Impact of Test Point Insertion on Silicon Area and Timing during Layout

Harald Vranken [1] Ferry Syafei Sapei [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
Institute of Comp. Eng. & Comp. Arch.
Pfaffenwaldring 47, 70565 Stuttgart
Germany
wu@informatik.uni-stuttgart.de

Abstract

This paper presents an experimental investigation on the impact of test point insertion on circuit size and performance. Often test points are inserted into a circuit in order to improve the circuit's testability, which results in smaller test data volume, shorter test time, and higher fault coverage. Inserting test points however requires additional silicon area and influences the timing of a circuit. The paper shows how placement and routing is affected by test point insertion during layout generation. Experimental data for industrial circuits show that inserting 1% test points in general increases the silicon area after layout by less than 0.5% while the performance of the circuit may be reduced by 5% or more.

1. Introduction

Test point insertion (TPI) is a well-known design-for-testability (DfT) technique that inserts additional logic into a circuit to increase the circuit's testability. TPI aims particularly at improving the observability and/or controllability of hard-to-test signal lines in a circuit. Various TPI methods have been proposed since the 1970s, and nowadays TPI is supported by commercial EDA tools and commonly applied in industry.

The testability improvement offered by TPI results in higher fault coverage, smaller test data volume, and shorter test application time. Unfortunately, TPI also has some well-known disadvantages: test points costs additional silicon area, they affect the circuit's timing, and resolving timing violations due to TPI complicates the design flow.

Several interesting papers have been published recently with case studies on the advantages and disadvantages of TPI [5][6]. However, they do not truly analyse the effects of TPI on placement and routing during layout generation. The intention of this paper is to fill this gap. The paper presents an experimental investigation on the impact of TPI during layout generation, and quantifies the effects on silicon area and timing. The experiments are performed on industrial circuits using existing, state-of-the-art methods and tools for TPI and layout generation.

In the remainder of the paper, prior work on TPI is discussed in Section 2. The TPI method and tool flow used for our experiments are outlined in Section 3. The experimental results are presented in Section 4, and discussed in Section 5. Section 6 concludes the paper.

2. Prior work on test point insertion

Most TPI methods are used with logic built-in self-test (LBIST) [2][7][9][10][11]. LBIST implements a pseudo-random stimulus generator on-chip. This costs very little silicon area, but the fault coverage achieved with pseudo-random patterns only is generally insufficient for high-quality IC testing due to pseudo-random persistent faults. Test points are therefore inserted to increase the detectability of these faults, which results in higher fault coverage. Recent case studies on successful industrial application of TPI with pseudo-random LBIST have been reported in [5][6].

More advanced, deterministic LBIST schemes implement an improved pattern generator on-chip for producing deterministic patterns. Combining TPI with bit-flipping deterministic LBIST has been proposed in [12]. The silicon area for TPI with DLBIST was shown to be smaller than the area when using only TPI or only DLBIST.

Recently TPI methods have been introduced to reduce the number of ATPG patterns for scan-based external testing [3][4]. Reducing the number of patterns leads to less test data volume and shorter test application time.

The main disadvantages of TPI are additional silicon area and its potential impact on the timing of a circuit. Resolving timing violations may cause several design iterations. Solutions for TPI with LBIST have been proposed in [2][5][8][12], although they do not analyse in depth nor quantify the impact of TPI on area and timing. In [2], timing analysis is performed on the circuit layout before TPI to identify paths with small slack. TPI is performed next on the gate-level netlist, and no test points are inserted in the identified paths. A new layout is generated including test points. A disadvantage is that placement of test points is restricted to the boundaries of the circuit, since otherwise the timing of the circuit would still be affected after TPI. A related approach is proposed in [12], where test points in critical paths are excluded, and deterministic LBIST hardware is added around the circuit. In [5], test points are inserted first without constraints. Timing analysis is performed next, and violations due to test points are simply solved by removing those test points, which however causes fault coverage loss. In [8], TPI is performed at the RT-level. This implies that test points are already considered during logic synthesis, which avoids later design iterations. The risk however is that logic synthesis may be unable to achieve the target frequency due to the RTL modifications.

3. Tools and flow

3.1 Test points and TPI

We used the TPI method as described in [3][4], which aims at reducing the number of compact ATPG patterns for scan-based external testing. This TPI method is supported by Philips' computer-aided test (CAT) tools. A test point is implemented by a transparent scan flip-flop (TSFF), as shown in Figure 1, which serves both as observation point and control point at the same time. A TSFF consists of a scan flip-flop with an additional multiplexer at the output. In application mode, both control signals *TE* and *TR* are 0. Inserting a test point implies that the propagation delay in application mode is increased by at least the delay of the two multiplexers. In scan shift mode, both *TE* and *TR* are 1. In scan capture mode, *TE* is 0 and *TR* is 1, which causes that the functional input value to the TSFF is captured in the flip-flop, while the TSFF output is controlled from the flip-flop. Hence, the TSFF now acts as both observation point and control point. For testing the path between the multiplexers in the TSFF, an additional scan flush test is used with *TE* set to 1 and *TR* set to 0.

The TSFFs are inserted as test points in an iterative process [3][4]. Several testability analysis measures are computed at the beginning of each iteration, including SCOAP, COP, and TC values for each signal line, and the sizes of fanout-free regions. The outcome of the analyses determines which TPI method and cost function are used for inserting test points. TPI stops when the maximum number of test points has been inserted, or when another user-specified constraint has been met such as the target fault efficiency or run-time.

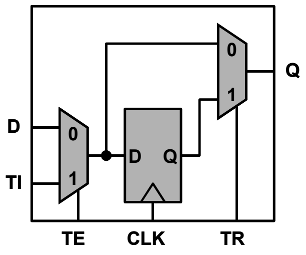

Figure 1: Transparent scan flip-flop (TSFF)

The actual insertion of test points takes place in three steps. The first step is to calculate all locations in the netlist where TSFFs should be inserted, using the TPI method as described above. The second step is to determine the appropriate clock signal for each TSFF, which is required for circuits with multiple clock domains. The third step actually inserts the TSFFs into the netlist, and connects the input and output signals of each TSFF.

3.2 Tool flow

Our tool flow for TPI, scan insertion, ATPG, layout generation, and timing analysis is shown in Figure 2. It includes the following steps:

1. The test points and scan chains are inserted into the gate-level netlist. The scan flip-flops are not connected into scan chains yet.
2. The floorplan of the layout is created and placement is performed. Figure 3a and 3b show the layout after floorplanning and placement. We create a square floorplan for the core area, in which standard cells are placed on horizontal rows. Each cell includes a power strip at the top and a ground strip at the bottom. Placing the cells contiguously on a row with the same alignment therefore creates continuous power and ground strips at the top and bottom of the row. Rows

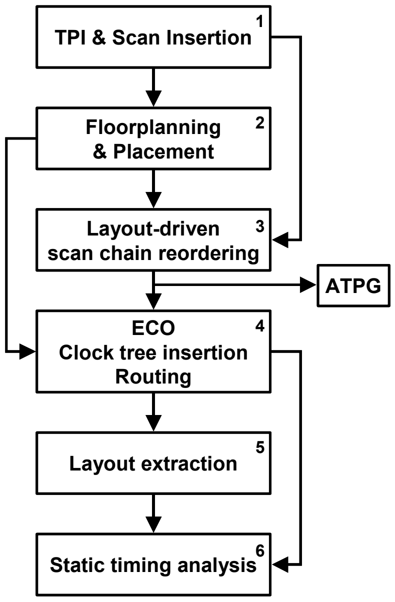
TPI & Scan Insertion 1
Floorplanning & Placement 2
Layout-driven scan chain reordering 3
ATPG
ECO Clock tree insertion Routing 4
Layout extraction 5
Static timing analysis 6

Figure 2: Tool flow

are abutted such that power or ground strips of two consecutive rows are adjacent. An IO ring, ground ring, and power ring are added around the core.

3. Layout-driven scan chain reordering is performed next. The scan flip-flops are assigned to scan chains using cell placement information, such that the wire length for the scan chains is minimized. Buffers and inverters may be added to the scan-enable signals of the scan flip-flops to prevent timing violations. The result is an updated netlist. ATPG is executed on this updated netlist to generate compact test patterns.
4. An ECO is performed on the layout as generated in step 2, such that the changes in the updated netlist of step 3 are included in the layout. Clock trees are inserted, and filler cells are inserted to fill up empty spaces in the rows. Filler cells prevent discontinuities in the power and ground strips at the top and bottom of the rows. Finally, the layout is routed. The resulting layout is depicted in Figure 3c.
5. Capacitances and resistances are extracted from the layout as generated in step 4.
6. Finally, static timing analysis is performed using the extracted capacitances and resistances.

We used the Philips CAT tools for TPI, scan insertion, layout-driven scan chain reordering, and ATPG. We used the Cadence tools SILICON ENSEMBLE DSM for place and route, CT-GEN for clock-tree insertion, HYPEREXTRACT for RC extraction, and PEARL for static timing analysis.

4. Experimental results

4.1 Setup

We performed experiments on ISCAS'89 circuit s38417 [1] and two Philips circuits. Both Philips circuits are cores used in large SoCs: circuit p67883 is a digital control core in a wireless communication IC, and circuit p261909 is a 24-bit DSP core. The gate-level netlists of these circuits are in Philips' 130 nm CMOS standard cell library with six metal layers. Circuit s38417 is mapped to this library by replacing each primitive gate with the corresponding standard cell with minimum drive strength.

We generated six layouts for each circuit: one layout for the circuit without test points, and five layouts for the circuit with 1%, 2%, 3%, 4%, and 5% test points respectively. The percentage of test points corresponds to the number of flip-flops in the design. For instance, circuit s38417 contains 1,636 flip-flops, and inserting 1% test points means that 16 TSFFs are inserted. All flip-flops (including TSFFs) are configured into multiple, balanced scan chains. For each circuit, we analysed the impact of test points on test data, silicon area, and timing.

In order to allow a fair comparison between layouts with and without test points, we always generated square floorplans with the same target row utilization and the same dimensions for power and ground rings. The layouts are optimised for area only, without timing optimisation.

4.2 Impact on test data

Table 1 shows the experimental results on the impact of TPI on test data. Column *#TP* reports the number of inserted test points, *#FF* reports the total number of scan flip-flops, *#chains* reports the number of scan chains, and l_{max} reports the maximum scan chain length. We inserted a variable number of scan chains in circuit s38417 and p67883 with a maximum, balanced length of 100 flip-flops per chain. For circuit p261909 we limited the number of scan chains to 32.

Column *#faults* reports the total number of stuck-at faults in the circuit. The number of faults increases when test points are inserted, since the logic and wires for each test point introduce additional faults.

Column *FC* and *FE* report the fault coverage and fault efficiency. It can be seen that the FC and FE slightly increase when test points are inserted. This is due to the additional faults introduced by the test points, which are relatively easy to detect, and furthermore some redundant faults may become detectable after TPI.

Column *SAF patterns* reports the number of stuck-at ATPG patterns. It can be seen that the number of patterns decreases significantly with TPI, even by 79% for circuit

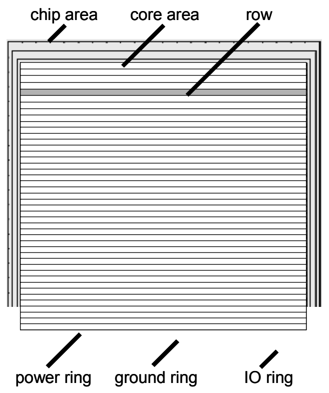

Figure 3: Layout after (a) floorplanning, (b) placement, and (c) routing

Table 1: Impact of TPI on test data

circuit	#TP	#FF	#chains	l_{max}	#faults	FC	FE	SAF patterns		TDV	TAT
						(%)	(%)	(#)	dec. (%)	(%)	(%)
s38417	0	1,636	17	97	89,586	99.68	99.98	92	0	100	100
	16	1,652	17	98	89,740	99.68	99.97	76	17.39	83.64	83.64
	32	1,668	17	99	89,898	99.68	99.98	80	13.04	88.87	88.87
	48	1,684	17	100	90,044	99.68	99.98	72	21.74	80.90	80.90
	64	1,700	17	100	90,199	99.68	99.98	66	28.26	74.25	74.25
	80	1,716	18	96	90,356	99.68	99.98	72	21.74	82.26	77.69
p67883	0	3,653	38	99	242,398	99.38	99.76	136	0	100	100
	36	3,689	38	100	242,709	99.66	99.96	99	27.21	73.72	73.72
	72	3,725	39	98	243,050	99.75	100	97	28.68	72.68	70.82
	108	3,761	39	99	243,371	99.81	100	78	42.65	59.18	57.66
	144	3,797	39	100	243,704	99.81	100	80	41.18	61.28	59.71
	180	3,833	40	98	244,036	99.83	100	73	46.32	56.29	53.47
p261909	0	9,968	32	312	957,832	99.16	99.69	2,539	0	100	100
	99	10,067	32	315	958,734	99.25	99.76	1,024	59.67	40.74	40.74
	198	10,166	32	318	959,670	99.31	99.79	785	69.08	31.54	31.54
	297	10,265	32	321	960,576	99.38	99.85	658	74.08	26.69	26.69
	396	10,364	32	324	961,515	99.40	99.86	619	75.62	25.35	25.35
	495	10,463	32	327	962,415	99.42	99.88	533	79.01	22.03	22.03

p261909 when inserting 5% test point. The reduction is very large when inserting 1% test points, while the gain levels off when inserting more test points. In practice, inserting 1% to 3% test points usually is sufficient.

Column *TDV* reports the test data volume for the scan test stimuli and responses, and *TAT* reports the test application time. The reductions of TDV and TAT are slightly smaller when compared to the reduction of the number of patterns. This is because the test data and test time per pattern slightly increase due to test points. The TDV and TAT are computed by equations 1 and 2, where *n* and *p* correspond to the number of scan chains and test patterns.

$$\mathrm{TDV} = 2 \cdot n \cdot ((l_{\max} + 1) \cdot p + l_{\max}) \qquad (1)$$

$$\mathrm{TAT} = (l_{\max} + 1) \cdot p + l_{\max} \qquad (2)$$

4.3 Impact on silicon area

Table 2 shows the experimental results on the impact of TPI on silicon area. Column *#cells* reports the number of standard cells in the layout. The number of cells increases after TPI due to the TSFFs and additional buffers/inverters in the trees for clock and scan-enable signals.

Column *#rows* reports the number of horizontal rows on which the cells are placed, and column L_{rows} reports the total length of all rows. It can be seen that the number of rows and/or the row length increases when inserting test points. In some cases, the number of rows remains the same while the row length increases. This causes the core area to become slightly rectangular instead of square. The aspect ratio of the core area is always between 0.9 and 1.1.

Column *core area* reports the area for the rows. It can be seen that the core area increases nearly linear with the number of inserted test points. Column *filler cells area* reports the percentage of the core area used for filler cells. When the number of rows does not increase, inserting test points leads to slightly less empty space in the rows. This implies that somewhat higher row utilization is obtained after TPI. We used 97% row utilization as target for circuits s38417 and p67883, and 50% for p261909. A higher row utilization target would lead to routing congestions.

Column *chip area* reports the total area for the core plus the power, ground, and IO ring. The chip area also increases nearly linear with the number of test points. The increase in chip area is sometimes larger than the increase in core area. The chip area is forced to be square, while the core area may become slightly rectangular. In those cases, the chip area contains more empty space, which is not used for placement, but is exploited for routing.

Column L_{wires} reports the total length of all the wires in the layout. It can be seen that the wire length decreases in some cases after TPI. This is due to the fact that separate layouts are generated from scratch for the circuit with and without test points. The core and chip area increase after TPI, which implies that more room is available for wiring. This typically implies that routing becomes easier, which results in shorter wires.

4.4 Impact on timing

Table 3 shows the experimental results on the impact of TPI on timing. Each row reports data on the critical path in a particular layout. Generally, different paths are critical in different layouts. Circuit p67883 contains two clock domains, and results are given for both domains.

Column $\#TP_{cp}$ reports the number of test points inserted in the critical path. Column T_{cp} reports the delay on the critical path, obtained with static timing analysis of the circuit in application mode under worst-case process/temperature/voltage conditions. We blocked all false paths that are only active in test mode, and verified that no hold

Table 2: Impact of TPI on silicon area

circuit	#TP	#cells	#rows	L_{rows}	core area		filler cells	chip area		L_{wires}
				(μm)	(μm^2)	inc. (%)	area (%)	(μm^2)	inc. (%)	(μm)
s38417	0	23,893	93	43,583	214,426	0	1.96	239,248	0	592,853
	16	23,917	93	43,659	214,802	0.17	1.77	240,051	0.34	590,384
	32	23,943	93	43,735	215,177	0.35	1.57	240,855	0.67	595,963
	48	23,965	93	43,811	215,552	0.52	1.39	241,661	1.01	629,464
	64	23,990	93	43,888	215,927	0.70	1.20	242,468	1.35	642,127
	80	24,015	94	44,475	218,818	2.05	2.15	243,478	1.77	590,514
p67883	0	20,895	121	73,324	360,752	0	2.54	392,477	0	1,036,166
	36	20,938	121	73,472	361,484	0.20	2.28	394,020	0.39	1,062,117
	72	20,991	121	73,671	362,461	0.47	2.06	396,081	0.92	1,095,705
	108	21,038	122	74,430	366,194	1.51	2.60	397,631	1.31	1,058,198
	144	21,089	122	74,630	367,179	1.78	2.39	399,702	1.84	1,077,852
	180	21,139	122	74,830	368,163	2.05	2.19	401,519	2.30	1,151,236
p261909	0	104,938	338	566,099	2,785,209	0	49.77	2,874,212	0	9,993,877
	99	105,073	339	568,747	2,798,236	0.47	49.84	2,883,951	0.34	10,177,809
	198	105,220	339	569,720	2,803,022	0.64	49.76	2,893,707	0.68	10,118,329
	297	105,357	340	572,376	2,816,092	1.11	49.82	2,903,480	1.02	10,223,386
	396	105,507	340	573,352	2,820,893	1.28	49.74	2,913,269	1.36	10,079,820
	495	105,640	341	576,017	2,834,005	1.75	49.80	2,923,074	1.70	10,139,882

Table 3: Impact of TPI on timing

circuit	#TP	$\#TP_{cp}$	T_{cp}		F_{max}	T_{wires}	$T_{intrinsic}$	$T_{load\text{-}dep}$	T_{setup}	T_{skew}
			(ps)	inc. (%)	(MHz)	(ps)	(ps)	(ps)	(ps)	(ps)
s38417	0	0	7,195	0	139	16	3,992	3,037	151	0
	16	0	7,779	8.12	129	18	4,062	3,571	150	-23
	32	1	8,095	12.50	124	18	4,364	3,587	154	-28
	48	1	8,289	15.20	121	18	4,378	3,755	152	-16
	64	1	8,445	17.37	118	20	4,394	3,898	150	-18
	80	1	7,946	10.43	126	13	4,297	3,504	153	-22
p67883 (8 MHz)	0	0	24,683	0	41	43	15,912	8,622	132	-26
	36	4	25,469	3.19	39	31	16,552	8,743	132	11
	72	6	25,770	4.40	39	33	17,628	7,980	132	-4
	108	7	26,525	7.46	38	62	17,927	8,393	132	12
	144	7	27,219	10.27	37	68	18,298	8,721	132	0
	180	8	27,496	11.40	36	37	18,453	8,901	132	-27
p67883 (64 MHz)	0	0	4,888	0	205	2	3,283	1,465	133	5
	36	0	5,126	4.86	195	4	3,316	1,684	132	-10
	72	1	5,081	3.94	197	13	2,289	2,640	139	0
	108	0	5,325	8.93	188	3	3,386	1,792	133	12
	144	3	5,099	4.31	196	4	3,068	1,894	132	0
	180	3	6,640	35.84	151	22	3,324	3,162	133	0
p261909	0	0	24,680	0	41	595	8,157	15,816	132	-19
	99	2	25,811	4.58	39	472	8,474	16,851	149	-134
	198	3	24,415	-1.07	41	852	8,881	14,535	145	3
	297	4	25,801	4.54	39	692	9,634	15,526	132	-183
	396	5	24,994	1.27	40	638	10,510	13,738	135	-28
	495	8	27,972	13.34	36	811	10,877	16,149	146	-11

and set-up time violations occur. It can be seen that the delay on the critical path roughly increases linearly with the number of test points. Column F_{max} reports the maximum frequency at which the circuit can run ($F_{max} = 1/T_{cp}$). The 40 MHz target frequency for circuit p261909 is not achieved in all cases after TPI. Both clock domains in circuit p67883 run much faster than 8 MHz and 64 MHz as required for the application, even after TPI. The delay on the critical path is computed according to equation 3:

$$T_{cp} = T_{wires} + T_{intrinsic} + T_{load\text{-}dep} + T_{setup} + T_{skew} \qquad (3)$$

T_{wires} is the delay due to the interconnect wires. The delay through a standard cell is composed of intrinsic delay and load-dependent delay. Intrinsic delay corresponds to the delay when an input signal with near-zero slew is applied without load on the cell output. Load-dependent delay is the additional delay due to the actual signal slew and effective capacitive output load. $T_{intrinsic}$ and $T_{load\text{-}dep}$ in equation 3 are the total intrinsic and load-dependent delay of all cells on the critical path. T_{setup} is the delay due to set-up time for the receiving flip-flop on the path. T_{skew} is the delay due to skew in the clock signals of the sending and receiving flip-flops on the path. It can be seen in Table 3 that the cell delay contributes most. Besides the delay of the TSFF cells, also placement and routing have a considerable impact on the delay of the critical path.

In some rare cases, the circuit becomes faster after TPI, e.g. for circuit p261909 with 198 test points. Although the delay increases due to the inserted test points, shorter wire length may be obtained after TPI, which decreases both wire delay and load-dependent cell delay.

PEARL computes cell delays as a function of input slew and output load values, using look-up tables. The cell delay for a particular slew and load is obtained by interpolating the table values. Slow nodes are cells with large slew and/or load that are outside the look-up table range. Extrapolation is used in these case, which however results in less accurate results. Slow nodes can be resolved by replacing cells with equivalent cells offering larger drive strength or inserting additional buffers/inverters. In our experiments, slow nodes are present in circuit s38417 and p261909 and we did not resolve these. The timing results in Table 3 should therefore not be interpreted as accurate absolute numbers. The results still allow a fair relative comparison of the timing in different layouts of the same circuit with/without test points.

5. Discussion

In our experiments, we optimised for area during placement and routing, and we did not perform timing optimisation. In theory, the circuits could therefore run at higher frequency when performing timing optimisation. Timing optimisation typically implies the use of cells with larger drive strengths and additional buffers and inverters, which comes at the cost of larger silicon area. Timing optimisation for the circuits with TPI would therefore result in layouts that run at the same frequency as the circuit before TPI, but with larger silicon area. However, timing optimisation may also be performed for the circuit without test points. In the latter case, the relative increase of silicon area and delay due to test points may be either larger or smaller than in our experimental results.

Our experimental results show that TPI typically causes new paths to become critical. A common technique for avoiding timing violations is to exclude test points from critical paths with small slack. Our results show that this approach is feasible, but it requires timing analysis for identifying all paths with slack below a certain threshold.

Excluding test points from critical paths lowers the positive effects of TPI on fault coverage and test data. For LBIST, the combination of TPI with DLBIST is therefore attractive [12]. The deterministic pattern generator can be added as a shell around the circuit layout, and it provides that still complete fault coverage is achieved.

6. Conclusion

We presented an experimental investigation on the impact of TPI on circuit size and performance. Our results confirm that TPI is very effective for reducing test data volume and test application time for scan-based test, while slightly increasing the fault coverage. We explored the impact of TPI on placement and routing. Inserting 1% test points increases the silicon area after layout by less than 0.5% while the performance of the circuit may be reduced by 5% or more in case no timing optimisation is performed. The silicon area and the critical path delay both increase nearly linear with the number of inserted test points.

Acknowledgements

The authors acknowledge Leo Sevat and Ad Vaassen at Philips Research Laboratories for their help with layout generation and interpreting the experimental results.

References

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

[2] 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.

[3] 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.

[4] M.J. Geuzebroek, J.Th. van der Linden, A.J. van de Goor, *Test Point Insertion that Facilitates ATPG in Reducing Test Time and Test Data Volume*, Proc. Int. Test Conf., IEEE, 2002, pp. 138-147.

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

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

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

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

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

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

[11] 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.

[12] H. Vranken, F. Meister, H.-J. Wunderlich, *Combining Deterministic Logic BIST with Test Point Insertion*, Proc. European Test Workshop, IEEE, 2002, pp. 105-110.