J Electron Test (2007) 23:445–455
DOI 10.1007/s10836-007-5013-7

Functional Constraints vs. Test Compression in Scan-Based Delay Testing

Ilia Polian · Hideo Fujiwara

Received: 29 August 2006 / Accepted: 6 June 2007 / Published online: 9 September 2007

Abstract We present an approach to prevent overtesting in scan-based delay test. The test data is transformed with respect to functional constraints while simultaneously keeping as many positions as possible unspecified in order to facilitate test compression. The method is independent of the employed delay fault model, ATPG algorithm and test compression technique, and it is easy to integrate into an existing flow. Experimental results emphasize the severity of overtesting in scan-based delay test. Influence of different functional constraints on the amount of the required test data and the compression efficiency is investigated. To the best of our knowledge, this is the first systematic study on the relationship between overtesting prevention and test compression.

Keywords Overtesting prevention · Functional constraints · Scan-based delay test · Test compression

1 Introduction

Extensive use of design for testability (DFT) techniques, including scan and test points, and non-nominal test methods such as low-voltage test and I_{DDQ} test [4, 6] lead to *overtesting*, i.e., the IC is demonstrated to fail, but under conditions which cannot occur in its normal operation mode. One reason for overtesting is the presence of latent defects, which are too small to cause a failure under nominal conditions or logically redundant. A further reason is the elevated level of IR drop and crosstalk effects which is caused by atypical power consumption during test that does not correspond to the power consumption profile in normal operation [20]. Last but not least, behavior which does not contradict the specification could be classified as 'faulty behavior' by the test process if design tricks such as multi-cycle paths are employed.

There appears to be no broad consensus whether overtesting should be maximized or prevented. On one hand, overtesting is assumed to be an efficient (and often the only) approach to detect latent defects, which are not critical yet but may deteriorate and become early-life failures [3, 6, 15]. On the other hand, it is argued that overtesting results in detections which are not necessarily due to a defect and that it mainly leads to yield loss, i.e., discarding good chips. It has also been reported that a non-functional test sequence can damage the chip by inducing heat dissipation that exceeds the limit the chip is designed for [16]. From this, the need to prevent overtesting by using only *functional test data*, which can occur in the IC's normal operation mode, is deduced [8–11, 16, 18, 20].

Several methods exist to prevent overtesting. They include generating functional patterns by a special ATPG [10–12], transforming existing non-functional test sets into functional test sets [16] and providing on-chip hardware to block non-functional patterns from being applied [5, 9, 14]. Whether test data is functional

Responsible Editor: R.A. Parekhji

I. Polian (✉)
Institute of Computer Science,
Albert-Ludwigs-University, Georges-Köhler-Allee 51,
79110 Freiburg im Breisgau, Germany
e-mail: polian@informatik.uni-freiburg.de

H. Fujiwara
Graduate School of Information Science,
Nara Institute of Science and Technology, Nara, Japan
e-mail: fujiwara@is.naist.jp

or not, is decided based on *functional constraints*. There are different types of such constraints and different exact and approximate methods for their computation, as will be explained in detail later.

In this article, we study overtesting prevention in a test compression scenario. Test compression is an essential technique for handling the growth of test data [17]. Modern test compression approaches work in two stages: first, an ATPG is used to generate test patterns, and then an encoding algorithm is run over these patterns. Since the efficiency of the encoding grows with the fraction of don't care (X) values in the test data, the ATPGs used in test compression are tuned to specify as few bits as possible. One goal in the design of our method is to minimize the impact on the existing flow. Thus, we do not propose any modifications to the existing ATPG tool (such as done in [10–12]) if it does not support functional constraints or supports only a subset of the needed constraints. We are also not considering adding any hardware to the design like in [5, 9, 14]. We focus on the use of scan in delay test as the source of overtesting for the following four reasons [18]: First, the application of non-functional test pairs may result in DFT overhead if enhanced scan design is employed. Second, only a small subset of all possible state transitions are realizable, i.e., many physical paths cannot be sensitized in normal operation. Third, if a non-functional test pair sensitizes a path which is not sensitizable in normal operation, this path may violate timing constraints. Then, design effort for timing optimization of such a path would be required to avoid rejection of good ICs. This effort would have no benefit for normal operation. Fourth, wrong paths could be sensitized in delay fault testing, leading to yield loss and debugging effort.

We introduce a tool, called FUJISAN (Functional constraint Justificator and Statistical Analyzer). FUJISAN accepts a delay test pair (TP) set (with Xes) as an input and transforms those TPs for which it is possible into *functional test pairs*. Several types of functional constraints are supported. The resulting TPs still have a high number of Xes (though this number is lower than in the original test pairs) and can be handed to the encoding algorithm. This is the main advantage over the method from [16], which ends up with fully specified tests that are unlikely to be good to compress. Moreover, FUJISAN employs exact algorithms while [16] is based on approximations, and it supports more constraints than [16]. FUJISAN does not require any modification of the ATPG nor the encoding software for test compression. Hence, it is easy to integrate into the flow. Figure 1 illustrates the difference between the conventional test compression flow and the flow using FUJISAN.

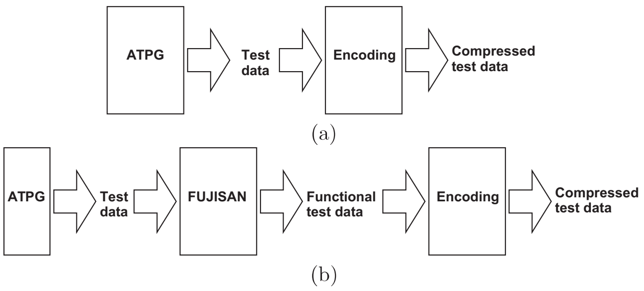

Fig. 1 **a** Conventional test compression flow; **b** test compression flow with FUJISAN

We apply FUJISAN to path delay fault test sets generated for the combinational core of the circuit without considering any constraints. We discuss the required amount of test data depending on the considered functional constraints. We track the percentage of Xes in the test sets before and after applying FUJISAN and make conclusions on the suitability of the data for test compression based on this information. We validate our conclusions by applying a simple representative test compression algorithm to the respective test sets.

The remainder of the article is organized as follows.[1] The next session discusses the constraints considered in the paper. FUJISAN is introduced in Section 3. Experimental results are reported in Section 4. Section 5 concludes the article.

2 Functional Constraints

This section discusses the constraints **Cube**, **FT**, **RS** and **SI**, and the incorporation of further constraints. We call a TP with X values a *test pair cube* and a fully-specified TP which matches all the specified positions of a TP cube an *instance* of that TP cube. An instance is called *functional* with respect to some constraints if it satisfies these constraints, otherwise it is called *non-functional*. A TP cube is called (non-)functional if all its instances are (non-)functional, and *partially functional* if some of its instances are functional and some are not. Speaking simply, our goal is to transform a partially functional TP cube into a functional TP cube while preserving as many Xes as possible.

Constraint **Cube** is not a functional constraint in a strict sense but complements other constraints in order to obtain a cube, which is suited for test compression

[1]Parts of the article have been presented at the Design Automation and Test in Europe (DATE) Conference 2006.

encoding software. For example, suppose that only instance `111` of pattern `1XX` violates a functional constraint (75% of the instances are functional). However, there is no cube representing the set {`100`, `101`, `110`}. But current test compression encoding algorithms require cubes as input. Hence, a cube representing a subset of {`100`, `101`, `110`} and having a large number of Xes must be used. The cubes `10X` and `1X0` both represent an optimal solution. Note that satisfying Constraint **Cube** dropped the share of functional pairs from 75 to 50%.

Constraint **FT** (functional transition) ensures the transition between the first and second vector of the TP exists.

Constraint **RS** (reachable state) requires that the first vector of the TP is reachable from an initial state of the circuit and thus can occur in normal operation.

Constraint **SI** (steady input) assumes that the external frequency (I/O frequency) of a chip is lower than its internal frequency, which is true for some of today's designs. At-speed transitions are allowed at the chip's flip-flops (FFs) but not at its primary inputs (PIs). A similar restriction was used in the LSI Logic study [13] (motivated by the shortcomings of the tester). Note that no path delay faults for paths starting at a PI can be tested, which is acceptable as these paths are not switched at-speed.

While Constraints **FT** and **RS** are valid for any digital synchronous circuit, Constraint **SI** is an example for a *design-specific constraint* which is derived from the knowledge about the characteristics of the chip (here, the difference in external and internal speed). Other design-specific constraints are possible, such as one-hot state encoding constraints. These constraints can be extracted from the HDL code if the designers formulate such restrictions as assertions. Although it is straightforward to integrate such or any other constraints into our framework, in this article we do not consider any constraints beyond those described in this section.

3 FUJISAN

FUJISAN transforms a set of delay TP cubes with Xes into functional TP cubes with respect to the constraints from the last section (any constraint can be switched on or off). FUJISAN is applied if the used ATPG tool does not support all required functional constraints. For instance, Constraints **FT** and **RS** may be required. The ATPG tool may provide a way to enforce Constraint **FT** but not Constraint **RS**. Then, FUJISAN can be applied on the test pairs generated by the ATPG tool to ensure that Constraint **RS** is also satisfied.

First, FUJISAN identifies for every TP its functional instances. Based on this information, a *statistical profile* of the test set is created. Each pair is classified as belonging to one of seven classes 0%, 0–20%, 20–40%, 40–60%, 60–80%, 80–100 and 100%. A pair belongs to Class 0% if it has no functional instance, to Class 100 if all of its instances are functional, to Class 0–20 if it has at least one functional instance, but less than 20% of its instances are functional, and so on.

If test compression is performed without running FUJISAN first, a pair from Class 0–20 will result in a non-functional test with a probability between 80 and 100% (assuming that the encoding algorithm assigns the Xes randomly). The probability is a lower bound if the encoding procedure is allowed to apply the same test several times. Hence, FUJISAN can be used to estimate the extent of overtesting in order to decide whether any corrective measures are necessary.

FUJISAN is implemented using BDDs [1]. For a circuit with N PIs and F FFs, let f_i be the logic function of the i^{th} FF (primary outputs are irrelevant for this analysis). Let the test pair be $(V, W) := (v_1 \dots v_{N+F}, w_1 \dots w_{N+F})$ with $v_i, w_i \in \{0, 1, X\}$ and $v_1, \dots, v_N, w_1, \dots, w_N$ are the PI values. In the following, the implementation of the constraints is explained in more detail.

3.1 Constraint **FT**

To calculate the number of pairs which correspond to a functional transition, all functional transitions are determined and their number counted. This is done implicitly using BDDs, such that explicit enumeration of functional transition is avoided.

Transitions which are functional with respect to the circuit with function f determined from the *transition relation* $TRel_f$ of function f. The transition relation is a predicate over input variables $y_1, \dots, y_N$, current-state variables $y_{N+1}, \dots, y_{N+F}$ and next-state variables $z_1, \dots, z_F$. $TRel_f$ contains the tuple $(y_1, \dots, y_N, \dots y_{N+F}, z_1, \dots, z_F)$ if and only if the circuit implementing function f with values $y_1, \dots, y_N$ on its primary inputs and values $y_{N+1}, \dots, y_{N+F}$ in its flip-flops produces values $z_1, \dots, z_F$ in its flip-flops in the next clock cycle.

Speaking formally, the transition relation of function f is represented by its *characteristic function* which we denote $TR_f : \mathbb{B}^{N+2F} \to \mathbb{B}$, defined as

$$TR_f(y_1, \dots, y_N, \dots y_{N+F}, z_1, \dots, z_F) = 1$$
$$:\Leftrightarrow \bigwedge_{i=1}^{F} \left(f_i(y_1, \dots, y_N, \dots, y_{N+F}) \equiv z_i \right). \quad (1)$$

To obtain only those functional transitions which are compatible with the test pair (V, W), the transition relation must be *restricted* to the test pair: for each specified position v_i of V, the corresponding input or current-state variable y_i must be forced to v_i, and for each specified position w_{N+j} of the state part of W, the corresponding next-state variable z_j must be forced to w_{N+j}. The unspecified positions allow arbitrary value assignments and do not lead to any forced values of variables. The resulting formula is:

$$TR_f(y_1, \ldots, y_{N+F}, z_1, \ldots, z_F) \wedge \bigwedge_{v_i \neq X} (y_i \equiv v_i) \wedge \bigwedge_{w_{N+j} \neq X} (z_j \equiv w_{N+j}). \tag{2}$$

Every assignment of $y_1, \ldots, y_{N+F}, z_1, \ldots, z_F$ satisfying Eq. 2 corresponds to a functional transition compatible with the test pair (V, W). The number of such assignments is the number of test pair instances satisfying Constraint **FT**. This number can be obtained by constructing the BDD of the formula in Eq. 2 and determining the size of its onset.

To reduce the computational complexity, we replaced calculating the transition relation and restricting it to a test pair by computing the BDD of the *transition function restricted to test pair* (V, W):

$$T(y_1, \ldots, y_{N+F}) = \bigwedge_{v_i \neq X} (y_i \equiv v_i) \wedge \bigwedge_{w_{N+j}=1} f_j(y_1, \ldots, y_{N+F}) \wedge \bigwedge_{w_{N+j}=0} \neg f_j(y_1, \ldots, y_{N+F}) \tag{3}$$

and calculating the number of its satisfying assignments, i.e., the size of its onset. The numbers of assignments which satisfy Eqs. 3 and 2 are equal because every assignment of $y_1, \ldots, y_{N+F}$ which satisfies Eq. 3 corresponds to an assignment of $y_1, \ldots, y_{N+F}, z_1, \ldots, z_F$ which satisfies Eq. 2 by setting $z_j := f_j(y_1, \ldots, y_{N+F})$. Hence, the size of the onset of $T(y_1, \ldots, y_{N+F})$ gives the number of the test pair instances which satisfy Constraint **FT**.

3.2 Constraint **RS**

The set S of reachable states (for Constraint **RS**) is obtained by Procedure reachable_states shown in Fig. 2. State traversal from the initial state is performed until a fixed point is reached. First, S is set to $\{s^0\}$ where s^0 is the initial state (Lines 1 and 4). Then, all states reachable from the states currently in S are calculated (Line 5). This step is iterated until S does not change in an iteration (Line 2). The resulting set is ANDed with function T from Eq. 3. This is the exact algorithm for finding all reachable states (which is an NP complete problem). Numerous approximate methods with a better scalability exist for this purpose. At this moment, FUJISAN does not incorporate any approximate technique.

```
Procedure reachable_states
Input: Circuit function $f$; initial state $s^0$
Output: Set $S$ of all states reachable from $s^0$
(1) $S := \emptyset$; $S' := \{s^0\}$;
(2) while $(S \neq S')$
(3) begin
(4)     $S := S'$;
(5)     $S' := S \cup \{z_1, \ldots, z_F \mid \exists p \in \mathbb{B}^N \; \exists s \in S \; \forall 1 \leq i \leq F :$
                $f_i(p_1, \ldots, p_N, s_1, \ldots, s_F) = z_i\}$;
(6) end while
(7) return $S$;
end reachable_states;
```

Fig. 2 Algorithm for obtaining the set of reachable states

3.3 Constraint **SI**

Constraint **SI** is implemented by propagating a specified value on a PI of first or second vector of a test pair to the respective position of the other vector of that pair before building the BDD for function T from Eq. 3. For example, if $v_3 = X$ and $w_3 = 1$, v_3 is set to 1. This results in one more conjunction $(y_3 \equiv 1)$ in Eq. 3.

3.4 Constraint **Cube**

If Constraint **Cube** is specified, FUJISAN writes out test pair cubes which have only functional instances (if any such instances exist). FUJISAN tries to find the largest functional sub-instance, i.e., one with a maximal number of Xes, in order to help subsequent encoding. This is done by finding the shortest path to a 1-terminal of the transition function BDD and extending it to a prime implicant.

For example, consider the following BDD in Fig. 3. It has five paths to 1-terminal: $\bar{v}_1\bar{v}_2\bar{w}_1w_2$, $\bar{v}_1v_2w_2$, $v_1\bar{v}_2\bar{w}_1w_2$, $v_1\bar{v}_2w_1\bar{w}_2$ and v_1v_2. They represent test pair cubes (00, 01), (01, X1), (10, 01), (10, 10) and (11, XX), respectively. Note that this BDD is given only for illustration and does not represent a test pair set for an actual fault in any circuit as, e.g., (10, 10) is obviously not a valid test. Only one of the test pair cubes can be selected, preferably the one with the maximal number of Xes, i.e., (11, XX). Its selection is achieved by determining the shortest path to a 1-terminal in the BDD, i.e., v_1v_2 (shown bold in the figure). This

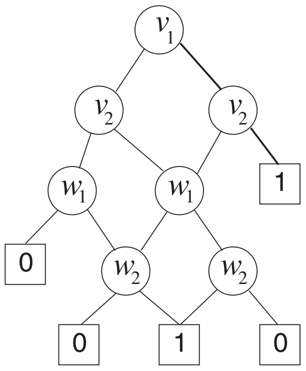

Fig. 3 Largest functional sub-instance as a shortest 1-path in a BDD

operation is performed automatically by state-of-the-art BDD packages.

3.5 Functional Constraints vs. Test Compression

It is interesting that the amount of data to be provided for the testing depends on the employed functional constraint. If no functional constraint is imposed, then any bit position (both PIs and FFs) can have an arbitrary logic value. Thus, for a circuit with N PIs and F FFs $2N + 2F$ bits must be provided per applied test pair. (While the question how to actually deliver this test data to the chip and trigger an at-speed transition is out of scope of this article, enhanced scan [2] is one possibility). If Constraint **FT** is enforced, then there is no need to provide the values for the FFs in the second time frame, as they can be calculated by the circuit using broadside test application (launch-by-capture) [19]. This reduces the amount of bits to be delivered per TP to $2N + F$. If, in addition, Constraint **SI** is considered, this amount is reduced to $N + F$, because the values on the PIs in the second time frame are simply the same as in the first time frame. Constraint **RS** has no influence on the amount of uncompressed test data.

It seems that considering more constraints results in test data reduction. On the other hand, in a test compression flow (which we are considering) the relevant number is the amount of *compressed data* that needs to be stored in the tester memory and not the data that is actually applied to the IC. It can be expected that justifying functional constraints will decrease the proportion of X values in the TP cubes even though FUJISAN will minimize this decrease. Hence, the compression ratio will probably be lower after justification. It is an interesting question whether the worsening in compression ratio outweighs the reduction in the size of the data to be encoded. The answer will be given by the experimental results.

4 Experimental Results

We applied FUJISAN to test sets generated by the tool TIP [7, 21] to ISCAS-89 circuits assuming no functional constraints (enhanced-scan mode). The test sets have 100% robust path delay coverage and are not compacted.

4.1 Statistical Analysis

Table 1 quotes the results considering only Constraint **FT** (functional transition). The first four columns contain the name of the circuit, number of PIs, FFs and TPs generated by TIP. The subsequent seven columns report the percentage of the test pairs belonging to one of the seven classes introduced above. For brevity, we write 'Class <20%' for 'Class 0–20%' etc. The final columns show FUJISAN's run time in seconds (including constraint assignment) and peak memory consumption in MB.

It can be seen that few TP cubes have only functional instances (6.6% on average), even with respect to Constraint **FT**, which is the weakest criterion. Hence, running test compression on the test data as generated by TIP would result in a large number of non-functional transitions applied to the circuit and thus overtesting. A significant amount of TP cubes have no functional instances at all (Class **0%**). If this is unacceptable but no ATPG supporting the required functional constraints is available, the following heuristic could reduce the number of such pairs: re-run the ATPG with different parameters (such as a different decision strategy) targeting faults which resulted in Class **0%** pairs and apply FUJISAN to determine whether these newly generated pairs have any functional instances.

Table 2 reports the implications of Constraint **Cube**. Note that it quotes absolute numbers of pairs and not percentages. The change due to the constraint is the difference between a table entry and the number in parentheses (which is the number of pairs in a class if constraint **Cube** is not considered). The final row contains the sum of changes over all considered circuits. Since not all circuits are shown in the table, the sum of changes over the circuits in the table is not equal to the number in the last row.

Constraint **Cube** has no influence on Classes **0%** and **100%**. If a TP cube has a non-empty subset of functional instances, then there must also be a non-empty subset of that subset which is a cube, so no pair can move to Class **0%** from a different class due to Constraint **Cube**. If all of the instances of a cube are functional, then the cube determined by FUJISAN is

Table 1 Statistical profiles considering Constraint **FT** (numbers are in per cent)

Circuit	PI	FF	TP	Percentage of test pairs belonging to a class							CPU Time (h)	Peak Mem.
				0%	< 20%	< 40%	< 60%	< 80%	< 100%	100%		
s27	4	3	32	56.2	0.0	3.1	0.0	0.0	0.0	40.6	0.01	4.5
s298	3	14	177	74.6	18.6	0.0	1.1	0.0	0.0	5.6	0.01	4.5
s208.1	10	8	209	79.4	11.0	3.3	2.4	0.0	0.0	3.8	0.01	4.6
s344	9	15	369	87.8	1.6	3.0	1.9	5.1	0.0	0.5	0.03	4.6
s349	9	15	369	87.8	1.6	3.0	1.9	5.1	0.0	0.5	0.03	4.6
s382	3	21	339	84.4	9.4	2.7	0.6	1.5	0.3	1.2	0.01	4.6
s386	7	6	232	81.0	6.5	1.7	4.7	1.7	0.0	4.3	0.02	4.5
s420.1	18	16	641	89.1	7.0	0.9	0.6	0.0	0.0	2.3	0.16	9.1
s444	3	21	303	82.5	13.2	1.0	1.3	0.7	0.0	1.3	0.03	4.6
s510	19	6	369	85.1	4.1	5.7	3.3	0.3	0.0	1.6	0.03	5.0
s526	3	21	356	87.4	8.1	0.3	1.4	0.0	0.0	2.8	0.04	4.7
s713	35	19	522	49.2	40.0	1.1	0.8	2.9	2.9	3.1	0.37	5.5
s820	18	5	475	76.0	8.2	4.6	4.4	2.5	0.0	4.2	0.11	4.7
s832	18	5	488	76.4	8.6	4.3	4.7	2.3	0.0	3.7	0.10	4.7
s953	16	29	839	65.9	20.7	1.2	4.1	1.8	0.8	5.5	0.37	5.0
s1488	8	6	738	93.5	2.0	1.4	1.2	1.2	0.3	0.4	0.08	4.7
s1494	8	6	725	92.4	2.8	1.2	1.7	1.2	0.3	0.4	0.04	4.6
s1196	14	18	1,494	23.8	4.5	10.2	8.5	5.4	3.2	44.3	0.11	4.9
s1238	14	18	1,502	24.6	4.5	11.9	8.3	7.9	1.1	41.7	0.14	5.0
s1423	17	74	12,756	88.4	11.4	0.0	0.0	0.0	0.0	0.0	94.51	15.1
s5378	35	179	8,471	35.2	61.1	1.2	1.4	0.1	0.4	0.6	12.45	27.9
s9234.1	36	211	9,446	72.7	27.2	0.0	0.0	0.0	0.0	0.0	22.13	31.0
s13207.1	62	638	7,310	79.2	20.3	0.2	0.3	0.0	0.0	0.0	28.02	13.6
s15850.1	77	534	29,871	60.9	36.1	1.7	0.6	0.5	0.1	0.2	324.70	43.1
s35932	35	1,728	2,016	75.3	6.3	4.3	8.3	2.3	0.0	3.5	32.85	7.9
s38584.1	38	1,426	27,729	85.8	13.7	0.3	0.1	0.0	0.0	0.1	24	248.1
Average				72.9	13.4	2.6	2.4	1.6	0.4	6.6		

just the original cube itself and it still belongs to Class **100%**. For other classes, a shift from high-probability to low-probability classes can be observed. Hence, the need to represent data in a format which encoding algorithms can read makes the overtesting problem more severe, although the extent is limited.

Table 3 illustrates the consequences of enforcing Constraint **RS** (reachable state) assuming the all-0 state

Table 2 Implications of Constraint **Cube**

Circuit	0%	< 20%	< 40%	< 60%	< 80%	< 100%	100%
s298	132 (132)	33 (33)	0 (0)	2 (2)	0 (0)	0 (0)	10 (10)
s344	324 (324)	10 (6)	7 (11)	26 (7)	0 (19)	0 (0)	2 (2)
s382	286 (286)	35 (32)	8 (9)	6 (2)	0 (5)	0 (1)	4 (4)
s420.1	571 (571)	45 (45)	6 (6)	4 (4)	0 (0)	0 (0)	15 (15)
s713	257 (257)	225 (209)	13 (6)	11 (4)	0 (15)	0 (15)	16 (16)
s832	373 (373)	43 (42)	20 (21)	34 (23)	0 (11)	0 (0)	18 (18)
s1488	690 (690)	15 (15)	10 (10)	20 (9)	0 (9)	0 (2)	3 (3)
s1196	356 (356)	192 (67)	167 (153)	117 (127)	0 (81)	0 (48)	662 (662)
s5378	2,982 (2,982)	5,308 (5,177)	47 (103)	85 (115)	0 (7)	0 (38)	49 (49)
s15850.1	18,177 (18,177)	11,090 (10,770)	344 (514)	192 (175)	0 (141)	0 (26)	68 (68)
s35932	1,519 (1,519)	237 (127)	49 (86)	141 (167)	0 (47)	0 (0)	70 (70)
Total	±0	+854	−239	+67	−521	−161	±0
Change	(0%)	(+1.1%)	(−0.3%)	(+0.1%)	(−0.7%)	(−0.2%)	(0%)

Numbers in parentheses are valid if the constraint is not enforced. The total change is calculated over all considered circuits.

Table 3 Implications of Constraint **RS**

Circuit	|RS|	0%	< 20%	< 40%	< 60%	<80%	<100%	100%
s208.1	100.0	166 (166)	23 (23)	7 (7)	5 (5)	0 (0)	0 (0)	8 (8)
s349	5.2	331 (324)	38 (6)	0 (11)	0 (7)	0 (19)	0 (0)	0 (2)
s386	20.3	193 (188)	17 (15)	4 (4)	10 (11)	2 (4)	0 (0)	6 (10)
s420.1	100.0	571 (571)	45 (45)	6 (6)	4 (4)	0 (0)	0 (0)	15 (15)
s510	73.4	318 (314)	17 (15)	16 (21)	16 (12)	1 (1)	0 (0)	1 (6)
s526	0.4	312 (311)	44 (29)	0 (1)	0 (5)	0 (0)	0 (0)	0 (10)
s713	0.3	436 (257)	86 (209)	0 (6)	0 (4)	0 (15)	0 (15)	0 (16)
s820	78.1	361 (361)	39 (39)	33 (22)	21 (21)	1 (12)	0 (0)	20 (20)
s953	10^{-6}	558 (553)	281 (174)	0 (10)	0 (34)	0 (15)	0 (7)	0 (46)
s1488	75.0	690 (690)	15 (15)	15 (10)	12 (9)	3 (9)	0 (2)	3 (3)
Total		+3.4%	+1.1%	−0.4%	−0.7%	−1.5%	−0.4%	−1.5%
+Cube		+3.4%	+1.3%	−0.4%	−0.7%	−1.7%	−0.4%	−1.5%

as the initial state. Column 2 (|RS|) shows the percental fraction of reachable states compared to all states. The implications are significant if that fraction is low. The second-last row shows the changes aggregated over all circuits for which reachable states could be calculated. The last row shows aggregated data if Constraints **RS** and **Cube** are considered simultaneously. The additional influence of Constraint **Cube** appears to be limited.

The implications of Constraint **SI** (steady input) are given in Table 4. The pairs which violated the constraints by having opposite values on matching PIs of the first and second vector have been removed beforehand, and their number is reported in Column Rem. Their fraction varies from insignificant (s344) to over 50% for s15850.1. It is interesting that the number of pairs in Class **100%** increases. This is because some non-functional instances are removed by specifying additional PI values. Apart from that, the changes are not very significant. In particular, not too many pairs lose all of their functional instances due to Constraint **SI**. The final rows show the aggregated numbers for Constraint **SI** only; in combination with Constraint **Cube**; and in combination with Constraint **Cube** and **RS**.

4.2 Impact on Test Compression

Tables 5 and 6 give results on test compression. Table 5 reports results for all of the test pairs generated by TIP. Columns 2 and 4 contain the number of bits before and after FUJISAN was run (US stands for ‘uncompressed size’), and columns 3 and 5 give the percentage of do not cares in the respective test sets. As discussed above, the amount of test data is reduced from $2N + 2F$ to $2N + F$ if a test pair has at least one functional instance. Otherwise, FUJISAN does not modify it (based on the philosophy that it is better to detect a fault with a non-functional test than not to detect it at all) and $2N + 2F$ bits are stored. We applied the 9C compression algorithm [22], which is a simple yet

Table 4 Implications of Constraint **SI**

Circuit	Rem	0%	<20%	<40%	<60%	<80%	<100%	100%
s344	8	324 (324)	5 (6)	10 (10)	2 (1)	19 (19)	0 (0)	1 (1)
s386	58	151 (150)	8 (9)	2 (2)	4 (6)	1 (2)	1 (0)	7 (5)
s420.1	88	524 (524)	10 (11)	1 (2)	2 (1)	0 (0)	0 (0)	16 (15)
s1488	123	574 (574)	8 (14)	14 (10)	14 (7)	3 (8)	0 (0)	2 (2)
s1196	1,395	47 (41)	4 (18)	10 (16)	8 (12)	12 (11)	6 (0)	12 (1)
s1423	898	10,495 (10,482)	1,357 (1,371)	3 (3)	3 (1)	0 (1)	0 (0)	0 (0)
s9234.1	1,177	5,995 (5,970)	2,273 (2,298)	1 (1)	0 (0)	0 (0)	0 (0)	0 (0)
s13207.1	1,563	4,850 (4,684)	875 (1,039)	11 (13)	6 (6)	2 (2)	2 (2)	1 (1)
s15850.1	16,550	10,062 (10,021)	3,192 (3,207)	23 (29)	41 (61)	2 (3)	0 (0)	1 (0)
s35932	538	1,178 (1,178)	82 (97)	51 (42)	118 (108)	10 (14)	0 (0)	39 (39)
Total		+0.96	−1.02	+0.01	+0.02	−0.03	+0.02	+0.06
+ Cube		+0.96	−0.78	−0.08	+0.07	−0.22	−0.01	+0.06
+ RS		+2.56	+0.27	−0.50	−0.23	−1.45	−0.06	−0.59

Table 5 Test compression results considering Constraint **FT** (all test pairs)

Circuit	Orig. pairs		Fct. pairs		Orig. pairs		Fct. pairs		
	US	%DC	US	%DC	CS	CR	CS	CR	CR_{ov}
s298	6,018	71.3	5,388	68.7	3,528	1.7	3,374	1.6	1.8
s208.1	7,524	58.2	7,180	57.4	4,510	1.7	4,347	1.7	1.7
s382	16,272	71.6	15,159	70.5	8,068	2.0	7,936	1.9	2.1
s510	18,450	72.6	18,120	73.2	10,067	1.8	9,564	1.9	1.9
s526	17,088	71.1	16,143	69.6	9,464	1.8	9,928	1.6	1.7
s713	56,376	81.1	51,341	79.8	21,661	2.6	21,383	2.4	2.6
s953	75,510	79.7	67,216	78.1	31,754	2.4	30,845	2.2	2.4
s1488	20,664	46.1	20,376	46.3	20,183	1.0	20,059	1.0	1.0
s1238	96,128	67.1	75,752	58.2	58,277	1.6	56,344	1.3	1.7
s1423	2,321,592	71.1	2,212,442	69.7	1,170,620	2.0	1,173,415	1.9	2.0
s5378	3,625,588	93.3	2,643,057	90.0	794,653	4.6	705,188	3.7	5.1
s9234.1	4,666,324	93.0	4,122,577	92.0	849,864	5.5	815,466	5.1	5.7
s13207.1	10,234,000	97.9	9,264,878	97.7	1,447,921	7.1	1,309,730	7.1	7.8
s15850.1	36,502,362	95.0	30,257,766	94.2	5,824,025	6.3	5,017,880	6.0	7.3
s35932	7,108,416	99.6	6,249,600	99.6	915,100	7.8	808,779	7.7	8.8

representative technique, to both of the test sets. We used the codewords and the parameter $K = 8$ given in [22]. The number of bits in the compressed data is denoted as CS (‘compressed size’) and the compression ratio is denoted as CR. The overall compression ratio CR_{ov} is defined as US of a test set before running FUJISAN divided by CS of the functional test set obtained by FUJISAN. Table 6 contains the same information for the subset of the test set consisting of test pairs with at least one functional instance.

The percentage of Xes goes down in most (but not all) cases after application of FUJISAN. The compression ratio also goes down, and the extent of the decrease is well correlated with the extent of the decrease of the percentage of Xes. However, the size of the compressed functional test is always below the size of the compressed original test, with one notable exception of s1423. For this circuit, the compression ratio reduction is so heavy that it outweighs the decrease in bits to be saved from $2N + 2F$ to $2N + F$ per pair. Finally, the results for the complete data (Table 5) and the subset with functional instances (Table 6) show the same trend but the magnitude of changes is larger for the

Table 6 Test compression results considering Constraint **FT** (pairs with functional instances)

Circuit	Orig. pairs		Fct. pairs		Orig. pairs		Fct. pairs		
	US	%DC	US	%DC	CS	CR	CS	CR	CR_{ov}
s298	1,530	74.2	900	60.7	760	2.0	636	1.4	2.4
s208.1	1,548	61.5	1,204	58.0	921	1.7	744	1.6	2.1
s382	2,544	74.8	1,431	65.0	1,234	2.1	827	1.7	3.1
s510	2,750	73.4	2,420	78.3	1,440	1.9	991	2.4	2.8
s526	2,160	79.3	1,215	65.6	868	2.5	769	1.6	2.8
s713	28,620	80.7	23,585	77.7	11,045	2.6	10,455	2.3	2.7
s953	25,740	79.2	17,446	72.8	10,678	2.4	9,699	1.8	2.7
s1488	1,344	49.3	1,056	52.2	1,129	1.2	901	1.2	1.5
s1238	72,448	68.0	52,072	55.4	44,273	1.6	41,459	1.3	1.7
s1423	268,450	76.4	159,300	61.0	108,438	2.5	117,063	1.4	2.3
s5378	2,349,292	94.2	1,366,761	88.4	492,941	4.8	409,252	3.3	5.7
s9234.1	1,273,038	93.7	729,291	88.5	216,434	5.9	181,517	4.0	7.0
s13207.1	2,126,600	98.2	1,157,478	96.8	294,209	7.2	172,193	6.7	12.4
s15850.1	14,290,068	94.9	8,045,472	92.0	2,310,583	6.2	1,528,542	5.3	9.3
s35932	1,752,422	99.7	893,606	99.5	224,440	7.8	118,123	7.6	14.8

Table 7 Test compression and Constraint **RS**

Circuit	Orig. pairs		Fct. pairs		Orig. pairs		Fct. pairs		
	US	%X	US	%X	CS	CR	CS	CR	CR_{ov}
s298	986	76.2	580	28.1 (60.7)	443	2.2	685	0.8	1.4 (2.4)
s382	2,208	75.9	1,242	32.1 (65.0)	1,038	2.1	1,308	0.9	1.7 (3.1)
s526	2,112	79.5	1,188	31.5 (65.6)	855	2.5	1,335	0.9	1.6 (2.8)
s713	9,288	82.8	7,654	71.4 (77.7)	3,226	2.9	3,960	1.9	2.3 (2.7)
s953	25,290	79.1	17,141	41.3 (72.8)	10,590	2.4	14,708	1.2	1.7 (2.7)
s1488	1,344	49.3	1,056	51.4 (52.2)	1,129	1.2	905	1.2	1.5 (1.5)

subset. This is because the complete data is amortized by the non-functional test pair cubes not modified by FUJISAN. Consequently, from this point we present only the data for the subset.

Table 7 shows the impact of adding Constraint **RS** for circuits with a small fraction of reachable states. Numbers in parentheses are taken from Table 6 for comparison. The reductions in both the don't care (X) fraction and the compression ratio are severe. Sometimes the compression ratio falls below 1, i.e., the 'compressed' data is larger than the original data. The compressed functional set is larger than the compressed original test set for several circuits.

Figure 4 shows the aggregated results for Constraints **FT** and **RS** in diagram form. Imposing Constraint **FT** allows to reduce the amount of applied data ('Size') because $2N + F$ instead of $2N + 2F$ bits are now required, but the percentage of Xes ('%X') also decreases. As a consequence, the compression ratio declines, but the overall compression ratio CR_{ov} is still higher than CR of original data. However, if Constraint **RS** is considered, the percentage of Xes drops so much that CR_{ov} falls below CR of the original data (note that the slight difference in average size reduction is due to exclusion of a different number of non-functional pairs). This means that, although less data is to be compressed, the size of the compressed data is larger for functional testing.

Figure 5 presents the aggregated results for Constraints **FT** and **SI** (only pairs without conflicting PI assignments in the original test data have been considered). The decrease in X percentage is much less than for Constraint **RS**, and additional N bits per TP can be saved as described above. As a consequence, considering both constraints results in the most compact compressed data. Note that the reduction in size and the compression ratios are higher than in Fig. 4, because larger circuits with many more FFs than PIs and a higher fraction of Xes are considered.

5 Conclusions and Future Work

We proposed a methodology to prevent overtesting due to scan-based delay test in a test compression flow. We introduced a tool, called FUJISAN, which restricts TP cubes generated by an ATPG with respect to a given set of functional constraints and hands them to the encoding routine. In contrast to existing approaches, the resulting TPs have a significant number of don't cares (Xes) and thus can be compressed. FUJISAN works with any ATPG which is suitable for a test compression flow, i.e., can generate tests with Xes, and any encoding procedure. The ATPG does not have to support any functional constraints, although such support will help yield better results. There is no requirement on the

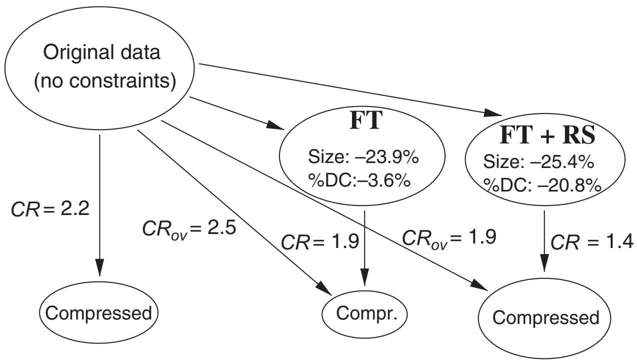

Fig. 4 Aggregated results for Constraints **FT** and **RS** (only circuits for which reachable states have been calculated)

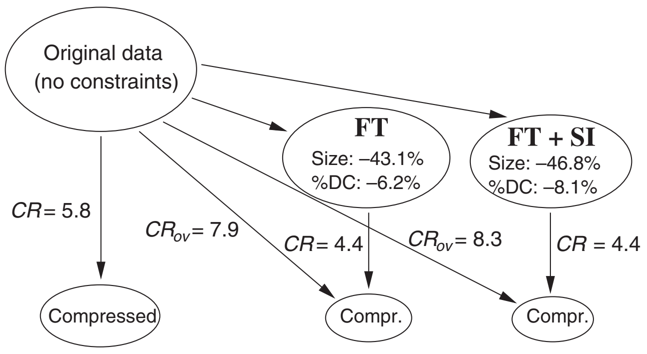

Fig. 5 Aggregated results for Constraints **FT** and **SI** (all circuits)

targeted delay fault model. FUJISAN is minimally intrusive for the existing flow as no modification of ATPG or the encoding procedure is needed.

We used FUJISAN to study the extent of overtesting for an off-the-shelf path delay fault ATPG with respect to various constraints and found it to be severe. In particular, the state reachability constraint lead to a significant decrease of functional instances. We also evaluated the effect of imposing functional constraints on test compression. We explored the tradeoff between the reduction in the size of the data to be compressed because of implicit relationships induced by the functional constraints on one hand and the decline of the compression ratio due to increased specification on the other hand. We found that most functional constraints result in decrease of the overall test data. One exception was again the state reachability constraint for which a drop in compression ratio was observed.

FUJISAN currently supports only exact methods. We plan incorporation of approximate techniques and hierarchical techniques such as [23] to make it scale for industrial-size circuits as future work. Support of circuits with incomplete specification is also a useful extension. A further needed feature is the automatic import of functional constraints from assertions in high-level HDL code.

Acknowledgements This work was supported in part by 21st Century COE (Center of Excellence) Program "Ubiquitous Networked Media Computing" and in part by JSPS (Japan Society for the Promotion of Science) under Grants-in-Aid for Scientific Research B(2)(No. 15300018). We are thankful to Prof. Sudhakar M. Reddy for fruitful discussions on overtesting.

References

1. Bryant R (1986) Graph-based algorithms for boolean function manipulation. IEEE Trans. Comput. 35(8):677–691
2. Dervisoglu B, Stong G (1991) Design for testability: using scanpath techniques for path-delay test and measurement. In: Int'l test conf., pp 365–374
3. Engelke P, Polian I, Renovell M, Seshadri B, Becker B (2004) The pros and cons of very-low-voltage testing: an analysis based on resistive short defects. In: VLSI test symp., pp 171–178
4. Fudoli A, Ascagni A, Appello D, Manhaeve H (2003) A practical evaluation of IDDQ test strategies for deep submicron production test application. experiences and targets from the field. In: European test workshop., pp 65–70
5. Giles G, Irby J, Toneva D, Tsai K-H (2005) Built-in constraint resolution. In: Int'l test conf.
6. Hao H, McCluskey E (1991) Resistive shorts within CMOS gates. In: Int'l test conf., pp 292–301
7. Henftling M, Wittmann H (1995) Bit parallel test pattern generation for path delay faults. In: European design and test conf., pp 521–525
8. Krstić A, Liou J-J, Cheng K-T, Wang L-C (2003) On structural vs. functional testing for delay faults. In: Int'l symp. on quality electronic design, pp 438–441
9. Lin Y-C, Lu F, Cheng K-T (2005a) Pseudo-functional scan-based BIST for delay fault. In: VLSI test symp., pp 229–234
10. Lin Y-C, Lu F, Yang K, Cheng K-T (2005b) Constraint extraction for pseudo-functional scan-based delay testing. In: Asia and South Pacific design autom. conf., pp 166–171
11. Liu X, Hsiao M (2003) Constrained ATPG for broadside transition testing. In: Int'l symp. on defect and fault tolerance in VLSI systems, pp 175–184
12. Liu X, Hsiao M (2005) A novel transition fault ATPG that reduces yield loss. IEEE Des. Test Comput. 22(6):576–584
13. Madge R, Benware B, Daasch W (2003) Obtaining high defect coverage for frequency-dependent defects in complex ASICs. IEEE Des. Test Comput. 20(5): 46–53
14. Mitra S, Avra L, McCluskey E (1997) Scan synthesis for one-hot signals. In: Int'l test conf., pp 414–422
15. Pecht M, Radojic R, Rao G (1998) Managing silicon chip reliability. CRC Press.
16. Pomeranz I (2004) On the generation of scan-based test sets with reachable states for testing under functional operation conditions. In: Design autom. conf., pp 928–933
17. Rajski J, Tyszer J, Kassab M, Mukherjee, N (2004) Embedded deterministic test. IEEE Trans. CAD 23(5):776–792
18. Rearick J (2001) Too much delay fault coverage is a bad thing. In: Int'l test conf., pp 624–633
19. Savir J (1994) Broad-Side delay test. IEEE Trans. CAD 13(8):1057–1064
20. Saxena J, Butler K, Jayaram V, Kundu S, Arvind N, Sreeprakash P, Hachinger M (2003) A case study of IR-drop in structured at-speed testing. In: Int'l test conf., pp 1098–1104
21. Tafertshofer P, Ganz A, Henftling M (1997) A SAT-based implication engine for efficient ATPG, equivalence checking, and optimization of netlists. In: Int'l Conf. on CAD, pp 648–655
22. Tehranipour M, Nourani M, Chakrabarty, K (2004) Nine-coded compression technique with application to reduced pin-count testing and flexible on-chip decompression. In: Design, automation and test in Europe, pp 173–178.
23. Vedula V, Abraham J (2000) A novel methodology for hierarchical test generation using functional constraint composition. In: Int'l high-level validation and test workshop, pp 9–14

Ilia Polian received the Diploma in computer science from the Albert-Ludwigs- University, Freiburg, Germany, and the Ph.D. degree in computer science (with distinction) from the same University, in 1999 and 2003, respectively. He was with the Micronas in Freiburg, IBM Germany R&D in Böblingen, and Nara Institute of Science and Technology (NAIST) in Nara, Japan. His research interests include defect modeling, design for testability and formal verification of hybrid and realtime systems. Currently he is a senior member of scientific staff at the Chair of Computer Architecture at the Albert-Ludwigs-University.

Dr. Polian was a European Champion and Vice World Champion at the 1999 ACM International Collegiate Programming Contest, Verband der Elektrotechnik Elektronik Informationstechnik e.V. (VDE) award laureate 1999 and Wolfgang-Gentner award laureate 2004. He is IEEE Member.

Hideo Fujiwara received the B.E., M.E., and Ph.D. degrees in electronic engineering from Osaka University, Osaka, Japan, in 1969, 1971, and 1974, respectively. He was with Osaka University from 1974 to 1985 and Meiji University from 1985 to 1994, and joined Nara Institute of Science and Technology in 1993. Presently he is a Professor at the Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan.

His research interests are logic design, digital systems design and test, VLSI CAD and fault tolerant computing, including high-level/logic synthesis for testability, test synthesis, design for testability, built-in self-test, test pattern generation, parallel processing, and computational complexity. He is the author of Logic Testing and Design for Testability (MIT Press, 1985). He received the IECE Young Engineer Award in 1977, IEEE Computer Society Certificate of Appreciation Awards in 1991, 2000 and 2001, Okawa Prize for Publication in 1994, IEEE Computer Society Meritorious Service Awards in 1996 and 2005, IEEE Computer Society Continuing Service Award in 2005, and IEEE Computer Society Outstanding Contribution Award in 2001.

He served as an Editor of the IEEE Trans. on Computers, Journal os Electronic Testing: Theory and Applications, Journal of Circuits, Systems and Computers, VLSI Design: An Application Journal of Custom-Chip Design, Simulation, and Testing, and several guest editors of special issues of IEICE Transactions of Information and Systems. He is currently an advisory member of IEICE Trans. of Information and Systems. Dr. Fujiwara is a fellow of the IEEE, a Golden Core member of the IEEE Computer Society, a fellow of IEICE (the Institute of Electronics, Information and Communication Engineers of Japan) and a fellow of the IPSJ (the Information Processing Society of Japan).