[9] Tracy Larrabee. Efficient generation of test patterns using Boolean Difference. In *Proceedings of International Test Conference*, pages 795–801. IEEE, 1989.

[10] Kuen-Jong Lee and Melvin A Breuer. Constraints for using IDDQ testing to detect CMOS bridging faults. In *Proceedings of the IEEE VLSI Test Symposium*, pages 303–308, 1991.

[11] W. Maly, P.K. Nag, and P. Nigh. Testing oriented analysis of CMOS ICs with opens. In *Proceedings of International Conference on Computer-Aided Design*, pages 344–347. IEEE, 1988.

[12] W. Maly and P. Nigh. Built-in current testing - feasibility study. In *Proceedings of International Conference on Computer-Aided Design*, pages 340–343. IEEE, 1988.

[13] W. Mao, R.K. Gulati, D.K. Goel, and M.D. Ciletti. QUIETEST: A quiescent current testing methodology for detectiong leakaage faults. In *Proceedings of International Conference on Computer-Aided Design*, pages 280–283. IEEE, 1990.

[14] E.J. McCluskey and F. Buelow. IC quality and test transparancy. In *Proceedings of International Test Conference*, pages 295–301. IEEE, 1988.

[15] A.K. Pramanick and S.M. Reddy. On the detection of delay faults. In *Proceedings of International Test Conference*, pages 845–856. IEEE, 1988.

[16] J.P. Shen, W. Maly, and F.J. Ferguson. Inductive fault analysis of MOS integrated circuits. *IEEE Design and Test of Computers*, 2(6):13–26, December 1985.

[17] J.M. Soden, R.K. Treece, M.R. Taylor, and C.F. Hawkins. CMOS IC stuck-open fault electrical effects and design considerations. In *Proceedings of International Test Conference*, pages 423–430. IEEE, 1989.

[18] R.L. Wadsack. Fault modeling and logic simulation of CMOS and MOS integrated circuits. *Bell System Technical Journal*, 57(5):1449–1474, May-June 1978.

[19] H. Walker and S.W. Director. VLASIC: A yield simulator for integrated circuits. *Proceedings of International Conference on Computer-Aided Design*, November 1985.

Modification of an existing SSF ATPG system so that it generates tests for I_{DDQ} faults requires only that the fault propagation requirement be removed and the fault simulator be similarly modified. The resulting system generates tests using less time or memory and produces fewer vectors with fewer untestable defects.

Acknowledgements

The authors thank Heather Trumbower and Martin Taylor for their work on our software and Leendert Huisman for providing SSF fault simulation data.

References

[1] John M. Acken. *Deriving Accurate Fault Models.* PhD thesis, Stanford University, Department of Electrical Engineering, September 1988.

[2] F. Brglez and H. Fujiwara. A neutral netlist of 10 combinational benchmark circuits and a target translator in fortran. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, 1985.

[3] E. Eichelberger and T. Williams. A logic design structure for LSI testability. In *Proceedings of the 14th Design Automation Conference.* IEEE, 1977.

[4] F. Joel Ferguson and John P. Shen. A CMOS fault extractor for inductive fault analysis. *IEEE Transactions on Computer-Aided Design*, 7(11):1181–1194, November 1988.

[5] C.F. Hawkins and J.M. Soden. Reliability and electrical properties of gate oxide shorts in CMOS ICs. In *Proceedings of International Test Conference*, pages 443–451. IEEE, 1986.

[6] Charles F Hawkins, Jerry M Soden, Ron R Fritzemeier, and Luther K Horning. Quiescent power supply current measurement for CMOS IC defect detection. *IEEE Transactions On Industrial Electronics*, May 1989.

[7] Dennis V. Heinbuch. *CMOS3 Cell Library.* Addison-Wesley Publishing Company, 1988.

[8] Leendert M. Huisman. The reliability of approximate testability measures. *IEEE Design and Test of Computers*, 5(6):57–67, December 1988.

Circuit Name	C880	C1355	C1908	C2670	C3540	C5315	C6288	C7552
number of vectors	36	88	110	50	71	58	32	95
number of faults	1028	1532	1566	2501	3274	5161	7216	6994
untestable faults	0	0	2	17	51	2	35	20
undetected faults	0	0	0	9	1	0	0	0
% bridge faults covered	100	100	100	100	100	100	99.9	100
processing time (secs)	2.0	20.2	34.8	125.8	25.4	23.8	11.5	90.8

Table 4: ATPG for I_{DDQ} stuck-on faults

Circuit Name	C880	C1355	C1908	C2670	C3540	C5315	C6288	C7552
number of vectors	21	53	38	25	39	36	29	46
number of faults	1900	1630	4385	5950	8330	11519	12064	17545
untestable faults	17	28	202	118	423	256	44	332
undetected faults	0	0	0	14	0	2	0	9
processing time (secs)	3.4	13.7	51.7	352.4	122.7	95.8	18.7	616.1

Table 5: ATPG for I_{DDQ} bridge faults

likely to go undetected. In order to ensure high quality levels, these fault types should be targeted by the test generation procedures.

With I_{DDQ} testing, we can detect all non-redundant bridge and transistor stuck-on defects in the circuit. This may not be possible with logical fault testing or delay fault testing because the fault may not change the logical function of the circuit, and there may be no robust delay fault test for the circuit. Since these defects may cause a loss of reliability or cause the circuit to not meet performance or power consumption specifications, they should be detected. Experimental evidence suggests that most signal-line breaks that are not detected as stuck-at faults can be detected as I_{DDQ} transistor stuck-on faults.

If a defect is detectable as an I_{DDQ} fault, the requirements for detecting that defect are a subset of the requirements for detecting that defect as a logical fault or a delay fault. That is, any vector that detects a defect that causes excessive I_{DDQ} as a logical fault or a delay fault can also be used to detect it as an I_{DDQ} fault. Since error propagation is not necessary, generating an I_{DDQ} test vector is usually much less expensive than generating a logical or delay fault test for the same defect.

thousands of clock cycles may be needed to fill the scan-path with a single test vector before the application of the system clock for the test[3]. This means that an I_{DDQ} measurement, and hence a reduced clock rate, is needed only once every time the scan-path is filled. For this case less than 0.1% of the clocks are needed for I_{DDQ} testing. Recent experiments in test pattern generation for commercial ICs showed that the number of I_{DDQ} tests was less than 1% of the number of functional tests for the circuit and increased the testing time by less than 200 milliseconds[13].

The time that is needed for the power supply current to reach quiescent levels increases with the size of the circuit, so another approach to help reduce the test application time is to internally partition large circuits into smaller ones. Each low-power subcircuit in the integrated circuit would have a Built-In Current (BIC) sensor[12]. The BIC sensor can be fabricated in with little area and has very high sensitivity to current.

There are fewer untestable I_{DDQ} stuck-on fault classes than untestable SSF fault classes for each circuit. Since the fault equivalence collapsing techniques are more powerful for SSA faults, there are on average more SSA faults in a SSA fault equivalence class than there are I_{DDQ} faults in an I_{DDQ} fault equivalence class. This was expected since some SSF faults are untestable due to an inability to propagate the error to a primary output and the associated I_{DDQ} fault is not.

Circuit Name	C880	C1355	C1908	C2670	C3540	C5315	C6288	C7552
number of vectors	72	92	128	149	202	154	45	245
number of faults	765	1444	1740	2626	3150	4909	7619	7194
untestable faults	0	8	9	106	129	59	34	131
undetected faults	0	0	0	11	0	0	0	0
processing time (secs)	37.3	22.5	71.1	361.9	269.7	74.4	147.2	739.3

Table 3: SSF ATPG

6 Conclusions

The primary purpose of ATPG is to detect enough local defects to ensure a high quality level in the integrated circuits that have passed the test. The vast majority of local defects cause bridge, signal-line break, and transistor stuck-on faults. Those local defects that cause logical faults may not be detected by a given SSF test set, and those that do not cause logical faults are even more

5 Automatic Test Pattern Generation for I_{DDQ} Faults

We generated tests for transistor stuck-on I_{DDQ} faults and bridging I_{DDQ} faults using the Nemesis SSF automatic test pattern generation (ATPG) system[9]. The fault equivalence collapsing routines, the parallel pattern and single pattern simulators, and test pattern generation portions of Nemesis were modified in order to generate tests for the new faults. The fault simulator was changed so that transistor stuck-on faults and bridging faults are flagged as detected when they are stimulated at the fault site, and the portion of the test pattern generator that extracts formulas was also modified to exclude the necessity of error propagation. Because we did not have a set of realistic bridging faults to test, we generated tests for each node bridged to the next five nodes in the TDL wirelist. A test vector compaction phase, consisting of a reverse order fault simulation was added to both the SSF and the I_{DDQ} versions of the ATPG system to provide a more realistic ratio of the number of required tests to detect all detectable SSFs to the number required to detect all detectable I_{DDQ} faults.

Tables 3, 4, and 5, show the statistics for ATPG of single stuck-at faults, transistor stuck-on I_{DDQ} faults, and bridge I_{DDQ} faults, respectively. The times given are for a Sparcstation 1[4].

For each circuit, generation of I_{DDQ} vectors is faster than generation of SSF vectors (by factors of between 1.1 and 20), and fewer vectors were generated to detect all I_{DDQ} faults than were generated to detect all SSF faults. The range of ratios between the number of SSF test vectors to the number of I_{DDQ} test vectors is from about 1-to-1.1 to 1-to-3. The ATPG generated vectors for I_{DDQ} stuck-ons detected most of the bridges that we considered: at least 99.9% of all I_{DDQ} bridges for each of the circuits were detected by the stuck-on test sets. This means that the total number of I_{DDQ} vectors required is very close to the number of I_{DDQ} stuck-on vectors.

Our conclusion is that although few vectors need to be monitored for increased I_{DDQ} in order to ensure that a circuit under test has no transistor stuck-ons and no bridges, the time to test these circuits for I_{DDQ} faults may be much greater than the time to apply SSF tests. However, most low-power integrated circuits are much more amenable to I_{DDQ} testing than the ISCAS benchmark circuits, which were created to measure the performance of SSF ATPG systems. A fault in a sequential circuit may require scores of vectors to propagate an error to a circuit output. This adversely impacts both test generation and test application time for stuck-at faults but not for I_{DDQ} faults. If the sequential circuit uses a full or partial scan-path to make it more easily testable,

[4]We ran the system without using the non-local clauses (learned implications) for easier comparison.

Table 1 shows the expected I_{DDQ} fault coverage of the ISCAS benchmark circuits when up to 500,000 random vectors are applied. Similarly, Table 2 shows the SSF fault coverage of the ISCAS benchmark circuits when up to 16,000,000 random vectors are simulated. The data for Table 2 was generated using single vector detection probabilities presented by Leendert Huisman of IBM[8].

Circuit Name	C880	C1355	C1908	C2670	C3540	C5315	C6288	C7552
number of stuck-ons	1458	2128	2996	4152	5878	8772	9600	12288
undetected stuck-ons	0	0	2	41	54	3	52	326
coverage of stuck-ons	100	100	99.9	99	99	99.9	99.5	99
number of bridges	1329	1761	2739	4278	5157	7455	7344	11157
undetected bridges	1	1	4	5	9	0	4	6
coverage of bridges	99.93	99.94	99.85	99.88	99.83	100	99.95	99.95

Table 1: I_{DDQ} bridge and stuck-on detection for 500,000 random vectors

The last line of Table 2 shows the ratio of the number of stuck-at vectors required to achieve the coverage shown and the number of stuck-on vectors required to achieve the same coverage. As expected, a target coverage of I_{DDQ} stuck-ons can be achieved with many fewer vectors than the same coverage for SSFs. The sole exception is circuit C6288. Note that the circuits with the most difficult-to-test faults under random SSF test offer the greatest improvement in testability under random I_{DDQ} test.

Circuit Name	C880	C1355	C1908	C2670	C3540	C5315	C6288	C7552
number of SSFs	942	1574	1879	2746	3425	5350	7744	7550
undetected faults	0	8	9	121	134	59	34	198
number of faults with $P(j) \leq 0.1\%$	30	8	87	432	175	63	34	641
coverage	100	99.5	99.5	95.6	96.1	98.9	99.6	97.4
vector ratio for same coverage	4	3	5	17,000	128	68	1	8,000

Table 2: SSF detection after 16,000,000 random vectors

The first disadvantage limits the types of circuits for which I_{DDQ} testing is effective. However, most CMOS ICs meet this requirement without changes. The second disadvantage may require that the IC remain on the tester for longer than is economically justifiable. In the next two sections we show that the number of tests that are needed to detect I_{DDQ} faults is less than the number to detect SSA faults, thus reducing the test time.

It appears that high coverage of bridge, break and transistor stuck-on faults requires that SSF testing be supplemented with either I_{DDQ} or delay fault testing. We next compare the costs of I_{DDQ} testing versus the cost of SSF testing. I_{DDQ} tests are considered for only transistor stuck-ons and bridges between signal lines because any I_{DDQ} test set that detects all transistor stuck-ons also detects all bridges between a signal node and power or ground, all gate oxide shorts to the channel, source, or drain, and all signal-line breaks that are not detected by SSF test sets. The costs of detecting signal line bridges and transistor stuck-ons are therefore the costs of detecting all I_{DDQ} faults discussed in the previous section.

4 Random Test Pattern Generation for I_{DDQ} Faults

We simulated the eight largest ISCAS combinational test generation benchmark circuits with up to 1/2 million pseudo-random vectors—accumulating the probability of detection by a single random vector for each fault[2]. We counted the number of times that each fault is detected using the criteria described in the previous section: transistor stuck-ons are considered detected when the gate is stimulated, and bridges are considered detected when the logic values for the two bridged lines differ. Because the total number of potential bridges in each circuit is very large, we considered a random subset for each node. In practice the circuit layout would be considered, and adjacent nodes would be tested for bridges. The expected fault coverage for a given number of vectors for each circuit was calculated using the probability of detection by a single random vector for each fault.

Given $P(j)$, the probability of a single random vector detecting fault j, the probability of k random vectors not detecting fault j is $(1 - P(j))^k$. The sum of $(1 - P(j))^k$ over all faults divided by the number of faults is the average probability for the fault to remain undetected after k vectors. This means that $E(k)$, the expected fault coverage after k vectors, is

$$E(k) = 1 - \left[\frac{\sum_{j=1}^{\#faults} (1 - P(j))^k}{\#faults} \right].$$

of nodes. Fortunately, we can use information available about the physical locality of structures in the circuit to produce a much reduced set of bridges to be considered[19,4].

3 Comparing Testing Methods

Detecting all non-redundant bridges, breaks and transistor stuck-ons in static CMOS circuits probably requires that either delay or I_{DDQ} tests be used: each technique has its disadvantages. The disadvantages of delay fault test generation are:

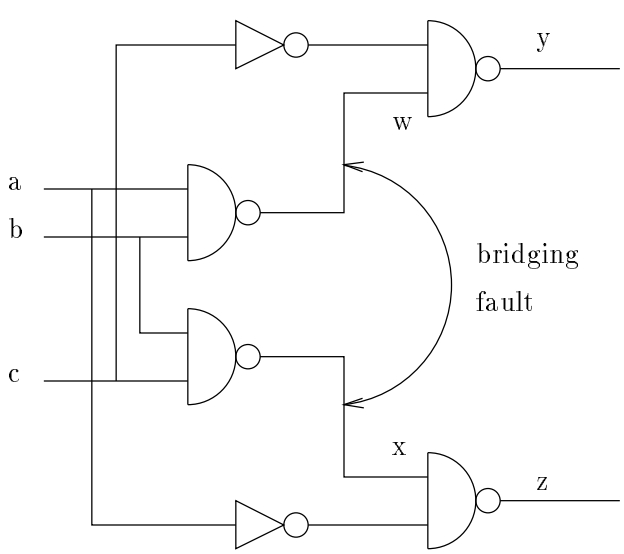
Delay testing is expensive. Generating delay fault tests is more difficult than generating SSF tests[15]. This is because standard delay tests require a sequence of inputs, and the sequence must produce a glitch-free path to propagate the delay error. In addition, the cost to generate a delay fault test for bridges may be greater than for standard delay faults (because of the constraints mentioned in Section 2.3). Also, there are more potential bridge, break, and transistor stuck-on faults than there are gate input delay faults.

Delay testing may not detect all defects. Detecting all defects as delay faults may be impossible. The constraints for delay fault detection for bridge, break, and transistor stuck-on defects make it difficult to generate robust tests (those that are not potentially invalidated by glitches[15]). Non-robust tests may not detect the delay fault. Also the size of the delay fault may be too small to be detected given the system clock and the normal delay of the paths connected to the gate.

A major advantage to detecting bridge, break, and transistor stuck-on defects as I_{DDQ} faults is that test generation is much less expensive than for SSFs and fewer test vectors are needed than are needed for SSF test sets. How much easier is shown in Section 5. But before we consider the advantages, we list the disadvantages of I_{DDQ} testing:

The circuit must be designed to have low I_{DDQ}. This means that multiple outputs cannot be simultaneously driving internal busses, and the I/O pads' power supply may need to be separated from the logic's power supply.

Measurement must occur after transients die. I_{DDQ} can not be measured until after the switching transients have died out. This reduces the test application rate.



a	b	c	w	x	y	z
0	0	0	1	1	0	0
0	0	1	1	1	1	0
0	1	0	1	1	0	0
0	1	1	D	0	1	1
1	0	0	1	1	0	1
1	0	1	1	1	1	1
1	1	0	0	D	1	1
1	1	1	0	0	1	1

D is a 1 in the good circuit
and a 0 in the faulty circuit

Figure 2: A Wired-AND Bridging Fault that is not a Logical Fault.

as a logic fault, even if the circuit is not redundant. Consider the bridging fault in Figure 2. The table to the right of the figure shows the values on w, x, y, and z given the values on a, b, and c, and assuming a wired-AND bridging fault between nodes w and x. This bridging fault causes no logical fault even though there is no redundancy. In this case, detecting the bridge requires a parametric test.

The detection of delay faults is considerably more difficult than SSFs [15]. In addition to the already considerable requirements in standard delay fault tests, the bridged nodes must have different logic values during the second (non-initializing) input.

The bridge fault shown in Figure 2 was simulated with SPICE to find the magnitude of the delay and I_{DDQ} faults. The NAND and inverter cells from the CMOS3 standard cell library [7] scaled to 1.2 micron transistor length were used for the simulation. There was 0.9 nanoseconds additional delay for the transition (abc)=010 $\rightarrow$ 011 to be seen at x and 2.6 milliamps additional current for when abc = 011. While the normal current through a large CMOS gate array can be orders of magnitude less than 2.6 milliamps, the delay to be measured through a block of logic is typically longer than 0.9 nanoseconds: the existence of the bridging defect is more definitively demonstrated using I_{DDQ} testing.

The number of possible bridges in a circuit is often prohibitively large to consider for test generation or for fault simulation: theoretically we would have to consider on the order of n^2 pairs

have one conducting and one not conducting transistor[11]. This would behave like a SSF[3]. The same research suggests that breaks that cause the gates of only nFET or only pFET gates to float in a gate, as in Figure 1-a, cause those transistors to be stuck-on. These are detected by transistor stuck-on I_{DDQ} tests.

In the remainder of this paper, we limit our discussion of break defects to those on signal lines because signal lines are typically much longer and more likely to be bridged or broken than nodes internal to a logic gate. We do not consider bridges or breaks on the power supply nodes or clock nodes because they are generally detected by SSF test sets. As previously mentioned, we will assume that single floating transistors are conducting[11].

To summarize, most signal-line breaks that do not cause SSFs can be detected as I_{DDQ} faults using transistor stuck-on tests.

2.3 Bridges

A bridge can be detected only if an input sequence is applied in which the two bridged nodes have different logic values in the non-faulty circuit. Most bridges can be detected as logical faults, delay faults or I_{DDQ} faults. For a bridge to be detected as a logical fault the two nodes must take on different values, one of the nodes must overpower the other's logic value, and a path of sensitized gates must lead from the overpowered node to a primary output where the error can be observed. If the path of sensitized gates leading to the primary output begins at the bridged node that has the greater drive, it may still be detected as a delay fault during a test sequence if the logic value changes on the overpowering node and if the overpowered node's logic value differs from the final value of the more powerful node. In addition, there can be no glitches that invalidate the test on the sensitized path. For a bridge to be detected as an I_{DDQ} fault, the two bridged nodes must be driven to different logic values so that there is low impedance path between V_{DD} and V_{SS} leading to an elevation in I_{DDQ}. Note that the sensitization requirements for the detection of a bridge as an I_{DDQ} fault are a subset of the requirements of detection as either a logical fault or a delay fault.

For ECL and TTL technologies it is assumed that the bridge fault behaves as a wired-OR or wired-AND. The logical behavior of a CMOS gate is often more complicated than this and is affected by transistor sizing, topology, and manufacturing process variations[17,1]. Whatever the logical function of the bridge fault, it may not be possible to meet the requirements for detection

[3]This situation can be complicated by the existence of capacitive coupling between floating gates and adjacent nodes [17]

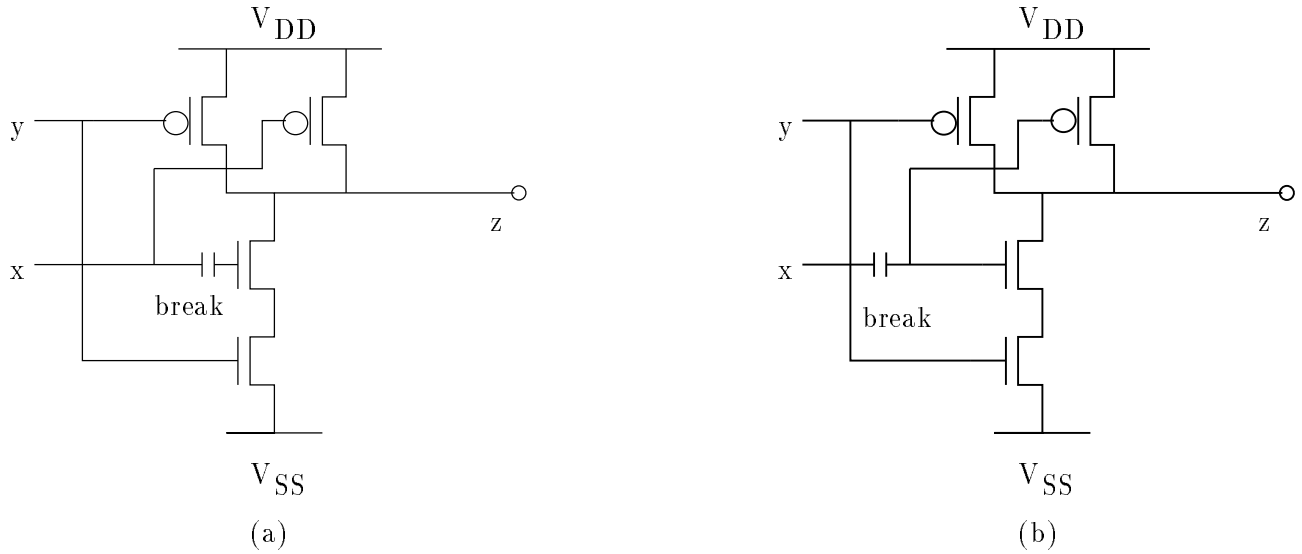


Figure 1: A break causing (a) a single nFET with a floating gate, and (b) nFETs and pFETs with floating gates.

2.2 Breaks

Breaks can be divided into two categories: *intra-gate breaks*, those involving nodes internal to the logic gates, and *signal-line breaks*, those involving signal lines. The two categories have markedly different faulty behavior.

If an intra-gate break severs all possible low impedance paths from either V_{SS} or V_{DD} to the gate's output, it causes the output of the gate to be stuck at a single logic value. If the defect breaks some, but not all, paths from either V_{SS} or V_{DD} to the gate's output, the gate may become a dynamic sequential circuit [18].

Signal-line breaks in static CMOS designs result in transistors with floating gates. There are two types of signal line breaks: those that cause floating gates in only the pFETs or only the nFETs, like that shown in Figure 1-a, and those that cause floating gates for both pFETs and nFETs of the logic gate like that shown in Figure 1-b.

The behavior of the logic gate affected by the signal-line break is determined by the state (conducting or non-conducting) of the transistors connected to a floating node. If the transistors with the floating gates are permanently non-conducting, the behavior of the circuit is equivalent to the behavior of intra-gate breaks. If the transistors with the floating gates are permanently conducting, the behavior of the gate would be the same as if it had a transistor stuck-on fault.

Recent research suggests that transistor pairs with floating gates, as in Figure 1-b, are likely to

2.1 Transistor Stuck-ons

A transistor stuck-on is a fault that can be modeled as a transistor that is always in the conducting mode. Transistor stuck-ons can be caused by several defect types: common causes are source to drain bridges and breaks in signal lines that cause a single FET in the logic gate to be floating (as shown in Figure 1-a).

SPICE simulations were performed for the 2-input NAND gate of Figure 1-a assuming that the gate of the floating transistor had a logic 1 and that the circuit's output fanned out to a single inverter. The lengths and widths of the transistors in the NAND gate and the inverter are those in the CMOS3 cell library [7] scaled to a transistor length of 1.2 microns. The transconductances and other relevant circuit parameters were those of a recent MOSIS run. A SPICE simulation assuming that the transistor is stuck-on at 5.0 volts resulted in a logical fault: the behavior was as if line x was stuck-at 1. A second SPICE simulation assuming that the stuck-on transistor was at 4.0 volts resulted in no logical fault, but it did result in an I_{DDQ} fault of 2.1 milliamps when xy=01, and a delay fault of 1.05 nanoseconds when xy changed from 00 to 01. Variations in transistor length or width ratios, in process parameters, and in defects determine the magnitude of the delay or IDDQ fault, as well as whether or not the transistor stuck-on can be modeled as a logical fault.

The detection of a logical fault requires that the test stimulate the fault (the logic values on the gate's inputs cause a logical error at the gate's output) and propagate the error to the circuit's output. The detection of an IDDQ fault requires only that the test stimulate the fault; propagation of the error is not necessary. In general, the detection of a delay fault requires that two consecutive inputs be given to cause a transition at the fault site and that the slow transition be propagated to an output of the circuit. The delay fault may not be detected if there are glitches on the propagation path or if the propagation path is too short [15].

For a static CMOS gate consisting of parallel-series nMOS and pMOS networks, any test set that detects all SSFs also detects all I_{DDQ} stuck-ons and any test set that detects all I_{DDQ} stuck-ons also detects all SSFs. This implies that any set of tests that detect all SSFs in an arbitrary circuit also detects all I_{DDQ} stuck-ons for that circuit. However, the converse is not true because of the additional requirement for fault propagation when detecting SSFs. As we show later, I_{DDQ} tests for a circuit are usually much easier to generate than SSF tests since fault propagation is unnecessary.

fault. For example, an I_{DDQ} test for a specific fault includes first applying appropriate inputs to the circuit to cause excessive I_{DDQ}, and then measuring I_{DDQ}.)

Defect-simulation experiments show that the vast majority of all local defects in MOS technologies cause bridges, breaks, and transistor stuck-ons[16,4]. Some bridges, breaks, and transistor stuck-ons are detectable as logical faults, although they may not be detected by specific complete SSF test sets, but others are only detectable as parametric faults. Defects can be undetectable as logical faults and still cause an IC to be unacceptable for shipment because of performance degradation or loss of reliability. To detect these non-logical faults requires that non-logical testing be employed. In a recent experiment, the addition of I_{DDQ} monitoring to logical testing increased the capture of defective chips by 1.6 and 2.8 times[6]. In this paper we compare testing strategies that detect these common defects.

We limit our discussion to testing circuits composed solely of fully-complementary static CMOS gates consisting of a network of pFETs' between V_{DD} and the output and a network of nFETs between the output and V_{SS}. The nFET network's switching function is the complement of the pFET network's switching function: for each input combination there is either a low impedance connection between V_{DD} and the output or between V_{SS} and the output. We restrict our domain to these circuits in order to simplify the presentation of our results[1].

2 Defect Detection

In this section we present bridge, break, and transistor stuck-on faults and show how they may affect the behavior of the circuit. For the standard cells mentioned previously, these three fault types accounted for over 99% of all faults likely to be caused by local defects[4]. Approximately 48% were bridges, 42% breaks, and 10% were transistor stuck-ons[2]. We show later that some of each of these three circuit-fault types may be detected as logical faults, some as delay faults, some as I_{DDQ} faults.

[1] There are low-power circuits that are not composed of static gates. Examples include low-power PLAs and domino CMOS logic. The parts of a low power circuit consisting of static gates can be tested for I_{DDQ}, although different techniques may be needed to test the non-static gates[10]. I_{DDQ} techniques can be used to test high-power circuits if the high-power portions of the circuit are partitioned from the low-power portions and power is supplied from different pins. Another strategy, called Built-in Current Monitor could also be used [12].

[2] Gate oxide shorts are a common fault that we do not consider. This defect is a bridge between the gate node of a transistor and either its source, drain, or channel region and is often activated after manufacture by burn-in procedures[5]. They can be detected using the techniques we present for detecting transistor stuck-on faults.

include cracks or scratches on the wafer, photolithographic mask misalignments, line dislocations, and major fabrication process control errors. Global defects can have prominent effects on the circuit's logical behavior, or they can manifest themselves as degradations in the performance of all components of the circuit. This paper explores test pattern generation for the detection of local defects only.

Most local defects can be modeled as a spot on a layer of the integrated circuit that changes the electrical properties at that point. A local defect that causes a change to the logical function of the circuit can be represented by a logic-level abstraction known as a *logical fault*. Similarly, a defect that causes a change in a continuous parameter of the circuit can be represented by an abstraction called a *parametric fault*. Some defects may cause only a logical fault, some may cause only a parametric fault, and some may cause both.

Digital integrated circuits are commonly tested for local defects using tests generated to detect single stuck-at faults. In the single stuck-at fault (SSF) model, each defect is modeled as a single gate input or output held to a logic 0 or a logic 1. Many fabrication defects result in undesirable circuit behavior that is not detected by tests generated using the SSF model. Local defects causing either logical faults or parametric faults can go undetected even when tested using input sets that cover 100% of the single stuck-at faults (such an input set is called a *complete SSF test set*).

In a defect-simulation experiment involving over 400,000 defects and two industrial standard cells, greater than 40% of the defects could not be modeled as single stuck-at faults[4]. In the same set of experiments, half of the defects (those causing electrical shorts between adjacent circuit nodes) were fault simulated with a complete SSF test set provided by the circuits' manufacturer: At least 10% of the bridges were not detected by the complete SSF test set.

If only the complete SSF test set was applied to the standard cells presented above, at least 4% of the defects would not be detected. This is quite significant since IC manufacturers typically claim quality levels of less than 200 defective parts per million (DPM). If the fabrication process has a yield of 75%, then the manufacturing tests must detect 99.93% of the yield affecting defects to reach this quality level[14]. Even with a manufacturing yield of 95%, 99.6% of the defects must be detected for a 200 DPM quality level.

An increase in the time it takes for specific sequences of inputs to propagate through a defective circuit is a parametric fault known as a *delay fault*. An increase in the quiescent power supply current (I_{DDQ}) for specific inputs is a parametric fault known as an *I_{DDQ} fault*. (Note that a *test* involves both the appropriate stimulus and the appropriate observation of the circuit to detect a

Feasibility Study on the Costs of IDDQ testing in CMOS Circuits

F. Joel Ferguson*
Tracy Larrabee†

Computer Engineering Board of Studies, University of California, Santa Cruz 95064

Abstract

Many manufacturing defects in static CMOS circuits are not detected by tests generated using the traditional single stuck-at fault model. Many of these defects may be detected as increased propagation delay or as excessive quiescent power supply current (I_{DDQ}). In this paper we compare the costs of detecting probable manufacturing defects by the resulting excess I_{DDQ} with the costs of traditional logical testing methods.

1 Introduction

Perturbations in the fabrication process and contaminants in the environment may cause an IC to deviate from the ideal. Deviations that cause the IC to function incorrectly, or that cause any other undesirable change in a circuit parameter, are called *defects*. Defects can be broadly divided into two groups: *global defects* which affect multiple integrated circuits across a relatively large area of the wafer and *local defects* which affect a relatively small area of an IC. Examples of global defects

*Current address: Baskin Center for CE and CIS, University of California, Santa Cruz, California 95064. Phone: (408) 459-4172. Email: fjf@ce.ucsc.edu. This work was supported by the National Science Foundation under grant MIP-8907380 and by the Semiconductor Research Corporation under Contract 91-DJ-141.

†Current address: Baskin Center for CE and CIS, University of California, Santa Cruz, California 95064. Phone: (408) 459-3476. Email: larrabee@ce.ucsc.edu. This work was supported by the National Science Foundation under grant MIP-9011254.