Embedded Deterministic Test Points

Cesar Acero, Derek Feltham, Yingdi Liu, Elham Moghaddam, *Member, IEEE*, Nilanjan Mukherjee, *Senior Member, IEEE*, Marek Patyra, *Member, IEEE*, Janusz Rajski, *Fellow, IEEE*, Sudhakar M. Reddy, *Life Fellow, IEEE*, Jerzy Tyszer, *Fellow, IEEE*, and Justyna Zawada, *Member, IEEE*

***Abstract*— There is mounting evidence that automatic test pattern generation tools capable of producing tests with high coverage of defects occurring in the large semiconductor nanometer designs unprecedentedly inflate test sets and test application times. A design-for-test technique presented in this paper aims at reducing deterministic pattern counts and test data volume through the insertion of conflict-aware test points. This methodology identifies and resolves conflicts across internal signals allowing test generation to increase the number of faults targeted by a single pattern. This is complemented by a method to minimize silicon area needed to implement conflict-aware test points. The proposed approach takes advantage of the conflict analysis and reuses functional flip-flops as drivers of control points. Experimental results on industrial designs with on-chip test compression demonstrate that the proposed test points are effective in achieving, on average, an additional factor of 2x–4x compression for stuck-at and transition patterns over the best up-to-date results provided by the embedded deterministic test (EDT)-based regular compression.**

***Index Terms*— Design for testability, embedded test, scan-based testing, test application time, test data compression, test points.**

I. Introduction

NOTWITHSTANDING the great success of test data compression [14], [26], the magnitude of test sets produced by the contemporary automatic test pattern generation (ATPG) tools continues to grow at a pace clearly surpassing Moore's law. The exploding number of test vectors is commonly attributed (although not limited) to the following reasons.

1) New test generation techniques use pattern-intensive transistor-level ATPG aimed at reducing test escapes; the same techniques for at-speed patterns consist of a large number of steps handling many clock domains.
2) Circuits comprise the logic of large combinational depths and feature staggeringly complex clocking schemes.
3) Tail pattern counts are excessive—although these tests contain very few specified bits, mutual conflicts prevent their compression-aware merging.
4) Automatically generated RTL poses nontrivial test challenges due to complex control logic.

Semiconductor scaling, extremely small feature sizes, the multiple patterning lithography, 3-D structures, and the mass-produced FinFET devices are just a few domains in which advances are explicitly pushing for rapid and deep reshaping of the traditional test solutions, especially that the gate-level abstraction and some fault models do not suffice to ensure high-quality and low-defects-per-million requirements for state-of-the-art designs. The next-generation solutions are to target novel timing- and layout-related fault models and patterns, such as *n*-detect, embedded-multidetect, or cell-aware [11], [12] based on postlayout transistor-level netlists. As a result, inflated test sets and lengthy test application times become efficiency-limiting and cost-increasing factors in the testing of embedded systems, system-on-chip designs, or automotive electronics.

In this paper, we present a design-for-test (DFT) technology that aims at reducing ATPG test pattern counts and test data volume through the insertion of conflict-aware test points. A key feature of the proposed scheme is its ability to identify and resolve conflicts between ATPG-assigned design's internal signals. This ability allows the new approach to increase the number of faults targeted by a single pattern, and to overcome the growing test set size problem by reducing both the number of deterministic test patterns and test data volume, leading eventually to shorter ATPG and test application times. The proposed scheme deploys a method to reuse functional flip-flops as test point drivers. This allows one to minimize the silicon area needed to instantiate control points (CPs).

Typically, test point insertion (TPI) techniques try to improve the fault detection likelihood while minimizing the necessary hardware [20], [23], [29]. They select circuit's internal nets to subsequently add CPs or observation points (OPs) in order to activate faults or observe them, respectively. Identification of test points in circuits with reconvergent fan-outs is a complex problem [5], and hence, numerous empirical guidelines and approximate techniques have been proposed [9], [19], [21], [24], [27], [30] to find suitable CP and OP locations while improving the overall circuit testability. Depending on how a test point is driven or observed, it may require a few extra gates and wires routed to or from additional (dedicated) flip-flops to be included in scan chains. As it introduces area and performance penalty, the number of test points is usually limited. Furthermore, a pragmatic TPI scheme

Manuscript received January 25, 2017; revised April 26, 2017 and June 7, 2017; accepted June 16, 2017. Date of publication July 3, 2017; date of current version September 25, 2017. The work of J. Tyszer and J. Zawada was supported by the Polish Ministry of Science and Higher Education under Grant DS-811/17. *(Corresponding author: Jerzy Tyszer.)*

C. Acero, D. Feltham, and M. Patyra are with the Intel Corporation, Hillsboro, OR 97124 USA (e-mail: cesar.f.acero@intel.com; derek.feltham@intel.com; marek.patyra@intel.com).

Y. Liu and S. M. Reddy are with the Department of Electrical and Computer Engineering, The University of Iowa, Iowa City, IA 52242 USA (e-mail: yingdi_liu@mentor.com; reddy@engineering.uiowa.edu).

E. Moghaddam, N. Mukherjee, and J. Rajski are with Mentor Graphics, Wilsonville, OR 97070 USA (e-mail: elham_moghaddam@mentor.com; nilanjan_mukherjee@mentor.com; janusz_rajski@mentor.com).

J. Tyszer and J. Zawada are with the Faculty of Electronics and Telecommunications, Poznań University of Technology, 60-965 Poznań, Poland (e-mail: jerzy.tyszer@put.poznan.pl; justyna.j.zawada@doctorate.put.poznan.pl).

Digital Object Identifier 10.1109/TVLSI.2017.2717844

must computationally be inexpensive despite the structural complexity of large designs.

The first systematic TPI [4] used simulation to yield fault propagation profiles and internal signal correlations. Next, test points were used to break the signal correlations down. Similarly, Iyengar and Brand [13] employ the fault simulation to identify gates blocking faults and insert test points to regain the propagation of fault effects. A divide and conquer approach of [24] partitions the entire test into multiple phases. Within each phase, a group of test points is activated to maximize the fault coverage calculated over the set of still-undetected faults. A probabilistic fault simulation computing the impact of a new CP in the presence of the CPs already selected is used as a vehicle to select test points.

To avoid time-consuming simulations, other methods use controllability and observability measures to identify the hard-to-control and hard-to-observe sectors of a circuit, where test points are subsequently inserted. In particular, the schemes of [7] and [17] use COP estimates [3] to extract testability data. Hybrid testability measures [25] that work with SCOAP metrics [10], cost functions [9], a gradient-based method [22], or signal correlation [6] are used as well to determine the best test point sites.

Traditional test points can reportedly reduce the number of deterministic test patterns for otherwise similar fault coverage numbers [8]. However, their performance is quite unpredictable as they may not affect test pattern counts at all with their average reduction being anywhere between 0% and 35% [15]. Contrary to earlier solutions, this work tackles the identification and usage of a new class of test points deployed to resolve conflicts between internal signals a test generation process is handling. Subsequently, these test points become necessary prerequisites to producing compact test sets with reduced pattern counts compared to baseline designs without any test points. Aside from the primary objective of reducing ATPG pattern counts, the conflict-aware test points also deliver capabilities that can be deployed to reduce test-related silicon overhead, in particular by reusing functional flip-flops to drive the majority of inserted test points.

The remainder of this paper is organized as follows. Sections II and III introduce all necessary concepts related to internal conflicts and fault blocking, respectively. Section IV demonstrates how to compute appraisal metrics that allow assessing the degree of actual conflicts. A TPI flow is presented in Section V. The method allowing the reuse of functional flip-flops as CP drivers is presented in Section VI. Sections VII and VIII discuss the verification of candidate flip-flops to prevent test coverage loss, and then devise a complete CP-drivers-search algorithm. The experimental results obtained for several industrial designs that feature conflict-aware test points are presented in Section IX. Conclusions are drawn in Section X. Preliminary versions of this paper were presented at the IEEE International Test Conference [1], [16], and in [2].

II. Internal Conflicts

Consider a simple example having no resemblance to any concrete fault model. In order for a test pattern to detect many

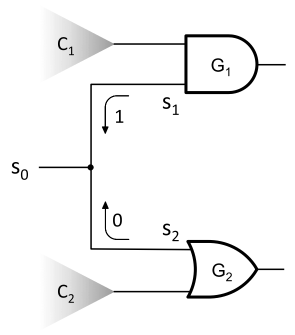


Fig. 1. Internal conflicts.

types of faults, it must apply a signal value (or values) at the fault site that is the opposite of the value produced by the fault, and must subsequently move the resulting fault effect forward (fault propagation) by sensitizing a path from the fault site to a scan cell or a primary output.

To reduce a pattern count, a single test vector could be used to detect several nonequivalent faults. An attempt to find such a pattern may lead, however, to conflicts, as shown in Fig. 1. For example, to propagate faults from the cone of logic C_1 through AND gate G_1, input s_1 must be set to 1. At the same time, to propagate faults from cone C_2, input s_2 of OR gate G_2 must assume the value of 0 resulting in conflicting values at stem s_0. For the sake of brevity, in the remaining parts of this paper, the term faults C_k refers to faults occurring within the cone of logic C_k (also described as native faults of C_k) unless stated otherwise. Clearly, due to conflicting values at stem s_0, faults C_1 and C_2 cannot be detected by the same test pattern. In general, conflicts between logic values within a given test stimulus occur due to incompatible decisions made by ATPG during backward justification or fault propagation preceded by fault excitation.

Unlike a conventional ATPG flow, faults C_1 and C_2 in Fig. 1 can be detected in parallel provided a control test point is placed on one of the stem s_0 branches to resolve the conflict. In particular, an additional OR gate on branch s_1 would achieve 1-controllability of this line; that is, it can be set to 1 whenever needed. Alternatively, an extra AND gate placed on branch s_2 would allow 0-controllability of that net. Unfortunately, traditional testability-based test points may not become aware of conflicts similar to the one illustrated in Fig. 1. This is because they use inadequate metrics when working with deterministic test patterns. For example, let a scan cell directly drive stem s_0. Clearly, the probability of setting s_0 to either 0 or 1 would be the same, i.e., 50%. As a result, a conflict between incompatible requirements for branches s_1 and s_2 is going to be overlooked.

In the following sections, we introduce a comprehensive characterization of ATPG-based internal conflicts that can be found in digital circuits and propose techniques to resolve such conflicts by inserting conflict-aware control test points at the most appropriate locations.

III. Fault Blocking

Internal conflicts can be quantified through a fault-blocking mechanism. We determine the number of blocked faults,

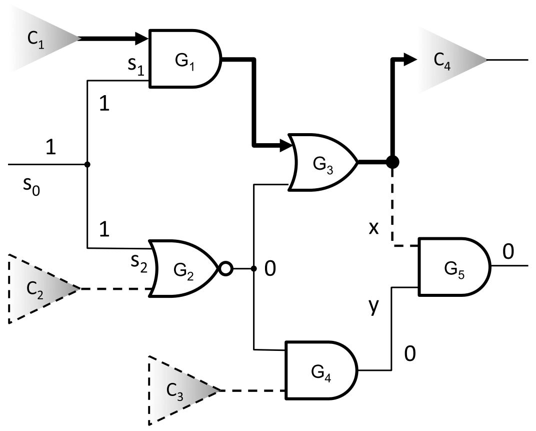


Fig. 2. Forward value propagation.

i.e., faults whose propagation is blocked, based exclusively on the results of forward value propagation. Clearly, a logic value v can propagate forward provided one can uniquely set the outputs of gates visited along propagation paths. In particular, it applies to scenarios where v is either a controlling value for a visited gate, or all its inputs assume a noncontrolling value. If a controlling value drives AND, NAND, OR, or NOR gates, then these gates are said to block faults. For example (see Fig. 2), if one sets stem s_0 to 1, then it causes the output of gates G_2, G_4, and G_5 to assume the value of 0. As a result, faults C_2 and C_3 are blocked (dotted lines) due to 1 at input s_2. Also, faults affecting input x of gate G_5 are blocked, as their only propagation path leads to the output of G_5. Consequently, the only propagation path for faults C_1 leads through gates G_1 and G_3, and then cone C_4 (bold lines). On the other hand, input s_1 of gate G_1 set to 0 blocks its second input, and thus, any fault observed on the output of C_1 cannot move toward a circuit output.

It is worth noting that XOR (XNOR) gates allow fault propagation only if all inputs of such a gate are specified. Furthermore, faults arriving at a multiplexer's data input are blocked if the address input does not select the very same input. Faults propagating through the select input are blocked if all data inputs assume the same value.

Additional fault propagation rules apply to fan-out-free regions (FFRs). Let us assume that cones presented in Fig. 2 are FFRs. If a single-output FFR is blocked, then all FFR native faults and faults whose only propagation route goes through this FFR are unable to propagate to the output (e.g., faults C_2 and C_3 in Fig. 2). However, if an FFR features an output fan-out (it drives more than a single gate), fault propagation depends on the status of its output branches. Consider the circuit shown in Fig. 3. Output s_0 of C_2 set to 0 blocks all fan-out branches of C_1, and therefore, faults C_1 cannot propagate to the output. On the other hand, having additional branches on the output of C_1 that do not reach gates or FFRs already blocked would allow faults C_1 to avoid blocking due to alternative propagation routes.

IV. Conflict Appraisal

Consider the circuit of Fig. 4. In order to enable the propagation of faults C_3, input x_1 of gate G_4 has to be set to 0 [Fig. 4(a)]. As a result, at least one of inputs of G_2 must assume the value of 0, too. It precludes, however, the propagation of faults C_1, C_2, (or both) and C_4. One could propagate these faults by having 1 on the output of gate G_2 [Fig. 4(b)]. Unfortunately, this assignment blocks, in turn, fault C_3. As can be seen, there is a conflict at branch x_1 between logic values being forward implied and those resulting from backward justifications that are needed to propagate different groups of faults.

In order to resolve the conflict, to make it possible to simultaneously detect faults C_1, C_2, C_3, and C_4, and thus to

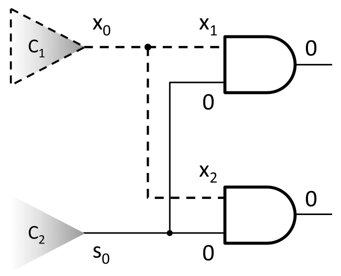


Fig. 3. Fault blocking for FFR.

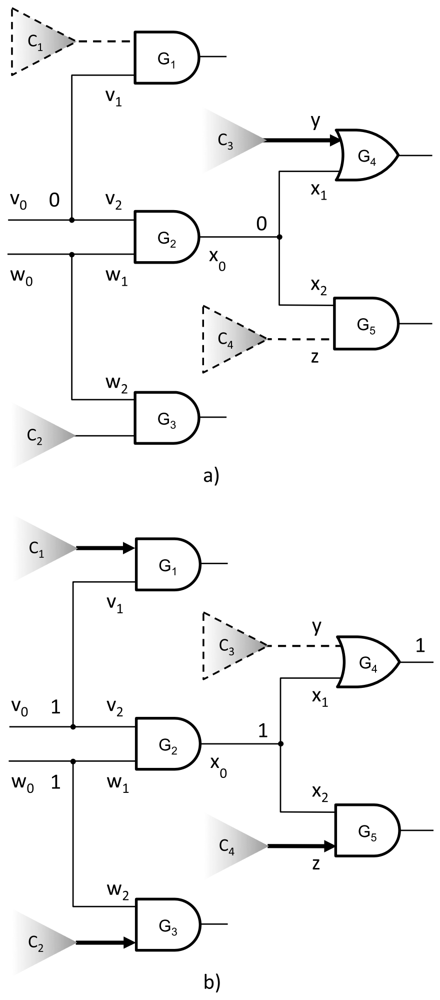


Fig. 4. Conflict on branch x_1. (a) x_0 set to 0. (b) x_0 set to 1.

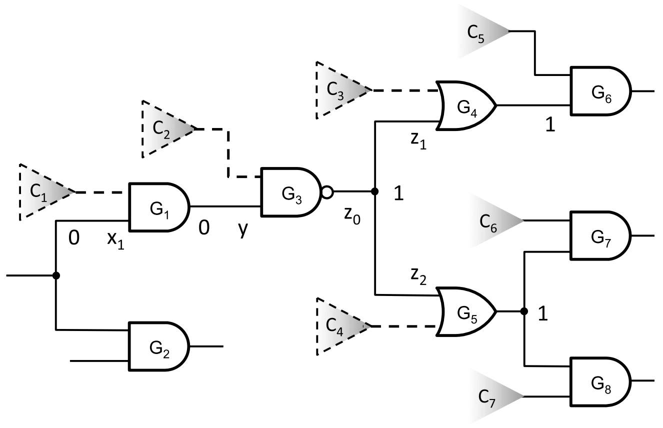


Fig. 5. Computation of metric B_{x_1}.

reduce finally a pattern count, one can insert a 0-injection circuit on branch x_1, where an extra AND gate (an AND CP) can be used to achieve 0-controllability of this net. A proper TPI is preceded by a conflict identification process. It employs the following four metrics being gradually assigned to internal lines of a circuit:

1) b_x and B_x—the number of 0s and 1s, respectively, needed on net x to enable propagation of faults through all relevant gates;
2) f_x and F_x—the number of forward-implied 0s and 1s, respectively, on line x due to earlier backward justifications.

It is worth noting that the definition of metrics b_x and B_x implies also their contrapositive meaning. Indeed, they can alternatively be regarded as the number of blocked faults when line x is set to 1 and 0, respectively. This rule can be used in conjunction with a circuit tracing to obtain estimations of b_x and B_x. For stem x_0 with fan-out branches $x_1 \ldots x_n$, one can determine b_{x_0} and B_{x_0} as follows:

$$b_{x_0} = \sum_{i \neq 0} b_{x_i} \tag{1}$$

$$B_{x_0} = \sum_{i \neq 0} B_{x_i}. \tag{2}$$

Example: We will determine the value of B_{x_1} in a circuit of Fig. 5. Let cone C_k host $|C_k|$ faults. The number of 1s required on branch x_1 to enable the propagation of faults through all relevant gates matches the number of blocked faults when x_1 is set to 0. As can easily be verified, after injecting 0 at x_1, the forward value propagation continues all the way until gates G$_6$, G$_7$, and G$_8$, where faults C_5, C_6, and C_7 remain observable due to a noncontrolling value reaching these gates. Now, going backward, 1s at branches z_1 and z_2 block faults C_3 and C_4, respectively. This also implies that branch z_1 must be set to 0 at least $|C_3|$ times to propagate fault C_3, i.e., $b_{z_1} = |C_3|$. Similarly, $b_{z_2} = |C_4|$. From (1), we get that $b_{z_0} = b_{z_1} + b_{z_2} = |C_3| + |C_4|$. Going further backward, we pass gate G$_3$ where B_y (note the inversion) becomes equal to $|C_2| + b_{z_0}$. This is because the output metric b_{z_0} adds up to the amount of blocked (due to 0 at line y) faults occurring within the cone of logic C_2. Finally, the value of B_{x_1} is computed as $B_y + |C_1|$ using the same principles as before.

Having defined metrics b and B, we can now introduce metrics f and F for outputs of successive gates and stem branches. As an example, consider gate G$_2$ of Fig. 4. Let v_0 and w_0 be primary inputs. Inputs v_1 and w_2 (and, hence, v_2 and w_1) must be set to 1 at least $|C_1|$ and $|C_2|$ times in order to propagate faults C_1 and C_2, respectively. However, to determine the total number of 0s and 1s seen at the inputs of gate G$_2$, one should take into account 0s and 1s implied forward through stems v_0 and w_0. Since they are primary inputs, metrics f_{v_0}, F_{v_0}, f_{w_0}, and F_{w_0} are all equal to 0. Thus, $F_{v_2} = |C_1|$ and $F_{w_1} = |C_2|$. Moreover, f_{v_2} and f_{w_1} are both 0, as 0 is a controlling value for gates G$_1$ and G$_3$ where it would block fault propagation. Clearly, the number F_{x_0} of forward-implied 1s through gate G$_2$ is a function of F_{v_2} and F_{w_1}. Since G$_2$ is the AND gate, we propose F_{x_0} to attain the minimum value of $\{F_{v_2}, F_{w_1}\}$, i.e., the numbers of 1s at the G$_2$ inputs. On the other hand, as 0 is the AND gate controlling value, the number f_{x_0} of 0s propagated through gate G$_2$ gets the maximum of $\{f_{v_2}, f_{w_1}\}$, i.e., the numbers of 0s at its inputs. In this particular case, f_{x_0} equals 0 because neither gate G$_1$ nor gate G$_3$ requires 0 to propagate faults C_1 or C_2.

Given a circuit and its fault list, a structural analysis of the design yields the values of f and F, in particular, by finding faults that would otherwise be blocked, if 1 and 0 are applied, respectively, to a gate input. Given the numbers f_k and F_k of desired 0s and 1s on a gate kth input, the corresponding output metrics f_s and F_s can be obtained by adopting the principles of the previous paragraph. Consequently, we get the following formulas for the basic types of gates:

$$\begin{aligned} f_s &= f_k \\ F_s &= F_k \end{aligned} \tag{3a}$$

$$\begin{aligned} f_s &= F_k \\ F_s &= f_k \end{aligned} \tag{3b}$$

$$\begin{aligned} f_s &= \max\{f_k\} \\ F_s &= \min\{F_k\} \end{aligned} \tag{4a}$$

$$\begin{aligned} f_s &= \min\{F_k\} \\ F_s &= \max\{f_k\} \end{aligned} \tag{4b}$$

$$\begin{aligned} f_s &= \min\{f_k\} \\ F_s &= \max\{F_k\} \end{aligned} \tag{5a}$$

$$\begin{aligned} f_s &= \max\{F_k\} \\ F_s &= \min\{f_k\}. \end{aligned} \tag{5b}$$

The above equations correspond to buffer (3a), inverter (3b), AND (4a), NAND (4b), OR (5a), and NOR (5b) gates. Slightly more complex formulas are employed to determine the output f/F metrics for a two-input XOR gate (6), and then a two-input multiplexer (7) whose data inputs are a and b, whereas c is its select input

$$\begin{aligned} f_s &= (\min\{f_a, f_b\} + \min\{F_a, F_b\})/2 \\ F_s &= (\min\{f_a, F_b\} + \min\{F_a, f_b\})/2 \end{aligned} \tag{6}$$

$$\begin{aligned} f_s &= \min\{\max\{f_c, f_a\}, \max\{F_c, f_b\}\} \\ F_s &= \max\{\min\{f_c, F_a\}, \min\{F_c, F_b\}\}. \end{aligned} \tag{7}$$

For the sake of illustration, consider a multiplexer. The number f_s of 0s at its output should take account of 0s at inputs a, b, and c. Since the multiplexer functionality is given by *ac*' + *bc,* we propose to estimate the number of 0s originating at input a by using (4a), i.e., as the maximum of setting the select input to 0 and the number f_a of forward-implied 0s seen at this input. Similarly, the number of 0s coming from input b becomes the maximum of f_b and the number F_c of 1s at the select input. Finally, since the output of multiplexer is driven by the OR gate, f_s in (7) estimates the amount of 0s at the output as the minimum—see (5a)—of the derived maxima. Similar computations are carried out for F_s.

When a metric f hits stem x_0 with fan-out branches $x_1 \ldots x_n$, the number f_{x_k} of 0s that will further propagate through branch x_k not only includes the value of f_{x_0} (the number of 0s reaching stem x_0), but also accounts for a phenomenon of reflecting back into a circuit 0s represented by metrics b associated with all branches but x_k. Similarly, one can determine the number F_{x_k} of 1s that will propagate through successive branches

$$f_{x_k} = f_{x_0} + \sum_{i \neq k} b_{x_i} \tag{8}$$

$$F_{x_k} = F_{x_0} + \sum_{i \neq k} B_{x_i}. \tag{9}$$

Consider the computation of f_{x_1} and F_{x_1} for gate G$_4$ in Fig. 4. Let cones C_3 and C_4 have $|C_3|$ and $|C_4|$ faults, respectively. The number f_{x_1} of 0s reaching input x_1 is equal to the number f_{x_0} of forward-implied 0s on stem x_0 plus b_{x_2} (these are 0s required at input x_2 of gate G$_5$ to propagate faults C_4). As can be seen, $b_{x_2} = 0$, since 0 is the AND gate controlling value. As a result, $f_{x_1} = f_{x_0}$. Furthermore, F_{x_1} equals the number F_{x_0} of 1s at x_0 plus the number B_{x_2} of 1s needed to propagate fault C_4. Thus, $F_{x_1} = F_{x_0} + |C_4|$. Similarly, we get $f_{x_2} = f_{x_0} + b_{x_1} = f_{x_0} + |C_3|$ and $F_{x_2} = F_{x_0} + B_{x_1} = F_{x_0}$.

We can now conclude our study of metrics b, B, f, and F by looking at how they relate to ATPG-based conflicts. One way to measure the degree of conflicts occurring at a given line x is to employ the following formulas:

$$c_x = \min\{b_x, F_x\} \tag{10}$$

$$C_x = \min\{B_x, f_x\}. \tag{11}$$

They estimate the amount of inconsistency between fault propagation conditions and the corresponding forward-implied values. In particular, the value of c_x gives the degree of conflict between the number b_x of 0s needed to propagate faults through certain gates (recall that this is the number of blocked faults when x is set to 1), and the number F_x of 1s that should propagate forward through net x.

V. Test Points Insertion

This section details our TPI procedure. Following the key observations of Section II, this approach repeatedly targets fan-out branches as the most suitable locations of potential conflict-aware CPs. To identify such sites, we compute metrics b, B, f, and F by traversing, in a gate-level order and in a single pass, the entire circuit. Starting from the first level, metrics f and F are determined for each gate by using (3)–(7). Clearly, (8) and (9) are employed whenever a fan-out is encountered. In this case, however, we also compute the number of blocked faults due to 1s and 0s occurring at a given branch to find metrics b and B to be subsequently employed, through (10) and (11), to assess the degree of conflicts observed at the fan-out branches. As a result, this single-pass traversal (the first iteration of our procedure) yields a sorted, in a descending order, list of conflicts represented by either c_x or C_x.

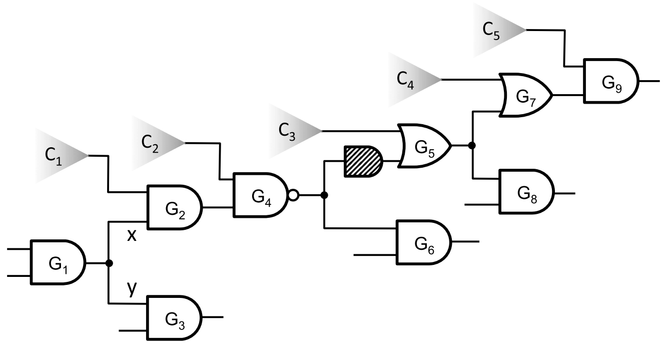


Fig. 6. Test point insertion.

Having all conflicts sorted, we repeatedly remove the top of the list, i.e., the largest degree conflict c_x or C_x, and insert the corresponding AND or OR CP, respectively, at a designated fan-out branch x. Since a new CP may affect metrics of all items currently on the list, it needs to be updated before we continue. The TPI procedure iterates until the number of inserted test points matches the desired and user-defined number of test points that can be added into a design.

Consider the circuit in Fig. 6. The number B_x of 1s needed at input x of gate G$_2$ (or, alternatively, the number of blocked faults when line x is set to 0) is equal to $|C_1| + |C_2| + |C_3| + |C_4|$. If an AND CP is added as shown in Fig. 6 (all CPs are represented by shaded gates), then B_x changes to $|C_1| + |C_2|$ indicating that input x must be now set to 1 at least $|C_1| + |C_2|$ times to ensure the propagation of faults C_1 and C_2. The new value of B_x alters—see (9)–(11)—C_x, F_y, and c_y, where y is another branch of the same fan-out as that of x. These particular changes give rise to subsequent updates in areas driven by gate G$_1$. Furthermore, addition of the same CP leads to recomputing of f and F associated with logic affected by the test point. For example, the branch with already inserted CP will have revised values of b, B, f, and F. Again, subsequent updates implied on other nets must be taken into account.

To update conflict metrics (10) and (11), values of b and B are recomputed first, beginning with the branch holding a newly inserted test point. Furthermore, the lowest level order stem affected by the same CP is also determined. Thereafter, the values of f, F, c, and C are recomputed, accordingly. Clearly, this phase is the following graph traversal [2].

1) Set the stem driving a branch with the new CP as a starting point.
2) While moving backward toward inputs, visit all structurally reachable stems, update values of b and B

Algorithm 1 Test Point Insertion Flow

```
Input: Netlist, fault list, the number of desired
    test points
Output: Netlist with inserted test points
1   For each gate compute metrics $f$ and $F$; in the
    case of fan-out determine metrics $b$, $B$, and then
    the degree of conflicts $c$ and $C$.
2   Insert a test point at a fan-out branch with the
    largest degree of conflict.
3   Update metrics for all affected gates.
3.1     Starting from a branch hosting a newly inserted
        test point, update metrics $b$ and $B$ moving
        backward towards inputs.
3.2     Recompute the values of $f$, $F$, $c$, and $C$ moving
        forward towards outputs.
4   Reconsider earlier TPs decisions.
4.1     Remove a given test point.
4.2     Recompute all relevant metrics.
4.3     Verify whether the degree of conflicts is
        smaller than a conflict metric of another poten-
        tial test point location. If this not the case, rein-
        sert the test point and restore all relevant
        metrics.
5   Continue steps 2 – 4 until the number of inserted
    test points matches their desired number.
```

associated with a path being traversed, and flag stems already visited, so that the path is pursued no further.

3) For a given path, continue step 2 until no changes can be made to both b and B.
4) Given all visited stems, pick the lowest level order stem s.
5) Beginning with s and using (3)–(9), move forward toward outputs in a gate-level order, updating values of f, F, c, and C, until no further changes can be made.

A newly inserted test point makes the above procedure revising earlier decisions, i.e., some existing CPs which turn out to be less effective than expected are given up, and the list of conflicts is updated accordingly. As a result, some list nodes can move forward in accordance with the values returned by (10) and (11). Hence, for each CP within a circuit area affected by the recent changes, we examine the feasibility of replacing this CP by another candidate through the following steps.

1) Remove a given test point at branch x_k.
2) Recompute all relevant values of b, B, f, F, c, and C by means of the traversal process defined earlier.
3) Verify whether the new value of c_{x_k} or C_{x_k} is smaller than a conflict metric assigned currently to a "past-the-end" element; that is, an item that follows the last item in the list that remains a potential test point site. If so, decrease the number of selected test points as there are better candidates than the one used so far. Otherwise, reinsert the test point and restore all relevant metrics.

A summary of our conflict-aware TPI flow is shown in Algorithm 1.

TABLE I
METRICS FOR FIG. 7

	Net	No CP	CP α	CP α+β	CP β
b	x_1	0	0	0	0
B		1,500	0	0	500
f		1,300	1,300	1,300	1,300
F		100	100	100	100
c		0	0	0	0
C		1,300	0	0	500
f	x	-	0	0	-
F		-	100	100	-
b	x_2	300	300	300	300
B		0	0	0	0
f		1,000	1,000	1,000	1,000
F		1,600	100	100	600
c		300	100	100	300
C		0	0	0	0
f	y_0	1,300	600	600	1,300
F		100	100	100	100
b	y_1	0	0	0	0
B		1,000	1,000	0	0
f		1,800	1,100	1,100	1,800
F		100	100	100	100
c		0	0	0	0
C		1,000	1,000	0	0
f	y	-	-	0	0
F		-	-	100	100
b	y_2	500	500	500	500
B		0	0	0	0
f		3,100	600	600	1,300
F		1,100	1,100	100	100
c		500	500	100	100
C		0	0	0	0
f	z_0	1,800	1,100	300	300
F		100	100	100	100
b	z_1	500	500	500	500
B		0	0	0	0
f		3,800	3,100	2,300	2,300
F		100	100	100	100
c		100	100	100	100
C		0	0	0	0
b	z_2	2,000	2,000	2,000	2,000
B		0	0	0	0
f		2,300	1,600	800	800
F		100	100	100	100
c		100	100	100	100
C		0	0	0	0

Example: Table I shows how the test point selection procedure determines CPs in a circuit of Fig. 7. Let the outputs of gates G_6 and G_7 be the primary ones. We also assume that other parts of the circuit (not shown) have no impact on metrics computed for nets discussed in this example. Let $|C_1| = 500$, $|C_2| = 300$, $|C_3| = 1000$, $|C_4| = 500$, $|C_5| = 500$, and $|C_6| = 2000$. Moreover, let the number of forward-implied 0s and 1s for lines x_0, v, and w be $f_{x_0} = 1000$, $F_{x_0} = 100$, $f_v = 600$, $F_v = 500$, $f_w = 300$, and $F_w = 700$, respectively. The four main columns of Table I list metrics associated with indicated lines for the following four scenarios: no CPs (No CP), a CP α inserted (CP α), two CPs α and β added (CP $\alpha+\beta$), and only a CP β present (CP β). Note that lines x_1 and y_1 drive gates G_2 and G_4 directly unless x and y replace them as the outputs of CPs.

First, we examine column "No CP," identify a branch with the largest degree conflict, and place an OR CP α on

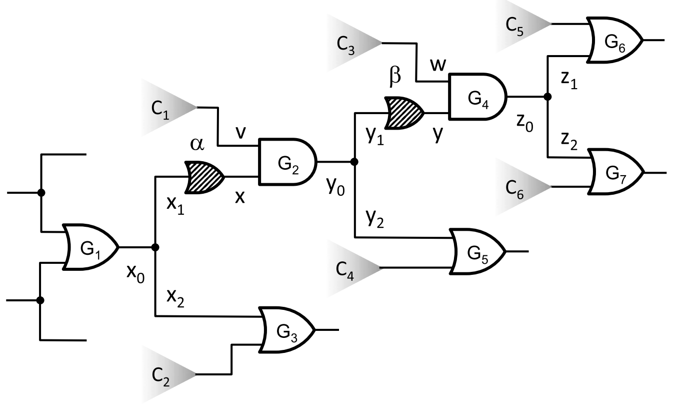
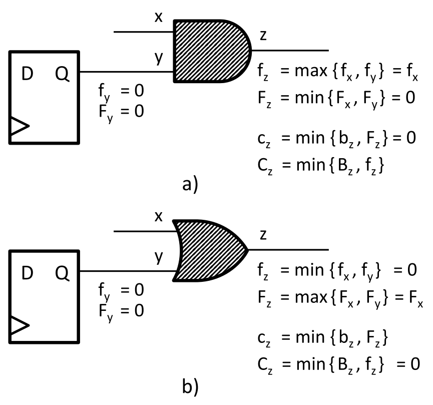
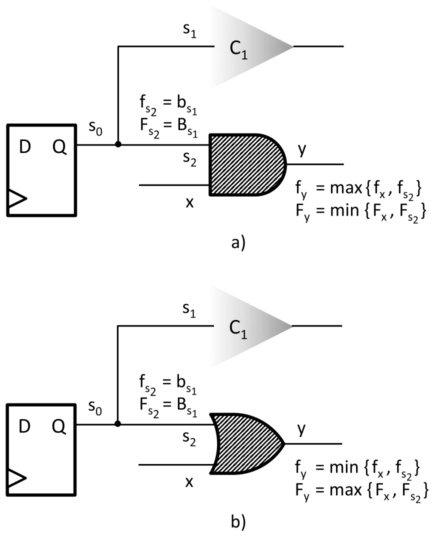


Fig. 7. Test point insertion example.



x
y
z
D Q
$f_y = 0$
$F_y = 0$
$f_z = \max\{f_x, f_y\} = f_x$
$F_z = \min\{F_x, F_y\} = 0$
$c_z = \min\{b_z, F_z\} = 0$
$C_z = \min\{B_z, f_z\}$
a)
$f_z = \min\{f_x, f_y\} = 0$
$F_z = \max\{F_x, F_y\} = F_x$
$c_z = \min\{b_z, F_z\}$
$C_z = \min\{B_z, f_z\} = 0$
b)

Fig. 8. Conflict analysis for dedicated flip-flops. (a) AND-type CP. (b) OR-type CP.



s_1
C_1
s_0
s_2
D Q
$f_{s_2} = b_{s_1}$
$F_{s_2} = B_{s_1}$
x
y
$f_y = \max\{f_x, f_{s_2}\}$
$F_y = \min\{F_x, F_{s_2}\}$
a)
$f_y = \min\{f_x, f_{s_2}\}$
$F_y = \max\{F_x, F_{s_2}\}$
b)

Fig. 9. CPs reusing functional flip-flops. (a) AND-type CP. (b) OR-type CP.

branch x_1 as its C metric is equal to 1300. Second, to update characterization of conflicts, we recompute certain metrics, beginning with the values of b and B on branch x_1. Since metrics b and B at the inputs of gate G_1 remain unchanged, stem x_0 is the lowest level order one affected by the new CP. Beginning with x_0, we move forward in a gate-level order and, by using (3)–(7), update values of f, F, and then c and C, as shown in column "CP α" of Table I. The very same column is subsequently searched for a branch with the largest conflict (here y_1) to be a new CP (β) site. Beginning with y_1, we again update all metrics, as illustrated in column "CP $\alpha+\beta$." Having inserted two CPs, we need to reexamine the role of CP α. Therefore, it is taken off followed by the recomputation of all relevant metrics, as shown in the last column of Table I. As can be seen, the conflict at branch x_1 is now reduced from 1300 in the circuit with no test points to 500 with only CP β being in use. Thus, depending on a conflict value assigned currently to a "past-the-end" element of the potential-test-point-sites list, CP α is either disallowed or restored.

VI. Reuse of Functional Flip-Flops

Typically, inserting test points may incur additional silicon area and has a potential impact on circuit performance [28]. Moreover, resolving timing violations results in additional design iterations. Fortunately, to minimize the area taken up by CPs, functional flip-flops (FF) can be used instead of dedicated flip-flops [29], [18]. In the following sections, we demonstrate that this scenario can also be successfully applied to the conflict-aware test points.

Fig. 8 shows the computations of conflict metrics with CPs in place. Note that f_y and F_y implied by a dedicated flip-flop are both 0s. The original functional path determines f_x and F_x. Clearly, before adding a CP in Fig. 8, we have $f_z = f_x$ and $F_z = F_x$, and the degrees of original conflicts are equal to $c_z = \min\{b_z, F_z\}$ and $C_z = \min\{B_z, f_z\}$. Once an AND CP forces the value of 0, we get $F_z = \min\{F_x, F_y\} = 0$ and a conflict represented by c_z is resolved. Similarly, inserting an OR CP reduces f_z and C_z to 0.

A different scenario applies, if CPs are driven by functional flip-flops rather than dedicated scan cells. While this can significantly reduce silicon area required to instantiate test points, functional flip-flops may already have some native fan-outs. From (8) and (9), it follows that metrics f and F of a given fan-out branch partially depend on values of b and B associated with the remaining fan-out branches of the same stem. Consequently, in contrast to dedicated flip-flops, the values of f and F may not be equal to 0. Thus, reusing functional flip-flops as CP drivers may not reduce the conflict degree to 0, and worse, it may introduce new conflicts at a net interfacing a CP with mission logic. Although reusing functional flip-flops may result in higher test pattern counts due to these extra conflicts, we will demonstrate that it is still possible to successfully tradeoff silicon area and test pattern counts by selecting appropriate drivers through a conflict analysis.

In Fig. 9, the number of forward-implied 0s and 1s at line s_2 is equal to $f_{s_2} = b_{s_1}$ and $F_{s_2} = B_{s_1}$, respectively. Conflicts due to a new flip-flop connection are $c_{s_2} = \min\{b_{s_2}, B_{s_1}\}$ and $C_{s_2} = \min\{B_{s_2}, b_{s_1}\}$. Similar results can be derived for the OR-type CP. As can be seen, functional flip-flop acting as a

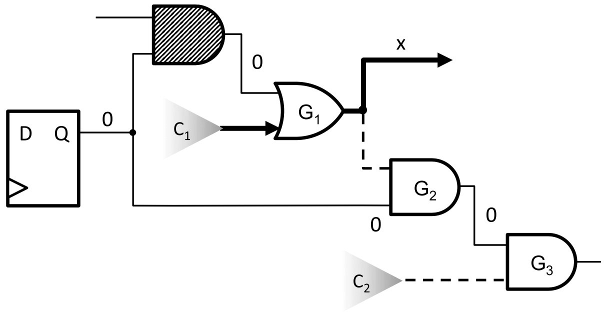


Fig. 10. Test point enabling fault propagation.

test point driver may alter both c and C at this point site. However, to maintain the CP's functionality, i.e., to reduce a conflict corresponding to the CP, and to avoid introducing large conflicts, we propose to select functional flip-flops with a minimal conflict ($c_{s_2} + C_{s_2}$ in the above example). In this case, the CP reduces the original large conflict and keeps other conflicts (caused by a new connection) low. We pick a proper candidate from a set of functional flip-flops that are logically close to the CP bounded by neighborhood criteria detailed in Section VIII. Moreover, since different circuits may have different conflict degrees, a user-defined threshold guides a searching algorithm. Candidates with conflicts exceeding this predefined threshold (or violating timing constraints due to increased fan-outs after adding test points) are not considered, and dedicated flip-flops are used instead. By varying the threshold, it is possible to tradeoff the number of functional flip-flops versus the number of dedicated flip-flops.

VII. Flip-Flops Verification

When reusing a functional flip-flop as a CP driver, newly added connections may form a reconvergent fan-out with the flip-flop output branches. We need, therefore, to check whether connecting functional flip-flops with CPs may compromise fault propagation. To avoid a test coverage drop, flip-flops failing this verification are not considered as drivers of CPs.

During test application, CPs are used either to force a specific value or to allow fault propagation. CPs working with dedicated flip-flops are not exposed to a reconvergence problem. Typically, however, a functional flip-flop acting as a CP driver may need additional connections. For example, a new branch enabling the CP may impact controllability of other gates. To propagate faults C_1 through OR gate G_1 (Fig. 10), one needs to activate the CP by setting the functional flip-flop to 0. But the very same 0 blocks gate G_2 and propagation of fault C_2 through gate G_3. Nevertheless, incompatible assignments made to the flip-flop are already accounted conflicts (see Section VI). Such an inconsistency will only increase a pattern count with no coverage loss. Thus, verification of the reconvergence issue is not needed in this mode.

It is not the case, if a test point is to secure fault propagation. As shown in Fig. 11(a), the driver is set to a noncontrolling value so that faults reaching the other input of the CP can propagate. However, CPs with a driver feeding a

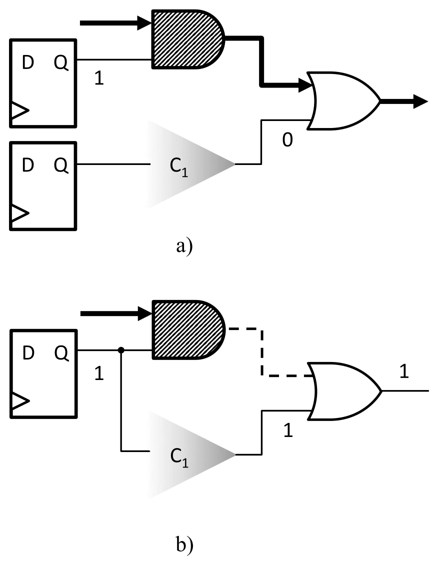


Fig. 11. Fault propagation through CP using (a) dedicated flip-flop and (b) functional flip-flop.

reconvergent fan-out may block faults whose only propagation path goes through that point. Consider the AND-type CP of Fig. 11(b) where the driver is set to 1 to enable the fault propagation through the corresponding CP. Faults whose only propagation path goes through the CP are blocked by the value of 1 dominating the OR gate. In this case, a functional flip-flop precluding fault propagation cannot be considered a driver. Consequently, additional verification is required to check whether the candidate flip-flop may effectively block fault propagation.

After running the conflict analysis of Section VI, the verification procedure sets a candidate functional flip-flop to the noncontrolling value and checks whether this value blocks faults whose only propagation path goes through the CP. For example, given an AND-type CP, we set a candidate flip-flop to 1 and propagate this value forward, as described in Section III. Next, beginning with the gates where the propagation stops, we trace a circuit backward to see whether the faults whose only propagation path leads through the CP, are not blocked. Clearly, candidate flip-flops passing verification will cause no coverage loss due to the reconvergence problem.

VIII. Drivers Search Flow

As discussed in Section VI, selection of functional flip-flops to act as CP drivers is done within a certain (and limited) search space. For prelayout designs, to avoid long paths from selected functional flip-flops to CPs, we select a suitable candidate among flip-flops, which are logically adjacent to the CPs, while a user-defined threshold limits the total number of flip-flops checked for each CP.

Fig. 12 illustrates a search for the most appropriate flip-flop among a limited number of functional flip-flops adjacent to a given CP. Starting with the cone of logic driving the CP, a breadth-first search finds a limited number of candidate flip-flops within the cone. If the cone has not enough

TABLE II
DESIGN CHARACTERISTICS

	Gates	Scan cells	Scan chains	EDT channels	Chain / channel	CPs	OPs	Reuse [%]	TPI runtime [%]	
									Stuck-ats	Transitions
D1	1.19M	72K	400	4, 4	100x	1,364	1,636	99.7	46.1	7.0
D2	103K	1K	10	1, 1	10x	300	300	90.0	44.7	5.7
D3	218K	14K	20	1, 1	20x	379	701	91.3	17.5	3.6
D4	2.08M	143K	400	4, 4	100x	750	750	95.0	4.6	1.6
D5	1.32M	52K	220	6, 6	37x	260	260	94.2	1.6	0.3
D6	1.04M	57K	400	4, 4	100x	600	600	97.2	3.2	1.9
D7	3.34M	325K	400	4, 4	100x	2,973	3,027	90.4	13.6	6.8
D8	2.60M	154K	1200	12, 12	100x	770	770	100.0	2.2	1.3
D9	1.69M	86K	400	4, 4	100x	944	1,056	95.2	5.4	3.0
D10	2.31M	252K	490	10, 10	49x	1,500	1,500	91.3	5.7	3.8

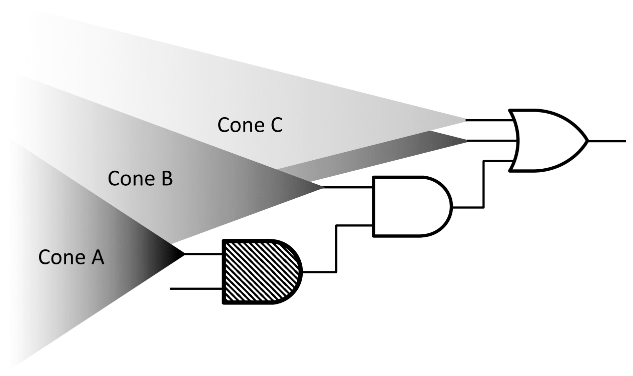


Fig. 12. Cone tracing to find extra candidate flip-flops.

flip-flops, the search space is gradually enlarged by tracing forward from the CP and then tracing backward from any further off-path inputs to find more flip-flops. For example (see Fig. 12), we first check flip-flops within cone A. Subsequently, we may seek more flip-flops by visiting cone B, and then cone C, until reaching a threshold.

The combined search flow to select a suitable functional flip-flop to be used as a driver of a single CP can be summarized as follows.

1) Find a flip-flop with a sum of conflicts below a design-specific threshold and within a certain search space. If there is no such flip-flop, gradually enlarge the search space through back tracing from any off-path input of the CP forward path (see Fig. 12).
2) Run verification for a given flip-flop to analyze fault blocking with the CP set to a nondominating value.
3) If verification fails, repeat steps 1–2 until either a desirable flip-flop is found or the search space is exceeded.
4) If none of the flip-flops within the search space satisfies the above conditions, use a dedicated flip-flop to drive the CP.

The above flow can easily be adapted to work with additional figures of merit that might also guide the search process. These factors include clock or power domains to avoid introducing cross-domain paths between a CP and a candidate flip-flop. A floor plan can be deployed to determine closely spaced functional flip-flops or to identify or exclude CP drivers based on the Euclidean distance between functional flip-flops and nodes where test points need to be inserted. Functional flip-flops may also be selected based on a clock tree supplying majority of flip-flops in a fan-in or fan-out cone of the CP, and only those functional flip-flops are subjected to searching. For multicycle or false paths, if a newly created path from a shared flip-flop cannot be optimized to meet timing closure criteria, the corresponding flip-flops need to be rejected.

IX. Experimental Results

The embedded deterministic test points have experimentally been verified on ten large industrial designs with on-chip test compression. The circuits represent different design styles, different scan methodologies, and mirror the latest technology nodes. In particular, they feature several voltage islands, individual voltage settings, and different clock domains, while the majority of digital logic is synthesized and implemented by standard cell elements, and by employing embedded memory blocks. The designs vary in terms of the initial test coverage and the corresponding pattern counts.

The basic data regarding the designs, such as the number of gates, the number of scan cells, the number of scan chains, embedded deterministic test (EDT) input–output interface, and input compression, are listed in the left part of Table II. The remaining part of Table II reports the number of inserted CPs and OPs, a reuse ratio defined as the percentage of CPs with functional flip-flop-based drivers, and finally a TPI time as a fraction of the corresponding ATPG runtime for stuck-at and transition faults. In all experiments reported in this section, OPs were inserted by the state-of-the-art method presented in [21]. Each OP is served by a single dedicated flip-flop. For each test case, the metrics of Section IV are used to guide the insertion of CPs (as shown in Section V), and subsequently, the method of Section VIII determines drivers of CPs among functional flip-flops. Note that by setting the conflict threshold

TABLE III
EXPERIMENTAL RESULTS FOR STUCK-AT AND TRANSITION FAULTS

	TC [%]	PC	PC reduction		ATPG runtime reduction		TC [%]	PC	PC reduction		ATPG runtime reduction		Post-TPI TC [%]	
			D	F	D	F			D	F	D	F	D	F
D1	96.93	3,078	1.89x	1.82x	1.61x	1.52x	91.62	14,720	2.47x	3.03x	1.76x	1.96x	93.11	93.48
D2	97.78	10,831	2.54x	2.36x	2.91x	2.81x	71.34	8,342	3.62x	3.83x	4.98x	5.21x	86.73	87.19
D3	99.99	10,982	2.68x	2.48x	1.77x	1.58x	91.92	30,400	3.32x	3.74x	2.47x	2.69x	93.47	93.46
D4	99.49	24,029	2.75x	2.70x	2.46x	2.36x	83.49	39,104	2.57x	2.57x	2.53x	2.59x	84.39	84.69
D5	97.70	32,926	1.34x	1.32x	1.99x	2.02x	83.49	28,224	3.24x	2.98x	3.51x	3.21x	92.57	92.17
D6	91.17	15,360	2.02x	1.97x	2.15x	2.11x	81.90	18,304	2.10x	2.33x	1.74x	1.93x	84.89	85.62
D7	96.48	3,520	1.36x	1.34x	1.42x	1.42x	87.10	6,922	1.61x	1.52x	1.60x	1.55x	88.91	88.75
D8	91.17	19,912	3.10x	3.07x	1.79x	1.74x	87.85	33,017	2.47x	2.37x	1.55x	1.63x	88.16	88.20
D9	97.77	15,872	1.71x	1.69x	1.19x	1.17x	88.31	25,920	2.00x	1.97x	1.62x	1.45x	90.10	90.03
D10	98.53	4,397	2.16x	2.07x	2.04x	1.98x	92.86	5,760	6.92x	6.43x	5.10x	4.98x	98.42	98.47
	STUCK-AT FAULTS						TRANSITION FAULTS							

to a design-specific value, we can achieve a large fraction of CPs working with functional flip-flops as their drivers. Thus, by having a small number of dedicated scan cells to drive CPs, we minimize area occupied by test points. As can be seen (column Reuse in Table II), at least 90% reuse ratio is achieved for each test case; that is, more than 90% of CPs require no customized drivers. Functional flip-flops deployed as drivers of CPs are selected within their "logical" proximity comprising 100 flip-flops, as described in Section VIII.

It is worth noting that test points employed in the presented method use exactly the same hardware and routing as those of conventional LBIST test points. Thus, the impact of the proposed technique on various aspects of design flow is virtually the same as that of the testability-based test points. Furthermore, in order to reduce the overall impact on timing closure, we have restricted the number of CPs on a single path to 5. It is also possible to avoid critical paths altogether by restricting the number of CPs on a given path to 0. Preliminary experimental results show that reducing the number of CPs from 5 to 1 per path across the entire design has impact on pattern count reduction, which varies between designs. However, when some critical paths are excluded, then the impact is usually less severe because of many other paths where CPs can be inserted. Hence, restricting the number of CPs globally to a smaller number for the entire design often has a larger impact on pattern count reduction compared to cases where only a subset of such paths is excluded.

The experimental results obtained for both stuck-at and transition faults are summarized in Table III. Column TC reports the baseline (no test points) test coverage with the corresponding test pattern count in column PC. Reduction of pattern counts attributed to test points using dedicated and functional flip-flops is listed under D and F columns, respectively. Note that the pattern count reduction is reported for the same test coverage as that of a baseline case. The next column provides ATPG runtime reduction relative to a baseline case. The right part of Table III presents similar results for transition faults.

As can be seen, with all test points inserted, column PC reduction expresses a visible drop in the number of test vectors necessary to get the same test coverage as that of the baseline case. Given stuck-at faults, columns D and F demonstrate, on the average, 2.2× and 2.1× test compression increase above the regular compression. Similarly, additional 3× and 3.1× pattern count reductions are achieved for transition faults. It appears that for designs of Table II, the final input compression factors for stuck-at and transition faults are, on the average, equal to 151.7× (D), 148.2× (F) and 188.4× (D), 191.8× (F), respectively. Clearly, the ATPG runtime reduction closely parallels the pattern count reduction. These compression numbers are significant factors that lead to overall test time and tester cost decrease. They also help to reduce the pattern simulation time, which is essential for pattern validation.

The last two columns of Table III list final transition test coverage numbers recorded after inserting all test points (post-TPI TC). As can be seen, regardless of type of test point drivers, there is a visible coverage increase compared to the baseline cases. For stuck-at faults, post-TPI test coverage remains virtually the same as that before TPI.

Furthermore, as shown in Section VI, using functional flip-flops as CP drivers may result in some conflicts, and thus in a pattern count growth. Nevertheless, results for stuck-at faults indicate that with the average 3.3% pattern count increase, conflict-aware CPs using functional flip-flops as their drivers still achieve a significant compression during deterministic test generation. Interestingly, designs D1, D2, D3, and D6 attain higher pattern count reduction for transition faults, if functional flip-flops are deployed rather than dedicated ones. It appears that adding new scan chains changes a fault ordering, and thus, it impacts ATPG performance and the resultant pattern count. In some designs with dedicated cells used as test point drivers, the increase in compression ratio causes an increase in a pattern count, even though ATPG may have more freedom to target faults in this case. For the remaining test cases, additional test vectors, compared to the solution using dedicated scan cells, amount, on average, to 4.7%.

X. Conclusion

In this paper, we introduce a DFT technology that aims at reducing deterministic test pattern counts and test data volume. The proposed scheme identifies the largest internal conflicts precluding efficient ATPG-based test compaction. Locations corresponding to such conflicts, excluding those on critical paths or otherwise not eligible, are modified by inserting conflict-aware test points to increase the number of faults targeted by a single pattern, and thus to reduce ATPG test pattern counts and the resultant test data volume. Experimental results obtained for large industrial designs with on-chip test compression demonstrate, on average, an additional factor of 2×–4× pattern reduction for stuck-at and transition faults over the best results provided by the EDT-based regular compression. Furthermore, much shorter ATPG runtimes and a similar reduction of test data volume are reported. Finally, the scheme minimizes silicon area needed to implement conflict-aware test points by reusing functional flip-flops as their drivers with at least 90% reuse ratio. As a result, it allows one to further extend the life of legacy automatic test equipment and is synergistic with growing transistor-level and defect-based tests deployed in contemporary nanometer designs.

References

[1] C. Acero *et al.*, "Embedded deterministic test points for compact cell-aware tests," in *Proc. ITC*, 2015, paper 2.2.

[2] C. Acero *et al.*, "On new test points for compact cell-aware tests," *IEEE Des. Test*, vol. 33, no. 6, pp. 7–14, Dec. 2016.

[3] F. Brglez, P. Pownall, and R. Hum, "Applications of testability analysis: From ATPG to critical delay path tracing," in *Proc. ITC*, 1984, pp. 705–712.

[4] A. J. Briers and K. A. E. Totton, "Random pattern testability by fault simulation," in *Proc. ITC*, 1986, pp. 274–281.

[5] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Norwell, MA, USA: Kluwer, 2000.

[6] S.-C. Chang, S.-S. Chang, W.-B. Jone, and C.-C. Tsai, "A novel combinational testability analysis by considering signal correlation," in *Proc. ITC*, 1998, pp. 658–667.

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

[8] M. J. Geuzebroek, J. T. van der Linden, and A. J. van de Goor, "Test point insertion for compact test sets," in *Proc. ITC*, 2000, pp. 292–301.

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

[10] L. H. Goldstein and E. L. Thigpen, "SCOAP: Sandia controllability/observability analysis program," in *Proc. DAC*, 1980, pp. 190–196.

[11] F. Hapke *et al.*, "Defect-oriented cell-aware ATPG and fault simulation for industrial cell libraries and designs," in *Proc. ITC*, 2009, paper 1.2.

[12] F. Hapke *et al.*, "Cell-aware test," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 33, no. 3, pp. 1396–1409, Sep. 2014.

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

[14] R. Kapur, S. Mitra, and T. W. Williams, "Historical perspective on scan compression," *IEEE Design Test Comput.*, vol. 25, no. 2, pp. 114–120, Mar. 2008.

[15] A. Kumar, J. Rajski, S. M. Reddy, and T. Rinderknecht, "On the generation of compact deterministic test sets for BIST ready designs," in *Proc. ATS*, 2013, pp. 201–206.

[16] Y. Liu, E. Moghaddam, N. Mukherjee, S. M. Reddy, J. Rajski, and J. Tyszer, "Minimal area test points for deterministic patterns," in *Proc. ITC*, Nov. 2016, paper 2.4.

[17] M. Nakao, K. Hatayama, and I. Highasi, "Accelerated test points selection method for scan-based BIST," in *Proc. ATS*, 1997, pp. 359–364.

[18] M. Nakao, S. Kobayashi, K. Hatayama, K. Iijima, and S. Terada, "Low overhead test point insertion for scan-based BIST," in *Proc. ITC*, 1999, pp. 348–357.

[19] I. Pomeranz and S. M. Reddy, "Test-point insertion to enhance test compaction for scan designs," in *Proc. ICDSN*, 2000, pp. 375–381.

[20] H. Ren, M. Kusko, V. Kravets, and R. Yaari, "Low cost test point insertion without using extra registers for high performance design," in *Proc. ITC*, 2009, paper 12.2.

[21] S. Romersaro, J. Rajski, T. Rinderknecht, S. M. Reddy, and I. Pomeranz, "ATPG heuristics dependant observation point insertion for enhanced compaction and data volume reduction," in *Proc. DFTVS*, 2008, pp. 385–393.

[22] B. H. Seiss, P. Trouborst, and M. Schulz, "Test point insertion for scan-based BIST," in *Proc. ETC*, 1991, pp. 253–262.

[23] 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. ATS*, 2006, pp. 339–348.

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

[25] H.-C. Tsai, C.-J. Lin, S. Bhawmik, and K.-T. Cheng, "A hybrid algorithm for test point selection for scan-based BIST," in *Proc. DAC*, 1997, pp. 478–483.

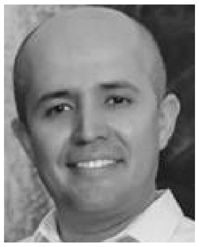
[26] N. A. Touba, "Survey of test vector compression techniques," *IEEE Des. Test Comput.*, vol. 23, no. 4, pp. 294–303, Apr. 2006.

[27] S. Udar and D. Kagaris, "Minimizing observation points for fault location," in *Proc. DFT*, 2009, pp. 263–267.

[28] H. Vranken, S. S. Sapei, and H.-J. Wunderlich, "Impact of test point insertion on silicon area and timing during layout," in *Proc. DATE*, 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] M. Yoshimura, T. Hosokawa, and M. Ohta, "A test point insertion method to reduce the number of test patterns," in *Proc. ATS*, 2002, pp. 298–304.



Cesar Acero received the B.S. degree in electronics engineering and the M.S. degree in computing engineering from the Universidad de Los Andes, Bogotá, Colombia.

In 2003, he joined the University of California, San Diego, CA, USA, as a Researcher, where he was in charge of noninvasive testing for microhemodynamics. In 2005, he joined the Intel Corporation, Hillsboro, OR, USA, where he is responsible for the advanced test methods development within the Intel's Worldwide Manufacturing Validation Engineering Organization. His current research interests include the cell-aware test pattern generation and test point insertion.

Derek Feltham received the B.S. and M.S. degrees in computer engineering from the University of Toronto, Toronto, ON, Canada, in 1985 and 1987, respectively, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 1992.

He is currently a Senior Director with the Intel Corporation, Hillsboro, OR, USA, and manages the Manufacturing Architecture Organization responsible for representing manufacturing needs during IP and product definition across devices, client, and data center. His current research interests include exploring all possible advanced test methodologies to ensure continuing quality, cost, output, and efficiency in the face of ever increasing challenges in next-generation process nodes.

Yingdi Liu received the M.S. degree in electrical engineering from The Ohio State University, Columbus, OH, USA, in 2012. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, The University of Iowa, Iowa City, IA, USA.

His current research interests include automatic test pattern generation, design for testability, and test compression.

Elham Moghaddam (S'07–M'11) received the B.S. and M.S. degrees in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2007, and the Ph.D. degree in computer and electronic engineering from The University of Iowa, Iowa City, IA, USA, in 2011.

She is currently a Software Developer with the Design-to-Silicon Division, Mentor Graphics Corporation, Wilsonville, OR, USA. Her current research interests include the design for testability, low-power embedded test, built-in self-test, and test data compression.

Nilanjan Mukherjee (S'87–M'89–SM'14) received the B.Tech. (Hons.) degree in electronics and electrical communication engineering from IIT Kharagpur, Kharagpur, India, in 1989, and the Ph.D. degree from McGill University, Montreal, QC, Canada, in 1996.

He was with Lucent Bell, Holmdel, NJ, USA. He is currently an Engineering Director with the Design-to-Silicon Division, Mentor Graphics Corporation, Wilsonville, OR, USA. He is a Co-Inventor of the EDT Technology and was a Lead Developer for the leading test compression tool in the industry, TestKompress. His current research interests include the next-generation test methodologies for deep sub-micrometer designs, test data compression, test synthesis, memory testing, and fault diagnosis. He has authored more than 75 technical papers and is a Co-Inventor of 44 U.S. patents.

Dr. Mukherjee was a co-recipient of the Best Paper Award at the 1995 IEEE VLSI Test Symposium, the Best Paper Award at the 2009 VLSI Design Conference, the Best Student Paper Award at the 2001 Asian Test Symposium, the 2006 IEEE Circuits and Systems Society Donald O. Pederson Outstanding Paper Award recognizing the paper on embedded deterministic test published in the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, and the 2012 IEEE International Test Conference Most Significant Paper Award. He served on the program committees of several IEEE conferences.

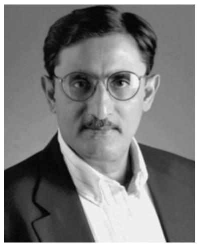
Marek Patyra (M'89) received the M.S. and Ph.D. degrees in electrical engineering from the Warsaw University of Technology, Warsaw, Poland, in 1978 and 1986, respectively.

From 1978 to 1989, he was a VLSI Design Engineer at the Institute of Electron Technology, Warsaw. In 1989, he joined Carnegie Mellon University, Pittsburgh, PA, USA, as a Research Associate. From 1991 to 1996, he was an Associate Professor at the University of Minnesota, Duluth, MN, USA. Since 1996, he has been with the Intel Corporation, Hillsboro, OR, USA, where he is currently a Product Development Engineer. His current research interests include the cell-aware and timing-aware test pattern generation for system-on-chip designs.

Janusz Rajski (A'87–SM'10–F'11) received the M.S. degree in electrical engineering from the Technical University of Gdańsk, Gdańsk, Poland, in 1973, and the Ph.D. degree in electrical engineering from the Poznań University of Technology, Poznań, Poland, in 1982.

From 1973 to 1984, he was a Faculty Member with the Poznań University of Technology. In 1984, he joined McGill University, Montreal, QC, Canada, where he became an Associate Professor in 1989. In 1995, he joined the Mentor Graphics Corporation, Wilsonville, OR, USA, as a Chief Scientist. He is also the Principal Inventor of the Embedded Deterministic Test Technology used in the first commercial test compression product TestKompress. His current research interests include the testing of VLSI systems, design for testability, built-in self-test, and logic synthesis. He has authored more than 180 research papers in these areas and is a Co-Inventor of 100 U.S. patents.

Dr. Rajski was a co-recipient of the 1993 Best Paper Award for the paper on logic synthesis published in the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the 1995 and 1998 Best Paper Awards at the IEEE VLSI Test Symposium, the 1999 and 2003 Honorable Mention Awards at the IEEE International Test Conference, the 2006 IEEE Circuits and Systems Society Donald O. Pederson Outstanding Paper Award recognizing the paper on embedded deterministic test published in the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the 2009 Best Paper Award at the VLSI Design Conference, the 2011 Best Paper Award at the IEEE European Test Symposium, and the 2012 IEEE International Test Conference Most Significant Paper Award. In 1999, he was a Guest Coeditor of the special issue of the *IEEE Communications Magazine* devoted to the testing of telecommunication hardware. He was also an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the IEEE TRANSACTIONS ON COMPUTERS, and the *IEEE Design and Test of Computers Magazine*. He has served on technical program committees of various conferences.



Sudhakar M. Reddy (S'82–M'82–SM'85–F'87–LF'04) received the B.Sc. degree in physics and the B.E. degree in electronic communication engineering from Osmania University, Hyderabad, India, the M.E. degree in electronics and communication engineering from the Indian Institute of Science, Bangalore, India, and the Ph.D. degree in electrical engineering from The University of Iowa, Iowa City, IA, USA.

Since 1968, he has been a Faculty Member with the Department of Electrical and Computer Engineering, The University of Iowa, where he is currently a University of Iowa Foundation Distinguished Professor. He served as the Chair of the Electrical and Computer Engineering Department, from 1981 to 2000. He has authored over 600 papers in archival journals and the proceedings of international conferences.

Prof. Reddy is a member of Tau Beta Pi, Eta Kappa Nu, and Sigma Xi. He received the von Humboldt Senior Research Fellow Award in 1995 and the first Life Time Achievement Award from the International Conference on VLSI Design. Several papers co-authored by him received the best paper nominations and awards. He has served on the technical program committees of several international conferences. He was the Technical Program Committee Chair of the 1989 Fault Tolerant Computing Symposium. He is an Associate Editor and twice a Guest Editor of the IEEE TRANSACTIONS ON COMPUTERS. He is also an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS.

Jerzy Tyszer (M'91–SM'96–F'13) received the M.S. and Ph.D. degrees in electrical engineering from the Poznań University of Technology, Poznań, Poland, in 1981 and 1987, respectively, and the Dr.Hab. degree in telecommunications from the Technical University of Gdańsk, Gdańsk, Poland, in 1994.

From 1982 to 1990, he was a Faculty Member with the Poznań University of Technology. In 1990, he joined McGill University, Montreal, QC, Canada, where he was a Research Associate and an Adjunct Professor. In 1996, he joined the Faculty of Electronics and Telecommunications, Poznań University of Technology, as a Professor. His current research interests include the design automation and testing of VLSI systems, design for testability, built-in self-test, embedded test, and computer simulation of discrete event systems. He has authored eight books and more than 140 research papers in the above areas and is a Co-Inventor of 67 U.S. patents.

Dr. Tyszer was a co-recipient of the 1995 and 1998 Best Paper Awards at the IEEE VLSI Test Symposium, the 2003 Honorable Mention Award at the IEEE International Test Conference, the 2006 IEEE Circuits and Systems Society Donald O. Pederson Outstanding Paper Award recognizing the paper on embedded deterministic test published in the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the 2009 Best Paper Award at the VLSI Design Conference, the 2011 Best Paper Award at the IEEE European Test Symposium, and the 2012 IEEE International Test Conference Most Significant Paper Award. In 1999, he was a Guest Coeditor of the special issue of the *IEEE Communications Magazine* devoted to the testing of telecommunication hardware. He has served on technical program committees of various conferences.

Justyna Zawada (M'14) received the M.S. degree in computer science from Adam Mickiewicz University, Poznań, Poland, in 2012, and the M.S. degree in telecommunications from the Poznań University of Technology, Poznań, in 2014, where she is currently pursuing the Ph.D. degree with the Faculty of Electronics and Telecommunications.

Her current research interests include the design for testability and test security.