Design-Specific Path Delay Testing in Lookup-Table-Based FPGAs

S. Sulochana
Department of ECE
Assistant Professor, Dr.Sivanthi Aditanar College of Engineering

Abstract- *Due to the increased use of field programmable gate arrays (FPGAs) in production circuits with high reliability requirements, the design-specific testing of FPGAs has become an important topic for research. Path delay testing of FPGAs is especially important since path delay faults can render an otherwise fault-free FPGA unusable for a given design layout. This paper presents a new approach for FPGA path delay testing which partitions target paths into subsets that are tested in the same test configuration. Each path is tested for all combinations of signal inversions along the path length. Each configuration consists of a sequence generator, response analyzer and circuitry for controlling inversions along tested paths, all of which are formed from FPGA resources not currently under test. Two algorithms are presented for target path partitioning to determine the number of required test configurations. Test circuitry associated with these methods is also described. The results of applying the methods indicate that our path delay testing approach requires seconds per design to cover all paths with delay within 10% of the critical path delay. The approach has been validated using Xilinx Virtex devices.*

I. INTRODUCTION

This paper is concerned with testing paths in lookup-table (LUT) based FPGAs after they have been routed. While this may be regarded as user testing , we are considering an environment in which a large number of manufactured FPGA devices implementing a specific design are to be tested to ensure correct operation at the specified clock speed. It is thus akin to manufacturing tests in that the time needed for testing is important. Ideally, we would like to verify that the actual delay of every path between flip-flops is less than the design clock period. Since the number of paths in most practical circuits is very large, testing must be limited to a smaller set of paths. Testing a set of paths whose computed delay is within a small percentage of the clock period may be sufficient in most cases. Thus, our goal is to determine by testing whether the delay along any of the paths in the set exceeds the clock period.

II. BASIC APPROACH

The goal of this work is to test a set of paths, called target paths, to determine whether the maximum delay along any of them exceeds the clock period of the circuit. These paths are selected based on static timing analysis using nominal delay values and actual routing information. Circuitry for applying test patterns and observing results is configured using parts of the FPGA that are not under test.

III. INTRODUCTION TO APPROACH

The delay of a path segment usually depends on the direction of signal transition in it. The direction of the signal transition in any segment is determined by that of the transition at the source and the inversions along the partial path leading to the particular segment. A test to determine whether the maximum delay along a path is greater than the clock period must propagate a transition along the path and produce a combination of side-input values that maximizes the path delay. This approach is not usually feasible because of the difficulty of determining the inversions that maximize the path delay and the necessary primary input values to produce them. Instead, we propose to test each target path for all combinations of inversions along it, guaranteeing that the worst case will also be included.

Although the number of combinations is exponential in the number of LUTs along the path, the method is feasible because application of each test requires only a few cycles of the rated clock. However, the results may be pessimistic in that a path that fails a test may operate correctly in the actual circuit, because the combination of inversions in the failing test may not occur during normal operation.

We shall first explain our method of testing a single path in a circuit and describe a test circuit for implementing it. Application of this method to test a number of paths simultaneously is discussed in the next section.

Our approach, first suggested in a recent paper [19], reprograms the FPGA to isolate each target path from the rest of the circuit and make inversions along the path controllable

by an on-chip test controller. Every LUT along the path is re-programmed based on its original function. If it is positive unate in the on-path input, the LUT output is made equal to the on-path input independent of its side inputs. Similarly, negative unate functions are replaced by inverters. If the original function is binate in the on-path input, the LUT is re-programmed to implement the exclusive-OR (XOR) of the on-path input and one of its side-inputs, which we shall call its controlling side input. As mentioned

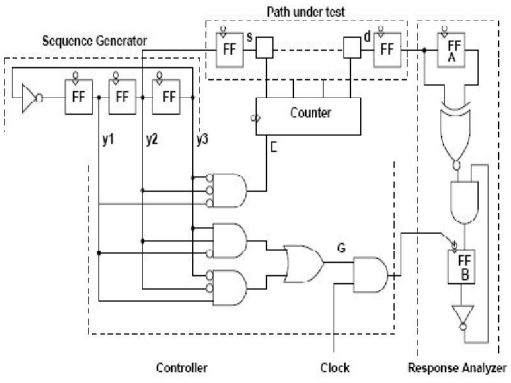


Fig. 1 Testing a single path for a negative edge-triggered design. For a positive edge-triggered design, all negative edge-triggered components are replaced by positive edge-triggered counterparts.

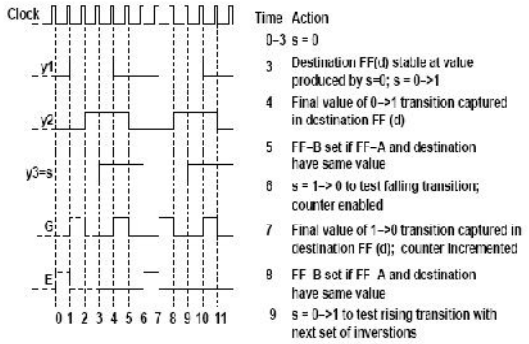


Fig. 2. Timing diagram for clock with period *T*

Mentioned earlier, this change of functionality does not affect the delay of the path under test because the delay through an LUT is unaffected by the function implemented. Inversions along the path are controlled by the signal values on the controlling side inputs. For each combination of values on the controlling side inputs we apply a signal transition at the source of the path and observe the signal value at the destination after one clock period. The absence of a signal transition will indicate that the delay along the tested path exceeds the clock period for the particular combination of inversions.

The basic method described above can be implemented by the circuitry shown in Fig. 1, consisting of a sequence generator, a response analyzer and a counter, that generates all combinations of values in some arbitrary order. A linear feedback shift register modified to include the all-0's output [20], [21] may be used as the counter. The controller and the circuitry for applying tests and observing results are also formed during configuration in parts of the FPGA that do not affect the behavior of the path(s) under test.

The sequence generator produces a sequence of alternating zeros and ones, with period equal to 6T,where T is the operational clock period. The response analyzer checks for an output transition for every test, and sets an error flip-flop if no transition is observed at the end of a test. The flip-flop is reset only at the beginning of the test session , and will indicate an error if and only if no transition is produced in some test. The counter has as many bits as the number of binate LUTs along the tested path.

The test for a path for each direction of signal direction consists of two parts, an initialization part and a propagation part, each of duration 3T.A paths is tested in time 6T by overlapping the initialization part of each test with the propagation part of the preceding test. In addition the change of counter state for testing a path for a new combination of inversions is also done during the initialization phase of rising transition tests.

Fig.2 shows the timing of the signals during the application of a test sequence. It can be seen from the figure that the source s of the test path toggles every three clock cycles. For correct operation, the input transition occurring at 3T must reach the destination within time T(i.e., before 3T+T).On the following clock edge at 3T+T,the result of the transition is clocked into the destination flip-flop at d.A change must be observed at the destination for every test, otherwise a flip-flop is set to indicate an error. In Fig.2, a test for the rising edge starts at time 3T, with the s steady at zero for the preceding three clock cycles. A test for the falling transition starts at 6T, with the input steady at one for the preceding three clock cycles. Results are sampled at d at time 4T (for rising edge s transition)and 7T (for falling edge s transition),respectively.Thus,both rising and falling transitions are applied at the source for each combination of inversions in time 6T.

As the falling transition is applied at 6T,the enable input E of the counter is set to 1.This action starts a state (counter)change at 7T to test the path for the next combination of inversions .A counter change at this time point allows2T of settling time before the following transition occurs at the sources. By ensuring that the counter reaches its final value within T and propagates to the path destination d within an additional T, d is ensured to be stable before the following

source transition. Thus, the destination will reach the correct stable value corresponding to the new combination of inversions if no path from the counter to the destination has a delay greater than 2T.This delay explains the need for a 3T period betweens transitions (1T to perform the test, 1T for possible counter state changes ,and 1T for subsequent propagation of the counter changes to d).

IV. TEST STRATEGY

The method described in the preceding section requires the test control circuitry to be reconfigured for every path to bet tested. The total time for testing a set of target paths in a circuit consists of the test application time and the reconfiguration time. Our goal is to reduce both components of the total time for testing a specified set of pahs.Since the time needed for configuring the test structure is usually larger than that for applying test patterns generated on chip we shall focus on reducing the number of test configurations needed by testing as many paths as possible in each configuration.

.

Two approaches to maximize the number of paths tested in a test configuration suggest themselves. First, we can try to select a set of target paths that can be tested simultaneously. This will also have the effect of reducing test application time. Secondly, we can try to select a set of simultaneously testable sets that can be tested in sequence with the same configuration. In this case, the number of simultaneously tested paths may have to be reduced so as to maximize the total number of paths tested with the configuration. These two approaches will be elaborated in the next two sections, but first we define a few terms..

The simultaneous application of a single rising or falling transition at the sources of one or more paths and observing the response at their destinations is called a test. The set of tests for both rising and falling transitions for all combinations of inversions along each path is called a test phase, or simply, a phase. As mentioned earlier, a single path with k binate LUTs will have 2 · 2k tests in a test phase. The application of all test phases for all target paths in a configuration is called a test session.

A. Single Phase Method

This method, first presented in [19], attempts to maximize the number of simultaneously tested paths. A set of paths may be tested in parallel if it satisfies the following conditions:

1) No two paths in the set have a common destination.
2) No fan out from a path reaches another path in the set.

The above conditions guarantee that signals propagating along paths in the set do not interfere with one another. Moreover, if the same input is applied to all paths in the set, two or more paths with a common initial segment will not interact if they do not re-converge after fan out.

All LUTs on paths to be tested in a session are reprogrammed to implement inverters, direct connections or XORs as discussed in the preceding section. The LUTs with control inputs are levelized, and all control inputs at the same level are connected to the same counter output. The source flip flops of all paths to be tested in the session are connected to the same sequence generator, but a separate transition detector is used for each path. The transition detectors of all paths are then ORed together to produce an error indication if any of the paths is faulty. Alternatively, a separate error flip-flop can be used for each tested path, connected to form a scan chain and scanned out to identify the faulty path(s).

B. Multi-phase Method

The single phase method described above requires that all paths tested in a session be disjoint. The number of test sessions needed for a large target set is therefore likely to be very large. The multi-phase method attempts to reduce the number of test sessions needed by relaxing the requirement that all paths tested in a session be disjoint. This, however, increases the test and cannot be tested simultaneously.

Consider sets of target paths S1, S2, Sp such that all paths in each set are disjoint except for common sources. Clearly, all paths in each set Si can be tested simultaneously, as in the single phase method, if each set can be selected and logically isolated from all other paths. This allows the testing of the sets Si in sequence, and is the basis of our multi-phase method. We also restrict the target paths for each session to simplify the control circuitry needed.

We assume that the LUTs in the FPGA are 4-input LUTs, but the method can be easily modified to allow a larger number of inputs. Since each LUT may need up to two control inputs, one for path selection and the other for inversion control, at most two target paths may pass through any LUT. Target paths satisfying the following conditions can be tested in a single session.

1) There is a path to each target path destination, called the main path to the destination.
2) Main paths may not intersect, but they may have a common initial section.

3) Additional paths to each destination, called its side paths, must meet only the main path and continue to the destination along the main path.
4) Main and side paths may not intersect any other path except that two or more paths may have a common source.
1. 5)No more than two target paths may pass through any LUT.
2. 6)The number of target paths to all destinations must be the same.

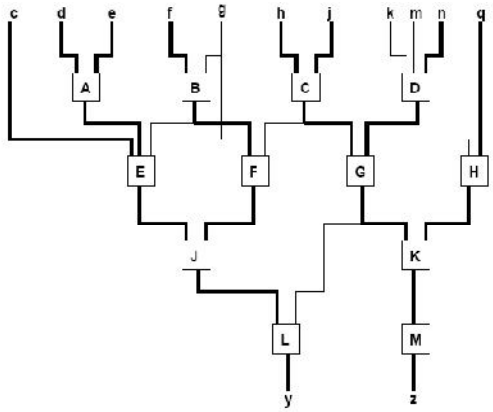
The above conditions allow us to select one path to each output and test all of them in parallel. The first two conditions guarantee that the signal propagating along main paths to different destinations will not interact. The main paths can therefore be tested in parallel. The restriction that a side path can meet only the main path to the same destination [condition 3)] allows a simple mechanism for propagating a signal through the main path or one of its side paths. Together with Condition 4, it guarantees that a set of main paths or a set of side paths, one to each destination, can be tested in parallel. Condition 5 allows for two control signals to each LUT, one for controlling inversion, and the other for selecting the path for signal propagation. A single binary signal is sufficient for selecting one of the target paths that may pass through an LUT. The last condition is required to produce a signal change at every destination for every test, simplifying the error detection logic.

With the above restrictions, LUTs on target paths will have one or two target paths through them. These LUTs are called 1-path LUTs and 2-path LUTs, respectively. The inputs that are not on target paths will be called free inputs.

The following procedure selects a set of target paths satisfying the conditions for multi-phase testing by selecting appropriate target paths for each set Si from the set of all target paths in the circuit. The union of these sets is the set of paths targeted in a test session. The procedure is then repeated for the remaining paths to obtain the target paths for subsequent test sessions until all paths are covered.

Procedure 1

1) Select a path that does not intersect any already selected path, as the main path to each destination.
2) For each main path, select a side path such that
    a. It meets the main path and shares the rest of the path with it.
    b. No other path meets the main path at the same LUT.
    c. It does not intersect any already selected target path (except for segments overlapping the main path).
3) Repeat Step 2 until no new side path can be found for any main path.
4) Find the number, n, of paths such that
    a. There are n target paths to each destination.
    b. The total number of paths is a maximum.
5) Select the main path and n − 1 side paths to each destination as the target paths for the session.



Example 1: Figure 3 shows all the target paths in a circuit. The source and destination flip-flops are omitted for the sake of clarity. We start Procedure 1 by (arbitrarily) selecting dAEJLy and hCGKMz as the main paths to the destinations y and z. Adding paths eAEJLy, cEJLy and fBFJLy to the first path, and jCGKMz, nDGKMz and qHKMz to the second, we get the set of target paths shown in heavy lines. Since there are four paths to each destination, the eight target paths shown can be tested in a single four-phase session.

The procedure can be repeated with the remaining paths to select sets of target paths for subsequent sessions. One possible set of test sessions is given in the following table, where the path(s) in the first row of each sessions were those chosen as the main path(s).

	Destination: y	Destination: z
Session 1	dAEJLy	hCGKMz
	eAEJLy	jCGKMz
	cEJLy	nDGKMz
	fBFJLy	qHKMz
Session 2	gBEJLy	gHKMz
	gFJLy	kDGKMz
Session 3	gBFJLy	mDGKMz
Session 4	hCFJLy	

jCFJL
kDGLy
Session 5 nDGLy
Session 6 mDGLy

The set of sessions may not be unique and depends on the choices made. Also note that not all sessions obtained are multiphase sessions. Session 3, for example, became a single-phase session because no path qualified as a side path of mDGKMz, which was arbitrarily chosen as the main path. No paths could be concurrently tested with those in Sessions 4, 5, and 6 because all paths to z had already been targeted. The sets of target paths obtained by Procedure 1 are such that each 2-path LUT has a main path and a side path through it. Thus, a single binary signal is sufficient to select the input through which the signal is to be propagated. Since the side path continues along the main path, selecting the appropriate input at the 2-path LUT where it meets the main path is sufficient for selecting the side path for testing. By using the same path selection signal, one side path to each destination can be selected simultaneously and tested in parallel.

The FPGA configuration for a test session is obtained by the following procedure:

Procedure 2

1) Configure a sequence generator and connect its output to the sources of all target paths of the session.
2) Configure a counter to control inversion parity, with the number of bits equal to the largest number of binate LUTs along any target path for the test session.
3) Configure a path selector to select the set of paths tested in each test phase, with the number of bits equal to the number of side paths to a destination.
4) Designate a free input of each LUT as its inversion control input p, and connect it to the counter output corresponding to its level.
5) Designate another free input of each 2-path LUT as its selector input s, and connect it to the path selector.
6) Modify the LUT of each 1-path LUT with on-path input a to implement f = a $\oplus$ p, if the original function is binate in a; otherwise f = a if it is positive or a if it is negative in a.
7) Modify the LUT of each 2-path LUT to implement f = $\overline{s} \cdot (a \oplus p) + s \cdot (b \oplus p)$

where a and b are on the main path and a side path, respectively.

The above modification for 2-path LUTs assumes that they are binate in both on-path inputs. If the output of a 2-path LUT is unate in a or b or both, a slightly different function f is needed. For example, if the LUT output is binate in a and negative in b, the modified LUT must implement $f = \overline{s} \cdot (a \oplus p) + s \cdot \overline{b}$.

Example 2:

Figure 4 shows the test structure for the circuit of Fig. 3.Only target paths that were selected for the first test session are shown, and all LUT functions are assumed to be binate in their inputs. The test circuitry consists of a sequence generator that produces a sequence of alternating 1's and 0's, a four-bit counter for inversion control and a path selector. The path selector is a shift register that produces an output sequence, 000, 100, 010, 001 for the 4-phase test of the first session in our example.

It can be verified from the figure that the main paths are selected when all selector outputs are 0. When any output is 1, exactly one side path to each destination is selected. Input transitions are applied to all paths simultaneously, but propagate only up to the first 2-path

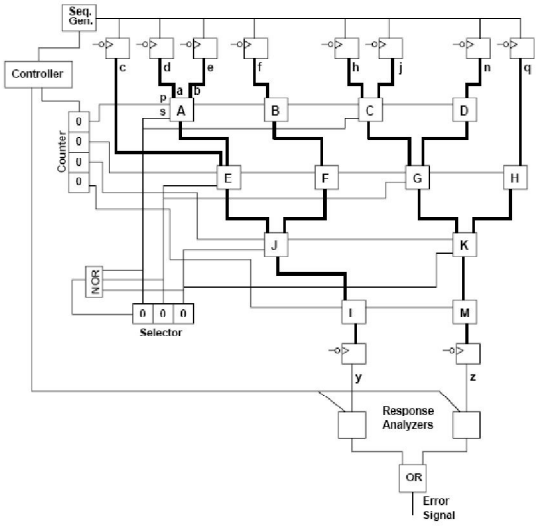


Fig. 4. Multi-phase test structure

LUT on all paths except the selected ones. Thus, only one path to each destination will have transitions along its entire length. since these paths are disjoint ,no interaction can occur among them.

V. CONCLUSION

In this paper, we have presented a new approach to testing selected sets of paths in FPGA-based circuits. Our approach tests these paths for all combinations of inversions along them to guarantee that the maximum delays along the tested paths will not exceed the clock period during normal operation. While the test method requires reconfiguring the FPGA for testing, the tested paths use the same connection wires, multiplexers and internal logic connections as the original circuit, ensuring the validity of the tests. Following testing, the test circuitry is removed from the device and the original user circuit is programmed into the FPGA.

Two methods have been presented for reducing the number of test configurations needed for a given set of paths. In one method, called the single-phase method, paths are selected so that all paths in each configuration can be tested in parallel. The second method, called the multi-phase method, attempts to test the paths in a configuration with a sequence of test phases, each of which tests a set of paths in parallel. Our experimental results with benchmark circuits show that these methods are viable, but the preferable method depends on the circuit structure. While our approach has been shown to be feasible, the algorithms presented are greedy algorithms that simply maximize the number of target paths tested in each configuration. They are by no means optimal and may not result in the smallest number of configurations or total test time. The use of other criteria, such as the total time for configuration and test application for each configuration, or better heuristics may lead to more efficient testing with the proposed approach.

REFERENCES

[1] M. Abramovici, C. Stroud, C. Hamilton, S. Wijesuriya, and V. Verma, “Using roving STARs for on-line testing and diagnosis of FPGAs in faulttolerant applications,” in IEEE Int. Test Conf., Atlantic City, NJ, Sept. 1999, pp. 28–30.

[2] M. Abramovici, C. Stroud, and J. Emmert, “Online BIST and BIST-based diagnosis of FPGA logic blocks,” IEEE Trans. on VLSI Systems, vol. 12, no. 12, pp. 1284–1294, Dec. 2004.

[3] I. G. Harris and R. Tessier, “Interconnect testing in cluster-base FPGA architectures,” in ACM/IEEE Design Automation Conf., Los Angeles, CA, June 2000, pp. 49–54.

[4] I. G. Harris and R. Tessier, “Testing and diagnosis of interconnect faults in cluster-based FPGA architectures,” IEEE Trans. on CAD, vol. 21, no. 11, pp. 1337–1343, Nov. 2002.

[5] W.K. Huang, F.J. Meyer, X-T. Chen, and F. Lombardi, “Testing configurable LUT-based FPGAs,” IEEE Trans. on VLSI Systems, vol. 6, no. 2, pp. 276–283, June 1998.

[6] C. Stroud, S. Konala, P. Chen, and M. Abramovici, “Built-in self-test of logic blocks in FPGAs (Finally, a free lunch),” in IEEE VLSI Test Symp., Princeton, NJ, Apr. 1996, pp. 387–392.

[7] C. Stroud, S.Wijesuriya, C. Hamilton, and M. Abramovici, “Built-in selftest of FPGA interconnect,” in IEEE Int. Test Conf., Washington, D.C., Oct. 1998, pp. 404–411.

[8] L. Zhao, D.M.H. Walker, and F. Lombardi, “IDDQ testing of bridging faults in logic resources of reprogrammable field programmable gate arrays,” IEEE Trans. on Computers, vol. 47, no. 10, pp. 1136–1152, Oct. 1998.

[9] M. Renovell, J. Figuras, and Y. Zorian, “Test of RAM-based FPGA: Methodology and application to the interconnect,” in IEEE VLSI Test Symp., Monterey, California, Apr. 1997, pp. 230–237.

[10] C-A. Chen and S.K. Gupta, “Design of efficient BIST test pattern generators for delay testing,” IEEE Trans. on CAD, vol. 15, no. 12, pp. 1568– 1575, Dec. 1996.

[11] S. Pilarski and A. Pierzynska, “BIST and delay fault detection,” in IEEE Int. Test Conf., Baltimore, MD, Oct. 1993, pp. 236–242.

[12] A. Krasniewski, “Application-dependent testing of FPGA delay faults,” in Euromicro Conf., Milan, Italy, Sept. 1999, pp. 260–267.