VHDL Fault Simulation for Defect-Oriented Test and Diagnosis of Digital ICs

F. Celeiro, L. Dias, J. Ferreira, M.B. Santos, J.P. Teixeira
INESC / IST, Apartado 13069, 1017 Lisboa Codex, Portugal
jct@spirou.inesc.pt

Abstract

For high quality VLSI products, exhibiting very low escape rates, defect-oriented testing becomes mandatory. The design activity is more and more supported by hardware description languages, like VHDL; hence, the testing activity needs to follow this trend. In this paper, a VHDL-based methodology for test preparation of digital ICs is proposed, and a new set of tools for defect-oriented VHDL fault simulation are presented, using a commercial VHDL simulator. The proposed methodology is also shown to be effective in supporting realistic fault diagnosis. Simulation results for benchmark circuits are presented.

1. Introduction

The pace of microelectronics technologies progress leads to ever increasing complexity of VLSI and ULSI products, posing difficult challenges for design and test methodologies and tools. At the same time, quality requirements are becoming more and more stringent, as Defect Levels [1] (or escape rates) of the order of 50 ppm (parts per million) are now common [2]. We refer as Defect Level the percentage of defective devices that pass successfully the production test, and thus are marketed as good.

Quality requirements put an additional burden on the test activity, as, for digital Integrated Circuits (ICs), test preparation based on the classic single Line Stuck-At (LSA) fault model is incapable of producing test patterns leading to a sufficiently high defect coverage. Defect-Oriented testing approaches are thus being developed [3], for evaluating the effectiveness of LSA-based testing [4-7] and to extend design and test methodologies, e.g., introducing for CMOS technologies I_{DDQ} (power supply quiescent) current testing [8] or test generation refinements [9].

VHDL has become an industry standard hardware description language for system specification and design. Design and test are two complementary, inter-related tasks in the product development flow. Hence, VHDL needs to be used also for test preparation (see the VITAL initiative [10,11]).

The use of VHDL for test preparation, in the design environment, has been mainly restricted to high-level circuit and fault descriptions. For gate level (the level of abstraction still used for *structural* test generation, and fault simulation), VHDL has been used with the LSA fault model [12]. Preliminary attempts to derive other fault models using VHDL are being proposed [10], namely for node short faults. However, the proposed model is not accurate enough for CMOS. More realistic fault models need to be considered, as test effectiveness [7] needs to be evaluated as the ability to cover *realistic faults* (those originated by physical defects, likely to occur during IC manufacturing), not just to cover abstract, arbitrary LSA faults. In particular, it has been shown that bridging faults, caused by likely spot defects of extra material, are dominant in CMOS process lines [13,14].

The Inductive Fault Analysis (IFA) approach [15] provides a method for realistic fault identification, using the technology, defects statistics and layout information. Since then, several tools for realistic fault extraction have been developed, such as **carafe** [16] and **lift** [17]. Such tools extracts sets of transistor-level realistic faults, weighted by their probability of occurrence. The impact of these faults on the IC logic behavior can be evaluated.

Now, three problems need to be solved, if high-quality VHDL test preparation (and, in particular, fault simulation for test effectiveness evaluation) is addressed:

EURO-DAC ’96 with EURO-VHDL ’96
0-89791-848-7/96 $4.00 ©1996 IEEE

- test preparation is usually carried out in the *top-down* design phase, prior to physical design. Hence, sets of anticipated realistic faults (that we refer as *Pseudo-Realistic (PSR) faults*) have to be identified, and modelled;
- in the *bottom-up* verification phase of the design, once the IC layout is defined, *gate-level* logic descriptions of the realistic faults need to be automatically generated;
- either in bottom-up, or top-down phases, automatic VHDL fault modeling and injection need to be carried out.

The purpose of this paper is to present a methodology for VHDL-based test preparation of CMOS digital ICs, which is supported in VHDL realistic fault modeling, injection and simulation. The methodology, and the supporting tools, are shown to efficiently evaluate the defects coverage of benchmark designs. Moreover, due to the enhanced diagnosability of bridging defects [18], the tools can be rewardingly used for fault diagnosis.

The paper is organized as follows. In section 2, the methodology is presented. Section 3 describe the new tools, **fanthom** and **fastpen**, which are to be used with a commercial VHDL simulator. Section 4 presents simulation examples, and section 5 summarizes the main conclusions of the work.

2. Methodology

2.1 Context and Approach

Our Test Group has developed a set of IFA-based tools, which interfaces with commercially available EDA design systems, such as Synopsys and Cadence. The main tools are **lift** (realistic fault extractor), **clift** (fault collapser), **fancy** (fault classifier), **tabloid** (logic circuit and fault extractor) and **iceTgen** (gate-level realistic test pattern generator and fault simulator). The tools implementing the proposed methodology are to be incorporated in our toolset, and to interface with the Synopsys system.

Test preparation is usually carried out at gate-level, using the LSA fault model, during the top-down phase. Our goal is to be able to still define, at gate-level, the test pattern for high defects coverage, during the top-down phase. Additionally, we aim at performing VHDL fault simulation, using a VHDL description (of the circuit and realistic faults), in order to evaluate the *test effectiveness*, defined, as in [7], as the ratio of defects coverage and LSA fault coverage. Such process can, in principle, be performed using either layout-extracted faults (realistic), during the bottom-up phase, or a set of pseudo-realistic faults (PSR), pre-computed during the top-down phase (Fig. 1).

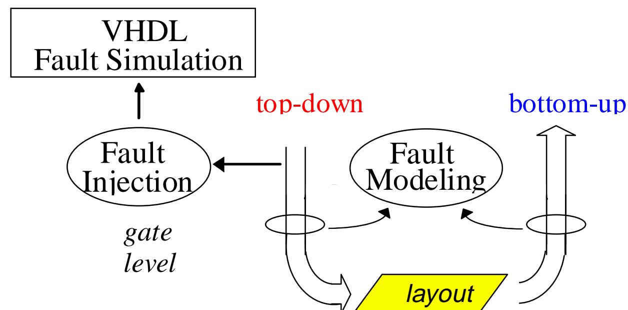

Fig. 1: Proposed approach

As shown, two major processes need to be defined, VHDL realistic fault modeling and injection. Moreover, heuristics for PSR fault generation need to be developed. Finally, the VHDL fault simulation process has to be carried out.

2.2 Fault Modeling

Bridging (BRI) defects are dominant in CMOS process lines; hence, BRIs as assumed as target realistic faults. Logic-level shorts (BRI1 and BRI3, in our classification) are retained for modeling. BRI1 are realistic LSAs (in fact, node stuck-at faults), or BRIs between a power node and a logic node (a I/O node of a logic element). BRI3 are BRI faults between two logic nodes (feedback, or non-feedback).

For the top-down phase, *heuristics for PSR fault generation* need to be derived. The goal is to define sets of BRI faults which, if used as target faults for deterministic test pattern generation, lead to high defect coverage. In this work, two heuristics are compared. First, the *proximity heuristic,* proposed in [9], is based on the anticipated local proximity in the physical design, of i/o terminals of cells and of interconnection lines. Such heuristic, based on the gate-level circuit netlist, aims at deriving a PSR fault set which mimics, as much as possible, the final, realistic fault set. Only easily detectable faults (such as those associated with high fanout nodes) are skipped. Second, the *hard fault heuristic* is investigated, based on the following reasoning: if we could anticipate, from the logic-level schematics, the most hard to detect BRI faults, and generate test vectors to uncover them, then high-quality

tests should be derived (although some of the hard BRIs will not likely occur in the final layout). The new heuristic is based on the evaluation of a testability measure for shorted nodes, from SCOAP controllability /observability measures [19]. Although the correlation between testability measures and fault detection is limited, as Mercer et. al. showed for LSA faults [20], it may be rewarding to investigate if the poor correlation still holds for BRI faults.

Prior to VHDL fault modeling, it is necessary to determine the impact of bridging defects on the IC logic behaviour. This is evaluated using transistor-level information, as the relative strength of shorted pull-up/pull-down paths defines the 0 or 1 dominance, and the wining node. In fact, the assumption in [10] that a node short can be modelled as a change of the references of one VHDL signal for the references of another signal is inaccurate. A local, *damaged sub-circuit* (comprising the logic gates driving the shorted nodes) is defined for each fault, and a truth-table of the damaged subcircuit, in the presence of each fault, is automatically built using the *voting model* [13], for the target technology. Hence, local test vectors, activating each fault, are identified, and later used for ATPG (currently, with **iceTgen**).

In order to define the VHDl modeling technique, it is useful to remind that a circuit described in VHDL is made up of *components* linked by *signals.* Each component may have various implemen-tations. To allow circuit simulation, a *configuration* mechanism assigns a single component description to each component. A component description is represented by an *entity* declaration and an *architecture.* A component description can be *structural* or *behavioral* (or a mixture of both). For structural testing, a structural description (at gate-level) of the circuit is required. However, in our modeling approach, *realistic faults are described by a behavioral description.*

As pointed out by [12], two techniques for fault characterization can be envisaged. A realistic fault can be modelled by an additional component, referred as *saboteur,* or by modifying the fault-free description of existing components, in which case the modified description is referred as *mutant.* For general BRI faults, *mutants* are not a good solution, as the modified description of one gate driving the shorted nodes depends on the input signals of the other gate driving the bridged nodes. Hence, a *saboteur* representation of realistic faults is chosen.

The proposed VHDL realistic BRI fault model is shown in Fig. 2. The correspondent realistic LSA fault model is a particular case of the general one. The proposed model is applicable both for feedback and non-feedback BRI faults. For the latter, a arrival node is identified on the schematics, as it guides the simulation.

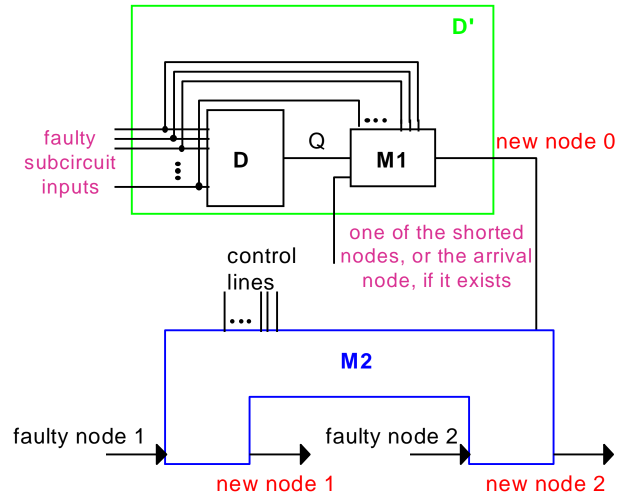

Fig. 2: VHDL model of a realistic BRI: (a) diagram

```
PROCESS (faulty nodes, subcircuit inputs, control
      lines)
BEGIN
      CASE   subcircuit inputs  IS
         WHEN    vector1   =>
                 new node 0 <= ´1´, ´0´ or  ´X´ ;
         WHEN    vector2   =>
                 new node 0 <= ´1´, ´0´ or  ´X´ ;
            *
            *
            WHEN   OTHERS    =>
              new node 0 <= one of the faulty nodes;
      END CASE ;
      CASE   control lines  IS
            WHEN   faulty code  =>
                   new  node 1 <= new  node 0 ;
                   new  node 2 <= new  node 0 ;
            WHEN   OTHERS    =>
                   new  node 1 <= faulty node 1 ;
                   new  node 2 <= faulty node 2 ;
      END CASE ;
 END  PROCESS ;
```

Fig. 2: VHDL model of a realistic BRI: (b) process description

The fault model comprises three blocks:

- a decoder, D, which implements the truth table associated with the fault and the damaged sub-circuit, assigning to Q the result of the voting, in case of conflict;
- a multiplexer, M_1, which assigns, for each local test vector, the value of Q to *new node 0*, and the correct value of each node for each vector not producing conflict; and
- a second multiplexer, M_2, that, when activated by the control lines, assigns to *new node 1* and *new node 2* the value of *new node 0;* when de-activated, M2 assigns to *new node 1* and *new node 2* the values of *faulty node 1* and *faulty node 2,* respectively. The explanation of the control lines is given bellow, as it results from the fault injection technique.

2.3 Fault Injection

Once a fault model has been derived, fault injection must take place for fault simulation. Conventional fault injection is made in such a way that, for each test vector, copies of the damaged circuits (under the presence of each single fault) are generated, whenever the faults are activated by the vector. This allows concurrent fault simulation and fault dropping techniques to be used, to reduce the computational costs. However, using VHDL, the insertion of *saboteurs* in the circuit description quests for a recompilation of the VHDL description, each time a new fault is injected. Such technique would severely increase the computer costs of fault simulation.

Therefore, a different fault injection technique is used. All faults are injected as *saboteurs* in the original VHDL description, using additional signals (control lines) to activate each single fault in sequence (Fig. 3). A number of $\log_2$(# faults) control lines is required. The injection technique allows both single and multiple fault injection, by simple control lines programming.

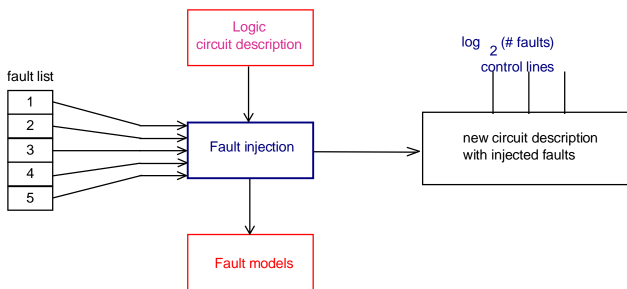

Fig. 3: Fault injection approach

As the main goal of this work is not the development of a VHDL fault simulator, but to prove the feasibility of defect-oriented VHDL fault simulation, a commercial VHDL simulator is used, after fault injection.

3. Tools: Fanthom, Fastpen

Two tools have been developed, **fanthom** and **fastpen**. The fault injector **fanthom** generates the VHDL circuit description of the circuit with the injected realistic faults, and the PSR fault list (when required). The fault simulator driver, **fastpen**, uses the VHDL faulty circuit description (from **fanthom**) and a test pattern to drive the Synopsys VHDL simulator, and to built defects coverage statistics, and the fault dictionary. The use of these tools in the two foreseeable scenarios (top-down, and bottom-up) is illustrated in Figs. 4 and 5, within the context of our proprietary toolset.

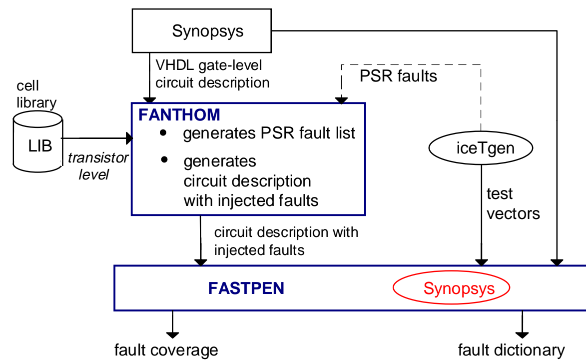

Fig. 4: VHDL fault simulation: top-down scenario

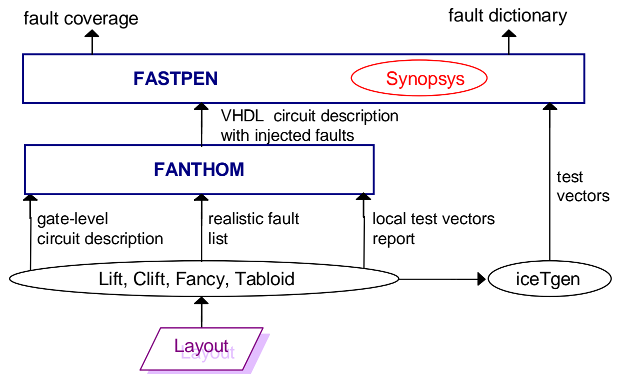

Fig. 5: VHDL fault simulation: bottom-up scenario

4. Simulation Results: Test and Diagnosis

The new tools are being extensively used with several IC macros. Here, only some results are presented, associated with an ALU, and ISCAS'85 benchmark circuits [21],

laied out in a standard cell layout style, using Cadence, ES2 ECD10 technology and the IDlib10 proprietary cell library (Table 1). The test patterns have been generated by **iceTgen** to cover the realistic BRIs extracted with **lift,** except for the c1908 and c3540, for which LSA-based test patterns have been generated. As it can be seen in Table 2, VHDL fault simulation shows that high defects coverage, DC, with these test patterns (generated using realistic faults as target faults) can be achieved. These results were also confirmed by **iceTgen**, except for minor differences, due to soft detection criteria differences.

	# logic gates	# of nodes	# real. faults	# poss. BRIs
ALU	146	160	114	12720
c432	378	414	258	85491
c1355	636	677	637	228826
c1908	1105	1138	5050	646953
c3540	2549	2599	5431	3376101

Table 1: benchmark circuits and faults

circ.	ALU		c432		c1355	
#vect	23		67		120	
stat.	# F	DC	# F	DC	# F	DC
total.	114	99.9	258	95.3	637	99.0
LSA	100	98.3	206	99.4	538	99.5
LSA0	25	100.	58	99.6	40	100.
LSA1	75	97.1	148	99.1	498	99.1
BRI3	14	100.	52	91.1	99	100.
3.1	10	100.	14	89.8	89	100.
3.1.1	9	100.	4	100.	60	100.
3.1.2	1	100.	10	84.3	29	100.
3.2	4	100.	38	100.	10	100.

circ.	c1908		c3540	
#vect	147		215	
stat.	# F	DC	# F	DC
BRI3.1	200	79.0	200	100.0
3.1.1	193	79.0	200	100.0
3.1.2	7	100.0	0	-

Table 2: Fault simulation results
(BRI3.1, between gates, 3.1.1, non-feedback, 3.1.2, feedback, BRI3.2, between logic nodes of a single gate; #F, number of faults; DC, defect coverage)

A second set of experiments was carried out, to identify which of the two heuristics for PSR fault list generation leads to higher quality tests. The **iceTgen** ATPG is used to generate test patterns, using the two PSR fault sets as target faults. Next, **iceTgen** is used as fault simulator, with the realistic faults (extracted from the layout) as target faults. For the c432 circuit, the *proximity heuristic* leads the PSR test set to uncover DC = 95.5% of the realistic faults, while the *hard fault heuristic* lead the correspondent PSR test set to DC = 83.9%. This (and other similar experiences) indicate that the poor correlation between testability measures and fault detectability still holds for BRIs; consequently, a test pattern that covers well the (assumed) most difficult to detect BRIs fails to cover well the realistic BRIs. Hence, the *proximity heuristic* is retained.

The last set of experiments concerns fault diagnosis. As the VHDL fault simulation with **fanthom**, **fastpen** and a commercial simulator does not perform fault dropping, a complete fault dictionary can be built, together with the information regarding the faulty output vectors, for each test vector. The majority of realistic faults tend to produce unique faulty signatures, i.e., distinguishable sequences of faulty output vectors. In Fig. 6, the cumulative defects coverage of the ALU example is provided, together with the cumulative fault diagnosis. As it can been seen, the majority of realistic faults is diagnosable, in this example; the only undistinguishable faults are some realistic LSA faults, due to stuck-at fault equivalence.

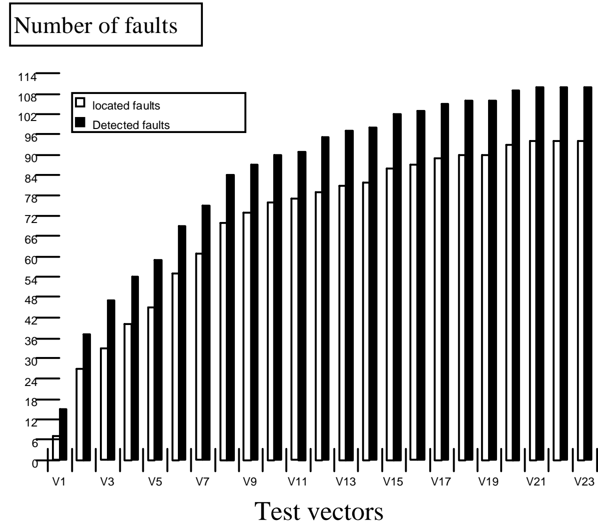

Fig. 6: Located and detected faults in VHDL fault simulation

The major limitation of the tools developed in this work is the computational effort, not due to modelling issues (as faults are described in VHDL by behavioural descriptions), but to the fault simulation strategy. In fact, as the simulation process is carried out by an independent, commercial tool, only sequential fault simulation, without fault dropping, is performed. A strategy to minimize this problem is under development, and will be reported in the future.

5. Conclusions

In this work, a novel methodology for VHDL-based test preparation, using IFA-based tools and VHDL fault simulation, has been proposed. The methodology is

implemented by two new tools, **fanthom** and **fastpen.** These tools can work in the top-down scenario, to allow high-quality test generation (with **iceTgen**) and VHDL fault simulation, or in the bottom-up scenario, to evaluate (in VHDL) test effectiveness. A novel BRI fault model (*saboteurs*) has been introduced, both for feedback and non-feedback bridgings. Two heuristics for PSR generation have been analysed, and the proximity heuristic has shown to lead to higher quality tests. Finally, the methodology and tools can be rewardingly used for realistic fault diagnosis, especially in limited complexity blocks, or test structures. In-house use is being currently made, in connection with a fast prototyping facility, which customizes pre-diffused wafers.

In conclusion, VHDL fault simulation is feasible for defect-oriented test and diagnosis of digital ICs. Further work concerns the development of a VHDL fault simulator and the exploitation of high level test generation techniques.

References

[1] T.W. Williams and N.C. Brown, "Defect Level as a Function of Fault Coverage", IEEE Trans. on Comp., vol. c - 30, pp. 987,988, Dec. 1981.

[2] S.R. Mallarapu, A.J. Hoffman, "I_{DDQ} Testing on a Custom Automotive IC", IEEE J. of Solid St. Circs., vol. 30, Nº. 3, pp. 295-299, March, 1995.

[3] C.F. Hawkins, J.M. Soden, A.W. Righter, F.J. Ferguson, “Defect Classes - an Overdue Paradigm for CMOS IC Testing”, Proc. Int. Test Conf. (ITC), pp. 413-425, 1994.

[4] P.C. Maxwell, R.C. Aitken, “The Effecti- veness of I_{DDQ} Functional and Scan Tests: How Many Fault Coverages do we Need?”, Proc. Int. Test Conf., pp. 168-177, 1992.

[5] M. Saraiva, P. Casimiro, M. Santos, J.T. Sousa, F.M. Gonçalves, I. Teixeira, J.P. Teixeira, "Physical DFT for High Coverage of Realistic Faults", Proc. Int. Test Conference (ITC), pp. 642-651, Sept., 1992.

[6] M Dalpasso, M. Favalli, P. Olivo, J.P. Teixeira, “Realistic Testability Estimates for CMOS ICs”, *Electronics Letters, IEE,* vol. 30, nº. 19, pp. 1593-1595, 15th September, 1994.

[7] L.C. Wang, R. Mercer, T.W. Williams, “On Efficiently and Reliably Achieving Low Defective Part Levels”, Proc. Int. Test Conf. (ITC), pp. 616-625, 1995.

[8] J.M. Soden, C.F. Hawkins, R.K. Gulati, W. Mao, "I_{DDQ} Testing: a Review", Journal of Electronic Testing: Theory and Applications (JETTA), vol. 3, Nº. 4, pp. 291-303, Dec., 1992.

[9] M.B. Santos, M. Simões, I. Teixeira, J.P. Teixeira, "Test Preparation for High Coverage of Physical Defects in CMOS Digital ICs", Proc. 13th. IEEE VLSI Test Symposium (VTS), pp. 330-335, 1995.

[10] J.L. Barreda, I. Hidalgo, V. Fernandez, P. Sánchez, E. Villar, "Fault Modeling in VITAL”, Proc. Workshop on Libraries, Comp. Modeling and Quality Assurance, pp. 191-209, Nantes, 1995.

[11] J.L. Barreda, P. Sánchez, E. Villar, "Current Fault Modeling in VITAL”, Proc. VHDL Users Forum Spring Conf., pp. 23-32, 1996.

[12] Eric Jenn, Jean Arlat, Marcus Rimén, Joakim Ohlsson, Johan Karlsson, "Fault Injection into VHDL Models: The MEFISTO Tool", Proc. IEEE Fault Tolerant Computer Symp. (FTCS),1994.

[13] B. Chess, A. Freitas, F.J. Ferguson, T. Larrabee, "Testing CMOS Logic Gates for Realistic Shorts", Proc. Int. Test Conf. (ITC), pp. 395-402, 1994.

[14] Steven D. Millman and John M. Acken, “Special Applications of the Voting Model for Bridging Faults”, IEEE Journal of Solid-State Circuits, Vol. 29, No. 3, March 1994.

[15] J. P. Shen, W. Maly and F. J. Ferguson, "Inductive Fault Analysis of MOS Integrated Circuits", IEEE Design and Test of Computers, vol. 2, nº. 6, pp. 13-26, Dec. 1985

[16] A. Jee, F.J. Ferguson, “Carafe: an Inductive Fault Analysis Tool for CMOS VLSI Circuits”, Proc. VLSI Test Symp. pp. 92-98, 1993.

[17] J.J.T. Sousa, F.M. Gonçalves, J.P. Teixeira, "Physical Design of Testable CMOS Digital Integrated Circuits", IEEE J. of Solid State Circs., Vol. 26 No. 7, pp. 1064-1072, 1991.

[18] R. C. Aitken, “Finding Defects with Fault Models”, Proc. Int. Test Conf. (ITC), pp. 498-505, 1995.

[19] L. H. Goldstein, "Controllability / Obser- vability Analysis of Digital Circuits," IEEE Trans. Cir and Syst., vol CAS-26, pp. 685-693, Sept. 1979.

[20] M.R. Mercer, B. Underwood, “Correlating Testability with Fault Detection”, Proc. Int. Test Conf. (ITC), pp. 697-704, 1984.

[21] F. Brglez, H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran", Proc. of the Int. Symp. on Circuits and Systems (ISCAS), pp. 662-698, 1985