Increased Stuck-at Fault Coverage with Reduced I_{DDQ} Test Sets

SAND--90-0608C

DE90 009509

Ronald R. Fritzemeier, Jerry M. Soden, R. Keith Treece
and Charles F. Hawkins*

Sandia National Laboratories
Albuquerque, NM 87185
Office (505) 845-8575
FAX (505) 846-5004

*The University of New Mexico
Electrical and Computer Engr. Dept.
Albuquerque, NM 87131
Office (505) 277-2264

ABSTRACT

Simplified ATPG and fault simulation algorithms, reduced test set sizes, and increased fault coverage are achieved with I_{DDQ} testing for stuck-at faults. In addition, I_{DDQ} testing will detect logically redundant and multiple stuck-at faults, and improve the detection of non-stuck-at fault defects.

I. INTRODUCTION

Concern over the test metric of single stuck-at fault (single SAF) coverage has plagued the CMOS IC industry for years. Many issues directly affect the ease or difficulty with which one can develop a test set that provides extremely high fault coverage as measured by the single SAF model. Issues which must be dealt with include:

1. cpu intensive software tools and hardware to support automatic test pattern generation (ATPG).
2. Hardware/software tools to perform single SAF simulation for large IC designs in acceptable cpu times.
3. Impact of ignoring undetectable faults.
4. Test economic constraints such as production test times.

In addition to the above issues, there remains the nagging question of whether or not the single SAF model is the appropriate main metric for CMOS IC test coverage given the mismatch between the SAF model and the dominant CMOS IC physical defects.

This paper examines the use of the quiescent power supply current (I_{DDQ}) measurement technique as a more efficient method of attaining high SAF coverage in CMOS ICs as compared to conventional logic response testing. The I_{DDQ} test method measures the quiescent power supply current in CMOS ICs for each static logic state and provides a clear indicator of defects. Levi [1] emphasized the sensitivity of the I_{DDQ} technique for various CMOS defects such as bridging defects, while Malaiya and Su [2] explicitly showed the link between detection of CMOS SAFs and elevation of I_{DDQ}. I_{DDQ} testing significantly changes the requirements for test generation and fault simulation algorithms/tools.

Fig. 1 (a,b) illustrates this I_{DDQ} SAF detection property showing a 2-NAND gate in which the output node C is (a) stuck-at one (SA1) and (b) stuck-at zero (SA0). The output node C-SA1 in Fig. 1(a) can be detected by the I_{DDQ} method if node C is logically driven to the zero state by AB = 11. A contention then occurs as the pull down transistors try to produce a zero state and the SA1 fault holds the one state, causing significant elevation of I_{DDQ} (I_{SSQ}), as measured in the V_{DD} (V_{SS}) path. A similar I_{DDQ} response, providing SAF detection, is observed for node C-SA0 in Fig. 1(b) when AB = 00, 01, or 10. The six SAFs associated with the 2-NAND gate could be detected with just two vectors (AB= 00, 11) if the I_{DDQ} test method were used. This compares to the normal minimum three vectors (AB = 11,10,01) when SAFs are evaluated using conventional logic response testing.

Since the I_{DDQ} method observes the presence of a SAF from elevated I_{DDQ} and not via the logic response of the IC's output pins, all of the computational complexity for output path sensitization that would

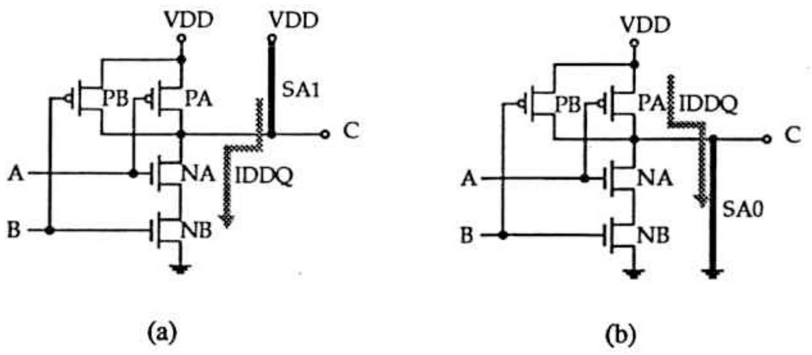


Fig. 1. (a) 2-NAND gate with a SA1 fault on node C.
(b) 2-NAND gate with a SA0 fault on node C.

otherwise be needed to propagate the fault effect to a primary output becomes unnecessary. Malaiya and Su referred to this property as "automatic observability" [2]. This property allows SAFs to be detected much more efficiently using I_{DDQ} techniques than by using logic response testing. Fig. 2 illustrates a SA1 fault on a 2-NAND gate that is buried logically deep inside the IC. Detection of the SA1 fault on the 2-NAND gate by the I_{DDQ} method requires only that a test vector be applied to the primary inputs to control node C to a logic zero.

The benefits of I_{DDQ} testing for ATPG and fault simulation, given this "automatic observability", are straightforward. Whereas logic response test generation requires both fault sensitization and propagation, I_{DDQ} test generation only requires fault sensitization. Fault simulation for grading logic response test vectors can be extremely time consuming because faulty machine behavior must be tracked, for each fault, until the fault effect is positively propagated to a primary output (or all vectors have been simulated). In contrast, "fault simulation" for I_{DDQ} test grading can be done with a single good machine logic simulation with only one added requirement. During logic simulation all SAF sites need to be monitored to record the use of both logic states (0 and 1). The fault coverage measure is simply the percent of individual node-states applied during the simulation. This type of simulation is referred to as *node-state counting*.

I_{DDQ} testing provides still other benefits over the use of conventional logic response testing in that several intractable or unmeasurable types of faults can now be detected [3,4]. These include:

1. logically redundant SAFs
2. multiple SAFs
3. many CMOS defects not modeled by the SAF model

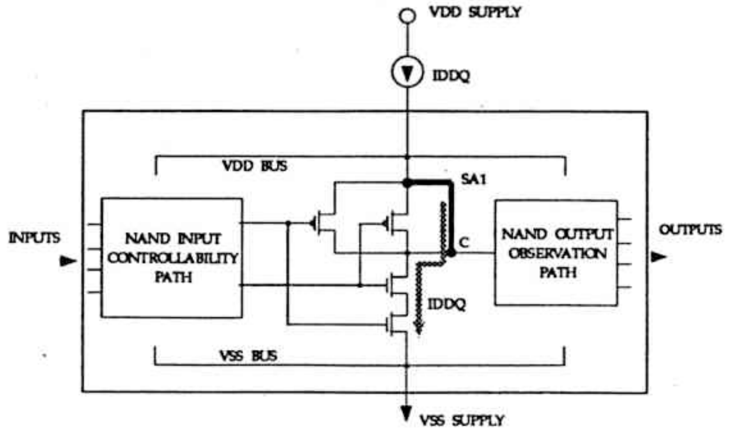
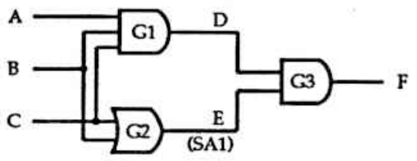


Fig. 2. A general schematic showing a SAF on a logic gate buried logically deep in the IC.

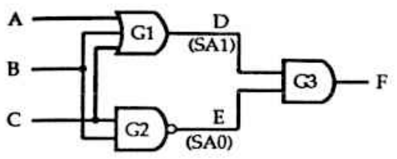
Logically redundant faults are defined as those that cannot be detected by logic response test vectors based on the single SAF model. Fig. 3 shows a circuit in which node E-SA1 is a logically redundant fault [5]. If nodes B and C are set to zero to sensitize the E-SA1 fault, then node D will be set to zero blocking the propagation of the E-SA1 fault effect through gate G3. However, this fault is detectable by the I_{DDQ} method. Any test vector that attempts to drive node E to a zero logic state will elevate I_{DDQ} due to the contention created between the pulldown transistors of gate G2 and the power supply voltage V_{DD} (the "stuck-at-one" fault). The logic table in Fig. 3 shows the minimal node-state test vector set for this circuit. All SAFs are detected by this test set including the logically redundant fault E-SA1 (E-SA1 is detected by I_{DDQ} monitoring on vector ABC = 000).



A B C	D E F	Detect E-SA1
0 0 0	0 0 0	yes
1 1 1	1 1 1	no

Fig. 3. A combinational logic circuit with redundant stuck-at fault E-SA1.

The I_{DDQ} monitoring method also guarantees the detection of both single and multiple SAFs, since multiple SAFs will always increase I_{DDQ} when these stuck nodes are placed in logic contention. In contrast, the detection of multiple SAFs is virtually intractable when using conventional logic response techniques [6]. Fig. 4 shows a simple logic circuit with two designated multiple SAFs (D-SA1, E-SA0). Included in this figure is a logic table showing the minimum test vectors required to achieve 100% I_{DDQ} node-state coverage. Complete node-state coverage for I_{DDQ} testing guarantees detection of all single as well as all multiple SAFs. For example, the table in Fig. 4 shows that the test vector ABC = 000 will detect both SAFs (D-SA1 and E-SA0). The minimum node-state test vector set shown in Fig. 4 will detect the presence of any combination of multiple stuck-at faults if the I_{DDQ} measurement is made for these test vectors.



A B C	D E F	Detect D-SA1	Detect E-SA0
0 0 0	0 1 0	yes	yes
1 1 1	1 0 0	no	no
0 0 1	1 1 1	no	yes

Fig. 4. A combinational logic circuit showing multiple SAFs.

The third additional benefit of the I_{DDQ} monitoring technique lies in its ability to detect CMOS defects that do not behave like classic stuck-at faults: examples include gate oxide shorts, shifted transistor thresholds, bridges, transmission gate opens, and certain circuit configurations of the stuck-open fault [1-4,7-12].

A major purpose of this paper is to present evidence that ATPG for stuck-at faults can be more efficient when the test pattern generation algorithms are redirected to generate test vectors that produce high node-state coverage for I_{DDQ} testing rather than generating conventional vectors for logic response testing. Data given in this paper compare the ATPG properties of cpu time, fault coverage, and vector count for the two testing approaches. A sequential ATPG tool called the Sequential Circuit Test Generator (STG) [13] was used for this work. It was developed by the AT&T Bell Laboratories Engineering Research Center (ERC) in Princeton, NJ, and was slightly modified to generate test vectors, suitable for I_{DDQ} testing, that would show the increase in efficiency over test vectors generated for logic response testing. The ATPG system was modified by forcing all logic gate output nodes to be classified as primary outputs. This removed the path sensitization requirement from the algorithm thereby allowing generation of test vectors that would place each logic gate node in both the zero and one logic states.

A Sandia CMOS IC design plus twenty-two sequential benchmark circuits (ISCAS'89) [14] and two combinational benchmark circuits (ISCAS'85) [15] were evaluated. The comparative results shown in this paper indicate that a significant increase in ATPG and fault coverage efficiency can be obtained when using I_{DDQ} methodology for SAF detection. The ATPG algorithms used in STG3 can be further optimized for generating I_{DDQ} test vectors. A discussion of these details is given later in the paper.

II. METHODOLOGY

The STG3 tool was designed specifically to operate on unconstrained sequential circuits with an option to make use of full or partial scan paths within a circuit design. STG3 incorporates a fault simulator (DSIM) to provide a complete ATPG capability. DSIM, which stands for Differential fault SIMulator, was also developed at the ERC [16].

STG3 was used on all of the circuits evaluated to generate test vectors that would satisfy the node-state condition for all possible logic gate nodes. This was achieved by modifying the netlist of each circuit to declare all logic gate outputs as primary outputs. Normal fault effect propagation to the output pins of the IC was not required in the modified ATPG algorithm since observability was declared at each gate output. STG3 then generated test vectors to provide SAF coverage for these netlists. Some untestable faults and faults abandoned due to STG3's termination criteria were not covered in the final test sets. It should be noted that no design changes, such as resets on flip-flops or scan path insertion, were made to the ISCAS'89 benchmark circuits for the ATPG process.

STG3 produced the data shown in the appendix (Tables A1 and A2) that list the test generation time, number of vectors generated, SAF count, number of faults detected, number of untestable faults, number of abandoned faults, and fault coverage. Table A1 gives results for the conventional (logic response) ATPG process while Table A2 gives results for the modified (I_{DDQ}) ATPG process. A fault is detectable if the logic response at a primary output can be made to differ from the response of the good machine (an X produced on an output in place of a 1 or a 0 is considered to be not detected). Untestable faults are defined by STG3 as those faults that are determined to be untestable by exhaustive ATPG attempts or by analysis. These untestable faults may be logically redundant faults, faults which cannot be detected because no control path exists to sensitize the fault, or faults which belong to nodes that are uninitializable by the STG3 algorithm. For the sequential benchmark circuits the untestable faults have been identified as being caused by uninitializable elements [13]. The faults listed as abandoned were not positively identified as untestable, but showed significantly increased resistance to the generation of test vectors. Faults that were abandoned had as much as four successive ATPG runs with a doubling of the cpu time limit for each successive run. The STG3 program had termination conditions such that untestable faults may not be verified as being untestable. This can occur when the analysis cannot prove a fault to be untestable before meeting the test generation termination criteria. The fault coverage column in Tables A1 and A2 is simply the ratio of detected faults to the total number of faults. Note that untestable faults have not been removed from the total fault count since uncertainty exists as to the reason for these faults being declared untestable; therefore, it may not be appropriate to assume that all of the untestable faults are really "don't-care". Don't-care faults are those faults assigned to nodes that are essentially wired to a power supply value.

The difference in untestable fault counts between the conventional ATPG data and the modified ATPG data shows the number of faults which become testable. These faults become testable for I_{DDQ} testing because logically redundant faults are observable or because the computational complexity is reduced to the point where the algorithm succeeds in finding a test prior to programmed termination. If an initialization capability were provided, either by global set/reset of flip-flops or by scan loading, then all of the untestable faults for the sequential benchmark circuits would be detectable [13].

III. RESULTS

Test vectors for node-state conditions were generated for twenty-five circuits. The first circuit to be described below involved a CMOS IC designed at Sandia. The next twenty-two circuits studied were from the ISCAS'89 sequential logic benchmark circuits [14]. The last two circuits evaluated were from the ISCAS'85 combinational logic benchmark circuits [15].

The Sandia CMOS IC had 3700 logic gates with several asynchronous loops and poor observability; it had a typical manually-generated functional test set in excess of 100,000 test vectors which provided an estimated 65% single SAF coverage. This circuit is an example of "impossible" testability for conventional ATPG methods, but quite efficient SAF coverage was obtained when running the modified ATPG to create vectors for use with I_{DDQ} testing methods. The conventional STG3 run on the circuit could detect only three faults after seven days on a Sun 4/260 workstation. STG3 was then used on this circuit to generate node-state vectors for I_{DDQ} monitoring by defining all logic gate output nodes of the circuit as primary output nodes. The results of these operations are summarized in Table 1. The SAF coverage obtained after 76 hours of Sun 4/260 cpu time was 99.97% and was obtained with only 259 test vectors. The thirty-two untestable faults were not caused by logic redundancy or by uninitializable nodes, therefore these faults were classified as don't-cares and removed from the total fault count.

Table 1. ATPG and fault simulation results for the Sandia CMOS IC.

ITEM	Conventional ATPG	I_{DDQ} ATPG
Total faults	11713	11713
Detected faults	3	11678
Vectors	NA	259
ATPG time	7 dys	76 hrs
Untestable Faults	NA	32
Abandoned Faults	NA	3
Fault Coverage	NA	99.97%

These data demonstrate one of the greatest potential uses of ATPG targeted at node-state test generation. Circuits that have not been designed to have "reasonable" testability features now may have a method capable of generating high coverage SAF test vectors when I_{DDQ} test methods are adopted.

Next, a series of ATPG runs were performed on twenty-two of the sequential ISCAS'89 benchmark circuits. The experience with the Sandia IC, along with theoretical projections of the efficiency of generating test patterns for I_{DDQ} versus conventional, led us to expect that the most significant change in the ATPG statistics would be in reduced vector counts. In fact, the decrease in cpu time, while achieving slightly higher fault coverage, appeared as a more significant result.

Table 2 shows the comparative data (from Tables A1 and A2 in the appendix) for generating conventional logic response vectors and for generating node-state I_{DDQ} vectors. The time factor (C/I) is the ratio of cpu time to generate the two test sets, vector factor (C/I) is the ratio of vector counts for conventional versus modified ATPG, and FC factor (I/C) is the ratio for the increase in fault coverage by the I_{DDQ} vectors. Note that a Sun 4/260 was used for all ATPG runs except for the modified ATPG run on the s1423 which was run on a Sun 3/60.

The data show significantly improved ATPG run times. The s510 was the only circuit that took longer to generate I_{DDQ} vectors than conventional vectors. The reason for this appears to be the change in untestable faults. For conventional ATPG the s510 is completely untestable without initialization and STG3 was able to quickly come to this conclusion and halt test generation. For all other circuits, ATPG for I_{DDQ} required less cpu time by factors ranging from 2 to 162 (The s1423 would show even greater reduction in cpu time if the modified ATPG run had been performed on a Sun 4/260 instead of a Sun 3/60). The only cases where the cpu time actually increased for the modified ATPG runs involved circuits where significant numbers of faults became testable, due to the enhanced observability of the I_{DDQ} methodology, and more time was required to generate tests for these faults. The faults that became observable apparently still had very complex controllability and therefore required significant increases in cpu time to either generate an appropriate test or to decide that a test still could not be found.

Table 2. ATPG Comparison Results for ISCAS'89 Sequential Circuits (C=Conventional, I=I_{DDQ})

Circuit Name	Time Factor [C/I]	Vector Factor [(C/I)]	FC Factor [I/C]
s27	6.56	1.07	1.00
s208	2.25	1.09	1.17
s298	161.85	1.78	1.06
s344	13.55	1.07	1.02
s349	2.13	1.26	1.01
s382	2.41	1.33	1.03
s386	1.69	1.69	1.12
s400	2.06	2.11	1.03
s420	29.26	1.80	1.34
s444	2.30	2.32	1.03
s510	0.76	N/A	N/A
s526	6.83	1.41	1.09
s526n	7.64	1.59	1.08
s641	2.87	1.77	1.03
s713	21.08	1.59	1.07
s820	77.29	3.74	1.04
s832	88.78	4.06	1.05
s953	2.30	0.69	3.76
s1196	5.41	3.17	1.002
s1238	7.35	3.44	1.05
s1423*	139.09	0.43	2.85
s1494	19.25	1.82	1.06

* - run on Sun 3/60 cpu instead of Sun 4/260

Only two circuits did not show a reduction in vector count, the s953 and the s1423. Both of these circuits, however, had extremely low fault coverage for the conventional ATPG vectors and fault coverage improved nearly 3 to 4 times for the I_{DDQ} vectors.

Two combinational logic circuits from the ISCAS'85 benchmark group were studied (c6288,c7552). Tables A3 and A4 in the appendix give the results of conventional STG3 runs and modified STG3 runs. Table 3 shows the comparative results from Tables A3 and A4. c6288 took more cpu time and generated one additional vector for the modified ATPG run. The difference was that several faults that were declared untestable for the conventional ATPG run were no longer easily identified as untestable faults (similar issue as for the s953 and s1423). It is probable that these faults are logically redundant faults that also happen to have very complex controllability requirements. The netlist modification performed to allow the I_{DDQ} version of ATPG to be run would have the effect of removing the logical redundancies but it would have no impact on improving controllability.

Table 3. ATPG Comparison Results for ISCAS '85 Combinational Circuits (C=Conventional, I=I_{DDQ})

Circuit Name	Time Factor [C/I]	Vector Factor [(C/I)]	FC Factor [I/C]
c6288	0.64	0.97	1.0009
c7552	10.51	2.05	1.0156

One reason for our interest in the combinational benchmark circuits was the applicability of scan testing to test vector generation for I_{DDQ} measurement. One could conceive of a testing process having a normal scan test procedure in which I_{DDQ} is measured in between each serial scan operation. The scan testing approach has the potential of significantly reducing the number of vectors which would require I_{DDQ} measurement; therefore, the test time time penalty would be extremely small or possibly even beneficial for the I_{DDQ} approach.

IV. DISCUSSION

This work shows the results obtained when an ATPG tool intended for conventional single SAF test sets undergoes a simple modification to generate a more efficient test set for detection of SAFs using I_{DDQ} test methods. The results from the Sandia IC show that the I_{DDQ} test vector set provides a dramatic solution to detecting faults in a previously untestable circuit. Very high SAF coverage of the circuit was obtained (99.97%) with only 259 test vectors. There is no other practical approach for this circuit that would allow ATPG to be performed in a tractable time period (76 hours for the modified ATPG). Also, a production tester time comparison can be made for the 259 I_{DDQ} test vectors against the more than 100,000 functional vectors. Assuming an I_{DDQ} test at 100 kHz versus a functional test at 10 MHz, the I_{DDQ} test would run about four times faster than the functional test. For this circuit, the I_{DDQ} method offers nearly 100% SAF coverage (compared to an estimate of 65% for the functional vectors) with a significantly shorter test time.

The results for the sequential benchmark circuits in Table 2 show significant improvement in ATPG cpu time (2 to 162 times improvement). The SAF coverage for the I_{DDQ} test vectors was also improved over that of the conventional ATPG vectors, but not as much as might be expected. Fig. 5 shows a plot of the difference in the number of SAFs detected by I_{DDQ} ATPG versus conventional ATPG for the sequential benchmark circuits. The least squares fit of the data clearly indicates a trend toward increased fault detection for increasing circuit size. Even with the improved fault detection of the I_{DDQ} vectors, 100% SAF detection was not always achieved. The lack of 100% SAF coverage is an indication of the difficulty that the STG3 tool had in simply controlling signal nodes. A topological analysis of the ISCAS'89 benchmark circuits showed wide variations in sequential properties among the circuits [17].

Fig. 6 is a plot of the ratio of vector counts for conventional ATPG versus modified ATPG. The three ISCAS'89 circuits that had significant numbers of untestable faults which became testable for modified ATPG were not included since these represent significant changes in overall fault coverage (thus a vector count comparison is not meaningful). The least squares fit of these data shows a strong positive trend. An extrapolation of the fitted line to a circuit size of 50,000 nodes gives a reduction in vector count of two orders of magnitude for modified ATPG.

Further improvements to the STG3 tool might provide better coverage if they enhance the ability of STG3 to determine node control requirements. For example, STG3 would become more efficient if all SAFs on each fanout net (SAFs on stems and branches) were treated as equivalent faults. For conventional test generation, all faults on branch nodes are not equivalent to the stem faults so that individual control for test generation purposes must be performed. For I_{DDQ} measurement, the branch SAFs are always

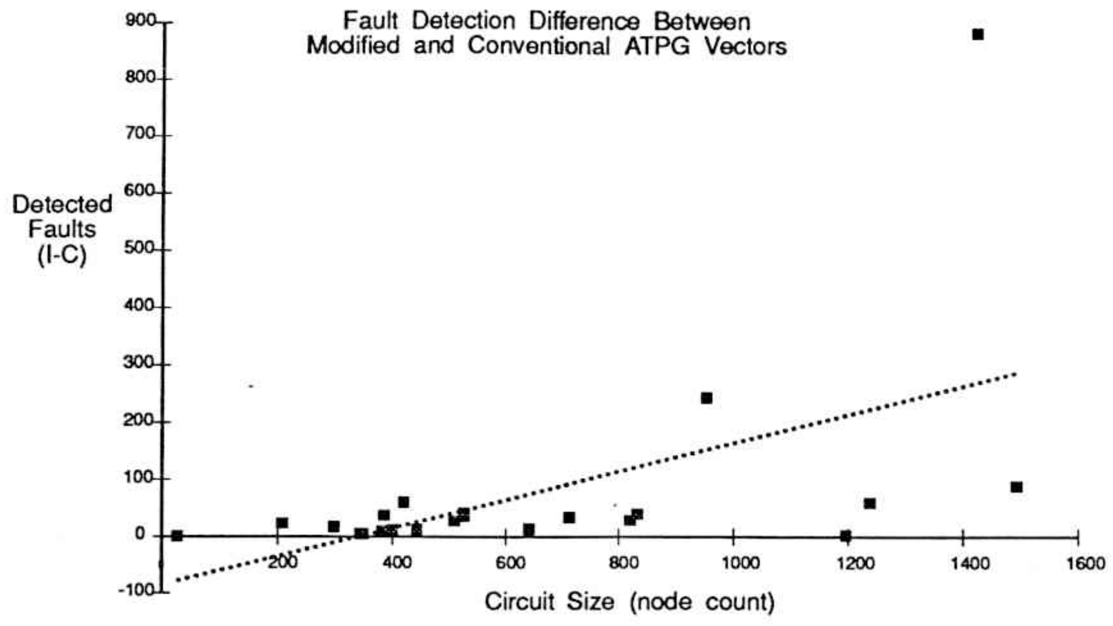


Fig. 5. SAF detection count difference between modified ATPG and conventional ATPG.

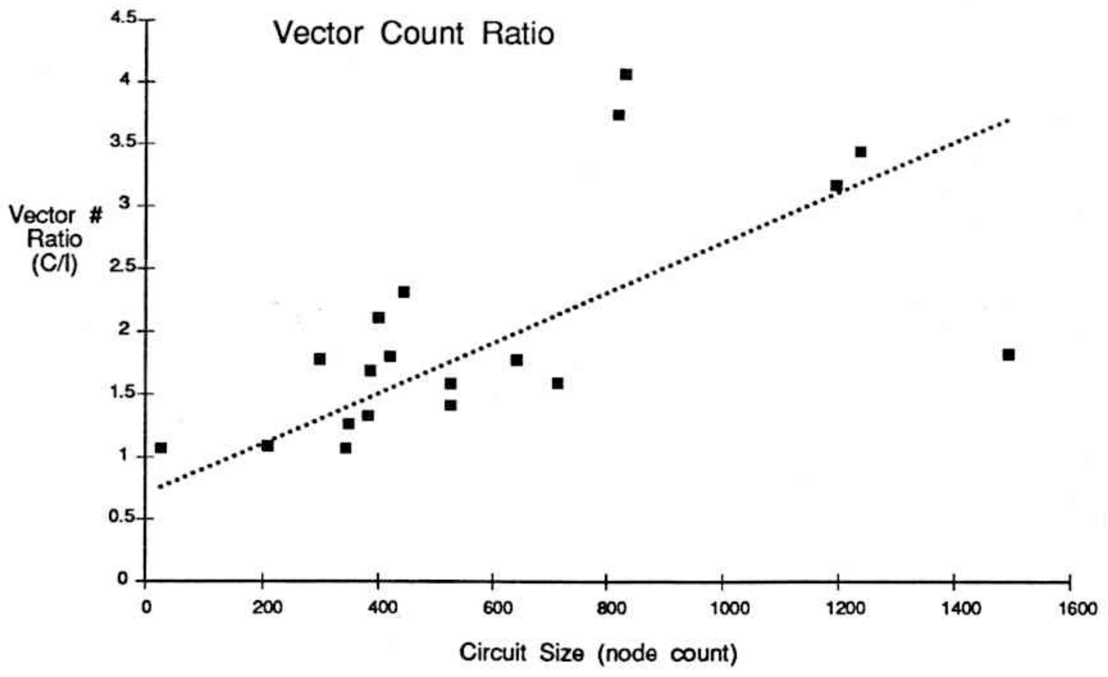


Fig. 6. Ratio of vector set sizes for conventional ATPG and modified ATPG.

equivalent to the stem SAFs. Increases in ATPG efficiency should reduce some of the abandoned faults since these occur due to the timeout criteria for test generation.

The comparative results (Table 3) on the two combinational benchmark circuits showed some improvement in coverage and reduction in cpu time (cpu time reduction was only for the c7552). As with the sequential benchmark circuits, 100% SAF coverage should be expected with further improvements in the STG3 tool. 100% SAF coverage would be possible only after removing the don't-care SAFs from the total fault count.

A significant effect of performing ATPG for I_{DDQ} is that faults previously classified by conventional ATPG as untestable can be further sorted. For uninitializable faults, the statistics will likely not be changed. However, uninitializable faults are rarely, if ever, simply left unattended by the design engineer. This type of fault is usually dealt with by providing an initialization path or by forcing the simulation to run with the uninitializable node(s) forced into one or both logic states. By forcing a fixed state for simulation purposes, the impact of allowing a node to start in a non-determinate state can be assessed. Once this assessment is performed, test generation can usually be accomplished by providing a fixed start-up logic state for the uninitializable node(s). Modified ATPG can generate appropriate tests for faults that are logically redundant. The remaining untestable faults can then be classified as don't-care faults because there really is no means by which the good circuit can be exercised to sensitize the fault (that is, to place the node into a logic state opposite that of the fault to be tested). If untestable faults are sorted in the above fashion, then there is a good basis for arguing that the untestable faults may be removed from the total fault count without negatively impacting the certainty of the fault coverage measurement.

V. CONCLUSIONS

Significant improvements can be achieved when conventional ATPG algorithms are modified to generate test sets suitable for I_{DDQ} testing. These improvements include reduced vector set sizes, increased SAF coverage, coverage of logically redundant SAFs and multiple SAFs, increased coverage of CMOS non-SAF defects, and significantly reduced cpu cost for ATPG and fault simulation. Additionally, untestable faults can be further categorized to identify SAFs that are truly don't-care, thereby offering a more realistic assessment of actual fault coverage.

ACKNOWLEDGEMENT

Much credit should go to Scott Davidson and Tapan Chakraborty of AT&T Bell Laboratories Engineering Research Center for their support. This support included the use of the STG3 tool and software modifications. We also thank John Guth for his helpful comments and suggestions. This work was performed at Sandia National Laboratories and supported by the U. S. Department of Energy under contract DE-AC-04-76DP00789.

APPENDIX

Table A1. ATPG Statistics for Sequential Benchmark Circuits Produced for Conventional Logic Response Testing

Circuit	ATPG	VEC	Fault	#	#	#	Fault
Name	Time(s)	Count	Count	Det	Untst	Aband	Cvg
s27	0.21	15	32	32	0	0	100.00%
s208	15.26	111	215	137	78	0	63.72%
s298	2099	162	308	264	35	9	85.71%
s344	332.97	91	342	329	9	4	96.20%
s349	488.98	91	350	335	10	5	95.71%
s382	5359	1093	399	364	11	24	91.23%
s386	877.97	167	384	314	70	0	81.77%
s400	6289	1754	424	382	15	27	90.09%
s420	478.0	173	430	179	251	0	41.63%
s444	8402	1900	474	424	16	34	89.45%
s510	0.98	0	564	0	564	0	00.00%
s526	34967	1409	555	445	41	69	80.18%
s526n	39156	1558	553	448	41	64	81.01%
s641	11.22	149	467	404	63	0	86.51%
s713	437.82	132	581	473	87	21	81.41%
s820	18805	889	850	809	15	26	95.18%
s832	26301	955	870	814	15	41	93.56%
s953	14.00	11	1079	88	991	0	8.16%
s1196	37.68	339	1242	1239	3	0	99.76%
s1238	60.62	392	1355	1283	72	0	94.69%
s1423	140700	91	1515	476	8	1031	31.42%
s1494	23764	726	1506	1403	30	73	93.16%

Table A2. ATPG Statistics for Sequential Benchmark Circuits Produced for I_{DDQ} Measurement Testing

Circuit	ATPG	VEC	Fault	#	#	#	Fault
Name	Time(s)	Count	Count	Det	Untst	Aband	Cvg
s27	0.03	14	32	32	0	0	100.00%
s208	6.78	102	215	160	55	0	74.42%
s298	12.97	91	308	281	27	0	91.23%
s344	24.58	85	342	334	7	1	97.66%
s349	229.98	72	350	340	9	1	97.14%
s382	2222.97	820	399	374	21	4	93.73%
s386	519.67	99	384	352	32	0	91.67%
s400	3052.90	831	424	394	24	6	92.92%
s420	16.34	96	430	240	190	0	55.81%
s444	3651.43	820	474	437	31	6	92.19%
s510	1.28	11	564	28	536	0	04.96%
s526	5116.68	996	555	487	42	26	87.75%
s526n	5123.35	982	553	484	43	26	87.52%
s641	3.92	84	467	418	49	0	89.51%
s713	20.77	83	581	507	73	1	87.26%
s820	243.32	238	850	838	2	10	98.59%
s832	296.27	235	870	854	2	14	98.16%
s953	6.08	16	1079	331	748	0	30.68%
s1196	6.97	107	1242	1242	0	0	100.00%
s1238	8.25	114	1355	1343	12	0	99.11%
s1423*	1011.59	211	1515	1357	9	149	89.57%
s1494	1234.25	399	1506	1492	4	10	99.07%

* - STG3 run on Sun3/60 for this circuit so the cpu time comparison is not entirely accurate.

Table A3. ATPG Statistics for Combinational Benchmark Circuits Produced for Conventional Logic Response Testing

Circuit Name	ATPG Time(s)	VEC Count	Fault Count	# Det	# Untst	# Aband	Fault Cvg
c6288	7434	35	7744	7695	34	15	99.37%
c7552	2783	474	7550	7418	131	1	98.25%

Table A4. ATPG Statistics for Combinational Benchmark Circuits Produced for I_{DDQ} Measurement Testing

Circuit Name	ATPG Time(s)	VEC Count	Fault Count	# Det	# Untst	# Aband	Fault Cvg
c6288	11674	36	7744	7702	18	24	99.46%
c7552	264.8	231	7550	7534	16	0	99.79%

REFERENCES

[1] M. Levi, "CMOS is Most Testable," Int. Test Conf., pp. 217-220, Oct. 1981.

[2] Y.K. Malaiya and S.Y.H. Su, "A New Fault Model and Testing Techniques for CMOS Devices," pp .25-34, Nov. 1982.

[3] P. Nigh and W. Maly, "Test Generation for Current Testing," European Test Conf., pp. 194-200, Paris, France, April 1989.

[4] J.M. Soden and C.F. Hawkins," Electrical Properties and Detection Methods for CMOS IC Defects," European Test Conf., pp.159-167, Paris, France, April 1989.

[5] R.R. Fritzemeier, H.T. Nagle and C.F. Hawkins, "Fundamentals of Testability," IEEE Trans. Ind. Electron., pp. 117-128, vol. 36, no. 2, May 1989.

[6] J.L.A. Hughes and E.J. McCluskey, "An Analysis of the Multiple Fault Detection Capabilities of Single Stuck-At Fault Test Sets," Int. Test Conf., pp. 52-58, Oct. 1984.

[7] G.F. Nelson and W.F. Boggs, "Parametric Tests Meet the Challenge of High-Density ICs," Electronics, pp. 108-111, Dec. 11, 1975.

[8] J.J. Zasio, "Non-Stuck Fault Testing of CMOS VLSI," Dig. of Papers, COMPCON, pp. 388-391, Spring 1985.

[9] J.M. Acken, "Testing for Bridging Faults (Shorts) in CMOS Circuits," Des. Auto. Conf., pp. 717-718, June 1983.

[10] F.J. Ferguson and J.P. Shen, "A CMOS Fault Extractor for Inductive Fault Analysis," IEEE Tran. Comptr., pp. 1181-1194. vol. 7, no. 11, Nov. 1988.

[11] C.F. Hawkins, J.M. Soden, R.R. Fritzemeier and L.K. Horning, "The Use of Quiescent Power Supply Current Measurements in Detection of Defects in CMOS ICs," IEEE Trans. Indust. Elec., pp. 211-218, vol. 36, no.2, May 1989.

[12] J.M. Soden, R.K. Treece, M.R. Taylor and C.F. Hawkins, "CMOS IC Stuck-Open Fault Electrical Effects and Design Considerations," Int. Test Conf., pp. 423-430, Aug. 1989.

[13] W.T. Cheng and S. Davidson, "Sequential Circuit Test Generator (STG) Benchmark Results," Int. Symp. Ckts. and Sys. (ISCAS), pp. 1939-1941, June 1989.

[14] F. Brglez, D. Bryan and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," Int. Symp. Ckts. and Sys. (ISCAS), pp. 1929-1934, June 1989.

[15] F. Brglez, H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and A Target Translator in FORTRAN," Int. Symp. Ckts. and Sys. (ISCAS), IEEE, June 1985.

[16] W.T. Cheng and M.L. Yu, "Sequential Differential Fault Simulation," 1989 Design Automation Conf., pp. 424-428, June 1989.

[17] A. Lioy, P.L. Montessoro, and S. Gai, "A Complexity Analysis of Sequential ATPG," Int. Symp. Ckts. and Sys. (ISCAS), pp. 1929-1934, June 1989.