Combining Deterministic Logic BIST with Test Point Insertion

Harald Vranken [1] Florian Meister [2] Hans-Joachim Wunderlich [2]

[1] Philips Research Laboratories
IC Design – Digital Design & Test
Prof. Holstlaan 4, 5656 AA Eindhoven
The Netherlands
harald.vranken@philips.com

[2] University of Stuttgart
Computer Architecture Lab
Breitwiesenstr. 20/22, 70565 Stuttgart
Germany
wu@informatik.uni-stuttgart.de

Abstract

This paper presents a logic BIST approach which combines deterministic logic BIST with test point insertion. Test points are inserted to obtain a first testability improvement, and next a deterministic pattern generator is added to increase the fault efficiency up to 100%. The silicon cell area for the combined approach is smaller than for approaches that apply a deterministic pattern generator or test points only. The combined approach also removes the classical limitations and drawbacks of test point insertion, such as failing to achieve complete fault coverage and a complicated design flow. The benefits of the combined approach are demonstrated in experimental results on a large number of ISCAS and industrial circuits.

1. Introduction

Built-in self-test (BIST) for random logic is becoming an attractive alternative in IC testing [28]. Advances in deep-submicron IC process technology and core-based IC design methods allow to implement system chips that contain millions of transistors. The traditional approach of external testing using only automated test equipment (ATE) is becoming more and more difficult and costly. BIST is therefore expected to be widely used for IC manufacturing test in the near future. In addition, BIST already is commonly applied for board-level IC test and field test of critical applications such as telecommunication systems.

Logic BIST is currently supported by a few commercial CAD tools that are all based on the STUMPS architecture for pseudo-random testing. STUMPS is a test-per-scan BIST scheme, in which BIST hardware is added to a scannable circuit-under-test (CUT) [1][2]. Pseudo-random test stimuli are generated by a linear-feedback shift register (LFSR) or cellular automaton (CA), while the test responses are compacted into a signature by a multiple-input signature register (MISR). A test control unit controls the operation of the LFSR, MISR, and the scan chains in the CUT. A phase shifter is often added to obtain decorrelated pseudo-random stimuli for multiple scan chains [21].

The STUMPS architecture is attractive since it requires only a small amount of silicon area. However, the fault coverage that can be achieved with pseudo-random test patterns is usually insufficient for manufacturing test or high-quality field test. A straightforward approach to achieve complete fault coverage of all non-redundant faults is to apply additional external test patterns which have been generated by ATPG ('top-off patterns'). However, this approach still requires a considerable amount of external testing. For instance, it has been reported in [3] that detecting the last 10% of undetected faults typically requires 70% or more of the test patterns in an ATPG test set.

An alternative approach for improving fault coverage is to use an enhanced BIST scheme, in which either the CUT or the on-chip pattern generator is improved. Commercial logic BIST tools currently offer test point insertion (TPI) to improve the random testability of the CUT. Test points however may affect the CUT's performance, and additional timing analysis and design iterations complicate the design flow. TPI improves the fault coverage, but restrictions on where to insert test points as well as limitations of TPI algorithms, cause that complete fault coverage is difficult to achieve with TPI. Hence, additional external ATPG patterns still have to be applied to improve fault coverage. For instance, top-off ATPG patterns are applied in [14] on top of logic BIST with TPI to improve the fault coverage from 95-96% to 96-97%, which still requires 25-65% of the patterns in the full ATPG test set.

Complete fault coverage without CUT modification and external test patterns can be achieved by using a BIST scheme containing a more sophisticated pattern generator. Examples are weighted pseudo-random or pseudo-exhaustive pattern generators, which however require a relatively large amount of additional silicon area. A more promising approach is deterministic logic BIST (DLBIST) in which a deterministic pattern generator is used. Various DLBIST schemes have been proposed recently, e.g. schemes based on reseeding [11][12], bit-fixing [25], bit-flipping [16][17][18][27], and folding counters [13][19]. The bit-flipping scheme seems to be most promising.

In this paper we propose a logic BIST scheme that combines the strengths of both test point insertion and deterministic logic BIST. The general idea is that inserting a limited amount of test points can already result in a major increase in fault coverage. TPI is however not suited for obtaining complete fault coverage, and therefore we apply a deterministic pattern generator in addition. This combined approach allows to achieve complete fault coverage, while avoiding the limitations and drawbacks of TPI, and reducing the silicon area when compared to a pure DLBIST implementation. We use a TPI method based on [23] and bit-flipping DLBIST. To the best of our knowledge, this is the first paper which proposes such a combined approach.

In the remainder of this paper, first prior work on bit-flipping DLBIST and TPI is summarized in Section 2. The combined DLBIST-TPI approach is described in Section 3, and details on the TPI algorithm are given in Section 4. The impact of the combined approach on the design flow is discussed in Section 5. Experimental results on a large set of ISCAS benchmark circuits and industrial circuits are reported in Section 6, and Section 7 concludes the paper.

2. Prior Work

2.1 Bit-Flipping DLBIST

The architecture of the bit-flipping DLBIST scheme is shown in Figure 1 [16][17][18][27]. The LFSR generates a sequence of pseudo-random test stimuli. Some of these stimuli are modified by the bit-flipping logic (BFL). The BFL is constructed by considering deterministic ATPG patterns for those faults that are not detected by the pseudo-random stimuli. Hence, the BFL changes some pseudo-random test stimuli as generated by the LFSR into deterministic stimuli, which ensures that complete fault coverage can be achieved. The stimuli are applied to the CUT via the scan chains, and the test responses are compacted in a MISR.

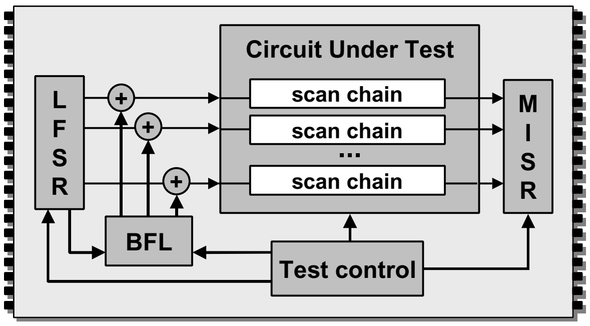

Figure 1: Bit-flipping DLBIST

The bit-flipping DLBIST scheme allows to achieve complete fault coverage and requires only a small amount of silicon area. Experimental results on industrial circuits as reported in [15] show that 100% fault efficiency can be achieved for circuits up to 100k gates with a test length of 10,000 test patterns at a total cell area cost for the DLBIST hardware of typically 5-15%. The silicon area for the LFSR, MISR, and test controller is more or less constant, and increases only slightly with increasing circuit sizes. The size of the BFL however depends mainly on the random testability of the CUT. For circuits that are poorly random testable, the BFL may become quite large. It is also shown in [15] that the silicon area for the BFL can be reduced considerably by making trade-offs in test time and/or test quality.

The main advantage of DLBIST is that no modification of the CUT is required, provided that the CUT is BIST-ready (i.e. no unknown values are generated in the CUT that propagate into the MISR and corrupt the signature). All the DLBIST hardware is added as a shell around the CUT. This simplifies the design flow, since the BIST hardware can easily be integrated into the IC design and no design iterations are required to solve timing violations as with TPI.

2.2 TPI

Already in the 1970s, TPI was considered as a useful technique to improve the testability of a CUT [10], and ever since a large number of TPI methods has been published (e.g. [7][8][20][22][23][24][26]). The goal of the work presented in this paper is to investigate the combination of DLBIST and TPI. For that purpose we selected a TPI method similar to [23] and bit-flipping DLBIST.

The TPI method in [23] is based on COP (controllability/observability program) testability analysis [4]. COP controllability and observability probabilities are computed for all signal lines in the CUT, which reflect the random testability of the combinational logic. The COP values can be used to identify suitable locations for inserting test points [20]. However, COP values have a very local nature. Better results for TPI are obtained by using a more global measure based on a cost function that considers the testability of all faults in the CUT. One way of identifying suitable locations for test points is to simply insert test points one at a time at all possible locations, and to consider how this affects the cost function. However, this is computationally expensive and hence intractable in practice. A more feasible approach is to first make a pre-selection of promising locations for inserting test points. This pre-selection can be made by estimating the impact of test points on the cost function using gradients (i.e. derivatives of the cost function with respect to controllability and observability) [20]. Estimating the cost change solely based on gradients is however also not very accurate, since the gradients represent changes of the cost function with respect to an infinitely small change in controllability or observability on a signal line. In reality, the insertion of a test point will cause controllability or observability to change rather drastically. The gradients are therefore used in [23] to calculate an approximation of the actual cost reduction, called the Cost Reduction Factor (CRF). The CRF is accurate enough to identify promising locations for inserting test points. The actual costs are now only computed for the pre-selected candidate test points, and the test point that offers the largest cost reduction is actually inserted.

The work in this paper is largely based on the TPI method in [23]. The TPI method in [26] is a further improvement of [23] that uses a more accurate CRF. The TPI method in [7] is another improvement that also considers the impact of TPI on the CUT's performance. The TPI method in [8] is partially based on [23][26], where TPI is used to reduce the size of compact test sets.

3. Combining DLBIST with TPI

In this paper we present a combination of bit-flipping DLBIST with TPI. In this combined approach, we first insert a limited number of test points into the CUT to improve its random testability. Next, a BFL is generated for the CUT with test points to achieve 100% fault efficiency.

The insertion of only one or two test points per thousand gates in general provides a significant testability improvement. However, TPI has several limitations and drawbacks. First, TPI implies modification of the CUT, which may have a negative impact on the performance if test points are inserted in critical paths. Timing analysis after test point insertion is therefore required as an additional task in the design process, and several design iterations may be required to solve timing violations. Hence, TPI complicates the design flow, while prohibiting the insertion of test points in critical paths might reduce the fault coverage.

Second, many TPI methods are not able to achieve complete fault coverage. TPI inserts additional wires and logic gates into the CUT, which consequently increases the number of faults in the CUT that have to be tested. Hence, at a certain moment adding new test points may not give any benefits if the reduction of the cost function due to the test point is exceeded by an increase of the cost function due to the new faults. Furthermore, inserting a large number of control points increases the probability that control points start to conflict with each other. Ignoring this effect may lead to a large number of futile control points and reduced fault coverage [24]. In general, the fault coverage improves well for inserting the first test points, but levels off when in more test points. Detecting the last remaining faults basically requires that a dedicated test point is inserted for each fault, which leads to a large number of test points. All these effects, combined with the restriction of inserting test points in critical paths, cause that complete fault coverage can generally not be achieved with TPI. This is confirmed in several industrial case studies [9][14], where only 95-96% fault coverage could be achieved by logic BIST with TPI, and improving the fault coverage required additional external testing with ATPG patterns.

On the other hand, bit-flipping DLBIST guarantees complete fault coverage, since the BFL can be improved by embedding deterministic ATPG patterns up to the point where all faults are detected. The size of the BFL may however become rather large. In [15] is reported that the BFL requires typically less than 10% silicon area, but this can increase up to 30% for circuits that have poor random testability.

The proposed combined approach therefore is to first insert a certain number of test points into the CUT. Since the testability increase tends to be the largest for the first few test points, only a limited number of test points is inserted. Next, a BFL is generated for the modified circuit with test points to achieve 100% fault efficiency. The increased random testability provided by test points, allows for a smaller BFL. Furthermore, inserting only a few test points provides that modification of the CUT is minimized, and hence the performance impact caused by test points can be kept small. A trade-off can be made between the number of inserted test points (and the resulting impact on CUT area and timing) and decrease of the BFL size.

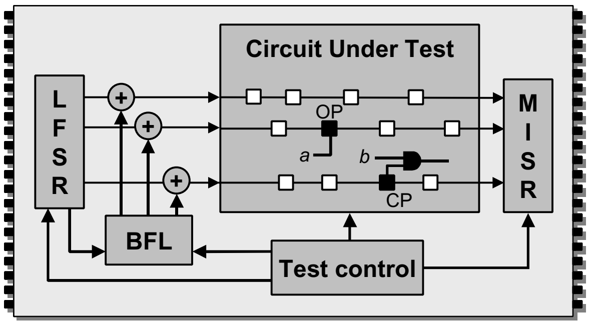

Figure 2: Combined DLBIST-TPI

The architecture for the combined DLBIST-TPI approach is shown in Figure 2. Test points in the form of observation points (OP) and control points (CP), are added to the CUT. An OP is implemented by an additional scan flip-flop to increase the observability of a signal line (e.g. line *a* in Figure 2). A CP is implemented by either a two-input AND-gate or a two-input OR-gate with an additional scan flip-flop to increase the 0-controllability or 1-controllability of a signal line (e.g. line *b* in Figure 2). During normal circuit operation, the control input of a CP is set to its non-controlling value (0 for OR-type CP, 1 for AND-type CP), while in test mode the control input is assigned by means of data that is shifted into the scan chain.

4. TPI Method for Combined DLBIST-TPI

Our TPI method for combined DLBIST-TPI is largely based on [23]. The TPI algorithm is outlined in Figure 3 and described in more detail below.

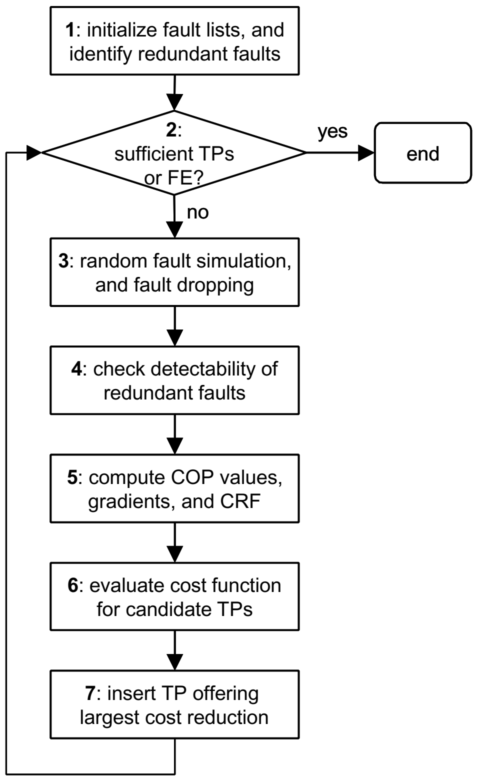
1: initialize fault lists, and identify redundant faults
2: sufficient TPs or FE?
yes
end
no
3: random fault simulation, and fault dropping
4: check detectability of redundant faults
5: compute COP values, gradients, and CRF
6: evaluate cost function for candidate TPs
7: insert TP offering largest cost reduction

Figure 3: TPI algorithm

1. The TPI algorithm operates on a global fault list that contains all faults in the CUT. At the beginning of the TPI algorithm, the redundant faults in the CUT are identified.
2. The algorithm inserts test points into the CUT until a certain stop criterion is reached. The stop criterion is specified by the user, e.g. stopping when a certain fault efficiency is reached, when a certain reduction of the cost function is achieved, or when the maximum number of test points is inserted. The algorithm may also stop prematurely if no more suitable locations for inserting test points can be found.
3. In order to speed up the TPI algorithm, fault simulation with pseudo-random patterns is performed after a certain amount of test points has been inserted. The faults that are detected multiple times correspond to easy-testable faults, and are dropped from the fault lists. In our experiments, we perform fault simulation each time after 10% of the maximum allowed number of test points is inserted, and we drop faults that are detected at least 3 times.
4. Test point insertion may cause some combinationally redundant faults to become detectable. It is checked at regular intervals whether redundant faults have become detectable, and these faults may be added to the fault lists. In our experiments however we ignore these faults, since we observed for some circuits that inserting a few test points may cause a large number of redundant faults to become detectable. In the combined DLBIST-TPI approach, many of those faults have to be detected by deterministic patterns and hence cause a considerable increase of the BFL size.
5. The next steps in the TPI algorithm are almost similar to the TPI method in [23]. COP controllability and observability values, gradients, and the CRF are computed. A modification is that candidate locations which would only improve detectability of the faults already dropped during pseudo-random simulation, are not considered as test point candidates.

6. Candidate test points are inserted temporarily one at a time at the pre-selected locations, and the cost function is re-evaluated.
7. Once all promising candidates have been evaluated, the one that gives the best actual cost reduction is selected and permanently inserted. A test point introduces new faults to the CUT, and these faults are added to the fault lists.

We refer to the above TPI algorithm as *global TPI*, since it considers all faults in the CUT. An alternative approach is *targeted TPI*, in which test points are inserted particularly to detect the hardest-to-test faults. These faults can generally not be detected by random patterns, and deterministic patterns to detect these faults require a large number of specified bits. As a consequence, these faults typically cause a considerable increase of the BFL. This effect has been clearly shown in [15], where increasing the fault efficiency from 99% to 100% caused an increase of the BFL size up to 70%. In this case, the last 1% of undetected faults corresponded to the hardest-to-test faults, which require deterministic patterns with the largest number of specified bits.

The hardest-to-test faults are identified first in targeted TPI. An intermediate BFL is generated which achieves only a moderate fault efficiency. The faults that are not detected by the BFL are considered as target faults. The explicit goal of targeted TPI now is improving testability for these target faults. After TPI, the actual BFL is generated for the modified circuit with test points to achieve 100% fault efficiency.

The TPI algorithm for targeted TPI is very similar to global TPI with a few modifications. Targeted TPI operates on the list with target faults, instead of the global fault list. Furthermore, the gradients of the non-target faults are set to zero. Hence, the non-target faults do not contribute to the CRF, and the CRF is based solely on the estimated detectability improvement of target faults. This also implies that possible test point locations which would only influence non-target faults, are ignored. A weight factor is used to control the contribution of target and non-target faults to the cost function. Our experiments indicated that best results are obtained using a weight factor of 0.5, which implies that both target and non-target faults contribute equally to the cost function.

5. Impact of DLBIST-TPI on Design Flow

The main drawbacks of TPI are its impact on the CUT performance, and the complicated design flow. Several solutions have been proposed to deal with this. In [7], timing-driven TPI is proposed, which considers the performance impact of test points in the cost function to avoid the insertion of test points in critical paths. In [9], TPI is performed in two stages. Unconstrained TPI is performed first, and timing analysis is performed to identify timing violations due to test points. This yields a set of constraints (i.e. paths in which no test points are allowed). The actual TPI is now performed in a second run which considers the constraints. Restricting the insertion of test points in critical paths however results in reduced fault coverage. An alternative approach is to insert test points at RT level, as proposed in [22]. This simplifies the design flow, but cannot guarantee that complete fault coverage is achieved at the gate level after synthesis.

In a combined DLBIST-TPI approach, it is well feasible to use timing-driven TPI or constrained TPI, and also RTL-TPI may be used. The combined DLBIST-TPI approach allows to limit the number of inserted test points, while 100% fault efficiency is still reached due to the BFL. The BFL and other DLBIST hardware is added as a shell around the CUT, and can easily be integrated into the IC design without affecting the CUT's performance.

6. Experimental Results

We evaluated the combined DLBIST-TPI approach using ISCAS benchmark circuits [5][6] that still have undetected faults after applying 10,000 random patterns, as well as industrial circuits from Philips. More details on the Philips circuits are available in [15], where the same circuits are used for evaluation of the bit-flipping DLBIST scheme. The primary goal of the experiments is to explore the impact on silicon area.

Table 1 shows the experimental results for the ISCAS circuits, and Table 2 for the industrial circuits. Column 1 shows the circuit names. The Philips circuits are named *pN*, where *N* denotes the number of signal lines in the circuit.

Columns 2 to 4 report the fault efficiency that is obtained either with 10,000 pseudo-random patterns (FE_{Random}), with pure DLBIST and combined DLBIST-TPI ($FE_{DLBIST\text{-}TPI}$), and with TPI only (FE_{TPI}). In principle it is possible to achieve 100% fault efficiency with pure DLBIST and combined DLBIST-TPI in all cases. However, run-time limitations of our prototype tools caused that some faults were aborted for the industrial circuits. For TPI only, 100% fault efficiency could not be achieved for all cases, and inserting more test points would not give further improvements. Hence, not achieving 100% with TPI is a limitation of the method, while not achieving 100% with DLBIST or combined DLBIST-TPI is purely a tool limitation and not at all a limitation of the method. Despite these limitations, higher fault efficiency is still obtained with combined DLBIST-TPI.

Columns 5 to 11 show the silicon cell area required for the BFL and test points, as percentage of the CUT area. The size of the LFSR, MISR, and test controller is more or less constant per circuit, and the area for these blocks is therefore not included. The area results are obtained using a 0.25 µm CMOS Philips library and Synopsys' Design Compiler. The area numbers cover the cell area for the logic gates only, and do not include the area for wiring and embedded memories. The best results per circuit are shown in bold.

Column 5 (*BFL100*) shows the results for pure bit-flipping DLBIST (without test points), as reported previously in [15]. Columns 6 and 7 show the results for the combined DLBIST-TPI approach with targeted TPI. The target faults are obtained by generating an intermediate BFL achieving either 95.0% (Column 6, *Targ95*) or 99.5% (Column 7, *Targ99.5*) fault efficiency. The undetected, testable faults are used next as target faults for targeted TPI. The actual BFL is generated on the circuit with test points to achieve 100% fault efficiency. Columns 8 to 10 show the results for the combined DLBIST-TPI approach with global TPI, where either at most one test point is inserted per thousand gates (Column 8, *Glob1pt*), or test points are inserted to achieve 95.0% (Column 9, *Glob95*) or 99.5% (Column 10, *Glob99.5*) fault efficiency. The BFL is generated next for the circuit with test points to achieve 100% fault efficiency. Finally, Column 11 (*TPI100*) shows the results for pure TPI (without BFL), where the target is to achieve 100% fault efficiency. However, this could not be reached for all circuits, as indicated in Column 4 (FE_{TPI}).

The last rows in the tables show the average results for all circuits. The averages for the silicon cell area are in fact weighted averages, obtained by dividing the cumulative area of BFL and test points by the cumulative CUT area of all circuits. The results indicate that pure bit-flipping DLBIST requires on average considerably more area than pure TPI or a combined DLBIST-TPI approach. Best results for both the ISCAS and Philips circuits are obtained by the combined approach of global TPI (*Glob99.5*) and

	Fault Efficiency (%)			Cell Area (%)						
1 circuit	2 FE$_{Random}$	3 FE$_{BFL-TPI}$	4 FE$_{TPI}$	5 BFL100	6 Targ95	7 Targ99.5	8 Glob1pt	9 Glob95	10 Glob99.5	11 TPI100
c2670	88.43	100	100	38.06	3.76	3.76	3.76	3.76	2.54	**2.18**
c7552	96.09	100	100	22.63	20.29	23.07	21.53	21.46	5.23	**4.96**
s641	97.84	100	100	6.56	5.50	**3.82**	5.50	6.56	7.18	6.72
s713	98.16	100	99.82	6.46	**3.56**	3.63	**3.56**	6.46	14.66	15.53
s838.1	59.72	100	100	49.03	30.36	60.30	30.36	18.26	18.83	**17.74**
s5378	98.57	100	100	2.64	**1.07**	1.83	1.35	2.64	1.46	1.28
s9234	89.71	100	100	13.51	4.37	8.37	5.89	6.34	2.81	**2.04**
s13207	93.92	100	100	2.98	1.40	1.34	1.61	1.27	1.41	**0.96**
s15850	87.86	100	99.97	6.46	4.59	4.23	5.69	5.83	3.55	**3.43**
s38417	92.66	100	100	10.21	3.99	4.55	5.02	6.50	3.12	**3.08**
s38584	97.67	100	100	1.89	1.05	**1.02**	1.22	1.89	1.20	1.87
Average	90.97	100	99.98	7.48	3.62	4.31	4.30	4.94	**2.59**	2.64

Table 1: Fault efficiency (%) and cell area (%) of BFL and test points for ISCAS circuits

	Fault Efficiency (%)			Cell Area (%)						
circuit	FE$_{Random}$	FE$_{BFL-TPI}$	FE$_{TPI}$	BFL100	Targ95	Targ99.5	Glob1pt	Glob95	Glob99.5	TPI100
p2221	88.52	100	100	9.50	5.53	4.03	5.22	3.28	**2.43**	2.87
p2441	92.20	100	100	8.94	5.02	9.15	5.33	5.55	3.42	**2.85**
p2675	99.70	100	100	0.48	**0.24**	**0.24**	1.62	0.48	0.48	2.65
p3295	92.01	100	100	7.16	6.30	5.85	5.89	6.79	5.47	**2.73**
p4210	94.47	100	100	12.01	5.83	12.22	5.83	7.91	6.01	**0.89**
p4250	96.09	100	100	8.92	**5.17**	8.51	5.58	8.53	5.26	9.14
p4919	97.06	100	100	4.46	3.68	3.91	4.48	4.46	4.37	**2.92**
p5918	97.65	99.95	99.96	6.02	3.39	6.47	**3.26**	6.15	3.32	5.18
p6291	96.80	100	100	7.25	6.60	5.51	4.77	8.33	**2.43**	2.60
p7318	99.84	100	100	**0.07**	0.10	0.10	0.16	**0.07**	**0.07**	**0.07**
p7890	98.64	100	100	3.65	1.27	1.22	**0.95**	3.65	1.32	1.68
p8689	88.57	99.99	100	19.72	17.06	17.16	14.97	14.97	8.28	**4.76**
p8873	96.40	100	98.52	6.12	6.51	4.40	3.60	6.12	2.86	**0.87**
p9041	97.01	100	99.99	12.06	6.98	9.41	7.66	12.06	5.24	**2.84**
p10705	95.50	99.99	100	6.93	5.83	4.94	5.92	6.70	3.61	**1.77**
p12292	97.64	99.74	99.83	15.43	7.36	5.88	6.29	15.43	5.29	**3.27**
p12903	92.92	100	100	8.25	4.83	6.74	5.28	8.19	4.93	**3.87**
p13033	95.57	100	100	7.92	8.66	8.59	8.66	8.66	**4.82**	14.16
p13651	99.91	99.93	100	**0.27**	0.68	0.68	2.13	**0.27**	**0.27**	8.52
p14473	86.47	100	99.57	30.27	21.59	23.52	20.83	16.15	10.52	**7.99**
p17828	96.82	100	100	4.34	2.17	2.54	2.30	4.34	2.11	**1.73**
p22383	93.02	99.97	100	18.93	3.40	3.55	3.85	11.83	3.08	**2.13**
p23572	85.53	100	100	8.70	5.00	5.98	4.25	4.61	2.70	**1.75**
p24370	94.45	99.97	99.79	16.69	12.47	15.27	12.08	16.14	9.25	**4.30**
p25015	98.41	99.93	100	3.95	2.70	3.04	**2.56**	3.95	3.04	5.67
p27369	99.31	99.96	99.97	2.04	4.56	3.65	**1.96**	2.04	2.05	12.96
p27530	98.28	99.98	99.71	13.04	9.14	10.73	7.10	13.04	7.02	**1.41**
p44177	97.30	99.95	98.69	3.47	3.27	3.27	3.61	3.47	3.58	**0.13**
p52251	99.58	100	99.99	0.61	0.70	0.70	**0.54**	0.61	0.58	0.63
p52922	92.77	99.98	99.39	7.57	6.45	9.87	4.97	6.54	**2.32**	7.82
p64984	94.47	99.98	99.79	10.45	5.41	5.90	5.11	8.55	3.08	**1.63**
p80590	99.16	99.94	100	2.33	2.75	4.02	**0.83**	2.33	0.97	1.06
Average	95.38	99.98	99.85	7.23	4.95	5.87	4.24	6.06	**3.03**	3.34

Table 2: Fault efficiency (%) and cell area (%) of BFL and test points for Philips circuits

DLBIST. The area results for pure TPI are only slightly worse, but 100% fault efficiency could not always reached with pure TPI. These results confirm that the combined DLBIST-TPI approach requires in general less silicon cell area for achieving 100% fault efficiency in comparison to pure bit-flipping DLBIST or even pure TPI.

In our experiments, we did not constrain the TPI algorithm to avoid insertion of test points in critical paths. Consequently, the performance of some circuits is affected by TPI. In Table 1 and 2, the circuits in which TPI affected critical timing paths are shaded. These results are obtained by considering the maximum logic depth before and after TPI. The results clearly show that inserting more test points increases the probability that performance is affected, which is particularly the case for global TPI with 99.5% target fault efficiency (*Glob99.5*) and pure TPI (*TPI100*). Although these approaches require minimum silicon area, they may not be the best choices in practice due to their impact on the performance.

For targeted TPI, a small target fault list restricts the TPI algorithm for selecting proper candidate test point locations, which leads to inferior results. This effect becomes worse as the target fault list becomes smaller, which explains why targeted TPI starting from 99.5% fault efficiency (*Targ99.5*), performs worse than targeted TPI starting from 95% fault efficiency (*Targ95*). In principle, best results can be achieved by targeted TPI, but this requires careful tuning of the target fault list.

Circuits p2675, p7318, p13651, p27369, p52251, and p80590 are very well random testable, and >99% fault efficiency is achieved with 10,000 pseudo-random patterns. The BFL for these circuits in pure DLBIST is very small. The silicon area increases slightly for the combined DLBIST-TPI approaches, while a huge increase is observed for some of these circuits (e.g. p13651) for pure TPI. Hence, pure DLBIST is the best choice for circuits that are very well random testable.

For circuits s641, s713, p4250, p13033, p25015, and p52922, pure TPI also requires more silicon area than pure DLBIST, while best results are obtained with combined DLBIST-TPI. These circuits have moderate random testability (95-99% fault efficiency).

The BFL in pure DLBIST is very large (>15%) for circuits c2670, c7552, s838.1, p8689, p12292, p14473 p22383, and p24370, which have poor random testability (<95% fault efficiency). For these circuits, the silicon area is reduced considerably by using a combined DLBIST-TPI approach or pure TPI.

The Philips circuits are in general better random testable than the ISCAS circuits: on average, 91% fault efficiency is achieved with 10,000 random patterns for the ISCAS circuits, versus 95% for the Philips circuits. This explains why the gain of combined DLBIST-TPI approaches and pure TPI versus pure DLBIST is higher for the ISCAS circuits than the Philips circuits.

7. Conclusion

We presented a combined DLBIST-TPI approach, which combines the strengths of both bit-flipping DLBIST and TPI. A limited number of test points is inserted first to improve random testability, and DLBIST hardware is added next to improve the fault efficiency up to 100%. This combined approach allows to achieve 100% fault efficiency at a small amount of silicon area. In addition, the performance impact due to test points can be limited or avoided, which simplifies the design flow.

Experimental results on a large number of ISCAS and industrial circuits show that best results are obtained with combined DLBIST-TPI.

Acknowledgements

The authors acknowledge Gundolf Kiefer and Valentin Gherman (University of Stuttgart) for their contribution to the work described in this paper. Also Jeroen Geuzebroek (Delft University of Technology) is acknowledged for discussions on the combined DLBIST-TPI approach.

References

[1] P.H. Bardell, W.H. McAnney, *Parallel Pseudo-random Sequences for Built-In Test*, Proc. Int. Test Conf., IEEE, 1984, pp. 302-308.

[2] P. Bardell, W.H. McAnney, J. Savir, *Built-in Test for VLSI*, Wiley-Interscience, New York, 1987.

[3] R.W. Bassett et al., *Low Cost Testing of High Density Logic Components*, Proc. Int. Test Conf., IEEE, 1989, pp. 550-557.

[4] F. Brglez, *On Testability Analysis of Combinational Networks*, Proc. Int. Symp. on Circuits and Systems, IEEE, 1984, pp. 221-225.

[5] F. Brglez, H. Fujiwara, *A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran*, Proc. Int. Symp. On Circuits and Systems, IEEE, 1985, pp. 663-698.

[6] F. Brglez, D. Bryan, K. Komzminski, *Combinational Profiles of Sequential Benchmark Circuits*, Proc. Int. Symp. on Circuits and Systems, IEEE, 1989, pp. 1929-1934.

[7] K.-T. Cheng, C.-J. Lin, *Timing-Driven Test Point Insertion for Full-Scan and Partial-Scan BIST*, Proc. Int. Test Conf., IEEE, 1995, pp. 506-514.

[8] M.J. Geuzebroek, J.Th. van der Linden, A.J. van de Goor, *Test Point Insertion for Compact Test Sets*, Proc. Int. Test Conf., IEEE, 2000, pp. 292-301.

[9] X. Gu et al., *An Effort-Minimized Logic BIST Implementation Method*, Proc. Int. Test Conf., IEEE, 2001, pp. 1002-1010.

[10] J.P. Hayes, A.D. Friedman, *Test Point Placement to Simplify Fault Detection*, IEEE Trans. on Computers, Vol. C-33, July 1974, pp. 727-735.

[11] S. Hellebrand et al., *Generation of Vector Patterns Through Reseeding of Multiple-Polynomial Linear Feedback Shift Registers*, Proc. Int. Test Conf., 1992, pp. 120-129.

[12] S. Hellebrand et al., *Pattern Generation for a Deterministic BIST Scheme*, Proc. Int. Conf. on CAD, IEEE, 1995, pp. 88-94.

[13] S. Hellebrand, H.-G. Liang, H.-J. Wunderlich, *A Mixed Mode BIST Scheme Based on the Reseeding of Folding Counters*, Proc. Int. Test Conf., IEEE, 2000, pp. 778-784.

[14] G. Hetherington et al., *Logic BIST for Large Industrial Designs: Real Issues and Case Studies*, Proc. Int. Test Conf., IEEE, 1999, pp. 358-367.

[15] G. Kiefer et al., *Application of Deterministic Logic BIST on Industrial Circuits*, Proc. Int. Test Conf., IEEE, 2000, pp. 105-114.

[16] G. Kiefer, H.-J. Wunderlich, *Using BIST Control for Pattern Generation*, Proc. Int. Test Conf., IEEE, 1997, pp. 347-355.

[17] G. Kiefer, H.-J. Wunderlich, *Deterministic BIST with Multiple Scan Chains*, Proc. Int. Test Conf., IEEE, 1998, pp. 1057-1064.

[18] G. Kiefer, H.-J. Wunderlich, *Deterministic BIST with Partial Scan*, Proc. European Test Workshop, IEEE, 1999, pp. 110-116.

[19] H.-G. Liang, S. Hellebrand, H.-J. Wunderlich, *Two-Dimensional Test Data Compression for Scan-Based Deterministic BIST*, Proc. Int. Test Conf., IEEE, 2001, pp. 894-902.

[20] R. Lisanke et al., *Testability-Driven Random Test-Pattern Generation*, IEEE Trans. on CAD, Vol. CAD-6, November 1987, pp. 1082-1087.

[21] J. Rajski, N. Tamarapalli, J. Tyszer, *Automated Synthesis of Large Phase Shifters for Built-In Self-Test*, Proc. Int. Test Conf., IEEE, 1998, pp. 1047-1056.

[22] S. Roy, G. Guner, K.-T. Cheng, *Efficient Test Mode Selection & Insertion for RTL-BIST*, Proc. Int. Test Conf., IEEE, 2000, pp. 263-272.

[23] B.H. Seiss, P.M. Trouborst, M.H. Schulz, *Test Point Insertion for Scan-Based BIST*, Proc. Eur. Test Conf., VDE Verlag, 1991, pp. 253-262.

[24] N. Tamarapalli, J. Rajski, *Constructive Multi-Phase Test Point Insertion for Scan-Based BIST*, Proc. Int. Test Conf., IEEE, 1996, pp. 649-658.

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

[26] H.-C. Tsai et al., *A Hybrid Algorithm for Test Point Selection for Scan-Based BIST*, Proc. Design Automation Conf., ACM/IEEE, 1997, pp. 478-483

[27] H.-J. Wunderlich, G. Kiefer, *Bit-Flipping BIST*, Proc. Int. Conf. on CAD, IEEE, 1996, pp. 337-343.

[28] Y. Zorian, *Testing the monster chip*, IEEE Spectrum, July 1999, pp. 54-60.