FPGA Bridging Fault Detection and Location via Differential I_{DDQ}

Erik Chmelař
Center for Reliable Computing
Stanford University
echmelar@crc.stanford.edu

Shahin Toutounchi
Xilinx, Inc.
shahin.toutounchi@xilinx.com

Abstract

Standard I_{DDQ} testing is limited by the ability to distinguish a small fault current from a large background leakage current: this limitation is overcome in FPGAs by differential I_{DDQ} testing. Partitioning of interconnects—dividing the interconnect resources of an FPGA into groups—further increases the detectability of a fault current.

Fault location can be achieved by iteratively applying partitioned differential I_{DDQ} testing to eliminate fault-free nets. The location algorithm, easily automated, requires very few configurations and I_{DDQ} measurements, logarithmic to the number of initially-suspected faulty nets.

1. Introduction

The background leakage current—the steady-state current drawn by an integrated circuit—has dramatically increased with the shrinking dimensions of deep sub-micron process technologies. Consequently, I_{DDQ} testing has become more challenging: it is difficult to distinguish a small fault current from a large background leakage current.

The I_{DDQ} of an FPGA is especially high due to (1) the large number of routing resources, and (2) the driving of unused interconnects to a fixed logic value to reduce noise and crosstalk. By using the magnitude of the difference between two I_{DDQ} measurements to detect an interconnect bridging fault, the large background leakage current is eliminated. Because the interconnection network consumes up to 80% of the die area, it is the primary challenge in FPGA testing.

In contrast to previous I_{DDQ} techniques, no test vectors are needed to activate faults: programming an FPGA with a test configuration sets the logic value of each interconnect. Additionally, by iteratively partitioning and eliminating interconnects, automated fault location is achieved.

The organization of this paper is as follows. Relevant I_{DDQ} techniques are surveyed in Sec. 2. A generic FPGA structure is given in Sec. 3. Differential I_{DDQ} is presented in Sec. 4, followed by partitioning in Sec. 5. Fault location is discussed in Sec. 6. Finally, the paper concludes in Sec. 7.

2. Previous Work

Certain shorts—shorts to power or ground and stuck-on transistor defects—are only detected via I_{DDQ} testing in various multiplexer implementations [1]. This is especially critical in an FPGA with switch matrices, which are implemented as transmission-gate multiplexers, [2, 3].

I_{DDQ} is used to detect an excessive current caused by a defect. Standard, or single-threshold, I_{DDQ} testing involves applying test vectors that set the logic value of some internal nodes to activate bridging faults. For each test vector an I_{DDQ} is measured (after transients settle). If any measurement is greater than some threshold, the device fails [4–8]. Differential I_{DDQ} testing, current signature analysis, and current ratios have been proposed as improvements.

Differential I_{DDQ} testing, or ΔI_{DDQ}, employs the discrete first derivative of I_{DDQ} as a function of vector number, i, to detect a fault, $\Delta I_{DDQ_i} = I_{DDQ_i} - I_{DDQ_{i-1}}$, [9–11]. Because the difference between successive pairs of I_{DDQ} measurements is used, vector order is very important.

Current signature analysis does not place a restriction on vector order: an I_{DDQ} is measured for every vector and then ordered based on magnitude. A discontinuity in the resulting I_{DDQ} *signature*, greater than some threshold, indicates a bridging fault is present [12, 13]. Storage and comparison of I_{DDQ} measurements to only the maximum and minimum previously-measured values is required for implementation.

Finally, the current ratio method also employs a comparison to previously-determined maximum and minimum values. Dynamic thresholds are calculated based on the ratio of the maximum to the minimum [14].

3. FPGA Structure

An FPGA contains both logic and routing resources. The logic resources are the hardware within the basic building blocks: *logic blocks*, *input/output blocks*, multiplier blocks, and block RAMs. A logic block contains the combinational and sequential elements needed to perform logic functions: *SRAM Look-up Tables* (LUTs) implement combinational functions and bistables are used in sequential designs. An input/output block is used to pass signals between an FPGA and an external device. Blocks are interconnected by the routing resources, or interconnection network: interconnects, *switch matrices*, *Programmable Interconnect Points* (PIPs), multiplexers, buffers, and vias. A switch matrix (or programmable multiplexer) is made up primarily of PIPs, joining interconnects to form a *net*. A PIP is a pass transistor controlled by an associated memory cell that determines whether the PIP is *on* (conducting) or *off* (non-conducting).

Finally, an FPGA is *configured*, or programmed, with a configuration *bitstream*, that determines which logic and routing resources are used, and in what manner.

4. Differential I_{DDQ} Testing

4.1. Overview

In the differential I_{DDQ} testing method for FPGAs, test configurations replace test vectors. Each configuration sets the logic value of some interconnects to the opposite logic value as that of the rest and the resulting I_{DDQ} is measured.

First, the device is configured such that all interconnects are driven to the same logic value, say logic-1, and a reference current, $I_{DDQ_{ref}}$, is measured. Since no bridging faults are activated, the reference current is simply the background leakage current, for example the sum of all source-to-substrate leakage and sub-threshold drain currents.

Next, the device is configured a number of times, each time setting some subset of the interconnects to the opposite logic value as that of the rest, say logic-0. For each configuration, a total current, $I_{DDQ_{tot}}$, is measured. Since any bridging fault between any interconnect driven to logic-0 and any driven to logic-1 is activated, each total current includes a possible fault current, $I_{DDQ_{fault}}$, an interconnect leakage current between the opposite-valued interconnects, $I_{DDQ_{int}}$, and the original background leakage current, $I_{DDQ_{ref}}$.

By subtracting the reference current (measured only once) from a single total current to get a single signature current, $I_{DDQ_{sig}}$, a small fault current is more easily detected. However, because the difference in two measurements is used, thereby doubling the variance, it is not possible to detect arbitrarily small faults currents.

$$\begin{aligned} I_{DDQ_{sig}} &= I_{DDQ_{tot}} - I_{DDQ_{ref}} \\ &= (I_{DDQ_{ref}} + I_{DDQ_{int}} + I_{DDQ_{fault}}) - I_{DDQ_{ref}} \\ &= I_{DDQ_{int}} + I_{DDQ_{fault}} \end{aligned}$$

4.2. Test Configurations

In a configured FPGA there are *used* and *unused* interconnects: the former are part of some net in a configuration while the latter are not. Each net is driven by either a LUT or a bistable. Only device configuration is needed to control these logic elements and thus set the logic values of the used interconnects. The Boolean function a LUT implements determines its output value: logic-0 when $F = 0$, logic-1 when $F = 1$. The initial condition of a bistable determines its output value: logic-0 when $init = 0$, logic-1 when $init = 1$. Unused interconnects are driven to a fixed logic value by hardware, logic-1 in Xilinx and Altera FPGAs.

4.2.1. Application-independent FPGA

An application-independent FPGA is one whose configuration may change several times throughout its lifetime. Since the customer's designs are not known *a priori*, all resources must be guaranteed to be defect-free by the manufacturer. Using differential I_{DDQ} testing to detect all interconnect bridging faults requires generating several test configurations that set adjacent interconnects to opposite logic values. In each configuration the used interconnects are driven to logic-0, activating any bridging fault to any unused interconnect (logic-1). Figure 1 shows the per-configuration steps for differential I_{DDQ} testing. Steps 1 and 2 can be interchanged with steps 3 and 4, respectively.

1. Configure FPGA to drive all interconnects to logic-1
2. Measure reference current, $I_{DDQ_{ref}}$
3. Configure FPGA to drive used interconnects to logic-0 and unused interconnects to logic-1
4. Measure total current, $I_{DDQ_{tot}}$
5. Determine signature current, $I_{DDQ_{sig}} = I_{DDQ_{tot}} - I_{DDQ_{ref}}$
6. Is $I_{DDQ_{sig}} > threshold$?
   (a) Yes, device fails configuration
   (b) No, device passes configuration

Figure 1: Per-configuration Test Flow

Minimizing the number of configurations, and thus the number of I_{DDQ} measurements, is paramount to reducing test time, since it takes roughly 4–8 ms to configure a Xilinx Virtex-II FPGA (dependent on size) and nearly 20 ms to measure the I_{DDQ}. Although there are thousands of interconnects, the tiled layout and bus-oriented (grouped) interconnection network can be exploited to minimize the number of configurations needed to test for all single and most multiple bridging faults between adjacent interconnects*.

In a Virtex-II FPGA with eight metal layers [2], assume each interconnect type is routed within a single layer: *horizontal long*, *vertical long*, *horizontal hex*, *vertical hex*, *horizontal double*, *vertical double*, *direct*, and *fast*. Driving

*Bridging faults involving more than two adjacent interconnects are not considered in the number-of-configurations estimate. To detect these faults, several additional configurations are needed.

Table 1: XC3S50 Differential I_{DDQ} Experimental Results

Device	$I_{DDQ_{ref}}$ (mA)	Fault-free		Faulty		$I_{DDQ_{fault}}$ (mA)	d_s	d_r	i
		$I_{DDQ_{tot}}$ (mA)	$I_{DDQ_{sig}}$ (mA)	$I_{DDQ_{tot}}$ (mA)	$I_{DDQ_{sig}}$ (mA)				
1	1.66	1.81	0.15	2.09	0.42	0.27	0.13	0.64	5.0
2	1.72	1.91	0.19	2.20	0.48	0.29	0.13	0.60	4.6
3	1.53	1.66	0.12	1.93	0.40	0.27	0.14	0.69	4.9
4	1.90	2.07	0.17	2.35	0.45	0.28	0.12	0.63	5.2
5	1.60	1.74	0.14	2.01	0.41	0.27	0.14	0.66	4.9

every other interconnect of a particular type to logic-0 (interleaving [15]) activates all faults between adjacent interconnects within a metal layer. Interleaving on only every other layer ensures that bridging faults between layers are also activated. Thus, in addition to the configuration used to measure the reference current, only four configurations are needed to detect all bridging faults: two for odd metal layers and two for even layers. Layout information is needed for a more accurate (and admittedly slightly larger) estimate.

4.2.2. Application-dependent FPGA

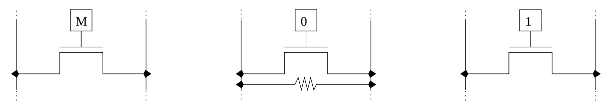
An application-dependent FPGA is one whose configuration remains unchanged throughout its lifetime. Only the resources enabling the customer's design to function—the used resources—are guaranteed to be defect-free: unused resources may be defective. Currently Xilinx offers application-dependent FPGAs through EasyPath [16].

Although bridging faults between used interconnects may be detected by Boolean testing [17–20], bridging faults between used and unused interconnects (used-unused bridging faults) may not be. Due to the extensive use of NMOS pass transistors in the interconnection network, which pass a weak logic-1 but a strong logic-0, a bridging defect behaves as a wired-AND fault: the stronger logic-0 dominates. Consequently, if unused interconnects are driven to logic-1, a used-unused bridging fault appears only as a delay on the used interconnect, which may not be detected by Boolean testing. Detection of used-unused bridging faults is important for high device reliability and low power consumption.

In addition to the configuration used to measure the reference current, just one configuration can detect all used-unused bridging faults. All used LUTs are configured to implement the function $F = 0$ and all used bistables are given the initial condition $init = 0$: routing of nets remains unchanged. To prevent a *set/reset* input from causing a bistable to drive a logic-1 on its output net, it must be made active-high or synchronous, or disconnected.

4.3. Experimental Results

A fault is injected in a dense configuration that uses 3.66% of all PIPs. An unused PIP joining a used and unused interconnect is programmed to be on, emulating a bridging defect of roughly 1 kΩ or stuck-on PIP [21]. Figure 2a shows a PIP and its associated SRAM cell, Fig. 2b shows a PIP in the off state and a real bridging defect, and Fig. 2c shows a PIP in the on state emulating a bridging defect.



(a) PIP Joining Interconnects (b) Real Bridging Defect (c) Emulated Bridging Defect

Figure 2: Bridging Defect Emulation

The *detectability* of a fault current is a measure of how large it is with respect to the I_{DDQ} measurement: a larger detectability means the fault current is more easily detected. In standard I_{DDQ} testing, the detectability, d_s, is the ratio of the fault current, $I_{DDQ_{fault}}$, and the total current, $I_{DDQ_{tot}}$. In differential I_{DDQ} testing, the detectability, d_r, becomes the ratio of the fault current, $I_{DDQ_{fault}}$, and the signature current, $I_{DDQ_{sig}}$. Finally, the *improvement*, i, of differential I_{DDQ} testing over standard I_{DDQ} testing is the ratio of the detectabilities of the two methods, d_r and d_s.

$$d_s = \frac{I_{DDQ_{fault}}}{I_{DDQ_{tot}}} \qquad d_r = \frac{I_{DDQ_{fault}}}{I_{DDQ_{sig}}} \qquad i = \frac{d_r}{d_s} = \frac{I_{DDQ_{tot}}}{I_{DDQ_{sig}}}$$

Table 1 shows the resulting reference currents, $I_{DDQ_{ref}}$, the total and signature currents for the fault-free and faulty cases, $I_{DDQ_{tot}}$ and $I_{DDQ_{sig}}$, respectively, and the fault currents, $I_{DDQ_{fault}}$, for five Spartan-III, 90 nm XC3S50 devices. It can be seen that differential I_{DDQ} testing has approximately a five-fold improvement over standard I_{DDQ} testing for the injected bridging fault. Furthermore, note that the total current of device 4 in the fault-free case is greater than that of devices 3 and 5 in the faulty case. If the threshold for standard I_{DDQ} testing is set to, say 2.0 mA, device 4 would fail when fault-free (false failure) and device 3 would pass when faulty (test escape). In contrast, a threshold for differential I_{DDQ} testing can be unambiguously determined, since for each device the faulty signature current is at least double the fault-free signature current.

5. Partitioning

5.1. Overview

The amount of background leakage current (reference current) increases with FPGA size. Also, interconnect leakage current, and thus total current, increases at a faster rate than reference current because of the larger number of PIPs contributing to the leakage between opposite-valued interconnects. This is further aggravated by shrinking manufacturing process dimensions. Consequently, signature current increases, making it more difficult to detect a bridging fault.

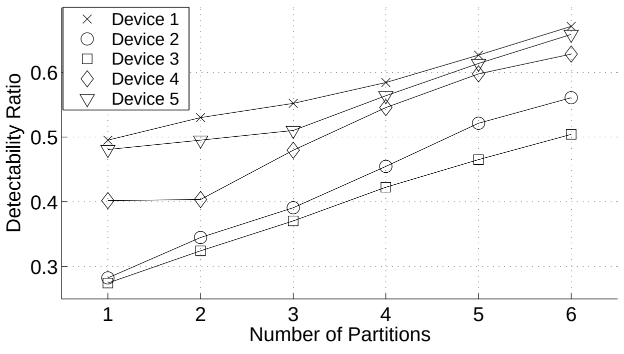
Consider a single test configuration whose used interconnects compose the set S. This configuration, that tests for bridging faults between any interconnect in S and any interconnect not in S, can be divided, or *partitioned*, into N smaller configurations, each testing only a subset of the interconnects of S. Thus, instead of measuring one total current for the original configuration, N total currents are measured, one for each of the N smaller configurations,

For simplicity and without loss of generality, assume that each partition contains $1/N^{th}$ of the interconnects in S. Because each partition also contains approximately $1/N^{th}$ the number of PIPs, each interconnect leakage current is reduced by N. Since the reference current, still measured only once, does not change, each signature current (except for any corresponding to a partition that contains a fault) is also reduced by N. Therefore, the detectability ratio increases: the fault current is more easily detected. Note that the partitions of S need not be disjoint: the only requirement is that every interconnect of S belongs to at least one partition. Additionally, N threshold values are now required.

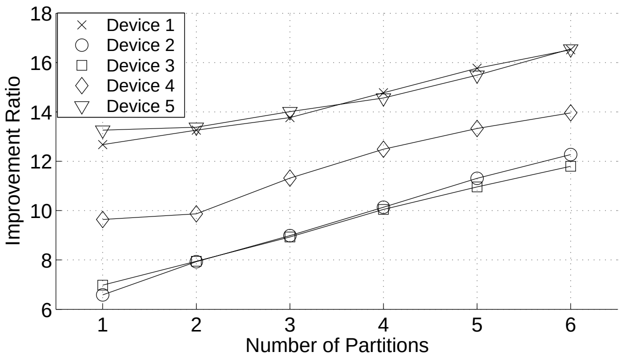
5.2. Experimental Results

A fault is injected in a very dense sequential configuration that uses 2.99% of all PIPs. For each of five Spartan-III, 90 nm XC3S1000 devices, the I_{DDQ} is measured as described in Sec 4.3. Figures 3a and 3b show the resulting detectability and improvement ratios, d_r and i, respectively, for one to six partitions. Both the detectability and improvement ratios increase with the number of partitions. Roughly six partitions are needed to obtain detectability ratios approximately equal to those of the smaller XC3S50 devices.

It is found for the larger XC3S1000 devices that the injected bridging fault results in a fault current, $I_{DDQ_{fault}}$, approximately twice as large as that for the smaller XC3S50 devices. Subsequently, the improvement of differential I_{DDQ} testing over standard I_{DDQ} testing is much greater for the larger FPGAs. For a single partition an increase between six- and fourteen-fold results, and for six partitions an increase between twelve- and sixteen-fold results.



(a) Detectability Ratio, d_r



(b) Improvement Ratio, i

Figure 3: Experimental Results for Partitioned FPGA

6. Fault Location

6.1. Overview

Partitioned differential I_{DDQ} testing can be used to locate a bridging fault to a single net, a *faulty net*, which contains one of the bridged interconnects. The location algorithm, locating single or multiple bridging faults, requires very few configurations and I_{DDQ} measurements, logarithmic to the number of initially-suspected faulty nets. The technique is therefore practical for even the largest FPGAs.

Initially, all of the used interconnects of a failing differential I_{DDQ} test configuration (for an application-independent FPGA) or failing design (for an application-dependent FPGA) are suspected faulty. Because an entire net, not an individual interconnect, is driven by a logic element, the location algorithm identifies a faulty net. Once a net is located, the individual faulty interconnect can be determined by a remove-and-reroute technique [22] using differential I_{DDQ} testing, which will not be discussed further.

A faulty net is located by iteratively applying partitioned differential I_{DDQ} testing to eliminate fault-free nets from the set of suspected faulty nets, S, until only a single net re-

Table 2: XC3S50 Fault Location

Iteration	Partition 1		Partition 2		S After Elimination
	P_1	$I_{DDQ_{sig_1}}$ (mA)	P_2	$I_{DDQ_{sig_2}}$ (mA)	
0	$CLK, n_0, n_1, n_2, n_3, n_4$	0.21	-	-	$CLK, n_0, n_1, n_2, n_3, n_4$
1	CLK, n_0, n_1	0.00	n_2, n_3, n_4	0.21	n_2, n_3, n_4
2	n_2	0.00	n_3, n_4	0.21	n_3, n_4
3	n_3	0.21	n_4	0.00	n_3

mains. Two criteria must be met during each iteration. To ensure convergence, no net of S can be included in all N partitions. Furthermore, without knowing the expected magnitude of the total current for each partition, S must be partitioned evenly, based primarily on the number of nets and the number of PIPs attached to each net, such that the interconnect leakage currents for all partitions are approximately equal. By balancing these currents, a partition containing a fault (faulty partition) will display a higher signature current than one that does not contain a fault (fault-free partition).

6.2. Fault Search

For simplicity and without loss of generality, assume a binary search is used to locate a single fault, such that in each iteration S is always divided into two partitions, P_1 and P_2. If the criteria discussed in Sec. 6.1 are met, the faulty partition will display a signature current noticeably higher than that of the fault-free partition. The nets of the fault-free partition, all fault-free, are therefore eliminated from S. The reduced set S is subsequently repartitioned, and a signature current for each partition is again obtained. The process repeats until S contains only the faulty net ($|S| = 1$). Figure 4 shows the general fault location algorithm.

```
while |$S$| > 1
   $P_1 \ldots P_N$ = partition($S$,$N$)
   foreach $P_i$
      measure $I_{DDQ_{sig_i}}$
   foreach $P_i$
      if $I_{DDQ_{sig_i}}$ < max($I_{DDQ_{sig_1}}, \ldots, I_{DDQ_{sig_N}}$)
         $S$ = $S$ - $P_i$
```

Figure 4: Fault Location Algorithm

During each iteration two current measurements are made and S is reduced in size by two. Given an initial set S with M nets ($|S| = M$), only $2\lceil log_2 M \rceil + 1$ configurations and current measurements are required to locate the faulty net ($+1$ is for the single reference current measurement and the corresponding configuration). The number of configurations and current measurements therefore scales logarithmically to the number of initially-suspected faulty nets: the technique can be used even for very large FPGAs.

To detect multiple faults, the fault location algorithm is applied independently to each partition displaying an elevated signature current during a particular iteration, or to all partitions in the case where all signature currents are equal. To decrease the probability that all partitions of a particular iteration display an elevated signature current, the number of partitions of S can be increased.

6.3. Experimental Results

6.3.1. Small FPGA

An XC3S50 is configured with a design—the decade counter shown in Figure 5—and a fault is injected on n_3 as described in Sec. 4.3.

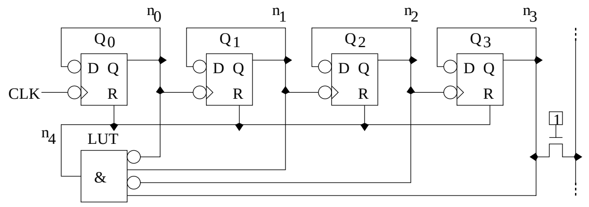
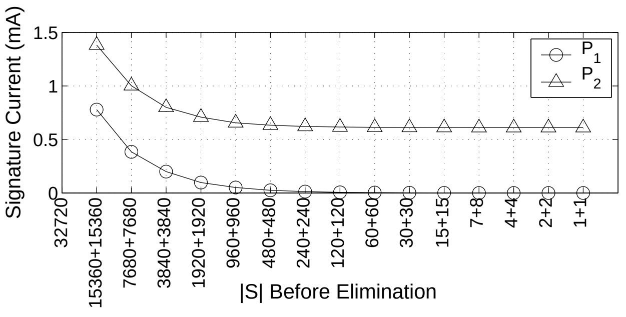


Figure 5: Decade Counter Sample Circuit

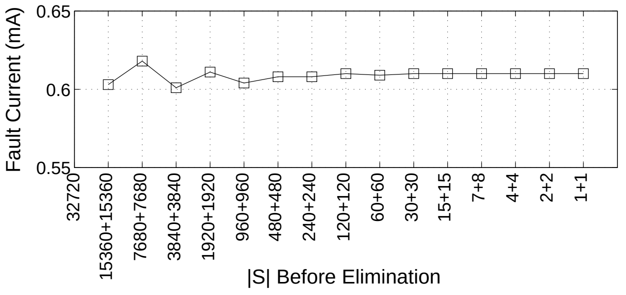
Table 2 shows the signature currents, $I_{DDQ_{sig_1}}$ and $I_{DDQ_{sig_2}}$, for partitions P_1 and P_2, respectively (a binary search is used). Iteration 0 (before beginning the actual fault location) shows a bridging fault exists due to the larger-than-expected signature current, $I_{DDQ_{sig_1}}$. Note that because the CLK net is driven from off-chip, setting its logic value is done by driving either a logic-0 or logic-1 from the tester. Additionally, the signature current of the fault-free partition always measures 0.00 mA because the corresponding interconnect leakage current is negligibly small when S contains very few interconnects.

6.3.2. Large FPGA

An XC3S1000 is configured with a test configuration that uses all LUTs and bistables of the device, resulting in a total of 30720 nets initially in S. A fault, injected as described in Sec. 4.3, is located using an automated version of the fault location algorithm via binary search. Figure 6a shows the signature currents for both partitions during each iteration, plotted such that P_1 is always fault-free. The signature current of the fault-free partition converges to zero and the signature current of the faulty partition converges to the fault current, $I_{DDQ_{fault}} = 0.61$ mA, shown in Fig. 6b.



(a) Signature Currents



(b) Difference in Signature Currents

Figure 6: XC3S1000 Experimental Results

Determining which partition is faulty and which is fault-free is very easy in the automated algorithm, since during each iteration the partitions are balanced: the difference in signature currents is always within 3% of the resulting fault current. Only $\lceil log_2 30720 \rceil = 15$ iterations, and thus $2\lceil log_2 30720 \rceil + 1 = 31$ configurations and current measurements, are required to identify the faulty net.

7. Conclusion

Differential I_{DDQ} testing shows a five-fold or greater improvement in fault current detectability over standard I_{DDQ} testing for the injected bridging defect (roughly 1 kΩ). Partitioning further increases the improvement at least 12-fold. Approximately five configurations can test for all interconnect bridging faults both within and between metal layers.

In addition to fault detection, fault location can be achieved by iteratively applying partitioned differential I_{DDQ} testing to eliminate fault-free nets. Easily automated, fault location requires very few configurations and current measurements, logarithmic to the number of initially-suspected faulty nets; it can therefore be used efficiently for even the largest FPGAs.

References

[1] S. Makar and E. McCluskey, "Some faults need an I_{DDQ} test," *Proc. IEEE Int. Workshop I_{DDQ} Testing*, pp. 102–103, 1996.

[2] Xilinx, Inc., *The Programmable Logic Data Book*. Xilinx, Inc., San Jose, CA, 2002.

[3] D. Fernandes and I. Harris, "Application of built-in self test for interconnect testing of FPGAs," *Proc. Int. Test Conf.*, pp. 1248–1257, 2003.

[4] R. Gulati and C. Hawkins, *I_{DDQ} Testing of VLSI Circuits*. Boston: Kluwer Academic Publishers, 1992.

[5] R. Rajsuman, *I_{DDQ} Testing for CMOS VLSI*. Artech House, 1994.

[6] S. Chakravarty and P. Thadikaran, *Introduction to I_{DDQ} Testing*. Boston: Kluwer Academic Publishers, 1997.

[7] C. Hawkins, J. Soden, R. Fritzemeier, and L. Horning, "Quiescent power supply current measurement for CMOS IC defect detection," *IEEE Trans. Industrial Electronics*, vol. 36, pp. 211–218, May 1989.

[8] L. Horning, J. Soden, R. Fritzemeier, and C. Hawkins, "Measurement of quiescent power supply current for CMOS ICs in production testing," *Proc. Int. Test Conf.*, pp. 300–309, Sept. 1987.

[9] C. Thibeault, "On the comparison of ΔI_{DDQ} and I_{DDQ} testing," *Proc. 17th VLSI Test Symp.*, pp. 143–150, 1999.

[10] A. Miller, "I_{DDQ} testing in deep submicron integrated circuits," *Proc. Int. Test Conf.*, pp. 724–729, Sept. 1999.

[11] T. Powell, J. Pair, M. S. John, and D. Counce, "Delta I_{DDQ} for testing reliability," *Proc. 18th VLSI Test Symp.*, pp. 439–443, 2000.

[12] A. Gattiker and W. Maly, "Current signatures [VLSI circuit testing]," *Proc. 14th VLSI Test Symp.*, pp. 112–117, 1996.

[13] A. Gattiker, P. Nigh, D. Grosch, and W. Maly, "Current signatures for production testing [CMOS ICs]," *IEEE Int. Workshop I_{DDQ} Testing*, pp. 25–28, Oct. 1996.

[14] P. Maxwell, P. O'Neill, R. Aitken, R. Dudley, N. Jaarsma, M. Quach, and D. Wiseman, "Current ratios: A self-scaling technique for production testing," *Proc. Int. Test Conf.*, pp. 1148–1156, 2000.

[15] E. Chmelar, "FPGA interconnect delay fault testing," *Proc. Int. Test Conf.*, pp. 1239–1247, 2003.

[16] Xilinx, Inc., "Xilinx easypath solutions." http://www.xilinx.com/xlnx/xil_prodcat_product.jsp?title=v2_easypath, 2003.

[17] M. Tahoori, "Using satisfiability in application-dependent testing of FPGA interconnects," *Proc. 40th Design Automation Conf.*, pp. 678–681, June 2003.

[18] M. Niamat, R. Nambiar, and M. Jamali, "A BIST scheme for testing the interconnects of SRAM-based FPGAs," *45th Midwest Symp. Circuits and Systems*, vol. 2, pp. 41–44, Aug. 2002.

[19] I. Harris and R. Tessier, "Interconnect testing in cluster-based FPGA architectures," *Proc. 37th Design Automation Conf.*, pp. 49–54, June 2000.

[20] I. Harris and R. Tessier, "Testing and diagnosis of interconnect faults in cluster-based FPGA architectures," *IEEE Trans. Computer-Aided Design Integrated Circuits*, vol. 21, pp. 1337–1343, Nov. 2002.

[21] S. Toutounchi, A. Calderone, Z. Ling, R. Patrie, E. Thorne, and R. Wells, "Fault emulation testing of programmable logic devices," *US Patent US6594610B1*, 2003.

[22] M. Tahoori, "Diagnosis of open defects in FPGA interconnect," *Proc. Int. Test Conf.*, pp. 328–331, Dec. 2002.