Open access • Proceedings Article • DOI:10.1109/ICVD.1999.745159

A low cost approach for detecting, locating, and avoiding interconnect faults in FPGA-based reconfigurable systems — Source link

D. Das, Nur A. Touba

Institutions: University of Texas at Austin

Published on: 10 Jan 1999 - International Conference on VLSI Design

Topics: Automatic test pattern generation, Reconfigurable computing, Programmable logic device and Field-programmable gate array

A Low Cost Approach for Detecting, Locating, and Avoiding Interconnect Faults in FPGA-Based Reconfigurable Systems

Debaleena Das and Nur A. Touba

Department of Electrical and Computer Engineering
University of Texas, Austin, TX 78712-1084
E-Mail: {ddas,touba}@ece.utexas.edu

Abstract

An FPGA-based reconfigurable system may contain boards of FPGAs which are reconfigured for different applications and must work correctly. This paper presents a novel approach for rapidly testing the interconnect in the FPGAs each time the system is reconfigured. A low-cost configuration-dependent test method is used to both detect and locate faults in the interconnect. The "original configuration" is modified by only changing the logic function of the CLBs to form "test configurations" that can be used to quickly test the interconnect using the "walking-1" approach. The test procedure is rapid enough to be performed on the fly whenever the system is reconfigured. All stuck-at faults and bridging faults in the interconnect are guaranteed to be detected and located with a short test length. The fault location information can be used to reconfigure the system to avoid the faulty hardware.

1. Introduction

A field-programmable gate array (FPGA) can be configured in the field to implement a desired logic function. A static RAM based FPGA architecture has a matrix of configurable logic blocks (CLBs), programmable interconnect, and programmable I/O cells. A user can specify a logic function and then use compiler software to map the logic function to a network of CLBs, which are then placed and routed. The end result is a particular configuration for the FPGA that implements the logic function.

One powerful and exciting application of FPGAs is in constructing *reconfigurable systems* (or custom computing machines). The hardware in such systems can be reconfigured on the fly to adapt to different computing requirements for different applications. Reconfigurable systems offer higher computational density and higher throughput for many applications compared with conventional fixed hardware systems. Reconfigurable computing is an active area of research.

An important and challenging issue for reconfigurable systems is reliability. Reconfigurable hardware is inherently less reliable than conventional hardware because of the large amount of additional circuitry needed to support the reconfigurability. Testing reconfigurable hardware to ensure that it is defect-free is substantially more difficult than testing conventional hardware. There are an exponential number of different ways that the programmable hardware can be configured. There is no way to test that all possible configurations are fault-free.

A reconfigurable system may contain several boards of FPGAs. This paper presents a novel approach for rapidly testing the interconnect in the FPGAs each time the system is reconfigured. The method presented here not only detects faults in the interconnect, but also locates the faults so that the system can be reconfigured to avoid the faulty hardware and thus continue fault-free operation.

In order to describe the relationship between the work presented here and previous work in FPGA testing, it is important to make the distinction between *configuration-independent testing* and *configuration-dependent testing*. In configuration-independent testing, no assumptions are made about the way in which the FPGA will be configured by the user. The goal is to maximize the fault coverage for all possible configurations.

Configuration-independent testing is done when the FPGA is manufactured (i.e., before it is shipped to the user). Previous work in FPGA testing has focused on configuration-independent testing. Configuration-dependent testing, on the other hand, involves testing that a particular FPGA configuration is fault-free. A higher fault coverage (for the particular configuration) can be achieved with less test time. The approach described here is a configuration-dependent test technique for interconnect. Configuration-independent techniques for testing FPGA interconnect have been described in [Liu 95], [Huang 96], [Renovell 97], [Zhao 98], and

techniques for diagnosis have been described in [Lombardi 96] and [Culbertson 97].

Three drawbacks of the configuration-independent diagnostic tests are described in [Culbertson 97]:

1. Time it takes to develop the diagnostic tests necessary to locate (not just detect) the faults.
2. Time to develop (through experimentation) test configurations that provide a high coverage.
3. Time to run the tests.

The configuration-dependent test method presented in this paper addresses all three of the problems described above. The method presented here provides a systematic procedure for detecting and locating all stuck-at and bridging faults in the interconnect for a particular FPGA configuration. This is done by modifying the "original configuration" by only changing the logic function of the CLBs to form "test configurations" that can be used to quickly test the interconnect. A systematic procedure is applied to the "original configuration" to generate a small set of test configurations. The logic function of the CLBs in the test configurations are chosen in such a way that the "walking-1" approach can be used to detect and locate all stuck-at and bridging faults in the interconnect with a small set of test vectors. Since only the logic functions of the CLBs are changed, time consuming placement and routing is avoided. The process of generating the test configurations and test vectors is fully automated and very fast. It can be performed on the fly whenever the system is reconfigured.

Since the area of an FPGA is dominated by the programmable interconnect, most faults occur in the interconnect. Thus, thoroughly testing of the interconnect for each configuration used in a reconfigurable system is very important. If an FPGA configuration is found to be faulty, the information provided by the proposed method can be used to avoid the faulty hardware. This can be done by either rerouting the configuration [Roy 95], or by recompiling the design with additional resource constraints [Culbertson 97].

2. Fault Model

The most common failure mode in interconnects is bridging faults between nets [Zhao 98]. Bridging faults can be of wired-AND or wired-OR type. In wired-AND the logic level 0 dominates, while the logic level 1 dominates in wired-OR. Besides bridging faults, stuck-at-0 and stuck-at-1 faults are also significant. Detection of bridging faults can be achieved by the *counting sequence algorithm* [Kautz 74],[Goel 82].

To diagnose bridging faults between two nets, complementary values have to be driven on the nets and each net has to be observed at the output. This can be achieved by setting all the nets to '0' and then walking a single '1' step by step through all the nets. This approach has been used for the diagnosis of interconnects in printed circuit boards (PCBs) using a boundary scan architecture [Hassan 88]. In this paper, we propose a novel and very efficient way of implementing the "walking-1" method in FPGAs. The logic function of the CLBs is programmed in a clever way to "walk" a '1' through all the nets in the design.

The proposed method for fault diagnosis targets both bridging and stuck-at faults. CLBs are configured to define pseudo scan paths. Note that we are not using any existing scan structure on the FPGA. We are forming scan paths by using the internal flip-flops in the CLBs and a programming scheme for the logic function of the CLBs. The scan paths defined are not independent. Two or more scan paths can merge. Multiple FPGA configurations are used to satisfy the following requirements:

Requirement 1: Each net is controlled to a '1' while the remaining nets are '0' and

Requirement 2: Each net is controlled to a '0' while the remaining nets are '1'.

The net being driven to '1' in Requirement 1 and '0' in Requirement 2 has to be made observable at the output after some number of clock cycles.

The "walking-1" test case was sufficient for fault diagnosis in [Hassan 88] since the scan paths used (boundary scans) were independent. In our case, the scan paths can merge, hence both "walking-1" and "walking-0" test cases are needed for fault diagnosis.

The problem at hand is given the interconnects in the "original configuration," we want to choose the logic function of the CLBs in each test configuration in such a way as to minimize the total number of test configurations that are required to satisfy Requirement 1 and 2. The easiest way to satisfy the requirements is to include each CLB on a scan path. Unfortunately, the number of scan paths cannot be greater than the number of primary inputs. However, we have developed a clever approach for overcoming this limitation. It involves identifying scan paths with more complex controllability conditions using linear programming techniques. We first discuss diagnosis with simple scan paths in Sec. 3 to illustrate the concepts. In Sec. 4, we introduce the more complex scan paths, which allow us to minimize the number of test configurations that are required.

3. Fault Diagnosis

In this section, we explain the FPGA configurations used for fault diagnosis.

3.1 Scan Paths

To walk a '1' through each net, the CLBs are configured to define pseudo scan paths that run from the primary inputs to the primary outputs. The FPGA is reconfigured such that the place and route information of the original design is used as it is; only the logic performed by the CLBs is changed. The CLBs are essentially used as routers to construct scan paths. Since the interconnects used are the same as the interconnects in the original design, every CLB in the configuration has the same set of fanins as the original circuit. The output of each CLB is taken through the internal flip-flop in the CLB so that all segments of the scan path are clocked. Any CLB on a scan path can be controlled and the CLB output is observable since it is uniquely routed to a primary output; its value can be scanned out.

Definition 1. *ACTIVE-SEG* and *INERT-SEG:* Every CLB on a scan path has to be uniquely routed to one and only one primary output since we need a single '1' walking through the design. This is ensured by controlling fanouts. For nets with multiple fanouts, only one fanout branch is allowed to be a controlling input to the CLB it feeds. This branch is labeled ACTIVE-SEG, the remaining branches are labeled INERT-SEG. When a net is justified to '1', the ACTIVE-SEG propagates the '1' to the next CLB. The remaining INERT-SEGs are non-controlling inputs and do not propagate the '1' forward. We are thus able to get a single '1' traveling through the CLBs on a scan path.

3.2 Fault Detection and Location

The overall procedure for fault diagnosis is given below.

Step 1. Load the FPGA configuration that walks a '1' through the circuit. This configuration (CONFIG-1) detects all stuck-at-1 and wired-AND bridging faults. There are two types of FPGA configurations used for fault diagnosis. Scan paths are constructed in both configurations as described in Section 3.1. The logic implemented by the CLBs in the configuration differ. In CONFIG-1 the reset value of each flip-flop is '0'. The output of a CLB is '1' when there is exactly one '1' in its fanins **and** this '1' is on a net labeled ACTIVE-SEG.

Step 2. Apply the test case that detects all stuck-at-1 faults on ACTIVE-SEGs. This test case (FLOOD-0) is a sequence of test vectors applied at the primary inputs, where each test vector is the vector of all '0's. The length of the test sequence i.e. the number of times the design is clocked is equal to the length of the longest path in the design. On applying FLOOD-0 to CONFIG-1, the fault free response are vectors of all '0's.

Step 3. Apply the test case that detects all stuck-at-1 faults on INERT-SEGs and all wired-AND bridging faults. This test case (WALK-1) is a sequence of test vectors applied at the primary inputs of CONFIG-1 that walk a '1' through each of the scan paths. This can be done by applying a test vector with one of the primary inputs set to '1' followed by vectors of all '0's. The '1' should appear at the primary output at the end of the scan path when the number of clocks applied is equal to the length of the scan path. The number of all '0' vectors applied i.e. the number of times the design is clocked after setting one of the primary inputs to '1' is twice the length of the longest path in the design.

Step 4. Load CONFIG-0. This configuration is used to detect all stuck-at-0 and wired-OR bridging faults.

Step 5. FLOOD-1. This test case is used to detect all stuck-at-0 faults on ACTIVE-SEGs.

Step 6. WALK-0. This test case is used to detect all stuck-at-0 faults on INERT-SEGs and all wired-OR bridging faults.

CONFIG-0, FLOOD-1 and WALK-0 are the exact duals of CONFIG-1, FLOOD-0 and WALK-1 respectively. If a fault is detected at the above steps, fault location is done by applying the appropriate test vectors [Das 98]. A few examples are illustrated using Figure 1.

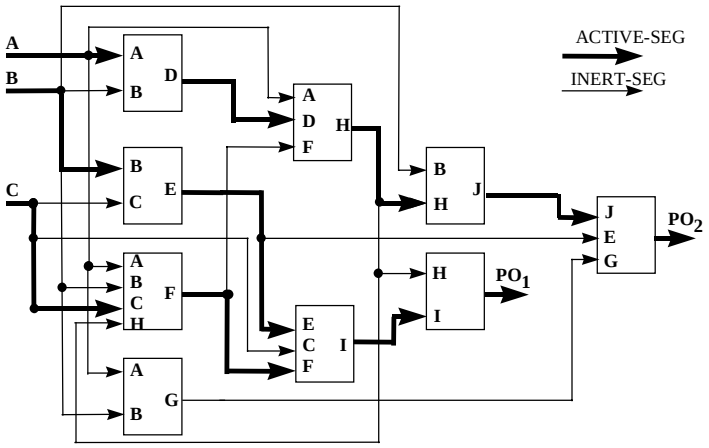

Figure 1. FPGA Configuration

Stuck-at-1 on ACTIVE-SEGs: These faults are detected by applying the FLOOD-0 test case in CONFIG-1. If any one of the ACTIVE-SEGs is stuck-at-1, the '1' will propagate along the scan path controlled by this ACTIVE-SEG and appear at a primary output.

The longest scan path in Figure 1 is four (*A-D-H-J-PO_2).* Any stuck-at-1 fault on the ACTIVE-SEGs will

result in an erroneous '1' arriving at one primary output within four clock cycles. For example, a stuck-at-1 on the ACTIVE-SEG of *H* will cause an erroneous '1' to appear on PO_2 on the second clock cycle. The clock cycle at which the erroneous '1' appears gives information about the location of the faulty net.

Stuck-at-1 on INERT-SEGs: Consider an INERT-SEG which is stuck-at-1. Let us denote the other net segments feeding the same CLB as this net segment as its sibling net segments. At least one of these sibling net segments is an ACTIVE-SEG for the CLB to be on a scan path. Consider the test case which walks a '1' along this ACTIVE-SEG. The presence of an extra '1' caused by the stuck-at-fault will force the walking '1' to a '0' and the fault can thus be detected.

To illustrate with an example, assume the INERT-SEG of *F* is stuck-at-1. The ACTIVE-SEG among its sibling nets is *D*. *D* lies on the scan path $A\text{-}D\text{-}H\text{-}J\text{-}PO_2$. While walking a '1' through this scan path, the output of CLB *H* will be forced to '0' since two of its inputs are '1', thus terminating the walking '1'.

Bridging fault (wired-AND) between two nets: Wired-AND bridging faults are detected by applying the WALK-1 test case on CONFIG-1. The bridging fault gets detected when a '1' is walked on to one of the nets while the other is at '0' and vice versa. Assume *E* and *H* are bridged with the '0' dominating (wired-AND). While walking a '1' through the scan path $A\text{-}D\text{-}H\text{-}J\text{-}PO_2$, *H* is set to '1' while *E* is at '0'. The bridging fault will get activated, the walking '1' gets terminated and the fault is detected. The same thing happens when a '1' is walked through the scan path $B\text{-}E\text{-}I\text{-}PO_1$. Fault location can be done by using the CONFIG-0 test configuration [Das 98].

4. Reducing Number of Configurations

The lower bound on the number of FPGA configurations required to put every CLB on a scan path is (Max number of gates in a level) divided by (Number of Primary Inputs). This number can become large for big designs having a relatively low number of inputs. In this section we describe a clever scheme for drastically reducing the required number of configurations. We introduce two terms: *Single Input Controlled* (SIC) scan paths and *Multiple Input Controlled* (MIC) scan paths.

SIC: In SIC scan paths, the output of the each CLB is set to '1' by any *one* of the ACTIVE-SEG inputs being set to '1'. The scan paths described in the previous sections were SIC scan paths.

MIC: In MIC scan paths, the output of the CLB at the head of the scan path is set to '1' by *two or more* INERT-SEG inputs being set to '1'. Thus, by introducing MIC scan paths, additional CLBs (which were not included in any scan path) can be put on scan paths.

Consider a CLB that is not on any scan path. To put it on a MIC scan path, two or more of its INERT-SEG inputs have to be set to '1'. In Figure 1, the CLB *G* can be put on a MIC scan path by modifying its logic function to perform *G* is '1' iff *A* **and** *B* are '1'. The ACTIVE-SEG of *A* and *B* will propagate '1's into the circuit, but then our condition of a single '1' in the circuit will be violated. However, if we can find a test vector, which will suppress these other '1's, then *G* can be enabled as a scan path. By choosing *A* and *B* equal to '1', *E* is set to '1'. This '1' has to be suppressed. This can be done by setting *C* to '1'. Thus a valid test vector to enable *G* as a scan path is *A* and *B* and *C* equal to '1'.

We now give a general procedure to find test vectors to put additional CLBs on scan paths. We define a matrix **A** where the rows correspond to the CLBs currently on scan paths, the columns correspond to the primary inputs and CLBs on scan paths (i.e. nets whose values can be controlled). The entries in the matrix are defined below:

$a_{ij} =$ 1 if j is an ACTIVE-SEG fan in to CLB i,
-1 if j is an INERT-SEG fan in to CLB i,
0 if j does not fan into CLB i.

Consider a column vector **X** whose rows correspond to the nets which can be controlled. ***Claim:*** Any solution of **X** over the set {0,1} which makes $(\mathbf{A.X} \leq \mathbf{0})$ can be used to add extra CLBs on to the existent scan paths. Any entry in **A.X** corresponding to a CLB is '1' iff its ACTIVE-SEG input is set to '1' and no other input is set to '1'. For all other cases, the entry will be less than or equal to '0'.

The matrix inequality can be solved by integer linear programming. The solution is a binary vector over the primary inputs and CLBs on scan paths. If the solution has two or more nets at a value '1' that are fanins to a CLB that is currently not on a scan path, the CLB can be added to the scan path by defining the logic function such that the output of the CLB is '1' iff these nets are '1'. Once a CLB has been made a MIC CLB, a row is added in matrix **A** corresponding to this CLB and the entire procedure is repeated to find more test vectors.

Given the original configuration to be tested, we first select SIC scan paths that will cover the largest number of "undetected" nets. Then the remaining "undetected" nets are added to MIC scan paths until either all nets are "detected" or no more MIC scan paths can be added. If there are still nets that have not been included in any scan path for any configuration, then a new test configuration is added, and the procedure repeated. When the procedure completes, a sufficient set of test configurations have been created to detect and locate all stuck-at and bridging

faults.

5. Experimental Results

The procedure described in this paper was used to generate test configurations for FPGA implementations of some of the largest ISCAS benchmark circuits. Results are shown in Table 1. The total number of test configurations required for 100% fault coverage is shown for each circuit. For a particular circuit, the test time can be reduced by using fewer configurations at the cost of some loss in fault coverage. Most of the faults are detected by the first few test configurations. Later test configurations detect only a small number of additional faults. In almost all cases, 90% fault coverage could be achieved with only 6 configurations.

Table 1. Number of Test Configurations

Circuit				FPGA		
Name	PIs	Pos	FFs	CLBs	Nets	Configs
C2670	233	140	0	207	275	2
C3540	50	22	0	422	477	8
C5315	178	123	0	496	817	6
C6288	32	32	0	1045	1113	16
C7552	207	108	0	734	1063	4
s1423	17	5	74	117	153	2
s5378	35	49	179	440	657	18
s9234	36	39	211	513	685	14
s13207	62	152	638	628	804	18
s15850	77	150	534	955	1303	16

6. Conclusions

The FPGA test method described here provides a number of innovations:

1. It is a configuration-dependent test method in which only the logic function of the CLBs are changed. Thus, the test configuration can be obtained without time consuming placement and routing.

2. It fully tests the programmable interconnect exactly as it will be used in system operation. Bridging faults between all pairs of nets are both detected and located.

3. It is rapid enough to test the system each time it is reconfigured on the fly. Thus it is able to detect both latent manufacturing faults (which have previously gone undetected) as well as faults that occur over time.

When a fault is detected, information about the location of the fault can be passed back to the compiler so that the system can be reconfigured to avoid the fault and resume fault-free operation. Thus, the proposed approach can be used to greatly increase the dependability of reconfigurable systems with low-cost. Such techniques are needed to increase the domain of applications for which reconfigurable computing can be used.

Acknowledgement

This work is part of the ROAR project and is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract Number DABT63-97-C-0024.

References

[Culbertson 97] Culbertson, W. B., R. Amerson, R. J. Carter, P. Kuekes, and G. Snider, "Defect Tolerance on the Teramac Custom Computer," *Proc. of the 1997 IEEE Symposium on FPGA's for Custom Computing Machines,* 1997, pp. 140-147.

[Das 98] Das, D. and N.A. Touba, "Diagnosis of Interconnect Faults in FPGA-Based Reconfigurable Systems," *Technical Report No. 98-2,* Computer Aided Test Lab, University of Texas, Austin, 1998.

[Goel 82] Goel, P., and M. T. McMahon, "Electric Chip-In-Place Test," *Proc. of International Test Conference,* 1982, pp. 83-90.

[Hassan 88] Hassan, A., J. Rajski, and V. K. Agrawal, "Testing and Diagnosis of Interconnects using Boundary-scan," *Proc. of International Test Conference*, 1988, pp. 126-137.

[Kautz 74] Kautz, W. H., "Testing for Faults in Wiring Networks," *IEEE Trans. On Computers*, Vol. C-23, No. 4, 1974, pp. 358-363.

[Liu 95] Liu, T., F. Lombardi, and J. Salinas, "Diagnosis of Interconnects and FPICs Using a Structured Walking-1 Approach," *Proc. IEEE VLSI Test Symposium,* 1995, pp. 256-261.

[Lombardi 96] Lombardi, F., D. Ashen, X. T. Chen, and W. K. Huang, "Diagnosing Programmable Interconnect Systems for FPGAs," *Proc. ACM Int. Symp. On FPGAs,"* 1996, pp. 100-106.

[Huang 96] Huang, W.K., X. T. Chen, and F. Lombardi, "On the Diagnosis of Programmable Interconnect Systems: Theory and Applications," *Proc. IEEE VLSI Test Symposium*, 1996, pp. 204-209.

[Renovell 97] Renovell, M., J. Figueras, Y. Zorian, "Test of RAM-Based FPGA: Methodology and Application to the Interconnect," *Proc. IEEE VLSI Test Symposium*, 1997, pp. 230-237.

[Roy 95] Roy, K., and S.Nag, "On Routability for FPGAs Under Faulty Conditions," *IEEE Trans. on Computers*, Vol. C-44, No. 11, pp. 1296-1305, Nov. 1995.

[Zhao 98] Zhao, L., D. M. H. Walker, F. Lombardi, "Bridging Fault Detetction in FPGA Interconnects Using I_{DDQ}," *Proc. ACM Int. Symp. On FPGAs,"* 1998, pp. 95-103.