BIST-Based Diagnostics of FPGA Logic Blocks

Charles Stroud, Eric Lee,
Dept. of Electrical Engineering
University of Kentucky

and Miron Abramovici
Bell Labs - Lucent Technologies
Murray Hill, NJ

Abstract: Accurate diagnosis is an essential requirement in many testing environments, since it is the basis for any repair or replacement strategy used for chip or system fault-tolerance. In this paper we present the first approach able to diagnose faulty programmable logic blocks (PLBs) in Field Programmable Gate Arrays (FPGAs) with maximal diagnostic resolution. Our approach is based on a new Built-In Self-Test (BIST) architecture for FPGAs and can accurately locate any single and most multiple faulty PLBs. An adaptive diagnostic strategy provides identification of faulty PLBs with a 7% increase in testing time over the complete detection test, and can also be used for manufacturing yield enhancement. We present results showing identification of faulty PLBs in defective ORCA chips.[1]

1. Introduction

An FPGA consists of an array of programmable logic blocks (PLBs) interconnected by a programmable routing network, and programmable I/O cells. The set of all programming bits establishes a *configuration* which determines the function of the device. In this paper, we consider in-circuit reprogrammable FPGAs, such as SRAM-based FPGAs, which may be reconfigured an arbitrarily large number of times. FPGA manufacturing tests are complicated by the need to cover all possible modes of operation of the PLBs and also to detect all the faults affecting the programmable interconnect network. Currently, these tests are generated manually by configuring several application circuits and exercising them with test patterns developed specifically for each application circuit. The FPGA manufacturing tests are not reusable for board and system-level testing, which require separate development efforts that rely on system diagnostic routines to test the FPGAs in their system mode of operation. The development of these diagnostic routines can be time-consuming and costly, and locating a faulty FPGA may be difficult.

Previous work in FPGA testing[7][8][18][19] took advantage of reprogrammability by treating testing just as another application to be implemented in the FPGA. Some of these methods also exploit the regular array structure of an FPGA by configuring it as one or more iterative logic arrays (ILAs)[7][8][19]. The techniques that rely on externally applied vectors[7][8] are applicable only for device-level manufacturing tests.

We have introduced a BIST approach for testing the PLBs in an FPGA, exploiting the reprogrammability of an FPGA to configure it exclusively with BIST logic during testing[18][19]. In this way, *testability is achieved without any overhead*, since the BIST logic "disappears" when the circuit is reconfigured for its normal operation. In contrast, conventional BIST approaches introduce both area overhead (typically between 10 and 30 percent) and delay penalties (typically two to three gate delays); the latter may result in speed degradation unacceptable in high-performance systems. The only cost of our technique is the additional memory for storing the data required to reconfigure the FPGA; however, this memory is usually part of the test machine environment (ATE, CPU, or maintenance processor) which controls the BIST sequence, and does not involve resources of the FPGA or the system under test. This approach is applicable to all levels of testing (wafer, packaged device, board, and system). Eliminating the need for adding BIST circuitry (or any design-for-testability logic) to the system logic in FPGAs reduces the design interval and increases the system functionality that can be implemented in each FPGA. Since our BIST is independent of the function implemented in the FPGA, all FPGAs (of the same type) in the system can be tested concurrently; this reduces diagnostic code development and the diagnostic run-time. The initial version of our approach[18] was difficult to implement, because it used a lot of global routing. This problem was overcome in the ILA-based BIST architecture[19], where most signals can be routed locally. However, the test time in the ILA-based BIST is 33% larger than that of the initial version. The first contribution of this paper is to introduce a new BIST architecture, which trades off a limited amount of global routing for a 33% reduction in the test time.

Diagnosis consists of mapping an incorrect response from the circuit under test into the defect(s) that can explain the obtained response. The required diagnostic resolution depends on the goal of the testing process. In system-level testing, the objective is to locate a replaceable defective component. Thus in-system identification of a faulty FPGA is

[1] This material is based upon work supported in part by the National Science Foundation under Grant No. MIP-9409682.

0-7803-4209-7/97 $10.00 © 1997 IEEE

sufficient in this context. However, in an environment where repair by replacement is not feasible or practical, one can take advantage of the reprogrammability and the regular structure of an FPGA to achieve fault tolerance by repairing the FPGA in place. This process requires the identification of faulty PLBs, which are bypassed and replaced with unused cells by reprogramming the FPGA[1][9][12]. The same resolution (to the level of a faulty PLB) is required for the "node-covering" fault-tolerant technique, where the repair occurs after manufacturing testing and is invisible to the user[4]. Another yield-enhancement technique[6] replaces an entire faulty row (or column) by a spare one, and hence its resolution requirement is only to identify a faulty row (or column). When the goal of testing is the improvement of the manufacturing process, then the most accurate resolution (locating faults inside a PLB) is required to support subsequent failure analysis.

Although accurate and efficient diagnosis is essential for these applications, there has been very little published work regarding diagnosis of faulty PLBs in an FPGA. Usually fault location is not targeted by manufacturing tests, whose goal is fault detection. After an FPGA fails its manufacturing test, one method applies a fault location test that simply propagates signals horizontally and vertically through the array[9]. However, this procedure is not reliable, since often a signal may propagate through a defective block. The method used to check the (non-commercial) FPGAs used in the Teramac custom computer[1] configures each row as a pseudo-random sequence generator and checks the final register contents after a given number of clock cycles against an expected signature. The same procedure is then repeated using column instead of rows, and the faulty cells are located at the intersection of the faulty rows with the faulty columns. However, the test provided to the blocks that form the pseudo-random sequence generator is unlikely to achieve full fault coverage and hence it cannot guarantee complete diagnostic resolution (in general, fault detection is a necessary condition for fault location). In addition, applying the test requires a fault-free finite-state machine in the FPGA, and developing the diagnostics tests is an expensive manual process.

In contrast, our approach is *the first FPGA diagnosis method guaranteed to achieve maximal diagnostic resolution at any required resolution accuracy* - chip, row, column, PLB, and even subcircuit inside the PLB. Our BIST does not require any fault-free core in the FPGA. Our basic fault detection test (determining the pass/fail status of the entire device) provides in-system identification of a faulty FPGA. In general, we can locate a faulty PLB with only two BIST configurations in addition to those used in the basic test. As a result, not only are all PLBs in the FPGA completely tested, but a faulty PLB can be identified with an increase in test time of only 7%. We can also diagnose large classes of multiple faulty PLBs. We used the Lucent Optimized Reconfigurable Cell Array (ORCA)[2] for the initial design and implementation of the BIST-based diagnostic approach, but we emphasize that our technique can be applied to any SRAM-based FPGA, such as Xilinx[13] or Altera Flex 8000[3] series FPGAs.

The remainder of this paper is organized as follows. Section 2 gives an overview of the previous FPGA BIST approach. Section 3 presents the new BIST architecture. Section 4 describes the diagnosis method, and Section 5 discusses results from the successful use of this approach in locating faulty PLBs in manufactured FPGAs. Finally, Section 6 presents our conclusions.

2. Overview of the Previous BIST Approach

The strategy of our FPGA BIST approach[18] is to configure groups of PLBs as test pattern generators (TPGs) and output response analyzers (ORAs), and another group as *blocks under test* (BUTs), as illustrated in Figure 1. The BUTs are then repeatedly reconfigured to test them in all their modes of operation. We refer to the test process that occurs for one configuration as a *test phase*. A *test session* is a sequence of test phases that completely test the BUTs in their various modes of operation. Once the BUTs have been tested, the roles of the PLBs are reversed so that in the next test session the previous BUTs become TPGs or ORAs, and vice versa. Therefore, we need at least two test sessions to test all PLBs in the FPGA. Note that all BUTs are tested in parallel, so that the BIST and BIST-based diagnostic run-time does not depend on the size of the FPGA.

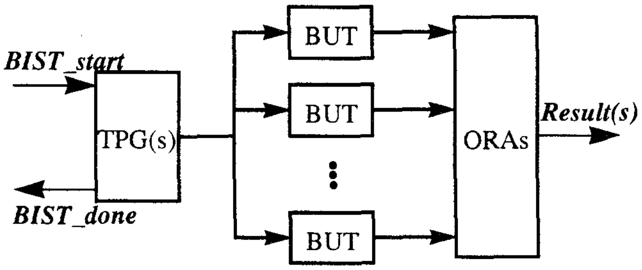


Figure 1. FPGA structure for a BIST session

Each test phase consists of the following steps: 1) reconfigure the FPGA, 2) initiate the test, 3) generate test patterns, 4) analyze responses, and 5) read the test results. In step 1, the test controller (ATE for wafer/package testing; CPU or maintenance processor for board/system testing) interacts with the FPGA(s) under test to reconfigure the logic by retrieving a BIST configuration from the configuration storage (ATE memory; disk) and loading it into the FPGA(s). The test controller also initiates the BIST sequence (step 2) and reads the subsequent results (step 5) using the FPGA's boundary-scan Test Access Port[16] or other system specific means. (Practically all recently developed FPGAs, such as ORCA[2], XC4000[13], and Flex 8000[3], feature boundary scan.) Steps 3 and 4 are concurrently performed by the BIST logic within

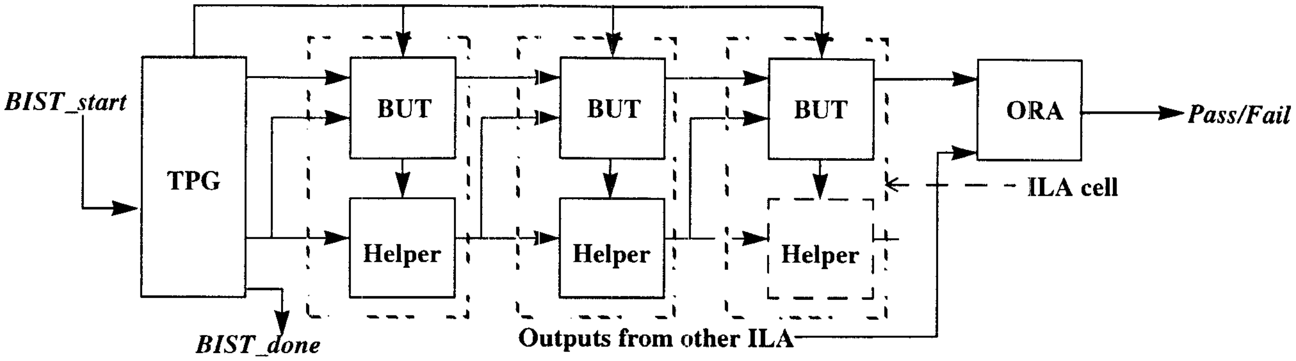


Figure 2. Basic ILA BIST structure with helper cells

the device. After the board or system-level BIST is complete, the test controller must reconfigure the FPGA for its normal system function; hence the normal device configuration must be stored along with the BIST configurations. The test application time is dominated by the FPGA reconfiguration time. Since testing time is a major component of the testing cost, an important goal is to minimize the number of configurations used for test and diagnosis.

Our strategy also relies on *pseudoexhaustive testing*[11], in which every subcircuit of a PLB is tested with exhaustive patterns. This results in maximal fault coverage without explicit fault model assumptions and without fault simulation. For example, all single and multiple stuck-at faults, and all non-feedback bridging faults are guaranteed to be detected. Although complete coverage cannot be guaranteed, the overwhelming majority of *any* other type of faults are also detected. Faults affecting the lookup tables are checked with RAM test sequences which are exhaustive for faults specific to RAMs. Thus *for any practical purpose the PLB test is complete*. Note that in every phase, a BUT is configured in a different mode of operation; hence its pseudoexhaustive test may also change from phase to phase. For example, the test sequence for combinational logic followed by flip-flops is different from the test sequence for a RAM. Thus the TPG block may have different structures depending on the sequence that it has to generate in a given phase.

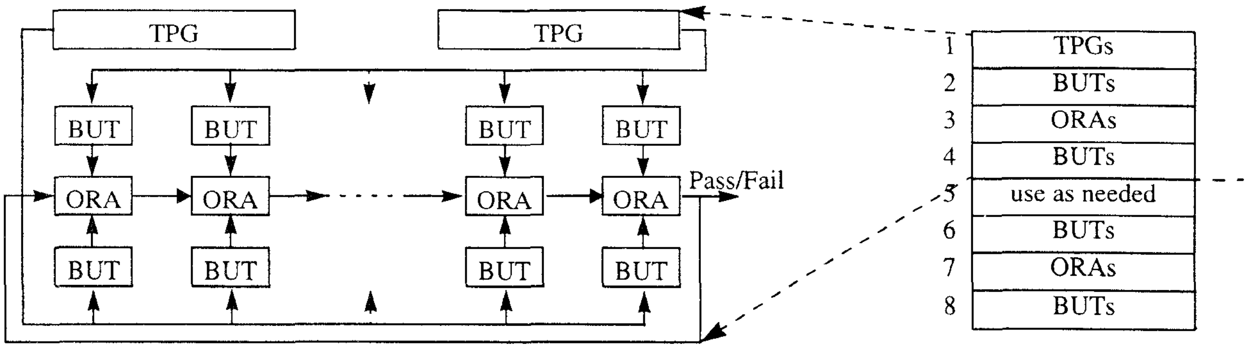
All BUTs are configured to have the same function and receive the same input patterns form the TPG block. Since all fault-free BUTs must produce the same output patterns, the ORAs simply compare corresponding outputs from different BUTs. Unlike the signature-based compression circuits found in most BIST applications, comparator-based ORAs do not suffer from the aliasing problem that occurs when a faulty circuit produces the good circuit signature. As long as the BUTs being compared do not fail the same way at the same time, no aliasing will be encountered with the comparison-based approach. Faulty TPGs and/or ORAs may preclude the detection of a fault in a BUT; but the PLBs composing the faulty TPGs and/or ORAs will become BUTs in a different session, hence a faulty FPGA will not escape detection.

The architecture outlined in Figure 1 uses a lot of global routing, which becomes more difficult to implement as the size of the FPGA increases. This scaling problem is overcome by the ILA-based architecture[19], where most signals are locally routed. Figure 2 shows one ILA spanning two rows of an FPGA. The use of helper cells allows creating ILAs within the FPGA array by providing test patterns to the inputs of the BUT which cannot be driven by the limited number of PLB outputs, and by combining and propagating BUT output responses which cannot be used as test patterns for subsequent BUTs. While most FPGAs structured for BIST based on the original architecture[18] can be tested using only two sessions, the use of helper cells requires a third session to completely test all the PLBs using the ILA-based BIST[19].

3. The New FPGA BIST Architecture

The basic structure of our new BIST architecture, shown in Figure 3a, is a hybrid between the two previous approaches, trading off a limited increase in global routing for saving one test session. Figure 3b outlines the floorplan for the first test session for an 8x8 FPGA; the first four rows correspond with those in Figure 3a. Every ORA compares two BUTs fed by different TPGs. To combine results of several ORAs, we use an iterative comparator based on one proposed by Sridhar and Hayes[15], shown within dotted lines in Figure 4. Here each ORA compares corresponding outputs from two BUTs to produce a local mismatch signal (*LMM*), which is ORed with the previous mismatch signal (*PMM*) from the previous ORA to generate the ORA mismatch (*MM*). The flip-flop is used to record the first mismatch encountered during the BIST sequence. The feedback from the flip-flop output to the first ORA disables further comparisons after the first error has been recorded. Except for this feedback signal, all the other ORA signals propagate like in an ILA, using only local routing resources.

Because the patterns from TPGs feed all BUTs in parallel, the new architecture has more global routing than the ILA-based BIST, where most patterns for a BUT are produced by the previous cell in the ILA. However, this structure



1	TPGs
2	BUTs
3	ORAs
4	BUTs
5	use as needed
6	BUTs
7	ORAs
8	BUTs

a) TPG, BUT, and ORA connections b) Floorplan for first test session

Figure 3. The new BIST architecture

is easily scalable, because the usage of the global routing resources required for distributing the TPG patterns does not change with the FPGA size. In other words, adding rows and columns to an array of PLBs will just extend the size of the vertical and horizontal global lines fed by TPG outputs. Since an ORA compares the outputs of its two neighbor BUTs, all signals from BUTs to ORAs can use local routing resources. Its regular structure allows the new architecture to be constructed and interconnected algorithmically as a function of the size (*N*) of the FPGA.

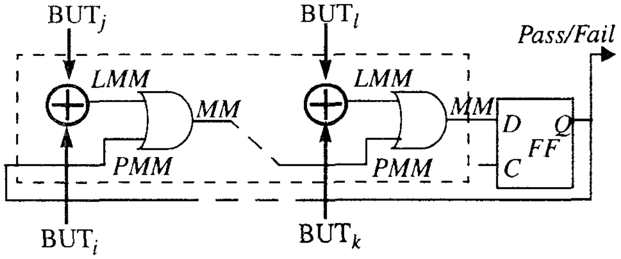


Figure 4. Iterative comparator with error locking

Figure 5 shows the floorplans for the two test sessions - called *NS* and *SN* - that completely test every PLB in an 8×8 FPGA. The name of a session denotes the direction of the flow of test patterns during that session. The floorplan for *SN* is obtained by flipping the floorplan for *NS* around the horizontal axis shown as a dotted line in the middle of the array. The row labeled "used as needed" in Figure 3b can be used for fanout drivers, additional TPGs, additional ORAs, or it may be left unused (note that this row will be configured as BUTs in the second session); in our implementation we used it for ORAs. The important feature of this architecture is that any FPGA can be completely tested in only two test sessions, which is an improvement over the ILA-based approach which required three test sessions. The architectural features that help the diagnosis task will be analyzed in the next section.

1	TPGs
2	BUTs
3	ORAs
4	BUTs
5	ORAs
6	BUTs
7	ORAs
8	BUTs

a) BIST test session *NS*

1	BUTs
2	ORAs
3	BUTs
4	ORAs
5	BUTs
6	ORAs
7	BUTs
8	TPGs

b) BIST test session *SN*

Figure 5. Floorplans for the two BIST sessions

4. BIST-Based Diagnosis

In this section, we describe the use of the BIST approach in diagnosing an FPGA that failed the test provided by the two test sessions described above. We begin by assuming a single faulty PLB in the FPGA and later we discuss the case of multiple faulty PLBs. First, we show that the test results can identify the row in which the faulty PLB reside. Then we present the additional test phases required to locate the faulty PLB.

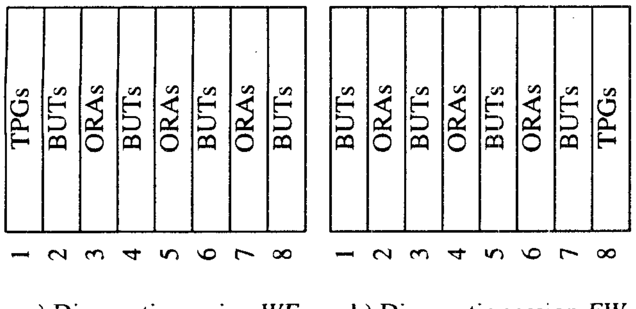
A faulty PLB in a TPG or an ORA may not produce an error if the fault does not affect the operation of the TPG or ORA. First we analyze the case when a fault in an ORA or a TPG PLB is detected only in the session when that PLB is configured as a BUT. One feature of the new architecture that helps diagnosis is that all BUTs (except those in the first and in the last row of BUTs) are always compared by two different ORAs. As a result, a fault in one of the middle rows of BUTs will produce errors at two ORA outputs, while a fault in the first or last row of BUTs will produce an error in only one ORA. Let *Oi* denote the output of the ORA located in row *i*. For example, a defective BUT in row 4 will cause errors at *O3* and *O5* in session *NS*, (because its outputs are compared by ORAs in rows 3 and 5), while a defective BUT in row 1 will be detected only at *O2* in session *SN*. Without this feature we could not distinguish between faulty BUTs in rows 2 and 4, because both would have been observed only at *O3*. The results of this analysis for an 8×8 FPGA are given in Table 1 under the heading "without ORA/TPG failures." Errors at ORA outputs are marked by X. We can observe that the error pattern of every faulty row is unique.

Table 1: Errors during BIST for a single faulty row of PLBs

Faulty Row	Function Session *NS*	Function Session *SN*	without ORA/TPG failures						with ORA/TPG failures					
			Session *NS*			Session *SN*			Session *NS*			Session *SN*		
			O3	*O5*	*O7*	*O2*	*O4*	*O6*	*O3*	*O5*	*O7*	*O2*	*O4*	*O6*
1	TPG	BUT				X			(X	X	X)	X		
2	BUT	ORA	X						X			(X)		
3	ORA	BUT				X	X		(X)			X	X	
4	BUT	ORA	X	X					X	X			(X)	
5	ORA	BUT					X	X		(X)			X	X
6	BUT	ORA		X	X					X	X			(X)
7	ORA	BUT						X			(X)			X
8	BUT	TPG			X						X	(X	X	X)

Now we analyze the case when some faults in a PLB may also be detected when that PLB is configured as an ORA or a TPG. The results of this analysis are given in Table 1 under the heading "with ORA/TPG failures." A fault in an ORA may cause an error only in that ORA when it reports a mismatch although all compared pairs of output values agree. Thus in addition to the error at *O3* in session *NS*, a fault in a PLB in row 2 may also cause an error at *O2* in session *SN*. This error is marked by "(X)" to denote a potential error. A fault in a TPG row may cause the two TPGs to produce different patterns, thus generating mismatches in every comparator and resulting in errors at all ORA outputs in that session; in rows 1 and 8, we use "(X X X)" to denote a potential groups of three errors. Since every row now contains one potential error or one potential group of errors, there are 2^8-1 possible sets of errors. However, it is easy to observe that for any combination of errors, the pattern of every faulty row is still different from all others. Therefore we can conclude that after the two BIST sessions we can accurately locate the row in which the faulty PLB resides.

By repeating the same process after rotating Figure 5 by 90°, so that the flow of test patterns is horizontal instead of vertical (see the diagnostic sessions *WE* and *EW* in Figure 6), the new arrangement will identify a faulty column instead of a faulty row. Then the faulty PLB is located at the intersection of the faulty row with the faulty column.



a) Diagnostic session *WE* b) Diagnostic session *EW*

Figure 6. Floorplans for diagnostic sessions

Although Table 1 shows only the failing sessions, the errors are actually recorded in a specific phase of a session. This allows us to significantly shorten the length of the diagnostic sessions. If we use an *adaptive diagnosis strategy*, in which subsequent tests are applied based on the results obtained so far, we can achieve the same resolution without two complete additional "horizontal" sessions. Namely, we will execute only one of the failing test phases, repeating it twice using the arrangement of Figure 6. In some cases it will be possible to identify the faulty PLB after the first diagnostic test phase without having to run the second diagnostic test phase, but the worst case would require both diagnostic test phases to be performed. Let *p* be the percentage of test time taken by one phase relative to the total time involved in the first two sessions used for detection (for ORCA, *p*=3.5%). Then the percent increase in test time is *p* when only one diagnostic test phase is required, and 2*p* when both diagnostic test phases are required to identify the faulty PLB.

Now we will discuss the diagnosis of multiple faulty PLBs. First it is easy to see that the analysis shown in Table 1 extends trivially for multiple faulty PLBs residing in the same row (or in the same column). Another large class of multiple faulty PLBs that can also be precisely diagnosed are PLBs in rows (or columns) which are observed at disjoint set of ORAs. For example, faults in row 7 can be detected only at *O6* and *O7*, and faults in row 2 can be detected only at *O2* and *O3*; hence any combination of faulty PLBs in rows 2 and 7 can be diagnosed. This class is large, because in an N×N array of PLBs there will be (N/2 - 1) ORA output signals, and faults in row *i* can be observed only at *O(i-1)* and *O(i+1)* for *i* = 3, 4, ..., *N*-2 when *i* is a row of BUTs, and potentially at *Oi* when *i* is a row of ORAs. Most combinations of faulty rows that affect overlapping sets of ORAs can also be diagnosed. Let {*i*, *j*, ...} denote a set of faulty rows. For example, we will show that {2,4} and {4}, which have *O3* as common ORA output, may never have the same pattern of errors. If the faults in rows 2 and 4 do not cause errors at *O3* in exactly the same BIST phases, then {2,4} and {4} can be distinguished at *O3*.

If the faults in rows 2 and 4 affect corresponding BUTs (that are compared by the same ORA), and they always have the same responses, then {2,4} will not have any errors at *O3* while {4} will. Although there exist some multiple faulty PLBs that cannot be accurately diagnosed, it appears that the corresponding fault situations are very unlikely to occur in practice. For example, assume that faulty ORAs and TPGs do not produce failures, and consider {4, 6} and {2, 8}; their error patterns would be identical only if the following conditions occur simultaneously: 1) the faults in rows 4 and 6 affect corresponding BUTs which always have the same responses (this would eliminate the errors at *O5*); 2) faults in rows 2 and 4 cause errors at *O3* in exactly the same BIST phases; 3) faults in rows 6 and 8 cause errors at *O7* in exactly the same BIST phases. Similar analyses show that in general, our BIST architecture imposes very restrictive conditions necessary for multiple faulty PLBs to produce identical error patterns.

Although a single TPG is sufficient for completely testing all PLBs, we can obtain better diagnostic resolution by having two different (but synchronized) TPGs feed the BUTs being compared by the same ORA. Otherwise, a defective single TPG may not supply the patterns needed to detect a fault in a BUT, but this will not cause any mismatch because all BUTs still receive the same patterns. With two separate TPGs, a fault affecting one TPG will cause an error at every ORA, since half of the BUTs will receive test patterns from the faulty TPG.

5. Diagnostic Results for ORCA FPGAs

In this section we present the results of the implementation of our BIST-based diagnostic approach using ORCA FPGAs, and discuss our experience with testing and diagnosis of known defective FPGAs. Our test consists of 14 phases, summarized in Table 2 in terms of the modes of operation of the look-up tables (LUTs) and flip-flop/latch circuits of the PLB tested during every test phase. The first 9 BIST phases are used to test all ORCA series FPGAs, while the last 5 BIST phases are added to test the 2CA series[2]. The number of PLB outputs for each BIST phase is shown in the last column.

The ORCA PLB has five outputs used in many of its modes of operation as a BUT, while only four pairs of outputs can be compared by a single PLB configured as an ORA. As a result, during the BIST test sessions, the "used as needed" row is used to compare the fifth output from up to eight rows of BUTs in those configurations which use all five outputs. The grouping of BUTs and ORAs with respect to the normal four outputs and the extra fifth output is a function of the size of the FPGA as illustrated in Figure 7. The minimum array size for most FPGAs is *N*=8, in which case the "used as needed" row is used to compare the fifth output from four BUTs. As *N* increases, the groups of four outputs from adjacent BUTs are compared pairwise by the rows labeled ORA4, while the fifth outputs are grouped and compared by the rows labeled ORA5. For *N*=16, the basic configuration of Figure 7a can be used twice, either with two sets of TPGs to reduce loading on the TPG outputs, or with an ORA4 in place of the second set of TPGs. While 16<*N*<32, combinations of the arrangements shown in Figure 7 can then be used until the arrangement in Figure 7a is replicated three times for *N*=32. For the second test session, the connection arrangement is rotated about the points indicated in Figure 7, so that every row of PLBs are BUTs in one test session.

Next we consider the BIST phases in which all five outputs of the PLB are tested. As a result of the regular BIST sequence, we determine which phase or phases fail, as well as whether the error is at one of the four outputs or at the fifth output. At that point we run four diagnostic test phases based on the failing output: either a diagnostic test phase for the four

Table 2: Summary of BIST Phases for BUTs

Phase	Flip-Flop/Latch Modes & Options					LUT	No.
No.	FF/Latch	Set/Reset	Clock	Clk Enable	Flip-Flop Data In	Mode	Outs
1	-	-	-	-	-	Asynchronous RAM	4
2	-	-	-	-	-	Adder/subtracter	5
3	-	-	-	-	-	5-variable MUX	4
4	-	-	-	-	-	5-variable XOR	4
5	Flip-Flop	async. reset	falling edge	active low	LUT output	Count up	5
6	Flip-Flop	async. set	falling edge	enabled	PLB input	Count up/down	5
7	Latch	sync. set	active low	active high	LUT output	Count down	5
8	Flip-Flop	sync. reset	rising edge	active low	PLB input	4-variable	4
9	Latch	-	active high	active low	dynamic select	4-variable	4
10	-	-	-	-	-	Multiplier	5
11	-	-	-	-	-	Greater/equal to Comp	5
12	-	-	-	-	-	Not equal to Comp	5
13	-	-	-	-	-	Synchronous RAM	4
14	-	-	-	-	-	Dual port RAM	4

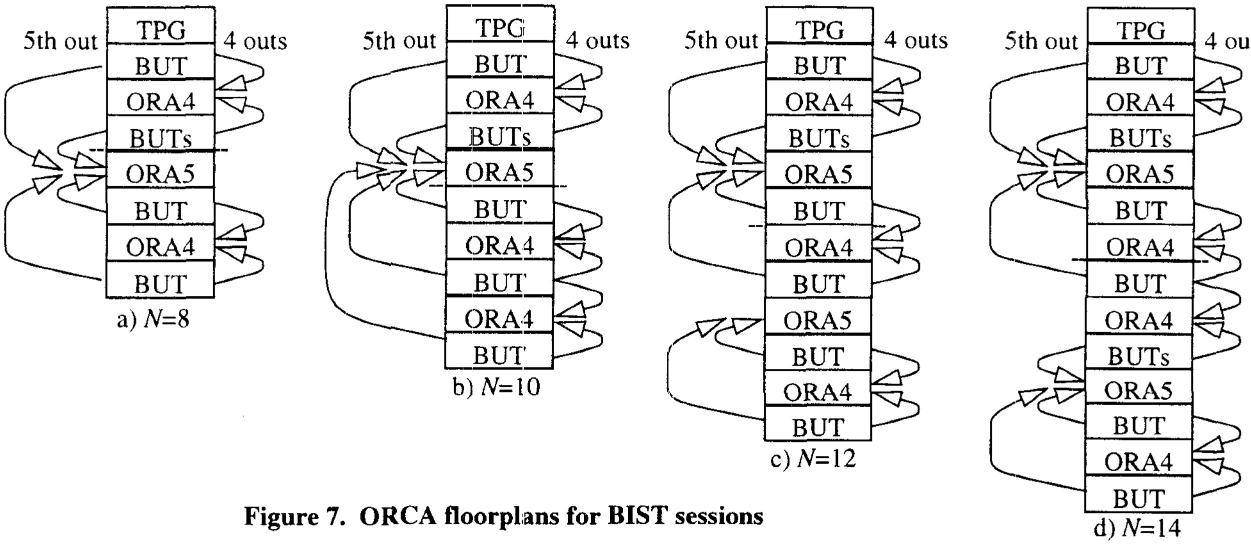


Figure 7. ORCA floorplans for BIST sessions

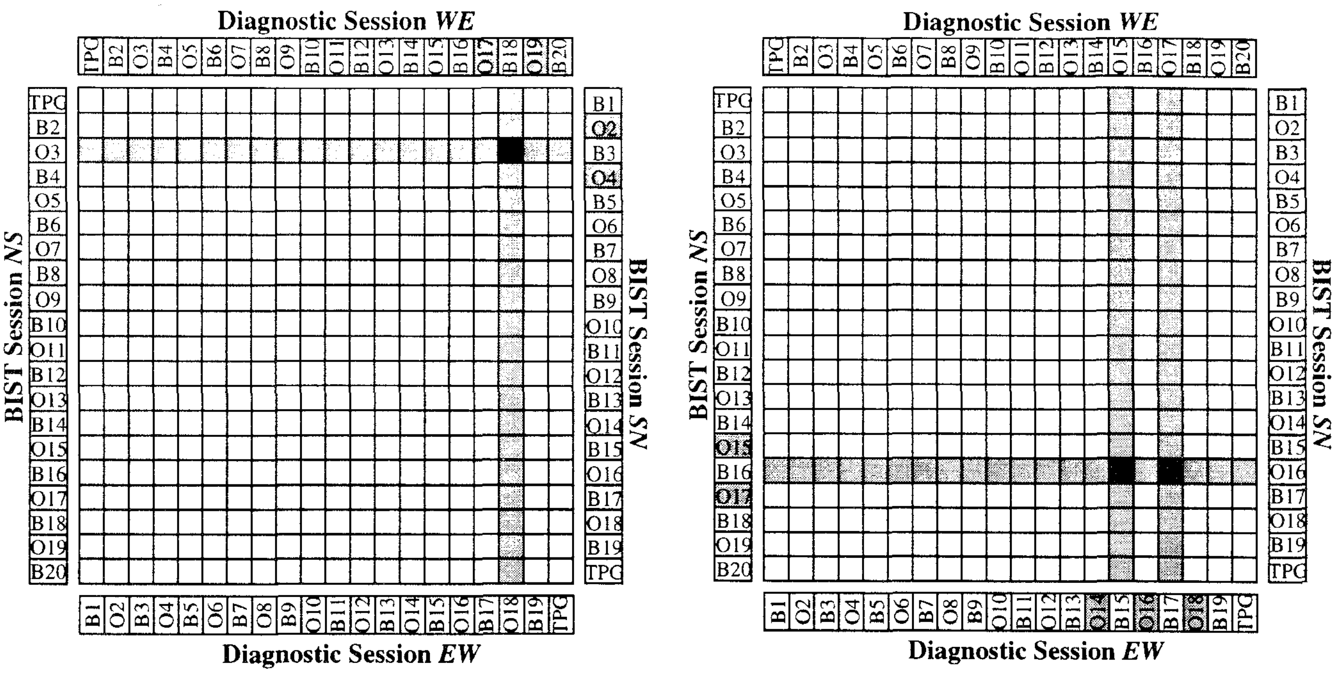
outputs or a diagnostic test phase for the fifth output with ORAs comparing adjacent BUT outputs. Each of the four diagnostic test phases uses a different PLB assignment with respect to rows and columns as illustrated in Figure 5 and Figure 6 (here, Figure 5a and Figure 5b represent diagnostic test phases rather than complete BIST sessions). From these results, we can identify the faulty PLBs in the FPGA in the same manner as was done in the case of the four output phases described above. As a result, at most four additional diagnostic test phases are required (as opposed to two additional diagnostic test phases in the case of the four output phases), for an increase in test time of about 14%.

In addition to the 28 BIST configurations (14 for each test session), we generated two diagnostic test phases for each of the seven BIST phases which test the PLB in operational modes which use only four outputs, for another 14 configurations. We also generated eight diagnostic test phases for each of the seven BIST phases which test the PLB in operational modes that use five outputs, for another 56 configurations. Four of the eight diagnostic test phases are used to diagnose the four PLB outputs in each of the four directions shown in Figure 5 and Figure 6, while the other four diagnostic test phases are used to diagnose the fifth PLB output in each of the four directions. As a result, we generated a total of 98 configurations (28 BIST phases and 70 diagnostic test phases) of which only 30 or 32 configurations will be used to identify a single faulty PLB as a result of the adaptive diagnostic approach.

The device targeted for experimentation with this BIST-based diagnostic approach was the ORCA 2C15A which requires the full set of 14 test phases for each test session to completely test all 400 PLBs in the 20x20 array. We were provided with five 2C15A devices by Lucent Technologies Microelectronics Group in Allentown, PA. From manufacturing test results, three of these parts were known to be fault-free and two were known to be defective. Applying the complete BIST sequence (two test sessions of 14 phases each), we successfully identified the defective FPGAs. Then, using the diagnostic phases, we attempted to identify the faulty PLB(s) in the two failing devices. The BUT-ORA interconnections for the 2C15A consisted of two sets of connections shown in Figure 7b, but with the second set of TPGs replaced by an ORA4 row. As a result, there were nine ORA outputs for each test configuration of the 2C15A with the 5th and 15th rows comparing the fifth PLB outputs during the first BIST session, and the corresponding rows/columns used in the second BIST session and the subsequent diagnostic phases.

Figure 8a summarizes the diagnosis results for the first faulty device. We had no errors in session *NS* and errors at *O2* and *O4* in phases 5 through 9 of session *SN*. These results indicate that row 3 is faulty, and that its faults are not detected when row 3 is an ORA. Reapplying one of the faulty phases twice as diagnosis sessions, we obtained no errors in session *EW*, and errors at *O17* and *O19* in session *WE*. These results indicate that row 18 is faulty, and that its faults are not detected when row 18 is an ORA.When the results of the BIST phases and diagnostic test phases are combined, we identify a single faulty PLB in row 3 and column 18 of the 20x20 array. From Table 2, we can also infer that the fault is probably located in the flip-flop/latch logic of the defective PLB, because all the failing phases (5 through 9) test this subcircuit.

For the second faulty device, whose results are summarized in Figure 8b, we had errors at *O15* and *O17* in phases 1 and 6 of session *NS* and no errors in session *SN*. During the corresponding diagnostic test phases, we obtained errors at *O14*, *O16*, and *O18* in session *EW*, and no errors in session *WE*. This combined set of error patterns identifies two faulty PLBs: one in row 5 and column 15 and the other in row 5 and column 17. However, the second faulty device also had a number of additional errors which pointed to faults in the programmable routing network, which our current test does not


Diagnostic Session WE
BIST Session NS
BIST Session SN
Diagnostic Session EW
TPC B2 O3 B4 O5 B6 O7 B8 O9 B10 O11 B12 O13 B14 O15 B16 O17 B18 O19 B20
B1 O2 B3 O4 B5 O6 B7 O8 B9 O10 B11 O12 B13 O14 B15 O16 B17 O18 B19 TPC

a) ORCA 2C15A with one faulty PLB

b) ORCA 2C15A with two faulty PLBs

Figure 8. Results of BIST-based diagnostic sessions for two faulty ORCA 2C15A devices

explicitly target. This was apparent from errors which occurred in only one of the four directions and can be explained by different routing resources being used for the different directions of the test (with the faulty routing segments used in the one direction only). Although this diagnostic approach is currently targeted only at PLBs, we could use the errors produced by routing segment faults to narrow down the possible area where these faults may reside to an area about one-eighth of the total area of the array.

6. Conclusions

In this paper, we have described a new BIST approach for programmable logic blocks in SRAM-based FPGAs. The new architecture facilitates the complete testing of all PLBs in only two test sessions, while its regular structure allows easy scalability with the size of the FPGA and algorithmic generation of placement and routing data for every test phase based on the size of the array. The main contribution of this paper is to show how our new BIST architecture can be used to locate faulty PLBs in a defective FPGA. The main result is the development of the first method able to achieve maximal diagnostic resolution at any required resolution accuracy - chip, row, column, PLB, and even subcircuit inside the PLB. Our technique can accurately diagnose any single PLB and most multiple faulty PLBs. The multiple faulty PLBs that cannot be diagnosed appear to be very restrictive situations unlikely to occur in practice.

Our approach can diagnose the faulty PLB(s) with a fixed test of four BIST sessions. Using adaptive diagnosis, we need only two additional diagnostic test phases after the first two BIST sessions which detect faulty PLBs. For ORCA, diagnosis increases the test time by only about 7%. We applied our BIST-based diagnosis to two known defective FPGAs, and we were able to identify the faulty PLBs in both devices. In addition, in one device we could identify the defective subcircuit inside the faulty PLB, and in the other one we determined a region where a routing faults is likely to exist.

In addition to testing of PLBs, testing of an FPGA also involves testing of its programmable interconnect network[10]. Accurate identification of the defective wire segments is required for repair either by rerouting an already programmed FPGA[14], or by the "segment covering" technique used after manufacturing testing[5]. Testing and diagnosis of the programmable interconnect will be the next phase of this project.

Acknowledgments

This work was also supported in part by grants from the University of Kentucky Center for Robotics and Manufacturing Systems and from Lucent Technologies, Inc. The authors gratefully acknowledge the support, assistance, and encouragement of C.T. Chen, Al Dunlop, Satwant Singh, and Carolyn Spivak of Lucent Technologies, Inc.

References

[1] B. Culbertson *et al.*, "Defect Tolerance on the Teramac Custom Computer," *Proc. 5th Annual IEEE Symp. on*

Field-Programmable Custom Computing Machines, pp. 140-147, April 1997

[2] *Field Programmable Gate Arrays Data Book*, Lucent Technologies, Oct. 1996

[3] *Flex 8000 Programmable Logic Device Family*, Data Sheet, Altera Corp., May 1993

[4] F. Hanchek and S. Dutt, "Node-Covering Based Defect and Fault Tolerance Methods for Increased Yield in FPGAs," *Proc. 9th International Conf. on VLSI Design*, pp. 225-229, 1996

[5] F. Hanchek and S. Dutt, "Design Methodologies for Tolerating Cell and Interconnect Faults in FPGAs," *Proc. International Conf. on Computer Design*, 1996

[6] F. Hatori *et al.*, "Introducing Redundancy in Field Programmable Gate Arrays," *Proc. IEEE Custom Integrated Circuits Conf.*, pp. 7.1.1-7.1.4, 1993

[7] W. K. Huang and F. Lombardi, "An Approach to Testing Programmable/Configurable Field Programmable Gate Arrays," *Proc. IEEE VLSI Test Symp.*, pp. 450-455, 1996.

[8] C. Jordan and W. P. Marnane, "Incoming Inspection of FPGAs," *Proc. European Test Conf.*, pp. 371-377, 1993.

[9] J. L. Kelly and P. A. Ivey, "Defect Tolerant SRAM Based FPGAs," *Proc. International Conf. on Computer Design*, pp. 479-482, 1994

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

[11] E. McCluskey, "Verification Testing - A Pseudoexhaustive Test Technique," *IEEE Trans. on Computers*, Vol. C-33, No. 6, pp. 541-546, June, 1984

[12] J. Narasimhan *et al.*, "Yield Enhancement of Programmable ASIC Arrays by Reconfiguration of Circuit Placements," *IEEE Trans. on CAD*, Vol. 13, No. 8, pp. 976-986, August 1994

[13] *The Programmable Logic Data Book*, Xilinx, Inc., 1993

[14] K. Roy and S. Nag, "On Routability for FPGAs Under Faulty Conditions," *IEEE Trans. on Computers*, Vol. C-44, No. 11, pp. 1296-1305, Nov. 1995

[15] T.Sridhar and J. P. Hayes, "Design of Easily Testable Bit-Sliced Systems," *IEEE Trans. on Computers*, Vol. C-30, No. 11, pp. 842-854, November, 1981

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

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

[18] C. Stroud, S. Konala, P. Chen, and M. Abramovici, "Built-In Self-Test for Programmable Logic Blocks in FPGAs (Finally, A Free Lunch: BIST Without Overhead!)", *Proc. IEEE VLSI Test Symp.*, pp. 387-392, 1996

[19] C. Stroud, E. Lee, S. Konala, and M. Abramovici, "Using ILA Testing for BIST in FPGAs", *Proc. IEEE International Test Conf.*, pp. 68-75, 1996