those cones. Fig. 5(a) shows that those control points with a large fanout cone size will usually have a large number of NC faults,

TABLE II
TEST-POINT AREA OVERHEAD AND TEST COVERAGE REDUCTION

Benchmarks	ctl_obs_del	sim_tp_ao_red (%)	area overhead reduction (%)			test coverage reduction (%)		
			est_ao_red	sim_ao_red	diff_ao	est_tc_red	sim_tc_red	diff_tc
s38417 AOtp 2.552%	ctl_85_obs_65	72	1.888	1.834	-0.054	-1.228	-1.720	-0.492
	ctl_75_obs_65	68	1.776	1.742	-0.034	-1.015	-1.330	-0.315
	ctl_85_obs_55	66	1.746	1.692	-0.054	-1.012	-1.206	-0.248
mem_ctrl AOtp 4.375%	ctl_95_obs_45	61	2.710	2.660	-0.050	-1.567	-1.590	-0.023
	ctl_65_obs_55	59	2.606	2.595	-0.011	-1.743	-1.720	0.023
	ctl_85_obs_45	57	2.562	2.490	-0.071	-1.288	-1.580	-0.292
usb_func AOtp 2.037%	ctl_85_obs_95	88	1.830	1.794	-0.036	-2.302	-1.610	0.692
	ctl_95_obs_85	86	1.807	1.755	-0.053	-1.758	-1.140	0.618
	ctl_85_obs_85	83	1.720	1.684	-0.036	-1.743	-1.140	0.603
dma AOtp 12.442%	**ctl_95_obs_95**	**95**	**11.802**	**11.774**	**-0.027**	**-1.155**	**-1.000**	**0.155**
	ctl_85_obs_95	92	11.471	11.447	-0.024	-1.125	-0.680	0.445
	ctl_75_obs_95	89	11.141	11.118	-0.024	-1.113	-0.480	0.633
vga_lcd AOtp 1.758%	**ctl_95_obs_95**	**95**	**1.669**	**1.666**	**-0.003**	**-1.058**	**-0.570**	**0.488**
	ctl_85_obs_95	93	1.641	1.637	-0.005	-0.881	-0.430	0.451
	ctl_75_obs_95	92	1.617	1.613	-0.004	-0.799	-0.430	0.369
b19 AOtp 4.057%	ctl_35_obs_85	59	2.416	2.413	-0.003	-3.564	-1.720	1.844
	ctl_45_obs_70	57	2.323	2.318	-0.005	-3.455	-1.460	1.995
	ctl_45_obs_65	55	2.224	2.220	-0.004	-3.105	-1.230	1.875
leon_proc AOtp 4.136%	ctl_95_obs_75	78	3.282	3.216	-0.065	-2.147	-1.890	0.257
	ctl_75_obs_75	74	3.110	3.076	-0.033	-1.624	-1.720	-0.096
	ctl_65_obs_75	72	3.011	2.995	-0.016	-1.507	-1.540	-0.033
leon_3s AOtp 3.730%	ctl_40_obs_95	88	3.277	3.272	-0.004	-1.396	-1.480	-0.084
	ctl_35_obs_95	87	3.251	3.248	-0.004	-1.070	-1.390	-0.320
	ctl_25_obs_95	86	3.205	3.199	-0.006	-0.815	-0.090	0.725

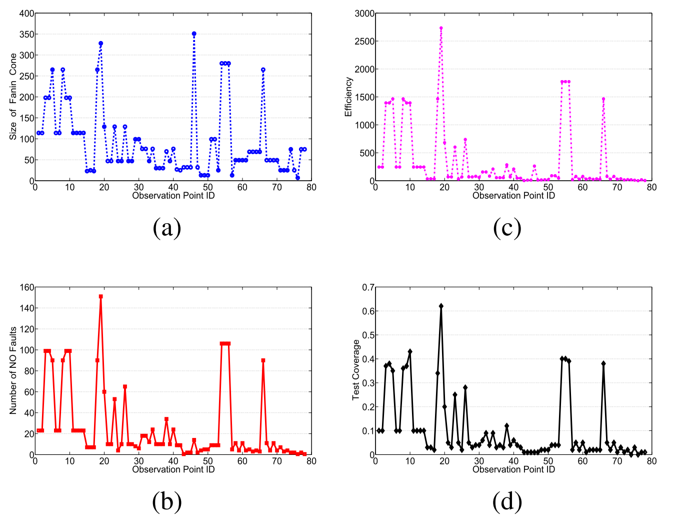


Fig. 6. Efficiency and test coverage of observation points in s38417. (a) Size of fanin cone for each obs. (b) Number of NO faults within fanin cone for each obs. (c) Efficiency of each obs (E_{obs}). (d) Test coverage of each obs (T_{obs}).

as shown in Fig. 5(b). Therefore, control points inserted at these locations will have a higher efficiency than other control point locations, as shown in Fig. 5(c), and will also have a larger impact on test coverage, as shown in Fig. 5(d).

Similarly, testability improvement on s38417 benchmark circuit is affected mostly by observation points. Fig. 6 shows the efficiency and the actual test coverage of observation points in s38417. The efficiency of observation points is mainly determined by the size of their logic cones and the number NO faults within those cones. Fig. 6(a) shows that those observation points with a large fanin cone size will usually have a large number of NO faults, as shown in Fig. 6(b). Therefore, observation points inserted at these locations will have a higher efficiency than other observation point locations, as shown in Fig. 6(c), and will also have a larger impact on test coverage, as shown in Fig. 6(d).

As shown in the two figures, test points with large logic cones will be more beneficial to the improvement of random pattern testability if the number of ND faults in those cones is large as well. It can be observed that, the efficiency of test points is consistent with their contribution to test coverage. Also, it can be found that many test points have little or no impact on test coverage.

Table II shows the experimental results on a number of benchmarks after applying the proposed metrics. To meet the needs of practical applications, different test point removal schemes, i.e., different combinations of control and observation points removed have been applied to each design. Column 1 specifies the name of each design and the total postsynthesis area overhead with all the test points inserted ($A_{\text{overhead, all\_tp}}$). Due to synthesis and optimization, design area before inserting test points and after inserting test points could be much different. In column 2, *ctl* and *obs* refer to control and observation points, respectively. The numbers following *ctl* and *obs* indicate the percentage of control and observation points removed by ETPI, respectively. For example, "ctl_75_obs_95" scheme indicates ETPI removed 75% of control points and 95% of observation points. Column 3 shows the percentage of area reduction of all the test points' area when applying ETPI to test points inserted by tools. Every time test points are removed, synthesis and fault simulation are performed in order to obtain the actual area overhead (i.e., simulation area overhead) and the actual test coverage (i.e., simulation test coverage) shown in columns 5 and 8, respectively. From columns 4 to 9, *est_TPRE* and *sim_ETPI* refer to the estimation values from TPRE and simulation results after ETPI, respectively, *diff* refers to the difference between estimation and simulation results which represents the accuracy of the estimation calculated by the (TPRE)

TABLE III
COMPARISON BETWEEN ETPI, RANDOM AND COMMERCIAL TOOL

Benchmarks	ctl_obs_del	sim_tp_ao_red (ETPI) (%)	area overhead reduction (%)			test coverage reduction (%)		
			ETPI	Random	Tool	ETPI	Random	Tool
s38417 AOtp 2.552%	ctl_85_obs_65	72	1.834	1.834	1.834	-1.720	-4.810	-2.160
	ctl_75_obs_65	68	1.742	1.742	1.742	-1.330	-4.790	-2.110
	ctl_85_obs_55	66	1.692	1.692	1.692	-1.206	-3.570	-1.560
mem_ctrl AOtp 4.375%	ctl_95_obs_45	61	2.660	2.660	2.660	-1.590	-4.410	-2.790
	ctl_65_obs_55	59	2.595	2.595	2.595	-1.720	-4.390	-2.250
	ctl_85_obs_45	57	2.490	2.491	2.490	-1.580	-3.990	-2.320
usb_func AOtp 2.037%	ctl_85_obs_95	88	1.794	1.794	1.794	-1.610	-2.340	-2.290
	ctl_95_obs_85	86	1.755	1.755	1.783	-1.140	-2.350	-1.850
	ctl_85_obs_85	83	1.684	1.684	1.684	-1.140	-2.320	-1.900
dma AOtp 12.442%	ctl_95_obs_95	95	11.774	11.773	11.774	-1.000	-2.840	-1.180
	ctl_85_obs_95	92	11.447	11.445	11.447	-0.680	-2.850	-1.190
	ctl_75_obs_95	89	11.118	11.116	11.118	-0.480	-2.600	-0.690
vga_lcd AOtp 1.758%	**ctl_95_obs_95**	**95**	**1.666**	**1.663**	**1.663**	**-0.570**	**-11.870**	**-11.890**
	ctl_85_obs_95	93	1.637	1.638	1.637	-0.430	-11.950	-11.990
	ctl_95_obs_85	92	1.613	1.514	1.513	-0.430	-11.680	-11.650
b19 AOtp 4.057%	**ctl_35_obs_85**	**59**	**2.413**	**2.413**	**2.413**	**-1.720**	**-7.240**	**-10.250**
	ctl_45_obs_70	57	2.318	2.318	2.318	-1.460	-7.840	-11.750
	ctl_45_obs_65	55	2.220	2.224	2.220	-1.230	-7.060	-11.070
leon_proc AOtp 4.136%	ctl_95_obs_75	78	3.216	3.216	3.216	-1.890	-3.210	-3.160
	ctl_75_obs_75	74	3.076	3.076	3.077	-1.720	-3.110	-2.220
	ctl_65_obs_75	72	2.995	2.995	2.995	-1.540	-3.060	-2.180
leon_3s AOtp 3.730%	ctl_40_obs_95	88	3.272	3.274	3.274	-1.480	-7.580	-2.740
	ctl_35_obs_95	87	3.248	3.249	3.249	-1.390	-7.280	-2.500
	ctl_25_obs_95	86	3.199	3.203	3.203	-0.090	-5.890	-1.330

metric. Columns 4 and 5 show the estimation values and simulation results for the percentage of area reduction of the total area when applying our framework to test points inserted by commercial tool. Column 6 shows the difference between estimation (TPRE) and simulation (ETPI) results. Columns 7 and 8 show the estimation values and simulation results for test coverage reduction. The difference between these two values is shown in column 9.

The results from different circuits demonstrate that removing most test points based on the evaluation framework have little impact on test coverage but reduce area overhead significantly. For example, when 95% of control points and 95% of observation points inserted in dma generated by tools are removed (row 10), test coverage reduction is only 1.00%, while the TPI area is reduced up to 95%, i.e., area overhead is reduced from 12.442% to 0.668%. When 95% of control points and 95% of observation points generated by TPI tools for vga_lcd are removed (row 13), test coverage reduction is only 0.57%, but the TPI area is reduced up to 95%, i.e., area overhead is reduced from 1.758% to 0.092%.

As shown in columns 3 and 8, the area reduction of all the test points' area ranges from 57% to 95% with test coverage loss ranges from 0.09% to 1.89%. The results can be influenced by the structure of each design and combinations of test points, however, a tradeoff between area overhead and test coverage can always be obtained. More importantly, it is essential to remove the least efficient test points, so as to reduce area overhead significantly, while achieving very limited test coverage loss in many critical and high-assurance applications. Otherwise, for optimized designs, gates and faults introduced by test points can sometimes outweigh the benefits due to negative test coverage and increased area overhead.

It can be observed that estimated values of the percentage of area reduction of the total area are slightly greater than the simulation results (columns 4–6). The results indicate the difference between them is limited to less than 0.10% for the most cases. The reason of this difference comes from several aspects. First, the wire load models are used to estimate the effect of interconnect nets on capacitance, resistance, and area, before real data are obtained from the actual layout. Because these models are statistical models, which estimate the wire length as a function of net's fanout, they could cause the area estimation error. More than that, since the dedicated registers are inserted as scan cells for test points, physical synthesis might add extra clocking buffers to drive those registers.

As can be seen from Table II, there are differences between estimation values and simulation results of test coverage. The results indicate the difference between them is limited to less than 1.00% for the most cases. These are due to test coverage being influenced by many factors, including the size of logic cones, the number of ND faults, timing constraints, tool's optimization algorithm, test point placement and so on. Specifically, test coverage is impacted by components and net interconnect of each design.In addition, since extra registers and more clocking buffers are added during TPI, they can have a large impact on physical synthesis, and hence, test coverage. More than that, the tool's timing and power optimization algorithms optimize each design differently relying on its structure and constraints. Even for the same design, since each time different test points are removed, the tool synthesizes each implementation differently depending on which path is being affected by test point placement. All of these factors could impact test coverage.

Table III shows the experimental results on different designs with different techniques, e.g., the ETPI metric, the random selection method, and the commercial tool. The random selection method refers to the method that test points are randomly removed from the set of test point candidates. The Commercial tool refers to the method that commercial tools are configured to insert fewer test points to match the number

TABLE IV
TEST-POINT AREA OVERHEAD AND TEST COVERAGE REDUCTION FOR NXP SEMICONDUCTORS DESIGNS

Circuits	ctl_obs_del	sim_tp_ao_red (%)	area overhead reduction (%)			test coverage reduction (%)		
			est_TPRE	sim_ETPI	diff_ao	est_TPRE	sim_ETPI	diff_tc
Design A AOtp 2.451%	ctl_10_obs_10	21.543	0.345	0.528	0.183	-0.825	-0.280	0.545
	ctl_10_obs_20	18.095	0.439	0.443	0.004	-1.236	-0.660	0.576
	ctl_20_obs_10	21.336	0.533	0.523	-0.010	-1.035	-0.740	0.295
	ctl_20_obs_20	27.681	0.626	0.678	0.052	-1.447	-1.110	0.337
	ctl_30_obs_10	29.833	0.716	0.731	0.015	-1.304	-1.200	0.104
	ctl_30_obs_20	41.719	0.809	1.022	0.213	-1.716	-1.450	0.266
	ctl_40_obs_10	35.855	0.865	0.879	0.014	-1.554	-1.300	0.254
	ctl_40_obs_20	44.122	0.959	1.081	0.122	-1.965	-1.630	0.335
Design B AOtp 1.486%	ctl_10_obs_10	15.953	0.266	0.237	-0.029	-0.692	-0.610	0.082
	ctl_10_obs_20	15.651	0.380	0.233	-0.147	-1.423	-1.290	0.133
	ctl_20_obs_10	19.674	0.334	0.292	-0.042	-1.545	-1.270	0.275
	ctl_20_obs_20	26.119	0.447	0.388	-0.059	-2.276	-1.910	0.366
	ctl_30_obs_10	21.955	0.374	0.326	-0.048	-1.779	-2.000	-0.221
	ctl_30_obs_20	28.367	0.487	0.422	-0.065	-2.509	-2.470	0.039

of test points removed by ETPI. To be consistent with Table II, the same test point removal schemes have been applied to each design as well as each method. Columns 1–3 have the same meaning as they did in Table II. Columns 4–6 show the percentage of area reduction of the total area after applying the different TPI methods, and columns 7–9 show their resulting test coverage reduction.

From this comparison, it can be observed that the percent area reduction for all three methods is almost the same when the same percentage of test points are removed. However, test coverage loss of the proposed metric, i.e., ETPI metric, is always lower than both the random selection method and the commercial tool. Sometimes test coverage loss induced by ETPI metric is much lower than others. For example, test coverage reduction of ETPI metric for vga_lcd is only within 0.60%, while for other methods, it is larger than 11.00% (rows 13, 14, and 15). For b19, test coverage reduction of ETPI metric is between 1.00% and 2.00%, but for other methods, it is between 7.00% and 12.00% (rows 16, 17, and 18).

Table IV shows the experimental results on NXP Semiconductors designs. All the columns have the same meaning as in Table II. As can be seen from it, test point area overhead of Design A ranges from 18.095% to 44.122% with test coverage reduction from 0.280% to 1.630%, and test point area overhead of Design B ranges from 15.651% to 28.367% with test coverage reduction from 0.610% to 2.470%. The results by applying the TPRE metric indicate that the difference between estimation values and actual simulation results for area overhead is less than 0.20% for most cases, and the difference between them for test coverage is less than 0.60% for most cases.

It can be seen that the differences between estimation values and simulation results of area overhead are slightly larger than the benchmark circuits. These differences are mainly due to additional paths and gates created as well as gates resized during optimization, since, a large number of control points are inserted in the failing paths.

It can also be observed that in few cases removing more test points (e.g., ctl_10_obs_10 versus ctl_20_obs_10) does not translate to the higher area reduction as it would be expected. The few exceptional cases are mainly because the synthesis tools optimize each design differently depending on which

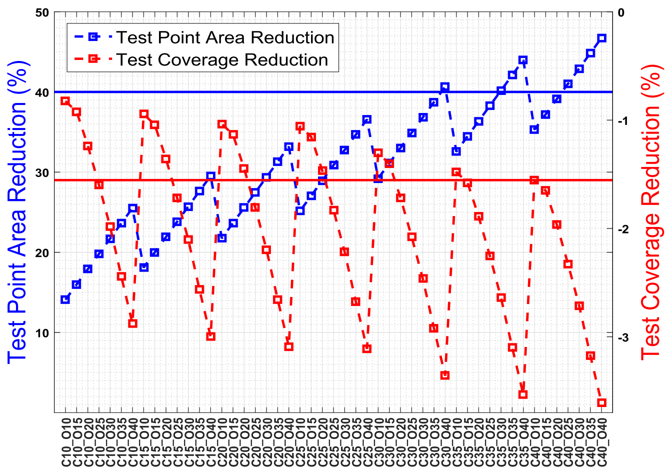


Fig. 7. Estimation results for Design A. (C: ctl, O: obs).

paths do not meet timing constraints. To be exact, even for the same design, since each time different test points are inserted, the tool optimizes each implementation differently based on the paths being affected by these test points.

As introduced in the previous section, the TPRE metric can generate the optimal configuration of test points based on different requirements, i.e., area overhead and test coverage. For example, if the requirement of area overhead is passed into the flow of TPRE metric as an argument, the test point combination that has the required area overhead with the minimum test coverage loss will be generated. Likewise, if the requirement of test coverage is passed into the flow of TPRE metric as an argument, the test point combination that has the required test coverage with the maximum area overhead reduction will be generated.

Fig. 7 shows the TPRE results for Design A from ctl_-10_obs_10 to ctl_40_obs_40 at 5% increments. For example, if we want to obtain the optimal test point removal scheme with test point area reduction around 40.000% indicated as the blue line, ctl_30_obs_35, ctl_30_obs_40, ctl_35_obs_30, ctl_40_obs_20, and ctl_40_obs_25 might be the choices. Based on the estimation results, ctl_40_obs_20 would be the best option since it could give us the required area reduction with the minimum test coverage reduction, i.e., 1.965%.

TABLE V
COMPARISON BETWEEN TPRE AND SAMPLING

Benchmarks	Number of Syntheses and Faults Simulations		CPU Time(s)		Sampling/ TPRE
	TPRE	Sampling	TPRE	Sampling	
b19	5	400	2548	194400	76.30X
vga_lcd	5	400	2343	174000	74.26X
leon_3s	5	400	4537	346000	76.26X
Design A	5	400	180630	12216000	67.63X
Design B	5	400	306510	24960000	81.43X

Likewise, if we want to obtain the optimal test point removal scheme with test coverage loss around 2.000%, indicated as the red line, ctl_10_obs_30, ctl_15_obs_30, ctl_30_obs_25, ctl_35_obs_20, and ctl_40_obs_20 might be the choices. Based on the estimation results, ctl_40_obs_20 would be the best option, since, it could give us the required test coverage loss with the maximum area overhead reduction, i.e., 0.959%. The estimation values and actual simulation results of ctl_40_obs_20 is shown in Table IV.

If we set the sampling interval between control points or observation points is 1% instead of 5%, the TPRE metric can provide us estimation results for each test point combination with 1% interval. Therefore, as can be seen from the figure, instead of running thousands of lengthy syntheses and fault simulations, the TPRE metric allows the designer to approximate the area overhead and the test coverage without the actual insertion of test points, thereby, providing the optimal reference of test point removal scheme accurately and rapidly.

Despite small differences between estimated values and simulation results, the TPRE metric is able to accurately and rapidly provide us with the optimal combination of control and observation points to remove in order to maximize area overhead reduction and minimize test coverage loss. Without the proposed TPRE metric, all the available combinations of control and observation points removed would have to be implemented and synthesized on the design in order to find the best configuration. A few selected combinations of test points could be applied, but this would not be optimal or practical, especially for large designs. Table V shows the comparison between the TPRE metric and a sampling approach. In the experiment, sampling refers to the process of synthesizing a design and performing fault simulation for each combination of test points exactly once. Specifically, to estimate area overhead and test coverage of TPI, the TPRE metric needs to perform synthesis and fault simulation of each design exactly five times, e.g., the design with all the test points, without any of them, with only control points, with only observation points, and with 50% of control points and 50% of observation points inserted. However, for the sampling approach, setting the control and observation points removed to range from 5% to 95% and setting the sampling interval to 5%, it may take $20 \times 20 = 400$ syntheses and fault simulations to find the best combination. In Table IV, CPU time refers to the amount of CPU time spent on executing the process. For the TPRE metric, CPU time is obtained by measuring the exact time spent on the synthesis and fault simulation of each design five times. For the sampling approach, since it is impractical to obtain CPU time through running 400 times syntheses and fault simulations, CPU time is obtained by 400 × CPU_time_for_ctl_50_obs_50, where CPU_time_for_ctl_50_obs_50 refers to the amount of CPU time spent on performing a single synthesis and fault simulation for the design inserted with 50% of control points and 50% of observation points. Column 4 is the ratio between CPU time required by sampling and the TPRE metric. It can be found that the proposed TPRE metric achieves more than 60 times speedup over the sampling approach on b19, vga_lcd, leon_3s, and industrial designs. As circuits become larger, the difference in CPU time between two approaches becomes more significant. More importantly, in critical and high-assurance applications, such as automotive applications, the design size is similar as or even larger than the industrial designs shown in the table. The CPU time spent on performing synthesis and fault simulation only once can exceed 24 h, thus, performing multiple syntheses and fault simulations is unacceptable. In those cases, the TPRE metric will be a better way to provide us the optimal configuration of control and observation points.

VI. CONCLUSION

In this paper, we proposed a timing-aware evaluation framework to aid TPI. This framework considers not only individual test coverage improvement but also area overhead, path slack, critical paths, and the effect of test points on these paths, therefore to optimize the test point area overhead and test coverage while meeting the significant timing constraints in industrial designs used for critical and high-assurance applications. Within this framework, two metrics were developed. To be specific, the ETPI metric was developed to ensure any improvement in test coverage gained from inserting test points was not outweighed by the additional area added to the design to fix timing violations caused by TPI. The TPRE metric was introduced to provide us the optimal reference of test point removal scheme accurately and rapidly. Experimental results on NXP Semiconductors circuits, Gaisler, OPENCORE, ITC'99, and ISCAS85 circuits demonstrated that the methodology proposed in this paper was able to implement TPI effectively while meeting the test time and optimizing area overhead and test coverage.

REFERENCES

[1] P. Bardel and W. McAnney, "Self-testing of multiple chip module," in *Proc. Int. Test Conf.*, 1982, pp. 200–204.
[2] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Embedded deterministic test," *IEEE Trans. Comput.-Aided Design Integr.*, vol. 23, no. 5, pp. 776–792, May 2004.
[3] G. Contreras, Y. Zhao, N. Ahmed, L. Winemberg, and M. Tehranipoor, "LBIST pattern reduction by learning ATPG test cube properties," in *Proc. Int. Symp. Quality Electron. Design*, 2015, pp. 147–153.
[4] G. Contreras, L. Winemberg, M. Tehranipoor, and N. Ahmed, "Predictive LBIST model and partial ATPG for seed extraction," in *Proc. Int. Symp. DFT*, 2015, pp. 139–146.
[5] H. J. Wunderlich, "Multiple distribution for biased random test patterns," in *Proc. Int. Test Conf.*, 1988, pp. 584–593.
[6] F. Brglez, C. Gloster, and G. Kedem, "Hardware-based weighted random pattern generation for boundary scan," in *Proc. Int. Test Conf.*, 1989, pp. 264–274.

[7] M. Bershteyn, "Calculation of multiple sets of weights for weighted random testing," in *Proc. Int. Test Conf.*, 1993, pp. 1031–1040.

[8] A. Jas, C. V. Krishna, and N. A. Touba, "Hybrid BIST based on weighted pseudo-random testing: A new test resource partitioning," in *Proc. VLSI Test Symp.*, 2001, pp. 2–8.

[9] S. Pateras and J. Rajski, "Cube-contained random patterns and their application to the complete testing of synthesized multi-LE," in *Proc. Int. Test Conf.*, 1991, pp. 473–482.

[10] N. A. Touba and E. J. McCluskey, "Altering a pseudo-random bit sequence for scan-based BIST," in *Proc. Int. Test Conf.*, 1996, pp. 167–175.

[11] R. Kapur, S. Patil, T. J. Snethen, and T. W. Williams, "Design of an efficient weighted random pattern generation system," in *Proc. Int. Test Conf.*, 1994, pp. 491–500.

[12] L. Lai, J. H. Patel, T. Rinderknecht, and W.-T. Cheng, "Hardware efficient LBIST with complementary weights," in *Proc. Int. Conf. Comput. Design*, 2005, pp. 479–481.

[13] S. Hellebrand, S. Tarnick, J. Rajski, and B. Courtois, "Generation of vector patterns through reseeding of multiple-polynomial linear feedback shift register," in *Proc. Int. Test Conf.*, 1992, pp. 120–129.

[14] 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.

[15] B. Koenemann, "LFSR-coded test patterns for scan designs," in *Proc. Eur. Test Conf.*, 1991, pp. 237–242.

[16] B. Koenemann, C. Barnhart, B. Keller, T. Snethen, O. Farnsworth, and D. Wheater, "A SmartBIST variant with guaranteed encoding," in *Proc. VLSI Test Symp.*, 2001, pp. 325–330.

[17] C. V. Krishna, A. Jas, and N. A. Touba, "Test vector encoding using partial LFSR reseeding," in *Proc. Int. Test Conf.*, 2001, pp. 885–893.

[18] J. Rajski *et al.*, "Embedded deterministic test for low cost manufacturing test," in *Proc. Int. Test Conf.*, 2002, pp. 301–310.

[19] J. P. Hayes and A. D. Friedman, "Test point placement to simplify fault detection," in *Proc. Int. Fault-Tolerant Comput. Symp.*, 1973, pp. 73–78.

[20] A. J. Briers and K. A. E. Totton, "Random pattern testability by fast fault simulation," in *Proc. Int. Test Conf.*, 1986, pp. 274–281.

[21] V. S. Iyengar and D. Brand, "Synthesis of pseudo-random pattern testable designs," in *Proc. Int. Test Conf.*, 1989, pp. 501–508.

[22] N. A. Touba and E. J. McClusky, "Test point insertion based on path tracing," in *Proc. VLSI Test Symp.*, 1996, pp. 2–8.

[23] M. J. Guezebroek, J. T. van der Linden, and A. J. van de Goor, "Test point insertion that facilitates ATPG in reducing test time and data," in *Proc. Int. Test Conf.*, 2002, pp. 138–147.

[24] B. Seiss, P. Trouborst, and M. Schulz, "Test point insertion for scan-based BIST," in *Proc. Eur. Design Autom. Conf.*, 1991, pp. 253–262.

[25] Y. Savaria, M. Yousef, B. Kaminska, and M. Koudil, "Automatic test point insertion for pseudo-random testing," in *Proc. Int. Symp. Circuits Syst.*, 1991, pp. 1960–1963.

[26] K.-T. Cheng and C.-J. Lin, "Timing-driven test point insertion for full-scan and partial-scan BIST," in *Proc. Int. Test Conf.*, 1995, pp. 506–514.

[27] C. J. Lin, Y. Zorian, and S. Bhawmik, "PSBIST: A partial-scan based built-in self-test scheme," in *Proc. Int. Test Conf.*, 1993, pp. 507–516.

[28] H. Vranken, F. S. Sapei, and H. J. Wunderlich, "Impact of test point insertion on silicon area and timing during layout," in *Proc. Design Autom. Test Eur. Conf.*, 2004, pp. 810–815.

[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] H. Ren, M. Kusko, V. Kravets, and R. Yaari, "Low cost test point insertion without using extra registers for high performance design," in *Proc. Int. Test Conf.*, 2009, pp. 1–8.

[31] 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. Asian Test Symp.*, 2006, pp. 339–348.

[32] R. Sethuram, S. Wang, S. T. Chakradhar, and M. L. Bushnell, "Zero cost test point insertion technique for structured ASICs," in *Proc. 20th Int. Conf. VLSI Design Held Jointly 6th Int. Conf. Embedded Syst.*, 2007, pp. 357–363.

[33] N. Tamarapalli and J. Rajski, "Constructive multi-phase test point insertion for scan-based BIST," in *Proc. Int. Test Conf.*, 1996, pp. 649–658.

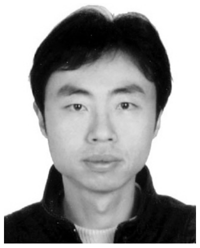
[34] M. Nakao, K. Hatayama, and I. Higashi, "Accelerated test points selection method for scan-based BIST," in *Proc. 6th Asian Test Symp.*, 1997, pp. 359–364.

[35] M. Nakao, S. Kobayashi, K. Hatayama, and K. Iijima, "Low overhead test point insertion for scan-based BIST," in *Proc. Int. Test Conf.*, 1999, pp. 357–384.

[36] M. Bushnell and V. Agrawal, *Essentials of Electronics Testing*. Norwell, MA, USA: Kluwer, 2000.

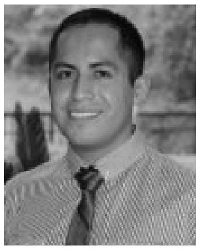
[37] M. T. He, G. Contreras, M. Tehranipoor, D. Tran, and L. Winemberg, "Test point insertion efficiency analysis for LBIST applications," in *Proc. VLSI Test Symp. (VTS)*, Apr. 2016, pp. 1–6.

[38] *DFT Compiler, DFTMAX, and DFTMAX Ultra User Guide, Version J*, Synopsys, Mountain View, CA, USA, Sep. 2014.



Miao (Tony) He (S'16) received the M.S. degree in electronic engineering from Tsinghua University, Beijing, China, and the B.S. degree in electronic engineering from the University of Electronic Science and Technology of China, Chengdu, China. He is currently pursuing the Ph.D. degree in computer engineering with the University of Florida, Gainesville, FL, USA.

His current research interests include computer-aided design and test for CMOS VLSI designs, and hardware security and trust, including test structure, test points, secure test, side-channel attacks, and logic encryption.



Gustavo K. Contreras (S'12) received the B.S. (*summa cum laude*) degree in electrical engineering, the B.S. (*summa cum laude*) degree in computer engineering, and the M.S. degree in computer engineering from the University of Connecticut, Storrs, CT, USA, in 2011 and 2015, respectively. He is currently pursuing the Ph.D. degree with the Electrical and Computer Engineering Department, University of Florida, Gainesville, FL, USA.

His current research interests include VLSI test and hardware security, including DFT, LBIST, test points, and in-field test and secure test and information flow tracking through hardware.

Dat Tran received the B.S.E.E. degree from The University of Texas at Austin, Austin, TX, USA.

He is currently a Design for Testing (DFT) Staff Member of the NXP's Automotive Microcontroller and Processors Research and Development Group, NXP Semiconductors, Austin, where he has been involved in implementing DFT on several automotive microcontrollers and embedded cores.

LeRoy Winemberg received the B.S.E.E. degree from the University of Notre Dame, Notre Dame, IN, USA, and the M.S.E.E degree from the University of California at Berkeley, Berkeley, CA, USA.

He was the DFT Manager with the Freescale's Automotive Microcontroller and Processors Group, Freescale Semiconductor Inc., Austin, TX, USA, and the TI's ASIC Division, Texas Instruments Inc., Dallas, TX, USA. He has also held various positions with Agilent Technologies, Inc., Santa Clara, CA, USA, HP, Palo Alto, CA, USA, Actel, Aliso Viejo, CA, USA, and LSI Logic, Santa Clara. He is currently the DFT Manager of the NXP's Automotive Microcontroller and Processors Research and Development Group, NXP Semiconductors, Austin, where he is involved in the zero defect test and methodology.

Mark Tehranipoor (S'02–M'04–SM'07) served as the Founding Director of the CHASE Center and the CSI Center, University of Connecticut, Storrs, CT, USA. He is currently the Intel Charles E. Young Professor in cybersecurity with the University of Florida, Gainesville, FL, USA. He has authored over 350 journal articles and refereed conference papers and has given over 160 invited talks and keynote addresses, since 2006 He has authored eight books and 22 book chapters. He has three patents. His current research interests include hardware security and trust, supply chain security, and VLSI design, test, and reliability.

Dr. Tehranipoor is the Golden Core Member of the IEEE Computer Society and a member of the ACM and ACM SIGDA. He is also a member of the Connecticut Academy of Science and Engineering. He was a recipient of 12 best paper awards and nominations. He received the 2008 IEEE Computer Society (CS) Meritorious Service Award, the 2012 IEEE CS Outstanding Contribution, the 2009 NSF CAREER Award, and the 2014 MURI Award. He serves on the Program Committee of over a dozen of leading conferences and workshops. He served as the Program Chair of the 2007 IEEE Defect-Based Testing workshop, the Program Chair of the 2008 IEEE Defect and Data Driven Testing (D3T) workshop, a Co-program Chair of the 2008 International Symposium on Defect and Fault Tolerance in VLSI Systems(DFTS), the General Chair of D3T-2009 and DFTS-2009, and the Vice-General Chair of NATW-2011. He co-founded the IEEE International Symposium on Hardware-Oriented Security and Trust and served as the HOST-2008 and HOST-2009 General Chair. He is currently serving as an Associate Editor of the *Journal of Electronic Testing: Theory and Applications*, the *Journal of Low Power Electronics and Applications*, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS and the *ACM Transactions on Design Automation of Electronic Systems*. He has served as an IEEE Distinguished Speaker and an ACM Distinguished Speaker, from 2010 to 2013.