BIST for Xilinx 4000 and Spartan Series FPGAs: A Case Study

Charles E. Stroud
Dept. of Electrical and Computer Engineering
Auburn University
Auburn, AL

Keshia N. Leach, and Thomas A. Slaughter
Dept. of Electrical and Computer Engineering
University of North Carolina at Charlotte
Charlotte, NC

Abstract: We discuss the development of Built-In Self-Test (BIST) configurations that test all of the programmable logic and interconnect resources in the core of Xilinx 4000E, 4000XL/XLA and Spartan series Field Programmable Gate Arrays (FPGAs). While there has been prior work in BIST for these FPGAs, the fast-carry logic has not been addressed and only a small portion of the total interconnect resources has been targeted. The programmable logic is completely tested in two test sessions of 12 BIST configurations each, but the programmable interconnect resources requires as many as 206 BIST configurations. Therefore, we also discuss architectural features that affect the testability of the FPGA and, in turn, the number of BIST configurations needed by comparing the BIST configurations developed for these Xilinx FPGAs with those previously developed for other FPGAs.[1]

1. Introduction

There has been considerable prior work in the development of BIST for FPGAs [1]-[20]. The basic idea is to program some of the programmable logic blocks (PLBs) to function as Test Pattern Generators (TPGs) and Output Response Analyzers (ORAs). Depending on whether the BIST is targeted at testing programmable logic or interconnect resources, PLBs or routing resources are configured as blocks under test (BUTs) or wires under test (WUTs), respectively. BIST will generally require more test configurations than external testing approaches for FPGAs since logic and routing resources used as TPGs and ORAs must be tested with additional configurations. However, the advantage of BIST over external testing is the ability to use the BIST configurations at all levels of testing (device through system-level testing) as well as eliminating the need for expensive external test equipment. For practical application, the total number of BIST configurations must be reasonable while providing high fault coverage.

One limitation of BIST for FPGAs is that while the BIST architecture is generic, the specific test configurations are not, and must be developed for each specific PLB and/or interconnect architecture. Once these BIST configurations have been developed, they can be applied to all devices of the same size and type. In some cases, the BIST configurations can be algorithmically scaled to fit different size FPGAs in the same family. However, developing BIST configurations for a new family of FPGAs typically requires development of new test phases specific to that architecture.

Some of the notable works on BIST and BIST-related testing of FPGAs include [1][5][7][8][10][15]-[20]. While many of these are related to Xilinx 4000 series FPGAs, none consider complete testing of the PLBs and interconnect network in the Xilinx 4000 series. For example, testing the PLBs in the original Xilinx 4000 series FPGAs was addressed in [6] and [8] with the number of test configurations ranging from 6 to 9, but the carry-logic circuitry for constructing fast adders, subtractors, and counters was not considered. Testing programmable interconnect resources in the original Xilinx 4000 series FPGAs was addressed in [4],[5], and [16], but only a small subset of the complete interconnect network was considered. Only global routing resources that span a single PLB (referred to as *by-1* routing resources) were considered, requiring a total of 6 test configurations [16].

We have developed BIST configurations that completely test all of the programmable logic and interconnect resources in the core of the Xilinx 4000E, 4000XL/XLA, and Spartan series FPGAs. Our effort has resulted in 12 logic BIST configurations, including complete testing of the carry-logic circuitry. The surprise came, however, in the development of the routing BIST configurations where as many as 206 configurations are required to test the programmable interconnect resources. This is a significant increase in the number of test configurations previously reported [16], including those BIST configurations reported for other FPGA interconnect architectures [15]. The primary reason for this increase is that the total number of BIST configurations required for complete testing of the logic and routing resources is a function of the architectural features of the FPGA logic and interconnect resources. Therefore, the question arises, what architectural features lead to testing problems in FPGA architectures, causing significant increases in the total number of test configurations required?

In this paper, we discuss the development of BIST configurations for the Xilinx 4000 and Spartan series FPGAs, and more importantly, some of the architectural features that lead to testing problems in FPGAs. We begin with a brief overview of the Xilinx 4000 and Spartan series FPGA architectures in Section 2. Next we discuss BIST for PLBs in Section 3 where we include a more detailed discussion of prior work in FPGA logic BIST. Similarly, in Section 4, we discussed prior work in BIST for programmable interconnect resources as well as the routing BIST configurations we have developed. All BIST configurations are generated automatically by programs we have developed (written as Perl scripts or in C) such that the user only needs to specify the series and size of the array. We summarize specific architectural features that affect the testability of FPGAs in Section 5 before we conclude the paper.

2. Overview of Xilinx FPGAs

The overall architecture of all Xilinx 4000 and Spartan series FPGAs is identical in that a 2-dimensional array of PLBs is interconnected by a programmable routing network

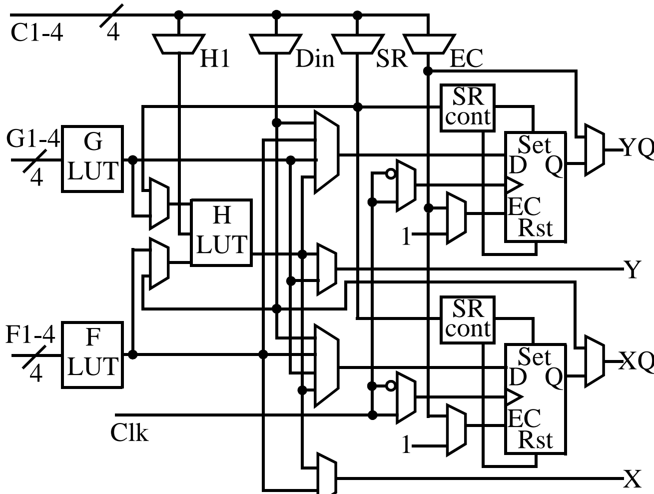
1. Project funded by the National Security Agency under contract MDA904-01-C-1825.

[22]. However, there are differences between the three main families we targeted (4000E, 4000XL/XLA, and Spartan) in terms of the programmable logic and interconnect resources [21]. In the following subsections, we provide an overview of the logic and routing resources along with the differences between the three families. Finally, we briefly overview the Optimized Reconfigurable Cell Array (ORCA) architecture [23] which will later be used for comparison in understanding FPGA testability issues.

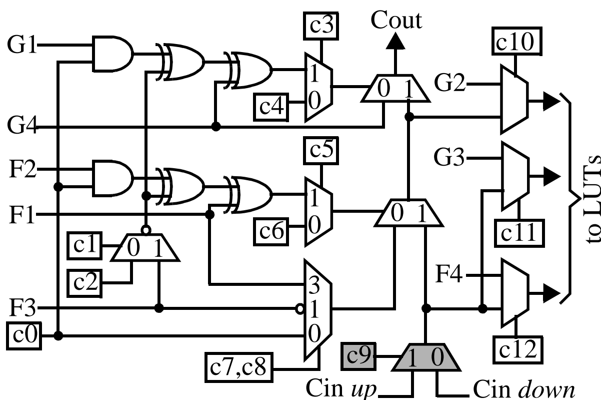
2.1 Overview of Logic

The PLB consists of two flip-flops, two 4-input Look-Up Tables (LUTs), one 3-input LUT, and a variety of configuration multiplexers (MUXs) for establishing the functionality and interconnection of the various logic components in the PLB, as illustrated in Figure 1a. The two 4-input LUTs are referred to as the *F* and *G* LUTs while the 3-input LUT is referred to as the *H* LUT. The PLBs are also referred to as Configurable Logic Blocks (CLBs) in Xilinx terminology. One of the main differences between the families in the 4000 series are the logic capabilities within the LUTs and flip-flops of the PLB. In all 4000 series FPGAs, the two 4-input LUTs can function as two independent 16x1 asynchronous Random Access Memories (RAMs) or together as either a 16x2 asynchronous RAM or a single 32x1 asynchronous RAM. In the 4000E series, the RAMs can operate synchronously and the two 4-input LUTs can be operated as a 16x1 Dual-Port RAM (DPRAM). In addition, the *H* LUT can operate independently from the *F* and *G* LUTs. The 4000XL has the same logic capabilities as the 4000E with the addition that the two flip-flops can also function as level sensitive latches.

The carry logic for implementing fast adders, subtractors, and counters is separate and dedicated logic illustrated in Figure 1b. As can be seen from the diagram, the upper portion of the carry-logic circuitry is observable only via the dedicated carry-out output that drives the next PLB up the column in the case of the 4000XL series. In the 4000E series, the carry-out output that drives the adjacent PLBs above and below; this accounts for the *c9* configuration bit and multiplexer shown in gray in Figure 1b that are not present in the 4000XL series. It should be noted that the various Spartan series (i.e., E and XL) FPGAs have logic and carry features similar to their 4000 series counterparts.



a) logic block architecture



b) carry-logic architecture

Figure 1. 4000 and Spartan PLB architecture

2.2 Overview of Interconnect

The programmable interconnect network consists of many wire segments that can be connected or disconnected by switches called programmable interconnect points (PIPs). The basic PIP structure (Figure 2a) consists of a transmission gate controlled by a configuration memory bit (CB). There are three basic types of PIPs: cross-point (Figure 2b), break-point (Figure 2c), and multiplexer (Figure 2d) [15]. Xilinx FPGAs also incorporate a *switch-box PIP* (also referred to as a Programmable Switch Matrix in Xilinx terminology) which is a combination of six break-point PIPs (Figure 2e). A signal path is formed by connecting wire segments in sequence with multiple closed (activated or on) PIPs. Wire segments and PIPs that are associated with a specific PLB are referred to as *local* routing resources while *global* routing resources refer to wire segments and PIPs that provide access to any PLB in the FPGA. The approximate percentage of the total number of PIPs in the 4000XL series FPGAs are given next to each PIP type in Figure 2; the MUX PIPs are by far the most prevalent.

The basic interconnect architecture for the 4000 and Spartan series FPGAs is illustrated in Figure 3 in terms of the general busses that span 1 PLB (referred to as *by-1*), 2 PLBs (referred to as *by-2*), 4 PLBs (referred to as *by-4*), and long lines that span a quarter, half, or all of the PLB array [22]. Note that the interconnect resources are shared between PLBs as indicated in Figure 3 by the MUX PIPs to adjacent PLBs that connect to the same routing resources as the PLB in row *i* and column *j*, denoted as PLB*i,j*. Also note that the inputs to the PLB are distributed equally along each of the four sides of

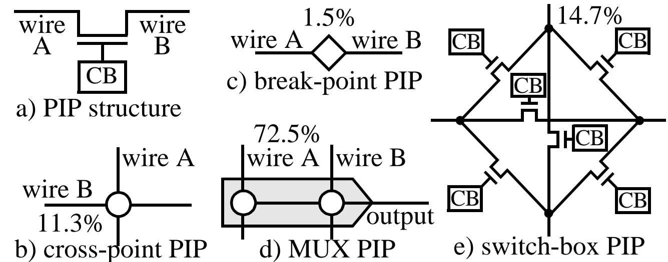


Figure 2. Programmable Interconnect Points (PIPs)

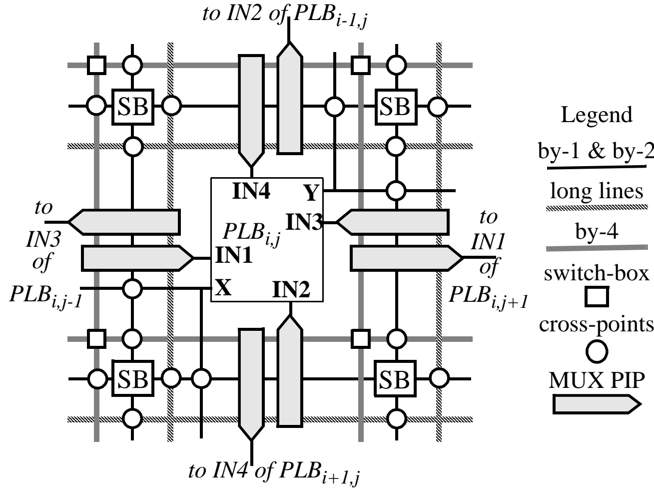


Figure 3. Programmable interconnect architecture

the PLB. The four sets of inputs are denoted *IN1-4* in the figure where each set consists of a single *F*, *G*, and *C* input to the PLB such that *IN1*, for example, consists of PLB inputs *F1*, *G1*, and *C1*. As a result, each set of inputs is associated with a specific set of interconnect resources such that the four different inputs to each of the *F* and *G* LUTs are sourced from routing resources on the four different sides of the PLB. Similarly, the four PLB outputs (*X*, *Y*, *XQ*, and *YQ*) are also distributed around the physical perimeter of the PLB. The interconnect resources shared between adjacent PLBs and the physical distribution of PLB inputs and outputs lead to problems in developing BIST configurations, as will be discussed.

The number of each type of routing resources are given in Table 1 for the 4000 and Spartan series FPGAs. There are more vertical than horizontal routing resources. Note that the direct lines to adjacent PLBs and carry lines between PLBs are not shown in Figure 3. The carry lines transfer the carry-out from the dedicated carry-logic in the PLB to one or more adjacent PLBs. In the 4000XL and XLA, the carry line only runs upward through the array while the carry lines in the 4000E series run both up and down the array. This dedicated carry routing will be another important factor in the logic BIST architecture and test configurations.

Table 1. Routing resources in Xilinx FPGAs

Routing Resource Type	4000E Spartan		4000XL 4000XLA		ORCA 2C OCRA 2CA	
	vert	horiz	vert	horiz	vert	horiz
by-1	8	8	8	8	8	8
by-2	4*	4*	4*	4*	0	0
by-4	0	0	12*	12*	8*	8*
long lines	10	6	18	6	8*	8*
direct lines	0	0	2	2	5	5
carry lines	2	0	1	0	2	2
Total	24	18	45	32	31	31
* denotes rotating and staggered busses						

2.3 Overview of ORCA

A brief overview of the ORCA 2C and 2CA series FPGA architecture will be of value when discussing some of the issues that effect the number of BIST configurations required to test a given FPGA architecture. The ORCA PLB architecture is similar to, but about twice as complex as, that of the Xilinx PLB in that it contains four 4-input LUTs and four flip-flops. The LUTs can function as 16x2 and 16x4 synchronous and asynchronous RAMs as well as a 16x2 DPRAM. Like Xilinx, the ORCA PLB flip-flops have programmable preset and clear as well as clock enable, and they can also function as a latch. In addition, the ORCA 2C flip-flops have a multiplexer at the front of the flip-flop, similar to a scan chain multiplexer which has been very useful in retrieving the BIST results at the end of each BIST sequence [2].

The ORCA 2C interconnect architecture, illustrated in Figure 4, is somewhat similar to the Xilinx 4000 and Spartan in that there are global wire segments of varying lengths including *by-1*, *by-4*, and long lines that span half or all of the PLB array [23]. The number wires in each type of bus are comparable for the Xilinx 4000XL [22] and ORCA 2C [23], as summarized in Table 1. However, there are four major differences in the programmable interconnect architectures that will ultimately influence the number of BIST configurations. First, the primary PIP resources in the ORCA FPGA are cross-point and break-point PIPs for global routing while the MUX PIPs in the ORCA are much smaller (5 inputs compared to 35 inputs in 4000XL). As a result, the cross-point PIPs account for about 69% of the total PIPs with the MUX PIPs accounting for only 19% and the break-point PIPs accounting for 12% of the total PIPs (there are no switch-box PIPs in the ORCA). A second difference is in the routing for the carry-logic. The ORCA has dedicated carry routing to all four adjacent PLBs, but the carry-out output of the carry-logic can be brought out on one of the five outputs of the PLB, increasing

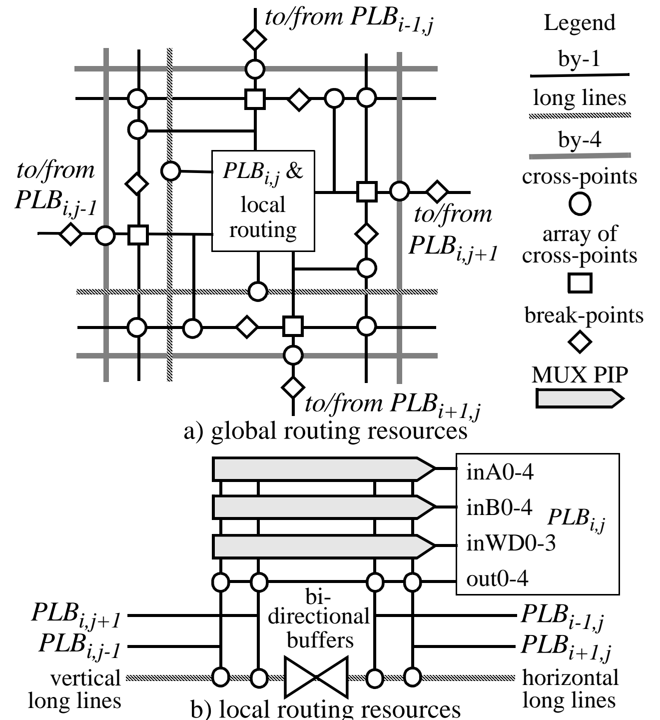


Figure 4. ORCA programmable interconnect architecture

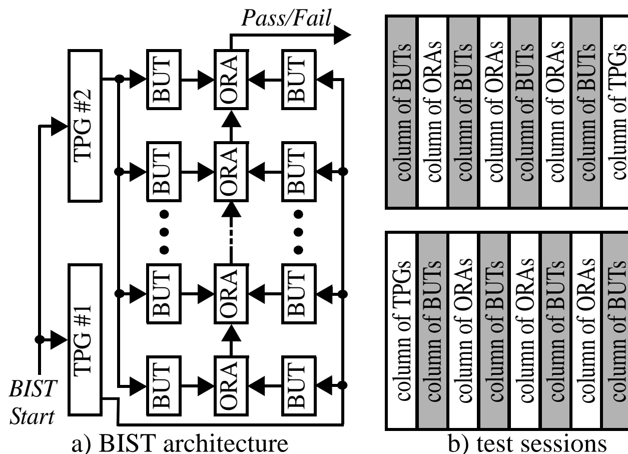
the observability of the carry-logic. The third major difference in the interconnect architectures is that routing resources shown in Figure 4a are associated with only the PLB shown and are not shared with adjacent PLBs (other than the direct lines that connect adjacent PLBs). Finally, the fourth major difference is that the local routing resources shown in Figure 4b allow inputs to and outputs from the PLBs to come from and go to busses on any side of the PLB. These last two differences account not only for a large increase in the number of BIST configurations but also in the difficulty encountered when developing BIST configurations, as will be discussed in later sections.

3. BIST for Programmable Logic Blocks

In this section we describe BIST for the PLBs in the Xilinx FPGAs. We begin with an overview of relative prior work in testing and BIST of FPGA logic blocks. Next, we summarize our BIST development for Xilinx 4000 and Spartan PLBs.

3.1 Prior Work in PLB Testing and BIST

The main BIST approach for testing PLBs in FPGAs is to configure groups of PLBs as TPGs and comparison-based ORAs, and another group of PLBs as blocks under test (BUTs), as illustrated in Figure 5a [1]. The PLBs forming the TPGs, BUTs and ORAs can be arranged in either rows or columns such that half of the PLBs are BUTs. Because of the many different modes of operation provided by the PLBs, complete testing will require multiple reconfigurations of the device. Once the BUTs have been tested in all of their modes of operation, the roles of the PLBs are reversed such that TPGs and ORAs become BUTs, and vice versa (see Figure 5b). A test configuration that tests the BUTs in one mode of operation is referred to as a *test phase* and the set of test phases that test the BUTs in all of their modes of operation is referred to as a *test session* [1]. A minimum of two test sessions are required to completely test all PLBs in the FPGA in all of their modes of operation, as can be seen in Figure 5b. This BIST approach has been implemented for the ORCA 2C and 2CA series FPGAs and required a total of 9 and 14 BIST configurations, respectively, per test session to completely test the PLBs including the fast carry-logic circuity [12].



a) BIST architecture b) test sessions

Figure 5. BIST architecture and test sessions for PLBs

3.2 BIST for Xilinx PLBs

Some FPGAs contain equal routing resources for each PLB for both rows and columns such that the PLBs of the BIST architecture in Figure 5a can easily be organized in either rows or columns. In Table 1, we see that the row and column routing resources of the Xilinx FPGAs are not equal with from 33% (4000E and Spartan) to 40% (4000XL) more vertical than horizontal routing resources. The Xilinx 4000XL series FPGA has 77% more horizontal and 87% more vertical routing resources than the 4000E and Spartan series FPGAs. Let N_I represent the number of inputs to the PLB that must be driven by test patterns from a given TPG. Then, a total of $2N_I$ signals must be driven from the two TPGs across (up/down) the entire FPGA to the BUTs in a column (row) based architecture. On the other hand, a total of N_I signals must be driven up/down (across) each column (row) of BUTs in a column (row) based architecture. Therefore, with more vertical routing resources than horizontal routing resources, as in the case of the Xilinx FPGAs, the BIST architecture should be column based to maximize the routability of the logic BIST configurations. However, regardless of the quantity of vertical and horizontal routing resources, the fact that the dedicated carry routing in the Xilinx FPGAs is oriented along the columns, and that the carry-out can only be observed on the dedicated carry routing, makes the column based BIST architecture a necessity for testing the PLB carry-logic. The carry routing also forces counter-based TPGs to be oriented in columns so that the carry can propagate between counter stages.

We found the minimum number of test phases to test the primary logic portion of the PLB (including the RAM modes of operation) to be 7, which is comparable in number to the 5 to 8 test configurations previously developed for the original 4000 series FPGA [6][8]. However another 8 configurations are needed to fully test the carry-logic circuitry. In addition, we could not merge the test phases for the RAM modes of operation with those for the carry-logic. Therefore, we developed 4 test phases for the RAM modes of operation and combined the 7 test phases for the remaining primary logic with the 8 test phases for the carry-logic for a total of 12 test phases per test session. The 4 RAM BIST test phases (summarized in Table 2) are automatically generated for any size array by Perl scripts using the process described in [14].

Table 2. RAM BIST Configurations

Test Phase	LUT RAM Mode	TPG	Clock
1	32x1 synchronous	March Y	rise
2	16x1 synchronous	March Y	fall
3	16x1 asynchronous	March Y	off
4	Dual-Port synchronous	DPR test	rise

The combined LUTs can be programmed to be a 32x1 synchronous RAM, a synchronous 16x2 RAM, an asynchronous 16x2 RAM, or as a synchronous DPRAM. To test the single port RAM modes, the March Y algorithm [17] is used. However, the DPRAM mode is not a true DPRAM since it has only one write port but dual read ports. As a result, the complex DPRAM March tests given in [18] are not needed to fully test the Xilinx PLB in its DPRAM mode of operation. Therefore, we developed the following algorithm (denoted “DPR test” in

Table 2) to test the DPRAM mode, where $\Downarrow r_{dp}$ denotes reading the entire RAM through the dual port.

DRP test: $\Uparrow\Downarrow$(w0) $\Uparrow$(r0,w1,$\Downarrow r_{dp}$,r1) $\Downarrow$(r1,w0,$\Uparrow r_{dp}$,r0) $\Uparrow\Downarrow$(r0)

The 8 BIST phases that test the remaining PLB logic and the carry-logic circuitry are summarized in Table 3 in terms of the configuration of the PLB. We have developed a C program to automatically generate these 8 BIST configurations for any size array in the textual intermediate design language known as Xilinx Design Language (XDL). In all 8 phases, a 12-bit binary up-counter is used to implement the TPG to exhaustively test the BUTs in the LUT mode of operation.

Table 3. Logic & Carry BIST Configurations

phase	1	2	3	4	5	6	7	8
clkX&Y	fall	fall	rise	fall	rise	rise	fall	rise
ecX&Y	ec	ec	off	ec	off	ec	ec	off
dX/Y	f/g	h	g/f	h	din	din	din	h
X/Ymux	h	f/g	f/h	f/g	h	h	h	f/g
setX&Y	off	off	off	off	sr	sr	sr	sr
srX&Y	set	reset	set	reset	reset	set	reset	reset
X/Yqmux	qx/qy	qx/qy	qx/ec	qx/qy	qx/qy	qx/qy	qx/qy	din/ec
H1	c1	c2	c3	c4	c1	c1	c1	c1
DIN	c2	c3	c4	c1	c2	c2	c2	c2
SR	c3	c4	c1	c2	c3	c3	c3	c3
EC	c4	c1	c2	c3	c4	c4	c4	c4
F&Gcarry	on	on	on	on	off	off	off	off
F,G&Hlut	xnor	xor	xnor	xor	xor	xnor	xor	xor
cinmux	cin	f3	f1	g4	1	0	f3	g4
carry	add-sub	add	sub	inc-dec	force	force	exam-ci	force
g3mux	g3i	g3i	cin	g3i	cin	cin	cin	cin
g2mux	cout0	cout0	cout0	cout0	g2i	g2i	g2i	cout0
f4mux	cin	cin	f4i	cin	f4i	cin	cin	cin

The TPGs have an important impact on the number of BIST configurations as a result of the number of PLBs required for TPG implementation. The two identical TPGs supplying test patterns to alternate columns of BUTs are critical in the fault detection capabilities of the overall BIST architecture [1]. Since the TPGs are confined to a single column (in this case) or row, then the minimum size FPGA to which this approach can be applied without incurring additional test sessions is determined by the number of PLBs required to implement the largest TPG for the set of test phases. Alternatively, additional test sessions must be added to allow the TPGs to occupy multiple columns or rows in smaller devices. The 12-bit counter requires a total of 6 PLBs for each TPG since the Xilinx PLB has two flip-flops per PLB. The March Y algorithm, on the other hand, requires a total of 8 PLBs to implement a single TPG. The DPR test algorithm requires a total of 9 PLBs per TPG and, as a result, determines the minimum size FPGA (an 18x18 array of PLBs) to which this BIST approach can be applied with only two test sessions. This means that FPGAs smaller than the 4008E (this includes the 4002XL, 4003E, 4005E/XL, and 4006E) and the Spartan 20 (this includes the Spartan 05 and 10) will require more than two test sessions. Our programs that automatically generate the logic BIST configurations do not support these smaller FPGAs.

For large FPGAs, the heavy loading on the TPGs during logic BIST and the delay incurred through many PIPs can limit the maximum operating frequency of the BIST circuitry. We used the internal 8 MHz oscillator for clocking the BIST circuitry. After performing timing analysis on all BIST configurations, it was determined that all test phases will operate at 8 MHz until the array size exceeds 36x36. For larger FPGAs, the array can be divided into quadrants with each quadrant configured with an independent BIST architecture and executed in parallel [1]. There is no increase in BIST execution time or the number of BIST configurations. Therefore, we perform this partitioning into quadrants for the 4036E/XL (which is a 36x36 array) and larger devices. The largest is the 4085XL which is a 56x56 array such that each quadrant consists of a 28x28 array and no additional partitioning is required to meet the 8 MHz operating frequency. The largest Spartan device is a 28x28 array and needs no partitioning.

The ORA for the RAM and logic BIST configurations is a comparison-based ORA implemented in a single PLB as illustrated in Figure 6. The *Y* and *YQ* outputs of the BUTs to the left and right of the ORA are compared in the *G* LUT while the *X* and *XQ* outputs of the BUTs to the left and right of the ORA are compared in the *F* LUT. The *H* LUT then ORs the outputs of the two comparators as well as the feedback from the YQ flip-flop such that once a mismatch occurs, a logic 1 is held in the ORA until the BIST sequence is complete and the ORA results have been read for the pass/fail determination of that particular test phase.

A scan chain interfacing to the IEEE 1149.1 Boundary Scan Interface common to most current FPGAs has proven to be an effective means for retrieving the BIST results contained in the ORAs at the end of the BIST sequence [2]. While the ORCA flip-flops have a front-end multiplexer that is excellent for constructing a scan chain, the Xilinx flip-flops have no such mechanism. Constructing a scan chain with the 4000 and Spartan series FPGAs requires the use of a LUT to construct the multiplexer. This means fewer BUT outputs can be compared by an ORA which, in turn, means more test configurations are required to completely test the PLB. An alternative is to read the contents of the ORA flip-flops by a readback operation on the configuration memory. Unfortunately this requires reading the entire configuration memory to obtain the ORA flip-flop contents which doubles the testing time [9]. However, this a an even trade-off since comparing half of the BUT outputs in the ORA in order to implement a scan chain would double the number of test configurations needed to test the PLBs. Therefore, we chose configuration memory readback for retrieval of the BIST results.

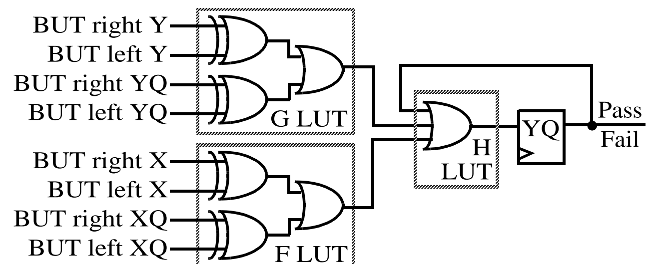


Figure 6. Comparison-based ORA

4. BIST for Interconnect

The development of BIST for programmable interconnect is by far more difficult than BIST for PLBs. In this section we overview our BIST configurations for the routing resources in the Xilinx FPGAs. We begin with an overview of relative prior work in testing and BIST of FPGA interconnect.

4.1 Prior Work in Interconnect Testing and BIST

There are two main approaches to BIST for programmable interconnect. In the first, a similar approach that of testing PLBs is taken, where some of the PLBs are programmed to operate at TPGs and comparison-based ORAs, while the wire segments and PIPs of the routing network are configured to form groups of wires under test (WUTs) [15]. This architecture is illustrated in Figure 7 where local interconnect resources are tested by WUTs that are routed through PLBs. To avoid equivalent faults in two sets of WUTs from escaping detection, each set of WUTs is tested at least twice and compared with a different set of WUTs each time it is tested. To detect bridging faults between the WUTs and adjacent wire segments not under test, opposite logic values must be applied to those adjacent wire segments not under test. Similarly, PIPs stuck-on (stuck-closed) faults are detected by applying opposite logic values to the wire segments on both sides of an open (de-activated or off) PIP. This means that additional signals must be routed aside from the WUTs themselves. This BIST approach has been implemented in ORCA FPGAs for all global and local routing resources in the core of the FPGA [15].

The second routing BIST approach is based on fault detection via parity and illustrated in Figure 8 [16]. As opposed to supplying test patterns to two sets of WUTs, there is only one set of WUTs with a parity bit generated for each test pattern and sent to the ORA over a separate signal path. In the ORA, a parity regenerate circuit calculates the parity on the incoming WUTs and compares this parity to the parity bit sent by the TPG to detect faults in the WUTs. One advantage of this BIST approach is that there is only one set of WUTs to be routed (instead of two as in the first routing BIST approach), improving the routability of the BIST approach. However, the TPG design is more complex than in the first routing BIST approach and will require more PLBs and routing resources for implementation. This was the first routing BIST approach implemented in Xilinx 4000 series FPGAs but the approach targeted only a subset of the routing resources in the FPGA; specifically, only *by-1* wire segments and switch-box PIPs. While the approach could conceivably be used to test all routing resources, only the BIST implementation for the *by-1* wire segments and switch-box PIPs was discussed in [16].

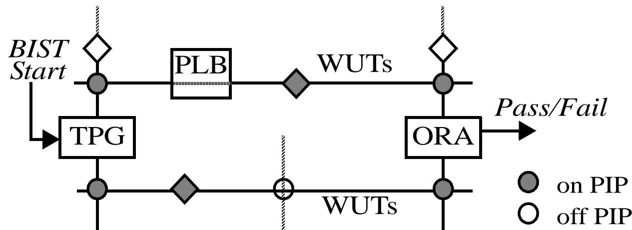


Figure 7. BIST architecture for interconnect

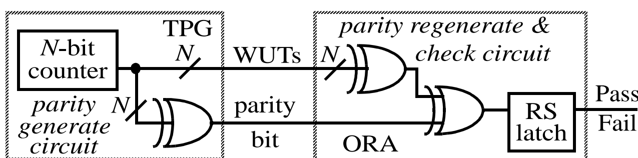
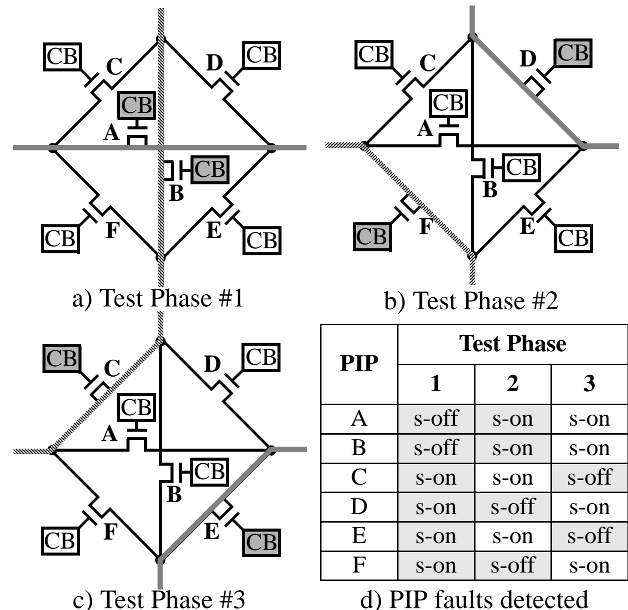


Figure 8. Parity-based interconnect BIST architecture

A set of 3 test configurations were originally developed in [4] to detect all faults in the *by-1* wire segments and their associated switch-box PIPs in Xilinx 4000 series FPGAs; these are illustrated in Figure 9a-c with the associated faults detected by each test phase given in the table of Figure 9d. In all 3 test phases, both logic values (0 and 1) are applied in turn to one signal line (i.e., the gray line) with opposite logic values applied to the other signal line (i.e., the striped line) in order to detect the stuck-on PIP faults [15]. These same 3 test configurations were used as BIST configurations in [16] to test the *by-1* wire segments and their associated switch-box PIPs. Since only half of the *by-1* wire segments and their associated switch-box PIPs in the FPGA were under test during one of these 3 BIST configurations, a total of 6 BIST configurations were needed to test all *by-1* wire segments and their associated switch-box PIPs in the Xilinx 4000 series FPGA. Not only are these *by-1* busses and switch-box PIPs a small subset (only about 10%) of the total interconnect network, but we found these routing resources to also be the easiest to test in terms of requiring fewer test configurations than most other routing resources.

While the set of 3 test configurations provide complete testing of the switch-box PIPs, all stuck-on break-point PIP faults are detected in two different test phases (see table in Figure 9d). After the first two test phases, all 6 break-point PIPs have been tested for stuck-on faults. As a result, the third test phase is only testing PIPs *C* and *E* for stuck-off faults. As an alternative, only a continuity test needs to be performed. This requires the control of fewer routing resources and the generation of fewer test signals; hence, the alternative is an



PIP	Test Phase		
	1	2	3
A	s-off	s-on	s-on
B	s-off	s-on	s-on
C	s-on	s-on	s-off
D	s-on	s-off	s-on
E	s-on	s-on	s-off
F	s-on	s-off	s-on

d) PIP faults detected

Figure 9. Switch-box PIP test phases

easier test configuration to implement. We obtained the idea for this alternative test phase from the work on external diagnosis of faults in Xilinx *by-1* switch-boxes described in [19] where the 3 test configurations of Figure 9a-c were used for fault detection, while the continuity test configuration of was one of 8 tests used for fault diagnosis.

In order to test MUX PIPs, one test phase is required for each input to the MUX PIP to test that input for stuck-off PIP faults [15]. During this test phase, one or more unselected inputs to the MUX PIP can be tested for stuck-on PIP faults by applying the opposite logic values to the unselected inputs. Therefore, the minimum number of test phases to test MUX PIPs is the number of inputs to the largest MUX PIP in the FPGA [7].

4.2 BIST for Xilinx Interconnect

We used the comparison-based BIST approach [13] illustrated in Figure 7 and partitioned the interconnect network into small Self-Test AReas (STARs) with each STAR containing a TPG, an ORA, and two sets of WUTS. This approach gives good diagnostic resolution when faults are detected, with the resolution being a function of the size of the STAR. In most cases, the STAR size is determined by the type of wire segments and PIPs under test. For example, a long line will require a larger STAR than a *by-1* wire segment. With the comparison-based approach all interconnect resources must be tested at least twice and compared to different sets of interconnect resources in order to prevent equivalent faults from escaping detection [13]. This may increase the number of BIST configurations but it does not double the total number since most routing resources will naturally come under test multiple times as a result of testing other interconnect resources. For example, *by-1* wire segments and MUX PIPs at the inputs to the PLBs are frequently used when testing other routing resources in order to route from the TPGs to the WUTs and from the WUTs to the ORAs. However, one must be careful to make sure that the proper test conditions exist on these frequently used resources to ensure that faults will be detected; otherwise, dedicated test phases must be constructed for each type of routing resource. In our development we took the latter approach where we targeted specific types of routing resources and then made sure that these same routing resources were tested a second time in test phases for other types of routing resources to ensure that equivalent faults did not escape detection. For the most part, the routing BIST configurations developed for the Xilinx 4000 and Spartan are similar to those described in [13]. In this section, we describe specific differences due to various characteristics of the Xilinx architecture.

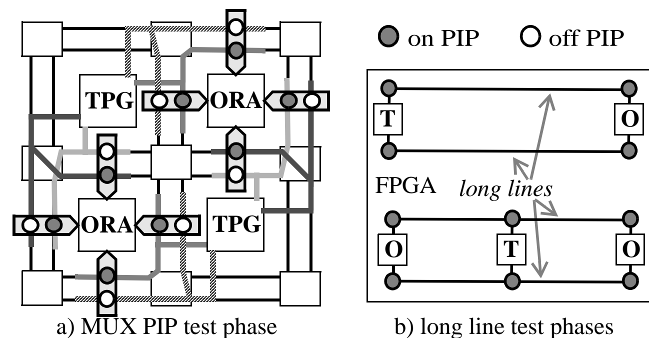
In order to minimize the number of BIST configurations required to test all of the interconnect resources, the number of wire segments in a set of WUTs as well as the number of WUTs should be maximized. This requires a TPG capable of driving a large number of wire segments in a set of WUTs. If multiple PLBs are required to construct the TPG for routing BIST, the interconnection of these PLBs can interfere with the routing of the WUTs and increase the number of test configurations required for complete testing of the routing resources. A binary counter offers a good TPG implementation in most FPGAs since we are interested in opposite logic values to be on two adjacent wires that can be shorted or on the opposite sides of an open PIP that could be stuck-on. But the number of bits of the counter, and the number of wires that can be driven by the counter is determined by the number of flip-flops in the PLB. There are only two flip-flops in the Xilinx PLB allowing one PLB to drive only two wires. Fortunately, the architecture of the Xilinx PLB that outputs both *X* and *Y* outputs as well as both *XQ* and *YQ* outputs can be taken advantage of to drive four wires per PLB instead of two. This is done by using the next state and the current state of the counter where the current state drives the *XQ* (counter LSB) and *YQ* (counter MSB) outputs while the next state drives the *X* and *Y* outputs. This is illustrated in Table 4 where it can be seen that any combination of two bits will produce both a (0,1) and a (1,0) on their respective wires during the test sequence. Therefore, a single Xilinx PLB programmed as a counter can function as a 4-bit TPG for routing BIST. A fifth test signal can be obtained by using a flip-flop in any neighboring PLB to delay the *YQ* output by one clock cycle.

Table 4. Current and Next States of 2-bit Counter

Count Value	Current State		Next State		YQ delayed
	YQ	XQ	Y	X	
0	0	0	0	1	1
1	0	1	1	0	0
2	1	0	1	1	0
3	1	1	0	0	1

In the 4000XL series FPGA, one out of as many as 35 inputs can be selected via a MUX PIP as an input to a given PLB. Similar MUX PIPs are located on each of the 4 sets of *F*, *G*, and *C* inputs to the PLB. The four *C* inputs must be tested separately from the *F* and *G* inputs due to PLB logic limitations for the comparison-based ORA. Therefore, at least 70 configurations are necessary for a complete test of these MUX PIPs. An example test phase is illustrated in Figure 10a where the TPG drives one signal on the line selected by the multiplexer as the input for that test, and another signal (producing opposite logic values) to at least one of the lines not selected. Shorts, stuck-on MUX PIPs, opens, and stuck-off MUX PIPs are detected as mismatches in the ORA. As a result, MUX PIPs can be tested in a number of phases equal to the number of inputs to the MUX PIP.

Shorts and opens in the switch-box PIPs for the *by-1* and *by-2* wire segments are tested simultaneously. As described



a) MUX PIP test phase b) long line test phases

Figure 10. BIST architecture for interconnect

previously, a minimum of 3 test phases are required for the switch-box PIPs. The limit on the number of inputs that can be compared by the ORAs increases the number of test phases to at least 9, since ORAs in these phases compare signals on routes that enter opposite sides of the ORA. Therefore only 4 wire segments of the 12 total *by-1*s and *by-2*s through a switch-box can be tested in a given test phase.

Long lines and *by-4* routing resources (with the exception of *by-4* switch-box PIPs) are tested as shown in Figure 10b for horizontal long lines (vertical long line test phases are rotated by 90 degrees). In some cases, the limited number of PIPs connecting the output of the TPGs to the long lines and the long lines to ORAs require placing the TPG in the center of a long line (the only access point) and the ORAs at either end.

Different routing resources under test often require different ORA implementations. In the MUX PIP test phases, the ORA is configured as shown in Figure 6 with four 2-input XOR gates to allow for comparison of 8 different signals applied to the *F* and *G* inputs. A different ORA design is necessary to compare the four *C* inputs since they lack direct access to the *F* and *G* LUTs, as shown in Figure 11, and since the *H* LUT lacks the logic resources to compare all four inputs simultaneously. The *C* inputs cannot be tested while the *F* and *G* inputs are tested because of limited logic resources and because the *C* inputs are used for the ORA feedback to latch up mismatches during the BIST sequence. A third ORA design is used when testing the switch-box PIPs. Because of the placement of the ORAs in relation to the WUTs, the ORAs for the switch-box PIP test phases are similar to that of Figure 6 but with a single 2-input XOR in the *F* and *G* LUTs.

The complete set of routing BIST configurations are summarized in Table 5 for the 4000E/Spartan and 4000XL/XLA series FPGAs. Due to the additional routing resources of the 4000XL/XLA series FPGAs, the total number of test configurations is considerably higher (over 60% more test phases) than that for the 4000E and Spartan series FPGAs. These additional routing resources, particularly the *by-4* lines and their associated switch-boxes, are also some of the most difficult to test and the more difficult test configurations to develop. Of the total 206 BIST configurations, 67 are generated using Perl scripts and the process described in [14]; these include the top three entries in Table 5. The remaining 138 routing BIST configurations are generated algorithmically by 13 different programs written in C. The development of these programs to generate BIST configurations for any size 4000E, 4000XL/XLA, or Spartan series FPGA required about 1 year for the three of us to complete and verify on actual parts.

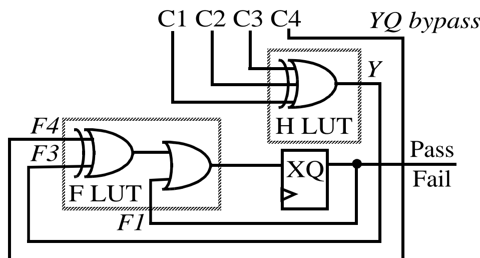


Figure 11. ORA for comparing C inputs to PLB

Table 5. Interconnect BIST Configurations

Interconnect Resources	4000E Spartan	4000X
by-1 lines & MUX PIPs	32	32
by-2 lines & MUX PIPs	24	24
by-1 & *by-2* switch-boxes	11	11
horizontal long lines	15	15
vertical long lines	18	24
tri-state buffers	12	12
clock lines & logic 0 tie-box	8	8
by-4 lines, MUX PIPs & switch-boxes	0	64
inter-quad routing	8	16
Total	**128**	**206**

5. Testability Analysis

The 12 logic BIST configurations for Xilinx 4000 and Spartan series are comparable to that of the ORCA which required 9 and 14 BIST configurations for the 2C and 2CA series FPGAs, respectively. However, routing BIST for the ORCA 2C required 27 configurations [15] or 44 configurations [13], depending on the BIST approach with the smaller STARs used in [13] requiring more configurations but providing much higher diagnostic resolution. This is compared to 206 routing BIST configurations for the 4000XL series FPGA and 128 for the 4000E and Spartan series FPGAs. The ORCA 2C and Xilinx 4000 and Spartan FPGAs are relatively comparable devices in terms of the number of routing resources in the interconnect network with the ORCA 2C falling between the 4000E/Spartan and the 4000XL as seen in Table 1. Yet, the 4000XL requires over 4 times the number of routing BIST configurations while the 4000E and Spartan series FPGAs require almost 3 times the number of routing BIST configurations when compared to the ORCA 2C. Therefore, the primary influence on the total number test configurations and, as a result, total test time is the routing architecture of FPGAs. In this section we discuss the FPGA architectural features that lead to testability problems and the increase the total number of BIST configurations as well as total testing time. We begin with the features that we believe have the most impact on testability.

As pointed out in Section 2.3, all ORCA PLB inputs can easily access routing resources from the top and bottom horizontal busses and the left and right vertical busses. The Xilinx PLB inputs, on the other hand, are associated with only one set of routing resources with *F1*, *G1*, and *C1* connected to the left vertical busses, F2, *G2*, and *C2* connected to the bottom horizontal busses, and so on around the perimeter of the PLB. This means that additional routing resources must be used to compare 4 signal running together on a bus (either vertical or horizontal) to test for shorts on that bus so that the four signals can be compared in a single LUT. These additional routing resources can often interfere with the routing of the WUTs such that more test configurations are needed to set up the proper test conditions on the WUTs ***and*** to route them to the ORA inputs. Note that a similar problem arises in logic BIST when we want to compare outputs from the left and right hand BUTs. The ability to access any PLB inputs from any of the four busses (top, bottom, left, or right) via local routing

resources eliminates this problem and helps to reduce the total number of BIST configurations.

Similarly, the outputs of the ORCA PLB can easily access any of the vertical and horizontal busses. However, the *X* and *XQ* outputs of the Xilinx PLB have direct access to only the left vertical and bottom horizontal busses while the *Y* and *YQ* outputs have direct access to only the right vertical and top horizontal busses. In addition, the number of wire segments within these busses that can be access by the *X*, *XQ*, *Y*, and *YQ* outputs are limited to a small subset of wire segments on each bus. This situation often requires the use of additional routing resources when driving TPG test signals onto WUTs to set up the necessary test conditions to test for shorts and PIP stuck-on faults. Again, these additional routing resources can often interfere with the routing of the WUTs such that more test configurations are needed to set up the proper test conditions on the WUTs. The ability to provide access from any PLB outputs to any of the four busses (top, bottom, left, or right) via local routing resources eliminates this problem and helps to reduce the total number of BIST configurations. However, it should be noted that this is not a problem in logic BIST for TPG routing but instead comes into play somewhat when making BUT to ORA connections.

Another influence on the number of routing BIST configurations is the fact that the routing resources are shared between PLBs in the Xilinx 4000 and Spartan series FPGAs. This leads to conflicts for interconnect resources when trying to route WUTs from/to the TPG/ORA resources as well as signals on wires not under test that are necessary for fault detection in the WUTs such as stuck-on faults in PIPs between WUTs and non-WUTs. Once again, this translates to more test configurations in order to avoid routing conflicts while maintaining necessary test conditions on the WUTs.

Rotating and staggered busses were a problem in both the Xilinx 4000/Spartan and ORCA 2C FPGAs and increased the number of routing BIST configurations in both FPGAs. The rotating busses typically rotate by 1 wire position as they pass each PLB and, as a result, their access to and from the PLB inputs and outputs changes with each rotation (as does the name of the wire segment as it moves from one PLB to the next). Staggered busses, on the other hand, will typically have one wire segment in the bus end at a given PLB while a new wire segment starts at that same PLB. As a result, access to the various wire segments of the staggered bus is more difficult, either from the TPG or to the ORA. This primarily adds considerable complexity to the test development, particularly when developing an algorithm to automatically generate the test configurations for any size array, and in some cases, increases the number of test configurations.

Routing resources at the periphery of the Xilinx PLB array go directly to I/O cells as well as to some global routing resources around the perimeter of the FPGA. In ORCA 2C, the PLBs at the periphery of the array had additional and dedicated routing resources to the I/O cells and perimeter routing. This allowed the core routing resources to be tested independently of the I/O cells. On the other hand, the switch-boxes around the periphery of the Xilinx PLB array, for example, cannot be tested with core routing of the FPGA and must be tested with the I/O cells. For manufacturing testing, this is no problem other than requiring additional test configurations. However, for system-level testing, all outputs from other devices on the PCB driving the FPGA must be tri-state in order to access the I/O cells to test the periphery core routing resources.

An obvious problem is the size of the MUX PIPs in FPGAs which require a minimum of one test configuration per input to the MUX. Unfortunately, MUX PIPs are becoming the dominant routing mechanism in most newer FPGAs (such as the Xilinx Virtex [21]) and the number of inputs to these MUX PIPs is becoming larger, increasing the number of test configurations as well as the test development effort.

A PLB architectural feature that affects development and test time associated with both routing and logic BIST configurations is the lack of multiplexers at the inputs to the flip-flops in the PLBs that could be used to create a scan chain for retrieval of ORA results at the end of the BIST sequence. There are a number of solutions to this problem [2]. One approach is to read the configuration memory to obtain the ORA contents as we have done in this development. This basically doubles testing time since the entire configuration memory must be read for each BIST configuration. Another problem we encountered was that the configuration memory readback circuitry is dynamic in the 4000 series FPGAs with maximum and minimum clock frequencies of 2 MHz and 1 MHz, respectively. We spent many days chasing phantom problems that were simply a result of violating the readback clock frequency specification. But this readback clock specification also has an impact on the total testing time. For example, in the 4010XL (a 20x20 array of PLBs), a single test phase with 283,376 configuration bits will require 18.9 msec at the maximum download clock frequency of 15 MHz while the BIST sequence takes 0.5 msec for 4000 clock cycles at maximum internal oscillator rate of 8 MHz, but BIST results retrieval via configuration memory readback requires 141.7 msec at the maximum readback clock frequency of 2 MHz for a total of 161.1 msec per BIST configuration.

An alternate approach to configuration memory readback is to construct a scan chain multiplexer in one of the LUTs but this also doubles testing time since twice as many BIST configurations must be typically be applied to compensate for the reduction in comparisons that can be performed by the ORA in any given test configuration. A final solution is to implement an incremental ORA [12] which ORs the results of all ORAs in a column (or row) and routes the output to a output pin of the FPGA or to the Boundary Scan chain in the device. However, considerable diagnostic resolution is given up with this approach [2]. The most desirable solution would be for FPGA manufacturers to either provide ability to address portions of the configuration memory or, even better, provide a static readback feature only for the contents of the PLB flip-flops [9].

A routing architectural feature that affects logic BIST is the dedicated carry routing for the fast carry-logic. By making the carry-out accessible on a regular PLB output (as was done in ORCA 2C) the carry-logic can be tested along with the PLB and independent of the carry routing architecture; thus giving

more routing flexibility when developing logic BIST configurations in a row or column based architecture. The dedicated carry routing can then be tested with the interconnect network.

The logic BIST architecture is very regular and, as a result, can be easily and algorithmically generated in terms of the physical placement of the TPGs, BUTs, and ORAs [1]. The major problem encountered when developing logic BIST configurations for the PLBs in Xilinx FPGAs comes in routing the interconnections between the PLBs using the Xilinx Place and Route (PAR) tool. The placement of the various TPGs, BUTs, and ORAs is not disturbed, however, in an effort to route the BIST configuration, the mapping of the BUTs and ORAs are modified to improve the routability. This is not a problem in the case of the ORAs since those PLBs are not under test during the test sessions in which they function as ORAs. However, in the case of the BUTs, the ability to ensure high fault coverage is lost since the intended configuration of the BUT is not maintained during the routing process. As a result, PLB configurations intended to detect specific faults are modified by PAR in many cases such that some faults will not be detected in BUTs where the configurations were modified. This problem can be overcome with considerable effort by manually routing the interconnections between the TPGs and BUTs, as well as the interconnections between the BUTs and ORAs, to eliminate the PAR tool from the BIST configuration development process. This was done by developing the BIST configurations in XDL where we could control both placement and routing via our own algorithms [14]. This adds considerable time and effort to the development process but is necessary in order to ensure the high fault coverage for which the logic BIST configurations are intended. In our case, this extra developed required two people about 1 month; a far cry from the year for routing BIST development, but still an extra month of development effort.

6. Conclusions

We have presented an overview of our development of BIST configurations for the Xilinx 4000 and Spartan series FPGAs. These BIST configurations test both the programmable logic resources (including the carry-logic circuitry) and the programmable interconnect in the core of the FPGA. All of the BIST configurations are automatically generated based on the size of the PLB array by programs we have developed. One of the most important results of this effort is the insight provided into the testability of FPGAs when one compares the number of BIST configurations required to test the Xilinx 4000XL (230 BIST configurations) and Spartan FPGAs (154 BIST configurations) with those previously developed for the ORCA 2C (62 BIST configurations) and 2CA (72 BIST configurations) FPGAs. We have compared the ORCA and Xilinx 4000/Spartan series FPGA architectures and have presented the architectural features that we believe are the underlying reasons for the increase in the total number of BIST configurations. Therefore, FPGAs are like most integrated circuits in that we can draw the following conclusions: 1) programmability does not translate to testability, and 2) the lack of attention to testability during the architectural design process does translate to long test development, long testing times, and high test costs.

References

[1] 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.

[2] C. Hamilton, G. Gibson, S. Wijesuriya and C. Stroud, "Enhanced BIST-Based Diagnosis of FPGAs via Boundary Scan Access," *Proc. IEEE VLSI Test Symp.*, pp. 413-418, 1999.

[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] M. Renovell, J. Figueras and Y. Zorian, "Test of RAM-Based FPGAs: Methodology and Application to Interconnect", *Proc. IEEE VLSI Test Symp.*, pp. 230-237, 1997.

[5] 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.

[6] M. Renovell, J. Portal, J. Figueras and Y. Zorian, "Minimizing the Number of Test Configurations for Different FPGA Families", *Proc. Asian Test Symp.*, pp. 363-368, 1999.

[7] M. Renovell, J. Portal, J. Figueras and Y. Zorian, "Testing the Configurable Interconnect/Logic Interface of SRAM-Based FPGAs", *Proc. IEEE Int'l Conf. on Design Automation and Test in Europe*, pp. 618-622, 1999.

[8] M. Renovell and Y. Zorian, "Different Experiments in Test Generation for Xilinx FPGAs," *Proc. IEEE Int'l Test Conf.*, pp. 854-862, 2000.

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

[10] C. Stroud, P. Chen, S. Konala and M. Abramovici, "Evaluation of FPGA Resources for Built-In Self-Test of Programmable Logic Blocks," *Proc. ACM Intn'l. Symp. on FPGAs*, pp. 107-113, 1996.

[11] C. Stroud, S. Konala, P. Chen, and M. Abramovici, "Built-In Self-Test for Programmable Logic Blocks in FPGAs," *Proc. IEEE VLSI Test Symp*., pp. 387-392, 1996.

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

[13] C. Stroud, J. Nall, M. Lashinsky and M. Abramovici, "BIST-Based Diagnosis of FPGA Interconnect," *Proc. IEEE Int'l Test Conf.*, pp. 618-627, 2002.

[14] C. Stroud, J. Nall, A. Taylor and L. Charnley, "A System for Automated Generation of Built-In Self-Test Configurations for Field Programmable Gate Arrays," *Proc. Int'l Conf. on Systems Engineering*, pp. 437-443, 2002.

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

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

[17] A. van de Goor, "Using March Tests to Test SRAMs," *IEEE Design & Test of Computers*, Vol. 10, No. 1, pp. 8-14, 1993.

[18] A. van de Goor and S. Hamdioui, "Fault Models and Tests for Two Port Memories," *Proc. IEEE VLSI Test Symp.*, pp. 401-410, 1998.

[19] S. Wang and C. Huang, "Testing and Diagnosis of Interconnect Structures in FPGAs", *Proc. Asian Test Symp.*, pp. 283-287, 1998.

[20] S. Wang and T. Tsai, "Test and Diagnosis of Faulty Logic Blocks in FPGAs," *Proc. IEEE Int'l. Conf. on Computer-Aided Design,* pp. 722-727, 1997.

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

[22] __, *The Programmable Logic Data Book*, Xilinx, Inc., 1996.

[23] __, *Field Programmable Gate Arrays Data Book*, Lucent Technologies, 1996.