Mixed PLB and Interconnect BIST for FPGAs Without Fault-Free Assumptions [*]

Vishal Suthar and Shantanu Dutt
Department of Electrical and Computer Engineering, Univ. of Illinois at Chicago
{vsuthar, dutt}@ece.uic.edu

Abstract **We tackle the problem of fault-free assumptions in current PLB and interconnect built-in-self-test (BIST) techniques for FPGAs. These assumptions were made in order to develop strong BIST methods for one class of components (PLBs or interconnects) while assuming that the other class is fault-free. This results in a cyclical conundrum that renders current PLB and interconnect BIST techniques impractical, since current deep-submicron FPGAs as well as those of emerging single-digit nanometer technologies are expected to have a profusion of hard (permanent) PLB as well as interconnect faults. We address this issue here and develop a novel method M-BIST that uses a combination of (i)** ***iterative bootstrapping*** **that without any knowledge of the state of any PLB or interconnect determines a minimum contingent of fault-free test circuit components with high probability, and (ii) mixed testing of PLBs and interconnects in an** *interleaved* **manner that identifies fault-free components that are used in subsequent testing phases until the entire FPGA is tested. This approach is overlaid on current state-of-the-art PLB and interconnect BIST techniques. Simulation results obtained for faults present in both PLBs and interconnects show significant improvements in both fault coverage and false positives yielded by M-BIST compared to the PLB-only and interconnect-only BIST techniques used within the M-BIST wrapper that make fault-free assumptions about the other component type.**

1 Introduction

Current technology trends for FPGA devices are in the very deep-submicron (VDSM) regime with recent chips using 90 nanometers and seven metal layers. Unfortunately, this trend has resulted in decrease of fabrication yield as well as decreased reliability in operation due to increased numbers of permanent faults. With emerging single-digit nanometer technology these yield and reliability concerns will be further exacerbated. Thus testing and diagnosis followed by fault-reconfiguration using the re-programmability feature of FPGAs are necessary for providing adequate reliability in FPGA-based systems. This paper addresses critical issues in test and diagnosis of FPGAs for *hard* or *permanent* faults using the *Built-in self-test* (BIST) approach. In BIST, a test circuit comprising of a *test pattern generator (TPG)* and an *output response analyzer (ORA)* are configured within the FPGA to test the remainder of the PLBs and interconnects; see Fig. 1c. A *gross syndrome (GS)* of a *session* of comparison-based testing of 2 components is the overall *pass/fail* (indicated as"X"/"$\surd$") observation over all test vectors for that session. In other words, the GS of a session is a "X" (*fail*) if the ORA output is a fail for any input test vector and is a "$\surd$" (*pass*), otherwise.

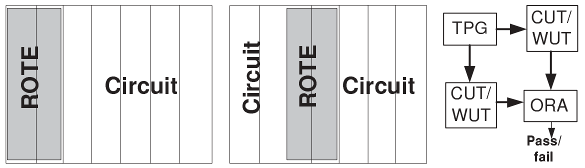


Figure 1: *Online testing concept: (a) Only a part of the FPGA (say, 2 columns) is under test using a roving tester (ROTE), while the remaining part of the FPGA performs the circuit function. Once testing of this part is completed, the ROTE is reconfigured to test another part of the FPGA, as shown in (b). (c) A Built-in self-tester (BISTer) tile; CUT is a "cell/PLB under test" and WUT stands for "wire(s)-under-test".*

[*]This work was supported in part by NSF grant CCR-0204097.

FPGAs can be tested either *off-line* or *on-line*. In off-line testing, the FPGA is tested while it is not executing any system operation [2, 3, 4, 5, 11], while in on-line testing the FPGA is tested with the application circuit mapped to and executing on it [1, 8, 9, 10, 16]; see Fig. 1a-b. In applications such as life critical systems, space hardwares and remote installations, where the circuit mapped on the FPGA needs to run continuously with minimal or no interruption, on-line testing is crucial [9]. While the enveloping techniques we present in this paper are for on-line testing, the basic BIST methods we develop can be readily adapted to off-line testing.

Although BIST in FPGAs has matured over the last decade [1, 6, 7, 8, 10, 11, 12, 13, 16], there are still some major practical and theoretical issues that need to be addressed. A key issue is that existing BIST techniques, for both PLB (programmable logic block) and interconnect testing, unrealistically assume that while testing PLBs, interconnects used in the BISTer are fault-free, and, similarly, while testing interconnects, the PLBs forming the test circuit are fault-free. In this paper, we present for this first time techniques for testing PLBs as well as interconnects in a manner that does not assume that either are fault-free. The technique uses a combination of *iterative bootstrapping*, and testing of PLBs and interconnects in a *mixed-and-interleaved* manner. We use state-of-the-art PLB BIST [15] and interconnect BIST [14] techniques as the basic elements on which our above *mixed BIST* approach is overlaid. Simulation results show the fallacy of any fault-free assumptions of any component, and the significant improvement in fault coverage and reduction in false positives yielded by our techniques.

In Sec. 2, we discuss the concept of iterative bootstrapping. Section 3 briefly discusses the basic PLB [15] and interconnect [14] BIST methods over which our technique is overlaid. Section 4 then discusses the methodology for testing both PLBs and interconnects sans fault-free assumptions that we term *M-BIST*. We discuss simulation results in Sec. 5 and conclude in Sec. 6.

2 Iterative Bootstrapping

Iterative bootstrapping is a mechanism we have developed to identify fault-free components with sufficient reliability starting with a testing circuit whose fault state is completely unknown. The idea is to iteratively progress from this circuit with a fault-free probability of q_1, identify probabilistically fault-free components which can form a similar test circuit with a fault-free probability of $q_2 > q_1$, and so forth until we identify fault-free components with a high enough probability such that testing circuits formed by them have a fault-free probability of $q_i > q_{i-1}$, and $q_i \approx 1$. These fault-free testing circuits are then used to test and diagnose the rest of the FPGA.

Figure 2a gives the flowchart of a generic iterative bootstrapping approach wherein redundancy like triple-modular redundancy (TMR) is used to determine fault-tolerant test circuits (e.g, comprising TPGs and ORAs) using components (PLBs and interconnects) with unknown fault states.

TMR-based Bootstrapping: Figure 2b shows a TMR-based redun-

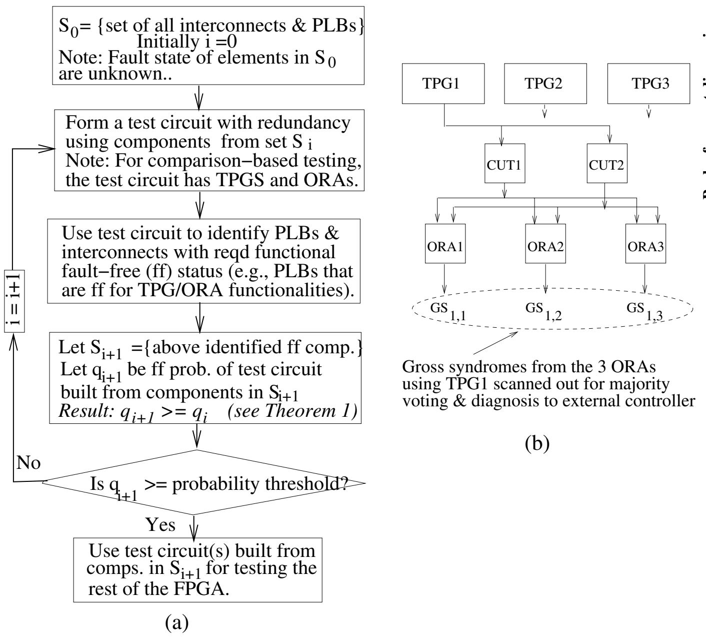

S0= {set of all interconnects & PLBs}
Initially i =0
Note: Fault state of elements in S 0 are unknown..
Form a test circuit with redundancy using components from set S i
Note: For comparison−based testing, the test circuit has TPGS and ORAs.
Use test circuit to identify PLBs & interconnects with reqd functional fault−free (ff) status (e.g., PLBs that are ff for TPG/ORA functionalities).
Let Si+1 ={above identified ff comp.}
Let qi+1 be ff prob. of test circuit built from components in Si+1
Result: qi+1 >= qi (see Theorem 1)
Is qi+1 >= probability threshold?
No
Yes
i = i+1
Use test circuit(s) built from comps. in Si+1 for testing the rest of the FPGA.
(a)
TPG1
TPG2
TPG3
CUT1
CUT2
ORA1
ORA2
ORA3
GS1,1
GS1,2
GS1,3
Gross syndromes from the 3 ORAs using TPG1 scanned out for majority voting & diagnosis to external controller
(b)

Figure 2: *(a) Flowchart of a generic bootstrapping technique. (b) One testing sub-session of a TMR-based bootstrapping approach. Three such sub-sessions, each corresponding to a different TPG producing the test vectors, constitute a TMR-session.*

dant test circuit formed of TPGs and ORAs. The GSs shown in the figure correspond to TPG1 generating the test vectors (TVs). The TMR'ed TPGs reduce the probability of skipping test vectors and the TMR'ed ORAs reduce the probability of incorrect comparison by a faulty ORA.

Let p be the probability of a PLB being faulty. Assuming for simplicity of exposition that the required TPG and ORA can each be realized on a single PLB, we get the following expression for the probability q_1 of correct diagnosis of a CUT or WUT using a TMR version of the basic BIST shown in Fig. 1b:

$$q_1 = [(1-p)^3 + 3p\cdot(1-p)^2]\cdot[(1-p)^3 + 3p\cdot(1-p)^2]$$

This implies that the probability of the components identified as fault-free by the TMR BISTer is q_1—this actually is a worst-case analysis since it assumes that the probability that a fault in the TPG PLB causes it to skip an essential TV for fault(s) in the CUT/WUT is 1 and similarly that the probability that a fault in the ORA PLB affects the ORA function is 1. If the newly identified fault-free PLBs are used as TPGs and ORAs in a new TMR BISTer, the probability q_2 of correct diagnosis becomes (note that $(1-q_1)$ now is the probability of a test circuit PLB being faulty):

$$q_2 = [q_1^3 + 3(1-q_1)\cdot q_1^2]\cdot[q_1^3 + 3(1-q_1)\cdot q_1^2]$$

and so forth.

Figure 3a shows how iterative bootstrapping helps in increasing the probability of identifying fault-free components significantly with only three iterations, starting from a completely unknown state.

Note from the plot that if $p > 0.2$, then using TMR can actually reduce the probability of correct diagnosis compared to the probability $(1-p)$ of correct diagnosis by randomly picking a component and identifying it as fault-free with probability $(1-p)$. In such a case, more reliable BISTers like a PMR (or 5-MR) configuration may be needed which extends that cutoff point beyond which "random pick" is better to around $p = 0.3$; see Fig. 3b. In general, however, TMR should suffice as it is expected that future nano-tech devices will have a fault density

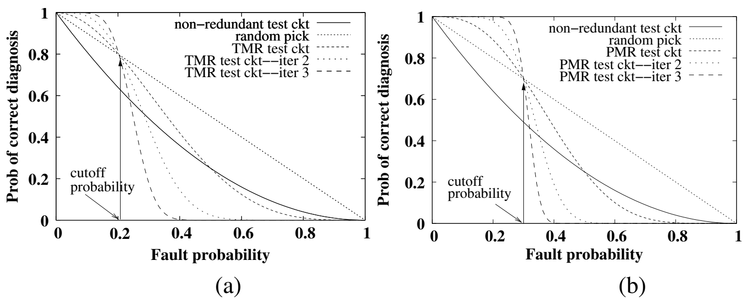


Figure 3: *Increasing probabilities of correct diagnosis using iterative bootstrapping using (a) a TMR BISTer and (b) a PMR (i.e., 5-MR) BISTer.*

(% of faulty components) around 10% ($p = 0.1$ for random faults). We have the following result that establishes the viability (and efficacy) of iterative bootstrapping.

Theorem 1 *Assume that for any redundant test circuit $\mathcal{T}$, the probability of correct diagnosis is a monotonically non-decreasing function $f(q)$ of the fault-free probability q of a single component (we know of no test circuit for which this does not hold, and for example, it is true of a TMR BISTer). Then, if $f(1-p) \geq (1-p)$, where p is the fault probability of any component in the FPGA, then every iteration $i \geq 1$ of the iterative bootstrapping method will determine components with fault-free probabilities $q_i \geq q_{i-1}$ ($q_0 = 1-p$) that form a monotonically non-decreasing sequence $q_0, q_1, \ldots q_i, \ldots, q_k$.*

Proof: At the beginning of iterative bootstrapping, $q = q_0 = 1-p$. The fault-free probability of a component identified as such by $\mathcal{T}$ is $f(q)$. Since $f(1-p) \geq (1-p)$ (the fault probability p at which $f(1-p) = p$ is called the *cut-off probability* for $\mathcal{T}$, e.g., ≈ 0.2 in Fig. 3a), after the first iteration, we have $\mathcal{T}$ formed of components with fault-free probability $q_1 = f(q_0) = f(1-p) \geq (1-p) = q_0$. The same situation applies for any iteration, i.e., $q_i \geq q_{i-1}$. $\diamondsuit$

The plots in Fig. 3 also show the probability $(1-p)^2$ (using worst-case analysis as alluded to earlier) of correct diagnosis for a non-redundant BISTer. This simple expression and its plot underscores the fact that it is inadvisable to perform testing using a test circuit of unknown fault state (or assuming without any reliable information that it is fault-free), since randomly picking a component and declaring it to be fault-free with a probability of $1-p$ is more reliable! This also stresses the importance, indeed the necessity, of iterative bootstrapping that is able to overcome this situation (as long as the fault probability p is below the cutoff for the redundant test circuit being used).

Finally, note that as indicated in Fig. 2a, once fault-free components are identified with a high enough probability, they can be used effectively in a non-redundant BISTer. Since multiple BISTers can operate simultaneously in the ROTE area, there can be many more non-redundant BISTers than redundant ones operating in parallel. This allows faster testing of all components after iterative bootstrapping than continuing to use, say, a TMR BISTer.

3 Basic BIST Elements of M-BIST

We briefly describe here two state-of-the-art BIST techniques one for PLBs and another for interconnects on which we overlay the M-BIST method.

3.1 A High-Diagnosability PLB BISter–HD-BIST

Each HD-BISTer consists of 6 PLBs–2 CUTs, 1 ORA and 3 TPG PLBs[1]. Unlike previous methods, the relative positions of TPG, ORA

[1]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 a strict

and CUTs are not fixed and all the PLB functions are not rotated in subsequent sessions.

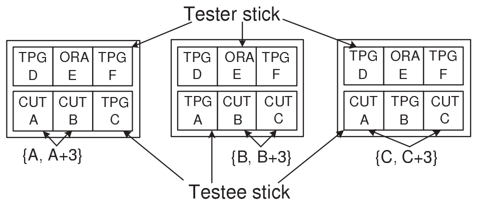
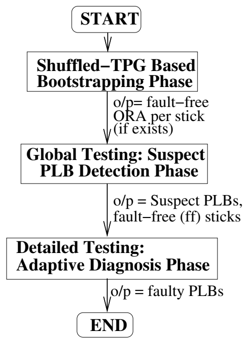


Figure 4: *Three sessions of HD-BISTer for functional testing. The $\{X, X+3\}$ labels correspond to the two functions PLB X will implement as the ROTE moves across the FPGA, and for which it is tested in one of the sessions [15].*

Instead, each HD-BISTer tile is conceptually 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. 4). In each session, a different pair of PLBs in the testee stick act as CUTs, giving a total of three sessions.

The flowchart of HD-BISTer is illustrated in Fig. 5a and consists of three stages: (1) Shuffled-TPG based bootstrapping, which uses a lower-redundancy BISTer than a TMR BISTer; this results in a faster iterative bootstrapping process. Also, its goal is to detect only one fault-free ORA in each stick, and only two iterations are used. It turns out that as a byproduct of this phase, we also identify PLBs with fault-free i/o pins that can be used to construct more relaible TPGs. (2) Global testing–Suspect PLB detection. (3) Detailed testing–Adaptive Diagnosis. We briefly describe the last two stages.

1. Global testing–Suspect PLB detection: In this stage the PLBs of the testee stick are tested as CUTs in three sessions as discussed above. Unlike previous methods, no upper-bound on the number of faulty PLBs are set (this is a probabilistic BISTer as opposed to deterministic ones that can diagnose only a certain [small] number of faults [e.g., 1-2] in a BISTer tile such as [1, 16]), and thus the gross syndromes will not always identify a unique set of faulty PLBs. From the gross syndrome obtained we can, however, determine which PLBs *may be faulty*. These PLBs are termed *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. Note that the CUTs are compared by the fault-free ORA found in the shuffled-TPG bootstrapping phase of the tester stick; if there does not exist a fault-free ORA for the testee stick, then all PLBs in it are designated suspect.



START
Shuffled–TPG Based Bootstrapping Phase
o/p= fault–free ORA per stick (if exists)
Global Testing: Suspect PLB Detection Phase
o/p = Suspect PLBs, fault–free (ff) sticks
Detailed Testing: Adaptive Diagnosis Phase
o/p = faulty PLBs
END

Figure 5: *Flowchart showing three phases of a high-diagnosability BISTer (HD-BISTer).*

2. Detailed testing–Adaptive diagnosis: In this phase, the PLBs in the suspect set that are actually faulty are identified. The following steps are performed:
(1) BISTer tiles are reconfigured to include one fault-free stick as the tester and one suspect stick (i.e., stick with suspect PLBs) as the testee in each tile (Fig. 6).
(2) Each suspect PLB (X in Fig. 6) is diagnosed by comparing it with a PLB of the fault-free stick (e.g., PLB D or E in Fig. 6c). The PLB in the fault-free stick that passed the ORA test in the shuffled-TPG bootstrapping phase will be the ORA in the adaptive diagnosis phase as well.

requirement. In accordance with this, a minimum of 3 and 1 PLBs are required for the TPG and ORA, respectively.

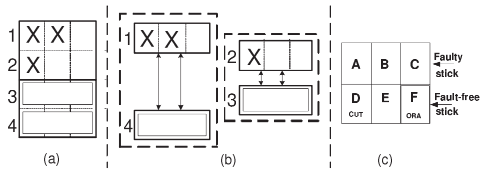


Figure 6: *(a) The results of the suspect detection phase. White sticks are fault-free sticks and X represents a suspect PLB. (b) The reconfiguration of BISTer tiles to form stick pairs, each with one fault-free and one faulty stick for the adaptive fault diagnosis phase. (c) Such a reconfigured BISTer tile with one fault-free stick (tester) and one faulty stick (testee).*

As established in [15], HD-BISTer 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 8.8% fault density, the fault coverages for HD_3, BISTer-1 [16] and STAR [1] are 98.9%, 82.5% and 44.5%, resp. Thus HD-BIST is among the best PLB BIST techniques.

3.2 Interconnect BIST

Interconnect testing involves testing and diagnosing faults in the routing area of the FPGA constituted of wire-segments and switch-boxes. The types of faults we consider are: *wire stuck-at 0/1 fault*, *wire open fault*, *bridge fault* and *switch stuck-open/closed fault*. The types of bridge faults considered are *wired-OR* and *wired-AND*.

We discuss here the I-BIST technique [14] for interconnect testing, which also represents the current state-of-the-art. I-BIST is performed in two phases: (1) *Global testing*: Here multiple *nets*, each formed of multiple interconnected wire-segments and the switches connecting them, span across the entire ROTE area (see Figs 1a-b) and are configured in different patterns and tested for the presence of faults; Fig. 7. (2) *Detailed testing*: Nets that were detected to have fault(s) in the global testing phase are further diagnosed to identify the faulty wire-segments/switches in a binary search process. We now discuss the global testing phase in detail.

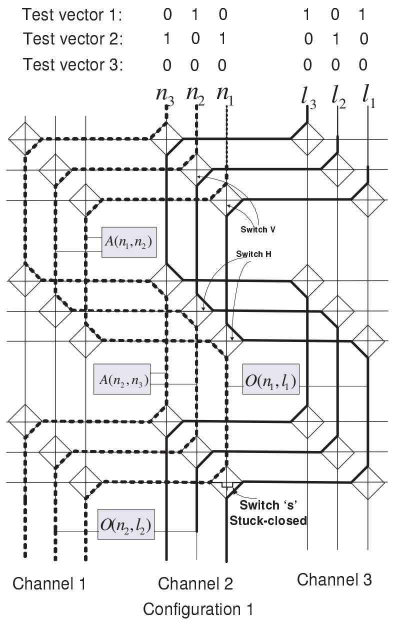


Figure 7: *Configuration 1 of the interconnect global testing phase.*

Global Testing The aim of the global testing phase is to isolate the possible fault locations to a very small number of interconnects, hence making the task of the detailed testing phase simpler. Also this needs to be achieved in a minimum number of configurations[2], since the number of configurations of the FPGA needed to complete testing dominates the testing time. I-BIST achieves this in only five configurations and detects any number of faults on any number of wire-segments and switches. In contrast, the previous minimum

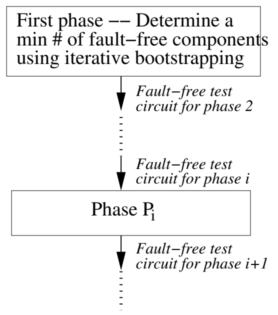
[2]A *configuration* in this context is a pattern of track-segment interconnections realized simultaneously in the ROTE area.

number of configurations reported was 10 [6], and that too without the multiple-fault detectability results of I-BIST.

For brevity, we explain the rationale behind the design of one the 5 configurations for global testing and discuss the interconnect faults that it addresses. The basic testing framework is called the *dual-type multiple-ORA* technique, and is used in the five configurations. Configuration 1, shown in Fig. 7, is used as a reference to explain this technique. As shown in Fig. 7, multiple nets are configured with each net having multiple wire-segments. We call a set of similarly configured nets as a *net-set*. Hence in Fig. 7, the nets shown by dashed lines form one net-set while the nets shown by bold lines form another net-set. In order to detect interconnect faults, two types of net comparisons are performed:

(a) **Intra-net-set comparison:** Each pair of adjacent nets in a net-set is compared by an ORA configured as a 2-bit XOR. E.g., in Fig. 7, A(n_1, n_2) is the ORA comparing nets n_1 and n_2; A(n_2, n_3) is the ORA comparing nets 2 and 3 of the net-set represented by the dashed lines. Similarly, all adjacent pairs of nets of the net-set represented by bold lines are also compared. This comparison is meant to detect all faults other than switch stuck-closed faults. E.g., a switch-open fault or wire-segment fault(s) on net n_2 of the dashed net-set in Fig. 7 will be detected by ORAs A(n_1, n_2) and A(n_2, n_3). Similarly, a bridge fault between nets n_1 and n_2 will be detected by the ORA A(n_1, n_2).

(b) **Inter-net-set comparison:** Nets i of the two net-sets are compared to each other by an ORA configured as a 2-bit XOR. E.g., O(n_1, l_1) represents ORA comparing nets n_1 and l_1 of the two net-sets. This comparison is targeted to detect switch stuck-closed faults among the open switches between the nets of the two net-sets. E.g., a switch stuck-closed fault in any switch H (horizontal switch in a switchbox) or V (vertical switch in a switchbox) (see Fig. 7) of track 1 of channel 2 (e.g., the switch-closed fault in the horizontal switch s shown in Fig. 7) will result in a short between nets n_1 and l_1 of the two net-sets and hence this fault will be detected by the ORA O(n_1,l_1).


First phase –– Determine a min # of fault–free components using iterative bootstrapping
Fault–free test circuit for phase 2
Fault–free test circuit for phase i
Phase P$_i$
Fault–free test circuit for phase i+1
Figure 8: *High-level flow of M-BIST*

The two nets in a WUT pair (in either intra-net or inter-net comparisons) are tested by passing three 2-bit test vectors $\{00, 01, 10\}$ from the TPG; these are enough to test all types of faults in wires and switches listed above across all five configurations [15]. These test vectors are simultaneously applied to all WUT pairs using the same TPG to generate them but of course using distinct ORAs to evaluate the outputs of each WUT pair.

As shown in [14], the above techniques result in I-BIST having provable detectability in the presence of an unbounded number of multiple faults, very high diagnosability of 99-100% even for high fault densities of up to 10% that are expected in emerging nano-scale technologies, and much lower test times or fault latencies than the previous best interconnect BIST methods (e.g., [6, 10, 13]).

4 Mixed BIST Sans Fault-Free Assumptions

In this section we present our mixed PLB and interconnect BIST technique M-BIST that alleviates the problems associated with assuming, erroneously, that when testing PLBs, the interconnects used are fault free and when testing interconnects, the PLBs in the test circuit are fault-free. M-BIST uses iterative bootstrapping (Sec. 2), and overlays its mixed-and-interleaved testing approach on state-of-the-art PLB and interconnect BIST techniques HD-BIST [15] and I-BIST [14] (Sec. 3).

As discussed in Sec. 2, redundant BISTers, while being very reliable, utilize more resources leading to increased test times as there are fewer BISTers that can operate in parallel. It is thus important to use redundancy selectively; we achieve this by using redundant BISTers only in the initial bootstrapping phase of testing and then ordering the remaining phases in such a way that a particular PLB or interconnect phase gets implemented only when (a) the test circuit components required to implement this current phase are already tested for fault-freeness in the previous phase(s), and (b) implementation of the current phase will test the test circuit components that will be required for the subsequent phases. Figure 8 shows a simplified high-level view of our approach. However, this idea cannot be employed if we perform all the phases of PLB testing followed by the phases of interconnect testing, or vice-versa. This is because we require fault-free interconnects to perform PLB testing, which can be obtained only after performing interconnect testing; but for interconnect testing we require fault-free TPGs and ORAs, which can be obtained only after performing PLB testing.

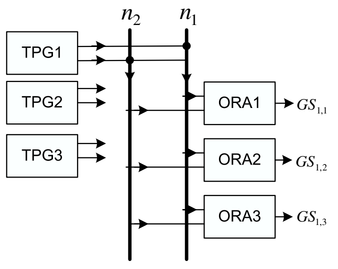
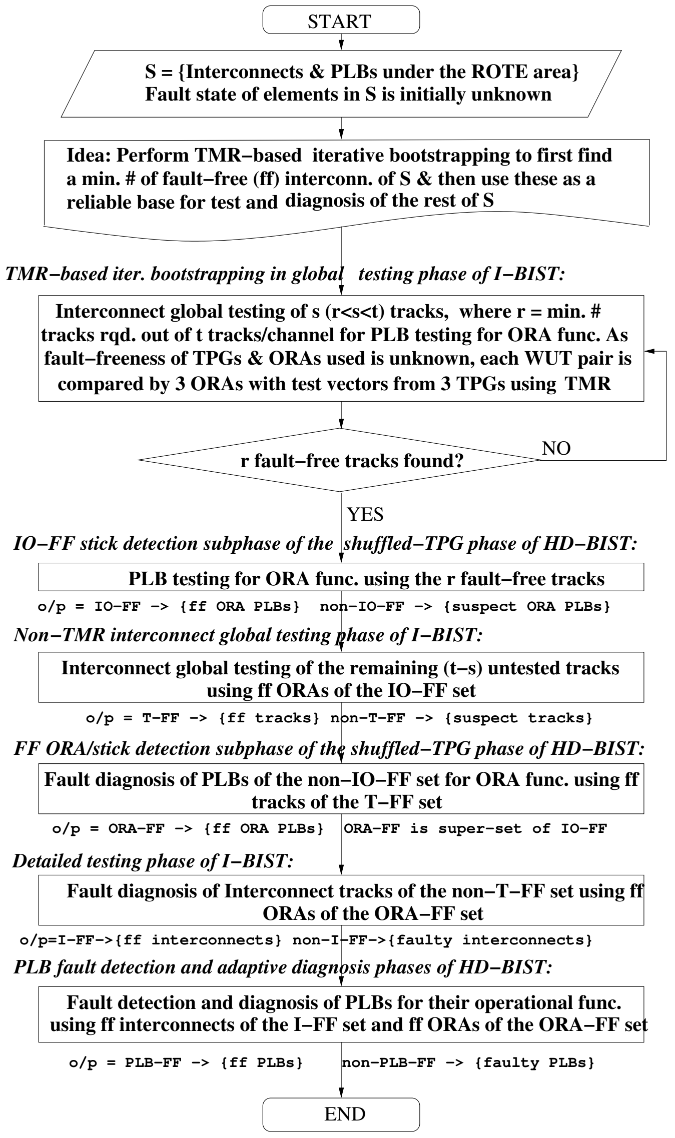


Figure 9: *TMR-based interconnect BIST. Gross syndromes of the three ORAs for test vectors from one of the three TPGs are shown.*

Our approach overcomes this problem by performing *mixed-and-interleaved* PLB and interconnect testing; i.e., we interleave the different phases of PLB testing (HD-BISTer discussed in Sec. 3.1) and interconnect testing (I-BIST discussed in Sec. 3.2). Since the fault-free interconnects required for PLB testing can be obtained from the global testing phase of I-BIST, and similarly the TPGs and ORAs required for interconnect testing are obtained from the shuffled-TPG bootstrapping phase of HD-BIST (see Sec. 3.1), we need to employ the TMR-based iterative bootstrapping (TIB) technique, discussed in Sec. 2, in either the interconnect global testing phase or the shuffled-TPG phase of HD-BIST and then interleave the remaining phases accordingly. In this context, mixed PLB and interconnect testing can be performed in two ways:

1. Considering the shuffled-TPG phase of HD-BIST as the initial phase of mixed testing, we can use the TIB technique in this phase (i.e., use TMR versions of shuffled-TPGs and ORAs) and find fault-free ORAs and TPGs with low probability of skipping test vectors. Using these fault-free ORAs and TPGs as test circuit components we can perform both the interconnect testing phases. Then, using the fault-free interconnects found in interconnect testing and the fault-free ORAs and TPGs found in the shuffled-TPG phase of HD-BIST, we can perform the PLB fault detection and adaptive diagnosis phases.
2. Considering the global testing phase as the initial phase, we can use TIB in this phase to find fault-free tracks; see Fig. 9. Using the interconnects of these fault-free tracks we can perform PLB testing. Then, using the fault-free ORAs and TPGs obtained from PLB testing, we can perform the interconnect detailed testing phase. Note that the interconnect detailed testing phase can be implemented either after the shuffled-TPG phase of HD-BIST or after complete PLB testing.

We determine the amount of test circuit components required for the initial phases of the above two approaches. The shuffled-TPG phase of



START
S = {Interconnects & PLBs under the ROTE area}
Fault state of elements in S is initially unknown
Idea: Perform TMR−based iterative bootstrapping to first find a min. # of fault−free (ff) interconn. of S & then use these as a reliable base for test and diagnosis of the rest of S
TMR−based iter. bootstrapping in global testing phase of I−BIST:
Interconnect global testing of s (r<s<t) tracks, where r = min. # tracks rqd. out of t tracks/channel for PLB testing for ORA func. As fault−freeness of TPGs & ORAs used is unknown, each WUT pair is compared by 3 ORAs with test vectors from 3 TPGs using TMR
r fault−free tracks found?
NO
YES
IO−FF stick detection subphase of the shuffled−TPG phase of HD−BIST:
PLB testing for ORA func. using the r fault−free tracks
o/p = IO−FF -> {ff ORA PLBs} non−IO−FF -> {suspect ORA PLBs}
Non−TMR interconnect global testing phase of I−BIST:
Interconnect global testing of the remaining (t−s) untested tracks using ff ORAs of the IO−FF set
o/p = T−FF -> {ff tracks} non−T−FF -> {suspect tracks}
FF ORA/stick detection subphase of the shuffled−TPG phase of HD−BIST:
Fault diagnosis of PLBs of the non−IO−FF set for ORA func. using ff tracks of the T−FF set
o/p = ORA−FF -> {ff ORA PLBs} ORA−FF is super-set of IO−FF
Detailed testing phase of I−BIST:
Fault diagnosis of Interconnect tracks of the non−T−FF set using ff ORAs of the ORA−FF set
o/p=I−FF->{ff interconnects} non−I−FF->{faulty interconnects}
PLB fault detection and adaptive diagnosis phases of HD−BIST:
Fault detection and diagnosis of PLBs for their operational func. using ff interconnects of the I−FF set and ff ORAs of the ORA−FF set
o/p = PLB−FF -> {ff PLBs} non−PLB−FF -> {faulty PLBs}
END

Figure 10: *Flowchart for the improved mixed testing of PLBs and interconnects sans fault-free assumptions.*

HD-BIST has two sub-phases, detecting sticks with fault-free i/o-pins (IO-FF stick detection) and fault-free ORA/stick detection [15]. In the IO-FF stick detection sub-phase, the CUT PLB LUTs (lookup tables) and the ORA PLB LUTs are each configured as 2-bit XORs. Thus, for a PLB with q LUTs, only two nets are required to supply the 2-bit test vectors to the CUT LUTs while q nets are required to supply outputs of the CUTs to the ORA. Thus, $r = q + 2$ tracks are enough to implement the IO-FF stick detection sub-phase. The fault-free ORA/stick detection sub-phase, on the other hand, may require $r' > r$ fault-free tracks since it is an adaptive sub-phase and the reconfigured BISTer areas may overlap, thus requiring independent interconnect tracks for each of the overlapping BISTers; see Fig. 6b. Thus, using TIB in the first approach will require $3r'$ interconnect tracks. In the second approach, interconnect global testing is the initial phase and using TIB in this phase will require $3k$ ORAs, where k is the number of inter-net-set and intra-net-set pair-wise comparisons (see Sec. 3.2), and is given by $k = m(t-1) + (m-1)t$, where m = number of net-sets in a configuration and t = number of tracks per channel. With a maximum value of $m = 3$ for configuration 3 of I-BIST [14], we get $k = 5t - 3$.

We next present an improvement of the above two approaches by further reducing the amount of test circuit components required in TIB. This is achieved by interleaving the sub-phases of the shuffled-TPG phase of HD-BIST and the interconnect global testing phase. Note that the IO-FF stick detection sub-phase of the shuffled-TPG phase of HD-BIST requires only $r < r'$ fault-free tracks as test circuit components. Hence, instead of testing all t tracks in the interconnect global testing phase using the TIB technique (as in Approach 2 above) we test only s tracks ($r < s << t$)[3] at a time using the TIB technique, with the aim of finding r fault-free tracks (required for the IO-FF stick detection sub-phase). This way we require only $3k' < 3k$ ORAs ($k' = 5s - 3$) in this sub-phase of the interconnect global testing phase (we will refer to this sub-phase as the TMR interconnect global testing sub-phase). If the required r fault-free tracks are found then using the r fault-free tracks, we perform the IO-FF stick detection sub-phase of the shuffled-TPG phase of[4]. However, if r fault-free tracks are not found then we perform TMR-based interconnect global testing of another set of s tracks.

The PLBs of the IO-FF sticks found in the above sub-phase are fault-free for the ORA and TPG functions with high probability. Hence we use these PLBs of the IO-FF sticks as non-TMR'ed ORAs and TPGs for testing the remaining $(t - s)$ tracks in the interconnect global testing phase (we will refer to this sub-phase of the interconnect global testing phase as the non-TMR interconnect global testing sub-phase). Once the non-TMR global testing sub-phase is completed and all the available fault-free tracks in the ROTE area t_{ff}, with $t_{ff} > r'$, are detected, then we perform the fault-free ORA/stick detection sub-phase of the shuffled-TPG phase of HD-BIST. Thus, in this approach we split the interconnect global testing phase into two sub-phases, implement one using TIB and the other without using TMR (i.e., non-redundant BIST), and interleave these two sub-phases with the two sub-phases of the shuffled-TPG phase of HD-BIST, both implemented without using TMR. The benefit of this interleaving is that only $3k'$ test circuit components are required for TMR-based BIST instead of $3k$ ($k' < k$), that were required in Approach 2.

Once the above two phases, the interconnect global testing phase and the shuffled-TPG phase of HD-BIST, are completed, then we can perform the remaining phases, interconnect detailed testing and PLB fault detection and adaptive diagnosis, in a similar fashion to either of the previous two approaches; i.e., performing the remaining phases in any order. However, in some cases, it is beneficial to perform the interconnect detailed testing phase before the PLB detection and diagnosis phases, because with higher fault densities the fault-free tracks found in the interconnect global testing phase may not meet the requirement of fault-free test circuit components for the PLB detection and diagnosis phases. In other words, the PLB detection and diagnosis phases may require $r'' > t_{ff}$ fault-free tracks. Fig. 10 gives the flow of the different phases in the improved M-BIST technique.

5 Simulation Results

A 32×32 FPGA array with PLBs having three 3-input 1-output LUTs each, and interconnect channels containing 16 tracks of single-length wire-segments, was functionally simulated in C++ with random functions mapped to each PLB—each LUT entry was randomly chosen to be 0 or 1 with a probability of 0.5. Three techniques were simulated: M-BIST (no fault-free assumptions), HD-BIST for PLBs with erroneous fault-free assumption for interconnects and I-BIST for interconnects with erroneous fault-free assumption for PLBs. Note that

[3]In our implementation we use $s = 2r$.

[4]Note that, we may not be able to perform the fault-free ORA/stick detection sub-phase of the shuffled-TPG phase of HD-BIST using the r fault-free tracks as this sub-phase requires $r' > r$ fault-free tracks.

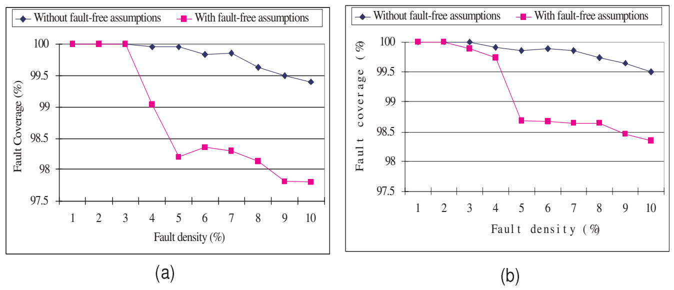


Figure 11: *Fault coverage results for random faults for (a) PLBs using M-BIST (no fault-free assumptions) and HD-BIST (with [erroneous] fault-free assumptions) and (b) interconnects for M-BIST (no fault-free assumptions) and I-BIST (with [erroneous] fault-free assumptions).*

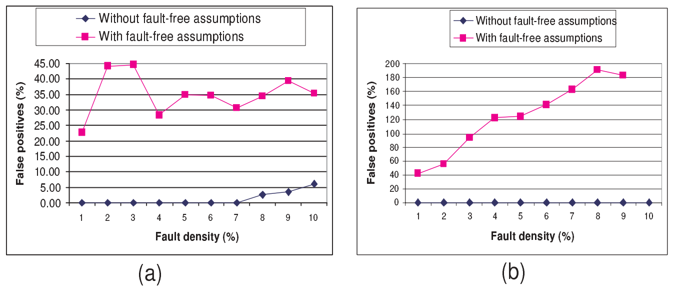


Figure 12: *False positive results for random faults for (a) PLBs using M-BIST (no fault-free assumptions) and HD-BIST (with [erroneous] fault-free assumptions) and (b) interconnects for M-BIST (no fault-free assumptions) and I-BIST (with [erroneous] fault-free assumptions).*

faults are injected in both PLBs and interconnects to simulate realistic conditions irrespective of the fault-free assumptions of two of the techniques. We compared the results of M-BIST with the corresponding results of HD-BIST (for PLBs) and I-BIST (for interconnects) for random and clustered faults[5]. Faults injected in the interconnects are those described in Sec. 3.2, while faults injected in the PLBs are stuck-at-0 and stuck-at-1 faults on the input/output pins and LUT configuration bits. We measured two metrics in our simulations, *fault coverage* defined as the percentage of faults that are correctly diagnosed, and *false positives* defined as the number of fault-free components that are incorrectly diagnosed as faulty. We report false positives as a % of the number of injected faults, i.e., a PLB false positive of 10% when there are 50 faulty PLBs means that 5 fault-free PLBs are mis-diagnosed as faulty.

Figs. 11-12 show the comparisons of fault coverage and false positives, respectively, between M-BIST and HD-BIST (part (a) of the plots) and I-BIST (part (b) of the plots), for randomly injected faults; while Figs. 13-14, show the same comparisons for clustered fault patterns. The results are for faults present in both PLBs and interconnects, and they show appreciably better fault coverage (between 98.5-100% for fault densities up to about 16%) and huge reductions in false positives (to the point of being negligible—on the average $< 1\%$) for M-BIST, for both PLBs and interconnects, compared to HD-BIST and I-BIST, respectively. The false positive %'s are quite dramatic for PLBs (up to 45%) using HD-BIST and interconnects (up to 200%!) using I-BIST when erroneous fault-free assumptions are made about the other type of components. The results clearly indicate the fallacy of making fault-free assumptions.

6 Conclusions

We presented a novel mixed PLB and interconnect testing technique (M-BIST), that, without any knowledge of the faulty state of any PLB

[5]Faults occur in a cluster around a center C with a probability distribution function of $\frac{g}{d+1}$ where d is distance from C and g is a suitable constant.

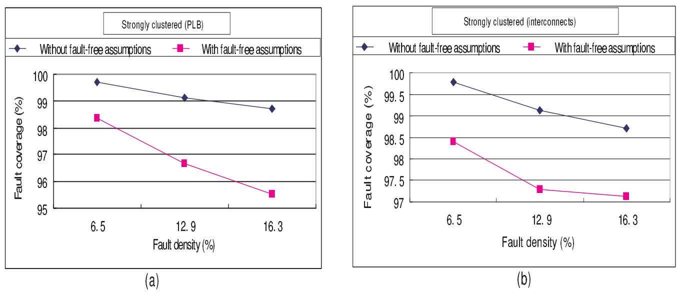


Figure 13: *Fault coverage results for clustered faults* ($g = 0.4$) *for (a) PLBs using M-BIST (no fault-free assumptions) and HD-BIST (with [erroneous] fault-free assumptions) and (b) interconnects for M-BIST (no fault-free assumptions) and I-BIST (with [erroneous] fault-free assumptions).*

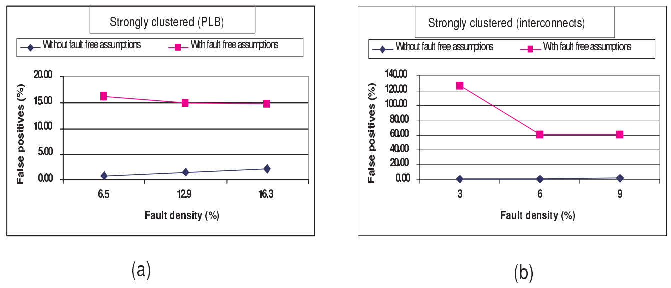


Figure 14: *False positive results for clustered faults* ($g = 0.4$) *for (a) PLBs using M-BIST (no fault-free assumptions) and HD-BIST (with [erroneous] fault-free assumptions) and (b) interconnects for M-BIST (no fault-free assumptions) and I-BIST (with [erroneous] fault-free assumptions).*

or interconnect and without making any fault-free assumptions, is able to achieve very high diagnosability (98.5-100%) even in presence of high fault densities (up to 16%) and clustered faults. To the best of our knowledge this is the first BIST technique that does not require any unrealistic assumption of interconnects being fault-free when testing PLBs and vice-versa. Our simulation results also confirmed the fallacy of making fault-free assumptions, especially on the issue of false positives (fault-free components mis-diagnosed as faulty). Our techniques should thus prove very effective for test and diagnosis of emerging VDSM as well as future nano-scale FPGAs that are expected to have high fault densities across both PLBs and interconnects.

References

[1] M. Abramovici, at al., "Improving on-line BIST-based diagnosis for roving STARs", Proc. 6th IEEE Int'l On-line Testing Workshop, pp. 31-39, 2000.
[2] C. Hamilton et al., "Enhanced BIST-based Diagnosis of FPGAs via Boundary Scan Access", Proc. IEEE VLSI Test Symp., pp. 413-418, 1999.
[3] W.K. Huang and F.Lombardi, "An approach to testing of Programmable/Configurable FPGAs", Proc. IEEE VLSI Test Symp. , pp. 450-455, 1996
[4] W.K. Huang, F.J.Meyer, X. Chen and F. Lombardi, "Testing configurable LUT-Based FPGAs", IEEE Trans. VLSI systems, Vol. 6, No. 2, pp 276-283
[5] T.Inoue and H. Fujiwara, "Universal Fault Diagnosis for lookup table FPGAs", IEEE Design & Test of Compt.,Vol 15, No.1, Jan 1998
[6] J. Liu and S. Simmons, "BIST-Diagnosis of Interconnect Fault Locations in FPGAs", CCECE 2003.
[7] M.Y. Niamat et. al., "A BIST scheme for testing interconnects of SRAM-Based FPGAs", Midwest Symposium, Vol 2, Aug 2003.
[8] N. R. Shnidman, W. H. Mangione-Smith, and M. Potkonjak, "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 .
[9] A.Steininger and C. Scherrer, "On the Necessity of On-Line BIST in Safety-Critical Applications", Proc. 29th Fault Tolerant Computing Symp., pp. 208-215, 1999.
[10] Chuck Stroud et al., "On-line BIST & Diag. of FPGA Interconnect Using Roving STARs", Proc. IEEE Int'l On-line Test. Workshop, 2001.
[11] C. Stroud, E. Lee and M. Abramovici, "BIST-based diagnostics of FPGA Logic Blocks", Proc. IEEE Int'l Test Conf., pp. 539-547, 1997
[12] C. Stroud et al., "Built-In Self-Test for Programmable Logic Blocks in FPGAs (Finally a Free Lunch: BIST Without Overhead!)", Proc. IEEE VLSI TEST Symp., pp. 387-392, 1996.
[13] X. Sun, S. Xu, B. Chan and P. Trouborst, "Novel Technique for Built-In Self-Test of FPGA Interconnects", Proc. IEEE Int'l Test Conf., pp. 795-803, 2000
[14] V. Suthar and S. Dutt, "Efficient On-line Interconnect Testing in FPGAs with Provable Detectability for Multiple Faults", submitted paper—available for reference at www.ece.uic.edu/~dutt/papers/ibist05.pdf
[15] V. Suthar and S. Dutt, "High-Diagnosability Online Built-In Self-Test of FPGAs via Iterative Bootstrapping", GLSVLSI, pp.78-83, 2005.
[16] V. Verma, S. Dutt and V. Suthar, "Efficient On-line Testing of FPGAs with Provable Diagnosabilities", Proc. DAC, pp. 498-503, 2004.