Test Point Insertion with Control Point by Greater Use of Existing Functional Flip-Flops

Joon-Sung Yang and Nur A. Touba

This paper presents a novel test point insertion (TPI) method for a pseudo-random built-in self-test (BIST) to reduce the area overhead. Recently, a new TPI method for BISTs was proposed that tries to use functional flip-flops to drive control test points instead of adding extra dedicated flip-flops for driving control points. The replacement rule used in a previous work has limitations preventing some dedicated flip-flops from being replaced by functional flip-flops. This paper proposes a logic cone analysis–based TPI approach to overcome the limitations. Logic cone analysis is performed to find candidate functional flop-flops for replacing dedicated flip-flops. Experimental results indicate that the proposed method reduces the test point area overhead significantly with minimal loss of testability by replacing the dedicated flip-flops.

Keywords: Test point insertion, BIST, control point, logic cone analysis, functional flip-flop.

Manuscript received Nov. 21, 2013; revised Jan. 29, 2014; accepted Mar. 5, 2014.

This paper was supported by Samsung Research Fund, Sungkyunkwan University, 2013.

Joon-Sung Yang (js.yang@skku.edu) is with the Department of Semiconductor Systems Engineering, Sungkyunkwan University, Suwon, Rep. of Korea.

Nur A. Touba (touba@ece.utexas.edu) is with the Department of Electrical and Computer Engineering, University of Texas at Austin, TX, USA.

I. Introduction

Testers used in the testing of chips have limitations in terms of I/O channels, bandwidth, speed, and so on; thus, their continual use in conventional external testing–based approaches is proving to be an impediment to attempts to reduce test costs. A built-in self-test (BIST) helps to reduce the dependency on external testers by reducing test time, tester investment cost, test data bandwidth, and test data storage requirements [1]–[3].

With a BIST, circuits that generate test patterns and analyze the output responses of the logic are embedded on-chip. The test pattern generator automatically generates the patterns for application to the inputs of the circuit under test (CUT). The output response analyzer compacts the output response of the CUT into a signature. This provides a number of important advantages, including the ability to apply a large number of test patterns in a short period of time, minimal tester storage requirements, at-speed testing, application of tests out in the field over the lifetime of the part, and a reusable test solution for embedded cores. In particular, a BIST is crucial for applications in areas such as aerospace, defense, automotives, computers, and so on, as well as for the reliability of an entire system.

The most economical logic BIST techniques are based on pseudo-random pattern testing. An on-chip input pattern generator, with compact structure and constructed from a linear-feedback shift register (LFSR), is most commonly used to generate pseudo-random patterns. In addition, an on-chip output response analyzer compacts the output responses into a signature, and this allows for significant compaction of test data. Pseudo-random pattern testing can also achieve high coverage of non-modeled faults that are not explicitly targeted during deterministic test generation. However, a major challenge is the presence of random-pattern-resistant (RPR) faults that have low detection probabilities as these may limit the fault coverage that can be achieved with such pseudo-random pattern testing. In light of this, there have been many research efforts to overcome such limited fault coverage. These efforts, to enhance the fault coverage, can be broadly grouped into two approaches. One approach has been to modify the pattern generator to generate patterns that detect hard faults. Various methods have been proposed, such as weighted pattern generation [2], [4]–[8], pattern mapping [9]–[11], bit-fixing [12], bit-flipping [13], and LFSR reseeding [14]–[19].

The other approach has been to make the CUT random testable by inserting test points [20], which in turn, enhances the detection probability of RPR faults. Test point insertion (TPI) involves adding control and observation points to the CUT. Observation points make a node observable by adding an extra flip-flop and sampling it in a scan cell. Control points involve ANDing or ORing a node with an activation signal, where the activation signal is driven by a dedicated flip-flop (this flip-flop is only used to drive an internal node only for test purposes and is referred to as a dedicated flip-flop or a test dedicated flip-flop) that receives pseudo-random values during a BIST and is set to a non-controlling value during normal operation. Additional hardware is needed to form the test points, which add area and performance overhead to a design. Since optimal test point placement is NP-complete [16], a number of TPI methods have been proposed using fault simulation [21]–[22] and testability measures [18]. TPI for minimizing performance overhead [23]–[24] and TPI for minimizing area overhead [25]–[28] are two general strategies for TPI methods.

A new TPI method was proposed in [29]. It replaces test dedicated flip-flops for driving control points by existing functional flip-flops via conservative replacement rules. It was shown that a significant test point area reduction can be achieved. However, the method used in [29] limits the candidate search space and leaves some test dedicated flip-flops as non-replaceable; thus, it may limit the area overhead reduction that is achieved.

In this paper, we propose a novel TPI method that replaces dedicated flip-flops for control points with functional flip-flops to further reduce the area overhead. Reference [30] introduces a way to replace non-replaceable flip-flops by relaxing the replacement rules in [29]. This paper proposes a different TPI method. Unlike in [30], the proposed method replaces dedicated flip-flops via logic cone analysis without relaxing any replacement rules. It is able to replace more test dedicated flip-flops and achieves significant area reduction. Preliminary results were presented in [31]. This paper shows an in-depth analysis with benchmark circuits, including industrial circuits. A key feature of the proposed approach is the greater effort to reduce the test point area overhead by removing the dedicated flip-flops used for driving the control points.

II. Functional Flip-Flop Driving TPI Method

This section gives an overview of the TPI method proposed in [29]. Usually, when test points are inserted, dedicated flip-flops for test purposes are added to drive control points and capture observation points to increase fault coverage. Extra dedicated flip-flops for control and observation points add to

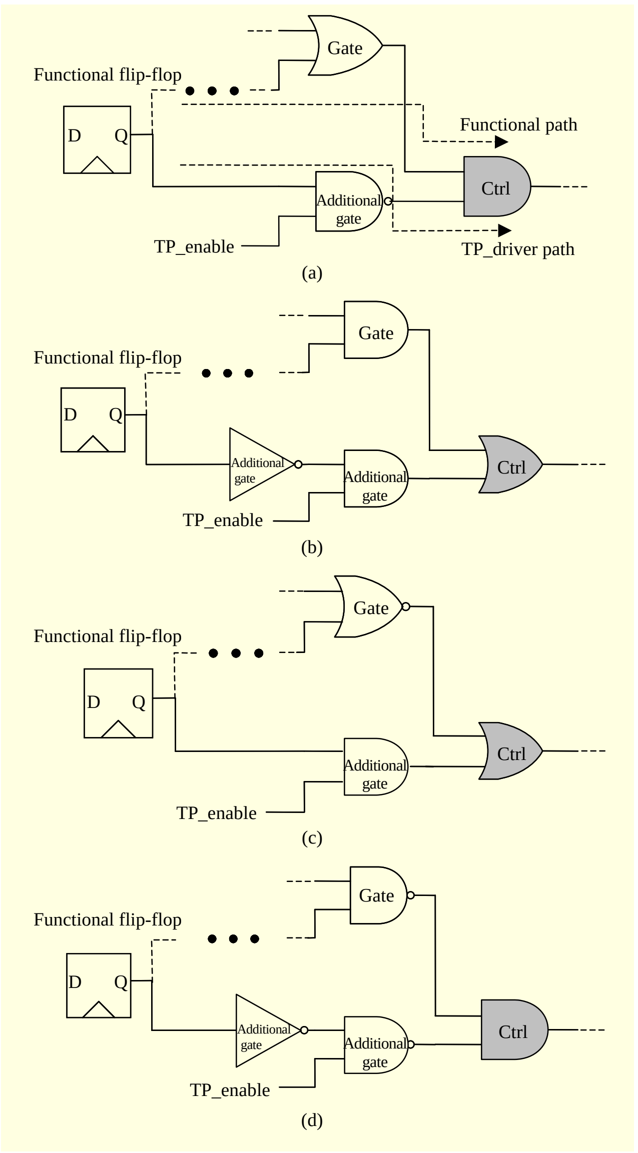

Fig. 1. Four new control point structures driven by functional flip-flops: (a) non-inverting functional path with AND Ctrl (type 1); (b) non-inverting functional path with OR Ctrl (type 2); (c) inverting functional path with OR Ctrl (type 3); and (d) inverting functional path with AND Ctrl (type 4).

the area overhead. Reference [29] proposed a TPI method that replaces the dedicated flip-flops by existing functional flip-flops for control points. This helps to minimize the area overhead by reducing the number of dedicated flip-flops for driving control points.

Since the TPI method in [29] replaces the dedicated flip-flops for control points, during the test point decision stage, functional flip-flops are identified that are suitable to drive the control point. The following guidelines need to be considered for a new TPI approach: no new timing constraints are to be imposed upon the system and there should be no loss to the testability of the system. Based on the above guidelines, four new control point structures that do not require test dedicated flip-flops are proposed (see Fig. 1). As can be seen from types 1–4, there are paths from a functional flip-flop to the control point (Ctrl). One is an existing functional path from a functional flip-flop to a control point (*Functional Path*), and the other is the newly introduced path, ANDed/NANDed, with the *TP_Enable* signal (*TP_Driver Path*). Because the *Functional Path* and *TP_Driver Path* converge, this inherently introduces re-convergent paths sourced by the functional flip-flop. Hence, there is a potential to introduce redundant faults. To avoid redundancies, the opposite path inversion parity is considered. Path inversion parity means the number of inversions by inverters, nand, or nor gates. Opposite path inversion parity along the paths from a functional flip-flop to a control point needs to be maintained. Having opposite inversion parity along these paths makes a path testable by appropriately applying either "0" or "1" to the gates on these paths. This is the first replacement rule used in [29]. Reference [30] relaxed this rule to replace more dedicated flip-flops. Section III-2-A gives a detailed explanation and an example. Test points are only activated in test mode, and this should not change any functional operations during a system's operation. To hold this transparency property, test points are deactivated while the system operates by setting their activation signals to a non-controlling value so that the functional logic value can go through the control gate. A global signal, "*TP_Enable*," is introduced to enable or disable test points for different modes of the system. The *TP_Enable* signal can also help to change the *TP_Driver Path* value if a functional flip-flop has a much skewed signal probability.

As a second replacement rule, [29] and [30] check the illegal re-convergence. Illegal re-convergence is defined as a path re-convergence blocking the fault propagation. Re-convergence from the functional flip-flop needs to be checked to avoid a situation where the propagation of hard-to-test faults is blocked. If the functional flip-flop used to drive the test point drives some gate in a fanout of a test point, then this may prevent hard-to-test faults from being detected.

The TPI in [29] is performed based on the following two replacement rules: opposite path inversion and illegal re-convergence check. This identifies candidate functional flip-flops for driving control points. To preserve the testability of the system, only functional flip-flops having the same inversion polarity, even or odd, are chosen as candidates. For example, if there are multiple paths with different inversion polarity from a functional flip-flop to a control point, then the functional flip-flop is discarded from the candidate list for dedicated flip-flop replacement. This may leave many control points not replaced by functional flip-flops. One limitation in [29] is that, depending on the design, it may not be possible for some control points to find flip-flops that have *Functional Paths* to the control point that are either of all even or all odd inversion parity. A second limitation is that dedicated flip-flops cannot be replaced when no functional flip-flops satisfy the re-convergence check to avoid testability loss. A further limitation could be the single functional flip-flop in the control point fan-in cone. There may be cases when only one functional flip-flop is found as a candidate. This occurs when a test point has a single functional flip-flop in its fan-in. This happens when the controllability to a certain value ("0" or "1") needs to be higher than 0.5. For example, AND or OR trees are more likely to have a skewed controllability on the functional path, either "0" or "1"; thus, they may have a single functional flip-flop to have a skewed value.

III. Enhanced TPI

This section describes the enhanced TPI algorithm employed to overcome the limitations addressed in the previous section — no inversion parity path found and illegal re-convergence path found. The proposed method uses a logic cone analysis to perform a TPI, which is a different approach to those taken in the TPIs of [29] and [30]. It always performs better TPIs than the conventional methods because the proposed method is not restricted by the replacement rules. In this paper, the differences between the conventional methods ([29] and [30]) and the proposed method are highlighted. To efficiently explain the proposed method and highlight the difference, we will show an example describing how the proposed method can replace more dedicated flip-flops — these being a proportion of the ones that are not able to be replaced by the methods of [29] and [30].

1. Enhanced TPI Flow

This section describes the enhanced TPI flow. Figure 2 shows a design synthesis flow that incorporates scan, BIST, and TPI. During the TPI stage, functional flip-flop candidates for each control point are identified. The proposed method

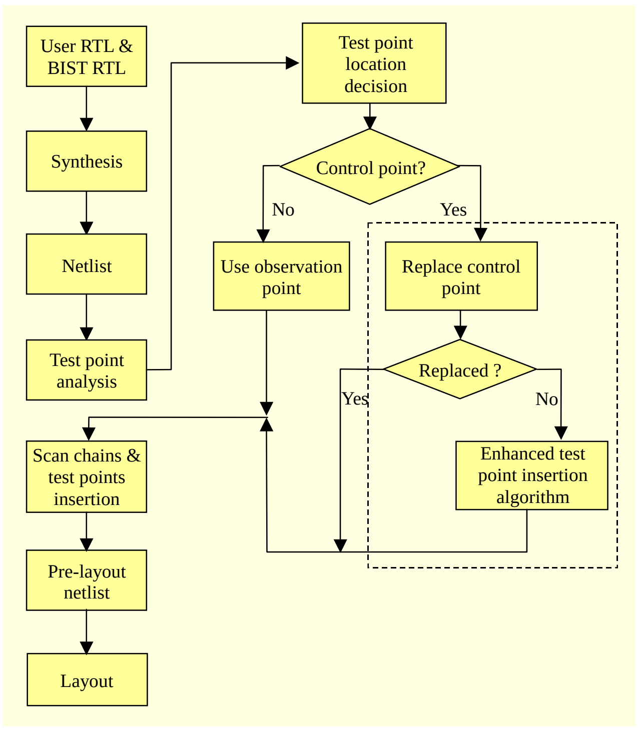

Fig. 2. Post-processing step to enhance TPI flow.

performs a post-processing step, whereby it identifies the control points that have not yet been replaced by functional flip-flop candidates. The dashed line in Fig. 2 indicates the post-processing flow that runs the enhanced TPI flow. The post-processing performs additional dedicated flip-flop replacement for those that are not replaced by TPI in [29] and [30]. Note that the proposed method is a different method from those of [29] and [30]; however, this paper explains the flow as if the proposed method works in addition to the methods in [29] and [30]. This is only to illustrate how the proposed method replaces more dedicated flip-flops by replacing a proportion of those that are not able to be replaced by conventional approaches. But, it should be noted that this is only for illustrative purposes, and all test dedicated flip-flops are replaced by the proposed flow.

2. Enhanced TPI Details

A. TPI in [29]

Figure 3 shows an example of logic with a conventional TPI that uses a dedicated flip-flop. This circuit has flip-flops (denoted A to K) and combinational elements (dented G1 to G11 and Ctrl). It has one control point highlighted in gray color (Ctrl) and a dedicated flip-flop E that drives the control point in test mode.

Based on the analysis [29], the following inversion and logical distance (or logical depth) information can be generated from functional flip-flops to Ctrl. The number of inversions is checked through paths that run from a functional flip-flop to Ctrl, and the logical distance from Ctrl is measured.

Candidate flip-flop	Inversions	Logical distance
A	2 and 3	2, 3 and 4
B	2 and 3	2, 3 and 4
D	2	3

As explained in the previous section, flip-flops A and B violate the inversion parity check. This is because they have even and odd inversion parity through multiple paths to Ctrl; thus, the dedicated flip-flop E for the control point cannot be replaced by A or B. Flip-flop D has an even parity to Ctrl; however, it does not satisfy the re-convergence check because it re-converges with a Ctrl output path. This may block the propagation of hard-to-detect faults. Flip-flop D cannot be used to replace dedicated flip-flop E; hence, there are no functional flip-flops that can replace E in Fig. 3. This may leave many dedicated flip-flops not replaced, since they are limited by the rules in [29]. This limits the area reduction.

B. Identifying Test Point Fan-Out Cone

Unlike the methods shown in [29] and [30], the goal of the enhanced TPI flow described here is to find more candidate functional flip-flops that can be used to replace the dedicated flip-flops to achieve more area reduction without relaxing the replacement rules — inversion parity check and illegal convergence check.

Assume that there are no functional flip-flops available in the current set of candidates for performing the replacement. In this case, the enhanced TPI flow can be used to find possible functional flip-flops that can replace a dedicated flip-flop. This flow finds a functional flip-flop that is not in the fan-in of the test point, which may add additional timing constraints. Therefore, the selection needs to guarantee that there is no performance penalty by the proposed method.

The enhanced TPI flow starts from the test point. The nodes are traversed from the test point and the fan-out cone is generated. Each node is traversed from the control point, and this gives functional flip-flops that are related to the outputs in the test point logic cone. In Fig. 3, via fan-out cone analysis, G6, G8, G10, and G11 are visited, and I and J are found as flip-flops that belong to the fan-out cone of the control point. Therefore, we choose I and J as candidates for replacing dedicated flip-flop E.

Visited gates	Flip-flops in fan-out cone
G6, G8, G10, and G11	I and J

Once the functional flip-flops in the fan-out cone of the control point are identified, the related timing information is obtained because the fan-in cone and fan-out cone analysis find all logic related to the control point. The other flip-flops outside of the fan-out cone of the control point do not satisfy the timing requirements; hence, they will not be considered.

C. Analyzing Fan-In Cones from Candidate Functional Flip-Flops

Once all inputs related to the test point have been found, then the next step is to select a functional flip-flop to replace the dedicated flip-flop. To avoid introducing additional timing constraints, only functional flip-flops appearing in the fan-out cone need to be considered. In the example in Fig. 3, flip-flops I and J are found in the Ctrl fan-out cone by the analysis described in the previous section.

From each output of the logic cone, the logic cone analysis can generate a fan-in cone. A fan-in cone analysis for I and J can now be performed, which will give all flip-flops belonging to I and J that lie in the fan-in cone. Based on the analysis, the following flip-flops are found in the fan-in cone of I and J.

Fan-in cone	Flip-flops belonging to fan-in cone
I	A, B, C, D, E, and F
J	A, B, C, D, E, F, and G

In Section III-2-A, flip-flops A, B, D, and E are initially considered; however, they are found to be unsuitable for replacing a dedicated flip-flop because they do not meet the replacement rules. Hence, we consider other possible candidates to replace the dedicated flip-flops — namely, C, F, and G. These flip-flops do not feature in the test point's fan-in cone, but are instead found to be possible candidates through logic cone analysis.

D. Replacing Dedicated Flip-Flops

Once the fan-in cone analysis finds all inputs related to the test point, a functional flip-flop needs to be selected to replace the dedicated flip-flop. The candidate inputs can be divided into two types. One is an input whose fan-out cone includes all the outputs of the test point's fan-out cone. The other is an input that only partially covers the outputs of the test point's fan-out cone. In Fig. 3, since C, F, and G are candidates, we generate their respective fan-out cones. From C, the logic cone analysis finds the fan-out cone that includes outputs I and J. The fan-out cone of F finds outputs I and J, and J and K are found by the logic cone analysis from G.

Fan-out cone start	Outputs of fan-out cone
C	I and J
F	I and J
G	K and J

The main concern is the additional timing constraints in the enhanced TPI flow. To guarantee that there is no performance penalty incurred, it is necessary to find functional flip-flops that cover all the outputs of the test point's fan-out cone. Inputs that have all of the outputs of the test point's fan-out cone as their outputs can be considered as candidates. In Section III-2-B, I and J are found by a fan-out cone analysis of the control point. As can be seen from the above, the C and F fan-out cones cover I and J. Therefore, G is not acceptable, and C and F can be considered as candidates for replacing the dedicated flip-flop E. It is noted here that random pattern testability is not degraded by the proposed TPI method. For testability, since there may be many connections from the functional flip-flops to other nodes in a circuit, it is necessary to check whether the fault propagation is blocked. Hence, the illegal re-convergence is also checked when the candidate function flip-flops are determined by fan-in and fan-out cone analysis. Re-convergence from the candidate functional flip-flop needs to be checked to avoid the case where it will block the propagation of hard-to-test faults. If the functional flip-flop used to drive the

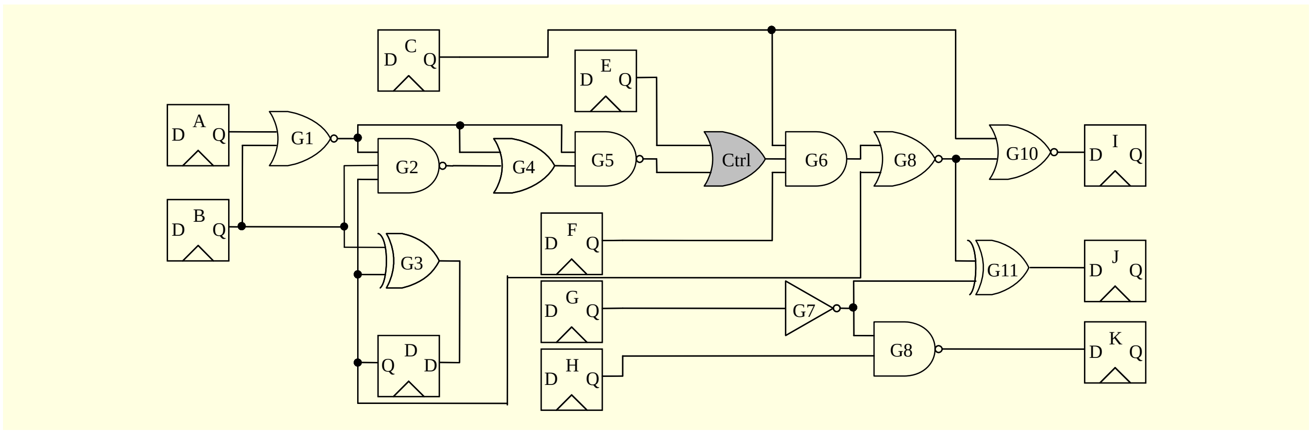

Fig. 3. Example of a circuit by conventional control point (Ctrl) insertion with dedicated flip-flop E.

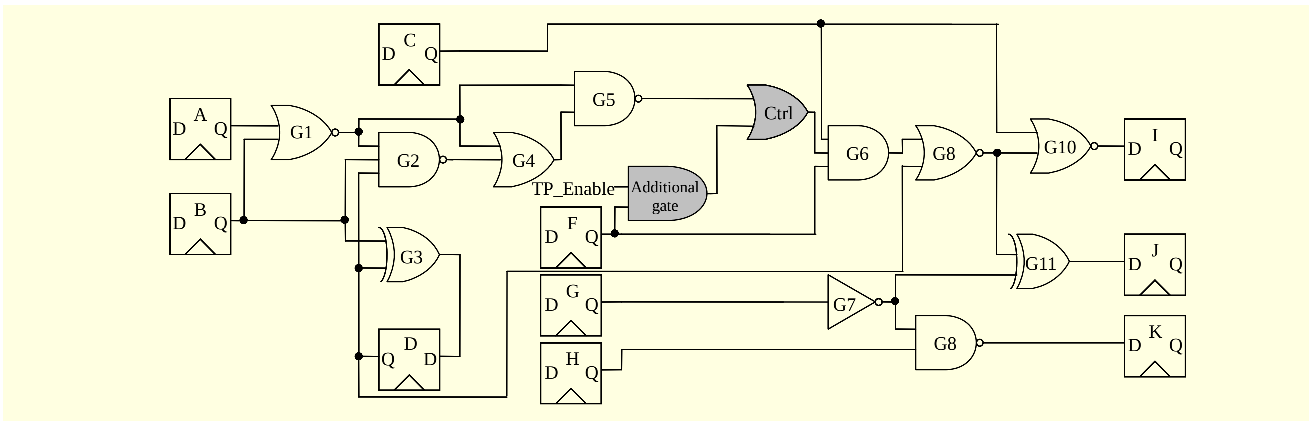

Fig. 4. Example of a circuit by proposed TPI with functional flip-flop F.

test point drives some gate in a fan-out of a test point, then this may prevent hard-to-test faults from being detected. For example, in Fig. 3, C and F are found as candidates to replace a dedicated flip-flop for the control point. For these flip-flops, the illegal re-convergence check needs to be performed for the purposes of testability. Flip-flop C drives G6 and also has a re-convergence path to G10; however, F is only fed G6. Flip-flop C violates an illegal re-convergence rule because it can block the fault propagation. Hence, flip-flop F is chosen to replace the dedicated flip-flop E. If there are many candidates that satisfy the fan-in and fan-out cone analysis as well as the re-convergence check, then the flip-flop driving a gate closest to the control point needs to be chosen. A functional flip-flop that is logically close to the control point is chosen to replace a dedicated flip-flop. Acceptable candidates' logic cones partly share the logic with the test point's fan-out cone. For each acceptable candidate, propagation from the test point is performed until the overlapped gate element with the test point's fan-out logic cone is first found. To minimize the length of the newly created test path from the candidate flip-flop to the control point, the input that has the closest overlapped element is selected for replacing those dedicated flip-flops that have not been replaced after using the methods in [29] and [30].

Once a functional flip-flop is determined for TPI, a new type of control point needs to be chosen. As can be seen in Fig. 1, the control point is driven by the combination of a functional flip-flop and a *TP_Enable* signal. In this manner, the control point is driven with zeros and ones, and the functional flip-flop drives a gate that is placed on the fan-out path from the control point. In Fig. 3, F will be used to replace the dedicated flip-flop; hence, not only does it drive the control point but also it is fed into G6. Flip-flop G6 is placed in the fan-out path from the control point. It can propagate the control point output when the value at F is one — a non-controlling value. Flip-flop F needs to drive the new control point with a non-controlling value to force zeros and ones to the control point.

Figure 4 shows the control point insertion by the proposed method based on the circuit in Fig. 3. The dedicated flip-flop E is replaced by the functional flip-flop F with one additional AND gate. A "type 3" control point structure is used with the functional flip-flop.

Note that although this paper explains how not-replaced dedicated flip-flops can be replaced by the proposed method to show the difference with conventional approaches, in reality, all dedicated flip-flops are replaced by the logic cone analysis described in Section III-2.

IV. Experimental Results

In this section, experimental results are presented with the proposed test insertion method to evaluate the improvements that are obtained through the flow proposed. Four industrial designs (A–D), OR1200 (OpenRisc Processor) [32], a network-on-chip design A (NOC-A) [33], and a NOC design B (NOC-B) [34] are used and test points are inserted. The Mentor Graphics Tessent tool [35] was used to determine the location of test points in each design. Dedicated flip-flops for driving the control points are replaced by the proposed flow to reduce the number of dedicated flip-flops without impacting delay and without increasing the loss of testability.

The proposed method determines the functional flip-flop that can be used to drive the control point. In Table 1, the number of dedicated flip-flops that are replaced by functional flip-flops using the proposed method is shown. The first column gives the design name. The second and third columns show the number of observation points and control points, respectively with a conventional TPI method. Each test point requires a dedicated flip-flop. For example, Design A has 5,222 test points, 1,451 observation points, and 3,771 control points. A conventional TPI method adds 5,222 dedicated flip-flops. Since the TPI method in [30] ([30] is a replacement rule–relaxed version of [29]) shows better results than [29] but still

Table 1. Control point dedicated flip-flop replacement improvement comparison.

Design	Conventional TP insertion		Replacement in [30]		Replacement by proposed method		Replacement improvement ratio over [30]
	Observation point	Control point	Dedicated flip-flop	Functional flip-flop	Dedicated flip-flop	Functional flip-flop	
Design A	1451	3771	573	3198	517	3254	9.8%
Design B	129	371	13	358	8	363	38.5%
Design C	3	24	0	24	0	24	N/A
Design D	70	179	15	164	2	177	86.7%
OR 1200	5	27	5	22	0	27	100%
NOC-A	9	35	5	30	0	35	100%
NOC-B	12	58	21	37	0	58	100%

leaves some dedicated flip-flops not replaced, we try to replace them. The fourth and fifth columns show the number of dedicated flip-flops and functional flip-flops used to drive control points by TPI in [30], respectively. The number in the fourth column gives the number of dedicated flip-flops for control points that are unable to be replaced by the method in [30]. The fifth column shows the number of functional flip-flops used to replace the dedicated flip-flops for control points in the third column. In Design A, 3,198 out of 3,771 dedicated flip-flops for control points are replaced by the method in [30]. For the dedicated flip-flops in the fourth column, the proposed method in this paper tries to replace them by the functional flip-flops, the results of which are shown in the sixth and seventh columns. The proposed method replaces 3,254 out of 3,771 dedicated flip-flops for control in Design A and 363 out of 371 dedicated flip-flops in Design B. Both the proposed method and that of [30] replace all control points with a dedicated flip-flop. Similarly, they both replace all control points with a dedicated flip-flop in Design C. For OR1200 and NOC, designs are manipulated to generate non-replaceable dedicated flip-flops by [30] (all dedicated flip-flops were initially replaceable). The proposed method was able to find the functional flip-flop and replace dedicated flip-flops in both the OR1200 and the NOC designs. The last column shows the replacement ratio. In Design D, OR1200, NOC-A, and NOC-B, the proposed method replaces most of the not-replaced dedicated flip-flops by [30]. The conventional method may work well for some circuits; however, the proposed method always enhances the performance of TPI to a significantly greater degree. For example, a conventional TPI performs very well in Design C. Moreover, regardless of the characteristics of the benchmark circuits, the proposed TPI significantly outperforms conventional methods, as can be seen from the replacement ratios. The proposed method in Design D achieves a very high replacement ratio, while the conventional method does not. This is particularly evident for NOC-B, where the conventional method replaces around 60% of the dedicated flip-flops using replacement rules, whereas the proposed method replaces 100% using logic cone analysis.

Designs A and B show a relatively low replacement ratio. They have a number of AND or OR tree structures, and the control points used in those trees require skewed controllability. In this case, the controllability to a certain value ("0" or "1") needs to be higher than 0.5; hence, dedicated flip-flops cannot be replaced. It should be noted that the low replacement ratio in designs A and B is caused not by the proposed TPI but by their design characteristics. They need dedicated flip-flops to have skewed controllability in testing.

As shown in Table 1, the proposed method further replaces dedicated flip-flops for control points. To understand the test point area reduction, OR1200 and NOC designs are synthesized with 130 nm TSMC technology [36]. In Table 2, synthesized results are shown for combinational, sequential logic, and the summation of them. The second column shows the area of the original design, and the third column describes the area when scan chains with TAP and logic BIST are inserted in the original design. The increase rate is shown in the fourth column. The area for conventional TPI is in the fifth column, and the sixth column shows the relative increase over the original design. The seventh and eighth columns show the synthesized area and the increase over the original design, respectively. As expected, since conventional TPI techniques involve adding dedicated flip-flops, there is a considerable increase in the sequential logic part compared to the proposed method, which replaces them with existing functional flip-flops. In addition, the proposed method introduces extra primitive gates (*Additional Gate* in Fig. 4), which gives a smaller area overhead compared to the conventional method; however, the total area reduction is significant because dedicated flip-flops are removed.

Table 2. Synthesized area results for OR 1200 [30] and NOC-A designs.

Design		Original	Scan insertion	Relative increase	Dedicated flip-flop	Relative increase	Functional flip-flop	Relative increase
OR1200	Combinational	178072	229417	28.83%	229745	0.18%	229951	0.30%
	Sequential	127543	155934	22.26%	157866	1.51%	156236	0.24%
	Total	305615	385351	26.09%	387611	0.74%	386187	0.27%
NOC-A	Combinational	83450	102860	23.26%	103236	0.45%	103549	0.83%
	Sequential	126377	165923	31.29%	168578	2.10%	166466	0.43%
	Total	209827	268783	28.10%	271814	1.44%	270015	0.59%

Table 3. Improvement comparisons for dedicated flip-flop reduction ratio and test point area reduction ratio.

Design	Conventional TP insertion		Replacement in [30]		Replacement by proposed method		Test point area reduction improvement ratio over [30]
	Reduction ratio	Test point area reduction	Reduction ratio	Test point area reduction	Reduction ratio	Test point area reduction	
Design A	N/A	N/A	84.8%	45.9%	86.2%	46.7%	1.7%
Design B	N/A	N/A	96.4%	53.7%	97.8%	54.4%	1.3%
Design C	N/A	N/A	100%	66.7%	100%	66.6%	N/A
Design D	N/A	N/A	91.6%	49.4%	98.8%	53.3%	7.5%
OR 1200	N/A	N/A	81.4%	51.5%	100%	63.2%	18.5%
NOC-A	N/A	N/A	85.7%	51.1%	100%	59.6%	14.3%
NOC-B	N/A	N/A	63.7%	39.6%	100%	62.1%	36.2%

In OR1200 and NOC-A, each of the new control points driven by a functional flip-flop takes approximately 1/4 of the area of the original control points driven with a dedicated flip-flop. Therefore, the extrapolated area for designs A–D and NOC-B can be calculated based on the following equation:

$$\frac{\text{New area}}{\text{Old area}} = \frac{\mathrm{N}_{\text{obs}} + \mathrm{N}_{\text{dedicated}} + k \times \mathrm{N}_{\text{functional}}}{\mathrm{N}_{\text{obs}} + \mathrm{N}_{\text{dedicated}} + \mathrm{N}_{\text{functional}}} = 1 - \text{Area\_reduction},$$

where N_{obs} denotes the total number of flip-flops or observation points in a design, and $\mathrm{N}_{\text{dedicated}}$ and $\mathrm{N}_{\text{functional}}$ indicate the number of dedicated flip-flops and functional flip-flops used for the control point, respectively. Since the factor k is approximately equal to 0.25 for both OR1200 and NOC, we can calculate the area reduction of designs A–D.

Table 3 compares the improvement ratios of dedicated flip-flop reduction and test point area reduction. The reduction ratio is computed as the number of functional flip-flops used to replace dedicated flip-flops divided by the total number of control points. In Design D, [30] replaces 164 out of 179 control point dedicated flip-flops, achieving a 91.6% reduction in the number of such flip-flops. The proposed method replaces 177 dedicated flip-flops, giving a 98.8% reduction in the number of such flip-flops. In terms of test point area reduction, the proposed method achieves 53.3% area reduction compared to 49.4% by [30] in Design D. For NOC-B, the reduction ratio is considerably enhanced from 63.7% to 100%, and the test area reduction is significantly improved from 39.6% to 62.1%. The area reduction is calculated by the proposed equation shown above. As Table 1 shows, the proposed method replaces a greater number of dedicated flip-flops, and this further reduces the area overhead in TPI. Table 3 shows the further area reduction results.

Fault coverage results are shown in Table 4, with and without TPI (a conventional TPI using dedicated flip-flops, a method in [30], and the proposed method). The fault coverage is measured when 100,000 random patterns are applied to designs A–D and 16,000 random patterns are applied to the OR1200 and NOC designs. The first column shows the designs. The second column shows the fault coverage when no test points are inserted. Testability results with a conventional TPI method using dedicated flip-flops, with a TPI method in [30], and with a proposed method are shown in the third, fourth, and fifth columns, respectively. The coverage results show that the proposed method achieves almost the same fault coverage as the conventional TPI method does. In some cases, the

Table 4. Fault coverage comparisons.

Design	No test points	Conventional TPI	Method in [30]	Proposed method
Design A	79.79%	96.82%	96.47%	96.45%
Design B	87.16%	98.25%	97.86%	97.85%
Design C	88.19%	93.35%	93.30%	93.30%
Design D	95.05%	98.71%	98.63%	98.25%
OR 1200	93.18%	98.96%	99.76%	99.76%
NOC-A	97.78%	99.41%	99.81%	99.71%
NOC-B	96.84%	99.07%	98.92%	98.92%

coverage is not as good; for example, Design D. This may be because of the noise related to vectors. Noise is caused by the fact that not exactly the same vectors are applied to a chip, and the slight differences in coverage are similar to the effect of changing the polynomial, seed, or chain ordering. Variations in fault coverage, of the proposed method, tend to decrease with an increasing number of vectors. This might result in a lower test coverage in the benchmark circuits. If the minimal loss of fault coverage of the proposed method in comparison with [30] is considered an issue, then this loss can be compensated by a combination of three options — applying more random patterns, calculating more top-up patterns, or adding more test points. However, note that since the fault simulator does not consider the internal faults of flip-flops, it appears that the proposed method has more faults than the standard implementation, even though, in reality, there are actually fewer.

V. Conclusion

In this paper, a new TPI technique performs logic cone analysis and finds more functional flip-flop candidates without having to depend upon a rule-based method, as in the case of [30]. This significantly increases the chances of being able to replace dedicated flip-flops that are driving control points with functional flip-flops. The experimental results indicate that the methodology proposed in this paper can significantly reduce the number of dedicated flip-flops by replacing them with functional flip-flops. The proposed method reduces the area overhead and helps to reduce the number of possible faults added by the additional logic. By considering the testability issues, significant area savings are achieved while preserving the random pattern testability of the circuit and without introducing new timing constraints that would complicate timing closure. Overall, the test point area was reduced by about more than half, while the fault coverage loss during the random pattern phase was kept very close to other TPI methods. Several options, such as applying more random patterns, calculating more top-up patterns, or adding more test points, can be applied to compensate for a slightly lower fault coverage.

The proposed method can be easily used on top of existing conventional TPI algorithms by replacing the added dedicated flip-flops. The proposed method only involves the static tracing of the fan-ins and fan-outs of gates, which are related to control points. Very efficient flows are available for performing these tasks, which are less complex than the algorithms used for test point selection itself. It should also be noted that the proposed new TPI method gives the flexibility of adding more test points to achieve even higher fault coverage or to reduce test time.

Future work includes a new test point replacement flow considering the physical layout. Layout-aware control point replacement flow might provide a greater number of functional flip-flop candidates from outside of the logic cone. This would help to further increase the number of control points, thereby reducing the area overhead.

References

[1] C. Barnhart et al., “OPMISR: The Foundation for Compressed ATPG Vectors,” *Proc. Int. Test Conf.*, Baltimore, MD, USA, Oct. 30, 2001, pp. 748–757.

[2] A. Jas, C.V. Krishna, and N.A. Touba, “Hybrid BIST Based on Weighted Pseudo-Random Testing: A New Test Resource Partitioning,” *IEEE Proc. VLSI Test Symp.*, Marina Del Rey, CA, USA, Apr. 29 – May 3, 2001, pp. 2–8.

[3] H. Ren et al., “Low-Cost TPI without Using Extra Registers for High Performance Design,” *Proc. Int. Test Conf.*, Austin, TX, USA, Nov. 1–6, 2009, pp. 1–8.

[4] M. Bershteyn, “Calculation of Multiple Sets of Weights for Weighted Random Testing,” *Proc. Int. Test Conf.*, Baltimore, MD, USA, Oct. 17–21, 1993, pp. 1031–1040.

[5] L. Lai et al., “Hardware Efficient LBIST with Complementary Weights,” *IEEE Proc. Int. Conf. Comput. Des.*, San Jose, CA, USA, Oct. 2–5, 2005, pp. 479–481.

[6] A. Paschalis, I. Voyiatzis, and D. Gizopoulos, “Accumulator Based 3-Weight Pattern Generation,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 20, no. 2, Feb. 2012, pp. 357–361.

[7] 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. Circuites Syst.*, vol. 12, no. 7, July 1993, pp. 1050–1058.

[8] T. Enmin, S. Song, and Y. Zham “Weighted Test Generator in Built-in Self-Test Design Based on Genetic Algorithm and Cellular Automata,” *Int. Conf. Electron. Meas. Instrum.*, Chengdu, China, Aug. 16–19, 2011, pp. 134–138.

[9] V. Gherman et al., “Efficient Pattern Mapping for Deterministic

Logic BIST," *Proc. Int. Test Conf.*, Charlotte, NC, USA, Oct. 26 –28, 2004, pp. 48–56.

[10] N.A. Touba and E.J. McCluskey, "Synthesis of Mapping Logic for Generating Transformed Pseudo-Random Patterns for BIST," *Proc. Int. Test Conf.*, Washington, DC, USA, Oct. 21–25, 1995, pp. 674–682.

[11] H.-J. Wunderlich et al., "Efficient Pattern Mapping for Deterministic Logic BIST," *IEEE Proc. VLSI Test Symp.*, Charlotte, NC, USA, Oct. 26–28, 2004, pp. 48–56.

[12] W. Li et al., "A Scan BIST Generation Method Using a Markov Source and Partial Bit-Fixing," *Proc. Des. Autom. Conf.*, Anaheim, CA, USA, June 2–6, 2003, pp. 554–559.

[13] H.-J. Wunderlich and G. Kiefer, "Bit-Flipping BIST," *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, San Jose, CA, USA, Nov. 10–14, 1996, pp. 337–343.

[14] S. Hellebrand et al., "Built-in Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers," *IEEE Trans. Comput.*, vol. 44, no. 2, Feb. 1995, pp. 223–233.

[15] H.-S. Kim and S. Kang, "Increasing Encoding Efficiency of LFSR Reseeding-Based Test Compression," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 25, no. 5, May 2006, pp. 913–917.

[16] X. Kavousianos et al., "Defect-Oriented LFSR Reseeding to Target Unmodeled Defects Using Stuck-At Test Sets," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 19, no. 12, Dec. 2011, pp. 2330–2335.

[17] J. Lee and N.A. Touba, "LFSR-Reseeding Scheme Achieving Low-Power Dissipation During Test," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 26, no. 2, Feb. 2007, pp. 396–401.

[18] J. Rajski et al., "Embedded Deterministic Test for Low Cost Manufacturing Test," *Proc. Int. Test Conf.*, Washington, DC, USA, 2002, pp. 301–310.

[19] Z. Wang et al., "Deviation-Based LFSR Reseeding for Test-Data Compression," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 28, no. 2, Feb. 2009, pp. 259–271.

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

[21] N.Z. Basturkmen, S.M. Reddy, and J. Rajski, "Improved Algorithms for Constructive Multi-phase TPI for Scan Based BIST," *Proc. Asia South Pacific Des. Autom. Conf.*, Bangalore, India, Jan. 7–11, 2002, pp. 604–611.

[22] Semiconductor Industry Association. *The Int. Technol. Roadmap for Semicond.*, 2007.

[23] K.-T. Cheng and C.-J. Lin, "Timing-Driven Test Point Insertion for Full-Scan and Partial-Scan BIST," *Proc. Int. Test Conf.*, Washington, DC, USA, Oct. 21–25, 1995, pp. 506–514.

[24] H.-C. Tsai et al., "Efficient Test Point Selection for Scan-Based BIST," *IEEE Trans. VLSI Syst.*, vol. 6, no. 4, Dec. 1998, pp. 667–676.

[25] R. Haoxing et al., "Low Cost TPI without Using Extra Registers for High Performance Design," *Proc. Int. Test Conf.*, Austin, TX, USA, Nov. 1–6, 2009, pp. 1–8.

[26] M. Nakao et al., "Low Overhead Test Point Insertion for Scan-Based BIST," *Proc. Int. Test Conf.*, Atlantic City, NJ, USA, Sept. 28–30, 1999, pp. 384–357.

[27] R. Sethuram et al., "Zero-Cost Test Point Insertion Technique to Reduce Test Set Size and Test Generation Time for Structured Asics," *Proc. Asian Test Symp.*, Fukuoka, Japan, Nov. 20–23, 2006, pp. 339–348.

[28] N. Tamarapalli and J. Rajski "Constructive Multi-phase Test Point Insertion for Scan-Based BIST," *Proc. Int. Test Conf.*, Washington, DC, USA, Oct. 20–25, 1996, pp. 649–658.

[29] J.-S. Yang, B. Nadeau-Dostie, and N.A. Touba, "Test Point Insertion Using Functional Flip-Flops to Drive Control Points," *Proc. Int. Test Conf.*, Austin, TX, USA, Nov. 1–6, 2009, pp. 1–10.

[30] 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. 10, no. 10, Oct. 2012, pp. 1473–1483.

[31] J.-S. 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," *Proc. IEEE Symp. Defect Fault Tolerance VLSI Syst.*, Chicago, IL, USA, Oct. 7–9, 2009, pp. 20–28.

[32] OPENCORES. Accessed Oct. 21, 2008. http://www.opencores.org

[33] J.-S. Yang and N.A. Touba, "Automated Selection of Signals to Observe for Efficient Silicon Debug," *Proc. VLSI Test Symp.*, Santa Cruz, CA, USA, May 3–7, 2009, pp. 79–84.

[34] W. Jang and D.Z. Pan, "Application-Aware NOC Design for Efficient SDRAM Access," *IEEE Trans. Comput.-Aided Des. Integr Circuits Syst.*, vol. 30, no. 10, Oct. 2011, pp. 1521–1533.

[35] Mentor Graphics Tessent Tools Reference Manual, 2010.

[36] Synopsys Design Compiler, A-2007.12-SP4.

Joon-Sung Yang received his BS degree from Yonsei University, Suwon, Rep. Korea, in 2003 and his MS and PhD degrees from the University of Texas at Austin, Austin, TX, USA, in 2007 and 2009, respectively, all in electrical and computer engineering. Upon graduating, he worked for the Intel Corporation, Austin, TX, USA, for four years. He is currently an assistant professor at Sungkyunkwan University, Seoul, Rep. of Korea. His research interests include VLSI testing; silicon debug and nanometer scale testing; and design methodologies. He was a recipient of the Korea Science and Engineering Foundation Scholarship, in 2005. He received the Best Paper Award at the 2008 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems and was nominated for the Best Paper Award at the 2013 IEEE VLSI Test Symposium.

Nur A. Touba received his BS degree from the University of Minnesota, Minneapolis, USA, in 1990 and his MS and PhD degrees from Stanford University, Stanford, CA, USA, in 1991 and 1996, respectively, all in electrical engineering. He is currently a professor with the Department of Electrical and Computer Engineering, University of Texas, Austin, TX, USA. He was the recipient of the National Science Foundation Early Faulty Career Award in 1997, the Best Paper Award at the 2001 VLSI Test Symposium, and the Best Paper Award at the 2008 Defect and Fault Tolerance Symposium. He was elevated to an IEEE Fellow in 2009. He served as program chair for the 2008 International Test Conference and general chair for the 2007 Defect and Fault Tolerance Symposium. He currently serves on the program committees for the Design Automation and Test in Europe Conference; International On-Line Test Symposium; European Test Symposium; Asian Test Symposium; and Defect and Fault Tolerance Symposium.