Test Point Insertion that facilitates ATPG in reducing test time and data volume*

M.J. Geuzebroek[+] J. Th. van der Linden[++] A.J. van de Goor[+]
[+] Delft University of Technology, Faculty of Information Technology and Systems,
Department of Electrical Engineering, Testing Lab,
Mekelweg 4, 2628 CD Delft, The Netherlands
[++] Scientificial, Delft, The Netherlands
Email: M.J.Geuzebroek@ITS.tudelft.nl

Abstract

Efficient production testing is frequently hampered because current digital circuits require test sets which are too large. These test sets can be reduced significantly by means of Test Point Insertion (TPI). The state-of-the-art TPI methods only focus on solving one or two possible testability problems, and sometimes even fail to result in test set size reduction because they focus on the wrong testability problem. In this paper we propose two TPI pre-process methods that analyze the circuit and select the TPI method that will focus on the testability problems that really exist. Experimental results indicate that with these pre-processes better test set size reductions can be achieved.

Gate-delay fault ATPG test sets tend to be even larger than stuck-at fault ATPG test sets. In this paper we have evaluated the impact of TPI on gate-delay fault test sets. Experimental results indicate that TPI also results in a significant test set size reduction for gate-delay fault ATPG.

Keywords: Test point insertion, ATPG, compact test sets, fault coverage, test length, stuck-at faults, gate-delay faults

1 Introduction

The increasing complexity of circuits has a major impact on testing. The demands on the *Automatic Test Equipment (ATE)* become higher and higher, because of higher clock-frequencies and pin counts. This increasing complexity also results in larger test sets, generated by *Automatic Test Pattern Generators (ATPGs)*, resulting in higher ATE vector memory requirements and longer test application times; both resulting in increasing test and ATE costs.

In [1] it was shown that *Test Point Insertion (TPI)* can be used to decrease the ATPG test set sizes. By inserting one or more *test points (TPs)*, a large cone of logic can be bypassed such that it becomes easier for the ATPG to generate test patterns for hard-to-test faults within the circuit. Fewer input assignments are necessary for a test pattern in order to cover the hard-to-test faults. Most of the state-of-the-art TPI methods [2, 3, 4, 5, 6] are only focused on improving the *Pseudo-Random (PR)* fault coverage for a circuit and are not specifically aimed at reducing ATPG test set sizes. [1] introduced an improved version of the Hybrid *C*ost *R*eduction *F*actor (*CRF*) TPI method [5], such that the TPI algorithm becomes more optimized to facilitate ATPG and experimental results showed that with this improved TPI method even better test set size reductions could be achieved than with the traditional state-of-the-art TPI methods.

Every circuit has its own characteristics and therefore also its own testability problems. Some circuits suffer from many *Random Pattern Resistant (RPR)* faults, while other circuits suffer from signal lines with high *Test Counts* ([7, 1]) as described in Section 2. It is also possible that the circuit suffers from a combination of testability problems. The (improved) Hybrid CRF TPI method uses a *Cost Function (CF)* to determine where in the circuit TPs should be inserted. It is important that the TPI algorithm uses a CF that tries to improve the circuit's testability problems that really exists. Using, for example, a test count based CF, while the circuit does not suffer from high test counts, does not seem to be useful.

In this paper we will propose a TPI pre-process that selects the CF used in the Hybrid TPI algorithm that tries to solve the hardest test problem that exists in the circuit. *Testability Analysis (TA)* measures are used to find these

*This work was funded by Philips Semiconductors N.V, The Netherlands

test problems.

On some very large industrial designs, TPI did not help reducing the compact ATPG test set sizes, even when the proposed CF selection method is used. These problem circuits often contain very large *fanout-free regions (FFRs)*. Experimental results have shown that also the Hybrid CRF TPI method with CF selection often does not insert TPs within these large FFRs, resulting in hardly any test set size reduction. In this paper we propose a second pre-process that analyzes the circuit to determine whether it contains very large FFRs. When such FFRs are found, the TPI algorithm will first insert TPs to reduce the sizes of the large FFRs, before it inserts TPs using the Hybrid CRF TPI method.

By inserting TPs in the very large FFRs and by using the pre-process to select the CF that targets the hardest test problem, even better compact ATPG test set size reductions are possible than with the state-of-the-art TPI methods, including the improved Hybrid TPI method of [1].

The ATPG test sets for gate-delay faults are generally even larger than the test sets for stuck-at faults. Also the test generation times for gate-delay faults are much larger than for stuck-at faults. In this paper we will show that TPI also results in significant ATPG test set size and generation time reductions for gate-delay faults.

Section 2 describes the TA measures used to locate the testability problems of a circuit. Section 3 describes the Hybrid CRF TPI method. Section 4 describes the CF selection procedure and shows corresponding experimental results. Section 5 describes TPI techniques used to reduce large FFRs and shows corresponding experimental results. Section 6 shortly describes the differences between stuck-at fault ATPG and gate-delay fault ATPG and the possible impact of TPI on gate-delay fault ATPG. Next, the experimental TPI results for gate-delay ATPG show reduced test set sizes and generation times. Finally, in Section 7 the conclusions are given.

2 Testability analysis measures

In the following subsections, several TA measures are described that can be used to find the test problems in a circuit. They are used by the TPI algorithms to find the spots in the circuit where a TP should be inserted.

2.1 Controllability Observability Program (COP)

COP [8] is a TA measure which is very popular in TPI algorithms [2, 9, 5, 6] that are used to increase PR fault coverages of circuits, i.e., for *Built-In Self-Test (BIST)*. For each signal line *n*, the COP measures give controllability estimates $C0_n$ and $C1_n$ expressing the probability that *n* will be 0 or 1, and an observability estimate W_n expressing the probability that a value discrepancy (fault effect) at *n* will be observable at a primary output. Given the controllabilities and observability probabilities for a signal line *n*, the testability (or detection probability) estimate $Pd_{n/V}$ for the stuck-at *V* fault at *n*, can be found with Equation 1. The detection probability for a fault is the probability that a PR pattern will cover that fault.

$$Pd_{n/V} = C\overline{V}_n \cdot W_n \quad (1)$$

The COP based TPI algorithms try to improve the detectability probabilities of the faults which have very low detectability probabilities. TPI does not only increase the PR fault coverage, but it becomes also easier for an ATPG tool to generate test patterns for the 'former' faults with low detectability probabilities.

2.2 Sandia Controllability/Observability Analysis Program (SCOAP)

SCOAP [11] is a TA measure that is very popular in ATPGs. The SCOAP controllability of a signal line is an estimate for the number of inputs that have to be set in order to get a 0(1) on that line. The SCOAP observability of a line is an estimate for the number of inputs that have to be set to propagate a fault-effect from that line to an output. Besides the number of inputs that have to be set, SCOAP also takes into account how 'deep'(level) the line is in the circuit and the number of fanout branches that will also be assigned due to the input assignments.

ATPGs try to choose the paths with the smallest SCOAP values, because the SCOAP values suggest that these paths require the fewest number of input assignments. When TPI is used to reduce the SCOAP values for lines, the ATPG can generate test patterns with fewer assigned inputs for the faults corresponding to these lines, such that more test patterns can be compacted into one single test pattern, resulting in better test set size reduction. The SCOAP values are far from exact, because they also take into account the level of signal lines in the circuit and because of reconvergent fanout. However, they give a good indication how difficult it is to control a signal line or to make it observable compared to other signal lines.

2.3 Test counts (TCs)

The test counts (TC) [7, 10] of a signal line give a lower bound on the number of tests (fault effects) that have to pass through that signal line such that all faults in its fan-in cone can be covered. Test counts are divided into:

Essential zeros (ones), E0(E1): The minimum number of

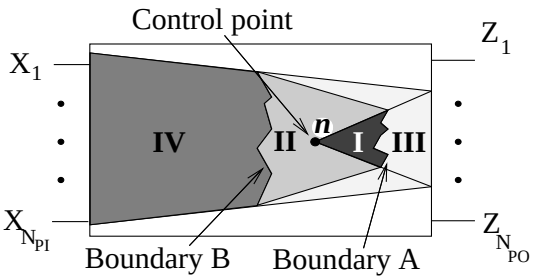

Figure 1. The Hybrid CRF method [5].

times a line must become 0(1) (during the application of a test set) and be observable on an output, such that all corresponding faults in its fanin-cone can be covered.
Total zeros (ones), T0(T1): The minimum total number of times a line must become 0(1) (during the application of a test set) (either observable or making other lines observable).

A signal line with high TCs (i.e., total counts) indicates that many test patterns have to pass through this line. When TPI is used to lower the TCs of signal lines, fewer test patterns have to pass through that line in order to cover all faults in its fan-in cone, which will very likely result in smaller test sets.

3 Hybrid CRF TPI method

The Hybrid CRF method uses a CF to determine the testability of a circuit. A test cost estimate is determined for each stuck-at fault in the circuit. This cost is calculated using TA measures as described in Section 2. Subsection 3.1 gives a short description of the original Hybrid CRF method found in [5]. Subsection 3.2 shortly describes ten other CFs that can be used by the HCRF method. Each of these CFs targets different testability problems in the circuit.

3.1 The original Hybrid CRF algorithm

The CF used in the original Hybrid CRF algorithm only relies on the COP detectability probabilities, described in Section 2. The cost contribution for the stuck-at *V* fault at *n* can be calculated with Equation 2. The global cost of the circuit is the sum of the cost contributions of all faults in the circuit.

$$K_{n/V} = 1/Pd_{n/V} \quad (2)$$

For each TP candidate, an event-driven mechanism is used to calculate the impact of the TP candidate on the global cost. This mechanism is illustrated in Figure 1 and is briefly described below; a more thorough description can be found in [5]. Given a TP candidate at line *n*, the event-driven mechanism will propagate the COP controllability/observability changes (and hence the cost changes) caused by the TP (Regions **I** and **II** in Figure 1). When these changes drop below a threshold (Boundaries **A** and **B**), the COP changes are not propagated and explicitly recalculated any further, instead a cost estimate is used to reflect the impact of the TP on the remaining part of the circuit (Regions **III** and **IV**). The cost reduction estimate for each TP candidate is calculated and finally the TP candidate with the highest cost reduction is the TP that will be inserted.

3.2 Cost functions targeting different test problems

Combining the CFs corresponding to the COP, SCOAP and TC measures, that are found in literature [5, 1], results in the eleven CFs shown in Table 1. Column *Based on* shows the TA measures on which the CF is based and which the TPI algorithm will try to improve. I.e., CF 1 only tries to improve the COP detectabilities, while CF 10 tries to improve the COP, SCOAP and TC values. In the CFs, *SC0* represents the SCOAP0-controllability and *SW* the SCOAP observability. *NPAT* is a parameter that can be used to tune the TPI algorithm as described below. CF *1* is used in the original Hybrid CRF algorithm [5] and CF *7* in the improved Hybrid CRF method [1]. The pre-process described in Section 4 tries to select from these eleven CFs, the CF that results in the best testability improvement, for a given circuit.

The state-of-the art TPI methods use different COP-based CFs. [2, 5] prefer CF *1*, while [3, 6] prefer CF *2*. The main difference between these two CFs is that with CF *1*, TPI will primarily focus on the hardest-to-detect fault(s), while with CF *2*, TPI will focus on a set of hard-to-test faults as is clarified in Table 2.

Given the faults listed in column *Fault* and their corresponding detectability probabilities listed in Column Pd_f. The cost contributions for these faults are listed in Columns *1* and *2* for respectively CFs *1* and *2*. In case CF *1* is used, the cost contributions of faults *f2-f4* are insignificant compared to fault *f1* and the TPI algorithm will only focus on improving the detectability of fault *f1*. In case CF *2* is used, the cost contributions of faults *f2* and *f3* are also significant compared to fault *f1* and the TPI algorithm will focus on improving the detectability of these three faults. Only fault *f4* is ignored, because its detectability probability is already high.

NPAT originally represented the number of PR patterns that will be applied [6]. CF *2* can be seen as the probability that fault *f* will not be detected after applying *NPAT* patterns. Although we do not know the number of test patterns, we can use *NPAT* as a parameter to tune the TPI process. The

Table 1. CFs for solving the test problems in a CUT*)

#	Cost function	Based on
1	$\frac{1}{Pd_{n/1}}$	COP
2	$(1-Pd_{n/1})^{NPAT}$	COP
3	$(SC0_n + SW_n)$	SCOAP
4	$\frac{SC0_n+SW_n}{Pd_{n/1}}$	COP & SCOAP
5	$(SC0_n + SW_n)\cdot(1-Pd_{n/1})^{NPAT}$	COP & SCOAP
6	$(T0_n)$	TC
7	$\frac{T0_n-E0_n}{C0_n} + \frac{E0}{Pd_{n/1}}$	COP & TC
8	$(T0_n - E0_n)\cdot(1-C0_n)^{NPAT} + E0_n\cdot(1-Pd_{n/1})^{NPAT}$	COP & TC
9	$(T0_n - E0_n)\cdot(SC0_n) + E0_n\cdot(SC0_n + SW_n)$	SCOAP & TC
10	$\frac{(T0_n-E0_n)\cdot SC0_n}{C0_n} + \frac{E0_n\cdot(SC0_n+SW_n)}{Pd_{n/1}}$	COP & SCOAP & TC
11	$(T0_n - E0_n)\cdot SC0_n\cdot(1-C0_n)^{NPAT} + E0_n\cdot(SC0_n+SW_n)\cdot(1-Pd_{n/1})^{NPAT}$	COP & SCOAP & TC

*) Only the stuck-at 1 parts of the CFs are listed.

Table 2. Difference between CFs *1* and *2*

Fault	Pd_f	Cost function 1	Cost function 2^*
$f1$	$4\cdot10^{-9}$	$2.5\cdot10^{8}$	1
$f2$	$9\cdot10^{-6}$	$1.1\cdot10^{5}$	0.99
$f3$	$3\cdot10^{-5}$	$3.3\cdot10^{4}$	0.97
$f4$	$2\cdot10^{-2}$	50	0

$^*NPAT = 1024$

cost contribution of faults can be divided into three sets of faults, visualized in Figure 2:

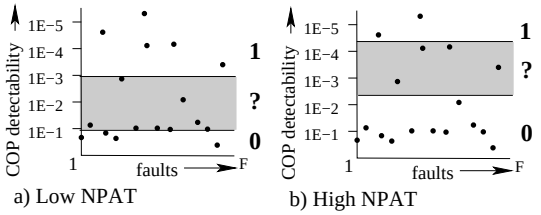

Figure 2. NPAT-based cost contributions

1. Faults with high COP detectabilities for which the cost contribution is (almost) 0, (Region **0** in Figure 2). The TPI algorithm will ignore these faults.
2. Faults with low COP detectabilities for which the cost contribution is (almost) 1, (Region **1**). The TPI algorithm will target on these faults.
3. Faults with COP detectabilities for which the cost contribution lies somewhere between 0 and 1 (Region **?**). The TPI algorithm will partially also target these faults.

Increasing *NPAT* in $(1-Pd_f)^{NPAT}$ results in shifting Region **?** more towards Region **1**, hence enlarging Region **0** while decreasing Region **1**; more faults will have a cost contribution of 0 and fewer faults will be targeted by the TPI algorithm. By changing the NPAT value, we can select the set of faults on which the TPI algorithm should focus.

4 TPI pre-process for cost function selection

Instead of inserting all TPs using a single CF, we propose to split the TPI process into multiple stages. In each stage, first a TA run is performed on the CUT (including already inserted TPs from previous stages). Based on the current TA measures, the CF is chosen which targets the hardest test problem in the CUT. Several TPs are inserted using this CF. In the next stage, a TA run is performed again to select a new CF that is used for the next set of TPs.

The CF selection procedure is described in Subsection 4.1 and the corresponding experimental results are shown in Subsection 4.2.

4.1 The cost function selection procedure

During the TA run, priorities are assigned to the COP, SCOAP and TC measures. A high priority for a TA measure, indicates that the test problem corresponding to this TA measure is high. For the different TA measures, the following priority schemes are used:

COP priority:

The COP priority is determined by the number of faults with a COP detectability (Pd_f) below a given threshold value. The more faults with a Pd_f below this threshold, the higher the priority. The priority is also higher, when there are faults with extremely low Pd_fs, e.g., below 1E-10, to make sure that the TPI algorithm will target these very hard-to-test faults, even when there are only a few of them.

SCOAP priority:

The SCOAP priority is determined in a similar way as

the COP priority. The number of faults with SCOAP values higher than a given threshold determine the priority. As mentioned in Section 2, relative SCOAP values (SCOAP values of a given signal line compared to SCOAP values of other lines) are more useful than absolute SCOAP values. Therefore the threshold is determined by the distribution of the SCOAP values and is set at $\overline{SCOAP} + 3 \cdot \sigma_{SCOAP}$ (standard deviation). This threshold equation has been chosen experimentally.

TC priority:

For the TC only two priority values are used, 0 and 1. Experiments with more priority values have not taken place. Again a given threshold is used to differentiate between these priorities. When there are TC values higher than this threshold, the priority is 1, otherwise 0. This threshold has experimentally been set at 40 (T0+T1).

When the COP priorities are higher than the SCOAP priority, a COP-based CF is selected. If the SCOAP priority is higher, a SCOAP based CF is selected. If COP and SCOAP are equally high, a COP and SCOAP based CF is selected. The CF will also be TC based, when the TC-priority is 1.

The selection between a $1/pd_f$ or $(1 - pd_f)^{NPAT}$ COP based CF is determined by the lowest pd_f found in the circuit. Experiments have shown that better results are achieved with a $1/pd_f$ CF when there are extremely low pd_fs in the circuit, otherwise $(1 - pd_f)^{NPAT}$ should be used. The NPAT value is set such that at least 5% of all faults contribute to the CF (non-zero value for $(1 - pd_f)^{NPAT}$).

It can be imagined that it is difficult to describe and determine the exact priority assignment and all existing threshold values. Therefore, most of these values are experimentally determined on a trial and error basis. In the next subsection it will be shown that with this TPI CF selection procedure, very good compact ATPG test set size reductions can be achieved.

4.2 Experimental TPI results for CF selection

This subsection shows experimental results for the CF selection procedure. Before TPI, the CF selection procedure is run to select the best CF. With this CF a set of four TPs[1] will be inserted. After that, the CF selection procedure is run again to find the CF for the next set of four TPs.

Table 3 shows the compact ATPG test set sizes for the ISCAS benchmark circuits [12, 13]. The column *Circuit* lists the circuit name (the number corresponds to the number of signal lines), the column *No TPI #Pat.* gives the number of compact ATPG test patterns without TPI and the column *#TP* lists the number of inserted TPs. The following three columns show the number of ATPG test patterns after TPI, using respectively CF 2 and 7 ([1]) and the CF selection procedure. CF 2 is used to show what can be achieved if TPI is only focused on the PR detectability. CF 7 is used to show what can be achieved when also the TCs are taken into account. Although not listed in the table, TPI has minimal impact on the achieved ATPG fault coverages. It slightly improved after TPI, because several ATPG untestable faults become testable. These experimental results show that by using the CF selection procedure, better test set size reduction has been achieved. With CF 2 and 7, respectively reductions from 39%(from 1938 to 1194) and 40%(from 1938 to 1165) are achieved, while with the CF selection procedure, a reduction of 45%(from 1938 to 1080) is achieved. Although the CF selection procedure does not always result in the best test set size reduction, its results are very consistent and never poor. Especially the circuits with larger test sets without TPI, these are the circuits for which test set size reductions are most wanted, benefit the most from the CF selection procedure; for circuit s13207.1 almost twice the reduction has been achieved with the CF selection procedure 36%(from 274 to 176), compared to 19% (from 274 to 222) for CF 2, respectively 19% (from 274 to 223) for CF 7.

Table 3. TPI with CF selection results for ISCAS circuits

Circuit	No TPI #Pat	#TP	Cost function 2	Cost function 7	Cost function Selection
c880	30	10	23	23	23
c1355	89	16	22	22	21
c1908	120	25	37	41	36
c2670	58	25	26	27	28
c3540	116	25	74	71	70
c5315	75	30	46	43	41
c6288	32	30	33	26	27
c7552	124	40	65	76	59
$\sum_{sub}$	644	201	326	329	305
s1196	133	20	45	51	51
s1423	39	20	19	19	19
s1488	114	20	92	88	81
s1494	112	20	89	90	82
s5378	117	30	56	59	58
s9234.1	139	50	51	49	53
s13207.1	274	50	222	223	176
s15850.1	147	50	103	77	81
s35932	15	12	14	16	15
s38417	93	50	73	71	68
s38584.1	124	50	122	112	98
$\sum_{sub}$	1307	372	886	855	782
$\sum_{total}$	1938	573	1194	1165	1080

Table 4 shows the experimental results for several indus-

[1] 4 TPs are chosen because experiments showed that, for most circuits, a set of 4 TPs resulted in the best compact test sets

Table 4. TPI with CF selection for industrial circuit

Circuit	No TPI #Pat	#TP	Cost function 2	7	Selection
p5973	56	8	38	46	42
p7653	341	15	155	176	165
p13138	66	20	62	63	60
p27811	200	38	167	100	83
p34592	240	34	167	175	167
p43282	401	49	219	206	198
p43663	194	50	96	127	112
p71553	129	71	98	131	102
p73133	472	50	339	280	277
p75344	279	55	198	173	181
p90685	598	90	396	380	406
p93140	554	93	368	367	330
p104649	1030	104	750	731	602
p114605	687	80	533	402	391
p137498	436	137	284	279	306
$\sum_{total}$	5683	894	3870	3636	3422

trial Philips circuits. For these industrial circuits, the reduction percentages for CFs 2, 7 and the CF selection procedure are respectively 32%(from 5683 to 3870), 36%(from 5683 to 3636) and 40%(from 5683 to 3422). Again CF 7, which takes into account TCs results in better test set size reduction than CF 2. However, when the CF is chosen based on the TA measures of the circuit, even better results are obtained. Similar to the ISCAS results of Table 3, the CF selection procedure especially results in the best test set size reduction for the circuits that suffer from large test sets without TPs the most, i.e., circuits p43282, p93140, p104649 and p1114605. Especially for circuit p104649, which has the largest test set without TPs and for which test set size reduction is most wanted, the CF selection procedure results by far in the best test set size reduction. With the CF selection procedure, the test set size is reduced from 1030 to 602(42%) compared with CF 2, to 750 (27%) and CF 7, to 731(27%).

5 TPI for reducing large fanout-free regions

A circuit with very large FFRs will very likely suffer from high TCs and hence large test sets. All faults in that FFR have to pass through the single output of the FFR. Many test patterns are required to make sure that all fault-effects in the FFR become observable on the FFR-output at least once; which can have a significant impact on the total size of the ATPG generated test set.

The high TCs can be reduced by TPI, which splits these large FFRs into two FFRs with fewer inputs. This is achieved by inserting a transparent *scan flipflop (SFF)*[2] and is illustrated in Figure 3. The *BIG_FFR* in Figure 3.a is split into two smaller FFRs, *FFR1* and *FFR2*, by the SFF at line *C*. In this example, the SFF almost reduces the FFR sizes and TCs into half the size and the TCs of the original *BIG_FFR*. Methods to split these FFRs are described in

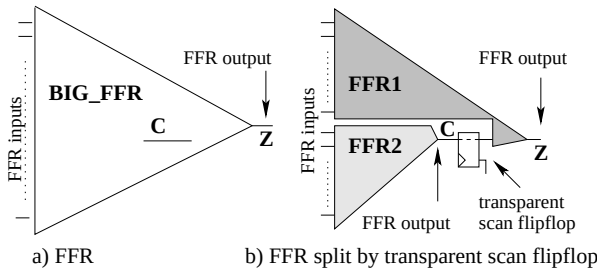

Figure 3. Splitting of an FFR

Subsection 5.1 and experimental results are shown in Subsection 5.2.

5.1 TPI method for reducing large FFRs

We have extended the TPI algorithm with a pre-process that searches for large FFRs in the circuit. If it finds such FFRs, the TPI algorithm will first insert TPs within these FFRs. In the following text four different methods used to split the FFRs are described. The size of a FFR will correspond to the number of inputs of the FFR.

The first question which has to be solved is the definition of a 'large FFR'. We use two definitions for a large FFR:
A: An FFR is large when it has more than `FFR_LARGE_SIZE` inputs. `FFR_LARGE_SIZE` is a user-defined value.
B: An FFR is large when it has more than $\overline{FFR}$ + 3*σ^{FFR} inputs. $\overline{FFR}$ is the average size of an FFR in the circuit and σ^{FFR} is the standard deviation of the FFR size distribution. Because most of the FFRs only have a small number of inputs, $\overline{FFR}$ and σ^{FFR} are only determined by the larger FFRs ($\geq$ 10 inputs).

As already described, large FFRs usually have high TCs. This has led to two strategies to reduce the FFR sizes:
1: The TP candidate that splits the FFR into two smaller FFRs of which the number of inputs are closest to each other, i.e., two approximately equal sized FFRs, is the TP that will be inserted.
2: The TP candidate that splits the FFR into two smaller FFRs with both the lowest TC values, is the TP that will be inserted.

These definitions and strategies are combined into four methods to reduce large FFRs, i.e., methods **A1**, **A2**, **B1**

[2] A transparent scan flipflop acts as a scan flipflop in test operation mode and as a buffer in normal operation mode

and **B2**.

5.2 Experimental results of TPI for reducing large FFRs

The experimental results of the four TPI methods for reducing large FFRs (TPI_{FFR}) described in the previous subsection are shown in Table 5. The stuck-at fault ATPG test set sizes before and after TPI are listed for the ISCAS89 circuit s9234.1 [13] and several Philips circuits; circuits which have large FFRs. The experiments were run on a 700 Mhz AMD Athlon (200Mhz FSB) using the ***D**elft **A**dvanced **T**est Pattern Generation Platform (DAT)*[14] ATPG tool. The column *Circuit* lists the circuit name, the column *#Pat.* gives the number of ATPG test patterns and the column $\textit{FFR}_{max}$ gives the size of the largest FFR. The columns $\textit{\#TP}_{tot}$ and $\textit{No TPI}_{FFR}$ *#Pat.* show respectively the total number of inserted TPs and the test set sizes after TPI in case no TPI_{FFR} method has been used. In the next columns the number of TPs ($\textit{\#TP}_f$) that are inserted using the TPI_{FFR} method and the test set sizes (*#Pat.*) after TPI are shown, for each of the four TPI_{FFR} methods. Note that the listed test set sizes are the sizes after $\textit{\#TP}_{tot}$ TPs are inserted. The remaining $\textit{\#TP}_{tot}$-$\textit{\#TP}_f$ TPs are inserted using the Hybrid CRF method.

The experimental results in Table 5 show that TPI results in significant test set size reductions. Without TPI_{FFR} the test set sizes are already reduced, from 18137 to 10198 patterns (45% reduction), but with TPI_{FFR} enabled, the reductions become even better, the test set sizes are between 9186 patterns (50% reduction) and 8671 patterns (53% reduction). Most of the circuits in Table 5 do not contain extremely large FFRs or suffer from very large test sets. For these circuits, only a few TPs are inserted by TPI_{FFR} and the extra reduction is small. However, the impact of TPI_{FFR} on the circuits which do contain very large FFRs and suffer from high test sets (circuit p73257, p104649, p160257 and p598004) is very large and for these circuits, TPI_{FFR} results in a significantly better test set size reduction than without TPI_{FFR}. Circuit p596922 has a large test set, but the test set size problem in this circuit is caused by many faults having very low detectability probabilities. The TPI_{FFR} results are worse compared to TPI without TPI_{FFR}, because fewer test points are inserted in order to improve the detectability probabilities.

The reductions of TPI_{FFR} methods A1 and A2 are better than those of methods B1 and B2. This is mainly due to the difference in reduction for circuit p73257. This circuit contains over 50 FFRs with more than 200 inputs and because of this, $\overline{FFR}+3\cdot\sigma^{FFR}$ is larger than 269 (the largest FFRs) and no TPs are inserted by TPI_{FFR}. Although this circuit only contains semi-large FFRs, it does suffer from a large test set and the results of TPI_{FFR} methods Al and A2 show that TPI_{FFR} can still result in significantly better test set size reduction.

Using FFR size definition **B** (methods B1 and B2) results in more inserted TPs in the larger circuits compared to definition **A** (method A1 and A2). The larger circuits have so many small FFRs that σ^{FFR} becomes small. As a result, $\overline{FFR}+3\cdot\sigma^{FFR}$ is small and even an FFR with not too many inputs will be defined 'large' according to definition **B**. For some larger circuits, e.g., circuit p114605 and p705050, it happens that too many TPs are inserted using the B1 and B2 TPI_{FFR} methods, even when the FFR sizes are no issue any further.

In p705050, only two TPs are inserted with TPI_{FFR} methods A2 and B2, while methods A1 and B1 insert more TPs. After two TPs, none of the TP candidates within the largest FFR results in a TC decrease, hence no more TPs are inserted with TPI_{FFR} methods A2 and B2.

6 TPI for facilitating gate-delay fault ATPG

Besides the impact of TPI on *stuck-at fault (SAF)* ATPG test set sizes, we have also tested the impact of TPI on *gate-delay fault (GDF)* ATPG. The GDF test sets tend to be even larger than the SAF test sets and hence also form a big problem for the ATE. It would be nice if TPI could also result in significant GDF test set size reductions.

Subsection 6.1 starts with a short description of GDFs and GDF ATPG, including the possible impact of TPI on GDF ATPG. Subsection 6.2 shows experimental results of TPI on GDF ATPG.

6.1 Gate-delay faults and gate-delay fault ATPG

Two kinds of delay fault models exist [15]:

1. Path-delay fault model: A path is defined faulty if it fails to propagate a transition from the path input to the path output within a specified time interval.

2. Gate-delay fault model: A gate is defined faulty if its gate defect results in at least one path-delay fault.

The frequently used GDF model is the slow-to-rise and slow-to-fall delay fault at a gate input or output. The GDF model is a localized fault model, similar to the SAF model, because it assumes an isolated fault at a distinct gate. In the following text it is assumed that a GDF is that large that the path does not matter. The gate-delay test problem is similar to a two time frame sequential stuck-at test problem, see Figure 4. In the initial time frame a vector is generated to initialize the required transition at the faulty node. In the final time frame, a stuck-at test is generated to provoke the required transition at the faulty node and to sensitize a path to a circuit output. Both time frames are connected through flipflops. Hence, during GDF ATPG, two test generation targets need to be specified: A SAF in the "final time frame"

Table 5. Experimental results of the TPI for reducing large FFRs methods.

Circuit	#Pat.	FFR_{max}	$\#TP_{tot}$	No TPI_{FFR} #Pat.	Method A1		Method A2		Method B1		Method B2	
					$\#TP_f$	#Pat.	$\#TP_f$	#Pat.	$\#TP_f$	#Pat.	$\#TP_f$	#Pat.
s9234.1	128	165	50	54	1	50	1	50	1	50	1	50
p13616	138	165	7	137	5	134	5	135	0	137	0	137
p14148	420	170	15	258	3	261	3	263	1	232	1	232
p31025	621	137	25	446	1	435	1	435	4	432	4	427
p72767	466	169	50	263	2	304	2	304	2	304	2	304
p73257	2016	269	73	969	54	561	54	581	0	969	0	969
p73133	454	169	50	247	2	252	2	252	2	252	2	252
p75344	277	181	55	169	1	171	1	171	5	170	5	170
p90685	539	474	90	364	11	347	11	347	24	339	24	331
p93140	512	474	93	317	11	325	11	325	24	331	24	327
p104649	1097	244	104	628	10	520	10	515	14	521	14	525
p114605	649	131	80	360	2	372	2	357	58	453	59	457
p128523	432	148	128	390	1	384	1	384	5	372	5	372
p137498	435	433	137	289	2	287	2	286	8	291	8	288
p162057	2072	1518	162	475	30	242	30	249	46	252	45	260
p445506	644	334	200	569	88	587	3	581	80	582	3	581
p596922	2606	467	317	667	26	703	26	707	26	703	26	707
p598004	2245	1518	100	2171	39	1391	39	1392	39	1391	39	1392
p608978	1226	272	250	678	59	572	59	606	187	597	132	609
p705050	663	256	250	287	24	307	2	292	132	370	2	292
p854266	677	292	300	460	18	466	18	461	80	438	42	441
$\sum_{total}$	18317	7986	2536	10198	390	8671	283	8693	738	9186	438	9123

part and an I_{DDq} modeled fault[3] in the "initial time frame" part.

As a result, one can imagine that it becomes even harder for the GDF ATPG to find tests than for the SAF ATPG, when the circuit contains hard-to-control and hard-to-observe nodes. A TP will make nodes easier controllable/observable in *both* time frames, which should result in significant test set size and generation time reduction.

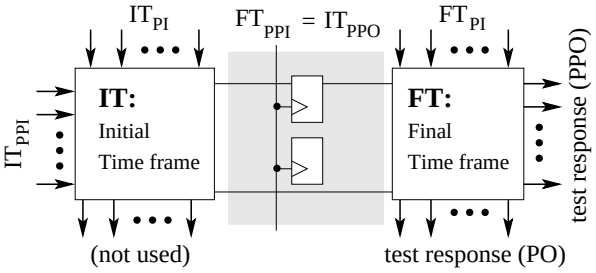

Figure 4. Two time frame model for GDF ATPG

6.2 Experimental results of TPI for GDF ATPG

This subsection shows experimental results of TPI for GDF ATPG. Table 6 shows the GDF test set sizes and generation times for several ISCAS89[13] and industrial circuits (Philips), before and after TPI, using the improved Hybrid TPI algorithm of [1] with TPI_{FFR} method B2. The experiments have been run on a 1200Mhz AMD Thunderbird Processor (266Mhz FSB).

Columns 2-5 (*GDF ATPG (No TPI)*) show respectively the number of GDF patterns (*#Pat.*), the fault coverage (*FC*, the percentage of *all* faults that are covered), the fault efficiency (*FE*, the percentage of *detectable* faults that are covered) and the ATPG CPU-time ($time_{ATPG}$) for the circuits listed in the column *circuit*, before TPI. Columns 8-11 (*GDF ATPG (TPI)*) show these results after TPI. Columns 6 and 7 show respectively the number of inserted TPs (*#TP)* and the CPU-time spent on TPI ($time_{TPI}$). The experiment was set up in such a way that the ATPG tool stops generating patterns for the circuit with TPs, as soon as the same FE [4] had been reached as for the circuit without TPs. This has been done such that we can measure the impact of TPI on the number of GDF patterns for a given FE.

The results presented in Table 6 show that TPI has a large impact on the number of GDF ATPG patterns. With TPI, the total number of GDF test patterns has been reduced from 12142 to 7884 test patterns (35% reduction). The reduction for the ISCAS89 circuits (37%) is almost the same as for the industrial circuits (34%). Especially the circuits with large GDF test sets benefit from TPI, just as is the case when TPI is used to reduce the SAF ATPG test set sizes [1]. The test set for circuit p104649 has even been reduced with 57% from 1895 to 833 patterns. The test set for circuit s35932 is so small that TPI does not result in a reduction.

Besides a reduction in the number of patterns, also a

[3] The fault only needs to be justified, not to be propagated like for an I_{DDq} fault; therefore the term I_{DDq} fault is used

[4] Due to ATPG tool limitation, it was not possible to stop at the exact same FE

Table 6. Experimental results of TPI to facilitate GDF ATPG

Circuit	GDF ATPG (No TPI)				TPI		GDF ATPG (TPI)			
	#Pat.	FC(%)	FE(%)	time$_{ATPG}$	#TP	time$_{TPI}$	#Pat.	FC(%)	FE(%)	time$_{ATPG}$
s1196	163	96.73	96.82	1.98s	20	1.60s	81	96.37	96.51	1.12s
s1423	73	86.74	97.61	1.61s	20	1.28s	35	87.34	95.61	0.77s
s1488	154	90.11	97.65	6.13s	20	1.98s	100	94.77	94.98	1.87s
s1494	153	89.50	97.58	6.20s	20	2.02s	97	94.04	94.39	1.87s
s5378	139	88.17	96.61	10.17s	30	6.27s	90	92.21	97.11	5.92s
s9234.1	238	81.90	93.30	61.47s	50	22.95s	85	85.62	92.90	15.70s
s13207.1	388	84.79	97.33	105.76s	50	36.09s	283	88.75	97.31	63.47s
s15850.1	216	81.44	96.31	75.49s	50	35.19s	107	85.72	95.56	44.37s
s35932	29	85.27	99.29	192.63s	12	47.47s	31	85.26	99.27	192.39s
s38417	163	95.39	97.34	145.25s	50	76.69s	121	95.11	97.28	99.41s
s38584.1	246	90.19	97.85	326.77s	50	74.85s	200	91.22	97.66	280.30s
$\sum_{sub}$	1962	88.51	97.54	933.46s	372	306.39s	1230	89.86	97.32	707.19s
p5973	97	91.67	94.04	106.24s	8	2.60s	48	91.41	93.74	20.11s
p7653	715	92.06	96.20	140.52s	15	9.88s	493	92.30	96.48	82.04s
p13138	153	96.20	96.74	31.77s	20	11.92s	115	96.34	97.14	22.29s
p27811	282	82.91	97.21	327.04s	38	40.85s	191	85.67	97.20	226.02s
p34592	304	98.57	98.59	471.24s	34	55.48s	257	98.90	98.92	452.50s
p43282	705	93.00	96.95	1240.18s	49	142.46s	480	93.78	96.73	797.25s
p43663	301	94.44	99.48	190.77s	50	64.59s	244	94.73	99.48	157.68s
p71553	213	96.68	99.92	34.08m	71	8.92m	130	96.63	99.70	69.86m
p73133	630	93.45	97.34	91.78m	50	5.54m	316	93.55	97.10	48.59m
p75344	514	92.43	97.68	1578.58s	55	157.93s	349	93.78	97.56	1227.58s
p90685	229	60.89	70.41	567.16m	90	17.05m	148	61.34	70.34	457.19m
p93140	650	93.90	99.22	92.65m	93	16.27m	523	94.00	99.23	69.57m
p104649	1895	67.71	89.47	624.26m	104	7.35m	833	68.67	89.59	314.54m
p114605	2361	95.33	98.86	338.98m	80	9.98m	1534	95.49	98.86	187.63m
p137498	1130	92.09	99.07	165.60m	137	11.27m	993	92.38	99.08	219.39m
$\sum_{sub}$	10180	87.58	94.79	33.04h	894	84.46m	6654	88.06	94.76	23.61h
$\sum_{total}$	12142	87.72	95.19	33.30h	1266	89.57m	7884	88.33	95.14	23.81h

Table 7. Gate-delay fault ATPG results after TPI, with and without the fault efficiency limitation

Circuit	No TPI				TPI (FE limit)				TPI (no FE limit)			
	#Pat.	FC(%)	FE(%)	time	#Pat.	FC(%)	FE(%)	time	#Pat.	FC(%)	FE(%)	time
s1488	154	90.11	97.65	6.13s	100	94.77	94.98	1.87s	111	95.77	95.98	1.94s
s1494	153	89.50	97.58	6.20s	97	94.04	94.39	1.87s	114	95.37	95.72	2.00s
s9234.1	238	81.90	93.30	61.47s	85	85.62	92.90	15.70s	120	88.45	95.73	18.04s
s15850.1	216	81.44	96.31	75.49s	107	85.72	95.56	44.37s	169	87.20	97.04	52.56s
s38584.1	246	90.19	97.85	326.77s	121	91.22	97.66	280.30s	246	91.72	98.17	301.21s
p5973	97	91.67	94.04	106.24s	48	91.41	93.74	20.11s	99	95.52	97.85	26.57s
p73133	630	93.45	97.34	91.78m	316	93.55	97.10	48.59m	419	94.16	97.71	52.88m
p75344	514	92.43	97.68	26.31m	349	93.78	97.56	20.46m	380	93.90	99.68	20.66m
p90685	229	60.89	70.41	567.16m	148	61.34	70.34	457.19m	231	64.53	73.52	506.79m
p93140	650	93.90	99.22	92.65m	523	94.00	99.23	69.57m	567	94.13	99.36	66.51m
p104649	1895	67.71	89.47	624.26m	833	68.67	89.59	314.54m	2023	74.67	95.47	337.57m
p114605	2361	95.33	98.86	338.98m	1534	95.49	98.86	187.63m	1631	96.28	99.62	186.77m
$\sum$	7383	83.99	92.62	1750.84m	4261	84.68	92.54	1104.05m	6110	86.56	94.63	1177.89m

significant reduction in the CPU times has been achieved. These CPU times have been reduced from 33.30h to 23.81h (28%). The total CPU-time spent on TPI is 89.57m, hence even the total time spent on TPI and GDF ATPG after TPI is much less than the CPU-time spent on GDF ATPG before TPI. For circuits p104649, s9234.1 and p5973, the reduction in CPU time is 50% or more. However, for other circuits, the GDF ATPG takes more time; e.g., for circuits p71553 and p137498. They are tri-state circuits in which bus-conflicts can occur, such that it possibly has become more difficult to generate bus-conflict-free patterns.

Although the FEs are the same before and after TPI (part of the experimental setup), the FCs often increase after TPI. Due to the insertion of TPs, ATPG untestable signal lines in the circuit become testable. Hence TPI not only decreases test set sizes and CPU time, it often also results in an increase in achievable FCs with a few per cent.

Table 6 showed the impact of TPI on the GDF test set sizes in case the FE before and after TPI is approximately the same. In Table 7 results are shown in case the GDF ATPG stops when it cannot generate any more GDF test patterns, instead of when the same FE is reached as for the circuit without TPI. In other words, Table 7 shows the impact of TPI on the FE that can be achieved; it only shows the results for the circuits which results differ from the ones presented in Table 6. Columns 2-5 show respectively the num-

ber of GDF patterns (*#Pat.*), the FC (*FC*), the FE (*FE*) and the ATPG time (*time*) for the circuit without TPs. Columns 6-9 show these results after TPI with the FE limit, as presented in Table 6. Columns 10-13 show these results without the FE limitation.

When the GDF ATPG is not limited to the same FE for the circuit with TPs as for the circuit without TPs, the FCs/FEs achieved for the circuits in Table 7 are increased with two per cent. However, this increase in FC/FE does cost extra test patterns. Instead of reducing the GDF test set sizes from 7383 to 4261 patterns (43%), the GDF test set sizes are only reduced to 6110 patterns (18%). This large difference is mainly caused by the results for circuit p104649. In order to get an extra 6% fault coverage, 1190 extra patterns were generated.

The GDF ATPG times without the FE-limit only marginally increased compared to the times with the FE-limit. The reduction in CPU time is still 33% (from 1750.84m to 1177.89m) compared to 37% with the FE limitation (from 1750.84m to 1104.05m).

7 Summary and conclusions

The growth in ATPG test set sizes results in higher memory requirements for the ATE and longer test application times. In [1] it was shown that with TPI the ATPG test set sizes could be reduced significantly at the cost of only a few test points. In this paper we have shown that by using testability analysis measures to select the cost function that is used in the TPI algorithm for the test point selection, even better test set size reductions are achieved. For the ISCAS benchmark circuit, the test set size reduction improved from 39% to 45% and for several Philips industrial circuits, the test set reduction even improved from 32% to 40%. On the circuits which do suffer from large test sets, even better gains in reduction are achieved, i.e., 42% reduction has been achieved on circuit p10469 with the proposed method versus 27% reduction otherwise.

However, experimental results, see Table 5 have shown that for circuits with very large FFRs, it sometimes takes a lot of test points before the test set sizes are reduced. In this paper we have shown that by extending the TPI method with specific techniques to reduce large FFRs, also for these circuits the test set sizes are reduced significantly. Experimental results show that for circuits with large FFRs, the Hybrid TPI method for compact ATPG [1], in combination with TPI for reducing large FFR, results in 10% *extra reduction* for the *same* number of test points.

In addition to the stuck-at fault (SAF) model, other fault models exist for which an ATPG tool is used to generate test sets; e.g., gate-delay fault (GDF) model. The GDF ATPG test set sizes and generation times tend to be even larger than those for SAF ATPG. Experimental results show that by using TPI, the number of GDF ATPG patterns are reduced with 35%. Besides reductions in test set sizes, also reductions of 33% in ATPG generation times are achieved.

TPI can also be used to increase the fault coverages and efficiencies for GDF faults with a few percent. When TPI is used to increase the fault coverage, the test set sizes are less reduced, but still a reduction of 18% has been achieved.

Acknowledgements

The authors want to thank the people from Philips Semiconductors and Philips Research for their contribution to this research. Especially for providing us the research topics, for the financial support and for providing the industrial tools and circuits to do the TPI and gdf ATPG experiments.

References

[1] M.J. Geuzebroek, J.Th. van der Linden and A.J. van de Goor. "Test point insertion for compact test sets". *Proc. of the IEEE Int. Test Conf.*, 292-301, 2000.

[2] B.H. Seiss, P.M. Trouborst, and M.H. Schulz. "Test Point Insertion for Scan-Based BIST". *Proc. of 2nd Euro. Test Conf.*, pp. 253-262, 1991.

[3] C. Schotten, and H. Meyr. "Test Point Insertion For An Area Efficient BIST". *Proc. of the IEEE Int. Test Conf.*, pp. 515-523, 1995.

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

[5] H.-C. Tsai, K.-T. Cheng, C.-J. Lin, and S. Bhawmik. "A Hybrid Algorithm for Test Point Selection for Scan-Based BIST". *Proc. of 34th Design Automation Conf.*, pp. 478-483, 1997.

[6] M. Nakao, S. Kobayashi, K. Hatayama, K. Iijima and S. Terada. "Low overhead Test Point Insertion for Scan-Based BIST". *Proc. of the IEEE Int. Test Conf.*, pp. 348-357, 1999.

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

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

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

[10] B. Krishnamurthy. "A Dynamic Programming Approach to the Test Point Insertion Problem". *Proc. of 24th ACM/IEEE Design Automation Conf.*, pp. 695-705, 1987.

[11] L.H. Goldstein and E.L. Thigpen. "SCOAP: Sandia Controllability/Observability Analysis Program". *Proc. of 17th Design Automation Conf.*, pp. 190-196, 1980.

[12] F. Brglez and H. Fujiwara "A Neural Netlist of 10 Combinational Benchmark Designs and a Special Translator in Fortran". *Proc. of IEEE Int. Symp. on Circuits and Systems*, 1985.

[13] F. Bgrlez, D. Bryan, and K. Kozminski. "Combinational Profiles of Sequential Benchmark Circuits", *Proc. of IEEE Int. Symp. on Circuits and Systems*. pp. 1929-1934, 1989.

[14] M.H. Konijnenburg, J. Th. van der Linden and A.J. van de Goor. "Accelerated Compact Test Set Generation for Three-State Circuits". *Proc. of IEEE Int. Symp. on Circuits and Systems*, pp. 29-39, 1996.

[15] G.L.Smith. "Model for delay faults based upon paths". *Proc. of the IEEE Int. Test Conf.*, pp. 342-349, 1985.