Center for Reliable Computing

TECHNICAL REPORT

CRC Publications in VLSI Test Symposium 2004

04-2
(CSL TR # 04-2)

February 2004

Center for Reliable Computing
Gates Bldg. 2A, Room #236
Stanford University
Stanford, California 94305-9020

Abstract:

This technical report includes the following publications in the 22nd IEEE VLSI Test Symposium (VTS'04):

1. "ELF-Murphy Data on Defects and Test Sets", by E. J. McCluskey, Ahmad Al-Yamani, James C.-M Li, Chao-Wen Tseng, Erik Volkerink, Francois-Fabien Ferhani, Edward Li, and Subhasish Mitra.
2. "Delay Defect Screening Using Process Monitor Structures", by S. Mitra, E. Volkerink, E. J. McCluskey, and Stefan Eichenberger.
3. " Relative IDDQ Testing for Bridging Faults in FPGAs", by E. Chmelar, and S. Toutounchi.

Funding:

This work was supported by LSI Logic under contract number 16517, by NSF under contract number 0098218, and by Xilinx under contract number 24287, and in collaboration with Advantest and Philips.

Imprimatur:

ELF-Murphy Data on Defects and Test Sets

E. J. McCluskey, Ahmad Al-Yamani, James C.-M Li, Chao-Wen Tseng,
Erik Volkerink, Francois-Fabien Ferhani, Edward Li, and Subhasish Mitra
Stanford Center for Reliable Computing[*]
http://crc.stanford.edu

Abstract

We at CRC have designed and LSI Logic has manufactured two test chip designs; these were used to investigate the characteristics of actual production defects and the effectiveness of various test techniques in detecting their presence. This paper presents a characterization of the defects that shows that very few defective chips act as if they had a single-stuck fault present and that most of the defects cause sequence-dependent behavior.

A variety of techniques are used to reduce the size of test sets for digital chips. They typically rely on preserving the single-stuck-fault coverage of the test set. This strategy doesn't guarantee that the defect coverage is retained. This paper presents data obtained from applying a variety of test sets on two chips (Murphy and ELF35) and recording the test escapes. The reductions in test size can thus be compared with the increases in test escapes. The data shows that, even when the fault coverage is preserved, there is a penalty in test quality. Also presented is the data showing the effect of reducing the fault coverage. Techniques studied include various single-stuck-fault models including inserting faults at the inputs of complex gates such as adders, multiplexers, etc. This technique is compatible with the use of structural RTL netlists. Other techniques presented include compaction techniques and don't care bit assignment strategies.

1. Introduction

This paper presents the data that we collected on a tester and compares it with fault model derived data. Accuracy and effectiveness demonstrated by this data are commented on. The paper focuses on results from the ELF35 chip and compares some of the results with their Murphy counterparts.

The two test chips were designed to permit very thorough and varied tests to be applied and the corresponding response data to be collected. The defects that are present on the chips are only those that occurred naturally during fabrication. No artificial defects were inserted. We were interested to compare the results for these two chips from different technologies. Reliability defects are also of interest, but will be discussed in another paper.

The Murphy chip design contains 4 copies each of 5 different very simple completely combinational cores (called Circuits under Test or CUTs in previous publications). Two cores are data path structures and the other 3 are control logic designs.

The ELF35 chip design contains multiple copies each of 6 different cores. Two of the cores are sequential data path structures (two different implementations of the 2901 arithmetic processor). The other 4 are combinational (three data path designs and one translator).

Besides nominal voltage testing, Very-Low-Voltage (VLV) testing and IDDQ testing were also applied to all the packaged chips. In VLV testing, the supply voltage is 1.4V for ELF35 and 1.7V for Murphy. These voltages are about two times the transistor threshold voltage [Chang 96]. In IDDQ testing, the threshold was set to 100 σA for ELF35 and 300 σA for Murphy, which are typical values that LSI Logic uses for chips of comparable sizes in these technologies. A weak suspect core is a core that passed every test at nominal voltage but failed VLV or IDDQ.

Figure 1 shows the ELF35 core classification tree. A total of 495 interesting cores are identified. Interesting cores are the union of defective cores and weak suspect cores. Among these 495 interesting cores, 324 of them are defective and 171 of them are weak suspects. Among the 324 defective cores, 101 of them are FOSTS (fail only some test sets) and the others are FATS (fail all test sets). Among the 171 weak suspect cores, 130 of them failed only IDDQ testing and 9 of them failed only VLV testing. The other 32 failed both VLV and IDDQ testing. Figure 2 shows the same classification tree for the Murphy chips.

Most of our tester data was collected by applying patterns obtained from ATPG programs. The rationale for using many sources of patterns was either to minimize any bias caused by a particular ATPG source or because some tools have capabilities lacking in other tools. We have generated (or tool vendors have donated) various test sets from many academic tools (including Rutgers University, Texas A&M, University of Illinois, University of Iowa, and Stanford CRC) and commercial tools (including Fastscan, Sunrise, Syntest, and Tetramax.)

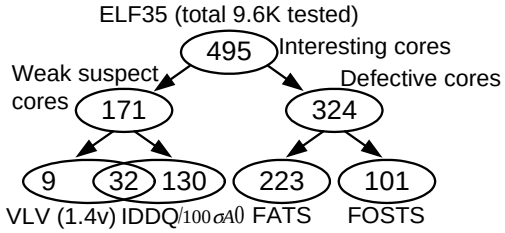


Figure 1. ELF35 classification.

[*] James Li is currently with National Taiwan University, Chaowen Tseng is currently with Zettacom, Edward Li is currently with Sun Microsystems and Subhasish Mitra is currently with Intel.

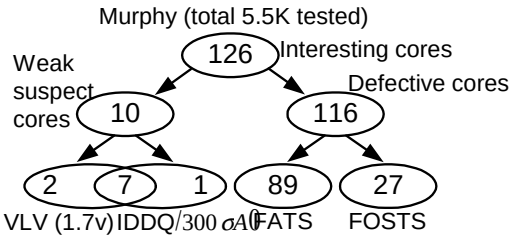


Figure 2. Murphy classification.

2. Characteristics of the defects

2.1. Sequence dependence

Since combinational circuits contain no memory elements, the response to a particular input combination should not depend on previous input combinations[§]. The insertion of a single- (or multiple-) stuck-at fault should not cause a combinational circuit to act as a sequential circuit, by exhibiting dependence of its output on previous inputs. Neither should other faults such as **non-feedback** bridging faults. We thought it would be interesting to check whether the defects on our chips transformed our combinational logic circuits into sequential circuits. To do this we applied **each** of our 100% single-stuck-at fault model test sets six times, each time using the same set of patterns but applying them in a different order. Order 1 is that obtained from the ATPG tool, order 2 is the same set of patterns with an all-0 pattern inserted between each pair of original vectors. Order 3 inserts an all-1 pattern instead of the all-0 pattern. Order 4 inserts the bit-wise complement between each pair of patterns. Order 5 inserts a one bit shift between each pair of patterns and order 6 applies the original patterns in the reverse order.

43% of the defective Murphy chips and 42% of the defective ELF35 chips had sequence dependent test responses. Clearly the defects in these chips are not acting like single-stuck-at faults. The defects in these chips changed them from combinational circuits to sequential circuits. Naturally, we wondered what kinds of defects were causing this behavior. One possible defect that could do this is one that acts like a stuck-open fault [Li 02].[→] By matching the tester traces (response data) to the simulated circuit response in the presence of a particular stuck-open fault, Li identified 9 of the 45 Murphy sequence-dependent chips that act as if they contain defects causing such faults**,** [Li 02]**.** We have not carried out failure mode analysis to confirm this diagnosis.

Another possible defect that could cause sequence-dependent behavior is one that causes a feedback bridging fault. We have not yet succeeded in diagnosing all of the chips with sequence-dependent test responses and are also trying to diagnose the defective Elf35 chips. This study is continuing.

[§] Often taken as the definition of a combinational circuit.

[→] This defect inserts a capacitive dynamic memory.

2.2. Timing-dependent defects

We define a timing dependent defect as one that escapes the test at some speeds within the specification range and is detected at some others of the same test set. The timing dependent defects were found by applying all of our 100% single-stuck test sets at 3 different speeds. Some of the defective chips with sequence-dependent behavior also have output responses that depend on the speed of the test: 105 (32%) of the defective Elf35 chips and 39 (34%) of the defective Murphy chips. Possible causes of such behavior are *resistive-opens*, connections that have significantly higher resistance than intended or transistors with lower drive than designed for.

2.3. Single stuck-at faults

A bare majority (58%) of the defects in the ELF35 chips are *combinational defects.* They cause the faulty chips to continue to act like combinational circuits. Some of these chips might be modeled as having single- stuck-at faults. To investigate this, we used the same technique of matching tester response data with simulated response; in this case the simulation was for circuits with single stuck-at faults, [Li 02]**.** Only 15 (5%) of the defective Elf35 chips act like circuits with single- stuck-at faults; more of the defective Murphy chips – 41 (35%) – behave like they have single-stuck-at faults.

The ratios of various defect types present in the Elf35 and Murphy chips are summarized in Figure 3 and Figure 4. This data clearly shows that the single-stuck fault model is not an accurate representation of the behavior of a chip in the presence of a manufacturing defect.[§] This suggests that the stuck-at fault model should not be relied on in diagnosing defects on faulty chips. On the other hand, the stuck-at fault model has been very effective when used to generate test patterns. The next section discusses using the stuck-at fault model for applications other than diagnosis.

3. The stuck-at fault model

This section describes using tester data for screening out defective chips rather than diagnosing defects. Most of the defective chips failed all of the test sets that we applied. We call these *FATS or Fail All Test Sets.* Out of the 324 defective ELF35 chips 101 are FOSTS and the 116 Murphy chips include 27 that fail only some test sets. Some of the test sets applied were not very thorough such as the 50% single stuck-at test set. Only the remaining FOSTS (fail only some test sets) chips are relevant to the study in this section of the effectiveness of various test techniques. Thus, the data presented here excludes the FATS chips.

One of the most important roles of the single stuck-at fault model is as a metric for evaluating the thoroughness of a test set. We will discuss this first and then mention some other applications.

[§] Multiple stuck faults do not avoid the difficulties of the single-stuck fault model. While there is evidence that some defective chips behave as if they had multiple faults, there are still the issues of sequence dependence and complexity.

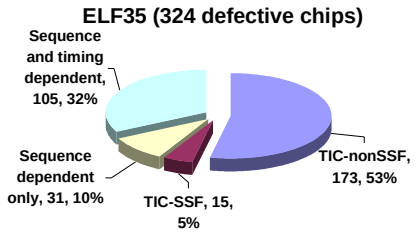


Figure 3. Defect characteristics for ELF35.(TIC = Timing-Independent Combinational).

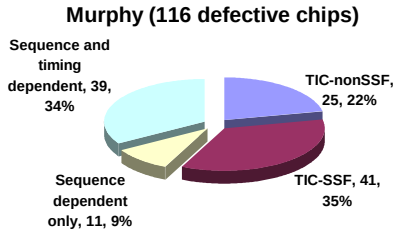


Figure 4. Defect characteristics for Murphy.

Everyone reading this paper knows what the single stuck-at fault model is; or do we each have our own definition? We would probably all agree that some node in the network is fixed at a logic value (0 or 1) independent of the values of any other nodes in the network. The areas of possible disagreement are: (a) which network representation and (b) which nodes should have fixed values.

The network could be represented using the design file made up either of gates from the cell library or using structural RTL. These library gates typically include the elementary gates: AND, OR, NAND, NOR as well as some complex gates such as XOR gates, multiplexers, full adders, etc. Another possible network representation would use only elementary gates, replacing each complex gate with a network of elementary gates having the same functionality.§ Thus, at least two different network representations are currently used.

The other issue is the set of nodes from which to choose the node with the fixed logic value. The most careful approach is to include all primary inputs, elementary gate inputs and outputs, and primary outputs. This and other models are listed in Table 1.

Models 1, 2 and 4 are each supported by some commercial ATPG tools. There are theoretical results suggesting that Model 5 can be just as effective as Model 1 in generating test patterns [Mei 75].

§ Some commercial ATPG tools provide an option of deriving this representation automatically. This representation may not correspond precisely to the actual silicon implementation since it isn't always possible to find the correct primitive gate equivalent of a complex gate (the library information may not be exact).

Table 1. List of Single Stuck-at Fault Models.

Model	Fault sites
1. Elementary Gate Faults	All elementary gate inputs, elementary gate outputs, primary inputs and outputs
2. Complex Gate Faults	All library gate inputs, library gate outputs, primary inputs and outputs (RTL faults)
3. Partially Complex Gate Faults	All elementary gate inputs and outputs, fan-out free library elements inputs and outputs, and primary inputs and outputs.
4. Gate-output Faults	All gate outputs (all nets), primary inputs and outputs. Gates can be complex or elementary.
5. Dominance-reduced faults	All inputs and output of fanout-free subnetworks of elementary gates, primary inputs and outputs

The way the single stuck-at fault model is used in connection with test pattern generation is by means of a program that attempts to generate input patterns causing the network output with the fault present in the network to differ from the output of the fault-free network. The *metric* or figure of merit for the set of patterns generated is the single stuck-at fault coverage, the percentage of the modeled faults that are detected by some pattern in the set. Clearly this value depends on which single stuck-at fault model is used.

But the real issue is the effectiveness of the model in producing test sets that detect the defects. We investigated this by generating test sets using each of these models, applying them to our faulty Murphy and Elf35 chips, and determining how many faulty chips were not detected by each of the test sets. A closely related issue is what percentage of the single stuck-at faults is detected by the test set, the *fault coverage*; if the fault coverage is less than 100%, does that mean that more defective chips will escape detection? Results answering this and many other test quality related questions are presented in the next section.

4. Single-stuck test data reduction

In this section, we present data relevant to the impact of test data reduction techniques on the quality of the test. We quantify the quality by the number of defective chips that escape a test set (test escapes).

4.1. Test set compaction

The number of patterns in a test set, *test set size*, is an important characteristic of the test set; it affects the amount of tester memory and test application time [Hamzaoglu 00]. Reducing the test set size is an important goal, especially if it can be done without sacrificing defective chip detection. Commercial *ATPG*, automatic test pattern generation, tools typically give the user a choice of (1) dynamic test compaction, (2) static test compaction, or (3) no compaction. These techniques take advantage of the fact that test patterns typically contain a large percentage of unspecified (don't care) bits [Barnhart 01]. *Dynamic compaction* is performed by running fault simulation at several stages of the test pattern generation process and dropping the faults that are detected by the generated patterns. *Static compaction* is performed by combining the patterns that don't have any conflicts in the specified bit

positions. Some ATPG tools reverse the order of the generated patterns and then perform simulation and drop the patterns that don't detect additional faults. This is believed to reduce the test data because the faults that are detected with the initial patterns are most of the time easy faults that have a high detectability (many patterns detect them). The last patterns in the test set normally detect low detectability faults. That is why reversing the pattern order eliminates the need for some of them.

Compaction preserves the fault coverage, but since there are fewer patterns it is possible that the defect detection suffers. This is sometimes discussed by calling the ability of patterns to detect defects that don't correspond to single stuck-at faults *collateral coverage* and the corresponding faults *unmodeled faults*. We now know that most of the defects are not accurately represented as single stuck-at fault; thus most of the defects correspond to unmodeled faults. In any event, it is important to determine whether compaction reduces the ability of the test set to detect defects. In order to investigate the effect of compaction on test escapes we tested the ELF chips using both compacted and uncompacted test sets. The results are shown in Figure 5. The figure shows the number of escapes that occurred with uncompacted test sets and the increase in the escapes due to compaction for various fault coverages. The results in the figure suggest that at all fault coverages used in the experiment, there is a significant price for compaction. Since we are not disclosing the tool used, we are reporting the results from the tool with the maximum increase in escapes at each fault coverage applied.

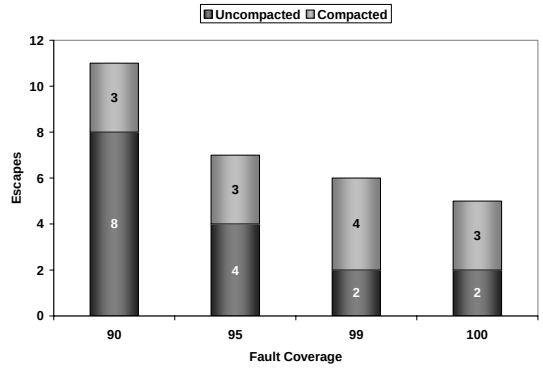


Figure 5 Increase in escapes due to compaction.

shows the test lengths for the uncompacted tool 1C test sets and the reduction in test length obtained by compaction. The other tools had similar results. Although compaction preserves fault coverage, the results show that it doesn't preserve defect coverage. On the other hand, the percentage reduction in test size gained by compaction is significant as shown in the summary of this section.

4.2. Fault coverage reduction

Reducing fault coverage requirements is another way to reduce test set size. This option is appealing because it's widely observed that the last small increase in fault coverage requires a considerable number of test patterns. Eliminating these patterns results in a far smaller percentage reduction in fault coverage than in test length.

Table 2. Uncompacted Test Lengths and the Compaction Reduction for Tool 1C.

	100% Fault Coverage		99% Fault Coverage		95% Fault Coverage	
	TL	÷ TL Comp	TL	÷ TL Comp	TL	÷ TL Comp
LSI	318	128	317	124	173	27
TOPS	518	202	502	174	310	122
SQR	42	20	40	18	36	13
M12	72	31	58	19	47	19
MA	103	50	66	6	32	0
PB	3176	489	2887	364	2198	174

Using three commercial ATPG tools, we generated test sets with fault coverage varying between 50% and 100% for the ELF35 cores. The number of escapes caused by decreasing the fault coverage is plotted in . Similar results for Murphy cores are plotted in .

The figures show that reducing fault coverage (even by a small fraction) consistently comes with a price in defect coverage. For all tools, the plots show that the higher the fault coverage used the higher the defect coverage achieved. Although it is not an accurate model for the actual defects (as shown in the previous section), the single-stuck fault model is a good measure of the thoroughness of the test.

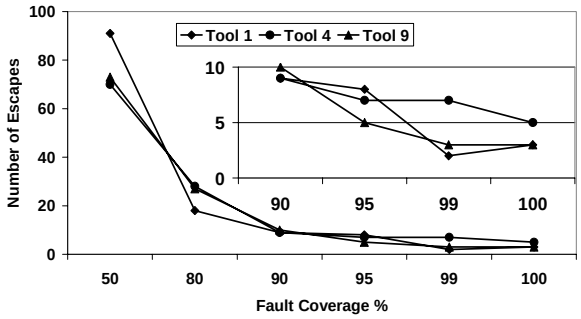


Figure 6. Test escapes vs. fault coverage for compacted test sets generated for complex gates.

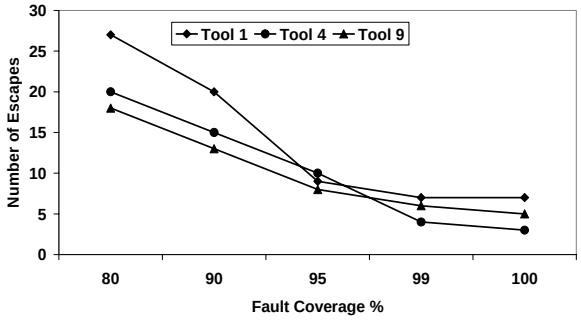


Figure 7. Test escapes vs. fault coverage for compacted test sets generated for complex gates.

4.3. Using complex gates for fault sites

Using complex gates nodes as fault sites instead of elementary gates leads to fewer fault sites in the circuit. This may reduce the number of test patterns required to test the circuit. In a way, placing faults at the RTL pins is an extreme in using complex gates as fault sites. We applied test patterns generated using both the elementary gate fault model and the complex gate fault model (Table 1). The results are shown in . The figure shows the number of escapes using elementary gates and the increase in escapes with complex gates using different fault coverage values (compacted and uncompacted). The increase in escapes shown is obtained from the tool that gave the maximum increase in escapes at each fault coverage value. The results demonstrate that there can be a definite penalty in the number of test escapes due to using the complex gate model rather than the elementary gate model. The figure also shows that this applies to compacted and uncompacted test sets. Based on these results, complex gate faulting results in a considerable degradation in test quality.

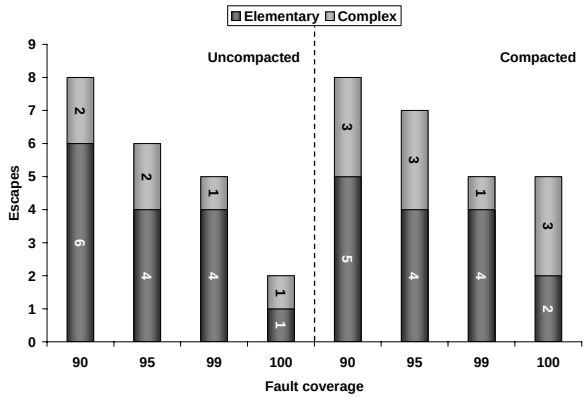


Figure 8. Escapes for complex gates vs. elementary gates as fault sites.

4.4. Using only gates outputs for fault sites

Another technique that is sometimes used to reduce the *fault list*, the number of faults for which to generate test patterns, is to consider stuck faults at only gate outputs rather than at both gate inputs and outputs. This, in effect, eliminates the possibility of a fanout branch having a stuck value while the other branches and the stem are fault-free. The test set is reduced as a consequence of using only gate outputs as fault sites. Table 3 shows the number of tester escapes that occurred for the ELF35 chip when test sets were generated using the design netlist with faults at both gate inputs and outputs compared to test sets with faults only at gate outputs. There is a consistent increase in the test escapes when the gate inputs are not faulted.

Table 3. Comparison of test escapes for faults at both complex gate inputs and outputs and faults only on complex gate outputs – Tool4

Coverage	90%	95%	99%	100%
SSF test for gates inputs and outputs as fault sites	6	4	4	4
SSF test for gates outputs only as fault sites	8	6	5	4
Difference	2	2	1	0

4.5. Assignment of unspecified bits

Compression techniques are widely used to reduce test data. In compression, some coding theory concepts are used to reduce the test pattern storage requirement. Additional decoding circuitry is needed on chip to decode the stored data into the actual test patterns. In many cases, a particular assignment of don't care values is used to maximize the compression ratio. This is the case when run-length encoding is used. Also, for some tester architectures, repeating the last care bit value through the following don't care bits reduces storage requirements.

We tried different don't care bit assignment options to find out their impact on test quality. We implemented *one-fill* (using value 1 for all don't cares), *zero-fill, repeat fill* (repeating the last care bit) and *random-fill.* The resulting data is shown in . The results show clear penalties for using the same value to fill the don't care bits.

Table 4. Comparison of different don't care assignment options.

	LSI2901		TOPS2901	
	Length	Escapes	Length	Escapes
One fill	5215	4	8590	5
Zero fill	5215	7	8590	9
Repeat fill	5215	0	8590	0
Random fill	5215	0	8590	0

4.6. Test data reduction summary

The previous subsections presented individual data for various test data reduction techniques. Two main criteria are what matters in test data reduction, the amount of reduction in test data and the impact on test quality. Having the chance to evaluate this impact based on real defect data with our ELF35 chips, we summarized these data for all test set reduction techniques using all tools we have.

Figure 9 shows a graph relating the increase in defective chips that escape the test and the percentage reduction in test set size with different reduction techniques for three commercial tools. The reference for the test reduction and the increase in escapes is a100% SSF test set generated for elementary gates with each tool. In this graph, the closer the technique is to the bottom left corner the better it is. Reducing the fault coverage requirement to 90% gives the maximum reduction in test data but at the same time increases considerably the number of escapes. The reader is invited to draw all combinations of conclusions that serve his or her interest. An interesting observation is that, for tool 9, reducing the coverage requirement to 95% is better than compaction. It results in further reduction in the test set while maintaining the same quality level. For the other two tools, compaction is even better than reducing the fault coverage to 99%. The inserted graph in the figure has the same data for tools 1 and 4 only. We separated these two tools from the third one because they, unlike tool 9, had comparable results for most of the reduction options.

Another interesting observation is that at 100% coverage, using complex gate pins as fault sites instead of elementary gates did not result in any penalties in test quality for tools 1 and 4. Doing the same with tool 9 resulted in a penalty in test quality. Using only gate outputs

as fault sites was available with one of the tools only and it was worse than compacting the test set.

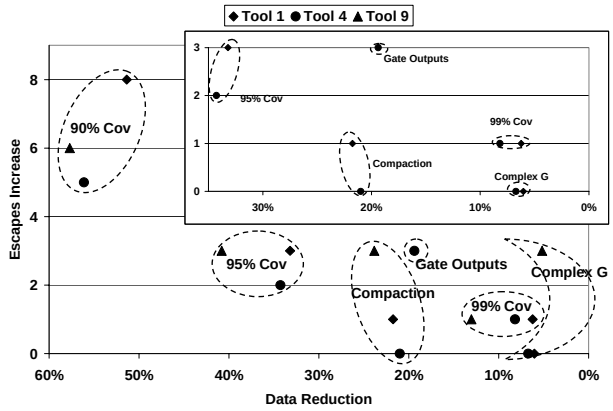


Figure 9.Increase in escapes vs. reduction in test data for the three commercial tools.

5. N-detect results

We learned from results in and related results that higher fault coverage leads to higher defect coverage and that 100% fault coverage tests missed some defects. This made us wonder whether there was some way to generate a more thorough single stuck-at fault test set. One way to do this is to have a test set in which each single stuck-at fault is detected more than once. This is called an *N-detect test set*. In a 2-detect test set, each single stuck-at fault is detected by at least two **different** test patterns [Ma 95], [McCluskey 00].

The data collected by applying test sets with varying fault coverages on the tester is shown in for ELF35 and for Murphy, where 2A is an academic ATPG tool. A *tester escape* or *escape* for short is defined as one defective chip that is not detected by any of the patterns in a particular test set. lists the number of tester escapes for test sets with single-stuck fault coverages varying from 15-detect to 50%. The columns labeled 1C, 4C and 9C correspond to three different commercial ATPG tools. The design file was used to generate the test sets.

Table 5. Number of test escapes vs. fault coverage of compacted test sets for ELF35.

Tools			1C	4C	9C
SSF	N-Detect	15	0	3	-
		10	1	2	-
		5	2	1	-
		3	2	4	-
		2	3	5	-
	Fault	1.00	3	5	3
		0.99	2	7	3
		0.95	8	7	5
		0.90	9	9	10
		0.80	18	28	27
		0.50	91	70	73

The data in the two tables above clearly suggests that the *thoroughness* of a test set measured by the number of defective chips that escape detection by that test set is strongly correlated to the test set fault coverage.

TARO (Transition faults propagated to All Reachable Outputs) from [Tseng 01] was applied to all combinational cores of ELF35 and Murphy. All of the TARO test sets resulted in zero escpaes for all cores they were applied to.

Table 6. Number of test escapes vs. fault coverage of compacted test sets for Murphy.

Tools			1C	4C	9C	2A
SSF	N-Detect	15	-	0	-	0
		10	-	-	-	0
		5	-	0	-	0
		3	-	-	-	0
		2	-	-	-	2
	Fault Coverage	1.00	7	3	5	3
		0.99	7	4	6	-
		0.95	9	10	8	-
		0.90	20	15	13	-
		0.80	27	20	18	-

Table 7 Test Escapes of Sequential and Scan Test Sets.

Test set#	Description	**LSI2901 (total 89 defective chips)**					**TOPS2901 (total 29 defective chips)**				
		Cov %	Test length	Escapes	Tester data	Tester Time	Cov %	Test length	Escapes	Tester data	Tester time
1C	Scan SSF	100	319	2	386K	174K	100	504	0	1M	485K
4C	Scan SSF	100	193	2	234K	105K	100	386	0	781K	371K
1C-.8	Scan SSF	80	64	5	78K	35K	80	114	1	230K	110K
4C-.8	Scan SSF	80	33	7	40K	18K	80	72	2	145K	69K
3C.s	Seq. SSF	77	517	4**	64K	0.5K	-	-	-	-	-
4C.s	Seq. SSF	75	710	4**	87K	0.7K	-	-	-	-	-
5A	Seq. SSF	78	888	4**	109K	0.9K	65	1,498	3*	154K	1.5K
D.0	Verif.	82	3,121	4**	384K	3K	55	429	12	44K	0.4K
T.3C	Scan transition	100	653	2	80K	356K	80	100	0	10K	96K
T.3C.s	Seq. transition	82	4,070	2*	500K	4K	-	-	-	-	-

* includes one slow escape ** includes two slow escapes

6. Sequential ATPG results

Table 7 compares the test escapes of sequential test sets and scan test sets (also known as structural tests) for the sequential cores in ELF35. Their single-stuck fault coverage and test length are shown for the reader's reference. The test lengths of scan test sets are the number of scan loads. There are 544 cycles in a scan load of LSI2901 and 961 cycles in a scan load of TOPS2901. The test lengths of sequential test sets are the number of system clocks. A dash "-" in this table means the corresponding test set is not available. The tester data column corresponds to the number of bits that need to be stored in the tester for each test set. The tester time column corresponds to the number of clock cycles needed to test the core with the given test set.

The sequential test sets were applied at three different speeds. There are some cores that failed sequential tests at characterized speed and escaped the test only at slow speed. For scan test sets, the system clocks are also applied at three different speeds and the scan load and unload operations are applied at a fixed clock rate of 1MHz.

The table shows that sequential test sets are effective when applied at speed. Figure 3 shows that more than 30% of the defects are timing defects. If the test is not applied at speed then scan testing yields a better test quality.

The verification test sets in Table 7 were provided by the designers. They are fault graded to have 82% SSF fault coverage for one core and 55% for the other. They had 4 test escapes for one core and 12 for the other. The last two rows in the table show the transition fault test sets generated by tool 3C. They both had two test escapes. The test length of the sequential transition fault test set is more than 4 times longer than the SSF sequential test set.

7. Conclusions

We classified the defects based on their behavior and found that even though 35% of the defects in Murphy behaved like SSFs, in the newer chip only 5% of the defects did. We also found that almost half of the defects for both chips exhibited sequential behavior. This suggests that ATPG techniques that ignore the order of the test patterns run the risk of missing many of these defects. For ELF35, only two test sets had no test escapes: TARO and 15-detect.

We studied a number of test set size reduction techniques: compaction, complex gates, gate outputs, fault coverage reduction, etc. Some of the techniques preserved fault coverage but none of them could be relied on to preserve test quality.

Acknowledgements

This research is supported by LSI Logic Corp, Agilent, Intel, NSF and SRC. We would like to thank Guy Dupenloup, Scott Keller and Prabhu Krishnamurthy.

We would like to thank Advantest, Mike Purtell (Advantest), Don Sireci (Advantest), Marc Loranger (Credence), Dr. Sassan Raissi (Digital Testing Services), and Steven Liaw (ARTest) for their donation of tester time.

We would like to thank Michael Grimaila, Gary Greenstein, Ilker Hamzaoglu, Michael Hsiao, Seiji Kajihara, Rohit Kapur, Ray Mercer, Irith Pomeranz , John Waicucauski, and L.T. Wang for test sets and ATPG tools.

We would like to thank Jonathan Chang, Ray Chen, Eddie Cheng, Kan-Yuan Cheng, Yi-Chin Chu, Siyad Ma, Samy Makar, and Sanjay Wattal from CRC for their help.

References

[Barnhart 01] K. Barnhart, B. Keller, B. Koenemann, and R. Walther, "OPMISR: The Foundation for Compressed ATPG Vectors", Proc. ITC, 2001.

[Chang 96]Chang, J., and E.J. McCluskey, "Quantitative Analysis of Very-Low-Voltage Testing," VLSI Test Symposium, pp.332-337, 1996.

[Chang 98] Chang, J. and et. al., "Analysis of Pattern-dependent and Timing-dependent Failures in an Experimental Test Chip," Proc. ITC, 1998.

[Franco 95] Franco, P. and et. al., "An Experimental Chip to Evaluate Test Techniques Chip and Experiment Design," Proc. ITC, pp.653-662, 1995.

[Hamzaoglu 00] Hamzaoglu, I., and J. Patel, "Test set compaction algorithms for combinational circuits," IEEE Trans on CAD, Vol. 19, No. 8, pp. 957-963, Aug. 2000.

[Li 99] Li, J.C.M., J.T.-Y. Chang, C.W. Tseng, and E.J. McCluskey, "ELF35 Experiment - Chip and Experiment Design," CRC TR 99-3, Oct. 1999.

[Li 02] Li, C.-M.J., and E.J. McCluskey, "Diagnosis of Sequence Dependent Chips," 20th IEEE VLSI Test Symposium (VTS'02), Monterey, CA, Apr. 28-May 2, 02.

[LSI 97] LSI Logic, "G-10p Cell Based ASIC Product," Feb. 1997.

[Ma 95] Ma, S.C., P. Franco, and E.J. McCluskey, " An Experimental Chip To evaluate Test Techniques Experimental Results," Proc. ITC, pp.663-672, 1995.

[Mei 75] Mei, K.C.Y., Dominance Relations of Stuck-at and Bridging Faults in Logic Networks, Ph.D. Thesis, Stanford University, Stanford, CA, June 1975.

[McCluskey 00] McCluskey, Edward .J. and C.W. Tseng, "Stuck-at Fault versus Actual Defects", Proceeding of International Test Conference, pp.336-344, 2000.

[Tseng 01] C.W. Tseng and E.J. McCluskey, "Multiple-output propagation transition fault test," Proc. ITC, 2001.

Appendix

The Murphy chip, was discussed along with some of its data in [McCluskey 00]. LSI Logic fabricated the Murphy chip in their LFT150K CMOS gate array technology (L_{eff} = 0.7 σ). It has 25k gates in a 120-pin Ceramic PGA package with 96 signal pins. V_{dd} is 5 volts. This paper presents data for the 116 chips that failed at least one of the 265 test sets applied at 3 supply voltages and 4 test speeds. One objective of this paper is to compare the Murphy data with the data collected on the ELF35 chip, a more recent technology.

LSI Logic fabricated the ELF35 chip in their G10P standard cell technology (Leff = 0.35 σ). It has 265k gates in a 272-pin plastic BGA package with 96 signal pins. Vdd is 3.3 volts. Over ten thousand chips were tested. This paper presents data for the 324 chips that failed at least one of the 278 test sets applied at 2 voltages and 3 test speeds.

DELAY DEFECT SCREENING USING PROCESS MONITOR STRUCTURES

Subhasish Mitra[1,2] Erik Volkerink[2] Edward J. McCluskey[2] Stefan Eichenberger[3]

[1] Intel Corporation, Sacramento, CA, USA
[2] Center for Reliable Computing, Stanford University, Stanford, CA, USA
[3] Philips Semiconductors, Nijmegen, The Netherlands

Abstract

This paper presents delay test data collected from test chips fabricated in a 0.18σ technology. The experimental data shows that process monitor structures such as on-chip ring oscillators are effective in identifying slow parts while performing transition fault testing at frequencies slower than the rated frequency.

1. Introduction

Delay defects affect the operating speed but not the static functionality of manufactured circuits. An IC which is supposed to operate correctly at 1 GHz, for example, may produce incorrect outputs for a few input combinations when operated at 1 GHz in the presence of a delay defect. However, depending on the amount of extra delay introduced by the defect, the part may produce correct outputs for all input combinations when operated at 800 MHz. Delay defects represent a significant fraction of total defect population so that it is necessary to apply delay tests to keep the quality level (often referred to as defective parts per million or DPM) acceptably low [McCluskey 00, Saxena 02, Kim 03].

The maximum frequency at which a part operates correctly for a given operating condition is influenced by three major factors: process spread, process shift and the presence of spot defects. Often, the term process variation is used to indicate both process spread and process shift effects. *Process variation* is a small natural variation in physical parameters from one manufactured part to the other. It is often assumed that the distribution of delays of manufactured parts due to process variation follows a normal distribution function. Figure 1.1 shows such a theoretical normal distribution. Process variation causes parts to have delays greater or less than the mean delay (Fig. 1.1) – this is referred to as the *process spread*. The more the standard deviation of the delay distribution, the more is the process spread. Process excursions can change the mean of the delay distribution – this is referred to as *process shift*. Tight process control generally keeps process variation within set boundaries but cannot completely eliminate it. Delay defects can affect a large portion of the die and hence the speed of many paths in a given design – these are often termed as *global delay defects* or *distributed delay defects*. A *random delay defect*, also called a *spot defect*, occurs in a single location and only affects the speeds of paths passing through that location.

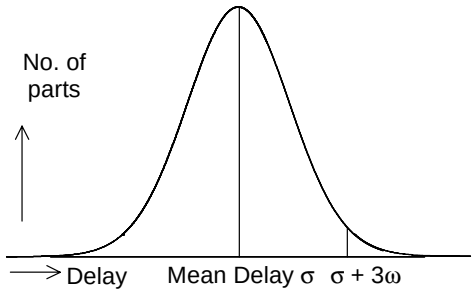


Figure 1.1. Normal distribution for process variation.

The basic objective of delay testing is to ensure that a shipped part produces correct outputs when operated at the *application clock period* (called the *rated clock period* [1]). In certain circumstances, delay testing is performed to identify the frequency at which the part will operate correctly – this is called *speed binning*. In this paper, we're not concerned with speed binning. The rated frequency of a design is chosen in such a way that the majority of the manufactured parts produce correct outputs when operated at a speed greater than or equal to the rated frequency. For example, in the normal distribution of Fig.1.1 suppose that the mean delay is σ units and the standard deviation is ω units. If the application clock period is σ units, then roughly 50% of the manufactured parts will have delays greater than the application clock period due to process variation. If most parts must have delays very close to the mean delay, then the standard deviation must be very small which means the process must be very well-controlled. From a normal distributions, 99.9999% parts will have delays less than σ + 6ω units. However, if the process is not very well-controlled (i.e., ω is not very small), then application clock period of σ + 6ω units may be too slow for the product to be sold in the market. More realistically, if the application clock period is σ + 3ω units, then roughly 0.13% of the manufactured parts will have a slower speed due to process variation. Hence, all parts must be tested for extra delays introduced due to process variation.

[1] The frequency is also referred to as the *rated frequency*.

In addition, suppose that a spot defect lands on the circuit introducing an extra delay of 2ns. The *slack* of a path is the difference between the application clock period and the propagation delay of that path. If all paths passing through the defective region have slacks more than 2ns, then the defective circuit will not produce incorrect outputs at that particular application speed. Of course, this defect can worsen during normal operation causing the circuit to fail in the field. In contrast, if the slack of a path passing through the defect location is less than 2ns, then incorrect outputs may be produced when the circuit propagates transitions along that particular path.

This paper is based on experimental data collected from more than 70,000 test chips fabricated in a 0.18σ technology by Philips – the Stanford CRC ELF18 experiment. The main idea is to detect parts with extra delays caused by process variation using on-chip process monitor structures such as ring oscillators. Parts with spot defects are detected using transition fault testing. Test escapes resulting from slow transition fault testing where the capture clock period is greater than the application clock period are reported. The *yield loss* (good parts declared defective, also called *overkill*) and *test escapes* (defective parts not detected) associated with this test technique are studied.

Section 2 briefly discusses the ELF18 experiment. The delay characteristics of the cores studied in this paper are discussed in Sec. 3. We analyze the delay testing technique combining transition fault testing and process monitor structure characteristics in Sec. 4 and 5. Open issues are discussed in Sec. 5 followed by a review of previous work in Sec.6 and conclusions in Sec. 7.

2. Overview of ELF18 Experiment

The ELF18 chips are manufactured in the Philips 0.18σ Corelib technology. The die size is 31.5mm^2 and each chip consists of 26 cores – die-id cores, RAM cores, ROM cores, analog cores, library evaluation cores, 6 DSP cores and a chip controller core. The R.E.A.L. Digital Signal Processor [Kievits 98] is implemented in the DSP cores. The characteristics of the DSP core implementation are reported in Table 2.1.

Table 2.1. Characteristics of DSP cores

Gate Count	Flip-flop count	Size	Clock	Inter-connect layers	Scan chains
53,732	1,428	0.76 mm^2	1	6	10

One of the DSP cores has two ring oscillators referred to as Ring 1 and Ring 2 in this paper. Hence, this DSP core, referred to as DSP1, is the main focus of this paper. The basic repeating block of the ring oscillators consists of an inverting delay element, and a load at the output of the inverting element. Both Ring1 and Ring2 consist of 200 inverting delay elements spread all over the DSP1 core. However, the distinction between these two oscillators is that, Ring 1 has 100 pairs of inverters spread over DSP1 core – i.e., the two inverters in a pair are connected locally and there is global routing between the inverter pairs. In contrast, Ring2 has all 200 inverters spread over the core.

3. DSP1 Delay Characteristics

Stuck-at and transition fault test patterns were applied to the DSP cores as summarized in Table 3.1. For the transition fault test patterns, for each DSP1 core the *minimum capture clock period* (referred to as the *minimum clock period* [2]) at which that core produced correct outputs was recorded.

Table 3.1. Tests Applied to DSP cores on ELF18 Chips

Test Type	Fault Coverage	Voltage	Capture clock period
Single Stuck-at	99%	1.8 V	100ns, 50ns
		1.65 V	100ns
		1.95 V	100ns
Transition: Launch on Capture	97%	1.8 V	Minimum clock period recorded

Figure 3.1 shows the distribution of the minimum clock periods of the DSP1 cores. The mean and the standard deviation of the minimum clock periods of DSP1 cores are calculated from the collected experimental data using standard techniques [Rumsey 03]. In Figs. 3.2a and 3.2b we plot the minimum clock periods of DSP1 cores and the time periods of the corresponding Ring1 and Ring2, respectively. We fitted the minimum clock period to Ring1 and Ring2 time periods using linear regression [Rumsey 03] to obtain the following relationships:

$$\text{Min. Clock Period} = 0.6 + 0.05251 \times \text{Ring1 Time Period},$$

$$\text{Min. Clock Period} = 0.52 + 0.1122 \times \text{Ring2 Time Period}.$$

As can be seen, the largest part of the sample population follows this prediction very well while outliers are clearly visible. The corresponding *root-mean-square error* is:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\text{Predicted min. period} - \text{Actual min. period})^2}$$

If Ring1 time period is used, for example, to predict the minimum clock period using the relationship shown above, then the root mean square error is 0.57ns when we consider all DSP1 cores and 0.09ns when we consider all DSP1 cores with minimum clock periods less than or equal to 12ns. The majority of DSP1 cores with minimum clock periods greater than 12ns do not correlate with the corresponding ring oscillator time periods.

In Figs. 3.2a and 3.2b, a few parts have predicted minimum clock periods larger than the actual ones. These can be explained by slow ring oscillators. There are

[2] This corresponds to maximum frequency called Fmax.

several parts with predicted minimum clock periods smaller than actual. Reason for these behaviors may be spot defects, such as resistive vias, contacts and metal lines or process variation induced delays that the ring oscillators are insensitive to. These parts are analyzed in Sec. 5.

4. Transition Fault Testing using Ring Oscillators

There are three major questions related to any transition fault testing strategy: 1. Transition fault pattern generation (e.g., TARO [Tseng 01], Transition along longest paths [Park 88, Sharma 02, Qiu 03]); 2. Test application in a scan-based design (e.g., launch on last shift, launch on capture); 3. Capture clock period (e.g., slower or faster than application clock period [Yan 03]).

In this section we study the third aspect. The total number of DSP1 cores that were used (after removing parts that failed single stuck-at tests) is more than 70,000. In Figs. 4.1a, 4.1b and 4.1c we examine three possible application clock periods of DSP1 cores – 10ns, 10.5ns and 11ns. Only 194, 62 and 8 of all DSP1 cores have minimum clock periods greater than 10ns, 10.5ns and 11ns, respectively, and less than 11.5ns. Cores with minimum clock periods greater than the application clock period and smaller than or equal to the transition fault test capture clock period are treated as *escapes*. Suppose that the application clock period is 10ns. If no transition fault test is applied, there will be 285 escapes. In contrast, if the transition fault test capture clock period is 15ns, there will be 216 escapes. Since no functional test patterns were applied, the assumption is that there will be no escapes if the capture clock period is the same as the application clock period – 10ns for this example.

For 10ns application clock period, there will be 131 escapes even if the capture clock period is 10.5ns. From quality level standpoint, 131 escapes out of more than 70,000 total parts indicate a serious problem.

In contrast, consider the following test strategy: (1) Apply transition fault tests with capture clock period of 15ns and reject all DSP1 cores that fail this test. (2) Examine time periods of on-chip ring oscillators (e.g., Ring2) of all passing cores and reject all cores whose Ring2 time periods are greater than or equal to 80ns. From Fig. 4.1a, there are only 34 escapes. Compare this number with 216 escapes if only transition fault testing is performed with 15ns capture clock period. At 10.5ns capture clock period, there are only 5 escapes using the ring oscillator threshold.

There are two fundamental parameters associated with this new test strategy: 1. The capture clock period, *T*, of transition fault testing; 2. The ring oscillator time period threshold, *R*. The test strategy is shown below:

> New Delay Testing Approach
>
> Input: *T*: Capture clock period of transition fault testing
> *R*: Ring oscillator time period threshold
>
> Steps:
> 1. Apply transition fault testing with capture clock period *T* to all parts. Reject all parts failing this test.
>
> 2. Reject all parts passing Step 1 but having ring oscillator time period greater than or equal to *R*.

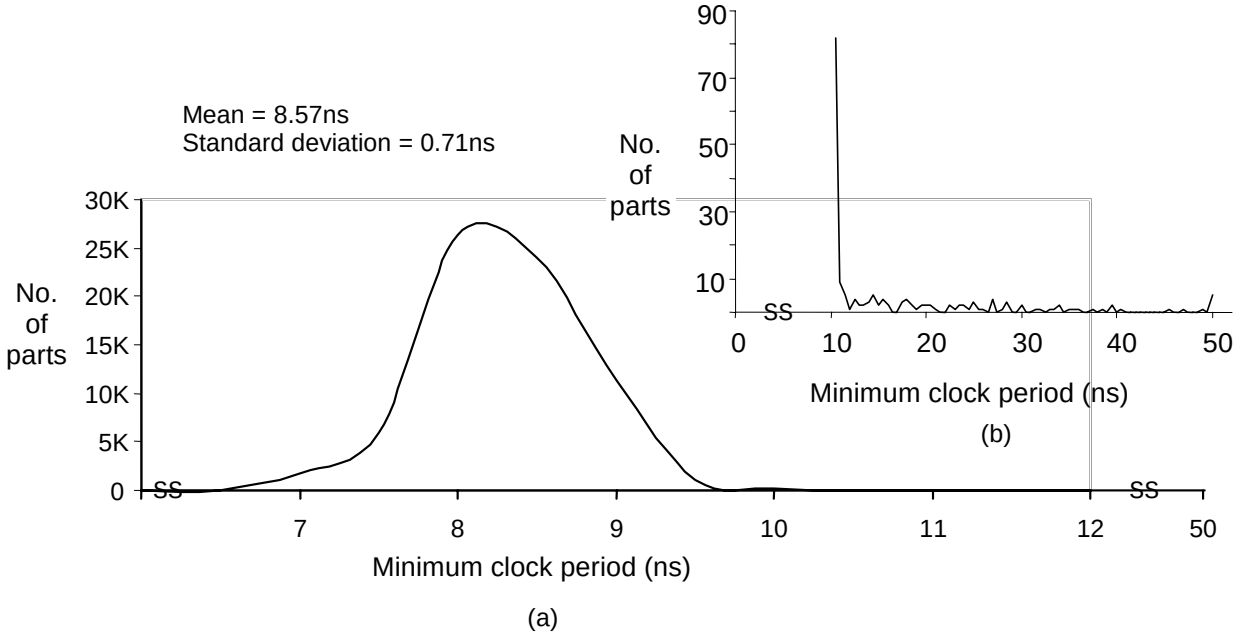
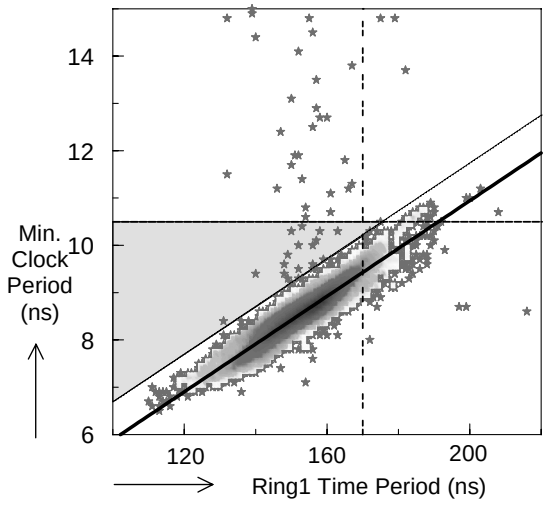
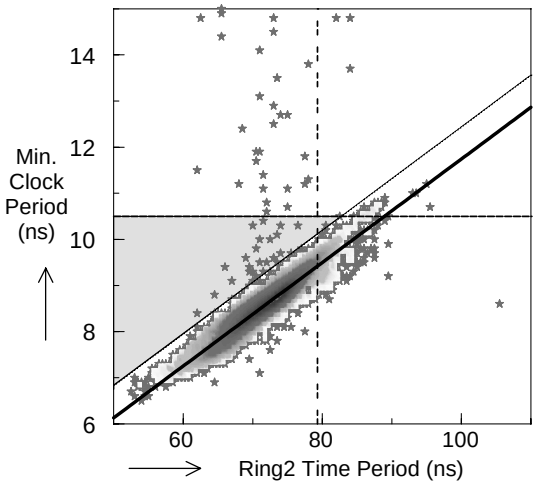


Figure 3.1. Distribution of minimum clock periods of DSP1 cores. (a) Distribution of majority of the DSP1 cores from 6ns – 10ns. (b) Tail part of the distribution in Fig. 3.1a from 10.5ns – 50ns



(a)

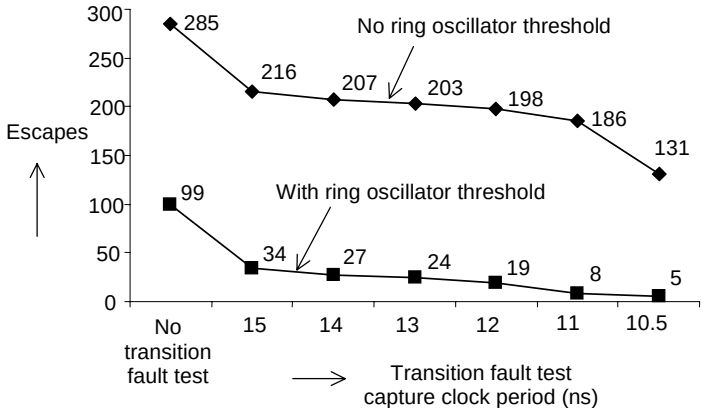


(b)

Figure 3.2. Plots of minimum clock periods of DSP1 cores vs. Ring1 & Ring2 time periods

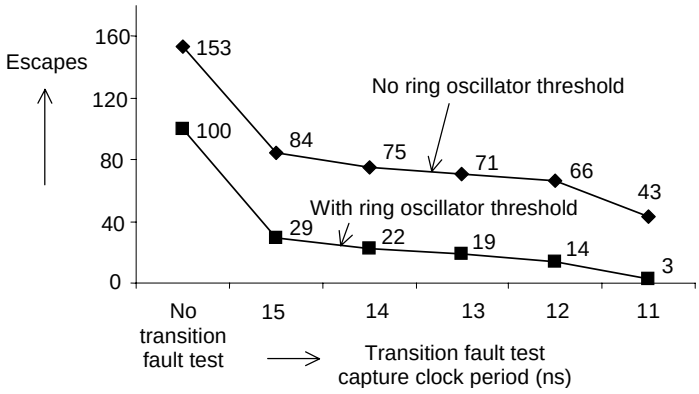
There are several trade-offs associated with the choice the two parameters. If *T* is large, there may be many escapes. On the other hand, if *T* is large the timing constraints on the clock signals (and the scan enable signal if launch-on-last-shift transition fault testing is performed) are much relaxed. From transition fault ATPG standpoint, a large value of *T* indirectly translates to fewer constraints on sensitizing transitions along longest paths or accounting for "small" delay defects in designs with time borrowing. More details on these issues can be found in [Kim 03]. Depending on the choice of *R*, there may be more test escapes or more "good" parts that may be rejected.

Tests using Ring1 and Ring2 yield similar number of test escapes with the thresholds shown in Fig. 4.1. Hence, test escapes for Ring1 and Ring2 aren't plotted separately.



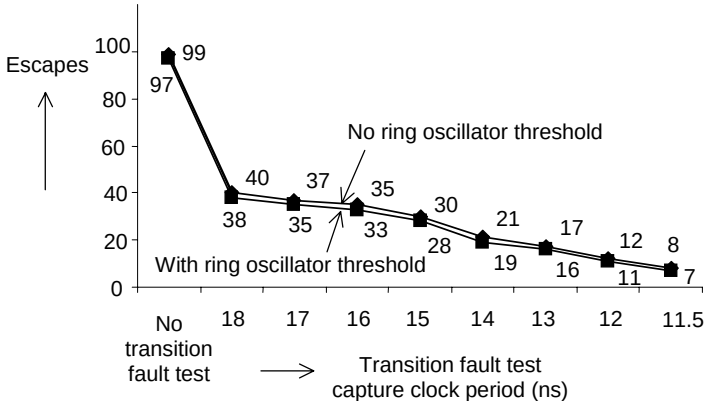
Ring1 threshold = 170ns, Ring2 threshold = 80ns

(a)



Ring1 threshold = 180ns, Ring2 threshold = 80ns

(b)



Ring1 threshold = 180ns, Ring2 threshold = 80ns

(c)

Figure 4.1. Escapes vs. Transition fault test capture clock period. Total parts > 70,000. Application clock period = (a) 10ns; (b) 10.5ns; (c) 11ns.

A part rejected by the ring oscillator threshold based test but having a minimum clock period less than or equal to the application clock period belongs to the category of *yield loss*. Table 4.1 reports the yield loss corresponding to the plots shown in Fig. 4.1. For example, when the application clock period is 10ns, from Fig. 4.1a there are 19 escapes when the capture clock period of transition fault test is 12ns and Ring2 threshold is 80ns. The yield loss (from Table 4.1) is 846 parts. When no ring oscillator test is applied, the yield loss is 0.

Note that, in this case the yield loss is a function of the application clock period and the ring oscillator threshold only – it doesn't depend on the transition fault capture clock period (because we assume that the capture clock period is greater than or equal to the application clock period).

Table 4.1. Yield loss for ring oscillator tests corresponding to escapes in Fig. 4.1.

Application clock period (ns)	Ring1		Ring2	
	Threshold (ns)	Yield loss	Threshold (ns)	Yield loss
10	170	1,334	80	846
10.5	180	132	80	973
11	180	183	80	1,024

From Fig. 4.1 and Table 4.1, the following observations can be made. First, with application clock period of 10ns or 10.5ns, the ring oscillators help significantly in reducing the number of test escapes when transition fault testing is performed at a slow speed. From Fig. 4.1a, if no ring oscillator test is performed, even a 5% slower transition fault test (10.5ns capture clock period with application clock period of 10ns) results in 131 escapes. On the other hand, when only ring oscillator test is performed, the number of escapes is 99. For typical quality levels (50-500 DPM), slow-speed transition fault testing or a ring oscillator test alone is not enough to reduce the number of test escapes. However, the combination of both can lead to an effective test strategy enabling slow-speed delay testing.

There isn't enough benefit obtained from ring oscillators when the application clock period is 11ns or more. We suspect this is mainly because most delay defects that cause the minimum clock periods of the DSP1 cores to be greater than 11ns are due to spot defects. The other explanation could be that the ring oscillators are insensitive to process variations that cause the minimum clock periods of DSP1 cores to exceed 11ns. It is interesting to note that the majority of slow parts (with minimum clock period > application clock period of 11ns) passing ring oscillator tests have their minimum clock periods greater than 150% of 11ns.

5. Further Analysis and Open Issues

The ELF18 test chip contains another ring oscillator called Ring3 that is distinct from Ring1 and Ring2 in two ways. First, Ring3 is routed within a completely different block – the analog block – that is different from the DSP1 core. The other difference is that there is no special type of load that's designed in Ring3 unlike the loads corresponding to global and local routing that are designed for Ring1 and Ring2.

We analyzed the effectiveness of Ring3 in detecting slow parts similar to the analysis in Sec. 4 using Ring1 and Ring2. Our objective was to find out whether we see any significant difference in escapes or yield loss when we use Ring3 instead of Ring1 or Ring2. We found that for application clock period of 10ns, using Ring3 results in either significantly more escapes or significantly more yield loss compared to Ring1 or Ring2. The results are summarized in Fig. 5.1.

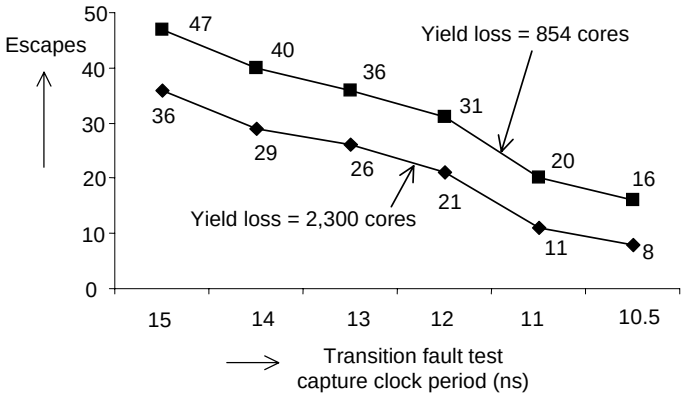


Figure 5.1. Escapes vs. Transition fault test capture clock period & yield loss for various Ring3 thresholds. Application clock period = 10ns.

One open issue we are currently working on relates to the outliers in Fig. 3.2. These are the DSP1 cores whose actual minimum clock periods are greater than the values predicted from the regression equations relating minimum clock periods and ring oscillator time periods. For example, consider the DSP1 cores in Fig. 3.2a that are clear outliers but have minimum clock periods less than 10ns. Some of these cores that have their corresponding Ring1 time periods greater than or equal to 170ns will be rejected by our testing technique – these cores will be included in the yield loss category. There will still be a few outlier cores that will not be rejected and not treated as test escapes because their minimum clock periods are less than 10ns and their ring oscillator time periods are less than 170ns. These cores probably have spot defects as a result of which their minimum clock periods didn't correlate with the ring oscillator time periods. There are three possibilities regarding what may happen if these cores are shipped to customers. These cores may be transition fault test escapes because transition fault test patterns don't guarantee propagation of transitions along longest paths. Alternatively, these cores may be

candidates for early-life failures if the spot defect worsens in the field. Finally, these cores may just work fine in the field.

These outliers cannot be detected by a traditional transition fault test that uses a fixed pre-determined capture clock period to test all parts. One way to detect these outliers is to determine the capture clock period for each part based on its ring oscillator time period. The relationship between the minimum clock period and the ring oscillator time period derived from the regression analysis (discussed in Sec. 3) can be used for this purpose.

For Ring1, the following relationship can be used:

$$\text{Min. Clock Period} \mid 0.6 \; 2 \; 0.05251 \Delta \text{Ring1 Time Period}.$$

Based on the Ring1 time period of a part, the expected minimum clock period of that part is obtained from the above relationship. Next, a guard-band, based on the standard deviation of the regression analysis, must be added to the expected minimum clock period to determine the capture time period for transition fault testing of that part.

6. Related Work

Use of process monitor structures such as ring oscillators to understand process characteristics and to obtain indications of the performance of a part isn't something new and is widely used [Milor 97, Bassi 03]. In [Jain 84], odd inverter chains were reported to be used to identify global shifts in propagation delay. The current paper is unique because it uses process monitor structures for delay defect screening purposes in conjunction with transition fault testing and presents actual data to demonstrate the effectiveness and understand the trade-offs. Some statistical post-processing techniques for outlier identification [Madge 02] can be used to further improve the techniques discussed in this paper.

7. Conclusions

The data presented clearly shows the effectiveness of process monitor structures such as ring oscillators to identify slow parts while performing transition fault testing slower than the rated speed. However, there is possible yield loss associated with such an approach. The data also shows that by using these ring oscillators we can't completely eliminate transition fault testing. If process monitor structures aren't used, then the capture clock period of transition fault testing must be very tight compared to the application clock period time. The benefits of using process monitors can reduce significantly when the process is very well-controlled such that a vast majority (e.g., > 99.99%) of parts have delays less than the application clock period. Of course, these conclusions must be validated for other technologies (65nm, 90nm and 130nm) and other designs.

Future efforts should focus on design of more sophisticated process monitor structures for delay defect detection, development of theoretical models and the use of process monitors for speed binning. An interesting extension of this work will be to examine whether process monitor s can be used to eliminate path delay fault testing.

8. References

We acknowledge the contributions of Mr. Kenneth A. Brand of the Stanford Center for Reliable Computing.

9. References

[Bassi 03] Bassi, A., A. Veggetti, L. Croce and A. Bogliolo, "Measuring the Effects of Process Variations on Circuit Performance by means of Digitally-Controllable Ring Oscillators," *Proc. Intl. Conf. Microelectronic Test Structures*, pp. 214 –217, 2003.

[Jain 84] Jain, R.K., "Picosecond Optical Techniques Offer a New Dimension for Microelectronics Test," *Test and Measurement World*, pp. 40-53, June 1984.

[Kievits 98] Kievits, P., E. Lambers, C. Moerman and R. Woudsma, "R.E.A.L. DSP Technology for Telecom Broadband Processing," *Proc. Intl. Conf. Signal Processing Applications and Technology*, 1998.

[Kim 03] Kim, K.S., S. Mitra and P.G. Ryan, "Delay Defect Characteristics and Testing Strategies," *IEEE Design and Test of Computers*, Vol. 20, No. 5, pp. 8-16, 2003.

[Madge 02] Madge, R., M. Rehani, K. Cota and W.R. Daasch, "Statistical Post-Processing at Wafer Sort – An Alternative to Burn-in and a Manufacturable Solution to Test Limit Setting for Sub-Micron Technologies," *Proc. IEEE VLSI Test Symp*., pp. 69-74, 2002.

[McCluskey 00] McCluskey, E.J., and C.W. Tseng, "Stuck-at Faults vs. Actual Defects," *Proc. Intl. Test Conf.*, pp. 336-343, 2000.

[Milor 97] Milor, L, L. Yu and B. Liu, "Logic Product Speed Evaluation and Forecasting during the Early Phases of Process Technology Development using Ring Oscillator Data," *Proc. Intl. Workshop Statistical Metrology*, pp. 20 –23, 1997.

[Park 88] Park, E.S., M.R. Mercer, and T.W. Williams, "Statistical Delay Fault Coverage and Defect Level for Delay Faults," *Proc. Int'l Test Conf.*, pp. 492-499, 1988.

[Qiu 03] Qiu, W., and D.M.H. Walker, "An Efficient Algorithm for Finding the K Longest Testable Paths Through Each Gate in a Combinational Circuit," *Proc. Intl. Test Conf.*, pp. 592-601, 2003.

[Rumsey 03] Rumsey, D., *Statistics for Dummies*, Wiley Publishing Inc., 2003.

[Saxena 02] Saxena, J., *et al.*, "Scan-based Transition Fault Testing Implementation and Low-Cost Test Challenges," *Proc. Intl. Test Conf.*, pp. 1120-1129, 2002.

[Sharma 02] Sharma, M., and J.H. Patel, "Finding a Small Set of Longest Testable Paths that Cover Every Gate," *Proc. Intl. Test Conf.*, pp. 974-982, 2002.

[Tseng 01] Tseng, C.W., and E.J. McCluskey, "Multiple-Output Propagation Transition Fault Test," *Proc. Intl. Test Conf.*, pp. 358-366, 2001.

[Yan 03] Yan, H., and A.D. Singh, "Experiments in Detecting Delay Faults using Multiple Higher

Frequency Clocks and Results from Neighboring Die," *Proc. Intl. Test Conf.*, pp. 105-111, 2003.

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 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 and up to 8 metal layers, 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. Logic resources are the hardware within the basic building blocks—primarily *logic blocks* and *input/output blocks*. Logic blocks contain 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. Input/output blocks 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 fault-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_d	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	0.43	0.28	0.13	0.65	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	0.13	1.93	0.40	0.27	0.14	0.68	4.9
4	1.90	2.07	0.17	2.35	0.45	0.28	0.12	0.62	5.2
5	1.60	1.74	0.14	2.01	0.41	0.27	0.13	0.66	5.1

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

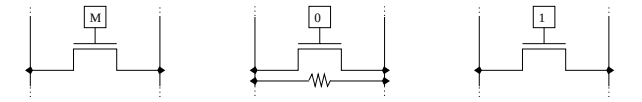
An application-dependent FPGA is one whose configuration remains unchanged throughout its lifetime. Only the resources enabling the customer's design to function—the used resources—are guaranteed to be fault-free: unused resources may be faulty. Currently Xilinx offers application-dependent FPGAs through 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 detects all used-unused bridging faults. All used LUTs are configured to implement the function $F = 0$ and all used bistables are configured with 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 a 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} magnitude: 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_d, 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_d and d_s.

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

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.

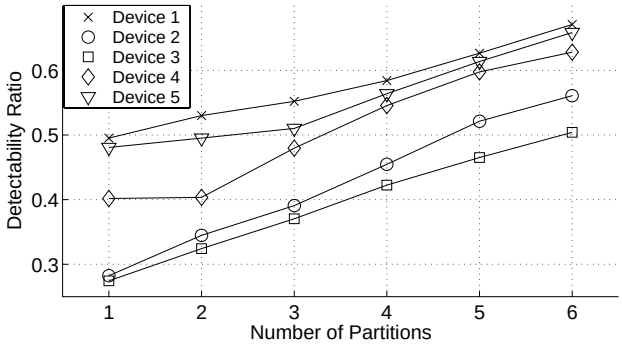
Consider a single test configuration whose used interconnects compose the set *S*. This configuration, that tests for bridging faults between any interconnect in *S* and any interconnect not in *S*, can be divided, or *partitioned*, into *N* smaller configurations, each testing only a subset of the interconnects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

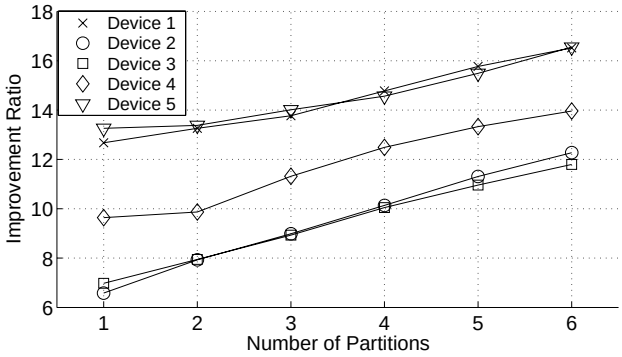
5.2. Experimental Results

A fault is injected in a very dense 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_d 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_d



(b) Improvement Ratio, i

Figure 3: Experimental Results for Partitioned FPGA

6. Fault Location

6.1. Overview

Iterative 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 ... P_N = partition(S,N)
   foreach P_i
      determine I_DDQ_sig_i
   foreach P_i
      if I_DDQ_sig_i < max(I_DDQ_sig_1,...,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 for even the largest 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.

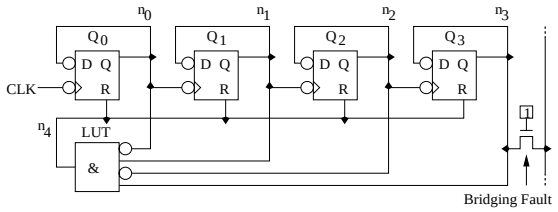


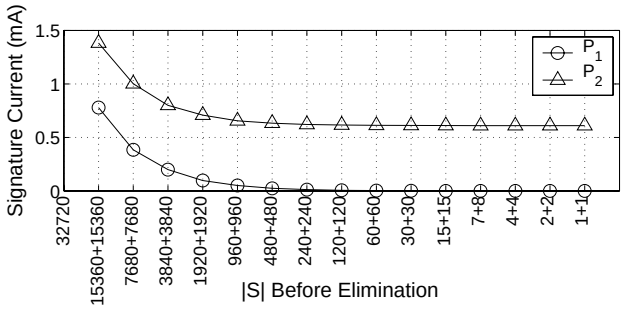
Figure 5: Decade Counter Sample Circuit

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.

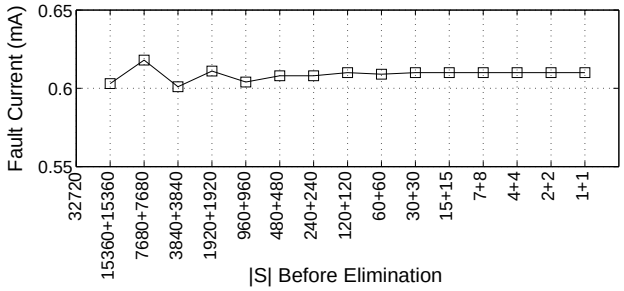
6.3.2. Large FPGA

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 sig-

nature 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 fault (roughly 1 kΩ). Partitioning further increases the improvement to at least 12-fold. Approximately five configurations test for all adjacent 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. St. 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.