Test Generation for Bridging Faults in CMOS ICs Based on Current Monitoring versus Signal Propagation

U. Gläser, H. T. Vierhaus, M. Kley, A. Wiederhold

German National Research Center for Computer Science (GMD)

Abstract

Bridge-type defects play a dominant role in state-of-the-art CMOS technologies. This paper describes a combined functional and overcurrent-based test generation approach for CMOS circuits, which is optionally based on layout information. Comparative results for benchmark circuits are given to demonstrate the feasibility of voltage-based versus IDDQ-based testing.

1 Introduction

The problem of generating tests that cover bridge-type defects has been dealt with by many authors throughout the late 80s and early 90s [1,2,3].
There are two basic problems: First, bridge-type defects will often not result in logic faults. Delay faults are more likely, but recent investigations [4] indicated only minor delays and even occasional speed-up effects for bridges within logic gates.
Second, the number of all possible bridging faults between nodes of a large circuit may be prohibitively high.
The first problem has found a partial solution by introducing static overcurrent (IDDQ) testing [3,5]. Compared with voltage-based testing, current test is a much more sensitive instrument with respect to the detection of bridge-type defects, at least in static CMOS circuits [4].
It was recently demonstrated [3] that a good coverage of bridging faults can be achieved with a very small number of test patterns. However, the results may be overoptimistic with respect to global bridges. No layout information for fault list reduction was used, and the storage space required was not shown.
The identification of "candidates" for bridge-type defects from layout was reported before [7]. Mostly an inductive fault analysis process was used, which may be very precise, but requires prohibitively high computational efforts for large circuits.
The main objective of this paper is to explore the feasibility of bridging fault IDDQ testing with and without previous layout analysis and to compare the results with a realistic voltage-based approach.

2 Layout Proximity Analysis

Based on an existing layout extractor tool [8], we developed a weighted proximity analysis program for CMOS layouts.

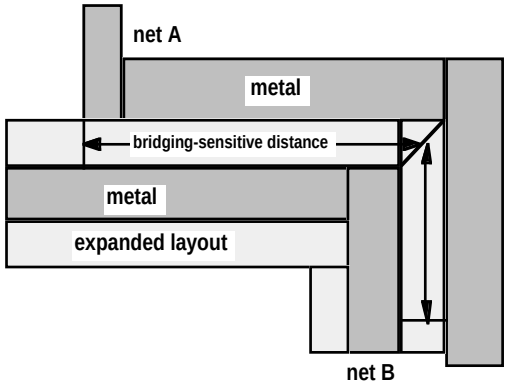

Fig. 1: Layout analysis for bridging fault sensitivity

The extractor is used to identify the association of layout structures with signals. Nodes are registered with annotation of associated layout structures. In general we do not assume bridging faults between non-overlapping structures on different layers.

Then layout structures are expanded by a user-definable number of microns, which will normally be slightly larger than the minimum distance between lines on the same layer.
Then overlap conditions where two nodes approach each other are registered, even the total area of the proximity section is recorded. This value may serve as a measure to evaluate the statistical importance of a bridging fault in such an area.
We also record crossovers between lines of different nodes (metal / polysilicon) optionally. Hence we can systematically evaluate a layout for nodes which are prone to

- bridging at the same level of interconnect (e. g. by dust particles)
- bridging at crossovers metal / polysilicon (e. g. due to pin-holes through oxide layers).

The resulting fault list then serves as an input to an adapted ATPG program such as MILEF [6] or a fault simulator. Run times required for the layouts of ISCAS 85 benchmark circuits are acceptable (see Tab. 1).
The time for the circuit extractor has to be added. It is in the order of a few minutes for most of the examples and slightly below 30 min. for the largest benchmark layouts on an advanced pre-commercial tool with hierarchical facilities [8].
The number of bridges extracted requires further treatment. We subtract bridges between signal nodes and VDD / GND, since such defects will most likely be detected by stuck-at patterns. We also subtract bridges at the polysilicon level within gates (also complex gates), since those are also quite likely to be detected by "normal" ATPG in combination with overcurrent tests [2,3].

Tab. 1: Analysis process for the layout of ISCAS 85 layouts (not including efforts for circuit extraction from layout)

circuit	extracted bridges	CPU-time (SPARC 2) min:s
mc17	36	0:00.47
mc 432	1214	0:19.72
mc 499	1315	0:22.00
mc 880	1451	0:25.19
mc 1355	1475	0:35.07
mc 1908	2012	0:42.79
mc 2670	3284	1:33.64
mc 3540	4406	2:46.70
mc 5315	9242	9:13.73
mc 6288	7354	7:46.75
mc 7552	10148	11:19.33

3 Bridging Fault Testability

3.1 Basics

In general we can distinguish between two basic types of bridging faults:
"Local" (or intra-) bridging faults occur within logic gates, e g. between different input lines, or between inputs and internal nodes.
"Global" (inter-) bridging faults may occur on (mostly metal) interconnects, which are arbitrarily placed close to each other by the routing process during chip construction.
Recent investigations [2,3,4] have shown that test patterns, which are generated for stuck-open-, stuck-on-, and transition faults within logic gates or complex gates, will often excite the fault condition in case of internal bridging defects. Then, if not by delays or functional faults, the testability via static supply current measurements is highly likely.
Hence our main concern is the generation of test patterns for bridging defects that occur on the global wiring.
Using the facilities in MILEF [6], the approach taken is quite different from previous work.
If a circuit level description of the circuit to be tested is available, an extraction process is performed first. If only primitive gates are identified, the ATPG process is based on gate-level information only. In this case bridging fault ATPG will be based on virtually all circuit nodes, because the number of purely "internal" nodes in basic gates is small (e. g. ANDs are split into NANDs plus inverters). If complex gates are used, a local switch-level ATPG process for such subcircuits is performed and coupled to a global gate-level ATPG procedure. As essentially all local bridging faults are excited by switch-level structural ATPG, the global ATPG process for bridges on interconnects is limited to external nodes.

3.2 Testing Non-Feedback Bridges

For the first type of fault, the situation is described in Fig. 2. A low-resistance bridge connects two nodes in a CMOS circuit. It will inevitably cause a static overcurrent condition for the driving stages, whenever the nodes are driven to non-equal values. IDDQ -based testability is almost guaranteed.

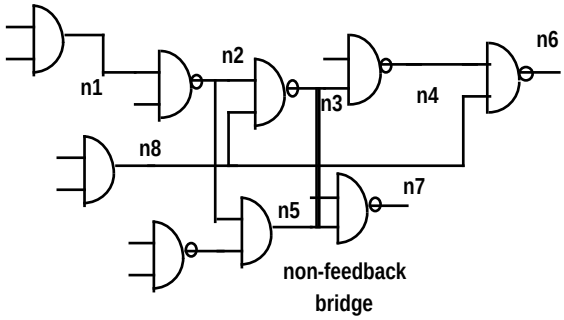

Fig. 2: Global bridging fault condition without feedback

Also a voltage-based testing approach, which tries to excite and to propagate "false" logic values, is of practical interest. Then the objective is to generate a fault condition, where the "stronger" driver distinctly drives the "weaker" node to a logic fault via the bridge. The propagation will then be possible via the loading network of the "weaker" driver. Here the objective was to define realistic logic strengths based on gate-level information only.
It is possible to define a hierarchy of logic strengths for gate-level elements at least for basic logic gates.
1. primary inputs
2. driver cells

3. output drivers of non-inverting gates
4. inverting gates where the state is driven by parallel "on" transistors (e. g. a NAND driving a "high" condition)
5. inverting gates where the state is driven via series "on" transistors (e. g. a NAND driving a "low" condition).

Implicitly this assumption is realistic if a constant width over length-ratio for all p- and n- channel transistors in logic gates, respectively, is applied, except for driver stages. At present we still identify a "strong" driver from one of the first four classes versus a "weak" driver from class 5. Based on such an evaluation scheme, we can distinguish between bridging faults that are likely to be tested via false logic levels and those which are not. Hence the distinction into testable versus untestable bridges is rather leaning towards a pessimistic view.
Based on an initial list of global fault conditions, we first excite the "wrong" condition on the "weaker" node and try to propagate it.

3.3 Testing Under Feedback Loop Conditions

If a feedback condition is excited via an odd number of inverting stages, this may result in circuit oscillations, depending on the input conditions of logic gates in the loop.

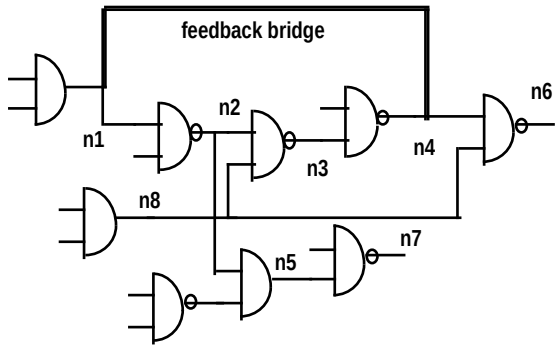

Fig. 3: Global bridging fault condition causing feedback

Assume a feedback condition between nodes n1 and n4. For propagating the false logic state in node 1 to the primary output via node 4, inputs of gates on the path will be assigned states to have their "controlling" input in the path. Hence the conditions for an oscillation are met, if the number of gates in the feedback loop is odd and at least 3. Using the oscillation effect itself for testing is attractive, but not really safe. Very fast oscillations will not certainly be propagated to a primary output, but may yield increased supply currents. For long feedback paths and oscillations below the test clock rate, detection is not safe either.
Safely testing faults via functional effects is possible, if we interrupt the feedback loop (Fig. 3) and surpress the oscillation. If n1 is known to dominate, we can safely propagate the fault effect to node n6.
However, if n4 dominates, we cannot use the path from the "faulty" node n1 to n6 via n4.
Then the procedure to be followed is:
1: Detect the nodes in the path affected by the feedback. 2: Starting at the second node (e. g. n2), find a fan-out node which allows to propagate the faulty value of n1 to a primary output via an alternative path. 3: Starting at the second but last node (e. g. n3), try to interrupt the loop by setting the input (n3) to non-controlling via other inputs. 4: The last gate in the branch affected by step 3 can be the one fed by the fan-out node detected in step 2.

4 Results for Benchmark Circuits

Available results are based on ISCAS 85 benchmark circuits [8]. For IDDQ testing we used the layouts available from MCNC (including complex gates).
The fault list contains all possible bridges of lines outside such macros, hence essentially all global wiring.

Tab. II: ATPG results for global bridging faults, based on complete fault lists, IDDQ-test only

circ. mc	faults	patt. tot.	flts. red.	flts. abt.	cov.	CPU-s	mem MB
17	66	7	0	0	100%	<1	0.1
432	17578	34	0	72	99.59%	<1	0.35
499	52650	42	0	1	99.99%	38	1
880	43660	66	0	2	99.99%	<1	0.87
1355	82621	59	0	4	99.99%	27	1.6
1908	98346	54	0	63	99.93%	4	2.0
2670	289180	144	65	110	99.93%	37	5.8
3540	353220	81	0	69	99.98%	5	7.0
5315	1073845	209	4	161	99.98%	1m:38	21.5
6288	1768140	57	3	27	99.99%	7m:10	35.4
7552	2001000	197	42	294	99.98%	1m:17	40.0

These results show that, even without a previous layout analysis and a corresponding high number of faults, the computing times are acceptably low and resulting test patterns are relatively few. The bottleneck is set by the exploding memory demand.
Results indicate that for workstations with state-of-the-art memory a circuit size of about 10 000 gates presents a practical limit for fast bridging fault ATPG without layout information. (Results in Tab II and III for SPARC 10, 1000 backtracks limit).
Tab. III shows the comparative results for a reduced fault list based on layout information.

Tab. III: ATPG results for global bridging faults, based on a reduced fault list using layout information, IDDQ-test only

circ. mc	flts total	patt.	faults ab.	faults red.	cov.	CPU s
17	9	3	0	0	100%	<1
432	211	12	0	1	99.52%	<1
499	262	9	0	0	100%	8
880	334	18	0	0	100%	<1
1355	371	17	0	0	100%	8
1908	381	12	0	1	99.73%	<1
2670	657	34	0	0	100%	1
3540	784	20	0	0	100%	<1
5315	1722	49	0	1	99.94%	1
6288	1272	17	0	0	100%	48
7552	2049	65	0	1	99.95%	3

The memory requirements for the largest circuits are in the area of 20-30 kB. The number of tests is typically down to 1/3 to 1/5 from the previous list.
For comparison we performed ATPG for bridging faults based on an exhaustive fault list for gate-level benchmark circuits. Bridging faults are excited and propagated where possible, based on the evaluation of signal strengths described in the previous chapter (Tab. IV).

Tab. IV: MILEF performance on ISCAS 85 benchmark circuits, bridging fault test via propagation of false logic values

circ. c	faults total	patt. n	flts abort.	cov. %	CPU s	mem. MB.
17	55	10	0	80	<1	kB
432	19110	24	193	54	1m:45	0.22
880	97903	556	0	50	14s	1.0
1355	171991	1911	57	89	12m:34	3.2
1908	416328	1145	15	64	5m:46	5.5
2670	1016025	4229	1868	33	30m:50	6.8
3540	1476621	2240	235	32	39m:37	11
5315	3086370	2047	472	33	28m:19	21
6288	2995128	313	2	98	5m:17	59
7552	6913621	5064	1119	48	143m:10	68

Results obtained here show that in most cases the number of test patterns exceeds the numbers necessary for IDDQ-based bridging fault test by far. Results also indicate a practical complexity limit of about 10 000 gates for global bridging fault test without layout proximity information.

5 Summary

We presented a comprehensive approach to bridging fault ATPG on interconnects, which is alternatively based on signal propagation or on static overcurrent testing.

As expected, an overcurrent-based test is superior with respect to fault coverage and test generation efficiency. Layout knowledge reduces the number of patterns by a factor of about 3 to 5. Voltage-based testing clearly has only a limited coverage potential, which is, however, far better than expected in some cases.

6 Acknowledgements

This work was partly funded within the ESPRIT Basic Research Action under the ATSEC (6575) contract.

7 References

[1] F. J. Ferguson and P. J. Larrabee, "Test Pattern Generation for Realistic Bridge Faults in CMOS Circuits", Proc. IEEE Int. Test Conf. 1991, pp. 492-499

[2] S. W. Bollinger and S. F. Midkiff, "On Test Generation for IDDQ Testing of Bridging Faults in CMOS Circuits", Proc. IEEE Int. Test Conf. 1991, pp. 598-607

[3] E. Isern and J. Figueras, "Test Generation with High Coverage for Quiescent Current Test of Bridging Faults in Combinational Circuits", Proc. IEEE Int. Test Conf. 1993, pp. 73-82

[4] H. T. Vierhaus, W. Meyer, U. Gläser, "CMOS Bridges and Resistive Transistor Faults: IDDQ versus Delay Effects", Proc. IEEE Int. Test Conf. 1993, Baltimore

[5] W. Maly, P. Nigh, "Built-In Current Testing, a Feasibility Study", Proc. IEEE ICCAD´88, pp. 340-343

[6] U. Gläser, U. Hübner, H. T. Vierhaus, "Mixed Level Hierarchical Test Generation for Transition Faults and Overcurrent Related Defects", Proc. Int. Test Conf. 1992, pp. 21-29

[7] G. Spiegel, "Optimized Test Cost using Fault Probabilities", Proc. 3rd European Test Conf., ETC 93, Rotterdam, pp. 188-193, 1993

[8] V. Henkel and U. Golze, "RISCE- A Reduced Instruction Set Circuit Extractor for Hierarchical VLSI Layout Verification", Proc. 25th ACM/IEEE Design Autom. Conf, 1988, pp. 465-470

[9] F. Brglez, H. Fujiwara, " A Neutral List of 10 Combinational Benchmark Circuits and a Target Translator in FORTRAN" , Proc. 1985 Int. Symp. Circ. and Systems, pp. 671-674, 1985