Test Point Insertion Using Functional Flip-Flops to Drive Control Points

Joon-Sung Yang[1], Benoit Nadeau-Dostie[2], and Nur A. Touba[1]

[1]Computer Engineering Research Center
Dept. of Electrical and Computer Engineering
University of Texas, Austin, TX 78712
{jsyang,touba}@ece.utexas.edu

[2]Logic Vision, Inc.
25 Metro Drive, Third Floor
San Jose, CA 95110
benoit@logicvision.com

Abstract

This paper presents a novel method for reducing the area overhead introduced by test point insertion. Test point locations are calculated as usual using a commercial tool. However, the proposed method uses functional flip-flops to drive control test points instead of test-dedicated flip-flops. Logic cone analysis that considers the distance and path inversion parity from candidate functional flip-flops to each control point is used to select an appropriate functional flip-flop to drive the control point which avoids adding additional timing constraints. Reconvergence is also checked to avoid degrading the testability. Experimental results indicate that the proposed method significantly reduces test point area overhead and achieves essentially the same fault coverage as the implementations using dedicated flip-flops driving the control points.

1. Introduction

Built-in self-test (BIST) involves the use of on-chip test pattern generation and output response analysis. BIST provides a number of important advantages including the ability to apply a large number of test patterns in a short period of time, high coverage of non-modeled faults, minimal tester storage requirements, can apply tests out in the field over the lifetime of the part, and a reusable test solution for embedded cores. The most efficient logic BIST techniques are based on pseudo-random pattern testing. A major challenge is the presence of *random-pattern-resistant (r.p.r.) faults* which have low detection probabilities and hence may limit the fault coverage that can be achieved with pseudo-random patterns. There are two approaches for detecting r.p.r. faults: either modify the pattern generator so that it generates patterns that detect them, or modify the circuit-under-test to eliminate the r.p.r. faults.

A number of techniques have been developed for modifying the pattern generator. These include weighted pattern testing [Schnurmann 75], [Wunderlich 87], [Pomeranz 92], [Bershteyn 93], [Kapur 94], [Jas 01], [Lai 05], pattern mapping [Chatterjee 95], [Touba 95a, 95b], bit-fixing [Touba 96], bit-flipping [Wunderlich 96], and LFSR reseeding [Konemann 91, 01], [Hellebrand 92, 95], [Krishna 01], [Rajski 02].

The other approach is to modify the circuit-under-test (CUT) by inserting test points [Eichelberger 83]. Test points are very efficient for eliminating r.p.r. faults and improving the fault coverage. Test point insertion (TPI) involves adding control and observation points to the CUT. Observation points involve making a node observable by making it a primary output or sampling it in a scan cell. Control points involve ANDing or ORing a node with an activation signal as illustrated in Fig. 1. When the activation signal is a '1', it controls the node to a 0 (1) for a control-0 (control-1) point. Typically the activation signal is driven by a dedicated flip-flop which receives pseudo-random values during BIST and is set to a non-controlling value during normal operation. Test points are added to the circuit before layout so that the performance impact can be minimized. Circuit re-structuring is routinely used during layout to take into account additional delay due to metal wires.

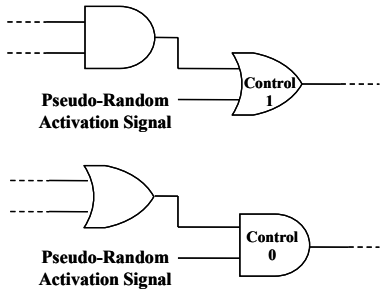


Figure 1. Example of Control Points

Since test points add area and performance overhead, an important issue for test point insertion is where to place the test points in the circuit to maximize the coverage and minimize the number of test points. Optimal placement of test points in circuits with reconvergent fanout has been shown to be NP-complete [Krishnamurthy 87]. A number of approximate techniques for placement of test points have been

978-1-4244-4867-8/09/$25.00 ©2009 IEEE

developed using fault simulation [Briers 86], [Iyengar 89], path tracing [Touba 96], or testability measures [Seiss 91] to guide them. Timing driven test point insertion [Cheng 95], [Tsai 98] avoids inserting control points on critical timing paths.

Some research has investigated more efficient ways to drive the activation signals for the control points. In [Tamarapalli 96], the entire test is partitioned into multiple phases by a divide and conquer method, and control points are activated only during certain phases and deactivated during other phases. This provides greater control over the interaction of the control points with each other which can help reduce the total number of test points required. [Youseff 93] and [Nakao 99] propose methods for having one dedicated flip-flop drive the activation signal for multiple control points, i.e., sharing the dedicated flip-flops among the control points to reduce the total number of dedicated flip-flops that are required.

In spite of the efforts to reduce the overhead for TPI, the International Technology Roadmap for Semiconductors (ITRS) [ITRS 07] predicts that logic BIST for random patterns will take about 3.1% of chip area whereas the area for test compression will vary from 1.1% to 1.7%. One unpublished industrial design evaluation shows that logic BIST adds 1.34% to the chip area of which about 30% is related to the test points (0.4% to the chip area). And the other data from the unpublished industrial design evaluation indicates that 2.68% chip area is increased by logic BIST and test points take 1.16% chip area. This suggests that test points correspond to 43% of the area increase in logic BIST. Test point area may vary depending on the circuit characteristics, the number of pseudo-random patterns used, and the fault coverage required. However, a considerable portion of the BIST area is usually related to test points, so it is important to find new techniques that can reduce the area overhead.

In this paper, a new method for reducing the area impact of test point insertion is proposed by removing the dedicated flip-flops used for driving the control points. As will be shown in Sec. 5, and shown earlier in [Seiss 91], more than half of the test points, inserted are control points. Hence, replacing the dedicated flip-flops used to drive the control points with functional flip-flops in the design can significantly reduce area overhead. In the proposed test point implementation method, the location of the test points can be determined using existing test point insertion techniques such as the one described in [Seiss 91]. The new software identifies functional flip-flops which are suitable to drive the control point. Only functional flip-flops in the fan-in of the control point are considered as candidates to ensure that no new timing constraints are introduced between any two flip-flops. The method inherently introduces reconvergent paths sourced by the candidates flip-flops and has the potential to introduce redundant faults. Redundancies are avoided by taking into account the path inversion parity of the reconvergent paths. The proposed method essentially achieves the same fault coverage as an implementation based on dedicated flip-flops, but with lower area cost. The method is neutral with respect to the handling of unknowns in the circuit and test power as it does not deal with the selection of the test points, only their implementation.

This paper is organized as follows. Sec. 2 gives an overview of the proposed scheme. Sec. 3 describes the control point replacement flow in detail. Sec. 4 discusses a technique to increase the fault coverage. Experimental results are shown in Sec. 5 and conclusions are given in Sec. 6.

2. Overview of Proposed Scheme

As mentioned earlier, area is the main issue for TPI. This section gives an overview of the main idea for significantly reducing the area without losing testability and introducing additional timing constraints.

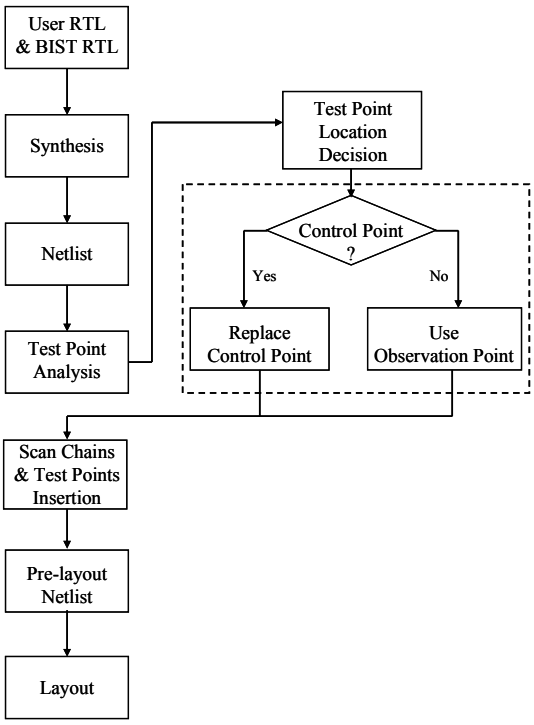


Figure 2. Proposed Design Synthesis Flow with Testability and Area Overhead Minimized Test Point Insertion

Fig. 2 shows a design synthesis flow that incorporates scan, BIST, and test point insertion. The conventional flow ends after test point insertion and generates the final design netlist. When test points are inserted, dedicated flip-flops are assigned to drive the control points and

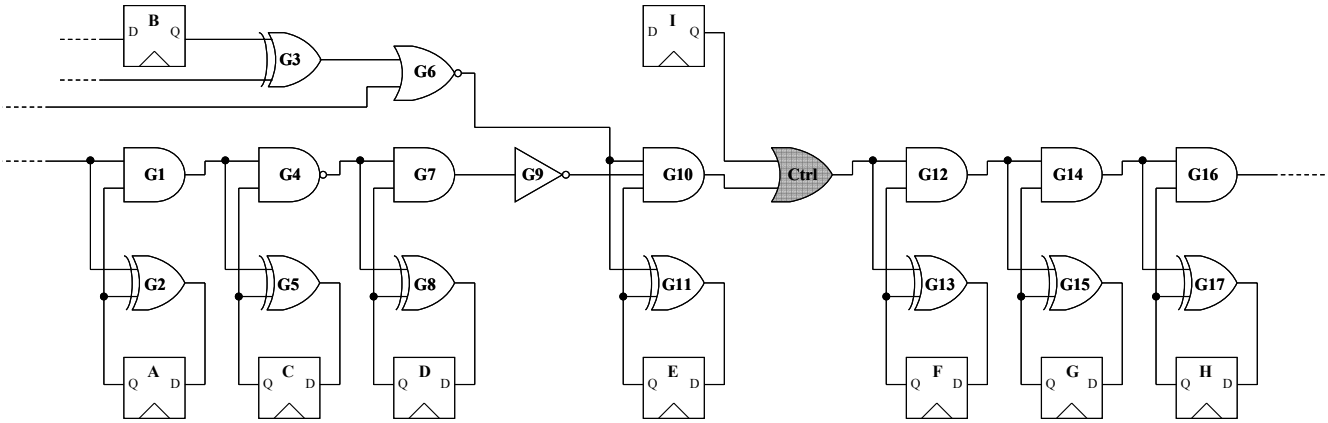


Figure 3. Example of a Circuit with Control Point Insertion (*Ctrl*)

capture the observation points to achieve higher fault coverage. The idea proposed here is to minimize the area overhead by replacing the dedicated flip-flops for driving control points with existing functional flip-flops in the design. The test points are first inserted with any TPI algorithm [Briers 86], [Cheng 95], [Iyengar 89], [Nakao 93], [Seiss 91], [Touba 96], [Youseff 93]. Then the proposed method performs a post-processing step in which functional flip-flops are identified for driving the control points via logic cone analysis. The observation points are not modified. The dashed box in Fig. 2 indicates the post-processing flow that finds and replaces the control points to generate the netlist.

In a BIST application, the activation signal is controlled by a flip-flop scanned in with a pseudo-random value for each scan vector. During system operation, and the activation signal is set to its non-controlling value so that the functional logic value can pass through the control gate. Therefore, when a dedicated flip-flop is replaced by a functional flip-flop, it should keep the same property of not changing the system function. For this purpose, a global signal *"TP_Enable"* is introduced. *TP_Enable* enables and disables the control points. When *TP_Enable* is *'1'*, a control point is driven by a functional flip-flop in the proposed method. Note that this creates a new timing path from the functional flip-flop to the control point.

The functional flip-flops which are "logically" near the control point are chosen as candidates to replace a dedicated control point flip-flop for two reasons. The first reason is to minimize the length of the newly created test path from the candidate flip-flop to the control point. The second reason is that the transitions through the control point will have roughly the same delay as those along the functional path from the selected functional flip-flop. As will be explained in detail in Sec. 3.2, the proposed method does not create any relationships between control points and unrelated registers, and hence no new timing constraints are introduced.

Since the proposed method does add additional primitive gates to a design, it can impact the testability of the design. Hence, the following rules need to be observed to minimize the number of redundant or untested faults introduced by circuit modification.

1. Maintain opposite path inversion parity along the paths from a functional flip-flop to a control point: There are two paths from a functional flip-flop to the control point. One is an existing functional path from a functional flip-flop to a control point and the other is the newly introduced path which is ANDed with the *TP_Enable* signal. Having opposite inversion parity along these two paths makes a path testable by appropriately applying either '*0*' or '*1*'. This is described in Sec. 3.3.1.
2. Check for illegal reconvergence from the candidate functional flip-flop: Hard to test faults could be blocked by a fanout of a test point if the functional flip-flop (which drives the test point) drives some gate in a fanout of a test point. This case needs to be avoided. This is described in Sec. 3.3.2.

3. Details of Control Point Replacement Flow

The following subsections describe each of the steps in the proposed method for replacing the dedicated flip-flops with functional flip-flops to drive the control points.

3.1. Finding Candidate Functional Flip-Flops

Fig. 3 is an example of conventional test point insertion. This circuit has flip-flops (denoted *A* to *I*) and combinational elements (denoted *G1* to *G17* and *Ctrl*). It has one control point highlighted in gray color (*Ctrl*) and a dedicated flip-flop *I* drives the control point in test mode. Flip-flops *A* to *H* are the functional flip-flops that are used for a system operation. During the system operation, *Ctrl* is made transparent by resetting flip-flop *I* so that the

value in *G10* can be transferred to the one input of *G12* without any change. And when the control point is activated, the output of gate *Ctrl* is fixed to a '*1*' (i.e., *control-1 point*). If an AND gate is used as *Ctrl*, the output of *Ctrl* is fixed to a *'0'* when it is activated (i.e., *control-0 point*).

To find the functional flip-flop for replacing the dedicated flip-flop, it is necessary to perform logic cone analysis. Logic cone analysis starts from the control point (*Ctrl*) and traces back to the flip-flops. In Fig. 3, the search initiates from *Ctrl*. Even though *I* is the first flip-flop found, since it is dedicated for the control point, *I* needs to be dropped from the search space. Hence, *G10* is the first gate visited. In a depth first search manner (a breadth first search can also be used), *G6* is visited next and then *G3* is visited. Flip-flop *B* is found along this branch from *Ctrl*. In the same fashion, flip-flop *A*, *B*, *C*, *D,* and *E* are found as candidates for replacing *I*.

While the nodes are traversed when searching, the inversion parity information is also checked. *G4*, *G6* and *G9* are inverting gates. Since *G6* is inverting, the flip-flop *B* has odd inversion parity to the control point. Flip-flop *A*, *C* and *E* have even inversion parity, and *B* and *D* have odd parity along their paths to the control point. If both inversion parities are found along paths from the control point to one flip-flop, that flip-flop is discarded from the candidate list. For example, if during logic cone analysis, a flip-flop is found to have one path with one inversion, and another path with two inversions, it is not considered as a candidate. Some elements such as XOR gates and MUXes always have both non-inverting and inverting paths (dual polarity). Gates with dual polarity are considered as non-inverting gates. Further analysis on dual polarity will be discussed in Sec. 5.

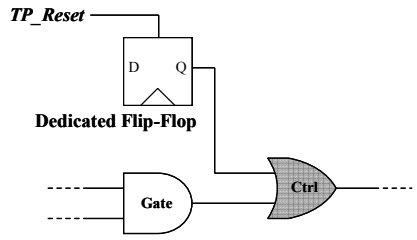
As shown in Sec. 2, a new timing path is a matter of concern in selecting a functional flip-flop to replace a dedicated flip-flop. Therefore, logical distance information needs to be considered so as not to introduce any delay paths that add performance overhead. The functional flip-flop distance to a control point can be measured based on the number of levels of logic. The logical distance is used to maximize the probability for the test point driver to be relatively close to the test point to minimize the length of the wires. The following shows the results of logic cone analysis for Fig. 3.

Candidate Flip-Flop	*Inversions*	*Logical Distance*
A	*2*	*5*
B	*1*	*3*
C	*2*	*4*
D	*1*	*3*
E	*0*	*2*

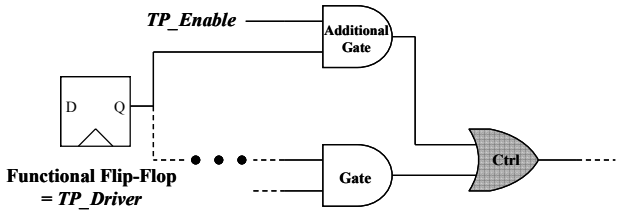
There may be cases when only one functional flip-flop is found as a candidate by logic cone analysis. This occurs when a test point has a single functional flip-flop in its fan-in. This happens when the controllability to a certain value (*'0'* or *'1'*) needs to be higher than 0.5. In this case, a dedicated flip-flop cannot be replaced.

3.2. Selecting Candidate Flip-Flop

Assume that a dedicated flip-flop is replaced by one of the functional flip-flops among *A* to *E*. If a functional flip-flop directly drives the control point, it affects the system function. To hold the transparency property during system operation, one global signal called "*TP_Enable*" is introduced and it is deactivated during system operation. Fig. 4(a) shows an example of a conventional control point that uses a dedicated flip-flop. Fig. 4(b) illustrates an example of the proposed control point that is driven by a functional flip-flop. The functional flip-flop not only drives the AND gate at the bottom but also operates as a test point driver via the additional gate path in Fig. 4(b) (this flip-flop can be named as *TP_Driver*). The *TP_Enable* signal can block the signal propagation by setting its value to '*0*'. This places a non-controlling value at the input of the control point. When *TP_Enable* is '*1*', *Ctrl* can have a value determined by a functional flip-flop.



(a) Conventional Control Point with Dedicated Flip-Flop



(b) Example of Proposed Control Point with Functional Flip-Flop

Figure 4. Conventional and Proposed Control Point

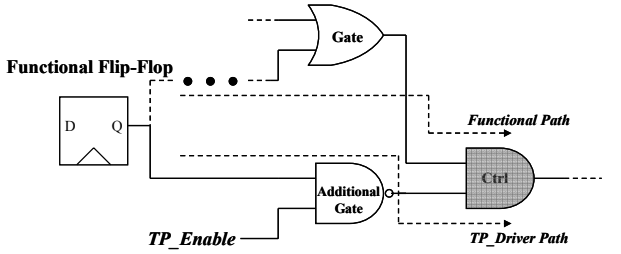
In Sec. 3.1, *E* is found to be the closest flip-flop to the control point location with a distance *'2'* and it is now chosen as the *TP_Driver*. Since flip-flop *E* is already in the fan-in of the control point, no new timing relationships are created with flip-flops in the fan-out of the test point. Note that from a testability point of view, it would have been preferable to use a flip-flop that is not in the fan-in of

the control point to avoid the correlation between the two inputs of the control point. However, new timing relationships would be created between functionally unrelated flip-flops and this is not acceptable. The consequences of the "no new timing relationships" rule are analyzed next.

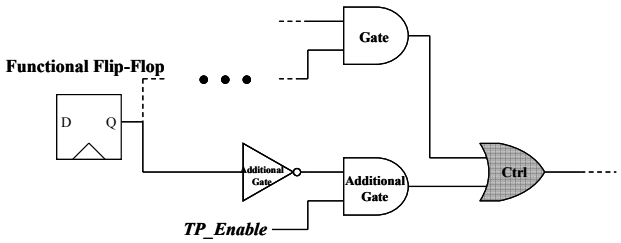
3.3. Testability Consideration

The proposed method modifies the CUT to try to maximize the random pattern testability. The following subsections describe the rules that need to be considered for improving testability.

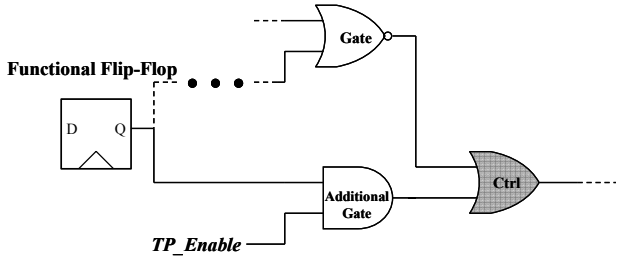
3.3.1 Path Inversion and Control Point Structures



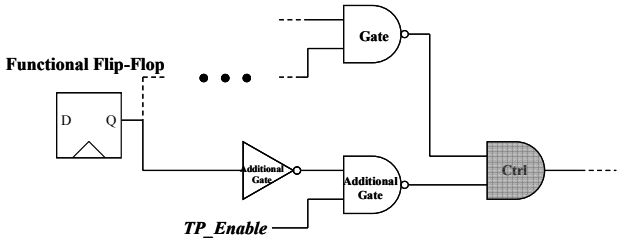
(a) *Type 1*: Non-Inverting Functional Path with AND Ctrl



(b) *Type 2*: Non-Inverting Functional Path with OR Ctrl



(c) *Type 3*: Inverting Functional Path with OR Ctrl



(d) *Type 4*: Inverting Functional Path with AND Ctrl

Figure 5. New Types of Control Point Structure for Different Path Inversion Parity

If a functional flip-flop is chosen to replace a dedicated flip-flop for the control point, a new path is created from the functional flip-flop to the control point. This new path will be referred to as the *"TP_Driver Path"*. The original functional path will be referred to as the *"Functional Path"*. A value in *Functional Path* can only propagate when *TP_Enable* is disabled in Fig. 4(b). However, the opposite inversion parity between the *TP_Driver Path* and *Functional Path* can enable propagation through *Functional Path* without disabling the test point. This increases the random pattern testability and helps to reduce the number of test patterns needed compared to having the same inversion parity along the two paths. Considering that either an AND or OR gate can be used for creating a control point, there are 4 types of control points that satisfy the inversion parity as shown in Fig. 5.

In Fig. 5(a) and 5(b), the *TP_Enable Path* needs to have inversion because *Functional Path* has a non-inverting path. Fig. 5(c) and 5(d) show the control point with an inverting path on *Functional Path*. When an inverter is added in the *TP_Driver Path* as in Fig. 5(b) and 5(d), either an inverter can be used or the flip-flop's *Q_bar* can be connected to the additional gate.

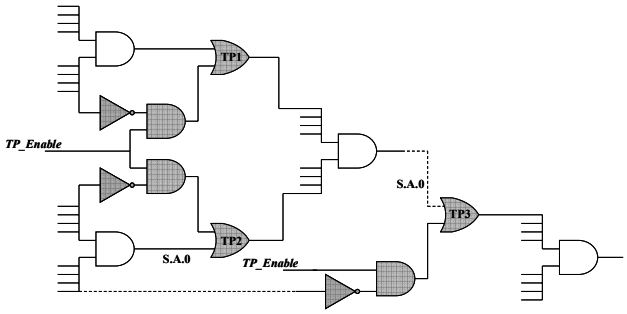


Figure 6. AND Tree Example with Proposed Control Point Structure

Fig. 6 shows an AND tree example with the proposed control point structure. Assume the path inversion is not considered, no inverter would be inserted on the *TP_Driver Path* when a dedicated control point driver flip-flop is replaced, as in Fig. 4(b). In this case, hard to test faults have a small probability of being able to propagate through the circuit if all test points are disabled (*TP_Enable = 0*). For example, the stuck-at-0 fault (S-A-0) at the input of TP2 is one of the hard to detect faults, and it would preferably need TP1 to be active in order to propagate S-A-0 through the AND gate which is located after TP1 and TP2, and then to propagate through TP3 and the remainder of the circuit. However, when path inversion is considered, the control point structure will solve this problem. S-A-0 at the input of TP2 could propagate even when *TP_Enable* is '*1*'. All inputs of the AND gate should be '*1*' to provoke S-A-0 at TP2, and it automatically sets the AND gate with a controlling value

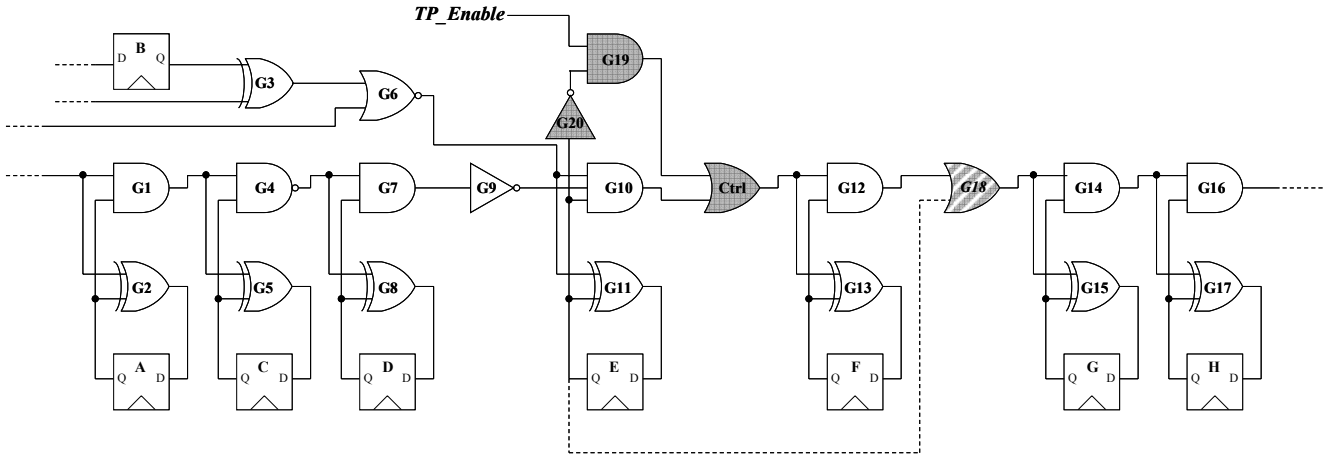


Figure 7. Example of a Circuit with Illegal Reconvergence

(*0*). This makes the control point disabled without setting *TP_Enable* to be *'0'*. *TP_Enable* reconvergence could be an issue in the proposed method, however, the path inversion analysis solves this problem so that the testability is not degraded. Hence, they are detected relatively easier than not having the path inversion information. The other hard to detect faults in Fig.6 are the S-A-1 faults on the *TP_Enable* input of the AND gates disabling the test points. Since it is unlikely to randomly detect the faults on *TP_Enable* branches, automatic test pattern generation (ATPG) patterns need to be used for detecting most of these faults. However, they represent a very small percentage of the total number of faults as we will see in Sec. 5.

3.3.2 Illegal Reconvergence

Logic cone analysis determines the functional flip-flop candidates for driving control test points. For testability, since there may be many connections from the functional flip-flops to other nodes in a circuit, it is necessary to check whether the fault propagation is blocked. Reconvergence from *TP_Driver* in the fanout of a control point can block the fault propagation. If any gate in the fanout of a control point is sourced by *TP_Driver*, it can obstruct fault propagation and it may result in the loss of testability.

Fig. 7 illustrates an example which is almost the same as a circuit in Fig. 2 with the exception of OR gate *G18*. As shown in the previous sections, *E* is selected as a *TP_Driver* based on the distance and a *Type 2* control point is inserted to satisfy the inversion parity requirement. Assume *E* has a branch to *G18* illustrated as a dashed line from *E* to *G18*. This forms reconvergence from *E* to a fanout of a control point. Due to the reconvergence, whenever *E* has *'1'*, it drives *G18* with a controlling value and this may block the fault propagation.

Illegal reconvergence analysis removes *E* from the candidate list, and the next closet flip-flop is chosen. In Fig. 7, *B* and *D* have a distance 3 and they do not introduce a longer timing path than the existing longest path. There is no reconvergence from *B* or *D* to the fanout of the control point, so they do not violate the illegal reconvergence condition. Since *B* and *D* have the appropriate inversion parity and an OR control point is used, a *Type 3* control point needs to be applied when replacing the dedicated flip-flop. If there is no flip-flop that satisfies the conditions for replacement, a dedicated flip-flop cannot be replaced.

4. *TP_Enable* Signal Probability

Test points are generally assumed to be always enabled with a controllability of 0.5. Therefore, *TP_Enable* will generally have a value of '1'. However, the stuck-at-1 fault on *TP_Enable* can only be detected when the *TP_Enable* signal is set to '0'. To detect this fault, *TP_Enable* needs to take on a value of '0' some times. To investigate what the optimal signal probability for *TP_Enable* is, experiments were performed using different input size OR gates to bias the signal probability of *TP_Enable.* If two equi-probable pseudo-random signals are ORed together, the signal probability is increased to 0.75. In the general case, driving the *TP_Enable* signal by a *k input* OR gate achieves a $(2^k - 1)/(2^k)$ signal probability.

Different *TP_Enable* signal probabilities change the controllability on the control points and detectability of the stuck-at-1 fault on the *TP_Enable* signal. In Sec. 5, experimental results are shown for different signal probabilities for the *TP_Enable* signal.

5. Experimental Results

In this section, experimental results are presented for six industrial designs and *OR1200* (OpenRisc Processor) [OR1200] and an *NOC* design [Yang 09]. The LogicVision testpointAnalyze tool [LogicVision] was used to determine the location of test points in each design.

Table 1. Area Overhead Reduction Results

Design	Conventional Test Point Insertion		Proposed Control Point Replacement Method		Reduction Ratio (%)	Test Point Area Reduction (%)
	Num. Observation Points	Num. Control Points	Num. Dedicated Flip-Flops	Num. Functional Flip-Flops		
Design A	3	24	2	22	91.7 %	61.1 %
Design B	2	85	2	83	97.7 %	71.6 %
Design C	104	233	29	204	87.6 %	45.4 %
Design D	70	179	39	140	78.2 %	42.2 %
Design E	221	1424	794	630	44.2 %	28.7 %
Design F	129	371	13	358	96.5 %	53.7 %
OR1200	5	27	0	27	100 %	63.0 %
NOC	9	35	0	35	100 %	59.4 %

The post-processing tool described here was used to determine the functional flip-flops that could be used as test point drivers. In Table 1, the number of dedicated flip-flops that are replaced by functional flip-flops using the proposed method is shown. The first column gives the design name. The number of observation points and control points calculated is shown in the second and third column, and the summation of both columns is the total number of test points. The fifth column shows the number of dedicated flip-flops that are replaced by functional flip-flops using the proposed method. As explained in Sec. 3, if there is only one functional flip-flop in the candidate list or no candidate meets the rules, a dedicated flip-flop cannot be replaced. The number of dedicated flip-flops which could not be replaced is shown in the fourth column. The sixth column shows the reduction ratio which is computed as the number of functional flip-flops used to replace dedicated flip-flops (the fifth column) divided by the number of total control points (the third column). These results show a significant area reduction by replacing the dedicated flip-flop using the proposed method. The last column represents the test point area reduction ratio. 130nm TSMC technology is used for OR1200 and NOC synthesis. The proposed method achieves 63.0% and 59.4% of test point area reduction in OR1200 and NOC respectively, when the dedicated flip-flops are replaced by functional flip-flops. Because we do not have access to the netlist for Design *A* – *F*, their results are extrapolated based on the results from OR1200 and NOC. In OR1200 and NOC, each of the new control points driven by a functional flip-flop takes approximately 1/4 of the area of the original control points driven with a dedicated flip-flop. Therefore, the extrapolated area for Design *A* – *F* can be calculated based on the following equation.

$$\frac{New\ Area}{Old\ Area} = \frac{Nobs + Ndedicated + k * Nfunctional}{Nobs + Ndedicated + Nfunctional} = 1 - Area\_reduction$$

where *Nobs* denotes the number of flip-flops for observation points, and *Ndedicated* and *Nfunctional* indicate the number of dedicated flip-flops and functional flip-flops used for control point respectively. Since the *k* factor is approximately 0.25 for both OR1200 and NOC, we calculate the area reduction of Design *A* – *F*.

In Table 2, a fault coverage comparison is shown between the proposed test point implementation method and the standard LogicVision implementation. The number of test points inserted and the number of control points replaced are given in Table 1. The first column of upper and lower tables in Table 2 gives the design name. The total number of faults is illustrated in the second column of the upper table. There are three different cases for which results were generated. These were when no test points are inserted (*NO TP*), test points are inserted with dedicated flip-flops (*Dedicated F/F*), and when the proposed method is used to replace dedicated flip-flops with functional flip-flops (*Proposed*). The proposed method was tried with three different *TP_Enable* signal probabilities (1/2, 15/16, and 63/64) to evaluate the random pattern testability. Since TPI adds extra gates in a design, more faults exist for the *Dedicated F/F* and *Proposed* cases than for *No TP*. When a dedicated flip-flop is replaced, the *type 1-4* control point structures in Fig. 5 are used. These structures add a few combinational gates in a design. Since the fault simulator does not consider internal faults of flip-flops, it appears that the proposed method has more faults than the standard implementation even though there are less. The third column of the upper table shows the number of redundant faults, and the second column of the lower table represents the number of aborted faults. Because Designs *A* to *F* are random pattern resistant circuits, 100000 random patterns are applied to get the fault coverage in the third column of the lower table. Since the OR1200 and NOC designs are found to be relatively random pattern testable circuits, 2048 random test patterns are applied and the coverage is shown. The coverage results show that the proposed method achieves almost the same or higher fault coverage as the conventional TPI method does.

Table 2. Testability Comparison of Proposed Method with Standard Implementation

Design	Num. of Faults				Num. of Redundant Faults			
	No TP	Dedicated F/F	Proposed	TP_Enable probability	No TP	Dedicated F/F	Proposed	TP_Enable probability
Design A	92631	92689	92726	1/2	186	186	186	1/2
			92739	15/16			186	15/16
			92741	63/64			186	63/64
Design B	20955	21131	21294	1/2	32	19	20	1/2
			21297	15/16			20	15/16
			21299	63/64			20	63/64
Design C	505254	506120	506457	1/2	2730	2559	2559	1/2
			506460	15/16			2559	15/16
			506462	63/64			2559	63/64
Design D	245688	246294	246474	1/2	1813	1690	1680	1/2
			246477	15/16			1683	15/16
			246479	63/64			1683	63/64
Design E	1665662	1299661	1300549	1/2	3029	2135	2136	1/2
			1300552	15/16			2134	15/16
			1300554	63/64			2134	63/64
Design F	500405	381139	381599	1/2	2706	1608	1647	1/2
			381602	15/16			1649	15/16
			381604	63/64			1649	63/64
OR1200	98690	98906	98984	1/2	2	0	0	1/2
			98987	15/16			0	15/16
			98989	63/64			0	63/64
NOC	56277	56383	56455	1/2	4	4	4	1/2
			56458	15/16			4	15/16
			56460	63/64			4	63/64

Design	Num. of Aborted Faults				Fault Coverage (%)			
	No TP	Dedicated F/F	Proposed	TP_Enable probability	No TP	Dedicated F/F	Proposed	TP_Enable probability
Design A	5	5	5	1/2	88.19	93.35	91.99	1/2
			5	15/16			93.28	15/16
			5	63/64			93.30	63/64
Design B	0	0	0	1/2	89.08	94.37	94.32	1/2
			0	15/16			94.82	15/16
			0	63/64			94.80	63/64
Design C	2738	601	600	1/2	95.68	98.90	98.84	1/2
			600	15/16			98.84	15/16
			600	63/64			98.83	63/64
Design D	936	683	709	1/2	95.05	98.71	98.61	1/2
			683	15/16			98.63	15/16
			690	63/64			98.61	63/64
Design E	1028	630	679	1/2	80.19	99.26	99.14	1/2
			681	15/16			99.23	15/16
			682	63/64			99.15	63/64
Design F	115	113	102	1/2	87.16	98.25	97.71	1/2
			114	15/16			97.86	15/16
			114	63/64			97.77	63/64
OR1200	15	22	28	1/2	93.18	98.96	98.44	1/2
			28	15/16			98.86	15/16
			30	63/64			98.86	63/64
NOC	214	0	13	1/2	97.78	99.41	99.40	1/2
			11	15/16			99.42	15/16
			11	63/64			99.42	63/64

The small fault coverage difference, about 0.05% ~ 0.1% from most of the benchmark circuits, between *Dedicated F/F* and *Proposed* with 15/16 signal probability essentially corresponds to the number of faults added by the new test points that can only be detected when *TP_Enable* is *'0'*. Those faults, the faults on the *TP_Enable* branches, are very difficult to detect randomly and will require ATPG patterns. The Fault coverage loss (0.4%) in Design *F* can be compensated by a combination of three options – applying more random patterns, calculating more top up patterns or adding more test points. In our experiment, either applying 100K more random patterns or 73 additional top up patterns could fully compensate the coverage loss and the coverage reached 98.27%.

An analysis of the dual polarity gates revealed that considering MUX primitives as non-inverting gates is not optimal. This is because paths going through the select input have an implied dual polarity. For example, in Fig. 5(a) illustrating a Type 1 control point, suppose that *OR Gate ("Gate")* is a MUX primitive and its select input is directly connected to the candidate functional flip-flop output, then the stuck-at-0 fault on the select input becomes impossible to detect without setting *TP_Enable* to *0*. Changing the control point for a Type 4 control point does not improve the situation as it makes the stuck-at-1 the hard to detect fault instead. Therefore, candidate flip-flops with a path going through the select input of a MUX primitive should be discarded. However, this non-optimal MUX primitives management had no impact on the experimental results of 5 of the 8 circuits since MUXes were implemented or modeled as AND/OR structures in those circuits. Design *D*, *E* and *F* have approximately 13,000, 38161 and 15612 instances of MUXes modeled as MUX primitives, however, the results for these designs show that the testability seems similar to that of other designs. Hence, in the experiments, the testability is not significantly affected by this issue.

To study how sensitive the fault coverage is to the *TP_Enable* signal probability, Design *E* is used with different signal probabilities of $(2^k - 1)/(2^k)$ for $k = 1$ to 8. 16000 and 100000 random patterns are applied to achieve random fault coverage. As can be seen in Fig. 8, the *TP_Enable* signal probability gives a significant improvement in the fault coverage. The fault coverage is increased about 0.5 % only by changing the signal probability. This is to be expected since the testpointAnalyze tool assumes that the TP_Enable probability is exactly 1. Both cases illustrate that there is saturation of the coverage. Therefore, the probability of *TP_Enable* needs to be kept high, say 15/16 or 31/32, so that the efficiency of the original test point insertion method is not affected. In the proposed method, the maximum fault coverage is obtained in this way. Design *A* show that the coverage goes down a little when *TP_Enable* has a signal probability 15/16 compared with 63/64. This happens because of the noise related to vectors. When different vectors are applied to CUT, they can introduce the noise. This may result in the lower test coverage in the benchmark circuits.

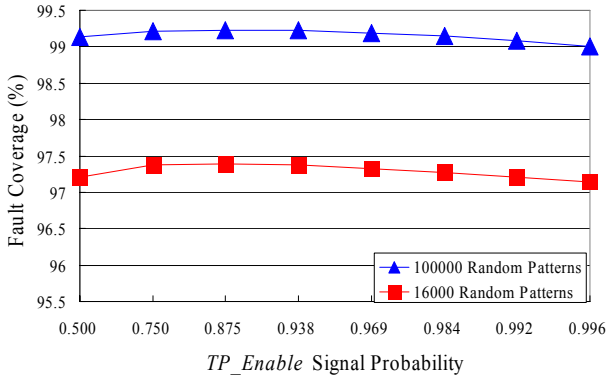


Figure 8. Testability vs. *TP_Enable* Signal Probability

6. Conclusions

The experimental results indicate that the methodology proposed in this paper can significantly reduce the number of dedicated flip-flops for driving control points by replacing them with functional flip-flops. Significant area savings are therefore achieved while preserving the random pattern testability of the circuit and without introducing new timing constraints that would complicate timing closure. The test point area was typically reduced by half while the fault coverage loss during the random pattern phase was limited to less than 0.1% for most circuits. Several options were identified to compensate for a slightly higher coverage loss observed for 2 circuits.

The proposed method can be used to implement test points calculated with existing algorithms without having to modify the algorithms. The method is therefore neutral with respect to the handling of unknowns in the circuit and test power as it does not deal with the selection of the test points, only their implementation. It should also be noted that the new test point implementation method gives the flexibility of adding more test points to achieve even higher coverage or reduce test time.

The run times of our software implementing the test points were not monitored. The proposed method only involves static tracing of the fan-in and fan-out of gates which are related to control points and very efficient algorithms are available for performing these tasks which are less complex than the algorithms used for test point selection itself.

Future work includes a more thorough investigation of circuit structures for which no suitable functional flip-flops could be identified to determine if a different test point implementation could be used. Also, a new implementation for observation points will be investigated. This will be useful for circuits for certain designs with a significant number of observation points.

References

[Barnhart 01] Barnhart, C., V. Brunkhorst, F. Distler, O. Farnsworth, B. Keller, and B. Koenemann**,** "OPMISR: the Foundation for Compressed ATPG Vectors," *Proc. of International Test Conference*, pp. 748-757, 2001

[Bershteyn 93] Bershteyn, M., "Calculation of multiple sets of weights for weighted random testing," *Proc. of International Test Conference*, pp. 1031-1040, 1993.

[Briers 86] Briers, A. J., and K.A.E. Totton, "Random Pattern Testability by Fast Fault Simulation," *Proc. of International Test Conference*, pp. 274-281, 1986.

[Chatterjee 95] Chatterjee, M., and D.K. Pardhan, "A Novel Pattern Generator for Near-Perfect Fault-Coverage," *Proc. of VLSI Test Symposium*, pp. 417-425, 1995.

[Cheng 95] Cheng, K.-T., and C.J. Lin, "Timing –Driven Test Point Insertion for Full-Scan and Partial-Scan BIST," *Proc. of International Test Conference*, pp. 506-514, 1995.

[Eichelberger 83] Eichelberger, E. B., and E. Lindbloom, "Random-Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self Test," *IBM Journal of Research and Development, Vol*. 27, No. 3, pp. 265-272, May 1983.

[Hellebrand 92] Hellebrand, S., S. Tarnick, J. Rajski, and B. Courtois, "Generation of Vector Patterns Through Reseeding of Multiple-Polynomial Linear Feedback Shift Register," *Proc. of International Test Conference*, pp. 120-129, 1992.

[Hellebrand 95] Hellebrand, S., J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois, "Built-in Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers," *IEEE Trans. on Computers*, Vol. 44, No. 2, pp. 223-233, Feb. 1995.

[ITRS 07] "The International Technology Roadmap for Semiconductors," Semiconductor Industry Association, 2007.

[Iyengar 89] Iyengar, V.S., and D. Brand, "Synthesis of Pseudo-Random Pattern Testable Designs," *Proc. International Test Conference*, pp. 501-508, 1989.

[Jas 01] Jas, A., and C.V. Krishna, and N.A. Touba, "Hybrid BIST based on Weighted Pseudo-Random Testing: A New Test Resource Partitioning," *Proc. of VLSI Test Symposium*, pp. 2-8, 2001.

[Kapur 94] Kapur, R., S. Patil, T.J. Snethen, and T.W. Williams, "Design of an efficient weighted random pattern generation system," *Proc. of International Test Conference*, pp. 491-500, 1994.

[Konemann 91] Koenemann, B., "LFSR-Coded Test Patterns for Scan Designs," *Proc. of European Test Conference*, pp. 237-242, 1991.

[Konemann 01] Koenemann, B., C. Barnhart, B. Keller, T. Snethen, O. Farnsworth, and D. Wheater, "A SmartBIST variant with guaranteed encoding," *Proc. of VLSI Test Symposium*, pp. 325-330, 2001.

[Krishna 01] Krishna, C.V., A. Jas, and N.A. Touba, "Test Vector Encoding Using Partial LFSR Reseeding," *Proc. of International Test Conference*, pp. 885 – 893, 2001.

[Krishnamurthy 87] Krishnamurthy, B., "A Dynamic Programming Approach to the Test Point Insertion Problem," Proc. of the 24th Design Automation Conference, pp. 695-704, 1987.

[Lai 05] Lai, L., J.H. Patel, T. Rinderknecht, and W.-T. Cheng, "Hardware efficient LBIST with complementary weights," Proc. of International Conference on Computer Design, pp. 479-481, 2005.

[LogicVision] ETAnalysis Tools Reference Manual, software version 6.0a, 2007.

[Nakao 99] Nakao, M., S. Kobayashi, K. Hatayama, and K. Iijima, "Low Overhead Test Point Insertion for Scan-Based BIST," *Proc. International Test Conference*, pp. 384-357, 1999

[OR1200] OPENCORES, *http://www.opencores.org*

[Pomeranz 02] Pomeranz, I., and S.M. Reddy, "3-Weight Pseudo Random Test Generation Based on a Deterministic Test Set for Combinational and Sequential Circuits," *IEEE Trans. on Computer-Aided Design*, Vol. 12, No. 7, pp. 1050-1058, 1993.

[Rajski 02] Rajski, J., J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, T. Kun-Han, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eider, and Q. Jun, "Embedded deterministic test for low cost manufacturing test," *Proc. of International Test Conference*, pp. 301-310, 2002.

[Seiss 91] Seiss, B.H., P. Trouborst, and M. Schulz, "Test Point Insertion for Scan-Based BIST," *Proc. European Test Conference*, pp. 253-262, 1991.

[Schnurmann 75] Schnurmann, H.D., E. Lindbloom, and R.G. Carpenter, "The Weighted Random Test-Pattern Generator," *IEEE Trans. on Computer*, Vol. C-24, No. 7. pp. 695-700, 1975.

[Tamarapalli 96] Tamarapalli, N., and J. Rajski "Constructive Multi-Phase Test Point Insertion for Scan-Based BIST," *Proc. of International Test Conference*, pp. 649-658, 1996.

[Touba 95a] Touba, N. A., and E. J. McCluskey, "Transformed Pseudo-Random Patterns for BIST," *Proc. of VLSI Test Symposium*, pp. 410-416, 1995.

[Touba 95b] Touba, N. A., and E. J. McCluskey, "Synthesis of Mapping Logic for Generating Transformed Pseeudo-Random Patterns for BIST," *Proc. International Test Conference*, pp. 674-682, 1995.

[Touba 96] Touba, N.A., and E.J. McCluskey, "Test Point Insertion Based on Path Tracing," *Proc. of VLSI Test Symposium*, pp. 2-8, 1996.

[Tsai 98] Tsai, H.-C., K.-T. Cheng, L.-J. Lin and S. Bhawmik, "Efficient Test Point Selection for Scan-based BIST," *IEEE Trans. VLSI Syst.,* Vol. 6, No. 4, pp. 667-676, 1998.

[Wunderlich 96] Wunderlich, H.-J., and G. Kiefer, "Bit-Flipping BIST," *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pp. 337-343, 1996.

[Yang 09] Yang, J.-S., and N.A. Touba, "Automated Selection of Signals to Observe for Efficient silicon Debug", *Proc. of VLSI Test Symposium*, 2009.

[Youssef 93] Youssef, M., Y. Savaria, and B. Kaminska, "Methodology for Efficiently Inserting and Condensing Test Points," *IEEE Proceedings Computers and Digital Techniques*, Vol. 140, pp. 145-160, May, 1993.