High-Diagnosability Online Built-In Self-Test of FPGAs via Iterative Bootstrapping*

Vishal Suthar
University of Illinois at Chicago
Chicago, IL 60607
vsutha1@uic.edu

Shantanu Dutt
University of Illinois at Chicago
Chicago, IL 60607
dutt@ece.uic.edu

ABSTRACT

We develop a novel on-line built-in self-test (BIST) technique for testing FPGAs that has a very high diagnosability even in presence of clustered faults, a fault pattern for which previous BIST methods proved ineffective. Using an iterative bootstrapping process, our method first finds a fault-free test circuit in each BISTer tile and then tests the PLBs functionally using a fault-detection-and-gross-diagnosis phase followed by a time-efficient adaptive diagnosis phase. We establish the correctness of the deterministic phases of our BIST technique. We also analyze the probability of correct diagnosis by our BISTer in the presence of multiple random faults. Simulation results show that our BIST technique has very high fault coverage (98.7% for 25% fault density for random faults and 98.9% for 8.8% fault density for clustered faults) and low fault latency, and supports the theoretical analysis.

Categories and Subject Descriptors

B.7.0 [**Integrated Circuits**]: General; B.8.1 [**Performance and Reliability**]: Reliability, Testing, and Fault Tolerance

General Terms

Algorithms, Design, Reliability

Keywords

built-in self-test (BIST), clustered faults, diagnosability, FPGAs, functional testing, on-line testing, roving tester

1. INTRODUCTION

With current technology trend heading towards nano-scale FPGAs, the fault densities in the FPGAs are expected to increase significantly. Current on-line testing methods have reasonable to high fault coverages for lower fault densities but their fault coverages decrease dramatically with increasing fault density. Also, these methods are ineffective in the presence of clustered faults, wherein a group of adjacent PLBs or interconnects are faulty, that occur frequently in fabrication defect scenarios. In this paper we propose a new BIST-based on-line testing method, High-Diagnosability BIST (HD-BIST), for testing the PLBs of the FPGA that has much higher diagnosability, for random as well as clustered faults, than previous on-line [1, 2, 3, 7, 8] as well as off-line [4, 5, 6] testing methods[1]. In particular, our method outperforms the two previous best on-line BIST methods [1, 2], both in terms of fault coverage and fault latency. Also, our method is easily adaptable to and efficient for off-line testing. The basic idea of on-line testing is to test a small area (referred here as the ROving TEster (ROTE)) of the FPGA while the remaining area is running the application circuit. Once testing of the area under the ROTE is completed, the ROTE is dynamically reconfigured to test another small portion of the FPGA. The ROTE comprises of multiple Built-in Self-tester (BISTer) tiles that test different sub-areas of the ROTE simultaneously. Each BISTer (Fig. 1a) consists of a test pattern generator (TPG) that generates test vectors that are applied to two identically configured blocks or cells under test (CUTs). The outputs of the CUTs are compared by an output response analyzer (ORA). Testing of two PLBs as CUTs in this manner is called a *session*. In the next session, the BISTer is reconfigured so that different PLBs will function as the TPG, CUTs and ORA (Fig. 1b). In this manner, the testing of the entire BISTer tile is completed in a few sessions.

Definition: The *gross syndrome* (GS) is an overall *fail/pass* result of a session; a *fail (X)* if the ORA output is 1 (indicating mismatch in two CUT's outputs) for any input test vector and a *pass* ($\surd$) if the ORA output is 0 (indicating match) for all input test vectors.

2. HIGH-DIAGNOSABILITY (HD) BISTER

2.1 HD-BIST versus Previous BIST Methods

We introduce the design (Fig. 1c) and the three phases (Fig. 1d) of HD-BISTer and compare it with two previous best online BIST methods, STAR [2] and BISTer-1 [1].

1. Design: Each HD-BISTer tile consists of 6 PLBs–2 CUTs, 1 ORA and 3 TPG PLBs[2]. Unlike previous methods, the relative positions of TPG, ORA and CUTs are not fixed and all the PLB functions are not rotated in subsequent sessions. Instead, each HD-BISTer tile is conceptually

*This work was supported in part by a grant from Xilinx Inc. and in part by NSF grant CCR-0204097.

[1]This paper focuses on PLB testing; interconnect testing will be addressed in future work and hence throughout this paper interconnects are assumed to have been tested and the faulty interconnects isolated.

[2]The ratio of the number of input to output pins in a PLB in many commercial FPGAs is $\approx 3:1$ and we have assumed the same here, although this is not stringent. In accordance with this, a minumum 3 and 1 PLBs are required for the TPG and ORA, respectively.

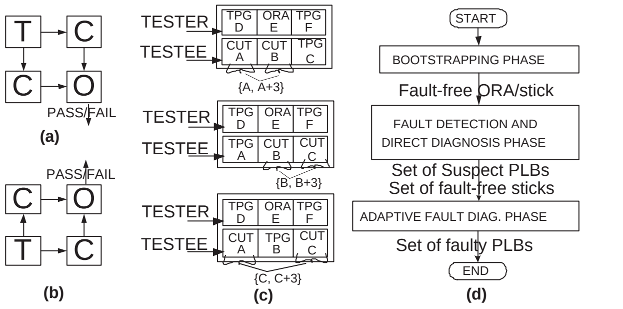


Figure 1: *(a),(b) Two different sessions of a general BISTer: T, C and O denote TPG, CUT and ORA, respectively. (c) Three sessions of testing a testee stick in HD-BISTer. X1 = X and X2 = X+3 indicate the two operational functions for testing CUTs in a session. (d) The three phases of HD-BISTer.*

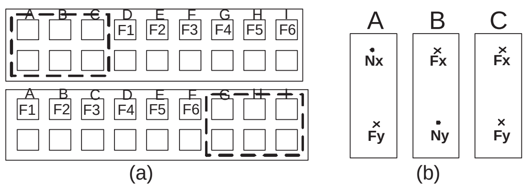


Figure 2: *(a) Concept of functional testing: Each PLB (say, D) performs only two functions (F1 and F4) during on-line testing. Area enclosed in dashed lines is the ROTE area. (b) A scenario exemplifying inference 2 of Case 2 in fault detection phase: Fx and Fy indicate faults and Nx and Ny indicate non-faulty locations corresponding to locations of faults in PLB C.*

divided into two 1×3 sub-areas, which we refer to as *sticks*. One of the two sticks acts as a *testee* stick and its PLBs are tested as CUTs in three different sessions, while the other stick acts as a *tester* stick providing the ORA and part of the TPG (Fig. 1c). In each session, a different pair of PLBs from testee stick act as CUTs. The tester's configuration remains unaltered in all sessions.

Assumptions:
We will analyze HD-BISTer under the following realistic (high probability) assumptions:
1: All three CUT PLBs in a stick, if faulty, do not have identical output responses for all the input test vectors.
2: There is at least one fault-free stick in each ROTE area and one fault-free ORA in the corresponding tester stick to correctly detect the fault-free stick in the fault detection phase.

2(a). Bootstrapping phase: One of the drawbacks of previous on-line BIST methods, e.g., [1, 2], is that they directly start testing CUTs without making sure that the TPG and ORA PLBs used in testing are fault-free either with certainty or with high probability, thereby reducing their diagnosability. Our technique does not directly start testing CUTs but first determines one fault-free ORA per stick using an *iterative bootstrapping process*. We also use a "TPG shuffling" scheme in this phase which results in the probability of a faulty TPG skipping test vectors to be very low.

2(b). Fault detection and gross diagnosis phase: This is the second phase in HD-BISTer wherein PLBs of the testee stick are tested as CUTs in three sessions as discussed above; for brevity we will refer to this phase as the *fault detection phase*. Note that, unlike previous methods, we are not setting any upper-bound on the number of faulty PLBs and thus the gross syndromes will not always definitely identify the set of faulty PLBs. From the gross syndrome obtained we can, however, determine which PLBs *may be* faulty. These PLBs are termed as *suspect* PLBs; the set of suspect PLBs is the superset of the set of faulty PLBs with high probability. In this phase we also determine which sticks are fault-free. We use the concept of functional testing introduced in [1]. As shown in Fig. 2a during on-line testing, each PLB X performs only two functions, one is the original function *X1* (X in Fig. 1c) assigned to it during circuit mapping, while the other is the function *X2* (*X+3* in Fig. 1c) assigned to it when the ROTE moves to its next location, three columns to the right (hence the *X+3* notation). *X1* and $X2$ are called the PLB's *operational functions*. Accordingly, a PLB is said to be *functionally faulty* (*f-faulty*) if faults in the PLB cause incorrect output(s) to be produced for one or more input vectors when the PLB is implementing its operational functions. Similarly *non-functional faulty (nf-faulty)* PLBs represent PLBs that are faulty but not f-faulty and thus can perform the circuit function correctly.

3(c). Adaptive diagnosis phase: The *adaptive diagnosis phase* is used to re-test the suspect PLBs to determine their correct fault state. BISTer-1 [1] does not have any adaptive diagnosis phase, further restricting its diagnosability. The adaptive diagnosis phase of STAR technique [2] uses the PLBs surrounding the faulty BISTer area to diagnose it. The problems with this method are, first, the surrounding PLBs (forming the test circuitry) may themselves be faulty, and second, it requires different ROTE shapes for diagnosing PLBs of even a single BISTer tile, making it very complex to implement. In HD-BISTer the suspect PLBs of a testee stick are diagnosed by pairing this testee stick with a fault-free stick found in the fault detection phase. Hence the test circuitry is already tested for fault-freeness and available in the current ROTE area. Note that according to our (realistic) assumption 2 there will be at least one fault-free stick in the ROTE area which can act as a fault-free tester stick.

Next we discuss each of the three phases of HD-BISTer in greater detail. For simplicity of exposition, the bootstrapping phase is explained after the fault detection phase and adaptive diagnosis phases. At this point we assume that the bootstrapping phase is completed and we have determined which sticks have a fault-free ORA and which sticks do not.

2.2 Fault Detection and Gross Diagnosis Phase

In the fault detection phase, we test the PLBs of each stick by comparing each PLB with the other two PLBs of the stick. E.g., for a testee stick with PLBs A, B and C, the three sets of CUT pairs corresponding to three different sessions of testing will be AB, BC and CA. As shown in Fig. 1c and Table 1, each CUT pair is tested for the operational functions of only one of the CUT PLBs. E.g., when A and B are CUTs, A and B are configured only for the two operational functions of A (A and $A+3$) and not for the operational functions of B. The outputs of the CUTs are compared by the fault-free ORA (found in the bootstrapping phase) of the tester stick. Note that the fault detection phase cannot be performed for a testee stick X if there is no fault-free ORA (determined in bootstrapping phase) in the corresponding tester stick Y. In this case, all the three PLBs of stick X are designated as suspect PLBs and are further tested only in the adaptive diagnosis phase. After three sessions, the functions of testee and tester are swapped among the two sticks of the BISTer tile. Figure 1c summarizes the testing sessions for a testee stick.

Table 1: *Gross syndromes of the* 3 *sessions of testing a stick.*

Ses.	*CUTs*	*Oper. Funcs.*	*Gross Syndrome*			
			Case 1	*Case 2*	*Case 3*	*Case 4*
1	AB	$A, A+3$	$\surd$	X	X	X
2	BC	$B, B+3$	$\surd$	$\surd$	X	X
3	CA	$C, C+3$	$\surd$	$\surd$	$\surd$	X

Case 1: All three sessions pass.
Inference: All the three PLBs A, B and C are f-fault free, as by assumption 1, all the three PLBs cannot be identically faulty.
Case 2: One session fails and two sessions pass.
Inference: Either of the following two possibilities can occur:
1: A is f-faulty or B is nf-faulty, as either A or B failed for the operational functions of A (when the CUT pair was AB) and not for the operational functions of B (for CUT pair BC).
2: All the three PLBs are f-faulty but faults in C mask faults in A and B in sessions 2 and 3, respectively. Figure 2b exemplifies this scenario. The probability of this scenario occuring is

$$P_{cs2} = C_2^3 \cdot p^2 \cdot P_{same}^2 \cdot (1 - P_{same}) \cdot P_{unaff}^2 \cdot (1 - P_{unaff}) \quad (1)$$

Here C_i^n indicates "n choose i", p is the probability of a PLB being faulty, P_{same} is the probability of having a fault in another PLB in the same stick which results in fault masking, and P_{unaff} is the probability that the fault that is not masked does not affect the operational function of the PLB under test. For a fault density of 10% ($p = 0.1$) P_{cs2} is 1.5×10^{-8}. Thus the probability of inference 2 is extremely low, and we neglect this scenario in further diagnosis.
Case 3: Two sessions fail and one session passes.
Inference: One of the following three possibilities can occur:
1: B is f-faulty as B is common in the two failing sessions.
2: B is not f-faulty but A and C are f-faulty with identical fault type, and hence when compared in session 3 give a gross syndrome of pass due to fault masking.
3: B is f-faulty, and A and C are f-faulty with identical fault types.
Case 4: All three sessions fail.
Inference: In this case all the three PLBs are treated as suspect PLBs.

2.3 Adaptive Fault Diagnosis Phase

In this phase, we determine which PLBs are faulty among the set of suspect PLBs. The following steps are performed:
Step 1: BISTer tiles are reconfigured to include one f-fault-free stick (i.e., a stick corresponding to case 1 of the fault detection phase) as the tester and one faulty stick (i.e., a stick corresponding to gross syndrome cases 2, 3 and 4, see Table 1) as the testee in each tile (Fig. 3).
Step 2: Each suspect PLB (X in Fig. 3) is diagnosed by comparing it with a PLB of the f-fault-free stick (e.g., PLB D or E in Fig. 3c). The PLB in the f-fault-free stick that passed the ORA test initially (in the bootstrapping phase to be discussed shortly) will act as the ORA in the adaptive fault diagnosis phase too. Before we compare a suspect PLB, say PLB A, with either D or E, we need to make sure that the PLB D (or E) is f-fault-free for the operational functions of the faulty PLB A. For this purpose, the three PLBs of the f-fault-free stick are tested (for the two operational functions of A) by comparing these PLBs with each other and this way

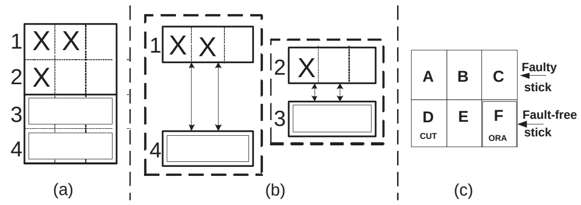


Figure 3: *(a) The results of the fault detection phase. White sticks are f-fault-free sticks and X represents a suspect PLB. (b) The reconfiguration of BISTer tiles to form stick pairs, each with one f-fault-free and one faulty stick for the adaptive fault diagnosis phase. (c) Such a reconfigured BISTer tile with one f-fault-free stick (tester) and one faulty stick (testee).*

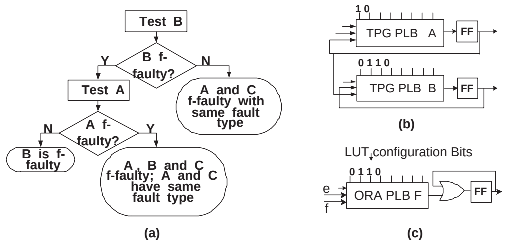


Figure 4: *The adaptive fault diagnosis phase. (a) The optimally-ordered test-and-diagnosis flowchart for case* 3 *of the fault detection phase. Schematics of (b) TPG and (c) LUT of ORA PLB, for the bootstrapping phase for a single-LUT PLB.*

one PLB in the f-fault-free stick, that is f-fault-free for the two operational functions of A, is found[3].

Step 2 is repeated for diagnosing each of the suspect PLBs in the faulty stick.

Based on faulty cases 2, 3 and 4 of the fault detection phase (Sec. 2.2), the adaptive diagnosis phase, if performed in a particular order, does not require testing all the suspect PLBs and so the diagnosis can be completed in fewer total sessions. E.g., Fig. 4a shows the optimal order for diagnosing case 3 faults. As shown, testing of suspect PLB B is done prior to testing of suspect PLB A. With this order of testing, if suspect PLB B is diagnosed as not f-faulty, then we don't have to test suspect PLB A, as it can be inferred that the only way we can get the gross syndrome corresponding to case 3 with a f-fault-free PLB B is to have PLBs A and C f-faulty. The same inference cannot be made by testing the suspect PLB A before PLB B, and so in that case, we are required to test B irrespective of the fault state of A.

2.4 Bootstrapping Phase

The bootstrapping phase is the first phase of fault testing in the HD-BISTer. The aim of this phase is to find one fault-free ORA, if it exists, in each stick in the ROTE area. The bootstrapping phase, being the first phase of fault testing, does not have any knowledge of the fault state of the PLBs. Also, the PLBs forming a TPG can be faulty and thus skip essential test vectors, giving incorrect results. As a solution to these issues, a two-step iterative bootstrapping process is adapted wherein: (1) we find TPGs having very low probabilities of skipping test vectors, and (2) using these TPGs, we find one fault-free ORA per stick, if it exists.
These steps are discussed in detail in sections 2.4.2 and 2.4.3.

[3]If such a f-fault-free PLB is not present in the f-fault-free stick then we pair the faulty stick with another f-fault-free stick of that ROTE area and so forth. By our assumption 2, there will exist atleast one fault-free stick in the ROTE area that meets our requirement for the suspect PLB in question.

2.4.1 The ORA and TPG for the bootstrapping phase

In current SRAM-based FPGAs (e.g., Xilinx Virtex series), each PLB is formed of multiple *Logic Cells* (LC), each containing a t-input 1-output lookup table (LUT), a carry logic and a storage element. In the bootstrapping phase, the ORA comprises of multiple LCs, each comparing a pair of outputs from the CUTs; see Fig. 4c. The flip-flops of the LCs of the ORA are connected to form a scan chain so that the corresponding gross syndrome bit stored in them can be scanned out for evaluation. Thus to find a fault-free ORA per stick, each LUT of the CUTs is configured for the 2-input XOR function. The TPG (Fig. 4b) thus needs to produce only two bit test vectors in the bootstrapping phase, which are then applied to each LUT (of the LC) in the CUTs. As shown in Fig. 4b, only two LUTs are required for the TPG function in order to test the CUTs for the ORA function. Hence the number of PLBs required for the TPG function in the bootstrapping phase is less than that required in the fault detection phase.

Without loss of generality we consider a single-LUT 3-input 1-output PLB for further analysis, although the HD-BISTer technique is applicable to multiple-LUT PLBs, as can be inferred from the concepts in this paper that are discussed for a general PLB.

2.4.2 TPGs with very low probability of skipping test vectors

These TPGs are found also using the same BISTer configuration of a tester and a testee stick, like in the fault detection phase. A combination of two techniques are used: 1. TPG shuffling, and 2. Input-output fault-free stick detection.

1. TPG shuffling: Instead of testing CUTs with a single TPG we use three different sets or combinations of the three PLBs forming a TPG; e.g., if the PLBs available in a BISTer for the TPG function are A, B and C and if (say) two PLBs are enough to form the TPG in the bootstrapping phase, then the three 2-PLB TPG configurations will be AB, BC and CA and they are all used sequentially to generate the same set of test vectors. Note that the positions (*place value*) of the PLBs in each of the three TPG configurations are also different. We call these three combinations of 2-PLB TPGs as a *shuffled TPG*.

Theorem 1: The probability of the shuffled TPG skipping test vectors is lower by a factor of at least p (the probability of a PLB being faulty) compared to that of the normal TPG, for FPGAs with PLB input:output pin ratio (I:O ratio) of 3:1 (this ratio is typical of current commercial FPGAs).

Proof: Let x and y, respectively, be the fraction of total locations in PLBs A and B (forming a TPG) where the presence of a fault will lead to skipping of the test vectors[4].

With only one combination of PLBs forming a TPG, the fraction of test vectors skipped due to a fault in one and two PLBs of the TPG will be 50% and 75%, respectively[5]. If p is the probability of a PLB being faulty then the probability

Table 2: *Outputs of CUTs, configured for the XOR function for test vectors of TVS under different fault conditions.*

	Output of CUT conf. for 2-i/p XOR under different fault conditions							
1	2	3	4	5	6	7	8	9
inp. vec.	*fault free*	ip_1 *s-a-*X	ip_1 *s-a-*X'	ip_2 *s-a-*X	ip_2 *s-a-*X'	*faulty LUT bit corr. to vec.*		
						XX	$X'X$	XX'
XX	0	0	1	0	1	1	0	0
$X'X$	1	0	1	1	0	1	0	1
XX'	1	1	0	0	1	1	1	0

P_s of the TPG skipping a test vector will be

$$P_s = 0.5 \cdot p \cdot (1-p) \cdot (x+y) + 0.75 \cdot p^2 \cdot x \cdot y \qquad (2)$$

For a 2-PLB shuffled TPG with PLBs A, B and C the fraction of test vector skipped due to one, two and three PLBs among A, B and C being faulty will be upper bounded by 0%, 25% and 50%, respectively. Hence the probability P_{3s} of each of the three 2-PLB TPGs (i.e., the shuffled TPG) skipping a test vector will be

$$P_{3s} = 0.25 \cdot C_2^3 \cdot p^2 \cdot (1-p) \cdot x \cdot y + 0.5 \cdot p^3 \cdot x \cdot y \qquad (3)$$

For a simple single-LUT 3-input PLB (I:O ratio is 3:1), $x = 4/12 = 0.33$ and $y = 7/12 = 0.58$, giving

$$P_s = 0.455 \cdot p \cdot (1-p) + 0.14 \cdot p^2$$
$$P_{3s}/p = 0.14 \cdot p \cdot (1-p) + 0.09 \cdot p^2$$

The first and second terms of P_s are greater than the respective first and second terms of P_{3s}/p in the above equations and hence $P_s > P_{3s}/p$. As the PLB size increases, x and y decrease, with product $x.y$ in the first term of P_{3s} decreasing more rapidly than the sum $(x+y)$ in the first term of P_s. Hence the inequality $P_s > P_{3s}/p$ will hold true for any size PLB with an I:O ratio of 3:1. E.g., with $p = 0.1$, for the above simple PLB, $P_s = 0.044$ and $P_{3s} = 0.0004$. $\diamondsuit$

Under the worst-case fault scenario where two PLBs' (say, A and B) outputs are s-a-X and the third PLB's (C) output is s-a-X', the shuffled TPG (TPG combinations AB, BC and CA) will still produce a three test vector set (TVS) = $\{XX, XX', X'X\}$ out of the four possible 2-bit test vectors.

2. Input-output (I/O) fault-free stick detection: Even though our above analysis assumes the worst-case situation of faults on the output pins of the TPG PLBs, we would like to reduce the skipping probability below those given by Eqns. 2 and 3 by identifying sticks with PLBs with no I/O pin faults and use them as TPGs. This is because the maximum number of test vectors are skipped by a TPG if the faults are on the input and output pins of the TPG PLBs.

Testing technique for I/O fault-free stick detection:

(1.a): Bisters are configured with the tester stick providing the shuffled TPG. Each PLB in the testee stick is configured for the 2-input XOR function (Fig. 4c) and is individually tested by passing test vector XX (from a single output of a TPG PLB) to its inputs.

(1.b): With the above configuration, each PLB is tested for the 2-input XOR function by passing test vectors from the shuffled TPG to its inputs.

(2): The PLBs of the testee stick are tested as CUTs, configured for the 2-input XOR function, in three sessions just like in the fault detection phase. The shuffled TPG is provided by the tester stick while the ORA is provided by the testee.

[4]E.g., considering PLB's inputs, output and LUT bits as possible fault locations, for TPG A in Fig. 4b, $x = (q + q + 2^q)/(3q + q + 2^{3q})$ while for TPG B, $y = (2q + q + 2^{2q})/(3q + q + 2^{3q})$ for a $3q$-input q-output PLB.

[5]We assume one fault per faulty PLB and consider the worst-case scenario wherein the fault is on the location where it affects the maximum loss of test vectors, e.g., on an output pin.

Theorem 2: Under assumption 1 *and with the assumption that there is exactly one fault in a faulty PLB, the testing technique given above will detect a stuck-at fault present on the inputs or the output of the PLBs in the testee stick, irrespective of the number of faulty PLBs in the tester and testee stick of the HD-BISTer tile.*
Proof Outline:

Part 1: Detecting a fault on the output of a PLB: In step (1.a) of the testing technique, a s-a-1 fault on the output of a PLB will produce GS=1 for test vector XX as opposed to the fault-free GS of 0, and hence this fault will be detected. In step (1.b), under the fault-free condition, the PLB configured for the 2-input XOR function will have GS=1 because of the application of vectors XX' and $X'X$ from the shuffled TPG, whereas with the output of the PLB s-a-0, GS=0, and hence this fault will also be detected.

Part 2: Detecting a fault on the inputs of a PLB: The detection of the faults on the inputs of the CUTs, by step 2 of the testing technique, can be proved from the following two properties:

Property 1: CUTs with unidentical input pin fault states are able to produce outputs 01 and 10 for the test vectors of TVS (see Table 2). For these CUT outputs, GS=1 (thus detecting input pin fault(s)) irrespective of the faulty state of the ORA PLB. This is because if the ORA has an input s-a-X then there exists a CUT output $XX'\in\{01, 10\}$ which results in GS=1. If the ORA has a fault on one of its LUT bits, then this fault can mask at most one of the two CUT outputs 01 or 10 and hence for the other CUT output GS=1.

Property 2: If two PLBs (say D and E) in the testee stick have identical faults and the third PLB (say F) has a s-a-X fault on one of its inputs, then the session in which D and E are CUTs will produce outputs 00 and 11, and hence for the faulty ORA F with an input s-a-X there exists a CUT output $X'X'\in\{00, 11\}$ which results in GS=1. If the ORA F is fault-free then the next session in which PLBs E and F are CUTs will result in the situation described in Property 1.
Using Table 2 it can be shown that under the given assumptions any fault combinations on the inputs and LUT bits of the PLBs in the testee stick will exhibit one of the above two properties and hence will produce GS=1, thus detecting faults on input pins of the testee PLBs.$\diamondsuit$

Thus from Theorem 2 we can conclude that if all the three sessions corresponding to testing of the testee stick pass then there are no faults on the input and output pins of all the three PLBs of the testee stick. We refer to input-output fault-free sticks as *IO-FF sticks*. Due to the reasons given earlier we use these IO-FF sticks as tester sticks providing the TPG in the next subphase of the bootstrapping phase where we will find one fault-free ORA per stick, if it exists.

2.4.3 Detecting one fault-free ORA per stick

1. Detecting a fault-free ORA in the IO-FF sticks: Configure BISTer tiles such that each BISTer includes a pair of IO-FF sticks. One IO-FF stick acts as a tester stick providing the shuffled TPG while the other IO-FF stick acts as a testee stick, providing ORA and CUTs, configured for the 2-input XOR function. Like in the fault detection phase, the CUTs are tested in three sessions, after which the functions of tester and testee are swapped among the two IO-FF sticks of a BISTer. Also, like in the fault detection phase, we determine fault-free sticks, i.e., sticks with GS=0 for all the three sessions of testing. After detection of the suspect PLBs, these PLBs are diagnosed for fault-freeness using the concepts of the adaptive diagnosis phase of Section 2.3.

2. Detecting a fault-free ORA in the non IO-FF sticks: Configure BISTer tiles such that each BISTer includes one IO-FF stick and one non IO-FF stick. The IO-FF stick, acting as a tester, provides (a part of or complete) shuffled TPG, a fault-free ORA and a fault-free CUT, detected in step 1 above. All the three PLBs in the testee (non IO-FF) stick are directly considered as suspect PLBs and diagnosed using the IO-FF stick For a 2-PLB shuffled TPG, one TPG PLB is obtained from the IO-FF stick while the other TPG PLB can be obtained from either the non IO-FF stick or from another IO-FF stick.

2.5 Probability of Correct Diagnosis

We analyze the probability that the three phases of HD-BISTer perform correct diagnosis. In the bootstrapping phase, the probability of the shuffled TPG skipping a test vector is upper-bounded by Eqn. 3. Also, we discussed that the faults at the input and output locations of the PLB tested for the ORA function will be detected. Hence the only case of incorrect fault-free ORA determination occurs when the TPG skips the test vector corresponding to a fault in one of the LUT bits of the ORA. The fraction of the LUT bits used by an ORA among the total number of (fault) locations is $z = 2^{2q}/(2^{3q} + 4q)$ for a $3q$-input q-output PLB. Hence the probability P_o of incorrect ORA diagnosis is

$$P_o = p \cdot P_{3s} \cdot z$$

In the fault detection phase, the probability of incorrect fault-free diagnosis of any PLB A will be

$$P_{fd} = p\cdot[P_{sk}+P_{same}^2-(P_{sk}\cdot P_{same}^2)+P_o-(P_o\cdot P_{sk\,\cup\,same})] \quad (4)$$

where P_{sk} is the probability of skipping all the test vector(s) corresponding to the fault in PLB A; P_{same} is the probability of having a fault in the same location in another PLB in the same stick thus resulting in fault masking; $P_{sk\,\cup\,same}$ is the probability of the event that is union of the events addressed by P_{sk} and P_{same}. With $(1/Y) = 2^{3q} + 4q$ possible fault locations, $P_{same} = p \cdot Y$.

Let α be the average number of essential test vectors per fault in a PLB. Then $P_{sk} = {P_{st}}^{\alpha}$; where P_{st} is the probability of a 3-PLB TPG skipping test vector(s) and it can be derived in a similar way to P_{3s} (Eqn. 3) which is the skipping probability for a 2-PLB TPG with three shuffled combinations.

Also, in the fault detection phase, the probability of incorrect diagnosis of an f-fault-free stick given a fault-free ORA will be

$$P_{fdff} = {P_1}' + {P_2}' + {P_3}'$$

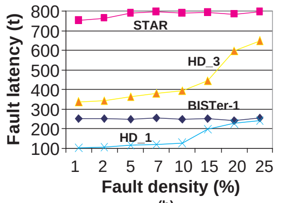
where ${P_i}'$ indicates the probability that i out of 3 PLBs in the stick is/are faulty but undetected. Hence the probability of correct diagnosis of the f-fault-free stick will be

$$P_{fdc} = (1 - P_{fdff})(1 - P_o) \quad (5)$$

In the adaptive diagnosis phase, the probability of incorrect diagnosis of the suspect PLB A being f-fault-free, when compared with PLB D of f-fault-free stick, will be
$P_{ad} =$ *p.[The probability of (skipping the test vector corresponding to the fault in A) or (incorrect diagnosis of PLB D for the op. functions of A, and A and D have the same*

p	P^{av}_{corr} (Eqn. 7)	Lower Bound of P^{av}_{corr} (Eqn. 8)
0.05	0.9915	0.9908
0.1	0.9783	0.9768
0.15	0.9693	0.9673
0.2	0.9626	0.9611
0.25	0.9549	0.9546

(a)



(b)

Figure 5: *(a) Results showing probabilities of HD-BISTer performing correct diagnosis under different fault densities. (b) Fault latency (in terms of the number of sessions) results for randomly distributed faults. For STAR BISTer, the PLBs were tested pseudoexhaustively for 16 out of 64 possible functions.*

faults) or (incorrect diagnosis of the ORA)]

$$P_{ad} = p \cdot [P_{sk} + P_{fd} \cdot Y - (P_{sk} \cdot P_{fd} \cdot Y) + P_o - (P_o \cdot P_{sk \cup fdy})] \quad (6)$$

Let S be the average number of suspect PLBs in a faulty stick. Then the probability of catching all faulty PLBs in an $n \times 3$ ROTE area, P^{av}_{corr}, will be as follows.

Let $P_{corr}(j)$ be the probability of correct diagnosis with j sticks diagnosed as fault-free.

$P_{corr}(j)$ = *probabibility that (the j fault-free sticks are correctly diagnosed) and (the average of S suspect PLBs in each faulty stick is correctly diagnosed in the adaptive diagnosis phase) and (on the average 3-S fault-free PLB in every faulty stick were correctly diagnosed in the fault detection phase).*

$$P_{corr}(j) = {P_{fdc}}^{j} \cdot (1 - P_{ad})^{S(n-j)} \cdot (1 - P_{fd})^{(3-S)(n-j)}$$

If $P_{ff}(j)$ is the probability of having j fault-free sticks in ROTE area then

$$P_{jff}(j) = C^{n}_{j} \cdot (1-p)^{3j} \cdot [1 - (1-p)^{3}]^{n-j}$$

$$P^{av}_{corr} = \sum_{j=1}^{n} P_{corr}(j) \cdot P_{ff}(j) \quad (7)$$

Let ff be the average number of fault-free sticks found in the fault detection phase then P^{av}_{corr} can be lower bounded by the following expression[6].

$$P^{av}_{corr} \geq {P_{fdc}}^{ff} \cdot (1 - P_{ad})^{S(n\text{-}\,ff)} \cdot (1 - P_{fd})^{(3\text{-}S)(n\text{-}ff)} \quad (8)$$

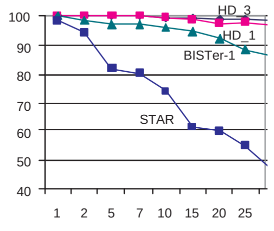
Figure 5a (obtained for $\alpha = 1.5$ with TPG shuffling) shows that the probability of HD-BISTer performing correct diagnosis is very high. These analytical results are borne out by the empirical high fault coverage results (correlated to P^{av}_{corr}) of Sec. 3.

3. SIMULATION RESULTS

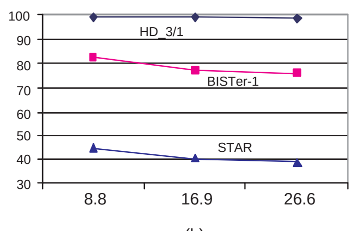
A 32×32 FPGA array with 3-input 1-output PLBs was functionally simulated in C. We implemented BISTER-1 of [1], fault detection and non-adaptive diagnosis phase of STAR BISTer of [2] and HD-BISTer. (Step 1 of the testing technique for I/O fault-free stick detection [Sec. 2.4.2] was not implemented as it increased the probability of correctly detecting I/O fault-free stick by only a very small amount [1.8×10^{-5} for $p = 0.1$]).

HD-BISTer was implemented with TPG shuffling (denoted by HD_3) and without TPG shuffling (HD_1) in the bootstrapping phase (Sec. 2.4). The three BIST techniques were

[6]The average of a function with a random variable in its exponent is lower-bounded by the function with the average of the random variable as the exponent.



(a)



(b)

Figure 6: *Fault coverage (%)(y-axis) vs. Fault density (%)(x-axis) results: (a) For randomly distributed faults. (b) For clustered faults; the three fault density values correspond to cluster densities of 1%, 2% and 3%, resp. HD_1 and HD_3's fault coverage plots overlap and hence are shown by $HD\_3/1$.*

tested for randomly inserted stuck-at faults and also for clustered faults in which faults occur in a cluster around a center fault C with a probability distribution function of $p = k/d$, where d is the distance of a PLB from C and k is a suitable proportionality constant; each cluster is distributed randomly across the FPGA with a certain density. Figure 6 gives the fault coverage results. Our technique is quite stable across fault densities, giving a fault coverage (for HD_3) of 98.7% at 25% fault density for randomly distributed faults, while BISTer-1's and STAR's coverages are 88.5% and 54.5%, resp. For cluster faults with 1% cluster density, the fault coverages for HD_3, BISTer-1 and STAR are 98.9%, 82.5% and 44.5%, resp. Figure 5b plots the fault latencies obtained for the different methods. For low fault densities (below 10%) STAR's and BISTer-1's latencies are respectively six and two times more than that of HD_1's. For higher fault densities HD_1's fault latency approaches that of BISTer-1's. HD_3's fault latencies are slightly higher than BISTer-1's latencies although STAR's latencies are twice that of HD_3's.

4. CONCLUSION

We have presented a new BISTer design HD-BISTer which theoretically as well as empirically has probabistically very high fault diagnosability even for high fault densities ($\geq$10%) thus outperforming previous techniques in terms of fault coverage. Its fault latency is lower or comparable to previous methods. HD-BISTer should thus prove very effective for the testing of emerging VDSM as well as future nanotechnology FPGAs that are expected to have high fault densities.

5. REFERENCES

[1] V. Verma, S. Dutt and V. Suthar, "Efficient On-line Testing of FPGAs with Provable Diagnosabilities", *Proc. DAC*, 2004 pp. 498-503 .

[2] M. Abramovici, C. Stroud, B. Skaggs and J. Emmert, "Improving on-line BIST-based diagnosis for roving STARs", *Proc. 6th IEEE Intl On-line Testing Workshop*, 2000.

[3] N. R. Shnidman et. al, "On-line Fault Detection for Bus-Based Field Programmable Gate Arrays", *IEEE Trans. on VLSI systems*, Vol. 6, No. 4, pp. 656-666, Dec 1998 .

[4] W.K. Huang et. al., "Testing configurable LUT-Based FPGAs", *IEEE Trans. VLSI*, Vol. 6, pp. 276-283

[5] S.J. Wang and T.M. Tai,"Test and Diagnosis of Faulty Logic Blocks in FPGAs", *Proc. ICCAD*, 1987

[6] W.K. Huang and F. Lombardi, "An approach to testing of Programmable/Configurable FPGAs", *Proc. VTS*, 1996

[7] A. Steininger et. al, "On the Necessity of On-Line BIST in Safety-Critical Applications", *Proc. 29th FTCS.*, 1999.

[8] A. Burress and P. Lala, "On-Line Testable Logic Design for FPGA Implementation", *Proc. Int'l Test Conf.*, 1997.