BIST-Based Diagnosis of FPGA Interconnect

Charles Stroud, Jeremy Nall, Matthew Lashinsky
Dept. of Electrical and Computer Engineering
University of North Carolina at Charlotte

Miron Abramovici
Circuits and Systems Research Lab
Agere Systems, Murray Hill, NJ

Abstract: We present a Built-In Self-Test (BIST)-based diagnostic approach for the programmable interconnect resources in Field Programmable Gate Arrays (FPGAs) that can be used for either on-line or off-line testing. The technique was originally intended for on-line diagnosis of faulty interconnect to support fault-tolerant applications. However, the technique has been proven to be an excellent approach for off-line testing and diagnosis as well, providing high-resolution diagnostics with the ability to identify the faulty wire segment or programmable switch. We have implemented this BIST-based diagnostic approach on the ORCA series FPGA and present the results of testing and diagnosing known defective FPGAs.[1]

1. Introduction

In-system reprogrammable (ISR) FPGAs facilitate fault-tolerant applications since the system function can be reconfigured to avoid faulty resources. FPGAs featuring partial *run-time reconfiguration* (RTR) allow on-line testing, diagnosis, and fault-tolerance [1]. Otherwise, the FPGA can be taken out-of-service for off-line testing and diagnosis, with fault-tolerant reconfiguration taking place when the system function is reprogrammed into the FPGA after completion of testing. In either case, diagnostic techniques must be capable of identifying the faulty resource with sufficient resolution to facilitate reconfiguration around the fault. High diagnostic resolution provides for enhanced fault-tolerant procedures with fewer spare resources required to successfully avoid the fault(s) [2].

Pioneering work in FPGA interconnect testing [3][4][5] relied on externally applied vectors and is applicable only for device-level manufacturing test. Most of this prior work considered only the “global” routing that can be tested as “pure wires” oblivious to the presence of programmable logic blocks (PLBs) [3][5]. The exception was [4], where the “local” routing resources bringing signals into and out of PLBs were also tested with externally applied vectors. We have previously addressed off-line BIST for FPGA interconnect [6] which completely tested all global and local routing, but in that approach we were unable to diagnose faults. This same off-line routing BIST approach was later used in [7] for initial development of an off-line approach to diagnosis of FPGA interconnect faults [8]. Another off-line approach for BIST of FPGA interconnect resources focuses on fault detection rather than diagnosis [9]. Diagnosis of the PLBs in FPGAs has received considerable attention for both off-line testing [10] and on-line testing [11] but there has been little work in the BIST-based diagnosis of FPGA interconnect resources.

In this paper, we present the first BIST-based approach for FPGA interconnect that can be used for either on-line or off-line testing and diagnosis. Therefore, in addition to on-line testing and diagnosis for fault-tolerance, it can also be used off-line for failure mode analysis (FMA) for manufacturing yield enhancement, or for fault recovery in systems that use FPGAs that do not support RTR. The ORCA series FPGA [12] was used for our initial implementation, but the technique can be applied to any FPGA that features RTR and/or ISR capabilities such as Xilinx series FPGAs [13]. The remainder of this paper is organized as follows. We begin with brief reviews of FPGA interconnect structures, fault models, and prior work in BIST for FPGA interconnect in Section 2. In Section 3, we present an overview of the roving STARs approach which includes our BIST architecture and configurations for interconnect testing. In Section 4, we describe the extension of the on-line routing BIST approach to off-line testing and the resulting diagnostic resolution that can be obtained with BIST alone. In Section 5, we describe diagnosis of interconnect faults with the ability to identify the faulty wire segment or programmable switch as well as the type of fault. In Section 6, we discuss diagnostic results obtained with known defective FPGAs and summarize in Section 7.

2. Faults in FPGA Interconnect

The programmable interconnect network consists of wire segments that can be connected or disconnected by *configurable interconnect points* (CIPs) (also referred to as *programmable interconnect points,* or PIPs). The basic CIP structure consists of a transmission gate controlled by a configuration memory bit (Figure 1a). There are three basic types of CIPs referred to as the *cross-point CIP* (Figure 1b), the *break-point CIP* (Figure 1c), and the *multiplexer* (*MUX) CIP* (Figure 1d) [12]. While a cross-point CIP connects wire segments located in disjoint planes (a horizontal segment with a vertical one), a break-point CIP connects two wire segments in the same plane. The MUX CIP comes in two varieties: decoded and non-decoded. A decoded MUX CIP is a group of 2^k cross-point CIPs sharing a common output wire and controlled by k configuration bits, such that the input wire being addressed by the configuration bits is connected to the output wire (in Figure 1d, k=1); the decoding logic is incorporated between the configuration bits and the transmission gates. A non-decoded MUX CIP contains a configuration bit for each transmission gate, such that k wire segments are con-

1. This material is based upon work supported by the DARPA ACS program under contract F33615-98-C-1318.

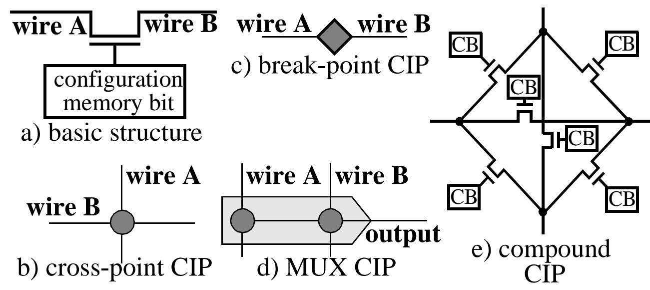


Figure 1. Configurable Interconnect Points

trolled by k configuration bits (in Figure 1d, k=2); only one of the configuration bits may be active for a given configuration. Wire segments in the programmable interconnect network are bounded by these CIPs. There is also a *compound CIP* (Figure 1e), which is a combination of four cross-point and two break-point CIPs, each separately controlled by a configuration bit [13]. Most recent FPGA interconnect architectures are primarily constructed from non-decoded MUX CIPs that are buffered to prevent signal degradation due to the series resistance of each transmission gate the signals pass through.

Wire segments within the FPGA comprise either global or local routing resources. Global routing resources are used to interconnect non-adjacent PLBs and I/O cells, while local routing resources are specific to a given PLB and used to connect that PLB to global routing resources or to adjacent PLBs. Figure 2 illustrates a simplified view of the routing busses associated with a single PLB in an ORCA 2C series FPGA [12]. Horizontal and vertical busses are prefixed *h* and *v*, respectively. The suffixes *x1*, *x4*, *xH*, and *xL* indicate wire segments that extend across 1 PLB, 4 PLBs, half the PLB array, and the full length of PLB array, respectively, before encountering a break-point CIP. The *direct* busses provide connections between adjacent PLBs and the four direct busses are denoted as *dn*, *ds*, *de*, and *dw* for directions north, south, east, and west, respectively. The local routing associated with each PLB typically includes multiplexer CIPs at the inputs to the PLB and bi-directional buffers with connections to the vertical and horizontal global and direct busses

The Xilinx FPGA routing architectures are similar to Figure 2 in many respects [13]. For example, the Xilinx 4000 series FPGA has *x1, x2, xH, xL,* and direct busses. Xilinx FPGAs primarily use arrays of compound cross-point CIPs

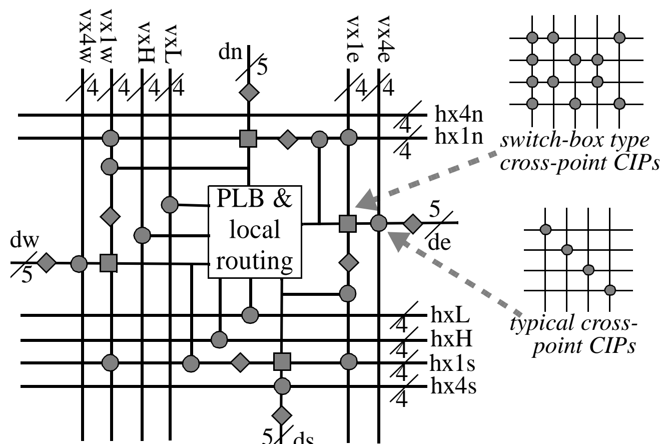


Figure 2. Single PLB view of routing resources

and MUX CIPs, with fewer cross-point and break-point CIPs. The main difference between the MUX CIPs in Xilinx and ORCA FPGAs is that the Xilinx MUX CIPs access any segment in a bus adjacent to the PLB in one direction (either north, south, east, *or* west), while ORCA 2C MUX CIPs access a subset of the segments in each north, south, east, *and* west bus. Xilinx FPGAs also incorporate bi-directional buffers for high fanout or long distance signals in the FPGA. Because of the similarity between the two interconnect architectures, the BIST approach we describe in this paper can be applied to either FPGA, as well as any FPGA with a similar interconnect structure.

2.1 Fault Models and Detection

The fault models typically considered for FPGA interconnect testing include: CIPs stuck-closed (stuck-on) and stuck-open (stuck-off), wires stuck at 0 or 1, open wires, and shorted wires (bridging faults) [5][6]. Detecting the CIP faults also detects stuck-at faults in the configuration memory bits that control the CIPs. A stuck-closed CIP also creates a bridging fault between its two wire segments or between the Source and Drain of the transmission gate transistor. While only wired-AND/wired-OR bridging fault models are sometimes considered as possible behavior for shorts between wire segments [9], detecting all dominant bridging fault models ensures that all wired-AND/wired-OR bridging faults are detected as well [14].

Usually, there is no detailed layout information available regarding adjacency relations between wire segments. Therefore, we use only the “rough” physical data (available in data books and in the physical design graphical editors available in the FPGA CAD tool suite) to determine the adjacency of groups of wires, where each group of wires *may* have pairwise shorts, but not every wire is necessarily adjacent with every other wire in the group. This treatment makes our method relatively layout-independent, and allows us to ignore bus rotations, which change the adjacency relations among the wires of the same bunch.

To detect these fault models, the applied tests must check that every wire segment and CIP is able to transmit both a 0 and a 1, and that every pair of wire segments that may be shorted can transmit both (0,1) and (1,0) values. A test that applies 0 and 1 values at one end of a wire and expects the same values at the other end, detects any stuck-open fault in any type of closed CIP (cross-point, break-point, or multiplexer) along the wire, and also detects any open or stuck-at fault affecting any wire segment. Detecting a stuck-closed CIP fault requires the application of opposite logic values to the wire segments associated with that CIP, so that the fault will yield an incorrect logic value on one of the two segments. Thus both combinations of values (01 and 10) must eventually be applied to the two wire segments separated by the open CIP. Similarly, opposite logic values must be applied to any pair of adjacent wire segments that have the potential for a bridging fault, so that the fault will yield an incorrect logic value on one of the two wire segments.

A MUX CIP requires one test configuration for each of its inputs. Both logic 0 and 1 are applied to the selected input, while opposite values must be applied to the non-selected inputs. This test detects the stuck-open fault in the closed CIP

that connects the selected input to the output wire, and the stuck-closed faults in the open CIPs corresponding to the unselected inputs. In most previous work, the non-selected inputs are ignored, because the model used by the FPGA CAD tools that generate the configuration bitstream never includes the functionally inactive subcircuits. But ignoring the "invisible logic" results in incomplete testing of the MUX [15]. The solution relies on separately configuring the invisible logic so that it will generate the proper values needed for the inactive MUX inputs, followed by "overlaying" the resulting configuration files over the main configuration file for the active logic.

2.2 Prior Work in Interconnect BIST

The strategy of the first FPGA interconnect BIST approach [6] was to configure a subset of routing resources (wire segments and CIPs) to form two groups of *wires under test* (WUTs). These WUTs received identical test patterns from a *test pattern generator* (TPG), while the values at the other end of the WUTs were compared by one or more *output response analyzers* (ORAs). Figure 3 illustrates the basic BIST architecture for FPGA interconnect testing. In some test configurations there were many of these BIST structures operating concurrently. A WUT may be composed of several wire segments connected by closed CIPs. To check local interconnect, WUTs may also go through PLBs that pass their inputs directly to their output). In Figure 3, the WUTs are shown by bold lines, and the activated (closed) CIPs are shown in gray, while the open CIPs are white. The first group of WUTs connects the segments *P, Q, R,* and *S*, while the second group connects *X, Y, Z, V, U*, and *W*. Since *Y* and *V* connect to inputs of PLBs, these PLBs are configured as identity functions. Note that without involving PLBs in configuring WUTs, one cannot test local routing and achieve a complete interconnect test.

The complete BIST session consists of a sequence of *test phases*, where each test phase requires the following steps: 1) reconfiguring the FPGA to create one or more BIST structures, 2) initiating the BIST sequence, and 3) reading the Pass/Fail results at the end of the BIST sequence. To minimize the number of reconfigurations used for testing the interconnect resources within a BIST structure, as many WUTs as possible were concurrently tested. A total of 27 test phases were required to completely test the global and local interconnect resources in an ORCA 2C series FPGA [6].

The TPG applied exhaustive patterns to the WUTs, that is, all possible 2^n test vectors to every group of n WUTs [6]. Although this test is longer than walking patterns (which also provide a complete set of tests [3]), the exhaustive patterns can be generated using fewer PLBs, and for reasonably small

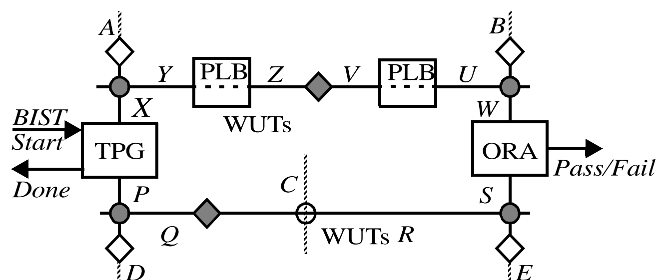


Figure 3. First BIST architecture for interconnect

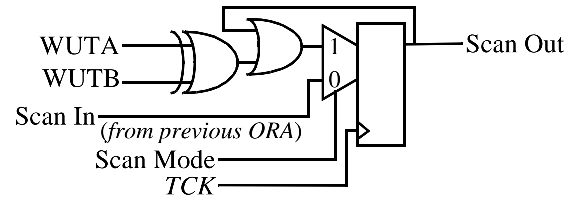


Figure 4. Integrated ORA and scan register

n, their application time is still negligible compared to the reconfiguration time. The open CIPs that isolate the WUTs from the rest of the interconnect must be tested for stuck-closed faults. To achieve this, the TPG controls any wire that may become shorted to some WUT (*A*, *B*, *C*, *D*, and *E* in Figure 3) such that when the TPG drives a 0(1) on the WUTs, it also sets *A, B*, *C*, *D*, and *E* to 1(0) at least once during the test.

An iterative, comparison-based ORA was used to latch any mismatches observed at the destination end of the WUTs [6]. This provided a single *Pass/Fail* output for the entire FPGA and, hence provided little, if any, diagnostic information. Another potential problem with the comparison-based ORA was that equivalent faults in two sets of WUTs (for example, each group having a short between its third and fourth wires) being compared by the ORA would escape detection [7]. This problem can be overcome by testing every set of WUTs at least two times, each time being compared with a different set of WUTs.

This first interconnect BIST approach reported in [6] was later adopted in [7] for diagnosis based on the BIST results [8]. However, the diagnostic resolution was limited to a given set of WUTs and far from the diagnostic resolution required for efficient fault-tolerant applications. Improved diagnostic resolution developed for BIST of PLBs using an integrated ORA and scan chain [15] (similar to that illustrated in Figure 4) could not be applied to the interconnect BIST approach in [6] due to the excessive routing congestion.

An alternative BIST approach is to calculate parity over the set of WUTs and to transmit the parity over a separate wire to the ORA where it is compared to the parity that is regenerated for the set of WUTs at the ORA, as illustrated in Figure 5 [9]. The TPG consists of an N-bit binary counter and an N-bit parity generate circuit. The ORA consists of an N-bit parity regenerate and check circuit with an RS latch. This approach is capable of detecting any single fault in the interconnect network as well as any multiple faults that produce an odd number of bit errors in the WUTs as they enter the ORA. This approach helps to alleviate some of the routing congestion problems encountered when developing interconnect BIST configurations. However, diagnostic resolution is limited to the set of WUTs, which incorporates a large subset of the routing resources in the FPGA.

Regardless of whether the BIST approach is comparison-based or parity-based, multiple groupings of TPG, WUTs,

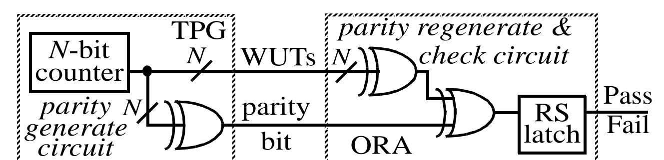


Figure 5. Parity-based BIST architecture

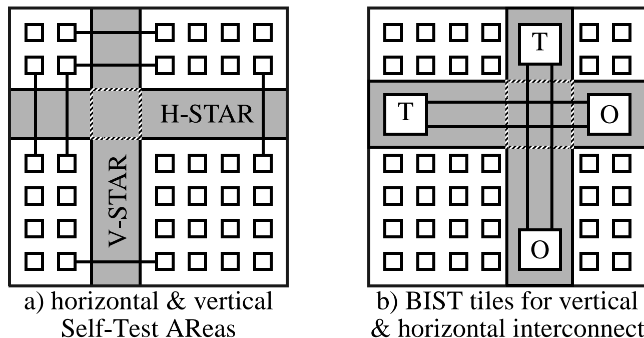
and ORA must be executed in parallel within the FPGA in order to minimize the total number of BIST configurations and, as a result, the total testing time. Diagnosis of faulty interconnect resources can require many more test phases with higher diagnostic resolution requiring more test phases. The interconnect diagnostic problem is further complicated by the fact that WUTs, composed of many wire segments and CIP, are not fixed entities as is the case with PLBs [15].

3. Roving STARs Overview

The roving STARs approach [1] proposed new techniques for on-line FPGA testing, diagnosis, and fault-tolerance, applicable to any FPGA supporting incremental RTR via its Boundary Scan interface [16]. A STAR (self-testing area) is a temporarily off-line section of the FPGA where self-testing occurs without disturbing the normal system activity in the rest of the chip. Roving the STARs periodically brings every section of the FPGA under test and guarantees complete testing of the FPGA, including all its spare resources. To efficiently tolerate faults, testing must first provide high-resolution diagnosis in order to accurately locate the faulty resources. Once located, the faulty resources can be avoided or may be reused [2]. As a simple example of reuse of faulty interconnect resources, one segment in a pair of shorted wire segments can still be used by the system function as long as the other segment is not used by the system function.

Figure 6a depicts an FPGA with a vertical STAR (V-STAR) and a horizontal STAR (H-STAR); the system application resides in the working areas with the white blocks representing PLBs performing the system function. After testing within a STAR has been completed, the STAR roves to a new location, by exchanging places with an equal-size slice of the working area. Roving the STARs across the FPGA is implemented by a sequence of precomputed partial reconfigurations and assures that the entire FPGA will be eventually tested. The roving process and the use of roving STARs for test and diagnosis of PLBs are described in detail in [1] and [10].

Depending on the positions of the two STARs, the working area may be contiguous, or it may be divided into two or four disjoint regions (see Figure 6a). All horizontal wire segments in the H-STAR and all vertical segments in the V-STAR are reserved for testing. Only the connections among the working programmable logic blocks (PLBs) and the connections with the I/O blocks of the FPGA are allowed to go through the STARs, using horizontal wire segments through the V-STAR and vertical segments through the H-STAR (see Figure 6a). While either STAR can be used to test all PLBs, both STARs are required for testing the programmable interconnect; the H-STAR is used to test horizontal resources and the V-STAR to test vertical routing resources as illustrated in Figure 6b.



a) horizontal & vertical Self-Test AReas
b) BIST tiles for vertical & horizontal interconnect

Figure 6. FPGA with STARs and BIST structures

Testing the cross-point CIPs connecting global horizontal and vertical routing resources must involve both STARs, as illustrated in Figure 7. Having interconnect testing done in both the V-STAR and H-STAR provides additional advantages. Since the TPGs and ORAs are typically located at the top and bottom of the V-STAR and left and right sides of the H-STAR, testing the local interconnect resources in the PLBs around the perimeter of the FPGA becomes more difficult because the interconnect resources of these PLBs are primarily used to establish the connections between the TPG/ORA and WUTs. Similar to global cross-point CIP testing, the TPG can be located in one STAR while the ORA is in the other STAR to facilitate complete testing of the local interconnect resources in the perimeter PLBs as illustrated in Figure 8a. Another important situation occurs when there are logic faults in the PLBs around the perimeter of the FPGA that would prevent proper operation of these PLBs as TPGs or ORAs. In this case, a known good PLB can be chosen from the other STAR to facilitate fault-free TPG or ORA operation as illustrated in Figure 8b and Figure 8c, respectively.

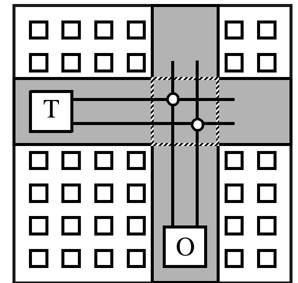
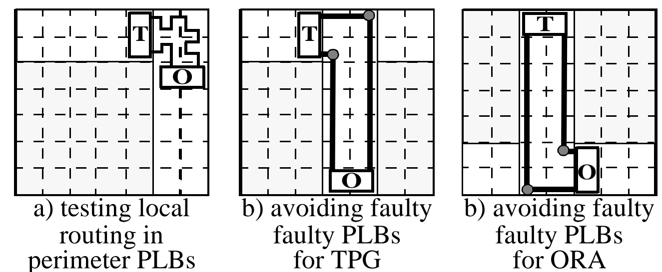


Figure 7. BIST for global cross-point CIPs

While the general approach to on-line testing the interconnect was proposed in [1], it had not been implemented at that time. The subsequent implementation of the on-line interconnect BIST configurations was presented in [17]. The set of BIST configurations (test phases) for an ORCA 2C15 is summarized in Table 1 and organized as a sequence of five *test sessions*, where a test session is a group of test phases targeting similar types of faults [17]. The number of test sessions and phases will vary depending on the architecture of the interconnect network and the specific arrangements of CIPs and bus structures.

4. Extending STARs to Off-Line Testing

During off-line testing (either at the manufacturing or the system level), there is no need for the system function to be programmed into the FPGA. Therefore, the entire FPGA can programmed with V-STARs to test all vertical routing



a) testing local routing in perimeter PLBs
b) avoiding faulty faulty PLBs for TPG
b) avoiding faulty faulty PLBs for ORA

Figure 8. Using both STARs for special testing

Table 1: Summary of interconnect BIST test sessions [17]

Test Session	Target Faults	V-STAR Phases	H-STAR Phases
1	shorts, opens, and some CIP faults in global busses	7	7
2	CIP faults in x1 busses, shorts & opens in local routing	4	4
3	faults in cross-point CIPs between global & local busses	3	3
4	multiplexer CIP faults	7	0
5	faults in cross-point CIPs between global busses	6	

resources in parallel or with H-STARs to test all horizontal routing resources in parallel as illustrated in Figure 9a. These STARs operate in parallel to reduce the number of times the FPGA must be reconfigured and, in turn, the total testing and diagnostic run-time. We refer to these parallel STARs as a "galaxy". The same BIST test sessions and test phases for horizontal and vertical routing resources given in Table 1 can be used in the STARs of the galaxy BIST approach. The exception is in the case of the global-to-global cross-point CIP test phases where the multiple STARs are populated with additional ORAs (as shown in Figure 9b) to minimize the total time required for complete testing. The interconnect BIST architecture used within the STARs for both on-line and off-line approaches is the same as that illustrated in Figure 3. The comparison-based, integrated ORA and scan chain illustrated in Figure 4 is used to capture and retrieve BIST results via the Boundary Scan interface.

Compared to the original off-line BIST for FPGA interconnect which required a total of 27 test phases [6], the galaxy BIST approach requires 41 test phases [17], or about 50% more test phases, to completely test the core interconnect resources of an ORCA 2C series FPGA. This is due to the fact that the BIST structure is confined within the STAR which, in turn, limits the amount of interconnect resources that can be tested in a given test phase. A by-product of this confinement is that the number of wire segments and CIPs comprising a set of WUTs is reduced such that the BIST sequence can be applied at a much higher clock rate. For example, timing analysis reveals that the galaxy BIST for the ORCA 2C15 FPGA can run over 10 times faster than the original off-line interconnect BIST approach in [6]; 23.15 MHz for galaxy BIST compared to 2.29 MHz for the original routing BIST. More importantly, the galaxy BIST can easily scale to test larger FPGAs, while we observed during the development of the original off-line routing BIST that there is a limit to the size of FPGA that the approach could handle. This limit is due to the number of series CIPs (around 100) that a test pattern can pass through before the signal quality is degraded to the extent that reliable BIST results cannot be obtained.

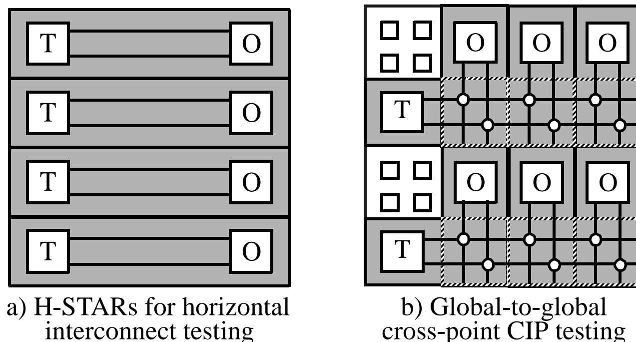
The 41 BIST configurations for the ORCA 2C15 tests over 100,000 CIPs and over 25,000 wire segments in the core of the FPGA. Each configuration is approximately 225K-bits requiring a total of about 1.15 Mbyte of memory to store the BIST configurations for system level testing. Application of the BIST configurations through a 20MHz Boundary Scan interface requires approximately 0.46 seconds. This includes downloading each 225K-bit configuration, executing the BIST sequence (about 500 clock cycles), and retrieving the BIST results from the ORAs (up to 380 clock cycles).

Another difference between the two approaches is the diagnostic resolution that can be obtained directly from the BIST results. In the original off-line routing BIST approach we could only surmise information regarding the possible type of interconnect resources that were faulty based on the failing test phases but we could not determine the location of the fault. This is illustrated in Table 2 where the failing test phases (denoted by an 'X') are given for two faulty ORCA 2C15 devices that contained interconnect faults [6]. From these results, it appears that Chip 1 has more faults than Chip 2 as a result of more failing test phases.

Table 2. Failing test phases for faulty ORCA 2C15s [6]

Session	1								3					4			
Phase	1	2	3	4	5	6	7	8	1	2	3	4	5	1	2	3	4
Chip 1							X	X	X		X		X		X		
Chip 2					X						X					X	X

The galaxy BIST approach, on the other hand, provides diagnostic resolution to within the failing STAR and in many test phases to an area less than a STAR as a result of multiple BIST structures operating concurrently within each STAR. The galaxy BIST results for the same two chips from Table 2 are shown in Figure 10 in terms of the failing STAR positions shown in gray. From the BIST results, we can deduce that there is at least one fault in Chip 1 and, if there is only one fault, it must be located at the intersection of the failing H-STAR and V-STAR positions. Chip 2, which appeared to have fewer interconnect faults based on the results in Table 2, has at least two interconnect faults. If there are only two faults in Chip 2, at least one fault must be located at the intersection



a) H-STARs for horizontal interconnect testing
b) Global-to-global cross-point CIP testing

Figure 9. Off-line "galaxy" BIST for interconnect

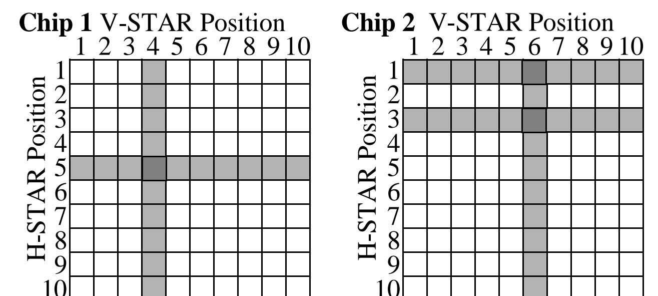


Figure 10. Galaxy BIST results for faulty ORCA 2C15s

of the failing V-STAR position and one of the two failing H-STAR positions. This translates to a diagnostic resolution between 1% to 10% of the total FPGA area, depending on the type of fault [17]. While this diagnostic resolution may be sufficient for FMA for manufacturing yield enhancement, the next section presents additional techniques to increase the resolution.

5. BIST-Based Diagnosis

Once a fault is detected, the defect must be located with sufficient resolution to facilitate either efficient reconfiguration to bypass the fault, or the reuse of the defective resource for fault-tolerant applications [2]. To be able to re-use the faulty interconnect resource, one must obtain sufficient information about the fault, including location and type, to determine whether the system configuration to be programmed into the FPGA is compatible with the fault; in other words, the faulty resource will not adversely affect the operation of the system configuration. In this section we discuss the diagnostic techniques that will provide the information about the fault necessary to make such a determination. These techniques can be used in both the on-line roving STARs or the off-line galaxy BIST approaches.

5.1 Two-Testing Analysis

Since the ORA is a comparator, an equivalent fault in two WUTs being compared would be masked and thus not detected. Therefore, every WUT is compared against two other WUTs in the interconnect fault detection configurations. This is referred to as two-testing. This additional information can be used for diagnosis. Four sets of WUTs are being compared by two ORAs, as is shown in Figure 11a. The fault is on the set of WUTs marked with an ‘X’ and a shaded ORA denotes that a failure has been recorded. When the one set of WUTs is swapped between TPGs, the other ORA records a failure, allowing the faulty set of WUTs to be identified from the original two. If the ORA failure indication had stayed with the original ORA, the set of WUTs to the left of the left-hand BIST structure would have been suspected of being faulty. However, in order for the two center sets of WUTs to swap pairs, they must use portions of different interconnect resources to attach to different TPGs and ORAs. If, for instance, the fault is in the local routing around the left-most TPG and this routing is now used to drive the other set of WUTs, the same ORA fails. This incorrectly makes it look like the left-most set of WUTs contains the fault using this analysis. Another possibility is that the fault could be totally avoided when the WUTs are rerouted causing both ORAs to pass. This is the problem with all diagnostic configurations: in order to get new information changes must be made to the WUTs; however, new WUTs may behave differently. Therefore, it is important to gather information from several techniques and to analyze this data together to remove any inconsistencies that may be caused by situations like these.

5.2 Adding ORAs and Flipping Configurations

The architectures of some of the fault detection test sessions and phases lend themselves to the addition of multiple ORAs along the sets of WUTs. These configurations may also be ‘flipped’ so that the TPG is now on the opposite end of the STAR as illustrated in Figure 11b. This can provide valuable information in identifying opens and, to a lesser

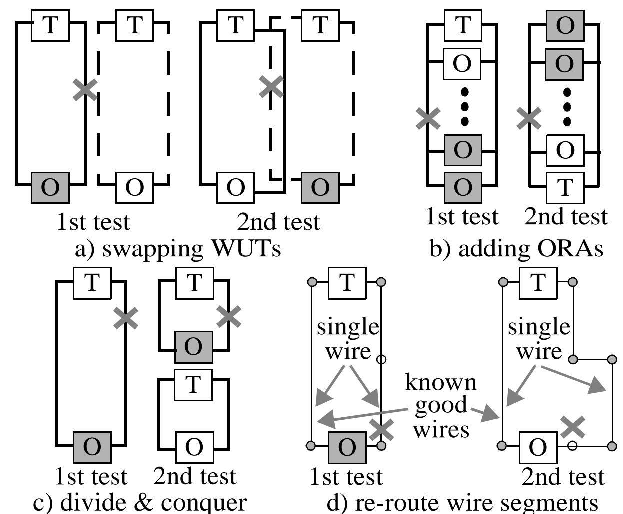


Figure 11. Diagnostic configurations

degree, shorts. With these two types of configurations up to two opens can be identified on a set of WUTs. The architecture of Session 1 (in Table 1) test phases allows for the additional ORA and flipping strategy. During off-line testing, this multiple ORA approach to diagnosis can also be applied to Session 5, as was shown in Figure 9b. The Session 3 phases already have multiple ORAs but may also be flipped to provide additional diagnostic information.

5.3 Divide-and-Conquer

Another good architecture for narrowing down the fault area and for separating the effects of multiple faults is dividing the BIST structure to create smaller BIST tiles. A smaller BIST tile maintains the original pattern of the configuration but spans fewer PLBs. The BIST tiles can then be run in parallel to increase the diagnostic resolution and/or to find multiple faults as shown in Figure 11c. This divide-and-conquer diagnostic method can be repeated to obtain higher diagnostic resolution. Since this will not test the CIPs that formerly connected the two halves of the sub-divided sets of WUTs, the BIST structure must “slide” to a new position in order to test these CIPs.

5.4 Scan ORAs

The comparator ORAs utilized by the interconnect fault detection configurations only return a pass/fail result for every WUT pair being compared. There is no way to tell which test patterns failed, which can be very useful information for determining shorts. The scan ORA was created to address this problem. This ORA captures the response to individual TPG patterns and allows the response to be scanned out through the Boundary Scan interface. The LUTs are programmed to simply pass the signals unchanged to the flip-flops which then latch the logic values as observed at the ORA. Multiple scan ORAs can also be chained together. Hence these scan ORAs can replace comparator ORAs in every diagnostic test configuration. Two scan ORA configurations, however, must be downloaded for each standard configuration since an ORA flip-flop used to latch the result from a pair of nets can now only return the value on one net. Therefore, only one WUT is examined at a time.

All patterns can be scanned out or just the patterns of

interest (for instance, all 1's, all 0's, a walking 1, and a walking 0). The number of clock cycles from the beginning of the BIST sequence to the specific test pattern of interest is known as a result of the counter-based TPG. As a result, the test clock can be cycled until the pattern of interest is captured. Then the ORA can be switched from capture mode to scan mode to scan out this pattern. The TPG counter continues to run while in scan mode but, if the counter passes over the next desired pattern during this time, the TPG can simply be rolled over to reach this test pattern again. This is not a problem as long as the TPG counter is not terribly large (we typically use up to 8 bits). Another option is to pause the TPG if clock enables are available on the flip-flops.

The ability to see which patterns fail and how they are observed at the ORA provides important diagnostic information. Two other strategies could also provide this information if comparison-based ORAs are used for this level of diagnosis instead of the scan ORAs. First, the set of WUTs could be reduced to a single wire under test; however this would require more diagnostic configurations to be downloaded than the scan ORA approach. An alternative to identifying the faulty wire from the set of WUTs is to rotate the wires being compared in the suspect set of WUTs, while keeping the test pattern assignments constant on the known good set of WUTs. This requires independent ORAs for each pair of wires being compared, but aids in obtaining good diagnostic resolution without significantly increasing the number of diagnostic test phases.

The patterns that fail can also distinguish between nets shorted together or nets stuck at constant values (which are caused by either opens or by shorts to power or ground). If wires are shorted together these pairs can be identified. This diagnosis can be performed by observing only the all 1's, all 0's, walking 1, and walking 0 patterns. If a net appears at the same logic level for all patterns then that net is either open or shorted to a power supply voltage. On the other hand, if a net fails only a walking pattern this will identify a pair of shorted wires. For instance if the pattern 1000 is expected but the pattern 1001 is observed then the first and last net can be identified as being shorted together.

5.5 Fault Dictionaries

A fault dictionary contains the fault signature for each failure mode or fault. In other words, given the failing interconnect fault detection configurations and their signatures (i.e. which ORAs recorded failures) what faults could cause these results? The dictionary can either be precompiled or generated as needed.

Two types of fault dictionaries can be utilized to aide in diagnostics. The first dictionary is related very closely to the two-testing analysis. Because all of the wire segments of the global busses are normally tested in the same test session and make up the majority of wire segments and CIPs in a set of WUTs, a fault dictionary for global busses can help in locating faults in these resources. Here each bus is treated as a single entity. The configurations and ORAs that test each bus are associated with that bus in the fault dictionary. Additional diagnostic techniques can then be used to verify these results and to identify the fault(s) specifically on this bus.

The other fault dictionary is a specific database of every resource currently under test for each phase. Assuming a single fault, the intersection of all failing interconnect detection configurations could quickly narrow the number of possible faulty resources. If any inconsistencies result from this intersection the single fault assumption can be abandoned until other techniques help isolate these fault areas. In any case, other diagnostic strategies will be needed, as always, to further verify and identify all faults.

5.6 Single Wires

As a final step, single wires can be manipulated around a fault to diagnose to the highest level of resolution. A diagram of this is shown in Figure 11d. Specific nets in the set of the WUTs known to contain no faults can be removed at this point to provide greater flexibility in routing around the suspected fault. We are left with a single WUT containing the fault and a known good WUT for comparison. As long as the faulty resource is contained in the WUT, we continue to obtain failure indications from the ORA. When the faulty resource is no longer contained in the WUT, we obtain a passing indication from the ORA.

One example of this approach provides a powerful technique for locating a short, which we refer to as *progressive net deletion*. This technique involves deleting portions of a net believed to be shorted to another net. Deleting, or un-routing, a portion of a net means to turn off a CIP so that the signal is no longer driven on this wire segment. This technique can pinpoint a short down to the length of the wire segment or, in other words, the distance between CIPs.

Figure 12 illustrates this technique where the three nets represent three single nets of the original WUTs. The right-hand two nets have been identified as being shorted together with the left-most net being dominant (which could be determined with scan ORAs). The failing test pattern as produced by the TPG and as observed at the ORA is also shown for each net. The ORA has been programmed to compare only the two outermost nets. The second test shows that when the first CIP on the dominant net is deleted the ORA still returns a failure indication. Therefore, the next CIP on the dominant net is turned off and the ORA then returns a pass result as shown in the third test. Thus, the deleted wire segment between the two CIPs can be identified as having a short.

5.7 Diagnostic Resolution

To illustrate the diagnostic resolution that can be obtained with these techniques, we will start by assuming only a single fault in a typical configuration of interconnect resources illustrated in Figure 13a. In this example, wire segment X is bounded by break-point CIPs that control the connections to segments Y and Z, while segment W is independent from

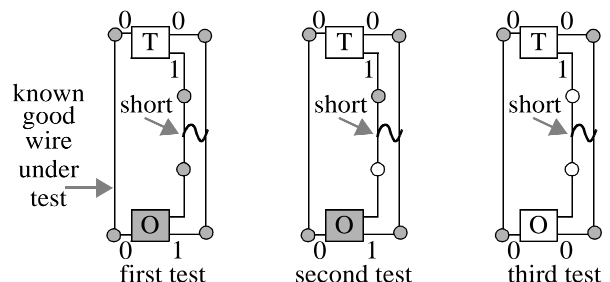


Figure 12. Diagnosis with progressive net deletion

(shares no CIPs with) segment *X*. The diagnosis depends on the type of fault and the physical location of the CIPs associated with a given wire segment. For example, if a cross-point CIP is stuck-open as illustrated in Figure 13b, then we can determine that the two wire segments associated with that CIP (including all other CIPs associated with the two wire segments) are still working and we can deduce that the CIP will not turn on. Similarly, if a wire segment has an open as illustrated in Figure 13c and we can access cross-point CIPs on either side of that open then we can determine that the segment is open between those CIPs. In these cases, diagnostic resolution is to a CIP or a segment. However, we cannot distinguish between an open in a segment and a break-point CIP stuck-open if the open in the segment is right next to the break-point CIP as illustrated in Figure 13d (but then these are equivalent faults since they behave identically and it doesn't matter where the fault actually resides). Similarly, a short between two wire segments separated by a CIP cannot be distinguished from that CIP being stuck-closed as illustrated in Figure 13d (but again we have equivalent faults). On the other hand, shorts between two wire segments that do not share a CIP, such as segments *W* and *X* illustrated in Figure 13e, can be diagnosed by seeing that each segment is fully functional while the other segment it is shorted to is unused, but when both segments are being tested then we see the failure occur and can determine that the fault is a short between these two segments. These examples have been for single faults in a given wire segment (or two wire segments in the case of shorts or CIPs stuck-closed).

For multiple faults (shorts, opens, CIPs stuck-closed, CIPs stuck-open) associated with a given wire segment (or multiple wire segments in the case of shorts or CIPs stuck-closed), we see some cases where we will not always be able to locate and identify the complete set of faults. These cases arise when the faults can begin to interfere with the routing configurations that would be needed to diagnose a given fault in the set of multiple faults. However, this does not mean that we cannot accurately diagnose multiple faults in the FPGA. If multiple faults do not interact with the same set of wire segments, then these faults are really a collection of disjoint single faults that can each be diagnosed independently. When multiple faults do interact to limit diagnostic resolution, then the collection of wire segments and CIPs associated with that group of faults can be declared faulty such that fault-tolerant techniques would avoid that collection of routing resources altogether.

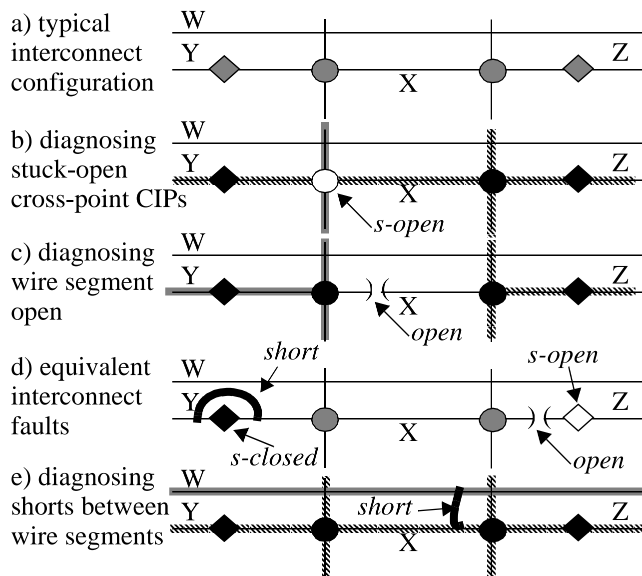


Figure 13. Diagnostic resolution examples

The definition of a segment depends on the type of fault. For opens, a segment is a wire bounded by two CIPs with no other CIPs in between. For example, in Figure 14, B_1 and B_2 are breakpoints, and X_1 and X_2 are cross-points. The heavy line denotes a signal net routed through X_1 and X_2. The dashed line denote segments that are connected to the same signal net, although they don't serve to connect it to other segments (note that a cross-point CIP does not break continuity along the horizontal or vertical segments). Each one of the three segments B_1X_1, X_1X_2, and X_2B_2 may have an open fault. For shorts, however, a segment is a wire bounded by break-point CIPs. For example, a short between B_1X_1 and another segment *Z* will affect the entire segment B_1B_2, which will be treated as a single faulty segment.

Let *P* and *Q* be two faulty segments. To distinguish between *P* and *Q* being both open and *P* and *Q* being shorted to each other, we configure a TPG to generate the four possible patterns (00, 01, 10, 11) for the two suspects, and record the cases that fail. Any mismatch for the 00, 11 test patterns would indicate an open while any mismatches for the 01, 10 patterns would indicate a short. In case of an open, we may initially only know the open segment between two break-points CIPs. However, we can diagnose further to determine which internal segment is open by using the strategy illustrated in Figure 11d. Here we re-route portions of the segment based on the cross-point CIP locations along that segment in order to determine the relative location(s) of the open(s). For example, in Figure 14 we can diagnose each one of the three segments B_1X_1, X_1X_2, and X_2B_2 independently by the TPG to ORA connections. In this example, the dotted lines would extend beyond the break-point CIPs and represent two different configurations of the BIST structure (one for diagnosing B_1X_1 and the other for X_2B_2), while the third configuration would be represented by the bold line.

5.8 Shorts Involving the Configuration Memory

Consider a short between a configuration memory bit *B* and an wire segment *S*, and assume that *B* controls a CIP *C*. Note that the value *b* of *B* is constant during a test phase. This short may be detected in any configurations where *S* is used and at least once when *B* is set to $\bar{b}$. The detection will occur in one of two cases: 1) the short creates a wrong value on *B* and *C* is under test, or 2) the short creates the wrong value on *S* and *S* is under test. The problem is that C may be under test in different configurations, and the value of B may be affected in only some of them. Similarly, S may be tested in different configurations, and the value of B may affect only

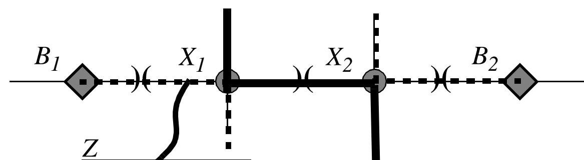


Figure 14. Wire segments and faults

some of them. This behavior seems to be that of an intermittent fault, as the same resource looks like it fails only some of the configurations where it is tested. As a result, a diagnostic procedure that looks for consistency is doomed to fail.

The practical solution in such cases is to label all of the suspected resources as faulty, even if a consistent diagnosis is not possible. Avoiding all the potential defects is a reasonable strategy in fault-tolerant applications and for yield enhancement as long as the known-to-be-correct portion of the device is has sufficient resources to contain the intended system application.

5.9 Automated Diagnostic Procedure

An automated procedure was developed to diagnose shorts and opens in global busses. We describe that procedure here as an example of how many of the diagnostic approaches described in this paper have been used in our applications. This procedure was implemented and successfully diagnosed emulated faults in an ORCA 2C15 FPGA down to the resolution of a single CIP. This is an adaptive procedure because the diagnostic configurations and techniques used depend on the results of the interconnect fault detection test results as well as previous diagnostic steps. This is also a dynamic procedure because some of the diagnostic test configurations are computed based on the actual response of the circuit.

The procedure begins during the execution of the interconnect fault detection configurations in a STAR position by recording the failing phases and their signatures for later analysis. At the completion of the fault detection BIST session, diagnosis begins if any test phases fail. First, the failing phases and their signatures are compared against the fault dictionary containing global bus failure modes. This dictionary takes into consideration that multiple busses could be faulty. This comparison then returns a set of target busses in which to search for faults.

The next step of the procedure checks for shorts on these busses using the scan ORA configurations of the failing test phases. Here the diagnostic software analyzes only the all 1's, all 0's, walking 1, and walking 0 patterns. If a short between nets is detected the procedure attempts to localize the fault using progressive deletion of wire segments on the dominant net. Otherwise, if a constant value is seen on faulty nets the diagnostic software assumes an open on this global bus and performs additional steps to verify this assumption.

The first of these additional steps is to download and analyze the results of the multiple ORA configurations and their ‘flipped’ counterpart. To determine which CIP may be faulty the diagnostic software must be programmed to know both the location (row and column) of the PLB programmed as an ORA in relation to its BIST pass/fail result within the total signature and the location of the breakpoint CIP in relation to where the signal is ‘tapped off’ from the bus onto the local routing. This second piece of information is necessary to determine if a CIP stuck-open in one row will affect this row or the next and is readily available to the programmer from the FPGA data book. If more than two breakpoint CIPs on the same WUT were identified as being stuck-open, a divide-and-conquer BIST tile searches between these areas for other CIPs that may be also be faulty.

The galaxy BIST for off-line interconnect fault detection simply runs the STARs in parallel with the ORAs connected in a longer scan chain. This allows the diagnostic techniques described above to be used in both on-line and off-line testing and diagnosis. By separating the concatenated ORA results into the original STAR positions, the same diagnostic procedures executed in each failing STAR position with absolutely no modification. Diagnosis in different areas of the chip could also be executed in parallel to reduce diagnostic runtime as long as the ORA results are connected together in the scan chain and different diagnostic configurations are dynamically generated based on different testing results in different STAR positions.

6. Experimental Results

Diagnostic test configurations were applied to the two faulty ORCA 2C15 FPGAs whose failing BIST results were shown in Figure 10. The diagnostic results identified a single routing fault in Chip 1 at the intersection of the two STARs. Furthermore, we were able to determine that the fault is a short in a single local routing segment in row 10 column 8 (recall that the STAR consists of two columns or rows of PLBs such that V-STAR position 4 in Figure 10a corresponds to columns 7 and 8 in the FPGA). However, the faulty segment is not shorted to another interconnect segment. Using the Scan ORAs it was observed that this wire segment appears always stuck-at-1. This information along with the short behavior indicates that this local wire segment is shorted to the power supply voltage.

Application of the diagnostic test configurations to Chip 2 confirmed that there are indeed two faults. The first fault is a short in a single local routing segment in row 1 column 12 (at the intersection of the failing V-STAR and the top failing H-STAR in Figure 10b). Like the short in Chip 1, the diagnostic results indicate that this fault in Chip 2 is also a short to power. The second fault in Chip 2 is in the horizontal global routing resources in row 5 (specifically the *hx*4s bus in Figure 2) and does not effect V-STAR testing. This fault is a short between three of the wires in the 4-wire bus and the short is located between columns 6 and 8. Here the scan ORAs allowed us to observe that these wires exhibit a majority voting behavior.

Since real faulty chips are difficult to come by, we also used the fault injection emulator we previously developed for FPGAs [18] to perform further evaluation and validation of the diagnostic tests and procedures described in this paper. This fault emulator injects faults into the FPGA by manipulating the configuration memory bits during download to emulate faults during BIST. The faults that can be emulated include CIPs stuck-open and stuck-closed as well as flip-flop and LUT stuck-at faults. The fault injection emulator was used in conjunction with the automated adaptive diagnostic procedure for shorts and opens in global busses to obtain the data in Table 3. In this experiment, faults were confined to a single STAR and to a single global bus type (in this case, CIP stuck-open faults in *vx*4 busses in the ORCA 2C15). This makes the diagnosis more difficult when compared to faults scattered around the FPGA and in different types of busses. As can be seen from the data, an average of 5 diagnostic configurations are generated per fault in the STAR with the generation of each diagnostic configuration taking about 10

seconds. The time required to download and execute the diagnostic configuration is insignificant compared to the time required to generate the adaptive diagnostic configuration, hence the long diagnostic run-time. Up to 5 faults in the STAR and in the same buss type could be uniquely diagnosed (meaning all of the injected faults were correctly identified). With 6 faults in the STAR, only 5 of the 6 faults were correctly identified in this particular experiment, but the diagnostic procedure did report an inconsistency which indicated that the diagnosis was not unique. These results suggest that the automated adaptive diagnostic procedure (as currently implemented) is capable of uniquely identifying up to 100 faults in each bus type in the entire ORCA 2C15 FPGA assuming 5 faults in each of the 10 V-STAR and 10 H-STAR positions. With 5 bus types in the ORCA 2C15, up to 500 faults in the entire FPGA interconnect could be uniquely identified.

Table 3. Adaptive diagnostic results from fault injection

Faults/ STAR	Configurations Generated	Diagnostic Run-Time	Diagnostic Results
1	7	1 min. 10 sec.	unique
2	9	1 min. 30 sec.	unique
3	16	2 min. 40 sec.	unique
4	22	4 min. 10 sec.	unique
5	23	4 min. 40 sec.	unique
6	26	5 min. 10 sec.	5 of 6

7. Conclusions

We have presented an approach for both on-line and off-line BIST and BIST-based diagnosis for faults in the interconnect networks of FPGAs. The BIST and diagnostic test techniques can be applied in general to any FPGA, including embedded FPGA cores in SoC applications [19]. In fact, some of the diagnostic techniques described here can be applied to other FPGA interconnect BIST approaches such as the parity-based approach proposed in [9]. However, the BIST and diagnostic phases as well as the number of test phases in each test session are a function of the specific FPGA routing architecture. We have implemented the adaptive diagnostic procedure for shorts and opens in the global busses for the ORCA 2C and 2CA series FPGAs. We have integrated these BIST-based diagnostic configurations with our on-line BIST and reconfiguration approach for fault-tolerance in FPGAs using roving STARs. In addition, we have extended this development to off-line testing and diagnosis for manufacturing testing to assist in FMA for yield enhancement as well as for off-line system-level testing to facilitate reconfiguration around faulty resources prior to bringing the system back on-line.

When we applied our routing BIST diagnostic techniques to known faulty FPGAs we were able to detect, locate, and identify the routing faults in all cases with diagnostic resolution to the faulty wire segment(s). Our diagnosis not only identified the faulty resources, which is sufficient for avoidance by fault-tolerant techniques, but also provided additional information on the behavior of these shorts. Finally, the automated diagnostic procedure was further verified using fault injection emulation to demonstrate that these techniques can be applied without human intervention. The next challenge is the ability to distinguish and diagnose interconnect faults in the presence of faults in the PLBs.

References

[1] 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," *Proc. Intn'l. Test Conf.*, pp. 973-982, 1999

[2] 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," *Proc. Third NASA/DoD Workshop on Evolvable Hardware*, pp. 73-92, 2001

[3] F. Lombardi, D. Ashen, X. Chen, and W. K. Huang, "Diagnosing Programmable Interconnect Systems for FPGAs," *Proc. ACM/SIGDA Intn'l. Symp. on FPGAs*, pp. 100-106, 1996

[4] H. Michinishi, T. Yokohira, T. Okamoto, T. Inoue, and H. Fujiwara, "A Test Methodology for Interconnect Structures of LUT-Based FPGAs," *Proc. IEEE Asian Test Symp.*, pp. 68-74, 1996

[5] M. Renovell, J. Portal, J. Figueras, and Y. Zorian, "Testing the Interconnect of RAM-Based FPGA," *Proc. IEEE Design & Test of Computers*, Vol. 15, No. 1, pp. 45-50, 1998

[6] C. Stroud, S. Wijesuriya, C. Hamilton, and M. Abramovici, "Built-In Self-Test of FPGA Interconnect," *Proc. Intn'l. Test Conf.*, pp. 404-411, 1998

[7] I. Harris and R. Tessier, "Interconnect Testing in Cluster-Based FPGA Architectures," *Proc. AMC/IEEE Design Automation Conf.*, pp. 49-54, 2000

[8] I. Harris and R. Tessier, "Diagnosis of Interconnect Faults in Cluster-Based FPGA Architectures," *Proc. IEEE Intn'l Conf. on Computer Aided Design*, pp. 472-476, 2000

[9] X. Sun, J. Xu, B. Chan and P. Trouborst, "Novel Technique for BIST of FPGA Interconnects," *Proc. IEEE International Test Conf.*, pp. 795-803, 2000

[10] M. Abramovici, C. Stroud, B. Skaggs, and J. Emmert, "Improving On-Line BIST-Based Diagnosis for Roving STARs," *Proc. IEEE Intn'l On-Line Test Workshop*, pp. 31-39, 2000

[11] C. Stroud, E. Lee, and M. Abramovici, "BIST-based Diagnostics for FPGA Logic Blocks," *Proc. IEEE Intn'l. Test Conf.*, pp. 539-547, 1997

[12] Lattice Semiconductor, *http://www.latticesemi.com/products*

[13] Xilinx, Inc., *http://www.xilinx.com/products*

[14] C. Stroud, *A Designer's Guide to Built-In Self-Test*, Kluwer Academic Publishers, 2002

[15] M. Abramovici and C. Stroud, "BIST-Based Test and Diagnosis of FPGA Logic Blocks," *IEEE Trans. on VLSI*, Vol. 9, No. 1, pp. 159-172, 2001

[16] "Standard Test Access Port and Boundary-Scan Architecture," *IEEE Standard P1149.1*, 1990

[17] C. Stroud, M. Lashinsky, J. Nall, J. Emmert, and M. Abramovici, "On-Line BIST and Diagnosis of FPGA Interconnect Using Roving STARs," *Proc. IEEE Intn'l On-Line Testing Workshop*, pp. 31-39, 2001

[18] T. Slaughter, C. Stroud, and B. Skaggs, "Fault Injection Emulator for Field Programmable Gate Arrays", *Proc. Intn'l Society of Optical Engineering*, Vol. 4525, pp. 1-9, 2001

[19] M. Abramovici, C. Stroud, and J. Emmert, Using Embedded FPGAs for SoC Yield Enhancement", Proc. ACM/IEEE Design Automation Conf., pp. 713-724, 2002