Minimal Area Test Points for Deterministic Patterns

Yingdi Liu[1], Elham Moghaddam, Nilanjan Mukherjee, Sudhakar M. Reddy[1]
Janusz Rajski, Jerzy Tyszer[2]

[1]University of Iowa
Iowa City, IA 52242, USA

Mentor Graphics Corporation
Wilsonville, OR 97070, USA

[2]Poznań University of Technology
60-965 Poznań, Poland

Abstract

Conflict-aware test points, introduced recently, facilitate significant reductions in deterministic test pattern counts. However, dedicated flip-flops driving control points increase test logic area. This paper presents a method to minimize silicon area needed to implement conflict-aware test points by reusing functional flip-flops as drivers of control points. Conflict analysis is applied during the test point selection process, and ATPG verification is run for every potential candidate. Experimental results show that functional flip-flops can be reused as drivers for more than 90% of the control points with the average of 5% penalty in pattern count increase as compared to methods using only dedicated flip-flops. After replacing dedicated flip-flops with functional flip-flops, conflict-aware test points can still achieve remarkable pattern count reductions.

1. Introduction

Contemporary Automatic Test Pattern Generation (ATPG) can deliver test patterns for different fault models and for large and complex nanometer designs with high fault coverage. Working synergistically with test compression [10], [21], ATPG is capable of producing highly compact test sets. However, as a result of fast growing semiconductor manufacturing processes and the increasing complexity of digital designs, the inflated test data volume becomes a significant contributor to the test application time, which, in turn, has a visible impact on the overall test cost. In response to these challenges, a conflict-aware test point insertion (TPI) technology [1] was recently proposed as a solution to reduce the size of test sets.

Typically, TPI adds control points (CPs) and observation points (OPs) to internal lines of a circuit to handle hard-to-detect faults and to improve test coverage. Control points use dedicated flip-flops to drive extra AND/OR gates which are capable of setting internal signal lines to desired values when the control points are enabled. Observation points are added as new scan cells to make internal nodes observable. Since finding optimal test points for many circuits is an NP-complete task [4], several empirical methods of finding suitable test point locations have been proposed [7], [13], [16], [19], [22], [24]. Circuit testability used by some of these techniques is usually assessed either through exact fault simulation [3], [9], [19], or approximate measures [2], [5], [7], [8], [12], [17], [20].

Although testability-based TPI technologies may incidentally decrease pattern counts [6], [11], their performance is significantly less predictable than that of the conflict-aware TPI scheme [1] whose primary objective is to reduce ATPG pattern counts and test data volume by increasing the total number of faults targeted by a single pattern. This is accomplished by identifying conflicts between internal signals, and by resolving those conflicts with the help of control points. Similarly to testability-based control points, the conflict-aware control points require dedicated drivers. However, these additional flip-flops inevitably introduce area and performance overhead. Several TPI methods that improve testability while minimizing silicon area have been discussed in [15], [18], [23].

The approach presented in this paper, instead of deploying dedicated flip-flops, reuses functional flip-flops as drivers of control points, hence reducing silicon area required to implement test logic. The selection of proper candidates takes advantage of a conflict analysis working synergistically with an ATPG-based verification. Moreover, the immediate vicinity of potential test points is also considered as a search factor to find the most suitable functional flip-flops within a limited search range.

This paper is organized as follows. In Section 2, we review the background of the conflict-aware test point insertion technique, including computation of conflict metrics and conducting successive insertion steps. Section 3 presents the method that uses conflict analysis to select functional flip-flops as drivers of control points. In Section 4, we discuss ATPG-based verification that must be applied to potential candidates to avoid fault blocking. Section 5 summarizes a complete search algorithm. Experimental results are presented in Section 6, and the paper concludes with Section 7.

2. Conflict-aware test points

In this section, we review the basics of the conflict-aware test point insertion technique [1] and its main steps. In order to detect as many faults as possible with a single pattern, these faults must become parallel targets provided there are no internal assignment conflicts. Given a specific node in a design, two groups of faults that require opposite values at that node cannot be detected simultaneously. As shown in Fig. 1, gate G1 requires an input value of 1 to propagate faults from group C1, whereas gate G2 requires an input value of 0 to enable propagation of faults C2.

978-1-4673-8773-6/16/$31.00 ©2016 IEEE

Clearly, the immediate conflict between backward justified values at G1 and G2 precludes detection of C1 and C2 at the same time. In general, incompatible assignments made

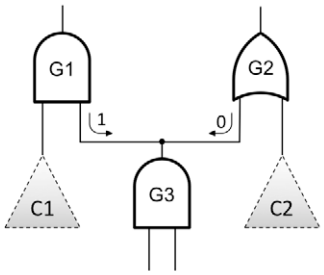

Figure 1 Conflict on internal lines

by ATPG during fault excitation, forward propagation, or backward justification lead to conflicts on internal signal lines that may stop propagation of many faults. Fortunately, by inserting appropriate control points, such conflicts can be successfully resolved. Note that faults C1 and C2 in Fig. 1 could be targeted simultaneously if an AND type control point (for the purpose of 0-controllability) was added to the input of gate G2, or an OR type control point (to achieve 1-controllability) was inserted at the input of gate G1.

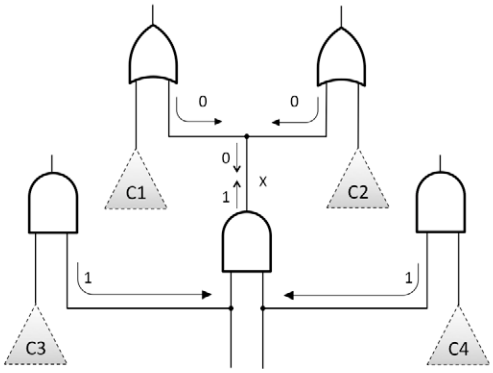

Figure 2 Conflict between forward propagation and backward justification

The example of Fig. 1 suggests that fault-blocking mechanisms can be used to quantify the effect of internal signal assignments on faults detected by one test pattern. Having a line set to a certain logic value may block forward propagation of several faults to an observation point. Thus, the opposite value on that line becomes a necessary ATPG assignment for the same group of faults. As shown in Fig. 2, faults C1 and C2 require stem x to be set to 0, whereas faults C3 and C4 require 1s on the inputs of respective AND gates. Clearly, there is a conflict at line x between the forward-implied value of 1 and the backward-justified value of 0.

In order to determine conflicts such as those illustrated in Figures 1 and 2, in [1] fault-blocking information is forward-implied and backward-justified throughout the circuit. Four metrics are used to characterize the internal signal assignments during the ATPG process. Given node x, these metrics are defined as follows:

- B_x – the number of faults whose propagation could be halted (blocked), if node x was set to 0; this is equivalent to the count, in the process of backward justification, how many times one needs to set x to 1 in order to propagate these faults through all relevant gates,
- b_x – the same as above but to characterize the process of backward justification with respect to 0,
- F_x – the number of forward-implied 1s on node x based on earlier backward justifications,
- f_x – the number of forward-implied 0s on node x based on earlier backward justifications.

For a fan-out stem x_0 with fan-out branches denoted as x_i, the above four metrics can be computed using the following formulas:

$$B_{x_0} = \sum B_{x_i} \tag{1}$$

$$b_{x_0} = \sum b_{x_i} \tag{2}$$

$$F_{x_k} = F_{x_0} + \sum B_{x_i} \quad i \neq k \tag{3}$$

$$f_{x_k} = f_{x_0} + \sum b_{x_i} \quad i \neq k \tag{4}$$

In addition, forward-implied F and f values can also be computed for outputs of different gates based on F and f values assigned to their inputs. This is done through a simple structural analysis. F and f values of output node s are determined as follows [1]:

$$\begin{matrix} f_s = f_k \\ F_s = F_k \end{matrix} \tag{5a} \qquad \begin{matrix} f_s = F_k \\ F_s = f_k \end{matrix} \tag{5b}$$

$$\begin{matrix} f_s = \max\{f_k\} \\ F_s = \min\{F_k\} \end{matrix} \tag{6a} \qquad \begin{matrix} f_s = \min\{F_k\} \\ F_s = \max\{f_k\} \end{matrix} \tag{6b}$$

$$\begin{matrix} f_s = \min\{f_k\} \\ F_s = \max\{F_k\} \end{matrix} \tag{7a} \qquad \begin{matrix} f_s = \max\{F_k\} \\ F_s = \min\{f_k\} \end{matrix} \tag{7b}$$

The above formulas correspond to buffer (5a), inverter (5b), AND (6a), NAND (6b), OR (7a) and NOR (7b) gates.

Once forward-implied and backward-justified information is known, we can measure the degree of conflicts for a given node x_k as follows:

$$c_{x_k} = \min\{b_{x_k}, F_{x_k}\} \tag{8}$$

$$C_{x_k} = \min\{B_{x_k}, f_{x_k}\} \tag{9}$$

An example of computing metrics (8) and (9) is shown in Fig. 3. For a stem x, forward-implied values on each fan-out branch are computed by the forward-implied values on x and the backward-justified values on other branches.

Test point insertion steps of [1] can be summarized as follows:

1. Compute the conflict metrics and sort, in a descending order, the list of internal lines by the conflict degrees.
2. Insert the corresponding control point to the net with the maximum value of the conflict metric and remove it from the list.

3. Update the conflict metrics with a newly added control point.
4. Check whether the control points added earlier remain as effective as expected. Some of these control points can be removed, if they are less effective.
5. Repeat steps 2, 3, and 4 until a pre-defined number of control points are obtained.

3. Reuse of flip-flops - conflict analysis

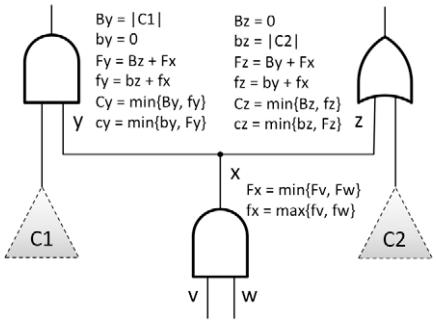

Figure 3 Conflict metrics

As discussed in Section 2, to resolve a conflict with a large value of degree *c*, an AND type control point needs to be added, and to resolve a conflict with a large value of degree *C*, we need to add an OR type control point to the corresponding node.

The basic structures of control points are shown in Fig. 4. A single control point consists of an AND/OR gate to set the control point to the dominating value, another gate to enable it, and a dedicated flip-flop to drive the control point. Dedicated flip-flops added for the purpose of test points are finally stitched to scan chains once the test point insertion is completed.

Fig. 5 shows computations of conflict metrics after insertion of control points. Note that F_R and f_R implied by the dedicated flip-flop are both 0s. The original functional path of the control point site implies F_L and f_L. Before adding a control point in Fig. 5, lines *L* and *S* are the same net, thus $F_S = F_L, f_S = f_L$ and original conflicts can be calculated as $C_S = \min\{B_S, f_S\}$ and $c_S = \min\{b_S, F_S\}$. According to (6a) and (6b), after adding an AND control point to force the value of 0, F_S of the control point is reduced to 0 and c_S (the "0" conflict) is resolved. Similarly, by adding an OR control point we can reduce f_S as well as C_S (the "1" conflict) of the control point to 0.

To minimize the area taken up by control points, functional flip-flops can be used to replace dedicated flip-flops acting as drivers of control points. As shown in Fig. 6, control points are now connected to existing functional flip-flops instead of adding new scan cells. This may significantly reduce the area required for each control point. However, functional flip-flops may already have fan-outs. According to formulas (1) – (4), forward-implied metrics (F and f values) of fan-out branches are determined by backward-justified (B and b) values occurring on other fan-out branches. Compared to dedicated flip-flop drivers where F = 0 and f = 0, the values of the same metrics for functional flip-flop drivers are determined by values of B

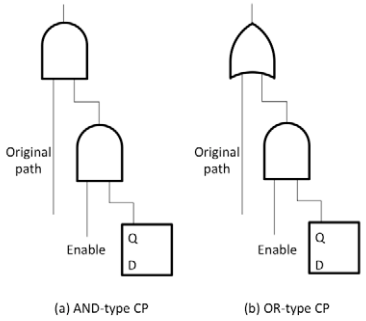

Figure 4 Control point templates

and b coming from native fan-out branches, and they are not equal to 0 any more. In such a case, reusing functional flip-flops as control point drivers may not reduce the conflict degree to 0, and, worse, it may introduce new conflicts at a connection interfacing a control point with origi-

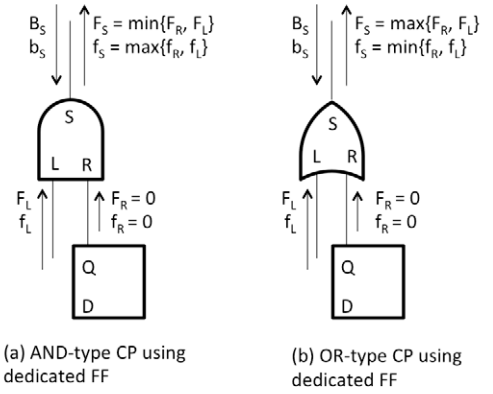

Figure 5 Conflict analysis for dedicated flip-flops

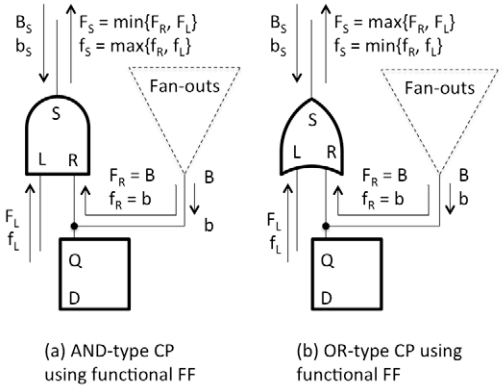

Figure 6 Control point reusing functional flip-flop

nal 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

trade-off silicon area and test pattern counts by selecting appropriate driver candidates through a conflict analysis.

Fig. 6 illustrates how conflict metrics can be computed using formulas (1) – (4). We get that $F_R = B$ and $f_R = b$. Thus, according to (6a), $F_s = \min\{F_L, B\}$ and $f_s = \max\{f_L, b\}$. From (8) and (9) it follows that for the AND type control point the degree of new conflict at the control point is equal to $c_s = \min\{b_s, \min\{F_L, B\}\}$ and $C_s = \min\{B_s, \max\{f_L, b\}\}$. Conflicts due to a new flip-flop connection are $c_R = \min\{b_R, B\}$ and $C_R = \min\{B_R, b\}$. Similar results can be derived for the OR type control point. To maintain the control point's functionality, i.e., to reduce the 1/0 conflict corresponding to a control point, and to avoid introducing large conflicts, we propose to select the functional flip-flops with a minimal sum of conflicts between its own F and f values yielded by backward justifications of other fan-out branches with the control point's B and b values. In this case, we can reduce the original large conflict (conflict 0 for the AND type control point or conflict 1 for the OR type control point) as well as keep the other conflicts low. We pick a proper candidate from a set of functional flip-flops that are "logically" close to the control point bounded by neighborhood criteria detailed in Section 5. Moreover, since different circuits may have different conflict degrees, a user-defined threshold is employed to guide a searching algorithm. Candidates with a large sum of conflicts exceeding this predefined threshold are not considered, and dedicated flip-flops are used instead. By varying the threshold, it is possible to trade-off the number of functional flip-flops versus the number of dedicated flip-flops.

4. Reuse of flip-flops - ATPG verification

When reusing a functional flip-flop as a control point driver, newly added connections may form a reconvergent fan-out with the flip-flop output branches. We need, therefore, to verify whether connecting functional flip-flops with control points compromise fault propagation or not. To avoid test coverage loss, flip-flops failing this verification procedure should not be employed as drivers of control points.

During test application, control points work in two modes: a control mode to force its value, and a propagation mode to allow faults to pass. In this section, we will discuss an ATPG-based verification for these two modes.

Control points having their own dedicated flip-flops do not experience a reconvergence problem when in the control mode. As shown in Fig. 7(a), where an AND type control point is used as an example, a dedicated flip-flop sets gate CP to 0 in the control mode, while a functional flip-flop may have a different assignment. Contrary to a control point driven by its dedicated flip-flop, a functional flip-flop acting as a driver may feature additional connections, as shown in Fig. 7(b). In this case, the new connection added for the purpose of enabling the control point may have different backward-justified ATPG-produced value than those of its remaining fan-out branches. As discussed in Section 3, incompatible assignments made to a flip-flop are quantified as conflicts, and they are already considered during the conflict analysis discussed in Section 3. Such inconsistency will only lead to the increase in pattern count with no coverage loss. Thus, it is not necessary to perform an ATPG-based verification for the reconvergence issue in the control mode.

It is not the case, however, when it comes to the propagation mode. As shown in Fig. 8, the driver is set to the non-controlling value so that faults from the other input can pass through the control point. Control points with a driver feeding a reconvergent fan-out may block faults whose only propagation path goes through the control point. Consider the AND-type control point of Fig. 8. In the propagation mode, the driver is set to 1 to let faults propagate through the control point's original path. As shown in the figure, faults whose only propagation path goes through

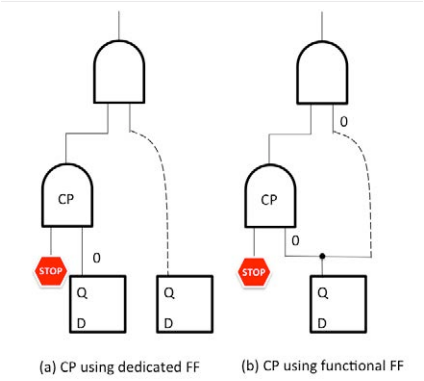

Figure 7 Control mode

the control point can be blocked by the value that sets the top OR gate to its dominating value. In this case, a functional flip-flop that precludes faults from further propagation cannot be considered as a driver candidate. Consequently, additional ATPG verification is required to check whether the candidate flip-flop is blocking any forward fault propagation through the corresponding control point

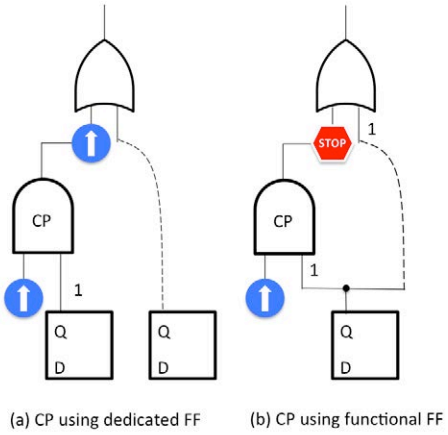

Figure 8 Propagation mode

in the propagation mode.

After running the conflict analysis of Section 3, the ATPG-based verification is performed by imposing the non-controlling value on a candidate functional flip-flop. It checks whether this implied value blocks faults whose only propagation path goes through the control point. Clearly, candidate flip-flops passing ATPG verification will not cause any coverage loss due to the reconvergence problem.

5. Driver candidates search flow

As discussed in Section 3, selection of functional flip-flops to act as control point drivers is done within a certain (and limited) search space. For pre-layout designs, to avoid long paths from selected functional flip-flops to the control points, we select a suitable candidate among flip-flops, which are "logically" adjacent to the control point, while a user-defined threshold limits the total number of flip-flops checked for each control point.

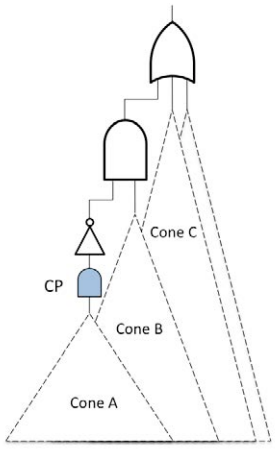

Figure 9 Cone tracing during locality analysis to find extra candidate flip-flops

Fig. 9 illustrates a search for the most appropriate flip-flop among a limited number of functional flip-flops "near" a given control point CP. Starting with the cone of logic that drives the control point, a breadth-first search is applied to find a limited number of candidate flip-flops within the cone. If this cone does not contain enough flip-flops, we gradually enlarge the search space by tracing forward from the control point and then tracing backward from any further off-path inputs to find more flip-flops. For example (see Fig. 9), we first check flip-flops inside cone A. Next, we may seek more flip-flops by visiting cone B, and then cone C, until reaching the threshold.

The combined search flow to select a suitable functional flip-flop to be used as a driver of a single control point can be summarized as follows (compare Fig. 10):

1. Start searching for a candidate flip-flop with minimal conflicts' sum within a predefined conflict threshold.
2. Gradually enlarge the search space to find more flip-flops through backward tracing from any off-path input of the forward path of the control point.
3. Perform ATPG verification to check fault blocking using a non-controlling value of the control point.
4. Repeat steps 1 – 3 until a predefined number of flip-flops are reached.
5. If no candidate is found, use a dedicated flip-flop to drive the control point.

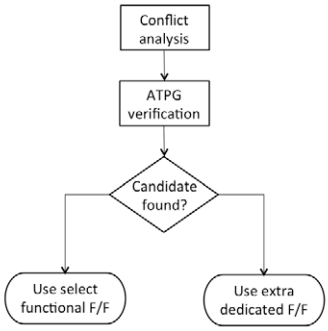

Figure 10 Combined search flow to reuse functional flip-flops for control points

It is worth noting that the above algorithm does not employ a few additional factors that might be also used to guide the search process. They include clock domains (to avoid introducing cross-domain paths) and power domains, design hierarchy and multi-cycle/false paths, as well as floor plan information to either determine or exclude TP drivers based on Euclidean distance between functional flip-flops and nodes where test points need to be inserted. As can be observed, the proposed technique can be easily adopted to work with these factors as well.

6. Experimental results

We have conducted experiments on 11 large industrial designs with on-chip embedded deterministic test (EDT) compression logic [14]. Table 1 highlights their characteristics, including the number of gates, the number of scan cells, the number of scan chains, the number of EDT input and output channels, as well as the baseline test coverage obtained with no test points involved. By setting the con-

Table 1. Circuit characteristics

	Gates	Scan cells	Scan chains	EDT channels	TC [%]
D1	1.19M	72K	400	4, 4	96.95
D2	103K	1K	10	1, 1	97.78
D3	218K	14K	20	1, 1	99.99
D4	2.08M	143K	400	4, 4	99.51
D5	1.32M	52K	220	6, 6	98.13
D6	1.04M	57K	400	4, 4	91.39
D7	3.34M	325K	400	4, 4	96.49
D8	2.60M	154K	1200	12, 12	91.34
D9	1.69M	86K	400	4, 4	97.82
D10	2.31M	252K	490	10, 10	99.06
D11	1.47M	162K	330	15, 15	99.05

Table 2. Reuse of functional flip-flops – ATPG results

	TC [%]	CPs	OPs	Reuse ratio [%]	PC	Pattern reduction
D1	96.97	1,364	1,636	99.7	1,628	5.31x
D2	98.78	300	300	90.0	4,644	2,21x
D3	100.00	379	701	91.3	4,502	2.43x
D4	99.55	750	750	95.0	9,021	2.66x
D5	99.42	260	260	94.2	24,763	1.33x
D6	92.11	600	600	97.2	8,185	1.87x
D7	96.52	2,973	3,027	90.4	2,590	1.36x
D8	91.66	770	770	100.0	6,585	3.02x
D9	97.99	944	1,056	95.2	9,397	1.66x
D10	99.70	1,500	1,500	91.3	2,084	2.11x
D11	99.66	3,000	3,000	93.6	3,552	1.38x

flict threshold to a design-specific value, we can achieve a large fraction of control points working with functional flip-flops as their drivers. Thus, by having a small number of dedicated scan cells to drive control points, we are minimizing silicon area occupied by test points. Functional flip-flops deployed as drivers of control points are selected within their "logical" proximity comprising 100 flip-flops, as described in Section 5.

Table 3. Pattern counts for dedicated (D) and functional (F) drivers

	Pattern count		PC increase [%]	Pattern reduction	
	D	F		D	F
D1	1,606	1,628	1.4	5.38x	5.31x
D2	4,292	4,644	8.2	2.39x	2.21x
D3	4,116	4,502	9.4	2.66x	2.43x
D4	8,772	9,021	2.8	2.73x	2.66x
D5	24,324	24,763	1.8	1.35x	1.33x
D6	7,625	8,185	7.3	2.01x	1.87x
D7	2,495	2,590	3.8	1.41x	1.36x
D8	5,812	6,585	13.3	3.41x	3.02x
D9	9,142	9,397	2.8	1.71x	1.66x
D10	2,020	2,084	3.2	2.17x	2.11x
D11	3,217	3,552	10.4	1.52x	1.38x

We ran ATPG for 11 netlists with test points inserted by means of the method presented in [1] and by using functional flip-flops as drivers of control points, as proposed in this work. Although our focus is on showing how pattern count reduction depends on sharing functional flip-flops with control points, observation points are also added to each design by following the rules presented in [1]. Consequently, Table 2 provides test coverage (column TC), the total number of control points (CPs), the total number of observation points (OPs), the reuse ratio defined as the percentage of control points using functional flip-flops as drivers, the pattern count (PC) and the pattern count reduction ratio compared to the baseline designs without control points. For all test cases, at least 90% reuse ratio is achieved. Thus, for more than 90% of control points, we can minimize the area without adding extra scan cells during test point insertion. As shown in the last column, on the average, we can get a 2 – 3 times pattern count reduction compared to the baseline designs having no control points.

Compared to the control points using fully dedicated flip-flops, reusing functional flip-flops reduces the total number of scan cells and minimizes the area required by the control points. As presented in Section 3, using existing functional flip-flops may result in additional conflicts. Hence, replacing dedicated flip-flops by functional flip-flops may increase the total pattern count. Table 3 shows a comparison that addresses this problem. Here, the pattern counts whilst using dedicated (functional) flip-flops are shown under D (F) in columns 2 and 3, and the percentage increase in a pattern count when functional flip-flops are used is given in the next column. In the last two columns we give the factor by which the pattern counts are reduced relative to the pattern counts for designs not using test points. The results indicate that with an average 5% increase in the pattern count, conflict-aware control points with functional flip-flops as their drivers can still achieve a significant compression during deterministic test pattern generation.

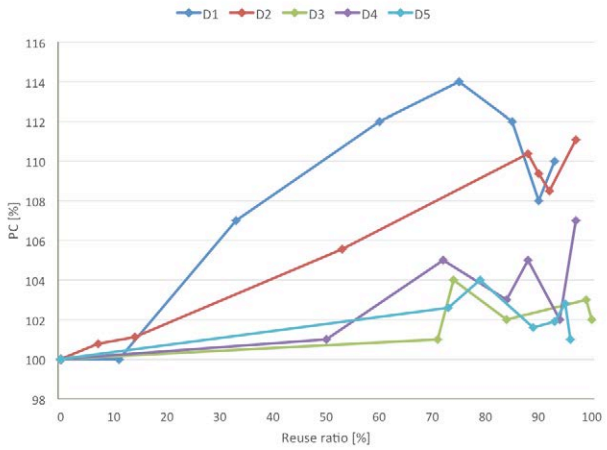

Figure 11 Pattern count vs. reuse ratio

Fig. 11 shows a relationship between a pattern count and different fractions of reused functional flip-flops for five test cases. The pattern counts obtained by reusing functional flip-flops are normalized with respect to the results for dedicated flip-flops. As discussed earlier, replacing dedicated flip-flops with functional flip-flops may introduce additional conflicts. As a result, a higher reuse ratio is expected to feature a higher pattern count. However, as shown in the figure, for some reuse ratios the pattern count is actually dropping. As of now, there is no satisfactory explanation of this phenomenon yet.

7. Conclusion

In this paper, we present a method to select existing functional flip-flops as drivers for the conflict-aware test points. This approach allows one to minimize the silicon

area required by control point-based test logic. Experimental results on industrial designs show that a 90% reuse ratio can be achieved under certain criteria. With an average of 5% increase in pattern counts (compared to fully dedicated flip-flops), the conflict-aware test points reusing functional flip-flops can still achieve a significant reduction in pattern count compared to baseline designs without any control points.

8. References

[1] C. Acero, D. Feltham, F. Hapke, E. Moghaddam, N. Mukherjee, V. Neerkundar, M. Patyra, J. Rajski, J. Tyszer, J. Zawada, "Embeded deterministic test points for compact cell-aware tests," *Proc. ITC*, 2015, paper 2.2.

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

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

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

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

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

[7] 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," *Proc. ITC*, 2002, pp. 138-147.

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

[9] V.S. Iyengar and D. Brand, "Synthesis of pseudorandom pattern testable designs," *Proc. ITC*, 1989, pp. 501-508.

[10] R. Kapur, S. Mitra, and T.W. Williams, "Historical perspective on scan compression," *IEEE Design & Test of Computers*, vol. 25, No. 2, pp. 114-120, 2008.

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

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

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

[14] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Embedded deterministic test," *IEEE Trans. CAD*, vol. 23, pp. 776-792, 2004.

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

[16] 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," *Proc. DFTVS*, 2008, pp. 385-393.

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

[18] 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," *Proc. ATS*, 2006, pp. 339-348.

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

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

[21] N.A. Touba, "Survey of test vector compression techniques," *IEEE Design & Test of Computers*, vol. 23, No. 4, pp. 294-303, 2006.

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

[23] J.S. Yang, N.A. Touba, and B. Nadeau-Dostie, "Test point insertion with control points driven by existing functional flip-flops," *IEEE Trans. Computers*, vol. 61, No. 10, pp. 1473-1483, 2012.

[24] M. Yoshimura, T. Hosokawa, and M. Ohta, "A test point insertion method to reduce the number of test patterns," *Proc. ATS*, 2002, pp. 298-304.