On-Line BIST and BIST-Based Diagnosis of FPGA Logic Blocks

Miron Abramovici, *Fellow, IEEE*, Charles E. Stroud, *Senior Member, IEEE*, John M. Emmert, *Senior Member, IEEE*

Abstract **- We present the first on-line Built-In Self-Test (BIST) and BIST-based diagnosis of programmable logic resources in Field Programmable Gate Arrays (FPGAs). These techniques were implemented and used in a roving Self-Testing AReas (STARs) approach to testing and reconfiguration of FPGAs for fault-tolerant applications. The BIST approach provides complete testing of the programmable logic blocks (PLBs) in the FPGA during normal system operation. The BIST-based diagnosis can identify any group of faulty PLBs, then applies additional diagnostic configurations to identify the faulty look-up table (LUT) or flip-flop within a faulty PLB. The ability to locate defective modules inside a PLB enables a new form of fault-tolerance that reuses partially defective PLBs in their fault-free modes of operation.**[1]

Index Terms **- Adaptive Computing, Built-In Self-Test, On-Line Testing, Fault Diagnosis, Fault Tolerance, Field Programmable Gate Arrays.**

I. Introduction

Field Programmable Gate Arrays (FPGAs) that can be reprogrammed in system allow a system using reconfigurable hardware to adapt to changes in its external environment, and to extend its initial capabilities by implementing new functions on the same hardware platform. In-system reprogrammability results in increased functional density and reduced power consumption - features very important in many domains, such as space missions or mobile devices. The reprogrammability and the regular structure of FPGAs are ideal to implement low-cost fault-tolerant hardware, which makes them very useful in systems subject to strict high-reliability and high-availability requirements, such as systems deployed in harsh and/or hostile environments or in remote unmanned long-life missions, or systems whose operation is critical and must continue uninterrupted. These types of applications rely on on-line testing to protect against both transient failures and permanent faults that appear during the lifetime of the system. On-line testing must also provide high-resolution diagnosis, because to bypass faulty resources, they must be first accurately located.

However, most previous methods for FPGA testing address only off-line testing using either externally applied vectors [1]-[9] or Built-In Self-Test (BIST) [10]-[17]. On-line FPGA testing methods based on coding techniques [18][19] do not diagnose the detected faults. On-line testing based on triple (or multiple) modular redundancy have a very high overhead. The "fault-scanning" method [20] achieves low-overhead on-line testing, but it is applicable only to bus-based FPGA architectures, it assumes that certain parts of the FPGA are fault-free, and it requires a modified design of the FPGA blocks.

One problem with conventional on-line testing is that it detects only faults that affect the current operation performed in the system. New faults, however, are equally likely to occur in spare resources or in the currently unused portion of the operational part of the system. These dormant faults may accumulate and have a very detrimental effect on the system reliability [21]. Thus to guarantee a reliable operation, the on-line test must completely check all the system resources, including spares.

In this paper, we present new techniques for on-line testing and diagnosis of non-transient faults in programmable logic blocks (PLBs), developed as part of the *Roving STARs* project [22][23]. A STAR, or a Self-Testing ARea, is a temporarily off-line section of the FPGA where self-testing goes on without disturbing the normal system activity in the rest of the chip. The roving of the STARs periodically brings every section of the device under test. This approach guarantees complete testing of the FPGA (PLBs, programmable interconnect, and configuration memory), including all spare resources, and does not require any part of the chip to be fault-free. Recently, the roving STARs approach has been extended for testing delay faults [24]. Our techniques are applicable to any FPGA supporting incremental run-time reconfiguration, which is the ability to reconfigure only a part of the FPGA, while normal operation continues uninterrupted in the rest of the device. FPGAs supporting this feature include ORCA (from Lattice), and Virtex and Spartan (from Xilinx).

Our on-line BIST approach detects any combination of multiple faulty PLBs, with the exception of a few pathological cases unlikely to occur in practice. Our diagnosis approach is based on analyzing the BIST results, followed (when needed) by an adaptive divide-and-conquer method to obtain maximum diagnostic resolution. These techniques uniquely identify any combination of multiple faulty PLBs, again with the exception of a few pathological cases. Additional BIST-based diagnostic techniques identify the defective parts inside a faulty PLB, such as a faulty look-up table (LUT) or flip-flop. We used ORCA [25] for our implementation, but we emphasize that our technique can be applied to any FPGA that features in-system incremental run-time reconfiguration, such as Virtex and Spartan series FPGAs.

The remainder of this paper is organized as follows. Section II outlines the roving STARs approach and describes the on-line BIST method. The fault detection capabilities of the approach is ddescribed in Section III and Section IV presents

1. Manuscript received November 11, 2002, revised December 17, 2003. This material is based upon work supported by the DARPA Adaptive Computing Systems program under contract F33615-98-C-1318.

Miron Abramovici is with DAFCA, 1671 Worcester Rd. Suite 303, Framingham, MA, 01701 USA (email: miron@dafca.com)

Charles E. Stroud is with the Department of Electrical and Computer Engineering, 200 Broun Hall, Auburn University, AL 36849 USA (phone: 334-844-1806; fax: 334-844-1809; email: cestroud@eng.auburn.edu)

John M. Emmert is with the Department of Electrical and Computer Engineering, University of North Carolina at Charlotte, Charlotte, NC 28223 USA (email: jmemmert@uncc.edu)

diagnosis based on analysis of the BIST results. Section V discusses diagnosis of faults internal to PLBs, and Section VI presents experimental results obtained on known defective ORCA FPGAs. Finally, Section VII presents conclusions and suggestions for future research directions. The acronyms and abbreviations used in the paper are summarized in Table I.

II. The Roving STARs

We assume that an embedded processor controls the test, diagnosis, and fault-tolerance functions associated to all the FPGAs in the system. This *Test and Reconfiguration Controller* (TREC), which has separate processor-specific fault-tolerance mechanisms [26], also maintains all the configurations that may be used in the future. The roving STARs approach targets non-transient faults that appear during the lifetime of the system. Here a fault is one faulty PLB that may have several arbitrary internal faults. Once faulty resources have been detected and identified, our fault-tolerant techniques, presented in [27], reuse faulty logic resources whenever possible, and bypass faults using fault-free spare resources otherwise. As a result, any combination of faulty logic resources can be tolerated given sufficient spare resources in the FPGA. For on-line testing of transient faults, the application logic implemented in FPGAs employs a concurrent error-detection technique, such as the one described in [19]. Fault tolerance for transient faults is achieved by periodically saving the state of the system (checkpointing) and restoring the last saved state when a transient is detected.

Fig. 1a illustrates the initial floor plan of an FPGA, having a contiguous working area and a spare area consisting of a vertical STAR (V-STAR) with two columns and an horizontal STAR (H-STAR) with two rows. A STAR tests both the PLBs and the programmable interconnect in its area, but in this paper we will focus only on PLB testing. A roving step involves relocating a slice of the system logic adjacent to the STAR in the current STAR position, and recreating the STAR logic in the area vacated by the system logic. This is illustrated in Fig. 1b after one roving step across of the V-STAR and one roving step down of the H-STAR. Testing the entire FPGA takes one full horizontal sweep of V-STAR and one full vertical sweep of H-STAR. While only one STAR would have been sufficient to test all the PLBs in the FPGA, having two roving STARs makes possible complete testing of interconnect [28], and also provides essential features for diagnosis and fault tolerance [27]. For an $N\times N$ FPGA, a full sweep of one STAR requires $N/2$ positions, thus the total number of roving positions is $N^2/4$. The roving process is based on run-time reconfiguration and it is controlled by TREC [22].

A STAR consists of several disjoint tiles, with the PLBs in each tile configured as shown in Fig. 2, in a structure called a BISTER. All BISTERs in the same STAR work concurrently. A BISTER inherits some of the concepts used in our off-line FPGA BIST methods for PLBs [11][14][29]. Specifically, the BISTER contains PLBs configured as a Test Pattern Generator (TPG) that applies pseudoexhaustive test patterns to two identically configured blocks under test (BUTs), whose outputs are compared by an Output Response Analyzer (ORA). The ORA latches and reports mismatches as test failures. Direct comparison avoids possible aliasing errors introduced by compressing responses into signatures [30]. The additional testing typically required in comparison-based testing to ensure that the comparator is fault-free is not needed in our approach, since every PLB in turn is tested as a BUT. The TPG and ORA are reset by TREC just prior to initiating the BIST sequence. Fig. 3 illustrates an ORA comparing four pairs of outputs (denoted O1 through O4) from two BUTs (denoted B1 and B2). The flip-flop stores the result of the comparison, and the feedback loop latches the first mismatch in the flip-flop. The *Pass/Fail* result flip-flops of all ORAs are connected to form the scan chain [11]. A *test phase* is defined as a test configuration that tests a BUT in one of its modes of operation, while a *test session* is defined as the set of test phases that test a BUT in all of its modes of operation [29]. In every test phase, the scan chain is also tested, to assure the integrity of the test results.

The TREC accesses the FPGA using its boundary-scan mechanism [31], so that this access is transparent to the normal function of the chip (most FPGAs support boundary scan). The TREC employs the boundary-scan interface to reconfigure the STARs for different test and diagnosis operations, to initiate the BISTERs, and to scan out the test results. BISTERs work with the boundary-scan test clock, *TCK,* and the configuration clock is also derived from *TCK*. The first configuration of a BISTER checks the proper operation of the scan register, inducing mismatches by comparing BUTs with different configurations. This also protects against the case of all ORA flip-flops being stuck at the “Pass” value. The TREC also controls the system clock(s) and has the capability to stop the system operation for short intervals to allow for safe relocation of a slice of the system logic in the last tested area. If faults are detected, the TREC starts the diagnosis process.

III. Detecting Faulty PLBs

The BUTs are repeatedly configured to be tested in all their modes of operation. In order to minimize fault latency, it is important to minimize the number of BIST configurations required to completely test the PLBs; this is particularly important since the configuration download time is much greater than the BIST application time. The modes of operation of a PLB may be determined only from the information available in the FPGA data book, without having a detailed knowledge of its implementation. Since the BISTER provides complete testing only for the BUTs, we have to reconfigure every BISTER several times so that each PLB will be a BUT in at least one configuration. The number of PLBs for an ORA and for a TPG depend on the target PLB architecture. In our implementation in the ORCA FPGA, a TPG needs three PLBs and an ORA only one. The same numbers of PLBs are needed for Virtex and Spartan II FPGAs.

Fig. 4a illustrates six floor plans of a 3×2 BISTER tile, where T, B, and O denote, respectively, a TPG cell, a BUT, and an ORA. The goal of the six configurations is to systematically rotate the functions of the PLBs, so that eventually every PLB in the tile is completely tested twice, each time being compared with a different BUT. This rotating strategy assures that every single faulty PLB and almost all combinations of multiple

faulty PLBs are guaranteed to be detected; these results are proven next.

Claim 1: Any single faulty PLB is guaranteed to be detected in at least two BISTER configurations.

Proof: The faulty PLB is a BUT in two BISTER configurations, where exhaustive input patterns are produced by a fault-free TPG, and its outputs are compared with a fault-free BUT by a fault-free ORA. Hence no fault (single or multiple) detected in the BUT can escape detection in these two configurations. ■

Claim 2: Except for a few pathological cases, any pair of faulty PLBs is guaranteed to be detected in at least one BISTER configuration.

Proof: Since any single faulty PLB is detected, a pair of faulty BUTs will escape detection only if they have a circular masking relation [32], where each faulty PLB masks the detection the other one. This masking should occur in any BISTER configuration where a single faulty PLB would be detected. Because a TPG or an ORA containing a faulty PLB may still work correctly, we analyze only configurations where at least one of the faulty PLBs is a BUT.

Case 1: Both faulty PLBs are BUTs in one BISTER configuration. Then the TPG and the ORA are fault-free, and the only way the faulty pair can escape detection is to have functionally equivalent faults, since then the compared outputs will not mismatch. Assume that the faulty PLBs are BUTs in the configuration 1 in Fig. 4b. Then in configurations 3 and 5, one of them is a BUT (compared with a fault-free BUT by a fault-free ORA), and the other is a TPG cell. The only way the faulty pair can escape detection is for the faulty TPG to skip exactly the patterns that detect the faulty BUT. We say that this is a pathological case, because it has an extremely low probability of occurrence: we need two faulty PLBs, and they must have functionally equivalent faults, and when a faulty PLB is part of the TPG, the TPG must skip those patterns that detect the faulty BUT.

Case 2: At most one faulty PLB is a BUT in any BISTER configuration. Assume, without loss of generality, that the faulty cells are in the first row of the BISTER, as shown in Fig. 4c. Let X be the faulty BUT and Y the faulty TPG cell in configuration 1. To have circular masking between X and Y, all of the following conditions must be true: 1) in configuration 1, the TPG must skip those patterns that detect X; 2) in configuration 4, the TPG must skip those patterns that detect Y; 3) in configuration 2, the ORA X must not record the mismatch between Y and the fault-free BUT; and 4) in configuration 3, the ORA Y must not record the mismatch between X and the fault-free BUT. Again, this is a pathological situation because we need the AND of four conditions, which are very unlikely by themselves. ■

Clearly, similar arguments can be made for more than two faulty PLBs. Hence we can conclude that:

Claim 3: In practice, any combination of faulty PLBs is detected in at least one BISTER configuration.

To overcome routing congestion problems, spare PLBs may be incorporated into the BISTER tile to provide additional routing resources. Here “spare” means “not involved in testing,” as all the PLBs in a STAR are “spare” with respect to normal operation. Fig. 5 illustrates the eight rotations of the 4×2 BISTER tile with two spare cells (denoted by S) used in our implementation. While these are not “pure” rotations, since they maintain the three TPG cells in the same column, they do achieve the same property as the rotations of the 3×2 tile. The presence of the spare cells makes circular masking relations even more unlikely, since any masking relation between two faulty PLBs disappears whenever one of them becomes a spare.

IV. Locating Faulty PLBs

In every BIST configuration, the TREC records the set of failures obtained at the ORA outputs. Here we present a diagnosis method that attempts to locate faulty PLBs based on analyzing the test results, and we will show that in many cases, accurate resolution can be obtained without additional reconfigurations. Most importantly, the frequent case of a single faulty PLB falls in this category.

In any fault location procedure, maximum diagnostic resolution is achieved when faults are isolated within an equivalence class containing all the faults that produce the observed response. If every equivalence class has only one fault, we say that the fault is *uniquely diagnosed*, that is, there is no other fault which can produce the same response. In our case, a fault is one faulty PLB that may have several internal faults, and the response is the set of failing test phases detected at the outputs of the ORAs. We begin by assuming a single faulty PLB in the FPGA, then we analyze the case of multiple faulty PLBs; in Section V we will discuss locating faults inside a defective PLB.

We can observe that the PLBs in a BISTER tile can be partitioned into two disjoint sets, so that every set contains the PLBs that are pairwise compared when configured as BUTs. For example, for the 3×2 tile, the BUTs in configurations 1, 3, and 5, are compared only among themselves and they are never compared to the BUTs in phases 2, 4, and 6 (see Fig. 4c). A similar partition exists for the 4×2 tile as well. We can represent the relations between BUTs that are pairwise compared and the ORAs that observe them by the graphs shown in Fig. 6, where a BUT is denoted by a node B_i, and the ORA that observes BUTs B_i and B_j is denoted by O_{ij}. We can view such a graph as representing a combined test session, in which half of the BUTs in the tile are concurrently tested. Each tile is completely tested in two combined test sessions, where the BUTs in one session are ORAs in the other one, and vice-versa.

Theorem 1: Any single faulty PLB is guaranteed to be uniquely diagnosed.

Proof: When the faulty PLB is configured as a BUT, it is detected at its two adjacent ORAs, and no other BUT is detected at the same two ORAs. Note that when the faulty PLB is configured as a TPG cell, no error is generated even when the fault modifies the TPG patterns, because the two BUTs still receive the same patterns. When configured as an ORA, the faulty PLB may generate an error, but this will be in addition to the failures observed in the combined session when it is a BUT, and these results are sufficient to distinguish it from any other single faulty PLB (in Section VI we will describe this situation as encountered in an actual faulty FPGA). ■

As a consistency check, we can verify that the sets of failures obtained at the two ORAs are identical.

The following results deal with the location of a group G of faulty PLBs that is detected in at least one test session. To uniquely diagnose G means to identify all its faulty blocks such that no other group (including subsets of G) can produce the same result. Although unique diagnosis is not always possible, there are many situations when it can be guaranteed.

We will analyze the interactions among faulty PLBs under the following assumption:

Assumption A1: A TPG with faulty PLBs does not skip the patterns that detect faults in a BUT. (Later in this section we will analyze the situation when this assumption is not true.)

Based on this assumption, the set of failing phases obtained at ORA O_{ij} is given by:

$$FO_{ij} = (FB_i \cup FB_j) - Feq_{ij} \quad (1)$$

where FB_k is the set of failing phases of BUT B_k, and Feq_{ij} is the set of failing phases of both B_i and B_j that have identical responses (and thus do not cause mismatches at O_{ij}). In Fig. 7, the area of FO_{ij} is marked by diagonal lines. Note that FO_{ij} is empty ($\varnothing$) when both B_i and B_j are fault-free, or when the faults in B_i and B_j are equivalent (since then $FB_i = FB_j = Feq_{ij}$). Also note that there exists one situation when FO_{ij} is the same, no matter if only one or both of the BUTs observed at O_{ij} are faulty; this occurs if the two BUTs have the same sets of failing phases ($FB_i = FB_j$), but their faulty responses are never the same ($Feq_{ij} = \varnothing$).

Knowing the set of failing phases observed at O_{ij} and the complete set of failing phases of one of the two faulty BUTs, we can determine the set of failing phases where the two BUTs have identical responses by

$$Feq_{ij} = FB_i - FO_{ij} = FB_j - FO_{ij} \quad (2)$$

Based on FO_{ij} and one of the two sets, we can also compute a lower bound on the other set by

$$FB_i \supseteq (FO_{ij} - FB_j) \cup Feq_{ij} \quad (3)$$

or by

$$FB_j \supseteq (FO_{ij} - FB_i) \cup Feq_{ij} \quad (4)$$

Note that (3) becomes an equality when FO_{ij} and FB_j are disjoint:

$$FB_i = FO_{ij} \cup Feq_{ij} \quad (5)$$

This occurs when $FB_i \subseteq FB_j$ and $Feq_{ij} = FB_i$.

For the diagnosis procedure, we make one more assumption:

Assumption A2: No more than two faulty BUTs have identical responses in the same failing phase.

To justify this assumption, recall that in every configuration the TPG applies exhaustive tests for that mode of operation. If assumption *A2* is not satisfied, it means that we would have three or more BUTs with equivalent faults for that mode of operation, which is quite an unlikely event.

The following results will be used by our diagnosis procedure.

Lemma 2: If none of the two ORAs observing BUT B fails in phase p, then B does not fail phase p.

Proof: Assume, by contradiction, that B fails phase p. Based on assumption *A1*, the vectors detecting its faults are generated by the TPG. Then the other two BUTs observed by the two ORAs must have equivalent faults to the fault of B in phase p, because p is not reported by either of the two ORAs. But then we would have three BUTs with equivalent faults in the same phase, which contradicts assumption *A2*. Therefore B does not fail phase p. ■

Lemma 3: If the two ORAs observing the same BUT report no failures, then their common BUT is fault-free.

Proof: Based on Lemma 2, the common BUT of the two ORAs does not fail any phase. ■

Lemma 4: Any BUT failure appears at least at two ORAs.

Proof: Assume, by contradiction, that phase p is reported only at one ORA. Let B be the BUT failing p and O be the other ORA observing B. Since O does not report p as a failure, the other BUT (say C) observed by O must have identical response in p. But the other ORA observing C does not report p as a failure either, so we have more than two BUTs with equivalent faults in p, which contradicts assumption *A2*. Therefore p must appear at the output of at least two ORAs. ■

Lemma 5: If we have failures only at two ORAs, they must have identical failures.

Proof: Based on Lemma 4, any failure must appear at least twice, and in this case we have only two failing ORAs. Hence their failures must be identical. ■

Theorem 2: (For the 4×2 tile) If only two ORAs observing the same BUT report failure in phase p, their common BUT fails in phase p.

Proof: In Fig. 8a, without loss of generality, assume that O_{12} and O_{14} are the two failing ORAs reporting p as a failure. Assume, by contradiction, that their common BUT B_1 does not fail phase p. Then both B_2 and B_4 must fail p. But since no other ORA reports p, then B_3 must also fail in phase p, and its response must be identical to that of B_2 and B_4. Hence we would have three BUTs with equivalent faults in phase p, which contradicts assumption *A2*. Therefore B must fail phase p. ■

Lemma 6: If two ORAs observing the same BUT report no failures, then all three BUTs observed by them are fault-free.

Proof: Based on Lemma 3, the common BUT of the two ORAs is fault-free. So each one of the other BUTs is compared with a fault-free one by an assumed fault-free ORA, and no failure is reported. Hence the other two BUTs are also fault-free. ■

Lemma 7: (For the 4×2 tile) If only two ORAs without a common BUT fail phase p, then at least one pair of BUTs between the two ORAs are faulty and have identical response in phase p.

Proof: In Fig. 8b, without loss of generality, assume that O_{12} and O_{34} are the two ORAs without a common BUT failing phase p. At O_{12}, p may be caused by B_1 or B_2. Let us assume that B_1 fails p. Because p is not observed at O_{14}, B_4 must also fail p with an identical response. Similarly, had we assumed that B_2 fails p, we would have concluded that B_2 and B_3, have equivalent faults in phase p. Note that all four BUTs may be faulty as well, but in this case the equivalent faults in the two pairs will be different. ■

Note that under the condition of Lemma 7, we cannot reach a unique diagnosis. However, we can move to a second phase of diagnosis where we divide the set of suspected PLBs and test each subset in separate BISTER tiles. The BIST results of each

separate BISTER is then analyzed to achieve a unique diagnosis. If unique diagnosis cannot be obtained for a given subset of suspected PLBs, it can be further subdivided and re-tested. A subsequent example will show how this technique can uniquely diagnose the faulty PLBs.

Lemma 8: Assume that two out of three ORAs fail, such that the middle one has no failures, and the other two have disjoint failures. Then the two BUTs observed by the non-failing ORA are fault-free and the other two BUTs are faulty.

Proof: In Fig. 8c, without loss of generality, assume that O_{34} has no failures, while O_{14} and O_{23} have disjoint failures. Then we want to prove that B_3 and B_4 are fault-free and B_1 and B_2 are faulty.

Assume, by contradiction, that one of the two BUTs observed by the non-failing ORA (say, B_3) fails phase p. Since O_{34} reports no failures, B_4 must have an equivalent fault in p. At most one of O_{14} and O_{23} may fail p, because their failures are disjoint. Then we have to analyze two cases:

Case 1: One of the failing ORAs (say, O_{23}) shows p as a failure. Since p is not observed at O_{14}, while B_4 fails p, B_1 must also have an equivalent fault in p. In this case we would have three BUTs with equivalent faults in p, which would contradict assumption *A2*.

Case 2: None of the failing ORAs reports p as a failure. But this would mean that both B_1 and B_2 must also have equivalent faults in p, since p does not appear at O_{14} or O_{23}. In this case we would have four BUTs with equivalent faults in p, which would contradict assumption *A2*.

Therefore both B_3 and B_4 are fault-free. Then B_2 is the source of the failures recorded at O_{23}, while those observed at O_{14} originate at B_1. ■

As a consistency check, we can verify that the failures recorded at the ORA between the faulty BUTs are the union of the disjoint sets of failures obtained at the other two failing ORAs (for the example above, $FO_{12} = FO_{14} \cup FO_{23}$).

The following examples deal with the results of only one combined test session. Under the assumption that a faulty PLB is detected only when configured as a BUT, the two sessions can be independently analyzed, but the their results must be cross-checked for compatibility.

Example 1: Fig. 9a shows the set of failing phases obtained at the four ORAs in one combined test session for a 4×2 BISTER tile. Because failures in test phase 1 are reported only at O_{23} and O_{34}, then B_3 is faulty and fails in phase 1 (by Theorem 2). In the same manner, B_4 is identified as failing phase 5, B_1 is identified as failing phase 9, and B_2 is identified as failing phase 7. Since failing phase 8 is reported at 3 ORAs, Theorem 2 does not apply and we cannot determine which BUTs fail phase 8 (should be at least two). Note that we have identified all four BUTs as faulty.

If our goal is also to identify the exact set of failing phases for every faulty PLB (to be able to reuse its fault-free modes of operation as a PUB), we can divide the set of suspected PLBs and re-test each subset to obtain additional information. Let us assume that we test B_3 in a separate BISTER with known fault-free PLBs and we find out that $FB_3 = \{1,8\}$. Then from equation (2) we obtain $Feq_{34} = \{1,8\}-\{1,5\} = \{8\}$, and from equation (4) we determine that B_4 must also fail phase 8. At this point we still do not know whether B_1 or B_2 is causing O_{12} to report failure in phase 8, so we again divide and re-test B_1 in a separate BISTER. Let us assume that we get $FB_1 = \{9\}$. Then from (2) we obtain $Feq_{12} = \varnothing$, and from (4) we determine that B_4 must also fail phase 8. Now we know the exact sets of failing phases for every BUT. ■

Example 2: Assume that we have $FO_{23} = FO_{34} = \{3,5\}$ and no failures at O_{12} and O_{14}. Based on Lemma 6, we conclude that B_1, B_2, and B_4 are fault-free, and from Theorem 2 we determine that B_3 is failing phases 3 and 5. ■

This example has illustrated the common pattern of a single faulty PLB.

Example 3: Assume that we have FO_{23}={2,5}, FO_{14}={3,4}, and no failures at O_{34} (see Fig. 9b). Because FO_{23} and FO_{14} are disjoint while O_{34} reports no failures, the conditions of Lemma 8 are satisfied, and we can conclude that B_3 and B_4 are fault-free and B_1 and B_2 are faulty. Moreover, from equation (1) we find that FB_1={2,5} and FB_2={3,4}. As a consistency check, we can verify that FO_{12} ={2,3,4,5}. ■

Example 4: Assume that we have FO_{23}={4}, FO_{12}={3}, FO_{34}={3,4}, and no failures at O_{14} (see Fig. 9c). Since only two ORAs with a common BUT fail phase 4, from Theorem 2 we determine that B_3 is faulty and fails phase 4. Because phase 3 is a common failure for two ORAs without a common BUT, from Lemma 7 we know that at least one of the pairs {B_1,B_4} or {B_2,B_3} have identical failures in phase 3. This is not a unique diagnosis, so we select one of the 4 suspects, say B_4, to be tested in a separate BISTER, and let us assume that we find that B_4 is fault-free. From Lemma 7 it follows that B_1 is also fault-free, and then B_2 and B_3 must be the faulty PLBs with identical responses in phase 3. ■

Note how by dividing suspect faulty PLBs into separate BISTER tiles, re-testing, and analyzing BIST results, we can uniquely diagnose three faulty PLBs that cannot be diagnosed by analyzing the failing BIST results alone.

In practice, we have extended the *MULTICELLO* (Multiple Faulty Cell Locator) algorithm, originally developed for off-line testing [29], to be used for identification of faulty PLBs in a 4×2 BISTER. Fig. 10 illustrates the algorithm analyzing the responses obtained at the ORAs in a 4×2 BISTER. The rows of the matrix are labeled to correspond to the BUT and ORA notation used in the previous examples, and columns correspond with phases. Note that the ORA denoted O_{14} has entries at the top and bottom of the table to account for the wrap-around effect of the BISTER and to allow direct application of the MULTICELLO diagnostic procedure. The 0 and 1 entries in matrix cells denote passing and failing results, respectively. In Fig. 10 we use the ORA results from Example 1, where all ORAs reported failures and all BUTs were determined to be faulty. The goal of the algorithm is to determine the set of failing phases for every faulty BUT. The algorithm first identifies only non-failing phases for BUTs, then proceeds to determine the failing ones based on the Theorems and Lemmas presented above.

Procedure MULTICELLO [29]*:*

1) Record ORA results and initialize the failures of every BUT

in each phase as unknown.

This initial state is shown in Fig. 10 step 1, where 0 and 1 entries for an ORA indicate, respectively, a passing and a failing result in the corresponding phase, and the empty cells denote unknown BUT failures. For example, O_{23} reports failures in phases 1, 7, and 8.

2) *In each column p, for every two consecutive ORAs with a 0 mark, enter a 0 for the BUT between them.*

This step applies Lemma 2 and Lemma 3, and its results are shown in Fig. 10 step 2 (new entries are shown in bold). We use the same 1 and 0 notation to respectively denote a BUT failure and a passing test.

3) *In each column p, for every two adjacent 0 marks followed by an empty cell, enter a 0 in the empty cell.*

This step applies Lemma 6; the two adjacent 0 marks belong to a BUT and an ORA, and the empty cell is the other BUT observed by the same ORA. The results are shown in Fig. 10 step 3. Note that one must consider the wrap-around effect of the BISTER to consider adjacent 0 marks.

4) *In each column p, for every adjacent 0 and 1 marks followed by an empty cell, enter a 1 in the empty cell.*

This step applies Theorem 2. The results are shown in Fig. 10 step 4. At this point, we have identified all 4 BUTs as being faulty.

5) *Consistency checks: If there is an ORA reporting a failure in phase p (marked with a 1), while neither of the two BUTs observed by the ORA fails in p (both are marked with a 0), then there is a potential inconsistency. If the failing ORA is reported as faulty in the other test session where it is a BUT, then go to* step 6. *Otherwise, divide the suspect PLBs into subsets, re-test and re-apply the procedure to each subset. If no further division is possible, then report inconsistency and exit.*

This step applies Lemma 4 and Lemma 5. A potential inconsistency in this step could be due to a faulty ORA, which will be identified when diagnosing the results of the second session, where that PLB is a BUT. As in any diagnosis algorithm, inconsistencies may arise when basic assumptions do not hold. *MULTICELLO* implicitly assumes that we are dealing with a logic fault, while the actual fault may be caused by an interconnect defect not equivalent to any logic fault; in this case the potential inconsistency is real, and it cannot be removed by the divide and re-test procedure. However, this procedure will remove an inconsistency caused by an invalid Assumption 1; we will analyze such a situation after this example.

6) *If every PLB has been identified as fault-free or faulty, the group of faulty PLBs has been uniquely diagnosed. Otherwise, divide the suspect PLBs into subsets, re-test and re-apply the procedure to each subset.*

In our example, all 4 BUTs have been properly identified as being faulty with the same failing phases as identified in Example 1. Although we have obtained unique diagnosis since all faulty BUTs have been identified, we cannot determine which BUTs fail phase 8 based on the BIST results obtained thus far. If such diagnostic information is required, it can be obtained by applying the same divide and re-test procedure mentioned in step 5.

End

Now we analyze situations when Assumption *A1* does not hold, and we show that unique diagnosis can be nevertheless achieved. First, note that even if Assumption *A1* is not valid, we still have at least one detection, because otherwise we will have a circular masking pathological case that we precluded in Claim 2 of Section III**.** Consider the 4×2 BISTER tile with two faulty PLBs illustrated in Fig. 11, which shows the four rotations for the test session in which these PLBs are under test. One faulty PLB is part of the TPG, while the other is a BUT. We assume that the TPG will skip all the patterns that detect the faulty BUT. There will be two different cases that can occur in which Assumption *A1* does not hold:

Case A: Only one PLB (the left one, for example) will cause the TPG to skip patterns that detect the faulty BUT, hence we do not get failures at O_1. For phase 1, MULTICELLO cannot make any entry in the table beyond step 1. For phase 2, MULTICELLO correctly identifies B_4 as faulty, but the fault in B_4 does not explain the failures at O_{13} in phase 1. Therefore, in step 6 we divide the suspect PLBs into subsets and re-test each subset. In at most two such steps, the two faulty PLBs will be separated and we will achieve unique diagnosis.

Case B: When either of the two faulty PLBs is part of the TPG, the TPG skips the patterns that detect the other faulty BUT. However, the faulty BUTs will be detected in different phases. As shown in Case B in Fig. 11, MULTICELLO completely fills the diagnosis table in step 3, but in step 5 it encounters an inconsistency, because all BUTs are identified as fault-free. Therefore we apply the divide and re-test procedure, which will eventually will separate the two faulty PLBs.

V. Diagnosis Within a Faulty PLB

During each test session, TREC records all ORA failures. This allows us to identify the failing mode(s) of operation of the faulty PLB and its faulty internal module(s). For example, ORCA 2C, Virtex, and Spartan II series PLBs contain four 4-input LUTs and four flip-flops with an output multiplexer network that can select any LUT or flip-flop output. The LUTs can also function as RAMs or fast adders. The flip-flops can also function as level-sensitive latches and have programmable set/reset and clock-enable features. The ORCA 2CA series FPGA PLB has the same architecture as the 2C series but has additional modes of operation including comparator, multiplier, and dual-port RAM [25]. Lines 1-9 in Table II summarize the test phases developed for the ORCA 2C series FPGA, while lines 10-14 describe the additional modes of operation tested for ORCA 2CA series. Phases 1-4 and 10-15 test the operation of the LUT/RAM module of the PLB, while phases 5-9 test the flip-flops and their modes of operation. Let us assume that a PLB fails only phases 5, 6, and 7. Since this faulty PLB passed the exhaustive tests for all its other modes of operation, it may be safely used for any function that does not involve counting (since phases 8 and 9 pass, the flip-flops work correctly as registers). This is the concept of a *partially usable block* (PUB) [22][27], where such a defective block is reused whenever possible in its fault-free modes of operation, to increase the

effective spare capacity of the FPGA in fault-tolerant applications. If phases 5-9 are the only ones that fail, the LUT module can still be used as a RAM or to perform any combinational logic function.

When the BIST results do not clearly indicate which component(s) of a faulty PLB is at fault, additional test phases are added to further isolate and identify the fault(s) and obtain even higher diagnostic resolution. Table III presents a list of additional diagnostic phases for the ORCA 2C15A FPGA. Phases 1 through 10 test the output multiplexer, ensuring that every LUT of flip-flop output can reach each output of the multiplexer. Phases 11 through 14 isolate and test the flip-flop modes of operation, while phases 15 through 18 exercise the LATCH operation.

Diagnosis within a PLB to determine which LUT(s) or flip-flop(s) are faulty requires additional BIST configurations. Once the normal BIST and diagnostic configurations for PLBs have been executed and the faulty PLBs are identified, special PUB diagnostic configurations are applied for the two or more rotations that will put the PLBs identified as faulty under test. A set of eight PUB diagnostic configurations are used to test the LUTs, while another set of four configurations tests the flip-flops. Four of the eight LUT diagnostic configurations program the LUT with a four-input exclusive-OR function while the other four program the LUT with a four-input exclusive-NOR function in order to test all LUT bits for stuck-at-0 and stuck-at-1. During each of the four LUT configurations for PUB diagnosis, the outputs of the LUTs are rotated through the four outputs of the multiplexer. In this way, if the error follows the rotation, the fault is known to be in a LUT and the faulty LUT is identified as illustrated in the example of a faulty LUT in Fig. 13. On the other hand, if the error does not rotate, then the output multiplexer is faulty and the faulty output is identified. However, a combination of faulty LUT and multiplexer output requires additional diagnostic configurations. The same rotation strategy is used for the flip-flop PUB diagnostic configurations as can be seen in Fig. 13 by substituting flip-flops for LUTs.

Any single faulty LUT or flip-flop can be identified and any combination of a single faulty LUT and single faulty flip-flop can be identified with the current ORA design in the ORAC 2C FPGA. In Fig. 3, the ORA flip-flop stores the combined result of comparing four pairs of BUT outputs which limits the PUB diagnostic resolution for this ORA design multiple faulty LUTs and flip-flops are encountered. Additional diagnostic resolution can be obtained by comparing each pair of outputs separately. In this way we can determine any combination of faulty LUTs and flip-flops, but at the expense of 24 PUB diagnostic configurations instead of 12 configurations. With the PLB architecture of more recent ORCA PLBs (like the 4C series FPGA), any combination of multiple faulty LUTs and flip-flops in the PLB can be identified. Fig. 14 illustrates this ORA implementation, which is also feasible in the Xilinx Spartan and Virtex series FPGAs [33].

Once the PUB diagnosis has completed for a given BISTER tile, the diagnostic procedure moves on to the next BISTER tile identified as having faulty PLBs until all PLBs that have been identified as faulty within a STAR have undergone PUB diagnosis. While this is the procedure in the current implementation, PUB diagnosis of multiple faulty BISTER tiles can be performed in parallel at the expense of slightly more complicated diagnostic software in TREC. When PUB diagnosis has been completed within a STAR, the identified faulty logic resources are processed by the fault-tolerance software to determine whether a faulty PLB can be utilized as a PUB by the next system function to occupy that position.

VI. Experimental Results

The diagnosis approach presented here was applied to three known-faulty ORCA 2C15A FPGAs [25] (originally from Lucent Technologies) that had been previously tested with the off-line approaches presented in [15] and [16]. All three FPGAs failed the on-line tests introduced in this paper. Applying the logic diagnosis techniques described in the previous sections produced inconsistent results for two FPGAs (Chip 1 and Chip 2). This was the expected result, because the faults in these devices are interconnect faults [16]. The logic fault located in the third faulty FPGA (Chip 3) matched the faulty PLB identified in [15].

Next we present the details of the on-line diagnosis of Chip 3. Fig. 15 shows the results of the test phases for both combined test sessions, where the failing test phases refer to Table II. In the first session, the same failures appear at two ORAs observing a common BUT, hence this BUT is faulty. In the second session, all test phases fail with failure indications at only one ORA which, at face value, results in an inconsistency in step 5 of the MULTICELLO diagnostic procedure since it contradicts Lemma 4. However, this ORA is being implemented by the faulty BUT identified in the first session, where the failing test phases indicate one or more faulty flip-flops in the BUT; then the failures in the second session are consistent with a single faulty PLB with faulty flip-flops. Application of the additional diagnosis configurations discussed in Section V and shown in Table III confirmed that the fault only affects the four PLB flip-flops, leaving the LUTs unaffected. This defective PLB was successfully reused as a PUB to implement combinational logic functions in a fault-tolerant application [27].

Total roving and testing time in a fault-free ORCA 2C15 FPGA is approximately 1.34 seconds when the ORCA 2C Boundary Scan interface is operated at its maximum specified clock rate of 10 MHz. As a result, in the worst case a fault could escape detection for almost 1.34 seconds if it were to occur in a STAR position that has just been tested. Since diagnosis of faulty PLBs within a BISTER tile can be performed based on the failing BIST results, the 1.34 second roving and testing time also includes diagnosis to a faulty PLB. On the other hand, PUB diagnosis requires a maximum of 7.2 milliseconds in the ORCA 2C15 to be able to identify a faulty LUT and/or flip-flop within the faulty PLB.

VII. Conclusion

In this paper, we presented BIST and BIST-based diagnostic methods for the roving STARs approach for on-line FPGA testing and diagnosis. The first step is to use the BIST approach to test all PLBs in the BISTER tile. Next, an analysis of the BIST

results facilitates locating most common faults - one or two - faulty PLBs - without any reconfiguration. If a unique diagnosis cannot be obtained, the suspected faulty PLBs are divided into subsets and re-tested. This combined technique achieves unique diagnosis with reaonably fast diagnosis time. The main result is the development of an on-line BIST and diagnostic method able to accurately diagnose any single faulty PLB and most combinations of multiple faulty PLBs within a BISTER tile. The combinations of multiple faulty PLBs that cannot be diagnosed appear to be very restrictive situations unlikely to occur in practice. We have applied our BIST-based diagnosis to actual faulty ORCA FPGAs and we correctly identified their known faulty PLBs. Our diagnostic technique provides the foundation necessary for implementing low-cost and efficient fault-tolerant approaches in FPGAs [23][27].

Acknowledgment

The authors wish to acknowledge Sajitha Wijesuriya, Carter Hamilton, and Brandon Skaggs of the Dept. of Electrical Engineering at the University of Kentucky as well as Thomas Slaughter of the Dept. of Electrical and Computer Engineering at the University of North Carolina at Charlotte for their contributions to BIST and BIST-based diagnosis of FPGA logic blocks during the Roving STARs project.

References

[1] W. Huang and F. Lombardi, “An approach to testing programmable/configurable field programmable gate arrays,” in *Proc. IEEE VLSI Test Symp.*, 1996, pp. 450-455.
[2] W. Huang, F. Meyer, X. Chen, and F. Lombardi, “Testing configurable LUT-based FPGAs,” *IEEE Trans. on VLSI Systems*, vol. 6, no. 2, pp. 276-283, 1998.
[3] T. Inoue, S. Miyazaki, and H. Fujiwara, “Universal fault diagnosis for lookup table FPGAs,” *IEEE Design & Test of Computers*, vol. 15, no. 1, pp. 39-44, 1998.
[4] C. Jordan and W. P. Marnane, “Incoming inspection of FPGAs,” in *Proc. European Test Conf.*, 1993, pp. 371-377.
[5] F. Lombardi, D. Ashen, X. Chen, and W. Huang, “Diagnosing programmable interconnect systems for FPGAs,” in *Proc. ACM International Symp. on FPGAs*, 1996, pp. 100-106.
[6] M. Renovell, J. Figueras, and Y. Zorian, “Test of RAM-based FPGA: methodology and application to interconnects,” in *Proc. IEEE VLSI Test Symp.*, 1997, pp. 230-237.
[7] M. Renovell, J. Portal, J. Figueras, and Y. Zorian, “Testing the interconnect of RAM-based FPGAs,” *IEEE Design & Test of Computers*, vol. 15, no. 1, pp. 45-50, 1998.
[8] M. Renovell, J. Portal, J. Figueras and Y. Zorian, “SRAM-based FPGA: testing the LUT/RAM modules,” in *Proc. IEEE International Test Conf.*, 1998, pp. 1102-1111.
[9] M. Renovell, J. Portal, J. Figueras and Y. Zorian, “SRAM-based FPGA: testing the embedded RAM modules,” *J. of Electronic Testing: Theory and Application*, vol. 14, no. 1/2, pp. 159-167, 1999.
[10] B. Culbertson, R. Amerson, R. Carter, P. Kuekes, and G. Snider, “Defect tolerance on the Teramac custom computer,” in *Proc. IEEE Symp. on Field-Programmable Custom Computing Machines*, 1997, pp. 116-123.
[11] C. Hamilton, G. Gibson, S. Wijesuriya, and C. Stroud, “Enhanced BIST-based diagnosis of FPGAs via boundary scan access,” in *Proc. IEEE VLSI Test Symp.*, 1999, pp. 413-418.
[12] C. Stroud, P. Chen, S. Konala, and M. Abramovici, “Evaluation of FPGA resources for built-in self-test of programmable logic blocks,” in *Proc. ACM International Symp. on FPGAs*, 1996, pp. 107-113.
[13] C. Stroud, S. Konala, P. Chen, and M. Abramovici, “Built-in self-test for programmable logic blocks in FPGAs,” in *Proc. IEEE VLSI Test Symp.*, 1996, pp. 387-392.
[14] C. Stroud, E. Lee, S. Konala, and M. Abramovici, “Using ILA testing for BIST in FPGAs,” in *Proc. IEEE International Test Conf.*, 1996, pp. 68-75.
[15] C. Stroud, E. Lee, and M. Abramovici, “BIST-based diagnostics for FPGA logic blocks,” in *Proc. IEEE International Test Conf.*, 1997, pp. 539-547.
[16] C. Stroud, S. Wijesuriya, C. Hamilton, and M. Abramovici, “Built-in self-test of FPGA interconnect,” in *Proc. IEEE International Test Conf.*, 1998, pp. 404-411.
[17] S.-J. Wang and T.-M. Tsai, “Test and diagnosis of faulty logic blocks in FPGAs,” in *Proc. IEEE International Conf. on Computer Aided Design*, 1997, pp. 722-727.
[18] A. Burress and P. Lala, “On-line testable logic design for FPGA implementation,” in *Proc. IEEE International Test Conf.*, 1997, pp. 471-478.
[19] C. Zeng, N. Saxena, and E. McCluskey, “Finite state machine synthesis with concurrent error detection,” in *Proc. IEEE International Test Conf.*, 1999, pp. 672-679.
[20] N. Shnidman, W. 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, 1998.
[21] A. Steininger and C. Scherrer, “On the necessity of on-line BIST in safety critical applications”, *Proc. Fault-Tolerant Computing Symp.*, 1999, pp. 208-215.
[22] M. Abramovici, C. Stroud, S. Wijesuriya, C. Hamilton, and V. Verma, “Using roving STARs for on-line testing and diagnosis of FPGAs in fault-tolerant applications,” in *Proc. International Test Conf.*, 1999, pp. 973-982.
[23] M. Abramovici, J. Emmert, and C. Stroud, “Roving STARs: an integrated approach to on-line testing, diagnosis, and fault tolerance for FPGAs in adaptive computing systems,” in *Proc. NASA/DoD Workshop on Evolvable Hardware*, 2001, pp. 73-92.
[24] M. Abramovici and C. Stroud, “BIST-Based Delay-Fault Testing in FPGAs,” in *Proc. IEEE International On-Line Test Workshop*, 2002, pp. 131-134.
[25] Lattice Semiconductor, Available: *http://www.latticesemi.com/products*
[26] D. Siewiorek and R. Swarz, *The Theory and Practice of Reliable System Design*, Bedford, Massachusetts: Digital Press, 1982.
[27] J. Emmert, C. Stroud, B. Skaggs, and M. Abramovici, “Dynamic fault tolerance in FPGAs via partial reconfiguration,” in *Proc. IEEE Symp. on Field-Programmable Custom Computing Machines*, 2000, pp. 165-174.
[28] C. Stroud, M. Lashinsky, J. Nall, J. Emmert, and M. Abramovici, “On-line BIST and diagnosis of FPGA interconnect using roving STARs,” in *Proc. IEEE International On-Line Test Workshop*, 2001, pp. 31-39.
[29] M. Abramovici and C. Stroud, “BIST-Based Test and Diagnosis of FPGA Logic Blocks,” *IEEE Trans. on VLSI Systems*, vol. 9, no. 1, pp. 159-172, 2001.
[30] C. Stroud, *A Designer’s Guide to Built-In Self-Test*, Boston: Kluwer Academic Publishers, 2002.
[31] “Standard Test Access Port and Boundary-Scan Architecture,” *IEEE Standard P1149.1-1990*, 1990.
[32] M. Abramovici, M. A. Breuer and A. D. Friedman, *Digital Systems Testing and Testable Design*, New York: IEEE Press, 1994.
[33] Xilinx, Inc., Available: *http://www.xilinx.com/products*

Miron Abramovici (S’76-M’80-SM’86-F’93) received the Ph.D. degree from the University of Southern California in 1980. He is co-founder and CTO of DAFCA (Design Automation for Flexible Chip Architectures), an EDA startup providing a reconfigurable infrastructure platform for SoCs. Previously he was a Distinguished Member of Technical Staff at Bell Labs in Murray Hill, NJ. He co-authored *Digital Systems Testing and Testable Design* (IEEE Press, 1994). He has over 70 publications and 19 issued U.S. patents. He has been a member of the editorial board of *IEEE Design & Test of Computers* and of *Journal of Electronic Testing: Theory and Applications*. He was Adjunct Professor of Computer Engineering at the Illinois Institute of Technology, Chicago, IL. He was principal investigator and project leader of the 3-year, DARPA-sponsored project on adaptive computing systems. He co-authored a paper included in the anthology "Twenty-Five Years of Electronic Design Automation."

Charles E. Stroud (S’74-M’88-SM’90) received the Ph.D. degree from the University of Illinois at Chicago in 1991. He is a Professor in the Dept. of Electrical and Computer Engineering at the Auburn University. Previously, he was a Distinguished Member of Technical Staff at Bell Labs in Naperville, IL, where he work for 15 years as a VLSI and printed circuit board designer, with additional work in CAD tool development and BIST for digital and mixed-signal VLSI. He has been a member of the editorial board of *IEEE Design & Test of Computers* and *IEEE Transactions on VLSI Systems*. He is author of *A Designer's Guide to Built-In Self-Test* (Kluwer Academic Publishers, 2002). He has over 100 publications and 13 issued U.S. patents for various BIST techniques for VLSI and FPGAs.

John M. Emmert (S’92-M’93-SM’04) received the Ph.D. degree from University of Cinncinati in 1999. He is Director of Computer Engineering and an Associate Professor in the Dept. of Electrical and Computer Engineering at UNC Charlotte. He is also an officer in the United States Air Force. While on Active Duty in the Air Force he flew Teledyne Ryan's AQM-34L Unmanned

Air Vehicles (UAVs); developed, implemented, and successfully tested a digital control system for flying multiple UAVs on a single channel; and developed computer aided design tools for mapping circuits onto ASIC and FPGA platforms. Currently he performs research in the areas of CAD tool development and mixed signal BIST for high speed, electronic warfare applications. He is also an active member of the United States Air Force Reserve.

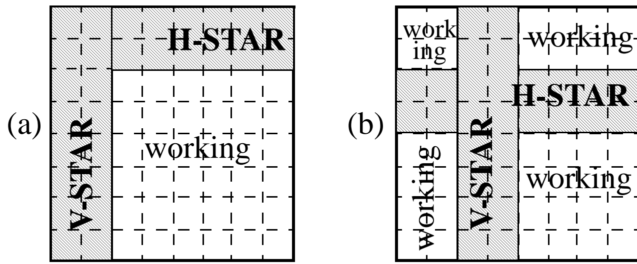

Fig.1. Roving STARs (a) in their initial position and (b) during roving.

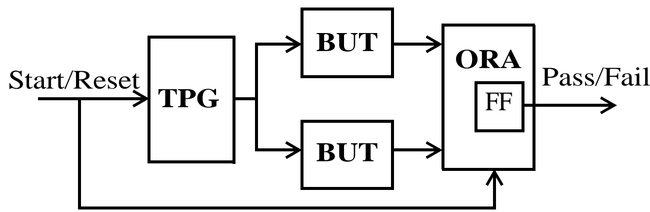

Fig.2. BISTER tile for PLBs.

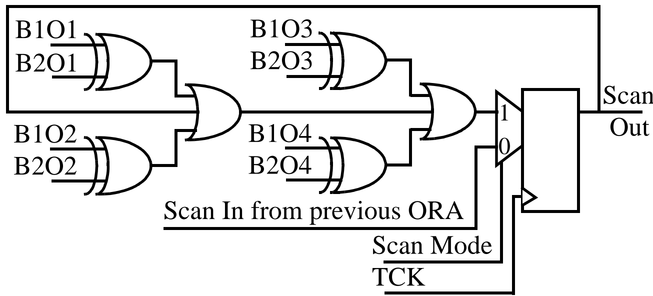

Fig.3. Integrated ORA/scan cell.

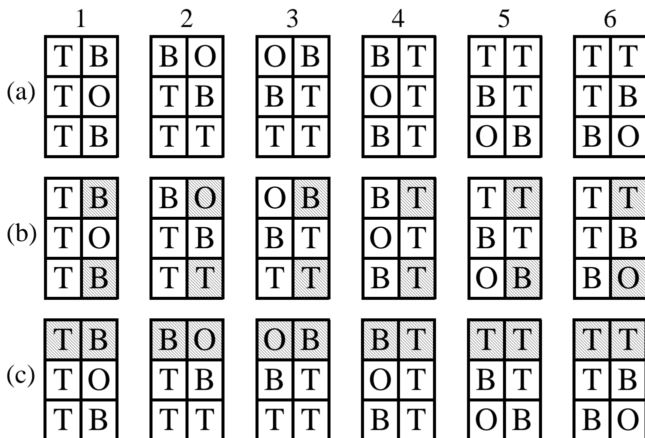

Fig.4. Rotations for a 3×2 BISTER tile (a) the 6 rotations, (b) with two faulty BUTs in one BIST configuration, and (c) one faulty BUT in one BIST configuration.

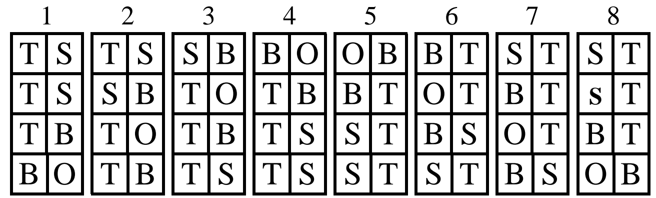

Fig.5. 4×2 BISTER tile rotations.

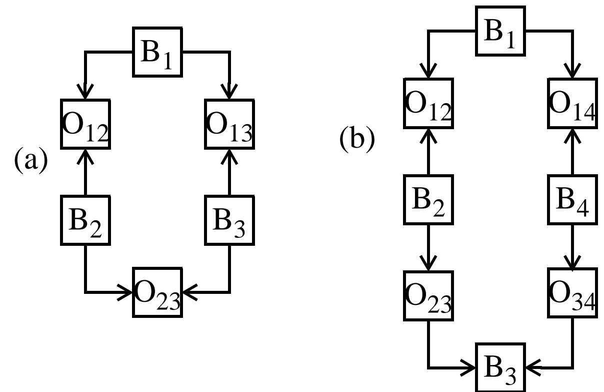

Fig.6. Combined test sessions for (a) 3×2 tile and (b) 4×2 tile.

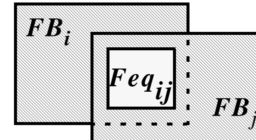

Fig.7. Graphical representation of (1).

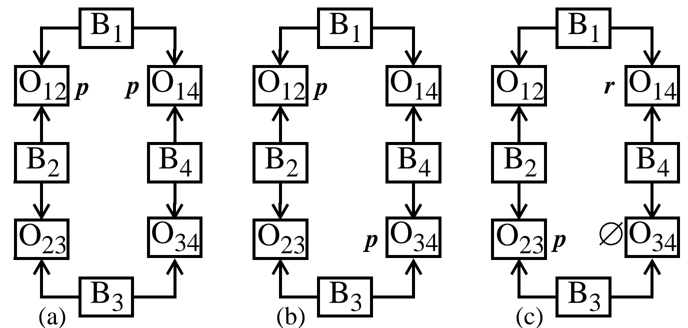

Fig.8. Graphical representions for proofs for (a) Theorem 2, (b) Lemma 7, and (c) Lemma 8.

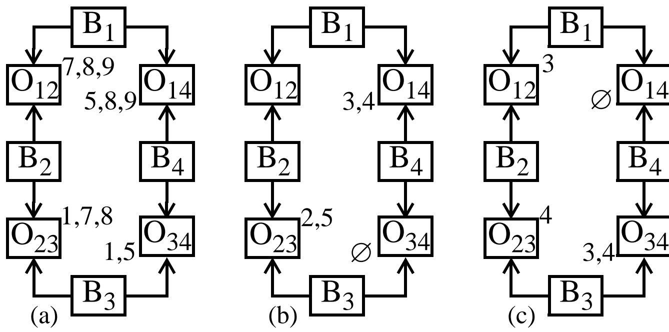

Fig.9. Graphical representations for (a) Example 1, (b) Example 3, and (c) Example 4.

phase	1	2	3	4	5	6	7	8	9
O_{14}	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**1**
B_1									
O_{12}	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**
B_2									
O_{23}	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**
B_3									
O_{34}	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**
B_4									
O_{14}	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**1**

step 1

phase	1	2	3	4	5	6	7	8	9
O_{14}	0	0	0	0	1	0	0	1	1
B_1	**0**	**0**	**0**	**0**		**0**			
O_{12}	0	0	0	0	0	0	1	1	1
B_2		**0**	**0**	**0**	**0**	**0**			
O_{23}	1	0	0	0	0	0	1	1	0
B_3		**0**	**0**	**0**		**0**			**0**
O_{34}	1	0	0	0	1	0	0	0	0
B_4		**0**	**0**	**0**		**0**	**0**		
O_{14}	0	0	0	0	1	0	0	1	1

step 2

phase	1	2	3	4	5	6	7	8	9
O_{14}	0	0	0	0	1	0	0	1	1
B_1	0	0	0	0	**0**	0	**0**		
O_{12}	0	0	0	0	0	0	1	1	1
B_2	**0**	0	0	0	0	0			**0**
O_{23}	1	0	0	0	0	0	1	1	0
B_3		0	0	0	**0**	0	**0**		0
O_{34}	1	0	0	0	1	0	0	0	0
B_4	**0**	0	0	0		0	0		**0**
O_{14}	0	0	0	0	1	0	0	1	1

step 3

phase	1	2	3	4	5	6	7	8	9
O_{14}	0	0	0	0	1	0	0	1	1
B_1	0	0	0	0	0	0	0		**1**
O_{12}	0	0	0	0	0	0	1	1	1
B_2	0	0	0	0	0	0	**1**		0
O_{23}	1	0	0	0	0	0	1	1	0
B_3	**1**	0	0	0	0	0	0		0
O_{34}	1	0	0	0	1	0	0	0	0
B_4	0	0	0	0	**1**	0	0		0
O_{14}	0	0	0	0	1	0	0	1	1

step 4

Fig.10. MULTICELLO diagnostic algorithm example.

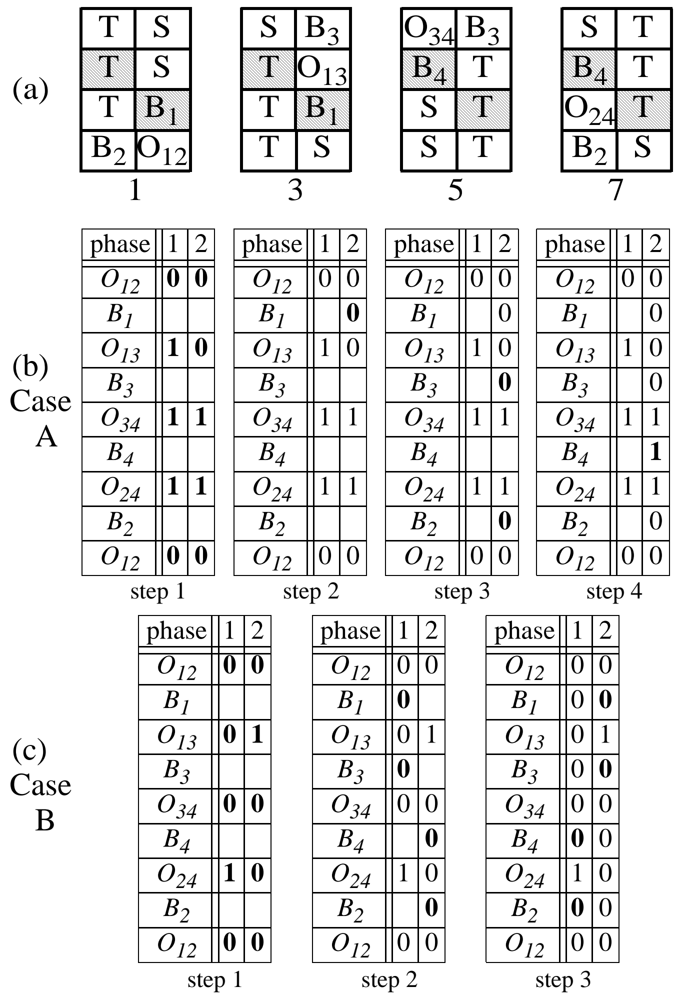

Fig.11. A 4×2 BISTER tile with two faulty PLBs and MULTICELLO diagnosis for two cases where Assumption A1 does not hold including (a) rotations of interest, (b) case A, and (c) case B.

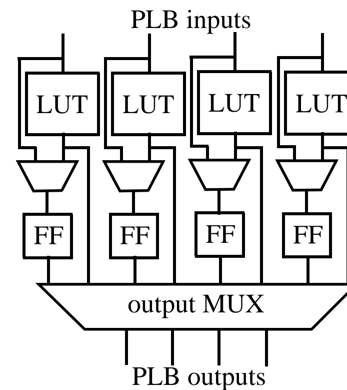

Fig.12. Basic PLB architecture.

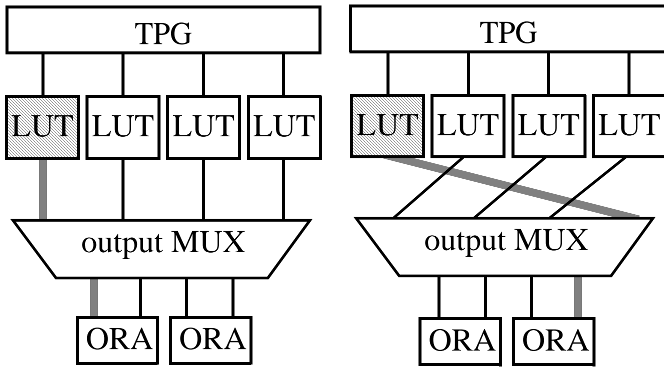

Fig.13. PUB diagnosis.

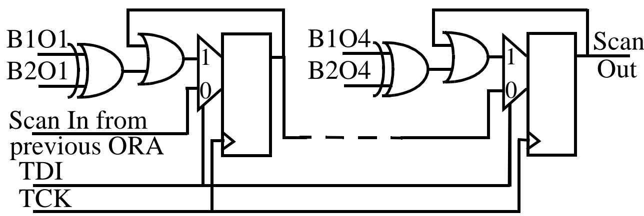

Fig.14. ORA with greater diagnostic resolution.

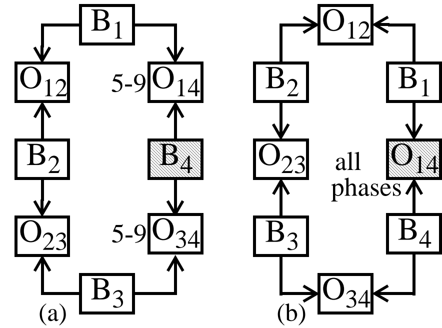

Fig.15. Diagnosis of an actual faulty PLB for (a) test session 1 and (b) test session 2.

TABLE I ACRONYMS AND ABBREVIATIONS

Acronym	Meaning
BIST	Built-In Self-Test
BISTER	group of PLBs forming a BIST unit
BUT	Block Under Test
FPGA	Field-Programmable Gate-Array
H-STAR	Horizontal STAR
LUT	Look-Up Table
ORA	Output Response Analyzer
ORCA	Optimized Reconfigurable Cell Array
PLB	Programmable Logic Block
PUB	Partially Usable Block
STAR	Self-Testing ARea
TPG	Test Pattern Generator
TREC	Test and REconfiguration Controller
V-STAR	Vertical STAR

TABLE II BIST PHASES FOR ORCA 2C AND 2CA SERIES LOGIC BLOCKS

Phase	Flip-Flop/Latch Modes and Options					LUT Mode	Number Outputs
	FF/ Latch	Set/ Reset	Clock	Clock Enable	FF Data Input		
1	-	-	-	-	-	async. RAM	4
2	-	-	-	-	-	adder/subtracter	5
3	-	-	-	-	-	5-variable MUX	4
4	-	-	-	-	-	5-variable XOR	4
5	FF	async. reset	falling edge	active low	LUT output	count up	5
6	FF	async. set	falling edge	enabled	PLB input	count up/down	5
7	Latch	sync. set	active low	active high	LUT output	count down	5
8	FF	sync. reset	rising edge	active low	PLB input	4-variable	4
9	Latch	-	active high	active low	dynamic select	4-variable	4
10	-	-	-	-	-	multiplier	5
11	-	-	-	-	-	greater/equal comparator	5
12	-	-	-	-	-	not equal com-parator	5
13	-	-	-	-	-	sync. RAM	4
14	-	-	-	-	-	dual-port RAM	4

TABLE III ADDITIONAL DIAGNOSTIC PHASES FOR ORCA 2C AND 2CA

Phase	Flip-Flop/Latch Modes & Options					LUT Mode	Logic Tested		
	FF/ Latch	Set/ Reset	Clock	Clock Enable	FF Data Input		LUT	FF	MUX
1-4	Latch	-	active high	enabled	PLB input	-			√
5-8	-	-	-	-	-	4-vari-able	√		√
9	FF	async. set	falling edge	active low	PLB input	-		√	
10	FF	async. reset	rising edge	active high	PLB input	-		√	
11	FF	sync. set	rising edge	active low	PLB input	-		√	
12	FF	sync. reset	falling edge	active high	PLB input	-		√	
13	Latch	async. set	active low	active low	PLB input	-		√	
14	Latch	async. reset	active high	active high	PLB input	-		√	
15	Latch	sync. set	active high	active low	PLB input	-		√	
16	Latch	sync. reset	active low	active high	PLB input	-		√	