

Deterministic logic BIST for transition fault testing

V. Gherman, H.-J. Wunderlich, J. Schloeffel and M. Garbers

Abstract: Built-in self-test (BIST) is an attractive approach to detect delay faults because of its inherent support for at-speed test. Deterministic logic BIST (DLBIST) is a technique that has been successfully applied to stuck-at fault testing. As delay faults have lower random pattern testability than stuck-at faults, the need for DLBIST schemes has increased. However, an extension to delay fault testing is not trivial as this necessitates the application of pattern pairs. As a consequence, delay fault testing is expected to require a larger mapping effort and logic overhead than stuck-at fault testing. With this in mind, the authors consider the so-called transition fault model, which is widely used for complexity reasons, and an extension of a DLBIST scheme for transition fault testing is presented. Functional justification is used to generate the required pattern pairs. The efficiency of the extended scheme is investigated using difficult-to-test industrial designs.

1 Introduction

Delay fault testing has become a standard option in today's technology with path-delay, segment-delay, gate-delay and transition fault models having been proposed so far [1–6]. These models have differing levels of complexity for both test generation and test application.

The path-delay fault model [5] is used to model defects that are correlated along a path from a (pseudo-)primary input to a (pseudo-)primary output of the circuit under test (CUT). Both the switching delays of devices and the transport delays of interconnects may perturb the propagation of signal transitions. Unfortunately, in the worst case, the number of path-delay faults may increase exponentially with the number of signal lines in the CUT.

The segment-delay fault model [1, 4] contains all path segments shorter than a predefined parameter L . As L increases, the number of test segments may become unacceptably large. When L is too small, some paths with larger distributed delay may escape the test. The value of L can be chosen based on statistics about the manufacturing defects.

The gate-delay fault model [2] is a quantitative model in which the exact size of the delay fault needs to be specified in advance. Unfortunately, the determination of the fault size is not an easy task.

The transition fault model [3, 6], also called gross-delay fault model, is a special case of the gate-delay fault model in which the fault is assumed to be of the same order of magnitude as the clock period. The transition fault model is used to cover delay effects that are generated by localised (spot)

defects and whose sizes are in the order of magnitude of the clock cycle or of the test pattern period. A slow-to-rise and slow-to-fall transition fault may be associated to each signal line in the CUT. Consequently, the number of transition faults increases linearly with the number of signal lines in the CUT. Because of its limited complexity, the transition fault model is the most widely used and is the model of interest in this paper.

In order to test delay faults, two patterns are required: an initialisation pattern that sets the circuit to a predefined state, and an activation pattern that launches the appropriate transition and propagates the fault effect to a (pseudo-) primary output. There are two approaches for the application of pattern pairs to a standard scan design [6–8]: functional justification (also called broadside), and scan shifting (also called skewed-load). In the functional justification approach, the circuit response to the initialisation pattern is used as the second pattern. In order to apply pattern pairs, the circuit is operated two consecutive clock cycles in functional mode after the first pattern has been scanned in. In the scan-shifting approach, the second pattern is generated by operating the scan path for one additional clock cycle after the first pattern has been applied. Since scan shifting may require additional efforts for a consistent clocking scheme beyond the BIST hardware, we will concentrate only on functional justification. Another advantage of this approach is the expected limitation of the scan-induced over-testing, as long as the activation pattern is computed by the CUT itself and is not scanned in, like in the scan-shifting approach. So, the impact on the yield should be less than in the case of the scan-shifting approach.

There is a wide range of deterministic logic built-in self-test (DLBIST) methods that apply deterministic test patterns and, hence, improve the low-fault coverage often obtained by pseudo-random patterns. Very attractive schemes, which require no interaction with the automatic test equipment (ATE) during the test application time, are the test set embedding schemes. These rely on a pseudo-random test pattern generator plus some additional circuitry that modifies the pseudo-random sequence in order to embed a set of deterministic patterns. Examples for such techniques are the bit-fixing [9] and the bit-flipping [10, 11] DLBIST schemes. The implementation of these

© The Institution of Engineering and Technology 2007

doi:10.1049/iet-cdt:20060131

Paper first received 29th August and in revised form 22nd December 2006

V. Gherman is with CEA, LIST, Gif-sur-Yvette F-91191, France

H.-J. Wunderlich is with the Universitaet Stuttgart, Pfaffenwaldring 47, Stuttgart D-70569, Germany

J. Schloeffel and M. Garbers are with NXP semiconductors, Georg-Heykenstrasse 1, Hamburg D-21147, Germany

This article is an extension of the paper entitled 'Deterministic Logic BIST for transition fault testing', published in proceedings of European Test Symposium, 2006, pp. 123–128

E-mail: valentin.gherman@cea.fr

schemes includes the mapping of a set of deterministic test cubes to an initial pseudo-random test sequence and the synthesis of the circuitry used to embed the target test cubes [10].

Here, the bit-flipping DLBIST scheme used for the test of stuck-at faults will be adapted to transition fault testing based on functional justification. The resulting DLBIST scheme can be used to test transition faults in circuits with scan design. Until now, no such scheme has been described in the open literature. Logic BIST approaches for the test of delay faults (especially, for path delay faults) have been presented in [12–20], among others. The schemes proposed in [13, 17] are pseudo-deterministic, which means that a flexible test length is required in order to guarantee that all the target deterministic patterns are embedded. Only the approaches presented in [16, 19, 20] are deterministic, but they address circuits without scan designs.

The extension of the bit-flipping DLBIST scheme for transition fault testing requires the modification of the test control unit such that the scan enable signal allows two consecutive functional clock cycles and not only one as in the case of stuck-at fault testing.

Since pairs of test patterns are required, transition faults are more difficult to test than stuck-at faults. Consequently, the pseudo-random transition fault coverage is significantly lower than the pseudo-random stuck-at fault coverage. The final effect is an increase of the mapping effort and the logic overhead.

As a solution, this paper proposes a trade-off between these costs and the test application time. Slightly larger test application times reduce logic overhead and enhance test quality in terms of both transition fault and (non-modelled) defect coverage [21].

2 Pseudo-random testability of transition and stuck-at faults

Here, the random detection probability is used as a measure of fault testability. The probability that a stuck-at ‘ i ’ fault ($i \in \{0, 1\}$) on the signal line J is tested by a random pattern is equal to the probability $P(J \text{ sa } 'i')$ that the line J can be controlled to the logic value ‘ $\neg i$ ’ and observed at a (pseudo-)primary output [22, 23]:

$$P(J \text{ sa } 'i') = P(J \text{ is controllable to } '\neg i' \text{ and } J \text{ is observable}) \quad (1)$$

The probability that a transition fault from ‘ i ’ to ‘ $\neg i$ ’ on the signal line J is tested by a pair of independent random patterns is equal to the probability $P(J \text{ slow from } 'i' \text{ to } '\neg i')$ that the first pattern controls the line J to the logic value ‘ i ’ multiplied by the probability that the second pattern controls the line J to the logic value ‘ $\neg i$ ’ and can

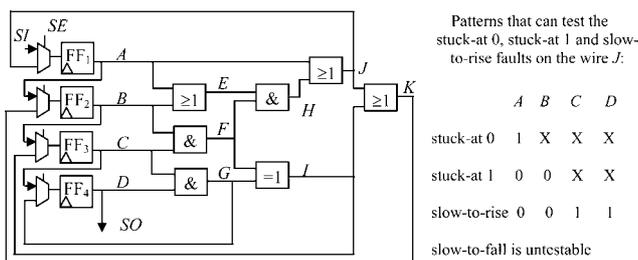


Fig. 1 Specified bits for testing stuck-at and transition faults

make the signal line J visible at a (pseudo-)primary output

$$P(J \text{ slow from } 'i' \text{ to } '\neg i') = P(J \text{ is controllable to } 'i') * P(J \text{ sa } '\neg i') \quad (2)$$

From the relation (2), the results is that if the stuck-at ‘ i ’ fault on a signal line is random pattern resistant, then the transition fault from ‘ i ’ to ‘ $\neg i$ ’ on the same line is random pattern resistant as well. Consequently, a digital circuit contains at least as many random pattern resistant transition faults as random pattern resistant stuck-at faults.

The following expression gives the number N_f of random patterns required to test a fault ‘ f ’ having the detection probability P_f ($\ll 1$) with a test confidence C (probability that the considered fault is tested by at least one pattern of the test sequence) [24]

$$N_f \simeq -\ln(1 - C)/P_f \quad (3)$$

Consequently, for two faults ‘ f ’ and ‘ g ’, we have

$$N_f \simeq N_g * P_g/P_f \quad (4)$$

From the relations (2) and (4), one can observe that in the case when controllability to i of a signal line is close to ‘0’, a slow transition fault from ‘ i ’ to ‘ $\neg i$ ’ on the same line may require a much larger number of random test patterns than the corresponding stuck-at fault, if the same test confidence is the objective.

As an example, consider the circuit sketched in Fig. 1. The inputs of this circuit are driven by four scan flip-flops. A scan enable (SE) signal is used to switch the scan flip-flops between scan and functional modes. The possible patterns to test the stuck-at ‘0’ and ‘1’ faults on the net J are $ABCD \in \{1XXX, X11X\}$ and $\{00XX, 0X0X\}$, respectively. Based on functional justification, the initialisation pattern $ABCD = 0011$ will generate an activation pattern 0111 such that a slow-to-rise fault on the net J can be tested. The slow-to-fall transition fault on the net J cannot be tested.

Note that there is only one initialisation pattern for the slow-to-rise fault on the net J , which has four specified bits, while each of the two corresponding stuck-at faults has two test patterns with one or two specified bits. The probabilities to randomly detecting the slow-to-rise, the stuck-at ‘0’ and the stuck-at ‘1’ faults on the net J are 1/16, 5/8 and 3/8, respectively. The reduced testability of the transition fault is because of the fact that the initialisation pattern must not only set the required initial logic value on the target line, but it must also generate appropriate logic values at the CUT outputs in order to define a useful activation pattern.

Fig. 2 presents a comparison between the cumulative stuck-at and transition fault coverage of a pseudo-random test sequence applied to a difficult-to-test industrial design

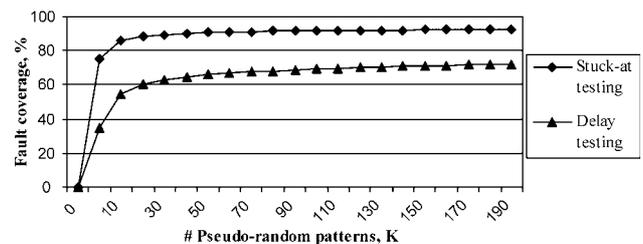


Fig. 2 Cumulative stuck-at and transition fault coverage of a pseudo-random sequence applied to a difficult-to-test industrial design

Transition fault test was based on functional justification

that contains 5116 flip-flops arranged into 11 scan chains. The lower level and the slower saturation of the transition fault coverage is because of the larger number of random pattern resistant transition faults and to the larger number of pseudo-random patterns required by these faults to achieve the same test confidence as in the case of the random resistant stuck-at faults.

This slow saturation is expected for any type of delay fault testing and it may be enhanced in the case of robust delay fault testing. It increases the necessity of having long test sequences when DLBIST is used. Moreover, a DLBIST scheme is necessary that is able to use the whole test sequence for pseudo-random fault detection.

3 Bit-flipping DLBIST for transition faults

The bit-flipping DLBIST scheme provides both pseudo-random and deterministic test patterns. Most of the pseudo-random test patterns do not contribute to the fault coverage as they only detect faults already detected by previous pseudo-random patterns. Such useless pseudo-random test patterns may be modified into useful deterministic test patterns to improve the fault coverage. The deterministic test patterns are determined by an automatic test pattern generation (ATPG) tool, and they target those testable faults not detected by the pseudo-random test sequence. In such a deterministic test pattern (cube), only a few bits are actually specified, while most of the bits are don't cares and hence can be arbitrarily set to '0' or '1'.

The modification of the pseudo-random patterns is realised by inverting (flipping) some of the linear feedback shift register (LFSR) outputs, such that they become compatible with the deterministic test cubes. This flipping is performed by combinational logic, which implements a bit-flipping function (BFF). The BFF can be kept quite small by exploiting the large number of useless pseudo-random patterns that may be modified, and carefully selecting the pseudo-random patterns that are altered to deterministic patterns.

As shown in Fig. 3, the BFF inputs are connected to the pattern counter (PC), shift counter (SC) and LFSR, while the BFF outputs are connected to the XOR-gates at the scan inputs. The BFF determines whether a bit has to be flipped based on the states of the PC, the SC and the LFSR. The PC is part of the test control unit, and counts the number of test patterns applied during the self-test. The SC is also part of the test control unit and counts the number of clock cycles required for shifting test data in (and out of) the scan chains. If a phase shifter (PS) is attached to the LFSR, its output may be used to control the operation of the BFF as well.

The BFF realises the mapping of deterministic test cubes to pseudo-random test patterns in the following way. Every specified bit (i.e. care bit) in a deterministic cube either matches to the corresponding bit in the pseudo-random pattern, in which case no bit-flipping is required, or the bit

does not match, in which case bit-flipping is required. For all unspecified bits (i.e. don't-care bits) in a deterministic cube, the corresponding bits in the associated pseudo-random pattern may be arbitrarily flipped or not.

The BFF must ensure that all conflicting bits are flipped, all matching bits are not flipped and that the don't-care bits may be arbitrarily flipped or not. A binary decision diagram (BDD)-based method for the generation (mapping) and the logic implementation of the BFF that scales well with the size of the CUT is described in [10].

Below, each pattern of a test sequence that can detect faults not detected by any of its precedent patterns will be referred to as an essential pattern. The embedding of deterministic test cubes into a pseudo-random test sequence may corrupt the essential pseudo-random patterns even if they are not assigned to deterministic cubes. This is because of the fact that the logic synthesis and optimisation of the BFF are intensively using its don't care (DC)-set (set of input assignments for which it does not matter whether they are mapped to '1' or '0') [10]. Consequently, the resulting bit-flipping logic flips more bits than necessary for the embedding of the target deterministic test cubes and the consequence is that essential pseudo-random test patterns may be corrupted.

Because of the slower saturation of the random transition fault coverage, the application of longer test sequences and the protection of all the essential pseudo-random patterns becomes critical. A way to prevent the corruption of the essential patterns without increasing the complexity of the BFF implementation is to utilise an additional combinational module, referred to here as correction logic (CRL), to enable/disable the outputs of the bit-flipping logic (Fig. 3).

In the case of transition fault testing based on functional justification, the test control unit (Fig. 3) has to be able to provide pairs of capture clock cycles. In such a scheme, only the initialisation pattern of each pair of test patterns is generated by the DLBIST hardware. The activation pattern is generated by the CUT as a response to the first pattern and only single test cubes have to be embedded into the pseudo-random sequence (just like in the case of stuck-at fault testing). This is also true in the case of transition fault testing based on scan shifting, with the observation that in this case, the test control unit should generate one additional shift clock cycle instead of the first capture clock cycle. For this reason, as long as the ATPG support is provided, the methodology presented here can also be applied to the scan-shifting approach. Moreover, this methodology is orthogonal to any method used to extend transition faults testing to circuits with multi-cycle path logic, like modifying the wave form of shift and capture clock signals or enhancing the scan design [25].

The steps used to implement the proposed scheme are detailed below.

1. Initial fault simulation is used to detect the transition faults that cannot be tested by the pseudo-random test sequence produced by an LFSR and, optionally, a PS. During this step, a list L_1 is generated containing the indices of all the essential pseudo-random test patterns.
2. An ATPG tool is used to generate a limited number of deterministic initialisation test cubes for all or a sub-set of the transition faults that remained undetected by the pseudo-random test sequence.
3. A pseudo-random test pattern is assigned to each deterministic initialisation test cube. For each such cube, only a limited number of pseudo-random patterns are checked starting with the ones at the minimum Hamming distance.

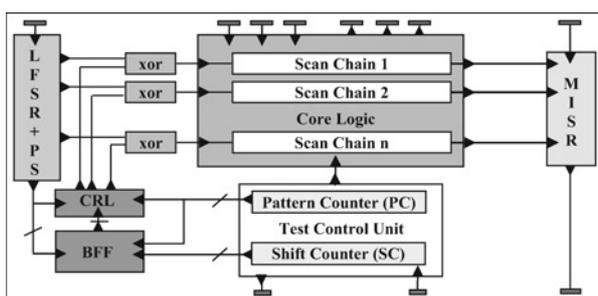


Fig. 3 Architecture of the bit-flipping DLBIST

The essential pseudo-random test patterns are not allowed to be assigned. Each assigned pseudo-random test pattern is modified by bit-flipping to become compatible with the corresponding deterministic test cube. During this step, a BDD-based representation is generated for the resulting BFF [10]. The BFF is only partly specified and has a large DC-set.

4. The BDD-based representation of the BFF is optimised and transformed into a register transfer level (RTL) circuit description [10, 26]. The logic optimisation procedure exploits the DC-set left by the incomplete specification of the BFF. Consequently, the optimised bit-flipping logic flips additional bits besides the conflicting bits in the assigned pseudo-random patterns, so that essential pseudo-random patterns (with the index included in L_1) may be corrupted.

5. In order to determine which of the patterns with the index in L_1 have to be protected from being corrupted, the embedded test sequence produced by the LFSR, PS (if any) and the bit-flipping logic, which is not allowed to act on the patterns with the index in L_1 , is simulated to generate a second list L_2 of indices corresponding to the essential patterns in the embedded test sequence. The patterns whose indices are included in $L_1 \cap L_2$ need to be protected from being corrupted by the bit-flipping logic. On the other hand, the patterns whose indices are included in $L_2 - L_1$ should remain modified by the bit-flipping logic.

6. A CRL (Fig. 4) is built to prevent the bit-flipping logic from flipping the essential patterns that are referenced in $L_1 \cap L_2$. The modification of the patterns with the index in $L_2 - L_1$ is allowed. In this way the essential patterns will not be corrupted, while the useless patterns that become useful by means of bit-flipping can still be generated. The CRL implements an incompletely specified function, whose ON-set and OFF-set (sets of input assignments that must be mapped to '1' and '0', respectively) contain the states of the PC, LFSR and the output of the PS (if any) corresponding to the first scan clock cycle of the test patterns with the index in $L_2 - L_1$ and $L_1 \cap L_2$, respectively. The DC-set of the CRL function is used to optimise its implementation, exactly as in the case of the BFF implementation [10, 26]. The output of the CRL has to be kept constant during the scan clock cycles corresponding to each test pattern. Because of the fact that some of the input signals to the CRL (the state bits of the LFSR and the output bits of the PS (if any)) change during the scan clock cycles, a latch is used to store the output of the CRL. The operation of the latch is controlled with the help of the scan enable (SE) signal, so that its state can be changed only before a new test pattern is scanned in. Without the CRL, all the inputs of the BFF corresponding to the essential pseudo-random patterns should be included in the OFF-set of the BFF. Consequently, the use of the CRL leaves more DC space to optimise the logic implementation of the BFF. On the other hand, storing the output of the CRL into a memory element that cannot be written during the shift cycles increases significantly the degrees of freedom for the optimisation of the CRL.

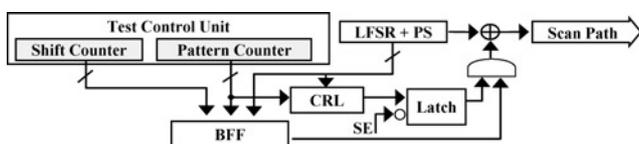


Fig. 4 Bit-flipping function (BFF) and correction logic (CRL)

While a BFF function has to be implemented for each scan chain, the CRL is common for all the scan chains.

7. In the end, a simulation of the embedded test sequence generated by the LFSR, PS (if any), BFF and CRL determines the final transition fault coverage.

In order to limit the size of the correction logic, the set of deterministic cubes is embedded (mapped) only at the end of the pseudo-random test sequence. The length of the pseudo-random sequence that can be modified is a fraction given by a negative power of two of the total test length. The first part of the test sequence is used only for pseudo-random fault detection and it is protected from being flipped by disabling the outputs of the BFF with the help of an AND gate per scan chain, which is controlled by a combination of the most significant bits of the PC.

4 Experimental results

The experimental results reported below have been obtained using GNU Linux machines equipped with 2 GB of memory and an Intel Pentium 4 processor running at 2.4 GHz. The industrial designs described in Table 1 have been used as benchmark circuits. An industrial computer-aided test (CAT) tool has been employed to provide ATPG and fault simulation support. In the tool version available to us, it was assumed that these circuits contained only single-cycle paths. The first column (Design) reports the circuit name encoded as pN , where N denotes the number of nodes in the design. The next two columns give the number of scan flip-flops (# Flip-flops) and scan chains (# Scan chains) contained by each design. The following column

Table 1: Characteristics of the considered industrial designs

Design	# Flip-flops	# Scan chains	Test length	Random stuck-at fault efficiency, %	Random transition fault efficiency, %
p19K	1407	29	10K	80.80	73.66
			32K	85.54	82.64
			64K	90.38	86.97
			1923K	95.87	90.74
p59K	4730	20	10K	97.12	81.87
			32K	97.94	85.63
			64K	98.11	87.31
			192K	98.35	89.67
p127K	5116	11	10K	84.42	55.83
			32K	89.39	64.80
			64K	91.65	68.53
			187K	93.75	73.82
p278K	9967	32	10K	84.29	63.86
			32K	88.66	71.02
			64K	90.62	75.00
			318K	93.38	82.81
p333K	20756	30	10K	95.62	66.66
			32K	96.73	73.39
			64K	97.14	76.39
			140K	97.51	78.26

Table 2: DLBIST applied to stuck-at and transition fault test, in the case of 10K patterns long test sequence

Design	Stuck-at fault testing				Transition fault testing			
	# Embedded patterns	Ratio of specified bits, %	Final fault efficiency, %	Cell area overhead, %	# Embedded patterns	Ratio of specified bits, %	Final fault efficiency, %	Cell area overhead %
p19K	181	26.48	99.19	25	145	10.64	94.40	17
p59K	137	2.77	99.10	4	1077	3.00	96.43	26
p127K	582	12.04	99.26	22	800	15.24	76.35	43
p278K	1549	6.10	98.87	35	800	14.48	86.66	62
p333K	1298	0.75	99.30	7	800	2.94	84.95	22

(Test length) shows the length of the test sequence. The last two columns report the pseudo-random stuck-at and transition fault efficiencies, respectively. The fault efficiency is defined as the percentage of the detected faults with respect to the total number of testable faults. For each design, the last entry line corresponds to a test sequence whose application would require 1 s at a frequency of 100 MHz. It can be easily observed that the transition fault efficiency is much lower. In general, increasing the test sequence length has a stronger impact on the transition fault detection than on the stuck-at fault detection.

In Table 2, one can compare the results obtained using the bitflipping DLBIST approach for the stuck-at and transition fault testing of the designs in Table 1. Table 2 reports the number of embedded deterministic test cubes, the ratio (percent) of specified bits in the set of embedded cubes, the achieved final fault efficiency and the cell area overhead of the BFF and CRL implementations for both fault models. The overhead of the other parts of the DLBIST hardware is relatively small and it may be neglected.

In order to limit the hardware overhead in the case of the three largest designs, the number of deterministic test cubes

embedded for transition fault testing has been limited to 800.

With the exception of the design p19K, the deterministic test cubes embedded for transition fault testing have larger ratios of specified bits. This is because of the lower transition fault testability. In the case of the design p19K, this lower testability has the effect that the ATPG tool delivers less deterministic test cubes with less detected faults and also less specified bits per cube than in the case of stuck-at fault testing.

In all cases, the final stuck-at fault efficiency is much higher than the final transition fault efficiency. Moreover, this has been achieved along with a lower cell area overhead with the exception of the design p19K. The reason for this difference is again the lower random testability of the transition faults with the consequence that more patterns have to be embedded and more bits have to be flipped or preserved in the pseudo-random sequence. This seems not to be the case in the p19K design, but, as mentioned before, it was just the ATPG that provided fewer deterministic test cubes to be embedded for transition fault testing. For a given test length, the DLBIST hardware overhead depends

Table 3: Results obtained with the bit-flipping DLBIST used for transition fault testing

Design	Test length	# Embedded patterns	Run-time, h:m	Memory, Mb	Final fault efficiency, %	Fault efficiency improvement, %	Cell area overhead, %
p19K	10K	145	00:16	58	94.40	20.74	17
	32K	125	00:24	61	94.40	11.76	11
	64K	105	00:23	67	94.40	7.43	7
	1923K	54	04:01	577	94.41	3.67	4
p59K	10K	1077	07:22	252	96.43	14.56	26
	32K	942	06:19	240	96.55	10.92	20
	64K	865	05:45	230	96.64	9.33	18
	192K	738	05:53	286	96.69	7.02	15
p127K	10K	800	32:55	716	76.35	20.52	43
	32K	800	32:09	738	82.20	17.40	44
	64K	800	31:47	755	84.98	16.45	44
	187K	800	30:17	786	87.75	13.93	42
p278K	10K	800	29:01	1408	86.66	22.80	62
	32K	800	30:34	1415	90.24	19.22	57
	64K	800	32:16	1431	91.84	16.84	54
	318K	800	48:37	1508	94.93	12.12	51
p333K	10K	800	33:40	758	84.95	18.29	22
	32K	800	35:58	760	86.77	13.38	19
	64K	800	35:29	742	87.61	11.22	17
	140K	800	35:39	801	88.25	9.99	15

Table 4: Use of the maximum test length which can fit in one second of test time at the frequency of 100 MHz

Design	Test length	# Embedded patterns	Run-time, h:m	Memory, MB	Final fault efficiency, %	Fault efficiency improvement, %	Cell area overhead, %
p127K	187K	10	5:16	477	76.83	3.01	1
		50	5:38	492	79.48	5.66	3
		100	6:41	510	80.56	6.74	5
		400	16:22	731	85.28	11.46	24
		800	30:17	786	87.75	13.93	42
p278K	318K	10	23:44	1150	87.96	5.15	1
		50	24:00	1169	88.98	6.17	4
		100	26:44	1190	90.71	7.90	6.5
		400	34:53	1306	93.26	10.45	27.5
		800	48:37	1508	94.93	12.12	51
p333K	140K	10	5:33	528	81.29	3.03	1.5
		50	7:36	630	83.32	5.06	4
		100	9:44	661	84.47	6.21	5.5
		400	31:05	786	87.28	9.02	11.5
		800	35:39	801	88.25	9.99	15

on the random testability of the CUT and on the amount with which the fault efficiency has to be increased.

In Table 3, one can observe the impact of increasing the test length on the final fault efficiency can be observed as well as the cell area overhead (BFF + CRL) of the considered DLBIST scheme used for transition fault testing. Run-time and memory requirements are reported as well.

As expected, the hardware overhead of the first two designs, for which the number of embedded patterns has not been limited, is significantly reduced by the increase of the test length. Extending the test length from 10K to 64K reduces the overhead by more than 10% of the CUT size. In the case of the last entry corresponding to the design p19K, increasing the test length by two orders of magnitude has reduced the overhead to half of the level from the previous entry, at the price of a large increase in run-time and memory requirements.

As in the case of the previous experiments, the number of embedded deterministic test cubes for the three largest designs has been limited to 800, in order to limit the hardware overhead. As long as the same number of deterministic test cubes is embedded, it is difficult to predict the dependence of the hardware overhead on the length of the test sequence. In this case, the overhead primarily depends on the average number of specified bits per embedded test cube, which is determined by the number and the difficulty of the target faults. Longer pseudo-random test sequences leave undetected faults that are more difficult to test. This tends to increase the number of specified bits necessary to detect the remaining fault. On the other hand, this may also decrease the number of newly detected faults per embedded test cube. That is why it is difficult to predict the evolution of the average number of specified bits per embedded test cube when the length of the test sequence is augmented. Increasing the length of the test sequence also improves the pattern embedding opportunities. In the case of the design p127K, increasing the length of the test sequence does not significantly change the overhead, but it improves the final fault efficiency by more than 11%. In the case of the designs p278K and p333K, increasing the test sequence length has a twofold beneficial impact. Choosing a test sequence length of 318K and 140K,

instead of 10K reduces the overhead by 11% and 7%, respectively. In parallel, the final fault efficiency is improved by more than 8% and 3%, respectively. The increase of the test length improves the coverage of the non-modelled defects as well [21]. In the case of the three largest designs, increasing the length of the test sequence has no significant impact on the run-time and memory requirements.

Table 4 reports possible trade-offs between the fault efficiency and the hardware overhead in the case of the largest three designs. The considered test sequences contain the maximum number of test patterns that can fit in one second of test time at the frequency of 100 MHz. In the case of the designs p127K and p278K, 10 deterministic patterns are already enough to obtain a larger fault efficiency than in the case when 800 deterministic test patterns are embedded into a 10K-long test sequence. In this way, the hardware overhead can be reduced to 1% from 43% and 62% (Table 3), respectively. In the case of the design p333K, a similar fault efficiency can be achieved by embedding 100 deterministic test patterns, at the cost of 5.5%, instead of 22%, hardware overhead.

5 Conclusions

We have presented an extension of the bit-flipping DLBIST scheme for transition fault testing. This is the first time that a DLBIST scheme has been used to test for transition faults in circuits with a standard scan design. A special combinational module, the correction logic, has been introduced to further improve the pattern embedding. Due to the rather low random-pattern testability of transition faults, the saturation of their random fault coverage requires significantly longer test sequences, which in turn is beneficial for both limiting the hardware overhead and improving the coverage of the modelled and non-modelled defects. The DLBIST scheme considered here can be applied to both approaches to transition fault testing in circuits with standard scan design, functional justification and scan shifting. Moreover, the presented methodology is orthogonal to any method that can be used to extend transition faults testing to circuits with multi-cycle path logic.

6 Acknowledgments

This research work was supported by the German Federal Ministry of Education and Research (BMBF) in the Project AZTEKE under contract number 01M3063C.

7 References

- 1 Heragu, K., Patel, J.H., and Agrawal, V.D.: 'Segment delay faults: a new fault model'. IEEE VLSI Test Symposium (VTS), 1996, pp. 32–39
- 2 Carter, J.L., Iyengar, V.S., and Rosen, B.K.: 'Efficient test coverage determination for delay faults'. IEEE Int. Test Conf. (ITC), 1987, pp. 418–427
- 3 Krstic, A., and Cheng, K.T.: 'Delay fault testing for VLSI circuits' (Kluwer Academic Publishers, Boston MA, 1998)
- 4 Sharma, M., and Patel, J.H.: 'Enhanced delay defect coverage with path-segments'. IEEE Int. Test Conf. (ITC), 2000, pp. 385–392
- 5 Smith, G.L.: 'Model for delay faults based upon paths'. IEEE Int. Test Conf. (ITC), 1985, pp. 342–349
- 6 Waicukauski, J.A., Lindbloom, E., Rosen, B.K., and Iyengar, V.S.: 'Transition fault simulation', *IEEE Design and Test of Computers*, 1987, **4**, pp. 32–38
- 7 Savir, J.: 'Skewed-load transition test. Part I: Calculus'. IEEE Int. Test Conf. (ITC), 1992, pp. 705–713
- 8 Savir, J., and Patil, S.: 'Broad-side delay test', *IEEE Trans. CAD Integr. Circuits Syst.*, 1994, **13**, (8), pp. 1057–1064
- 9 Touba, N.A., and McCluskey, E.J.: 'Altering a pseudo-random bit sequence for scan-based BIST'. IEEE Int. Test Conf. (ITC), 1996, pp. 167–175
- 10 Gherman, V., Wunderlich, H.-J., Vranken, H., Hapke, F., Wittke, M., and Garbers, M.: 'Efficient pattern mapping for deterministic logic BIST'. IEEE International Test Conf. (ITC), 2004, pp. 48–56
- 11 Wunderlich, H.-J., and Kiefer, G.: 'Bit-Flipping BIST'. IEEE Int. Conf. on CAD (ICCAD), 1996, pp. 337–343
- 12 Chen, C.A., and Gupta, S.K.: 'BIST test pattern generators for two-pattern testing—theory and design algorithms', *IEEE Trans. Comput.*, 1996, **45**, (3), pp. 257–269
- 13 Dufaza, C., and Zorian, Y.: 'On the generation of pseudo-deterministic two-patterns test sequence with LFSRs'. IEEE European Design and Test Conf. (EDAC), 1997, pp. 69–76
- 14 Furuya, K., and McCluskey, E.J.: 'Two-pattern test capabilities of autonomous TPG circuits'. IEEE International Test Conference (ITC), 1991, pp. 704–711
- 15 Girard, P., Landrault, C., Moreda, V., and Pravossoudovitch, S.: 'An optimized BIST test pattern generator for delay testing'. IEEE VLSI Test Symposium (VTS), 1997, pp. 94–100
- 16 Keim, M., Polian, I., Hengster, H., and Becker, B.: 'A scalable BIST architecture for delay faults'. IEEE European Test Workshop (ETW), 1999, pp. 98–103
- 17 Li, W., Yu, C., Reddy, S.M., and Pomeranz, I.: 'A scan BIST generation method using a markov source and partial bit-fixing'. IEEE Design Automation Conf. (DAC), 2003, pp. 554–559
- 18 Mukherjee, N., Chakraborty, T.J., and Bhawmik, S.: 'A BIST scheme for the detection of path-delay faults'. IEEE Int. Test Conf. (ITC), 1998, pp. 422–432
- 19 Starke, W.: 'Built-in test for CMOS circuits'. ITC'84, 1994, pp. 309–314
- 20 Wurth, B., and Fuchs, K.: 'A BIST approach to delay fault testing with reduced test length'. IEEE Europ. Design and Test Conf. (EDAC), 1995, pp. 418–423
- 21 Engelke, P., Gherman, V., Polian, I., Tang, Y., Wunderlich, H.-J., and Becker, B.: 'Sequence length, area cost and non-target defect coverage tradeoffs in deterministic logic BIST'. IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2005, pp. 11–18
- 22 Brglez, F.: 'On testability analysis of combinational networks'. IEEE Int. Symp. on Circuits and Systems (ISCAS), 1984, pp. 221–225
- 23 Savir, J., Ditlow, G.S., and Bardell, P.H.: 'Random pattern testability', *IEEE Trans. Comput.*, 1984, **33**, (1), pp. 79–90
- 24 Wunderlich, H.-J.: 'PROTEST: a tool for probabilistic testability analysis'. ACM/IEEE Design Automation Conf. (DAC), 1985, pp. 204–211
- 25 Pateras, S.: 'Achieving at-speed structural test', *IEEE Design and Test of Computers*, 2003, **20**, (5), pp. 26–33
- 26 Gherman, V., Wunderlich, H.-J., Mascarenhas, R., Schloeffel, J., and Garbers, M.: 'Synthesis of irregular combinational functions with large don't care sets'. ACM Great Lakes Symp. on VLSI (GLSVLSI), 2007