On Adaptive Diagnostic Test Generation

Yiming Gong and Sreejit Chakravarty
Department of Computer Science, State University of New York
Buffalo, NY 14260

Abstract: *Adaptive Diagnosis, a paradigm for diagnosis, is defined. A system based on this paradigm, for I_{DDQ} measurement based diagnosis of bridging faults, is reported. Experimental evaluation of the system shows it to be substantially superior to existing systems, especially for larger circuits.*

1 Introduction

Given a circuit and the observed faulty response of the circuit, diagnosis locates physical faults which result in the faulty response. It aids in gathering information to improve fabrication processes.

The prevalent diagnosis paradigm is static diagnosis. The tasks in static diagnosis are as follows. A diagnostic test sequence (DTS), targeting a given set of faults F, is precomputed. The **static DTS** is evaluated as to its effectiveness in distinguishing between faults in the set F (**diagnostic simulation** [12]). A **static fault dictionary** is computed either independently or as a by product of diagnostic simulation. Static diagnosis has several disadvantages.

• Diagnostic simulation requires considerable time and space[17], especially for bridging faults (BFs)[4].

• Static fault dictionaries require considerable space thereby leading to maintenance problem. The time required to compute them is also very large. The improved techniques[3] fall apart for larger circuits and fault models with large number of faults.

• Since detection oriented test set don't have good diagnostic resolution, they need to be augmented. This results in large DTS, resulting in storage and maintenance problems, especially for large circuits and fault models with large number of faults in them[4].

Static diagnosis is useful when a large number of different faulty responses needs to be analyzed. The high cost is then amortized over the large number of faulty responses. However, it is often the case that many of the faulty chips have the same faulty response. Thus, the number of distinct responses to be analyzed is small. In this case the disadvantages of static diagnosis become important.

BF occurs when two or more distinct nodes of the circuit are unintentionally connected. BFs model 50% of physical defects in MOS circuits[8]. Once we target BFs the disadvantages of static diagnosis become very significant.

Motivated by this, alternatives to static diagnosis have been studied[1,2,5,6,16]. In dynamic diagnosis[14], no static fault dictionary is used. This eliminates the storage and maintenance problems. The expensive step of diagnostic fault simulation is still required. Storing and maintaining large DTS is still a problem.

In this paper we first define **adaptive diagnosis**, a new diagnosis paradigm. In adaptive diagnosis we *do away with the costly steps of computing static fault dictionaries, diagnostic simulation and static diagnostic test generation. Fault dictionaries* and *DTS are not stored* thereby avoiding the above problems.

We developed a diagnosis system based on this paradigm. It assumes BFs between two nodes, and I_{DDQ} measurement as the testing technique. Experimental evaluation of this system shows it to be substantially better than existing systems, especially for the larger circuits.

2 Adaptive Diagnosis

Adaptive diagnosis is dynamic in that the DTS is generated during diagnosis. Figure 1 shows the structure of the diagnosis system required.

It is also a multi-phase, iterative process. Each phase consists of two parts. In the first part, a small DTS targeting the remaining faults in the fault list is generated. It is then applied to the circuit under test (CUD). The responses, which could either be the logic levels at POs, or the quiescent power supply current, or a combination of the two, of the chip to the test sequence are fed back to the diagnostic test generator (DTG). This information is used in the second part for fault dropping.

The next phase starts with the reduced fault list. DTG generates a new set of vectors. The process continues till no fault dropping is possible, i.e., all faults in fault list are equivalent. Thus, adaptive diagnosis is always "complete" with respect to the targeting faults and the measurements it used. Note that the system requirement is no different from what is required by static diagnosis.

Ideally, in each phase only one test is generated, making the DTG "well informed" about the status of the diagnosis process. But this implies a huge communication overhead. We therefore generate tests in groups, and each group is processed in one phase. The number of phases and the number of tests in each group depends on the fault model, the fault dropping, and test generation process.

There is a potential problem with adaptive diagnosis. The time the CUD spends in the tester (tester time) may be large because the DTG spends too much time during one or more of the phases; or the number of phases is too large. For the case we study we show how this can be avoided very effectively.

3 An Adaptive Diagnosis System For BFs

A BF between two gate outputs can be detected by I_{DDQ} measurement *iff the two shorted lines have different values*[6]. In Figure 2, the two activated paths on application of the vector are shown. If BF $\langle G5, G6\rangle$ exists, a conducting path from V_{dd} to V_{SS} is created. The resulting non-zero I_{DDQ}, detected either by an internal or an external device, signals the presence of a BF.

Our DTG uses a compact representation for BFs[6]. BF between lines x, y are represented by $\langle x, y\rangle$. $\{\langle A, B\rangle\}$ represents a set of faults $\{\langle u, v\rangle | u \in A \; and \; v \in B\}$. Representation $\{\langle A, B\rangle, \langle C, D\rangle, \ldots\}$ means $\{\langle A, B\rangle\} \bigcup \{\langle C, D\rangle\} \bigcup \ldots$. Thus, $\{\langle\{1, 2, 3\}, \{4, 5\}\rangle\}$ represents the set of faults $\{\langle 1, 4\rangle, \langle 1, 5\rangle, \langle 2, 4\rangle, \langle 2, 5\rangle, \langle 3, 4\rangle, \langle 3, 5\rangle\}$. Fault dropping for BFs using I_{DDQ} is done as in [7].

3.1 The Adaptive DTG System

$V(x)$ is the logic value of line x. Two BFs, $\langle x, y\rangle$ and $\langle u, v\rangle$, are $\boldsymbol{I_{DDQ}}$ **Equivalent** iff $V(x) \neq V(y) \iff V(u) \neq V(v)$. Such faults can not be distinguished using I_{DDQ} measurement alone. The objective of the diagnosis system is to reduce the set of faults to one I_{DDQ} equivalent class.

The system consists of a **random test generation** phase, a phase of generating tests **targeting low controllability lines**, and a phase of **deterministic test generation**. In between, after the phase of targeting low controllability lines, a procedure is used to **identify equivalent faults**. This procedure does not generate any test vectors and hence does not form a "phase".

StepI (PhaseI): **In PhaseI, $\log(n)$ number of random input vectors are generated**, using equiprobable distribution. Here n is the number of nodes. Using this test set faults are dropped.

StepII (PhaseII): PhaseI does not generate all distinguishing tests because all lines cannot be set to 1 or 0 with equal probability; and lines are correlated. We address the first issue here and the second in StepIII and StepIV. If line x is hard to set to 0(1) then we say that x has low 0(1)-controllability[11].

Consider faults $\langle x, y\rangle$, $\langle a, b\rangle$ and assume that each of x, y, a, b has low 0-controllability. Then the probability that a randomly selected test vector sets one of them to 0 and the rest to 1 is low. The probability that it sets three of them to 0 and only one to 1 is even lower. Such faults are usually not detected during the random phase. A similar conclusion can be drawn if x, y, a, b have low 1-controllability. **In PhaseII vectors that distinguish faults between low 0(1)-controllabilities lines are generated.** Line-controllabilities are estimated in phaseI as in STAFAN[11].

A set of lines with low 0(1)-controllabilities, corresponding to the faults in the fault list, are identified. If line x has low 0(1)-controllability, we generate test to detect line x stuck at 1(0). These tests are used to drop faults.

StepIII: At this point faults remaining in the fault list are highly correlated, although not all of them are I_{DDQ} equivalent. We next identify as many I_{DDQ} equivalent faults from this set of faults.

Figure 3 shows two kinds of I_{DDQ} equivalent faults. In (a) faults $\langle a, d\rangle$ and $\langle c, b\rangle$, involving "cross coupled" NOT gates, are equivalent (TypeI). In Figure 3(b) faults $\langle x_i, y\rangle$ and $\langle x_j, y\rangle$, where $x_i, x_j \in \{x_0, x_1, \ldots, x_{n-1}\}$, y is an arbitrary line in circuit, are I_{DDQ} equivalent if logic functions $F_1 = F_2 = \cdots = F_{n-1}$ (TypeII).

For TypeII, finding if two lines in a given circuit are equivalent or not is NP-Complete[9]. Therefore, heuristics are used to determine such sets[15]. Our heuristic[10] is a generalization of the one used in [15].

To identify TypeI equivalence, for each pair of equivalent fault sets, we search the fault list to find if "cross coupled" NOT gates exists. If it exists, the pair of equivalent fault sets are equivalent and can be merged into one. For example, if $\{\langle\{e, g, j\}, \{u\}\rangle, \{\langle\{a, b, c, d, f, h, i, k\}, \{v\}\rangle\}$ are two equivalent sets of faults. If $\langle e, u, \rangle$ and $\langle f, v\rangle$ are equivalent, then $\{\langle\{a, b, c, d, f, h, i, k\}, \{v\}\rangle\}$ and $\{\langle\{e, g, j\}, \{u\}\rangle\}$ form one I_{DDQ} equivalent set of faults.

StepIV (PhaseIII): **In PhaseIII, vectors distinguishing all pairs of as yet undistinguished faults are deterministically generated to complete the diagnosis process**. It is similar to redundancy identification.

Two faults $f = \langle x, y\rangle$, $g = \langle w, z\rangle$ are I_{DDQ} equivalent iff in Figure 4 v s-a-0. The test for v stuck-at-0, if it exists, distinguishes f, g.

In the last phase for all pairs of faults in fault list, by adding the circuitry of Figure 4, a new circuit is constructed and handed over to a deterministic stuck-at test generator.

A straight forward implementation of above approach for PhaseIII results in a large test set. We note that this is not necessary. A test distinguishing a pair of fault may also distinguish other pairs of faults. Accordingly, PhaseIII has two sub-phases.

In the first sub-phase, we select pairs of faults with different 0(1)-controllability because faults with different 0(1)-controllability are more likely to be "non-equivalent". Tests to distinguish these pairs of faults are generated. In the second sub-phase, we generates test to distinguish all fault pairs in fault list, one at a time.

After applying the vectors generated in phaseIII, the remaining faults, with their equivalences, are the possible set of faults. These faults are I_{DDQ} equivalent and can not be distinguished further by I_{DDQ} measurements alone. Hence the diagnosis is complete.

4 Experimental Results

The proposed system was implemented using C++ on SUN 4. ISCAS85 and scan version of ISCAS89 benchmark circuits were used. We randomly generated 500 faults to

simulate 500 faulty circuits. The experimental results were obtained using all BFs involving two gate outputs only. Any other set of faults can also be used. We used this extensive fault model to stretch the system.

We present the performance of our system and compare it with a static diagnostic test generation system(SDTG)[4], as well as, a system that uses stuck-at (SSF) test sets. The rationale for comparing with a SSF test set is twofold. Firstly, it might be possible to use the SSF test set, which is readily available, for diagnosis. In this case, we are interested in ascertaining if this approach is superior to what we proposed. Secondly, one could start with the available SSF test set (rather than the random test set we use) and augment it in a similar manner. In this case we are interested in knowing if this leads to a better diagnosis approach.

We call the remaining faults after diagnosis **residual faults**. In our comparison we use the following figures of merit. The most important criteria is the number of residual faults. In our approach the residual faults always reduces to one equivalence class. The second important criteria is the size of the test set. Size of the test set is very important for I_{DDQ} measurement based analysis because I_{DDQ} measurement is a very slow process.

Table 1 presents the performance of our ADTG system. The data under column *Residual Faults* are the (average *Ave*, maximal *Max*) number of residual faults. Note that the number of faults cannot be reduced any further. They are the best you can get if you use only I_{DDQ} measurement. Data under column *Vectors*(*Time*) are the (average, maximal) number of input vectors generated(time used, in CPU seconds) using the proposed system. Time does not include the time for down loading the vectors into the tester, making the I_{DDQ} measurements and getting back the faulty response.

Table 2 presents a comparison between our system and the system which uses SSF test sets for diagnosis. The SSF test set used is from ATALANTA[13]. The data under AD is from our system, and the data under SSF is from using the SSF test set. Time for SSF is only the time required for fault dropping. The two systems share all the diagnosis procedures except that our system generates the tests dynamically and SSF uses the precomputed SSF test set.

For smaller circuits, the number of residual faults are about the same in both cases. But for large circuits ADTG gives substantially better results, both on an average as well as in the worst case. For example, see S15850.1, S35932, S38584.1, $\cdots$. Note that ADTG will never result in a larger residual set than SSF because ADTG is a complete diagnosis system where as SSF is not. In addition, the performance of SSF, so far as reducing the number of faults is concerned, is inconsistent.

Another important advantage of ADTG over SSF is the small size of the test set. In all cases (except for C6288), the average number of I_{DDQ} measurements required, if ADTG is used, is substantially smaller. The average is almost always less than half the SSF test set size. Even if we take the worst case number of vectors generated by ADTG it is substantially smaller than the stuck-at test set. This is very crucial for I_{DDQ} measurement based diagnosis since I_{DDQ} measurement is very time consuming and can adversely affect the total diagnosis time. In addition, it also points to the fact that a stuck-at test set is not a good starting point for dynamic diagnostic test generation.

Table 3 gives the comparison between our system and SDTG system[4]. The data under AD are the average and maximal number of vectors(times) over 500 instances. The data under SD are the vectors(times) the system generated(used). We have not reported data on the residual set size because ADTG will always out perform SDTG. On an average ADTG requires much less measurements (smaller test set). However, the major difference between these two systems, as we can see from the given table, is the test generation time. Even the worst case time for ADTG is order of magnitude smaller than SDTG. In fact, SDTG could not be used to generate tests for any thing beyond C7552. On the other hand we were able to use ADTG effectively for all the benchmark circuits.

References

[1] M. Abramovici and M. A. Breuer, "Multiple Fault Diagnosis of MOS Combinational Networks", *IEEE Trans. on Comp.*, pp.451–460, June 1980.

[2] M. Abramovici and M. A. Breuer, "Fault Diagnosis in Synchronous Sequential Circuits Based on an Effect-Cause Analysis", *IEEE Trans. on Comp.*, pp.1165–1172, Dec 1982.

[3] V. Boppana and W. K. Fuchs, "Fault Dictionary Compaction by Output Sequence Removal", *ICCAD*, 1994.

[4] S. Chakravarty, K. Fuchs, and J. Patel, "Evaluation and Generation of I_{DDQ} Diagnostic Tests for Bridging Faults in Combinational Circuits", Tech. Rep. 95-11, SUNY at Buffalo, Comp. Science, 1995.

[5] S. Chakravarty and Y. Gong, "An Algorithm for Diagnosing Two-Line Bridging Faults in Combinational Circuits", *DAC*, pp. 520–524, 1993.

[6] S. Chakravarty and M. Liu, "I_{DDQ} Measurement Based Diagnosis of Bridging Faults", *Journal of Electronic Testing: Theory and Application (Special Issue on I_{DDQ} Testing)*. Kluwer Academic Publishers, 1993.

[7] S. Chakravarty and S. Suresh, "I_{DDQ} Measurement Based Diagnosis of Bridging Faults in Full Scan Circuits, *7th Int'l Conf. on VLSI Design*, pp. 179–182, 1994.

[8] F. J. Ferguson and T. Larrabee, "Test Pattern Generation for Realistic Bridging Faults in CMOS ICs", *ITC*, pp. 492–499, 1991.

[9] M. R. Garey and D. S. Johnson, *"Computers and Intractability — A Guide to the Theory of NP-Completeness"*, W. H. Freeman and Company, 1979.

[10] Y. Gong and S. Chakravarty, "A Dynamic Diagnostic Test Generation System for I_{DDQ} Measurement Based Diagnosis of Bridging Faults", Tech. Rep. 95-18, Comp. Science, SUNY at Buffalo, 1995.

Circuit	Residual Faults		Vectors		Time	
	Ave	Max	Ave	Max	Ave	Max
C1355	2.186	112	40.83	83	0.7804	2.8
C1908	28.93	1131	36.62	69	3.691	83.18
C2670	9.9	1024	27.33	54	8.095	124.7
C3540	13.73	265	37.84	106	4.248	171.2
C5315	13.66	1746	29.98	61	68.68	29950
C6288	1.312	17	28.05	47	3.392	125.3
C7552	8.93	197	33.24	68	7.365	87.55
S1196	2.26	173	33.51	69	0.9722	7.3
S1238	1.194	4	32.56	68	0.8032	2.76
S1423	8.84	445	26.91	52	1.029	14.51
S5378	22.67	3755	48.19	140	15.82	5583
S13207.1	102.2	21384	72.16	250	141.6	55650
S15850.1	53.8	2595	59.07	259	52.39	9846
S35932	134.6	2729	35.91	49	20.63	64.05
S38417	63.4	2565	59	306	58.26	854.6
S38584.1	3257	259364	95.74	436	855.1	211900

Table 1: Performance of ADTG for METAL

Circuit	Residual Faults				Vectors				Time			
	Ave		Max		Ave		Max		Ave		Max	
	AD	SSF	AD	SSF	AD	SSF	AD	SSF	AD	SSF	AD	SSF
C1355	2.186	2.186	112	112	40.83	85	83	85	0.7804	0.7356	2.8	1.3
C1908	28.93	28.93	1131	1131	36.62	120	69	120	3.691	1.452	83.18	4.75
C2670	9.9	9.9	1024	1024	27.33	105	54	105	8.095	2.614	124.7	6.41
C3540	13.73	13.73	265	265	37.84	153	106	153	4.248	3.362	171.2	4.12
C5315	13.66	13.66	1746	1746	29.98	113	61	113	68.68	4.137	29950	13.67
C6288	1.312	1.836	17	68	28.05	28	47	28	3.392	1.329	125.3	1.64
C7552	8.93	8.93	197	197	33.24	210	68	210	7.365	10.46	87.55	12.13
S1196	2.26	2.26	173	173	33.51	144	69	144	0.9722	1.005	7.3	2.98
S1238	1.194	1.218	4	4	32.56	149	68	149	0.8032	1.008	2.76	1.26
S1423	8.84	13.91	445	880	26.91	71	52	71	1.029	0.8257	14.51	1.55
S5378	22.67	22.77	3755	3756	48.19	251	140	251	15.82	9.695	5583	46.29
S13207.1	102.2	103.2	21384	21567	72.16	469	250	469	141.6	51.64	55650	366.5
S15850.1	53.8	114.2	2595	32765	59.07	440	259	440	52.39	54.15	9846	164.9
S35932	134.6	10266	2729	446670	35.91	70	49	70	20.63	20.97	64.05	35.63
S38417	63.4	63.4	2565	2565	59	888	306	888	58.26	269.1	854.6	302.5
S38584.1	3257	5226	259364	1010106	95.74	676	436	676	855.1	475.8	211900	49400

Table 2: Comparison with SSF Test Set (METAL)

[11] S. K. Jain and V. D. Agrawal, "STAFAN: An Alternative to Fault Simulation", *DAC*, pp. 18–23, 1984.

[12] K. Kubiak, S. Parkes, W. K. Fuchs, and R. Saleh, "Exact Evaluation of Diagnostic Test resolution", *DAC*, pp. 347–352, 1992.

[13] H. K. Lee and D. S. Ha, "On the Generation of Test Patterns for Combinational Circuits", Tech. Rep. 12-93, Dept. of Electrical Eng., Virginia Polytechnic Institute & State University, 1993.

[14] I. Pomeranz and S. M. Reddy, "On the Generation of Small Dictionaries for Fault Location", *ICCAD*, pp. 272–279, 1992.

[15] R. S. Reddy, I. Pomeranz, S. M. Reddy, and S. Kajihara, "Compact Test Generation for Bridging Faults under I_{DDQ} Testing", VLSI Test Symposium, pp. 310–316, 1995.

[16] P. G. Ryan, S. Rawat, and W. K. Fuchs, "Two-Stage Fault Location", *ITC*, pp. 963–968, 1991.

[17] S. Venkatraman, et. al, "Rapid Diagnostic Fault Simulation of Stuck-at Faults in Sequential Circuits Using Compact lists", *DAC*, pp. 133–138, 1995.

Circuit	Vectors			Time		
	AD		SD	AD		SD
	Ave	Max		Ave	Max	
C1355	40.83	83	78	0.7804	2.8	668.43
C1908	36.62	69	74	3.691	83.18	1116.73
C2670	27.33	54	36	8.095	124.7	1331.03
C3540	37.84	106	65	4.248	171.2	2246.38
C5315	29.98	61	45	68.68	29950	3415.43
C6288	28.05	47	53	3.392	125.3	3104.72
C7552	33.24	68	58	7.365	87.55	8230.52
S1196	44.79	69	55	0.9994	7.3	626.58
S1238	33.51	68	70	0.9722	2.76	703.80
S1423	32.56	52	40	0.8032	14.51	687.45
S5378	26.91	140	113	1.029	5583	10152.9

Table 3: Comparison with SDTG System (METAL)

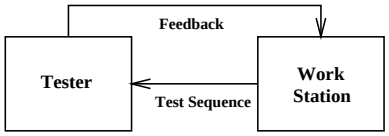


Figure 1: Diagnosis System

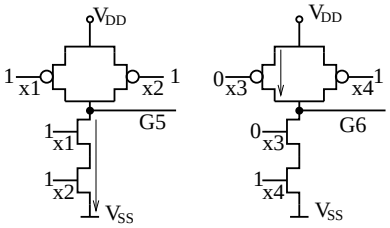
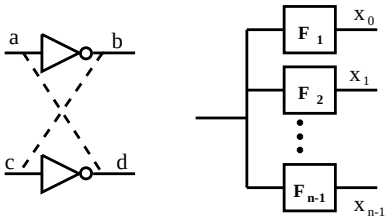


Figure 2: Example Circuit



(a) TypeI **(b) TypeII**

Figure 3: Simple Equivalent Faults

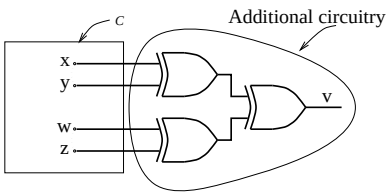


Figure 4: Construct New Circuit for SSF TG