

Binary decision diagrams (BDDs) for the test pattern generation

M. Dalpasso, M. Favalli

*DEIS - Università di Bologna, Viale Risorgimento 2,
I-40136 Bologna, Italy*

Abstract

Since fault-free CMOS digital integrated circuits are characterized by very low values of quiescent power-supply current (I_{DDQ}), and the most part of their faults provoke abnormally high values of such current, I_{DDQ} testing is one of the most effective way of testing them. The classical test pattern generation at the logic level is not accurate enough for I_{DDQ} testing, because the values of actual faulty current must be estimated and compared with the accuracy of the current monitor to decide on the fault detection; on the other side, electrical-level approaches are not feasible for VLSI ICs. The test pattern generation for I_{DDQ} testing can be effectively performed with a mixed-level approach, featuring boolean manipulation by means of BDDs at the logic level, and using look-up tables for electrical-level data obtained by preliminary SPICE simulations on library cells.

1 Introduction

The quiescent power-supply current monitoring (I_{DDQ} testing) has been shown to be very favourable for testing digital VLSI CMOS ICs (Agrawal[1]). In fact, fault-free integrated circuits fabricated with such a technology are characterized by very low steady-state power-supply currents, while the most part of their faults (that are bridges, Maly[2]) provokes very larger currents (Mallarapu[3]).

The actual effectiveness of I_{DDQ} testing in terms of the defect level (Williams[4]) of tested ICs depends on many parameters, including the sensitivity of the monitoring equipment (that can be either integrated,

Maly[5], or external to the circuit, Hawkins[6]), the clock frequency during testing, the length of the test sequence, and the achieved fault coverage. As regards the two latter issues, so far (Nigh[7], Mao[8], Ferguson[9], Chakravarty[10], Isern[11], Bollinger[12], Reddy[13]) test pattern generation as well as fault simulation for I_{DDQ} testing considered only the activation of faults (i.e., the satisfaction of logic conditions that drive the two faulty-bridged nodes to opposite logic values, Nigh[7]), without any regard to the actual value of current provoked by the fault in comparison with the minimum current that is detectable by the sensor. On the contrary, such a comparison strongly influences the fault coverage, as shown in Dalpasso[14].

Since the value of I_{DDQ} depends on the conductance of the involved networks of MOS transistors as well as on the resistance of the bridging fault, the fault simulation as well as the test pattern generation cannot be simply considered at the logic level, but electrical-level simulations must be performed. However, full electrical-level simulations (for instance, with the SPICE simulator) cannot be performed for VLSI integrated circuits, because of memory and CPU time requirements.

As a viable alternative, a method of library characterization with SPICE will be outlined to consider all the faulty situations, and to fill-in some tables to be looked-up within a logic-level automatic test pattern generator (ATPG). The main engine of the ATPG relies on symbolic manipulation of boolean satisfiability conditions, performed by means of Binary Decision Diagrams (BDDs, Bryant[15]).

Several benchmark circuits have been used to assess the proposed ATPG method: the fault coverage achieved with the generated test sequences ensures a significantly better quality as compared with traditional approaches, with a negligible increment of the test length.

2 Motivation

The following of this work will be focused on bridging faults connecting two signal lines (external bridges); the handling of internal faults (connecting nodes inside the same functional cell) has been achieved with straightforward modifications that will not be mentioned here (see Dalpasso[14] and Dalpasso[16]).

To analyze the typical behavior of a faulty circuit as regards the quiescent power-supply current, let us consider the simple condition of Fig. 1, where the output lines of gate G1 and G2 are bridged by a fault. A possible activating condition is shown in the figure: the pull-down of G2 can be conductive only if both its inputs are driven by a high logic level, while the pull-up of G1 can be conductive in two different ways. Depending on whether or not both the pull-up transistors of G1 are ON,

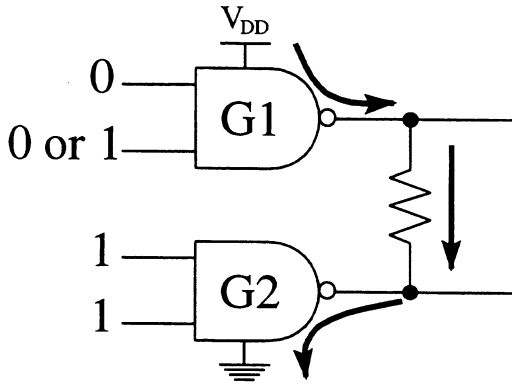


Figure 1: The bridging fault is activated, for instance, by the shown logic configurations. However, depending on the number of conducting transistors in the pull-up of G1 (either one or two), the actual value of the flowing current is different, namely I_1 and I_2 : if the minimum value of current I_t that is detectable as faulty by the sensor is such that $I_1 < I_t < I_2$, then the fault is detected only when both inputs of G1 are driven by 0. The activation is a necessary but not sufficient condition.

two different values of faulty current can be measured (with a given value R of the bridging resistance): I_1 with inputs 0 and 1, I_2 with both inputs at 0. If the minimum current that can be detected as faulty by the sensor is I_t , and the condition $I_1 < I_t < I_2$ occurs, then the fault is detected only by the test vector driving both inputs of G1 to 0. On the contrary, traditional fault simulators for I_{DDQ} consider the fault detection as achieved by both the proposed input conditions.

Such a problem can lead to over-estimate the fault coverage achieved by means of I_{DDQ} testing. Hence, the test quality assessment is invalidated, because it is optimistic.

In fact, Fig. 2 shows the fault coverage evaluated as a function of the sensor accuracy, when a test sequence is applied to a benchmark circuit (C880 from the ISCAS85 set, Brglez[17]): the simulated test sequence give full coverage (i.e., 100%) with classical approaches, that do not consider the actual sensor accuracy, while the actual fault coverage rapidly decreases to useless values if the sensor is not accurate enough. Details on such simulations will be given in Section 5.

Hence, a better test pattern generation strategy must be devised, by properly considering the actual characteristics of the current monitor, in order to guarantee a reliable test quality assessment.

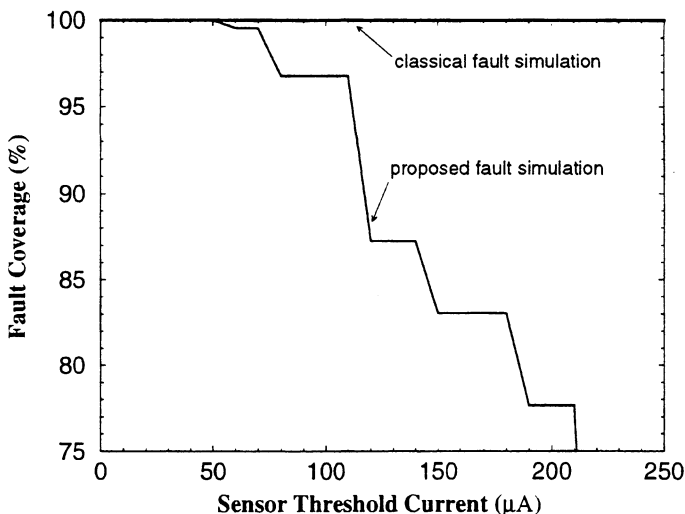


Figure 2: Fault coverage of an I_{DDQ} test sequence applied to a benchmark circuit, evaluated as a function of the sensor accuracy. The result of the “classical” fault simulation is represented by the horizontal line in the plot. In real cases, however, the fault coverage can be very lower than 100% if the sensor is not accurate enough, as shown by the plotted curve.

3 Electrical Characterization

In order to estimate the value of I_{DDQ} that is provoked by a particular logic configuration in a circuit affected by a bridging fault, the two bridged gates should be simulated at the electrical level, along with the gates involved in the fan-out cones of the faulty signal lines that feature faulty intermediate voltages different from the standard values (Favalli[18]), thus making the CPU time unfeasible.

However, as shown in Segura[19] and in Dalpasso[14], the main contribution to the faulty I_{DDQ} comes from the current flowing through the bridging fault, i.e., the fan-out gates can be mostly disregarded. Such an assumption allows for a preliminary characterization of the library cells at the electrical level.

Each pair of cells in the library is simulated for each logic configuration at their inputs, assuming that a bridging fault connects their output. The corresponding faulty current is recorded in look-up tables. The resistance of the bridging fault is assumed to be null, as justified in Dalpasso[16].

4 Test Pattern Generator

The proposed approach for ATPG oriented to I_{DDQ} testing has been implemented in C on Unix workstations (about 6,000 lines of source code). The necessary characterization of the library is directly handled by the program, running the SPICE simulator on the proper subcircuits in turn. Such a characterization is performed once for all over a given library of functional cells.

After such initialization of the electrical data, collected in indexed tables, the fault list is considered: for each target fault, a procedure determines the set of logic configurations at the inputs of the involved cells that provoke a faulty current larger than I_t , by looking-up data in the tables. Finally, such “local” logic conditions are tried for justification at the circuit primary inputs, in order to determine a test vector. The generation of each test vector is followed by simulation and consequent fault dropping.

At first, it should be pointed out that splitting the problem in two separate and independent steps is made possible by the assumption of disregarding the current in the fan-out cells (Dalpasso[16]). Otherwise, the interaction between the logic-level and the electrical-level data cannot be de-coupled.

The most interesting part of the program is the handling of the justification problems, that has been implemented by means of symbolic manipulation of logic expression. The most effective programming strategy for such a task is the exploitation of BDDs. To this purpose, we used the BDD library developed at Carnegie Mellon University (Long[20]).

When the subset of the local logic conditions that provoke a faulty current larger than I_t is identified, the algorithm builds the BDD representing the boolean function that evaluates to 1 when the fault is detected. The boolean variables in the support set of such a BDD are the *local* inputs of both the involved cells, hence such a symbolic manipulation is not very CPU-intensive. For instance, if the *effective* local logic conditions are 1101, 0011 and 1001 (with the local inputs A, B, C and D labelled from left to right), then the fault detection is achieved if and only if the function

$$F(A, B, C, D) = ABC'D + A'B'CD + AB'C'D$$

evaluates to 1.

Then, such identified local configurations must be justified at the primary inputs of the circuit, i.e., a test vector must be found that provokes one of such local configurations in the required locations. The local support of the BDD is, therefore, substituted with the corresponding expressions of signals as a function of the circuit primary inputs, obtaining a function whose support is the set of primary inputs of the circuit.

Hence, an input vector that satisfies such a function can be found with usual operators that are available in the BDD manipulation library (Long[20]). Such an approach guarantees for a solution to be found if it exists, hence faults that are marked as undetectable are really so.

As a general comment on the time performance and memory requirements of the algorithm, they suffer from all the well-known drawbacks that are inherited by using BDD's, as well as they gain all the consequent advantages. To overcome some possible limitations, a particular attention has been paid to the variable ordering, that is a critical problem; even though the used library (Long[20]) implements some dynamic ordering approaches, they have been found to be less effective than static ordering as regards the considered problem: hence, the approach presented in Malik[21] has been followed.

5 Results and Discussion

All the presented results have been achieved on a Sun SPARCstation20 model 50 with 64 MBytes of main memory and 80 MBytes of swap area.

Many combinational benchmark circuits have been considered, both from the ISCAS85 set (Brglez[17]) and the MCNC set (Lisanke[22]): the circuits have been synthesized with SIS (Sentovich[23]), performing the technology mapping on a standard library of static FCMOS functional cells featuring NAND and NOR gates, as long as some more complex cells with up to four inputs.

All transistors in the library cells are minimum-sized, with intrinsic transconductance parameter $k_n = 200\mu A/V^2$ and $k_p = 100\mu A/V^2$ for n- and p-channel MOSFET's, respectively. Their threshold voltage is $|V_T| = 0.8V$.

As regards any data of defect level presented throughout this work, the simple formula of Williams[4] has been used, assuming a technological yield of the fabrication process equal to 75%.

The results will be discussed in detail for the benchmark circuit C880 from the ISCAS85 set. The key issues outlined here can be found for almost any other considered circuit, with obvious differences due to their design and functionality.

Fig. 3 shows the main advantage in terms of fault coverage achieved by using the proposed ATPG strategy (solid curve) in comparison with a typical result achieved by traditional approaches (dashed curve). Since the proposed approach maximizes the faulty current for any fault, the corresponding fault coverage represents the upper-bound for any possible fault coverage achievable by other ATPG strategies. Hence, each sensor gives a higher (or equal) fault coverage when our test sequence is used.

The most interesting gain is related to the sensor accuracy values

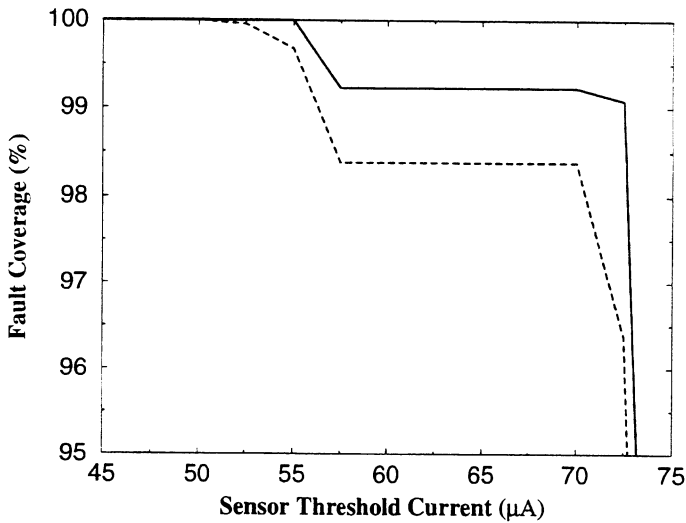


Figure 3: Fault coverage achieved on the benchmark circuit C880 by two test sequences: the former (solid curve) has been generated with the proposed ATPG strategy, while the latter (dashed curve) has been generated with a classical approach that does not consider the actual values of current during the test generation.

ranging from $50\mu A$ to $60\mu A$, where the fault coverage begins to drop from its highest value. In fact, the range of sensor accuracies that guarantees the highest fault coverage is extended by our approach, from $50\mu A$ to $55\mu A$, thus allowing for using a 10% worst sensor and, consequently, obtaining a faster test because it typically needs less time to perform the current measurement. Since in the case of C880 the two test sequences have the same length (66 vectors), the total test time is reduced as compared to classical approaches (the test lengths in the two cases are slightly different for other benchmark circuits).

From a different point of view, using a sensor capable of detecting a faulty current of $55\mu A$, the corresponding defect level can be decreased by our test sequence from 925ppm (part per million) to about 10ppm, as compared to the results obtained by traditional approaches.

Another interesting gain can be achieved when using sensors ranging from $57\mu A$ to $70\mu A$ of accuracy: in this case, the fault coverage is slightly lower, but it can be enough for fast fault grading.

As regards the time performance of the implemented algorithm, the figure of merit is 316 CPU ms per generated test vector for the C880 benchmark circuit, featuring 60 input lines, 268 functional cells, 1816 internal bridging faults, and 35778 external bridging faults. The simulation refers to all the faults possibly affecting the circuit, regardless of their actual



102 Software for Electrical Engineering

occurrence probability, in order to provide worst-case assessment of the time performance.

6 Conclusions

Since traditional approaches to the test pattern generation for I_{DDQ} testing do not take into account for the values of faulty current and suffers from optimism in estimating the test quality, we presented an algorithm to generate test sequences that are optimized by targeting the activation of a current that is detectable by the used monitor.

In such a way, either the actual fault coverage can be significantly increased, or less accurate sensors can be used in detecting currents, as shown by results presented with regard to benchmark circuits.

References

- [1] V. D. Agrawal (editor), "Special Issue on I_{DDQ} Testing," *J. of Electronic Testing, Theory and Applications*, vol. 3, 1992.
- [2] W. Maly, "Realistic Fault Modeling for VLSI Testing," in *Proc. of Design Automation Conf.*, pp. 173–180, 1987.
- [3] S. R. Mallarapu and A. J. Hoffman, "IDDQ Testing on a Custom Automotive IC," *IEEE J. of Solid State Circuit*, vol. 30, pp. 295–299, Mar. 1995.
- [4] T. W. Williams and N. C. Brown, "Defect Level as a Function of Fault Coverage," *IEEE Transactions on Computers*, vol. C-30, pp. 987–988, Dec. 1981.
- [5] W. Maly and P. Nigh, "Built-In Current Testing – A Feasibility Study," in *Proc. of IEEE Int. Conf. on Computer Aided Design*, pp. 149–156, 1988.
- [6] C. F. Hawkins, J. M. Soden, R. R. Fritzmeier, and L. K. Horning, "Quiescent Power Supply Current Measurements for CMOS IC Defect Detection," *IEEE Transactions on Industrial Electronics*, vol. 36, pp. 211–218, 1989.
- [7] P. Nigh and W. Maly, "Test Generation for Current Testing," *IEEE Design & Test*, pp. 26–38, Feb. 1990.
- [8] W. Mao, R. K. Gulati, D. K. Goel, and M. D. Ciletti, "QUIETEST: A Quiescent Current Testing Methodology for Detecting Leakage Faults," in *Proc. of IEEE Int. Conf. on Computer Aided Design*, pp. 280–283, 1990.
- [9] F. J. Ferguson and T. Larrabee, "Test Pattern Generation for Realistic Bridge Faults in CMOS ICs," in *Proc. of IEEE Int. Test Conf.*, pp. 492–499, 1991.



- [10] S. Chakravarty and P. Thadikaran, "Simulation and Generation of I_{DDQ} Tests for Bridging Faults in Combinational Circuits," in *Proc. of IEEE VLSI Test Symp.*, pp. 25–32, 1993.
- [11] E. Isern and J. Figueras, "Test Generation with High Coverages for Quiescent Current Test of Bridging Faults in Combinational Circuits," in *Proc. of IEEE Int. Test Conf.*, pp. 73–82, 1993.
- [12] S. W. Bollinger and S. F. Midkiff, "Test Generation for I_{DDQ} Testing of Bridging Faults in CMOS Circuits," *IEEE Transactions on CAD*, vol. 13, pp. 1413–1418, Nov. 1994.
- [13] R. S. Reddy, I. Pomeranz, S. M. Reddy, and S. Kajihara, "Compact Test Generation for Bridging Faults under I_{DDQ} Testing," in *Proc. of IEEE VLSI Test Symp.*, pp. 310–316, 1995.
- [14] M. Dalpasso, M. Favalli, and P. Olivo, "Correlation between I_{DDQ} Testing Quality and Sensor Accuracy," in *Proc. of IEEE Eur. Design and Test Conf.*, pp. 568–572, 1995.
- [15] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Transactions on Computers*, vol. C-35, pp. 677–691, 1986.
- [16] M. Dalpasso, M. Favalli, and P. Olivo, "Test Pattern Generation for I_{DDQ} : Increasing Test Quality," in *Proc. of IEEE VLSI Test Symp.*, pp. 304–309, 1995.
- [17] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran," in *Proc. of IEEE Int. Symp. on Circuit And Systems*, pp. 663–698, 1985.
- [18] M. Favalli, P. Olivo, M. Damiani, and B. Riccò, "Fault Simulation of Unconventional Faults in CMOS Circuits," *IEEE Transactions on CAD*, vol. 10, pp. 677–682, May 1991.
- [19] J. A. Segura, V. H. Champac, R. Rodríguez-Montañés, J. Figueras, and J. A. Rubio, "Quiescent Current Analysis and Experimentation of Defective CMOS Circuits," *J. of Electronic Testing, Theory and Applications*, vol. 3, pp. 337–348, 1992.
- [20] D. E. Long, "A binary decision diagram (BDD) package." ftp: //emc.cs.cmu.edu: /pub/bdd/ bddlib.tar.Z.
- [21] S. Malik, A. R. Wang, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Logic Verification using Binary Decision Diagrams in a Logic Synthesis Environment," in *Proc. of IEEE Int. Conf. on Computer Aided Design*, pp. 6–9, 1988.
- [22] R. Lisanke, "Logic Synthesis and Optimization Benchmarks – User Guide," tech. rep., Microelectronics Center of North Carolina, FTP address: mcnc.org, 1988.



104 Software for Electrical Engineering

- [23] E. Sentovich et al., "Sequential circuit design using synthesis and optimization," in *Proc. of IEEE Int. Conf. On Computer Design*, pp. 328–333, Oct. 1992.