Interconnect Testing in Cluster-Based FPGA Architectures

Ian G. Harris
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01003
harris@ecs.umass.edu

Russell Tessier
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01003
tessier@ecs.umass.edu

ABSTRACT

As IC densities are increasing, cluster-based FPGA architectures are becoming the architecture of choice for major FPGA manufacturers. A cluster-based architecture is one in which several logic blocks are grouped together into a coarse-grained logic block. While the high density local interconnect often found within clusters serves to improve FPGA utilization, it also greatly complicates the FPGA interconnect testing problem. To address this issue, we have developed a hierarchical approach to define a set of FPGA configurations which enable interconnect faults to be detected. This technique enables the detection of bridging faults involving intra-cluster interconnect and extra-cluster interconnect. The hierarchical structure of a cluster-based tile is exploited to define intra-cluster configurations separately from extra-cluster configurations, thereby improving the efficiency of the configuration definition process. By guaranteeing that both intra-cluster and extra-cluster configurations have several test transparency properties, hierarchical fault detectability is ensured.

Categories and Subject Descriptors

T2.1 [**Testing and DFT**]: FPGA Testing

General Terms

field-programmable gate array, interconnect testing, hierarchical test

1. INTRODUCTION

Field programmable gate array (FPGA) technology has drastically reduced the cost of hardware manufacture, making hardware implementation economically feasible for applications which were previously restricted to software. As the use of FPGAs in commercial products becomes more commonplace, the significance of reliability and test has a greater financial impact. The increasing importance of the FPGA test problem has driven substantial research activity in a variety of FPGA test approaches. Because the cost of manufacture is effectively zero for the FPGA customer, the cost impacts of testing and design-for-test (DFT) are a major part of the overall cost. The impact of test on overall cost is even greater for FPGA technology than it is for traditional ASICs.

FPGA architectures have several properties which make the FPGA test problem unique from the test problem for ASICs. The ability to alter functionality through reconfiguration is both a blessing and a curse for test generation. The ability to reconfigure has the potential to enable FPGA testing with no real area or performance overhead in the functional circuit. Reconfigurability also allows operational faults to be tolerated on-the-fly by using the hardware redundancy inherent in FPGAs [8]. Many ASIC DFT approaches involve modifying the circuit functionality to incorporate test functionality and to ease the testing problem. A reprogrammable FPGA can implement this test functionality with none of the circuit overheads which occur with ASIC DFT. Reconfigurability does incur other test costs, including increased test generation complexity and increased test application time. The flexibility of the FPGA functionality introduces a new problem of identifying a set of test configurations which does not exist for ASIC testing. Although reconfigurability can enhance testability, the complexity of the test generation problem can affect design time and therefore time-to-market. The choice of test configurations determines which faults are detectable, and therefore determines the maximum achievable fault coverage. Since it is not possible for all faults to be detectable in a single configuration, several test configurations are needed. The multitude of configurations required for testing impacts test application time because of the relatively slow rate of reconfiguration.

In order to provide a solution for real architectures now and in the future, it is necessary to understand the current trends in industrial FPGA architectures. Several major FPGA manufacturers are moving toward cluster-based FPGA architectures [4]. A characteristic of cluster-based architectures, as shown in Figure 2, is that connectivity inside the clusters is typically high. High density local interconnect serves to improve FPGA utilization, but also greatly complicates the testing problem. Since pad density increases much more slowly than logic density, the high density interconnect in a cluster-based architecture creates a test access problem for embedded lines and logic. Novel testing approaches are needed to address and effectively test densely interconnected

cluster-based architectures.

We present a tool which automatically generates a test plan to detect pairwise interconnect bridging faults in an arbitrary cluster-based FPGA architecture. Each test plan is composed of a set of FPGA configurations which mutually enable all target faults to be detected. By exploiting the hierarchy inherent in the structure of a cluster-based tile, our approach partitions the test configuration definition process to greatly improve the efficiency of the process. A built-in self-test (BIST) technique is used to increase access to embedded FPGA logic. Since test configurations in our approach are replicated across the tile array, the process of defining test configurations is independent of the size of the FPGA array.

2. PREVIOUS WORK

Research in FPGA testing has investigated a wide range of test architectures and techniques, but to our knowledge, the problem of testing a cluster-based architecture has not been investigated. The FPGA test problem has been divided by several researchers into the interconnect test problem [10, 16, 14], and the FPGA logic test problem [17, 9]. The limited number of I/O pads greatly reduces test access from off-chip. The pad limit has been overcome by several researchers by using a number of BIST techniques [6, 12, 13, 14, 3] to reduce the need for pads. Some portion of the FPGA hardware is configured as test generation and response analysis circuitry which is used to test the remainder of the FPGA. In order to test all of the FPGA logic, several configurations are required by these techniques to ensure that all FPGA logic is tested in some configuration. Several approaches to on-line fault detection have been introduced which implement BIST by exploiting unutilized FPGA logic and routing to implement modular redundancy [11, 3].

The need for external controllability and observability has also been reduced by using an iterative logic array (ILA) test architecture [7, 9, 6, 12, 13]. An ILA architecture composed of an array of identical cells allows the controllability and observability of each cell to be effectively accomplished through its neighboring cells.

3. CLUSTER-BASED FPGAS

We assume an island-style FPGA architecture [5] which is composed of an array of identical tiles as shown in Figure 1a. Each tile is composed of a *cluster* [4] and surrounding interconnect. The interconnect structure of each tile is a set of lines which can be connected by a set of *programmable interconnect points* (PIP) which act as switches. A typical tile interconnect structure is shown in Figure 1b. The *switch matrix* shown in Figure 1c is a commonly used structure in FPGA architectures which is composed of a set of lines entering each side. A PIP connects each line to one line on each side of the matrix. Each PIP in the switch matrix is seen as a dashed line in Figure 1c.

For the purposes of testing, it is necessary to distinguish the *tile I/O* from the *cluster I/O*. Cluster I/O are the input and output pins of the cluster, while tile I/O pins refer to the points at which a tile can communicate with a neighboring tile. The tile I/O pins include the endpoints of wire segments which can connect to a neighboring tile via a PIP.

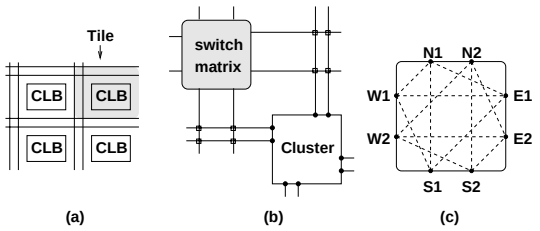

Figure 1: FPGA Structure (a) Tile array, (b) Extra-Cluster Interconnect, (b) Switch Matrix

We assume that each cluster is composed of a set of *basic logic elements* (BLE) [4], each of which is composed of a set of programmable *lookup tables* (LUT), multiplexers, and flip-flops. The most general assumption is that each BLE input can connect to the output of any other BLE and to any cluster input. The output of each BLE is assumed to be connected directly to a cluster output.

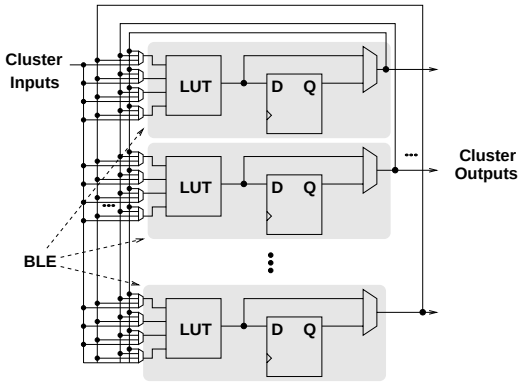

Figure 2: FPGA Cluster

4. FPGA TESTING METHODOLOGY

We propose the use of a built-in self-test (BIST) strategy for the testing of an FPGA structure. BIST techniques in general are associated with high performance and area overhead incurred by on-chip test hardware. BIST overhead is not an issue for FPGA BIST because the test hardware is easily inserted and removed by reconfiguration. By embedding test logic inside the FPGA, BIST enables test access to internal components. This is particularly important for the testing of cluster-based FPGA structures which have higher localized interconnect density than other FPGAs.

In each configuration, FPGA circuitry dedicated as BIST logic will perform test generation and response analysis to test non-BIST FPGA circuitry. To accomplish BIST, we use the test structure presented in [14] in which the FPGA is configured as many independent BISTERs structures, shown in Figure 3.

Each BISTER is composed of a test pattern generator (TPG), an output response analyzer (ORA), and two blocks under test (BUTs). The TPG is simply a counter which applies an exhaustive test sequence to the BUTs. Each BUT is a single tile in the FPGA which is being tested. The ORA is a comparator which sets the Pass/Fail flip-flop to '1' if the outputs of both BUTs do not agree. Each BISTER will

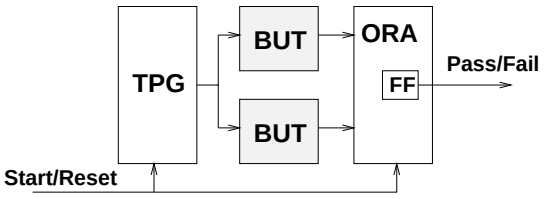

Figure 3: BISTER Test Structure

be implemented as a rectangular block of tiles, and many BISTERs will be implemented on the FPGA to cover the tile array. The number of tiles in a BISTER will depend on the number of tiles needed to implement the TPG and ORA logic.

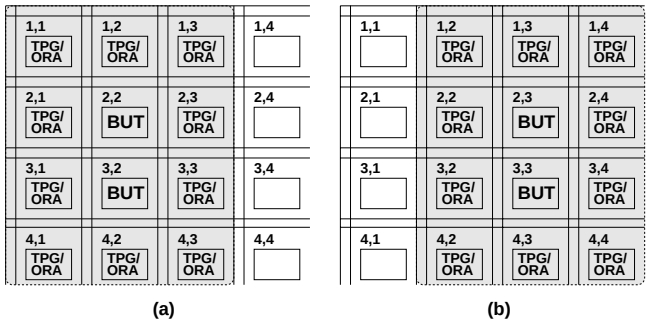

Figure 4: Shifting BISTERs Across FPGA Array, (a) BISTER in lower left, (b) BISTER shifted right

It is important to notice that the tiles which are dedicated to the TPG and ORA logic are not completely tested. In order to guarantee testing of all tiles, the FPGA will be reconfigured to shift the BISTERs across the entire array as shown in Figure 4. Over the course of several reconfigurations, all tiles will be tested by acting as a BUT in a BISTER. Since the tiles adjacent to a BUT must implement either TPG or ORA logic, the perimeter tiles cannot be tested by simply shifting the BISTERs. In order to ensure that perimeter tiles are tested, the layout of the BISTER must be modified to use the I/O pads to access the tiles on the periphery.

In addition to providing good test access, the use of this BIST strategy has several significant effects on the test configuration definition and test sequence definition problems. The BIST strategy decomposes the testing problem of the entire FPGA into many identical problems of a size which is fixed by the test requirements for a single tile. Since the size of the smaller problem is fixed, the BIST approach is easily scalable to FPGA arrays of any size.

5. FPGA INTERCONNECT FAULTS

Detection of interconnection faults in cluster-based architectures is a difficult problem because the high density of internal cluster interconnect makes test access difficult. We propose a formulation of the problem which includes the testing of *intra-cluster interconnect* which is internal to the cluster, as well as *extra-cluster interconnect* which surrounds each cluster. All pairs of lines are classified as either *connectable* if there is a PIP between them, and *non-connectable* if there is no intervening PIP. We assume the possibility of two types of defects, a *short defect* which causes two lines to be crossed, and an *open defect* which causes a single line to be broken, or causes a connectable line pair to be unconnectable. Given the two classes of line pairs and the two defect types, we assume 4 fault classes which are previously presented in [10]. The interconnect faults which we target are subsets of bridging faults whose detection requirements have been outlined in previous work [15, 2]. We summarize the 4 fault classes and the detection requirements for each class in terms of the controllability and observability of each line involved.

- **Permanent Connection (PC)** - A short on any pair of lines. Both affected lines must be separately controllable and at least one affected line must be observable. Also, any PIP between the two affected lines must be configured to be off.
- **Permanent Disconnection (PD)** - An open on any pair of C lines. Both affected lines must be controllable and observable. Also, the PIP between the two affected lines must be configured to be on.
- **Stuck-At 0 (SA0)** - A short between a line and ground (special case of a PC fault). The affected line must be controllable and observable.
- **Stuck-At 1 (SA1)** - A short between a line and power (special case of a PC fault). The affected line must be controllable and observable.

6. TEST CONFIGURATION DEFINITION

The goal of test configuration definition is to identify a set of configurations for the tiles acting as BUTs in a BISTER. The set of configurations must have the property that the fault detection requirements stated in Section 5 must be satisfied for all faults in at least one configuration. The size of the set of test configurations should be minimized to reduce test application time. The test configuration definition process is hierarchical, defining the intra-cluster configurations separately from the extra-cluster configurations. Test transparency constraints are placed on the intra-cluster and extra-cluster configurations to ensure hierarchical controllability and observability.

6.1 Intra-Cluster Configurations

The intra-cluster configurations are defined to ensure that all intra-cluster interconnect faults are detectable in at least one configuration, and to facilitate the testing of the extra-cluster interconnect. The cluster will be contained in the control and observe paths of many extra-cluster interconnect lines. The cluster must be configured to be *transparent* from a controllability and observability perspective. The cluster outputs are not identical to the cluster inputs, but the cluster outputs must have the following transparency properties with respect to the cluster inputs.

1. A fault effect on a cluster input must propagate to at least one cluster output. This condition ensures the propagation of fault effects on extra-cluster which feed the cluster inputs.

2. The cluster outputs must be separately controllable. This condition ensures the controllability of the extra-cluster interconnect which is driven by the cluster outputs.

6.1.1 BLE Configurations

The observability of the cluster inputs and BLE output branches must be achieved by propagating fault effects through the BLEs to reach the cluster outputs. Also, the controllability of the BLE outputs must be achieved through the BLEs. The configuration of the BLEs is central to ensuring maximal controllability and observability inside the cluster. The configurations of components inside the BLEs are important to enable controllability of the BLE output lines, as well as observability of the cluster inputs lines and the BLE output branches.

Each BLE is composed of a LUT and a multiplexer, both of which must be configured. To maximize the controllability and observability through a BLE, we have chosen to configure each LUT to act as a 4-input XOR gate. The XOR operation provides good controllability because the output value may be determined by controlling any single input. The XOR also provides good observability because a fault effect on any single input is guaranteed to propagate to its output. To simplify the interconnect testing process, we configure the multiplexers inside the BLEs to drive the BLE output with the LUT output directly, bypassing the flip-flop. This eliminates sequential behavior during testing and ensures that the application of an exhaustive test pattern set is sufficient to detect all faults which are non-redundant in each configuration.

6.1.2 BLE Input Multiplexer Configurations

The configurations of the BLE input multiplexers (IMUX) affect both the controllability and observability of the cluster interconnect. The IMUXes determine controllability of BLE outputs by determining the function which defines the output of each BLE n. Because all LUTs are configured as XOR gates, each output BLE function is an XOR of a subset of cluster inputs as seen in Figure 5a. In Figure 5a, the input sources determined by the multiplexer configurations are labelled and shown in bold. Based on the multiplexer configurations, the BLE output functions are expressed as follows: $BLE1 = IN1 \oplus IN2 \oplus IN3 \oplus IN4$, $BLE1 = IN1 \oplus IN2 \oplus IN3 \oplus IN4 \oplus IN5 \oplus IN6 \oplus IN7$. Notice that the most general cluster-based architecture allows a multiplexer to be configured to create a loop as shown in Figure 5b, which would either create sequential activity (if the BLE output is clocked) or an asynchronous activity. Both of these possibilities would greatly complicate testing, so we do not allow multiplexers to be configured to create a loop. This assumption matches the implementation of Xilinx Virtex [1] part which is a cluster-based architecture which does not contain interconnect to implement self-loops inside a cluster.

We have developed an algorithm to define the configuration of each IMUX in each overall FPGA configuration. We have identified the following IMUX configuration goals which must be satisfied by the algorithm. These properties are required to make all intra-cluster faults detectable in at least one configuration, and to ensure the transparency of the cluster for the testing of extra-cluster faults.

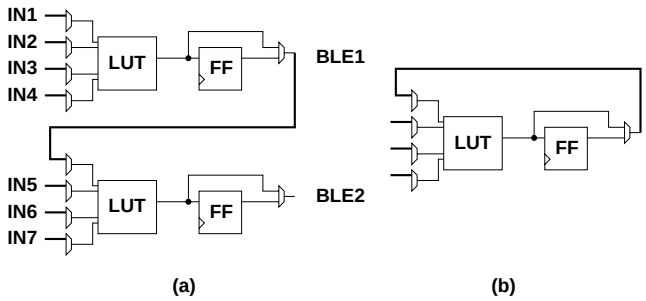

Figure 5: Input Multiplexer Configurations, (a) BLE output function determined by input mux configurations, (b) Illegal input mux configuration creating a self-loop.

- **All BLE outputs are separately controllable from each other, and from all cluster inputs** - This property ensures that each intra-cluster fault can be activated in each configuration, and enables the activation of extra-cluster faults associated with extra-cluster lines driven by cluster outputs. Guaranteeing this property is accomplished by defining input multiplexer configurations so that each BLE output function is different, and is not dependent on a single cluster input.

- **Each input multiplexer is configured to select data from each of its inputs in at least one configuration** - This property ensures that all cluster input branches and BLE output branches are observable in at least one configuration.

- **There is a sensitized path from each cluster input stem to a cluster output in every configuration** - This property ensures the transparent propagation of extra-cluster fault effects through the cluster. This property is accomplished by configuring at least one input multiplexer to receive data from each cluster input in each configuration. Every cluster input stem can be associated with at least one BLE output whose value is dependent on that cluster input stem.

Algorithm 1 Intra-Cluster Configuration Algorithm

```
label all intra-cluster faults as undetected
repeat
  repeat
    select a BLE which is not configured, b
    initialize IMUX configurations of b
    repeat
      enumerate next IMUX configuration
      compute BLE output function
    until BLE function is unique
  until all BLEs are configured
  identify detectable faults
until all faults are detectable in some configuration
```

The algorithm for intra-cluster test configuration definition is shown in Algorithm 1. The algorithm contains 3 main loops. The inner loop, defines the configuration of a single BLE by enumerating the configurations on all 4 of its input multiplexers until a satisfactory configuration is found. A set of BLE IMUX configurations is considered satisfactory if the resulting BLE output function is unique from the functions of all other BLEs, and is unique from all single cluster inputs. The middle loop invokes the inner loop with each BLE until all BLEs are configured to produce a complete cluster configuration. The outer loop invokes the middle loop to define a single configuration, and then evaluates the detection of intra-cluster bridging faults. The outer loop continues to invoke the middle loop until all intra-cluster faults are detected in at least one configuration.

6.2 Extra-Cluster Configurations

The extra-cluster configuration defines current flow paths through the extra-cluster interconnect. These current flow paths between tile input and output pins are used to control and observe each interconnect segment on the path. We model the extra-cluster configuration definition problem as a flow problem through an *interconnect graph*. Each node in the graph represents an extra-cluster interconnect segment, and each edge represents the existence of a PIP between two segments.

One goal of extra-cluster test configuration definition is to create flow paths between tile I/O nodes which allow the detection criteria of each fault to be satisfied in at least one configuration. In addition to enabling the detection of extra-cluster faults, the extra-cluster configuration must enable transparent controllability and observability of the embedded cluster. This goal is accomplished by creating flow paths from tile I/Os to every cluster input, and from every cluster output to tile I/Os, in every configuration. An example of a configuration which exhibits this type of test transparency is shown in Figure 6. The bold lines indicate segments which are contained in flow paths, and the bold PIPs indicate which PIPs are switched on in the configuration in order to instantiate the paths. Each cluster input and output is directly connected to the edges of the tile via a set of flow paths. Notice that flow through the cluster does not impact the testability of extra-cluster interconnect because the cluster is transparent for controllability and observability purposes.

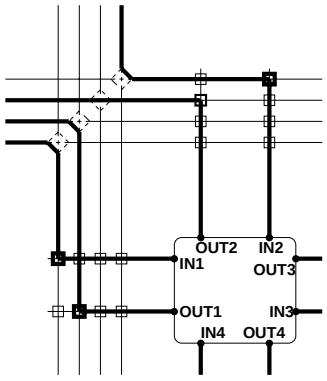

Figure 6: Transparent Extra-Cluster Configuration

Algorithm 2 Extra-Cluster Configuration Algorithm

```
create interconnect graph
repeat
  label all nodes as untouched
  repeat
    select an untouched node n
    identify an untouched path from n to a cluster I/O
    label all nodes on the path as touched
    identify an untouched path from n to a tile I/O
    label all nodes on the path as touched
  until paths connect all cluster I/O to tile I/O
  repeat
    select an untouched node n
    identify an untouched path from n to a tile I/O
    label all nodes on the path as touched
    identify an untouched path from n to a tile I/O
    label all nodes on the path as touched
  until no addition untouched paths can be created
  identify detectable faults
until all faults are detectable in some configuration
```

The algorithm for extra-cluster test configuration definition is outlined in Algorithm 2. The two inner loops define a single test configuration by identifying a set of paths through the extra-cluster interconnect which must be activated. The first inner loop guarantees that the cluster I/O are directly controllable and observable from the tile I/O. The second inner loop serves to increase the number of extra-cluster interconnects which are controllable and observable. Each pass of the outer loop defines a single test configuration. The tasks, *select an untouched node n*, *identify an untouched path from n to a cluster I/O*, and *identify an untouched path from n to a tile I/O* are performed using several heuristics which target lines associated with faults which are undetected in the current configuration.

7. EXPERIMENTAL RESULTS

We have implemented the presented algorithms for test configuration definition and we have applied the algorithms to define test configurations for a range of cluster-based tiles of different sizes. In test results we assume that the cluster has the structure shown in Figure 2 [4], with N BLEs and I cluster inputs. We assume that the extra-cluster structure of each tile is of the form shown in Figure 1b. We assume that cluster inputs and outputs are equally distributed around the sides of the cluster. Each cluster I/O on the north face may connect to all horizontal tracks via a set of PIPs, and the same is true between cluster I/O on the west face and the vertical tracks. The cluster I/O on the east and south faces are assumed to connect directly to tracks in the neighboring tiles.

These results are summarized in Table 1. The first two columns of Table 1 are the *Clus. Prms* which indicate the size of the cluster in terms of the number of cluster inputs, I, and the number of BLEs in a cluster, N. The remainder of the columns in the table are divided into the results of *Intra-Cluster* configuration definition, and *Extra-Cluster* configuration definition. Two results are presented for both intra-cluster and extra-cluster configuration definition: (1) *Confs* - the number of configurations defined by our algorithm, and (2) *FCov* - the percent of bridging faults de-

Clus. Prms.		Intra-Cluster			Extra-Cluster	
N	I	Confs	min	FCov	Confs	FCov
4	8	13	11	100.0%	9	100.0%
4	10	14	13	100.0%	9	100.0%
4	12	17	15	100.0%	8	100.0%
6	12	19	17	100.0%	9	99.5%
6	14	20	19	100.0%	11	99.1%
6	16	22	21	100.0%	11	99.0%
8	16	28	23	100.0%	11	99.3%
8	18	28	25	100.0%	11	98.9%
8	20	28	27	100.0%	13	99.1%

Table 1: Experimental results with a variety of cluster sizes

tected across all configurations. In addition to these results, a *min* result is provided for intra-cluster results, indicating a theoretical lower bound on the number of intra-cluster configurations required. This lower bound is computed as the fanin of the input multiplexers ($I+N$), less 1 to account for the self-loop multiplexer input which we do not test.

The results in Table 1 show that testing the intra-cluster interconnect is the bottleneck in the number of configurations required. This is expected because the ratio of intra-cluster interconnect segments to cluster I/O pins is much higher than the 1:1 ratio between extra-cluster interconnect and the tile I/O. The intra-cluster fault coverage achieved is always 100%, while the fault extra-cluster fault coverage is sometimes slightly less than complete. This is a result of a weakness in the heuristic used to define flow paths through the extra-cluster interconnect, rather than a result of inherent complexity in the extra-cluster configuration definition problem. By manually adding a single extra-cluster configuration, the extra-cluster fault coverage can be increased to 100% in all cases.

8. CONCLUSIONS

We have presented a hierarchical technique to define test configurations for the detection of interconnect faults in cluster-based FPGA architectures. We have used the concept of test transparency to define configurations which enable test access to the high-density logic cluster embedded within each FPGA tile. We have demonstrated that this technique can be used to successfully define a small set of test configurations which allow the detection of nearly all targeted interconnect faults. In the future, we intend to consider a formulation of the FPGA test problem which incorporates the testing of the LUTs and multiplexers inside each BLE.

9. REFERENCES

[1] Virtex data sheet. *Xilinx Corporation*, 1998.

[2] M. Abramovici and P. R. Menon. A practical approach to fault simulation and test generation for bridging faults. *IEEE Transactions on Computers*, C-34(7):658–663, July 1985.

[3] M. Abramovici, C. Stroud, C. Hamilton, S. Wijesuriya, and V. Verma. Using roving STARs for on-line testing and diagnosis of FPGAs in fault-tolerant applications. In *International Test Conference*, September 1999.

[4] V. Betz and J. Rose. Cluster-based logic blocks for FPGAs: Area-efficiency vs. input sharing and size. In *IEEE CICC*, pages 551–554, 1997.

[5] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic. *Field-Programmable Gate Arrays*. Kluwer Academic Publishers, 1992.

[6] G. Gibson, L. Gray, and C. Stroud. Boundary scan access of built-in self-test for field programmable gate arrays. In *IEEE International ASIC*, pages 57–61, September 1997.

[7] W. K. Huang, F. J. Meyer, X.-T. Chen, and F. Lombardi. Testing configurable LUT-based FPGAs. *IEEE Transactions on Very Large Scale Integration Systems*, 6(2):276–283, June 1998.

[8] V. Lakamraju and R. Tessier. Tolerating operational faults in cluster-based FPGAs. In *8th International ACM/SIGDA Symposium on Field Programmable Gate Arrays*, February 2000.

[9] M. Renovell, J. M. Portal, J. Figueras, and Y. Zorian. SRAM-based FPGAs: Testing the LUT/RAM modules. In *International Test Conference*, pages 1102–1111, October 1998.

[10] M. Renovell, J. M. Portal, J. Figueras, and Y. Zorian. Testing the interconnect of RAM-based FPGAs. *IEEE Design & Test of Computers*, 15(1):45–50, January-March 1998.

[11] N. R. Shnidman, W. H. Mangione-Smith, and M. Potkonjak. On-line fault detection for bus-based field programmable gate arrays. *IEEE Transactions on Very Large Scale Integration Systems*, 6(4):656–666, December 1998.

[12] C. Stroud, E. Lee, and M. Abramovici. BIST-based diagnostics of FPGA logic blocks. In *International Test Conference*, pages 539–547, November 1997.

[13] C. Stroud, E. Lee, S. Konala, and M. Abramovici. Using ILA testing for BIST in FPGAs. In *International Test Conference*, pages 68–75, October 1996.

[14] C. Stroud, S. Wijesuriya, C. Hamilton, and M. Abramovici. Built-in self-test of FPGA interconnect. In *International Test Conference*, pages 404–411, October 1998.

[15] M. J. Y. Williams and J. B. Angel. Enhancing testability of large-scale integrated circuits via test points and additional logic. *IEEE Transactions on Computers*, C-22(1):46–60, January 1973.

[16] L. Zhao, D. M. H. Walker, and F. Lombardi. Bridging fault detection in FPGA interconnects using i_{DDQ}. In *International Symposium on Field Programmable Gate Arrays*, pages 95–104, February 1998.

[17] L. Zhao, D. M. H. Walker, and F. Lombardi. Detection of bridging faults in logic resources of configurable FPGAs using i_{DDQ}. In *International Test Conference*, pages 1037–1046, October 1998.