Analysis and Evaluation of Routing BIST Approaches for FPGAs

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

Abstract: We present stuck-at and bridging fault simulation results for previously proposed Built-In Self-Test (BIST) approaches for the programmable interconnect resources in Field Programmable Gate Arrays (FPGAs). In addition, new BIST approaches are proposed and analyzed via fault simulation. The fault simulation results are used to compare and evaluate the fault detection capabilities and effectiveness of these BIST approaches for testing routing resources in FPGAs.[1]

1. Introduction

FPGAs have many wire segments and programmable switches that make up its programmable interconnect network and typically account for 80% of the total configuration bits in an FPGA. Ensuring the fault-free status of the interconnect network is a difficult task. The adoption of BIST for programmable interconnect resources in FPGAs has reduced the difficulty by allowing all testing to take place within the FPGA itself. Even with BIST, FPGA programmable interconnect testing still proposes several challenges [1]. The most obvious challenge is the sheer number of wire segments and switches that have to be tested. Since every wire segment and programmable switch has to be checked for faults, the number of test configurations needed to test an entire FPGA can become large. For this reason, developing a routing BIST approach that will maximize the number of wires under test (WUTs) per test configuration will, in turn, minimize the total number of test configurations required. The other challenge of interconnect testing is minimizing the size of the self-test structure. By reducing the area of each self-test structure, a detected fault can be diagnosed to a smaller region of the FPGA [1]. Meeting these challenges of interconnect testing ultimately relies on the architectural constraints imposed by the FPGA.

In this paper, we present a comparative analysis of routing BIST approaches for the programmable interconnect resources of FPGAs that have been utilized in prior routing BIST work. Based on this analysis, we also propose BIST approaches for current and future work. We then evaluate our analysis to determine the best approach for the Xilinx Virtex-4 FPGA routing BIST. We begin with a simple summary of FPGA interconnect resources and an introduction to routing BIST methodology in Section 2. In Section 3, we present the different routing BIST approaches that have been used in previous FPGA interconnect testing. In Section 4, we discuss new approaches we are considering for Virtex-4 routing BIST. Our methodology for simulating stuck-at and bridging faults is discussed in Section 5 along with the fault simulation results. The paper concludes in Section 6 with a summary of our current implementations in Xilinx Virtex-4 FPGAs.

2. Interconnect Fault Models

The programmable interconnect network of an FPGA is comprised of many wire segments that can be connected and disconnected by programmable interconnect points (PIPs) [2]. There are several types of PIPs that make up a FPGA: the break-point PIP, the cross-point PIP, and the multiplexer (MUX) PIP. A PIP itself is composed of a transmission gate which is controlled by a configuration memory bit. Most of the interconnect network of current FPGAs are comprised solely of MUX PIPs. The wire segments these PIPs control are considered to be either local or global routing resources. The local routing resources are associated with a single programmable logic block (PLB). They are used to either connect the PLB to an adjacent PLB or to the global routing resources. The global routing resources connect I/O blocks and non-adjacent PLBs.

The fault models that are used in FPGA programmable interconnect testing consist of wire segments stuck-at 0 or 1, shorted wires, and open wires [1]. Stuck-open (stuck-off) and stuck-closed (stuck-on) PIP faults are also considered in interconnect testing as they are also able to detect stuck-at faults in the configuration memory bits that control PIPs. It should be noted that in this paper we consider only hard faults and do not consider delay faults [4] in the programmable routing resources.

Hard fault detection in FPGA programmable interconnect follows a straightforward methodology for every applied test [1]. Every wire segment and PIP tested must be able to transmit both a 0 and a 1. Each pair of wire segments that could possibly short must be tested for transmission of both (0, 1) and (1, 0) pairs. Applying both logic values at one end of a wire and observing those values at the other end can be used to detect any stuck-at fault on any wire segment to which the test is applied. This also detects any open or stuck-open fault for any type of activated PIP along the path. Applying different logic values at the ends of wire segments connected by a deactivated PIP while observing both sides of the PIP will detect a stuck-closed fault in that PIP. The MUX PIPs that comprise most of the FPGA programmable interconnect need a separate test configuration for every one of its inputs [1]. Logic 0 and 1 values are applied on the input that is selected while opposite values are applied on non-selected inputs. This reveals if a stuck-open fault exists in the PIP that connects the

[1] This work was sponsored by the National Security Agency under contract H98230-04-C-1177.

input that was selected with the output of the MUX PIP. It also reveals if a stuck-closed fault exists on PIPs that are associated with the non-selected inputs.

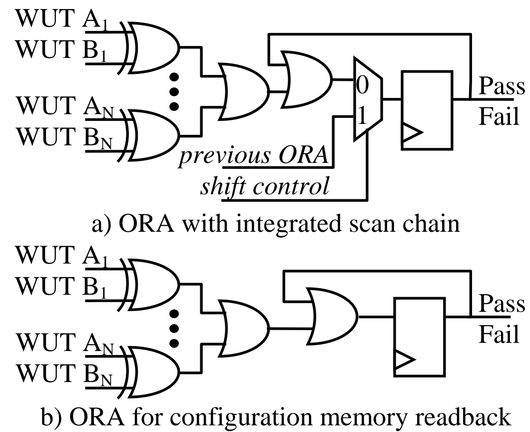
Shorts between wires, also know as bridging faults, can be modeled with wired-AND/OR, dominant, or dominant-AND/OR fault models, with the latter being the most accurate but also the most difficult to test [3]. It is important to note that we have observed both dominant and dominant-AND bridging fault behavior in actual FPGAs but have never observed wired-AND/OR bridging fault behavior. As a result, in this paper we only consider dominant and dominant-AND/OR bridging faults in addition to stuck-at gate-level faults.

3. PREVIOUS ROUTING BIST APPROACHES

Testing the programmable interconnect network of FPGAs is a difficult process as can be seen in Table 1 where the number of test configurations required to test the programmable routing is compared with that for PLBs for a number of different FPGAs. Throughout the previous work in routing BIST, several approaches have been taken and assumptions have been made in order to achieve adequate fault coverage for different FPGA architectures. In general, the goal has been minimizing the number of test configurations by maximizing the number of wires under test in any give test configuration. Since the test pattern generators (TPGs) and output response analyzers (ORAs) are constructed from PLBs and since routing resources must be used to interconnect the logic to form TPGs and ORAs, a common assumption has been that the routing is assumed to be fault-free when testing PLBs, and that the PLBs are fault-free when testing routing. Another assumption has been knowledge of the relative physical positions of wire segments with respect to each other when considering possible bridging fault sites. These assumptions have their benefits, but also have several shortcomings. The following subsections discuss some of these approaches and assumptions in more detail.

Another important consideration is the method by which test results are retrieved. There are several methods for retrieving the ORAs contents [3], but two methods were most frequently used in the previous routing BIST approaches. In the previous BIST approaches for ORCA [1] and Atmel [6] FPGAs, a scan chain was incorporated in the ORAs to scan out the test results stored in the ORAs at the end of the BIST sequence. This method of results retrieval, illustrated in Figure 1, can require additional logic resources for the multiplexer and routing resources to construct and to control the scan chain. The additional logic and routing resources required for the integrated ORA scan chain limit the amount of routing resources that can be tested in one configuration. This requires more configurations to fully test the FPGA unless, as is the case in the Cypress Delta39K, there is a built-in scan chain associated with the flip-flops [7].

In the BIST approach for Xilinx 4000 and Spartan series FPGAs [8], configuration memory readback was used to retrieve ORA results. This requires reading the entire configuration memory for each BIST configuration to retrieve the ORA flip-flop contents. This effectively doubles the test time since the configuration memory readback time is about the same as the test configuration download time; the BIST execution time is insignificant by comparison. However, this reduces the number of configurations needed to test the FPGA since more wires under test can be configured for each test configuration. In newer FPGAs, such as Xilinx Virtex FPGAs, partial configuration memory readback capabilities can be used to only read the portions of the configuration memory that contain ORA flip-flops contents [9].



a) ORA with integrated scan chain

b) ORA for configuration memory readback

Figure 1. Comparison-based ORAs

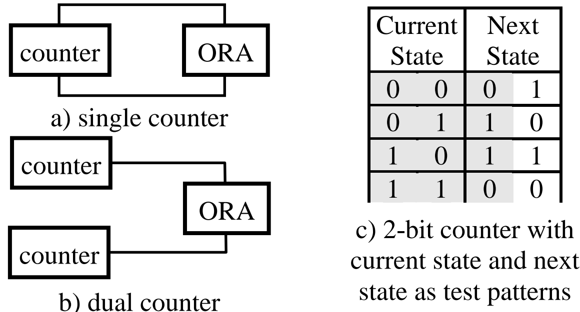
Table 1. Previous FPGA Test Configurations

FPGA		# Test Configurations		
Vendor	Series	PLBs	Routing	Ref
Lattice	ORCA 2C	9	27	[2]
	ORCA 2CA	14	41	[1]
Atmel	AT40K/AT94K	4	56	[6]
Cypress	Delta39K	20	419	[7]
Xilinx	4000E/Spartan	12	128	[8]
	4000XL/XLA	12	206	
	Virtex/Spartan-II	12	283	[9]

3.1 Comparison-Based Counter Approaches

The first BIST approach for FPGA interconnect resources used counter-based TPGs to generate test patterns that were transmitted across the wires under test and compared by the ORAs [2]. Depending on the routing resources being tested, either a single counter or dual counters were used as illustrated in Figure 2a and Figure 2b, respectively. This approach was used for ORCA FPGAs for both on-line and off-line testing [1] as well as for 4000 series and Spartan FPGAs [8] and later for Virtex and Spartan II FPGAs [9]. This ap-

proach was also used in [10]. In ORCA, Virtex, and Spartan II FPGAs, 4-bit counters were used since the PLBs contained four flip-flops. However, a 2-bit counter approach was used in the 4000 series and Spartan FPGAs since the PLBs contained only two flip-flops [8]. In this approach, the current state and next state of a 2-bit counter provided a set of four signals with opposite logic values between any two pair of signals during the four vector test sequence, as can be seen in Figure 2c. In all of these approaches, there were eight wires under test for each set of TPGs and ORAs.



Current State		Next State	
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

c) 2-bit counter with current state and next state as test patterns

Figure 2. Counter-Based Approaches

3.2 Parity-Based Approaches

The original parity-based routing BIST approach was proposed for Xilinx 4000 series FPGAs in [11]. In this approach, a counter-based TPG sourced exhaustive test patterns over a set of *N* wires under test and produced a parity bit that is also sent to the ORA. The ORA performed a parity check function to detect faults. One problem with the original approach was the assumption that the parity bit was routed over fault-free interconnect resources. However, the faulty/fault-free status of the routing resources is unknown at the start of testing.

This parity-based approach was later modified in [6] to: 1) send the parity bit over a wire under test for a total of *N*+1 wires under test, as illustrated in Figure 3, and 2) incorporate multiple types of TPGs and ORAs. For example, an up-counter producing even-parity was combined with a down-counter producing odd parity. These complementary TPGs produced opposite logic values needed to detect PIP stuck-on faults and bridging faults. The parity approach is particularly efficient for FPGAs with small PLBs. For example, a 2-bit up-counter generates even parity as the next state for the most significant bit, as illustrated in the shaded entries of Figure 2c. This approach was used for Atmel AT40K and AT94K series FPGAs where 2-bit up-counters initialized to all 0s and generating even parity were combined with 2-bit down-counters initialized to all 1s and generating odd parity [6]. The count and parity values were applied to opposite sides of deactivated PIPs to detect stuck-closed faults in those PIPs while the opposite logic values produced between any pair of signals of the count value and/or parity bit detected bridging faults between adjacent wire segments. In this implementation, there were six wires under test for each set of TPGs and ORAs.

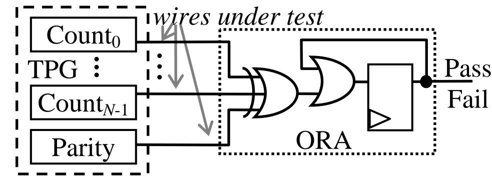


Figure 3. Parity-Based Approach

3.3 Previous Routing BIST Assumptions

Aside from the assumption of fault-free routing for the parity bit in the original parity-based BIST approach, the common assumptions used in all of these previous routing BIST approaches included:

1. The logic resources used to construct the TPGs and ORAs had been previously tested and found to be fault-free.
2. Routing resources for the feedback connections in the TPG counters were also fault-free.
3. A knowledge existed of the relative position of wire segments to be tested to ensure that the same signal was not transmitted over adjacent wire segments.

Assumptions 1 and 2 are essential for the single counter comparison-based and the parity-based routing BIST approaches. This is because a logic fault which prevents the counter from counting will go undetected along with any faults in the programmable interconnect resources that form the wires under test. This is not a serious problem when using configuration memory readback to retrieve the contents of the ORAs at the end of the BIST sequence ***if*** we also read the current state of the TPG to see that it is in the proper state after having advanced the state through the complete BIST sequence and passed the starting state. This is also not a problem in the dual counter comparison-based approach since the ORA will detect any differences in the states of the two counters.

Assumptions 2 and 3 were reasonable for the targeted FPGAs which were relatively small by today's standards. All of those FPGAs had dedicated feedback routing. Furthermore, they had relatively few wire segments and PIPs per PLB; the number of wire segments ranged from 42 to 77 while the number of PIPs ranged from 128 to 206 [8]. As a result, information regarding the relative physical positions of wire segments could be obtained from data sheets and graphical editors for physical design. However, most recent, complex FPGAs do not incorporate dedicated feedback connections and have as many as 406 wire segments and 4,100 PIPs per PLB [12]. In addition, there is little, if any, information about the relative position of wire segments in the data sheets, with only vague and questionable information in graphical editors.

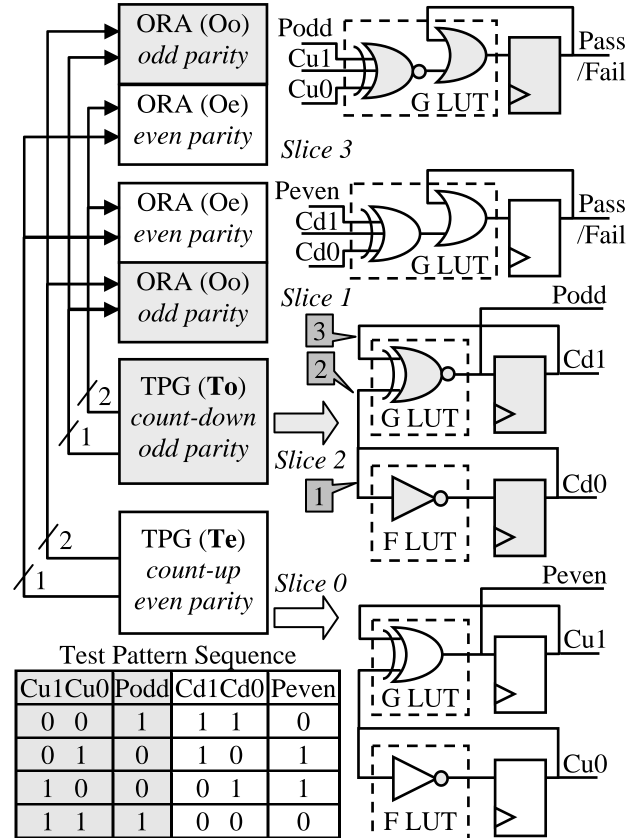
4. NEW ROUTING BIST APPROACHES

With recent FPGA architectures becoming much more complex, the feasibility of testing routing resources based on the assumptions used by previous BIST approaches for the interconnect network is becoming unrealistic. The absence of dedicated feedback routing resources for the construction of TPGs means that one cannot consider the TPG to be fault-free. This is specifically the case for Xilinx Virtex-4 FPGAs where feedback routing for counters, etc., uses the same programmable interconnect resources as being considered for wires under test. Furthermore, the vast majority of the routing resources as constructed from buffered directional multiplexer PIPs as opposed to bi-directional break-point, cross-point, and compound cross-point PIPs as was the case in the earlier FPGAs targeted by previous BIST approaches. Therefore, we investigated new routing BIST approaches that can eliminate many, if not all, of the assumptions used by previous routing BIST approaches. These BIST approaches include cross-coupled parity and cellular automata register (CAR) based approaches discussed in the following subsections.

4.1 Cross-Coupled Parity

The parity-based routing BIST approach has the advantage over the comparison-based counter approach in that it supports an odd number of wires under test and the wires under test are easier to route [3]. However, a fault that inhibits the count sequence can go undetected if the parity remains the correct sense. This fault can be detected by reading the current state of the counter along with the ORAs via using partial configuration memory readback. However, by cross-coupling the parity from dual counters, any fault affecting one of the counters will be detected by the ORAs. This cross-coupled parity architecture is illustrated in Figure 4 for implementation in a Virtex-4 PLB which consists of four slices, each containing two flip-flops and two 4-input look-up tables (LUTs). In this case, a 2-bit up-counter initialized to all 0s and a 2-bit down-counter initialized to all 1s drive two ORAs each, where each ORA consists of a single LUT and flip-flop, as illustrated in Figure 4. The next state of the most significant bit of each counter is cross-coupled to the ORAs receiving count values from the other counter. Hence, the even parity (Peven) produced by the up-counter also provides even parity for the even parity ORA (denoted Oe in Figure 4), and vice versa. The resultant test pattern sequence is summarized in the table in Figure 4.

This cross-coupled parity arrangement also allows the routing resources constructing the feedback paths in the counter to be considered as wires under test. This includes the feedback of each counter bit to its own driving LUT as well as the feed forward path to the next bit of the counter. As a result, each counter incorporates three wires under test in its construction (indicated by the highlighted call-out boxes for the down-counter in Figure 4) as well as three wires under test to each ORA that it drives. Therefore, there are a total of 18 wires under test for the architecture shown in Figure 4, which corresponds to a single Virtex-4 PLB.



Cu1Cu0	Podd	Cd1Cd0	Peven
0 0	1	1 1	0
0 1	0	1 0	1
1 0	0	0 1	1
1 1	1	0 0	0

Figure 4. Cross-Coupled Parity-Based BIST

4.2 Counters/LFSRs as Combined TPG/ORA

Since the feedback routing of a counter can be considered as wires under test, then a counter itself could be used as a combined TPG and ORA. At the end of the BIST sequence, the current state of the counter is read to determine the pass/fail status of the BIST sequence. Of course, this requires that the final state of the counter be different from that of the starting state and may require proceeding through the complete count sequence, rolling over, and moving on to some particular count value. However, when we investigated this approach we found that at least two states of the counter must be read in order to achieve high fault coverage. Another possible TPG/ORA combination is the linear feedback shift register (LFSR). Unfortunately, the LFSR has fewer feedback paths compared to a counter. Therefore, we investigated the cellular automata register (CAR) as a potentially more attractive candidate for a combined TPG/ORA implementation.

4.3 Cellular Automata Register

An 8-bit CAR can achieve a maximum length sequence (2^8-1 = 255) using any of the sets of rules given in Table 2 with null boundary conditions [3]. The CA rules 90 and 150 for the ith bit of the register are given in Table 3. Null boundary conditions simply substitute a logic 0 for the missing bits at the ends of the register as summarized in Table 3 where bits 1 and *N* are assumed to be the least significant bit (LSB) and most significant bit (MSB) of the CAR, respectively.

Obviously, the more CA 150 rules in the implementation of the CAR, the larger number of wires under test since each CA 150 rule implies three wires under test while each 90 rule implies only two wires under test. For the null boundary conditions, the CA 150 rule implies only two wires under test with only a single wire under test for the CA 90 rule. Therefore, the top two sets of rules in Table 2 provide the largest number of wires under test (19 for an 8-bit CAR) in a given BIST configuration, as illustrated in Figure 5.

We found, via fault simulation analysis, that by initializing either CAR to all 1s and applying 18 clock cycles, we obtain excellent fault coverage with only one read of the current state of the CAR. Furthermore, the current state of the CAR can be read at the end of the subsequent 18 clock cycle sequence to obtain equally high fault coverage. The contents of the CAR at the end of each sequence of 18 clock cycles are given in Table 4 for the first two CAR implementations in Table 2. This process can be continued for at least three sets of 18 clock cycles, but some subsequent sets of 18 clock cycles produce lower fault coverage. The importance of this will be discussed in more detail later in the paper.

We also investigated an 8-bit CAR using all CA 150 rules with cyclic boundary conditions, which would allow for 24 wires under test. For any given starting value, this CAR generates at most only four unique test patterns before it begins to repeat. As a result, the bridging fault coverage obtained is inferior to that of the maximum length sequence CAR, as will be shown in the next section, because the four test patterns generated have pairs of bits that have the same logic value.

Table 2. 8-bit CAR Rules for Maximum Length

Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7	Bit8	#150 rules
150	150	90	150	90	150	90	150	5
150	90	150	90	150	90	150	150	5
90	90	150	90	150	90	150	90	3
90	150	90	150	90	150	90	90	3

Table 3. CAR Rule Implementations

Rule	Bit 1 (LSB)	i^{th} Bit	Bit *N* (MSB)
150	$Q_1^+ = Q_1 \oplus Q_2$	$Q_i^+ = Q_i \oplus Q_{i-1} \oplus Q_{i+1}$	$Q_N^+ = Q_N \oplus Q_{N-1}$
90	$Q_1^+ = Q_2$	$Q_i^+ = Q_{i-1} \oplus Q_{i+1}$	$Q_N^+ = Q_{N-1}$

Table 4. 8-Bit CAR 18 Clock Cycle Sequences

Sequence	Table 2 Entry 1	Table 2 Entry 2
1st 18 cycles	10110111	11101101
2nd 18 cycles	10011111	11111001

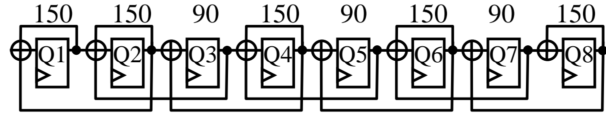


Figure 5. Maximum Length Sequence 8-Bit CAR

5. Experimental Results

We analyzed the various BIST approaches described in the previous two sections for their gate-level stuck-at fault coverage and for their bridging fault coverage. The stuck-at fault simulation not only evaluates the ability to detect stuck-at values on the wires under test, but also provides considerable insight into other aspects of the BIST approaches. These include the validity of the assumption that the logic resources have been tested as well as what routing resources can be considered as wires under test for any given routing BIST approach.

5.1 Stuck-At Fault Simulation Analysis

The collapsed gate-level stuck-at fault coverage for the various BIST approaches is summarized in Figure 6. The fault coverage is given for the entire BIST circuit including TPG, wires under test, and ORA (denoted "w/ ORA" in the figure). In addition, Figure 6 gives the fault coverage without considering the faults associated with the ORAs (TPG and wires under test only) for those circuits that have ORAs. As can be seen, the two CARs obtain complete fault coverage and do not have associated ORAs since the current state of the CAR is used to determine the pass/fail status of the BIST sequence. It should also be noted that the length of the BIST sequence for the maximum length sequence CAR is 18 clock cycles.

By removing the ORA faults, we can see the resultant fault coverage obtained for the TPG and wires under test. As can be seen from the figure, the only faults detected in the single counter comparison-based BIST approach reside in the ORA. Furthermore, the single counter approach achieves the lowest fault coverage of any of the BIST approaches. Fortunately, the dual counter approach was used in most cases in [2], [8], and [1] with the single counter being rarely implemented.

Note that the dual counter and cross-coupled parity approach achieve complete fault coverage in the TPGs and wires under test. This shows that the feedback in the counter can be considered to be wires under test in these BIST approaches, at least in terms of stuck-at faults. It was these results that led us to investigate combined TPG/ORA implementations like counters, LFSRs, and CAR in terms their bridging fault coverage.

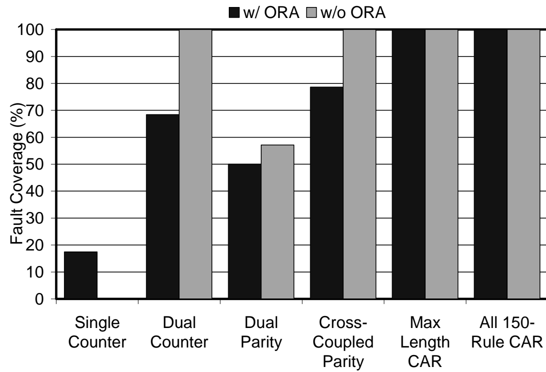


Figure 6. Stuck-at Fault Simulation Results

5.2 Bridging Fault Simulation Analysis

Bridging fault simulations of the wires under test were performed using the dominant fault model and using the dominant-AND/dominant-OR fault models. Unlike stuck-at faults where faults on the fanout stem are treated differently than faults on the fanout branches, bridging faults affect the entire signal net. Therefore, in order to accurately emulate of the buffering effects of the directional multiplexer PIPs used to construct the routing network, we add buffers to the circuit to provide the same isolation as a PIP. We add one buffer to the fanout stem and one to each fanout branch as illustrated in Figure 7 before performing bridging fault simulation.

The fault simulation results are summarized in Figure 8 where the number of wires under test is given above the bars representing the fault coverage. As can be seen, the fault coverage achieved is almost the same for both dominant and dominant-AND/OR bridging fault models for each BIST approach. It should be noted that this is not always the case since the dominant-AND/OR model is more difficult to detect than the dominant model and, as a result, the dominant-AND/OR fault coverage is typically lower than the dominant fault coverage when both are less than 100%, as in the case of the single counter and dual parity approaches. The single counter and dual parity approaches achieved the lowest fault coverage. However, it should be noted that these BIST approaches were applied to interconnect architectures for which there was knowledge of the relative physical positions of the wires under test such that the test patterns were positioned to achieve high fault coverage in the actual implementation.

In Figure 8, we see that the all-150 rule CAR has inferior fault coverage to the maximum length sequence CAR due to the limited four test patterns generated. Not only does the maximum length sequence CAR achieve the highest bridging fault coverage, it also supports the highest number of wires under test when we disregard the all-150 rule CAR approach. The cross-coupled parity approach achieves fault coverage very close to that of the maximum length sequence CAR and supports almost as many wires under test. As a result, it is also a good candidate for routing BIST implementations. However, the lower stuck-at fault coverage that result from faults not detected in the ORA requires testing the logic resources for the ORA prior to routing BIST.

All of the fault simulations summarized in Figure 8 used the PIP emulation buffers shown in Figure 7 including bridging faults with the fanout stem. To illustrate the importance of emulating the buffered multiplexer PIPs, we performed fault simulations without the buffers for the PIPs. In addition, we performed simulations where the fanout stem was not considered a bridging fault site. These fault simulation results are summarized in Figure 9 for the dominant fault model (the middle bar is the dominant bridging fault simulation results from Figure 8). As can be seen, failure to emulate the buffering effects of the PIPs yields inaccurate and optimistic fault coverage results. While this trend is reversed in the case of the single counter and dual parity approaches, these approaches are not viable candidates for the new, more complex FPGAs on the market due to their lower stuck-at fault coverage and bridging fault coverage.

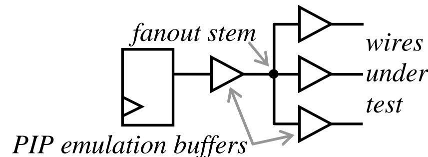


Figure 7. Emulating Buffering Effects of PIPs

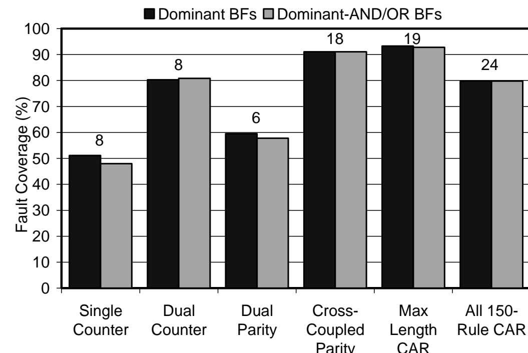


Figure 8. Bridging Fault Simulation Results

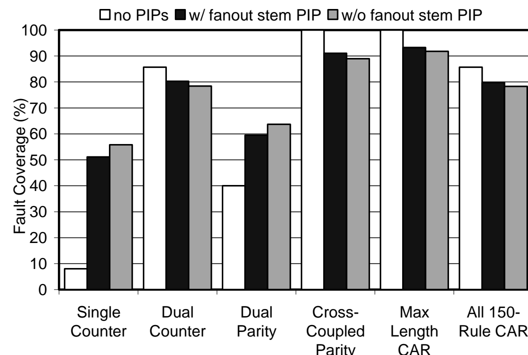


Figure 9. PIP Bridging Fault Simulation Results

6. CONCLUSIONS

We have presented fault simulation results for previous and newly proposed routing BIST approaches. These results clearly show that the cross-coupled parity-based BIST approach and the maximum length sequence CAR approaches provide better stuck-at fault coverage and better bridging fault coverage then any previous routing BIST approach. These BIST approaches eliminate the assumptions used by previous approaches and support more wires under test in a given BIST configuration. As a result, these two BIST approaches provide higher quality testing with fewer test configurations.

Based on these results, we developed a routing BIST configuration generation program for Virtex-4 utilizing the maximum length sequence 8-bit CAR approach. The program generates a routed template file that fills every PLB with an all-150 rule CAR using three inputs to every LUT. We also included an option of which three of the four LUT inputs were used such that in the course of using both options we test all LUT inputs.

The template is then run through a modifier program that initializes the LUTs to their proper rules for implementation of the top two entries in Table 2, such that in one BIST configuration the top entry is implemented with the second entry implemented in the next BIST configuration. This allows a small partial reconfiguration file to be used to reconfigure the LUTs such that 24 wires under test are tested in two consecutive BIST configurations. During the first BIST configuration, the BIST sequence is executed for 18 clock cycles and the contents of the CAR can be read for the pass/fail status of that BIST sequence. Partial reconfiguration is then used to modify the LUT contents for the second CAR implementation and the BIST sequence is executed for another 18 clock cycles with the initial state for the second BIST sequence being the final state of the first BIST sequence. Hence, we obtain 100% stuck-at and bridging fault coverage with only 36 BIST clock cycles and one small partial reconfiguration of the FPGA. Local routing resources are tested by interconnecting the CAR bits within a single PLB to form the 8-bit CAR. Global routing resources are tested by interconnecting CAR bits in different PLBs to form the 8-bit CARs.

We have also implemented a routing BIST configuration generation program for Virtex-4 utilizing the cross-coupled parity-based BIST approach. We are proceeding with implementations of both BIST approaches in our development of BIST for Virtex-4. The remaining step is to determine the sets of wires under test that will completely test the routing resources in a minimum, or near minimum number of BIST configurations. We will continue to evaluate these two BIST approaches and, at some point, may choose one over the other. However, it is also possible that our final implementation of BIST for the Virtex-4 routing resources will consist of a combination of both cross-coupled parity and maximum length sequence 8-bit CAR BIST approaches.

REFERENCES

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

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

[3] C. Stroud, A Designer's Guide to Built-In Self-Test, Boston: Springer, 2002

[4] J. Smith, T. Xia and C. Stroud, "An Automated BIST Architecture for Testing and Diagnosing FPGA Interconnect Faults", *J. Electronic Testing: Theory & Applications*, vol. 22, no. 4, pp. 239-253, 2006

[5] 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.

[6] J. Sunwoo and C. Stroud, "Built-In Self-Test of Configurable Cores in SoCs Using Embedded Processor Dynamic Reconfiguration," *Proc. International System-on-Chip Design Conf.*, pp. 174-177, 2005

[7] C. Stroud, J. Bailey, J. Emmert, D. Nickolic, and K. Chhor, "Bridging Fault Extraction from Physical Design Data for Manufacturing Test Development," *Proc. IEEE International Test Conf.*, pp. 760-769, Oct. 2000

[8] C. Stroud, K. Leach, and T. Slaughter, "BIST for Xilinx 4000 and Spartan Series FPGAs: A Case Study," *Proc. IEEE International Test Conf.*, pp. 1258-1267, 2003

[9] S. Dhingra, S. Garimella, A. Newalkar, and C. Stroud, "Built-In Self-Test for Virtex and Spartan II FPGAs Using Partial Reconfiguration," *Proc. IEEE North Atlantic Test Workshop*, pp. 7-14, 2005

[10] I. Harris and R. Tessier, "Testing and Diagnosis of Interconnect Faults in Cluster-Based FPGA Architectures," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 21, no. 11, pp. 1337-1343, 2002

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

[12] L.T. Wang, C. Stroud, and N. Touba, System-on-Chip Test Architectures: Nanometer Design for Testability, Amsterdam: Elsevier, 2007