An intra-cell defect grading tool

Alberto Bosio, Luigi Dilillo, Patrick Girard, Aida Todri-Sanial, Stefano Bernabovi, Paolo Bernardi

An Intra-Cell Defect Grading Tool

A. Bosio, L. Dilillo, P. Girard,
A. Todri-Sanial, A. Virazel
LIRMM-UM2/CNRS
France
<lastname>.lirmm.fr

S. Bernabovi, P. Bernardi
Politecnico di Torino
Italy
<lastname>.polito.it

***Abstract*—With the continuous scaling down of the transistor size, the so-called intra-cell defects are more and more frequent. In this paper we propose a defect grading tool able to evaluate the efficiency of the applied test set. The test set efficiency is quantified w.r.t. the intra-cell defect coverage and the intra-cell diagnosis resolution.**

Keywords—intra-cell defect; test; diagnosis; fault simulation

I. INTRODUCTION

The ever-increasing growth of the semiconductor market results in an increasing complexity of digital circuits. Smaller, faster, cheaper and low-power consumption are the main challenges in semiconductor industry. The reduction of transistor size and the latest packaging technology (i.e., System-On-a-Chip, System-In-Package, Through Silicon Via 3D Integrated Circuits) allow the semiconductor industry to satisfy the latest challenges. Although producing such advanced circuits can benefit users, the manufacturing process is becoming finer and denser, making chips more prone to defects. In modern deep submicron technologies, systematic defects are becoming more frequent than random defects [1].

Today, systematic defects appear not only in the cell interconnection, but also inside the cell itself (intra-cell defect). In literature, existing works prove that these defects can escape classical test solutions. In [2] a statistic carried out over 1 million tested devices shown that a significant number of defects appear inside the standard cell (i.e., intra-cell defects). In [3] it is shown that those defects cannot be detected by using the approaches based on classical fault models (i.e., stuck-at fault model, transition fault model, bridging fault model). Some works targeted the intra-cell defect diagnosis. Basically in [4][5][6] a diagnostic approach taking into account the presence of such defects has been presented.

Despite the fact that previous work already proved that classical test sets lead to a low coverage of intra-cell defects, none of them characterize the applied test set from the diagnostic point of view. Basically the question is how good is the applied test set to diagnose such defects. Moreover, to the best of our knowledge, only one work targets the intra-cell defects [3] fault simulation.

This paper proposes a defect grading tool able to characterize a given test set w.r.t to the intra-cell defects coverage and diagnosability. This tool is composed of two main parts: (1) the library cell characterization and (2) the deductive fault simulator engine.

The paper is organized as follows: Section 2 depicts the overall flow. Section 3 and 4 detail the main steps of the flow, while section 5 presents the experimental results. Conclusions are given in section 6.

II. OVERALL FLOW

Fig. 1 sketches the overall flow that is composed of two steps. The first one is the **technology library characterization**. In this step an automatic tool extracts all the possible defect location for every library cell. Then, for each location a defect injection campaign is executed. It exploits a transistor-level simulator to determine the faulty behavior of each injected defect. The result is the **Defect DataBase**. Please note that this step is applied only one time for a given technology library. The details about the considered defects and the location extraction will be given in the next section.

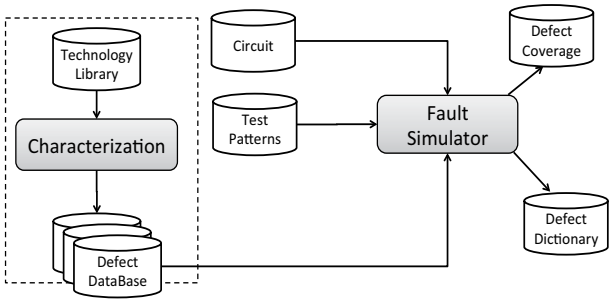

Fig. 1. Overall Flow

The second step is the **fault simulator**. Three inputs are required: (i) the previous computed defect database, (ii) the applied test set and (iii) the gate-level circuit netlist. The fault simulator is based on the deductive fault simulation techniques [7]. It provides two main outputs: the defect coverage value and the defect dictionary. Thanks to the defect dictionary is possible to quantify the diagnosis resolution achieved by the simulated test set w.r.t. to the considered intra-cell defects. The details of the fault simulator will be given in section 4.

III. TECHNOLOGY LIBRARY CHARACTERIZATION

This step aims at characterizing the library cells by means of defects injection campaign. For each library cell, we have to determine all the possible defect location (i.e., where a defect can appear) and the type of defect. In our work the location can be any cell internal net. As already described in previous work [2][3][4], the defect location is guided by a cell layout analysis in order to identify the realistic defect locations. Then, for each

978-1-4799-4558-0/14/$31.00

realistic defect location the defect injection is performed to evaluate if the behavior induced by the injected defect is covered or not by the applied set of stimuli. Finally the defect database is created. Any transistor-level simulator can be used to perform this analysis.

IV. FAULT SIMULATOR

The adopted fault simulator is based on the open source fault simulator presented in [9][10]. Here we modify the original tool by adding two new facilities: (i) the inclusion of the defect database for each library cell and (ii) the deductive fault simulation technique. The first facility is mandatory in order to address the intra-cell defects, while the second one is done in order to save time during simulation compared to the original serial fault simulation (i.e., speed up the simulation time).

As already described, the simulator is based on the deductive fault simulator technique [7]. Basically, every time that a circuit gate is traversed we extract from the defect database the set of sensitized defects. Then, depending on the logic value applied to the gate, we propagate or not the list of sensitized defects. The lists reaching the circuit primary outputs contain the detected defects. Table 1 reports the basic defect lists propagation rules for the classical gates as detailed in [7]. The rules depend on the values applied to the gate inputs. La and Lb are the defect list coming from the previous gate while L is the list of defects sensitized in the current gate due to the application of the input pattern.

TABLE I. PROPAGATION RULES

Gate type	Inputs		Output List
	a	*b*	
AND	0	0	$\{La \cap Lb\} \cup L$
	0	1	$\{La \cap \overline{Lb}\} \cup L$
	1	0	$\{\overline{La} \cap Lb\} \cup L$
	1	1	$\{La \cup Lb\} \cup L$
OR	0	0	$\{La \cup Lb\} \cup L$
	0	1	$\{\overline{La} \cap Lb\} \cup L$
	1	0	$\{La \cap \overline{Lb}\} \cup L$
	1	1	$\{La \cap Lb\} \cup L$
NOT	0	-	$La \cup L$
	1	-	$La \cup L$

For the complex gates, the rules are determined by exploiting the knowledge of the gate structure.

For example, Fig. 2 depicts the internal structure of a multiplexer (i.e., MUX). Thanks to the knowledge of the internal gate structure it is possible to determine the list of defects by applying the rules shown in Table 1.

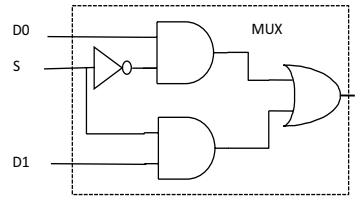

Fig. 2. Mux structure

Fig. 3 gives an example of the simulation algorithm. The example circuit is composed of 6 gates, 11 primary inputs and 2 primary outputs. It reports the simulated pattern. For each primary input, internal net and primary output the actual logic value is showed.

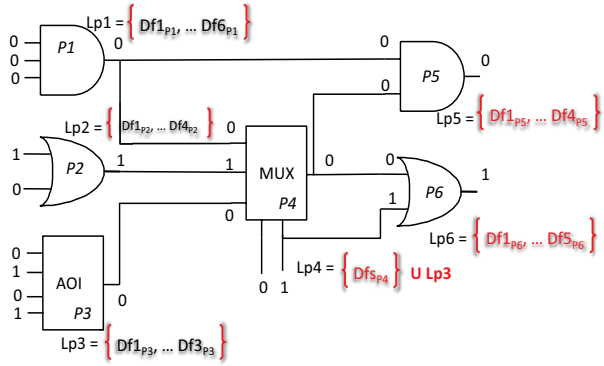

Fig. 3. Simulation Example

During the simulation, one gate per time is processed. During the gate process, the simulator determines the output values and the list of sensitized defects.

TABLE II. SIMULATION EXAMPLE

Step	Processed Gate	Output	Defect List
1	P1	0	L_{P1}= {Df1 $_{P1}$, … Df6$_{P1}$}
2	P2	1	L_{P2}= {Df1 $_{P2}$, … Df4$_{P2}$}
3	P3	0	L_{P3}= {Df1 $_{P3}$, … Df3$_{P3}$}
4	P4	0	L_{P4} = {Df1 $_{P4}$, … Df6 $_{P4}$} U L_{P3}
5	P5	0	L_{P5}= {Df1 $_{P5}$, … Df4$_{P5}$}
6	P6	1	L_{P6}= {Df1 $_{P6}$, … Df5$_{P6}$}

Table 2 shows all the simulation steps. For each step it reports the processed gate, the output value and the defect list. The first three steps process gates P1, P2 and P3. These gates are directly connected to primary inputs, thus the associated defects list contains the sensitized defect of each gate. To be clearer, the defect list of gate P1 (i.e., L_{P1}) contains the defects of P1 (an AND gate) when the input values are "000". In our example these defects are 6 (i.e., from Df1 $_{P1}$ up to Df6$_{P1}$). The same for gate P2 and P3.

During the step number four, the gate P4 is processed. In this case the defect list (i.e., L_{P4}) contains the defects sensitized by the applied input values plus the defects coming from P3.

To obtain this defect list we applied the rules of table 1 to the internal structure of P4. To better clarify this point, we can simply consider the all defects from P3 can invert the logic value of P3 output. This effect will be propagated through P4 due to its input configuration.

In steps 5 and 6, the defects lists coming from previous gates are not propagated through gate P5 and P6. Let us consider the case of gate P5. The input configuration is "00", thus the rule to be applied is $\{La \cap Lb\} \cup L$ (i.e., from Table 1). The list La corresponds to the list L_{P1} coming from gate P1, while Lb corresponds to L_{P2}. The intersection between them is Ø because there are no common defects (i.e., defects coming from the same gate). At the end, the P5 defect list only contains the defects of the gate itself when "00" is applied. These defects are 4 as reported in the table. The same consideration can be done for gate P6.

At the end of the simulation, the defect lists reaching the primary outputs contain the detected defects. For our example these lists are L_{P5} and L_{P6}. Thus, the applied pattern detects 9 defects: 4 defects of gate P5 plus 5 defects of gate P6.

The defect coverage is the first metric used to measure the quality of the applied test set. The second metric is the capability of the applied test set to diagnose the defects. To better introduce the diagnosis metric let us continue the example of Fig. 3. First of all we define the applied test set. Table 3 gives the three applied patterns. The first pattern TP1 corresponds to the one used in the simulation example illustrated in the Fig. 3.

TABLE III. TEST SET

Pattern	Logic Values
TP1	10101001000
TP2	00111110000
TP3	01110110111

After the simulation of the complete test set, the simulator builds the detect dictionary shown in Fig. 4. The defect dictionary is the classical pass/fail dictionary ad defined in [11]. In our example the defect list contains 88 intra-cell defects. For each test pattern simulation, the defect list is divided in subsets depending on which defects are detected and which are non-detected by the simulated pattern. In Fig. 4 the root node contains the 88 intra-cell defects, then after the application of TP1, the initial defect set is dived in two small sets: the defects detected by TP1 are 9 (as already illustrated in the example of Fig. 3) and the un-detected defects that are 79. The process is repeated until all the input patterns have been simulated. At the end of the simulation, the leaves of the defect dictionary contain the so-called *equivalent defects* sets. A set of equivalent defects is a set of defects that shows the same behavior when the test set is applied. Please note that an equivalent defect set depends on the applied test set thus, if another test set is applied the equivalent defect set can change. Thanks to the knowledge of the equivalent defect sets is possible to measure the diagnosis capability of the test set. From the diagnosis point of view, the most important thing is to distinguish between all the detected defects. It means that in the ideal case all the equivalent sets contain only one defect. In our example, we have 8 leaves. One of them, the black one in the figures contains the 39 undetected faults. This leaf is not considered for the diagnosis, simply because logic diagnosis look for the root cause of observed failures. The leaf containing 0 defects means that even after the application of the third pattern no more defects are distinguished. Therefore there are 6 equivalent defect sets useful for the diagnosis. The average size of these sets is 8.16 meaning that the applied test set can identify ~8 possible defects as the root cause of observed failures. Finally, only one leaf contains one defect. Therefore the applied test set can identify one defect among the initial 88. The computed diagnosis capability for this example is very low; good diagnostic results are achieved by larger pattern sets as demonstrated in the next section.

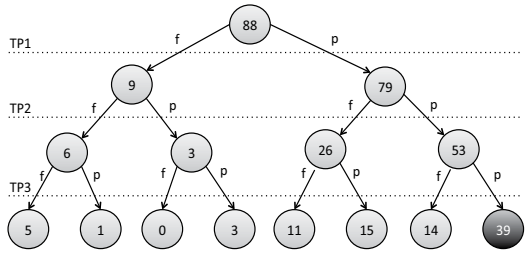

Fig. 4. Pass/Fail Defect Dictionary

V. EXPERIMENTAL RESULTS

In order to evaluate the effectiveness of the illustrated method, we perform several experiments on a set of ITC99 full-scan circuit benchmarks. All the circuits were synthetized using a 90nm technology library composed of 9 logic cells. The characterization of the library gives a total number of 119 defects. For each circuit we generate three test sets by using a commercial ATPG tool. The first test set has been randomly generated and then fault simulated targeting the stuck-at fault (SA). The second and the third are deterministic test sets. One targets the SA faults while the other target the Transition Fault model (TF). Each deterministic test set has been generated by using the "`ndetect = 10`" option meaning that for each fault the ATPG generates 10 different test patterns to test it. We use this option in order to have test sets more likely to detect intra-cell defects as described in [3][4].

Table 4 gives obtained results. The first column reports the circuit name, the second the number of faults (determined by the ATPG) and the third column the intra-cell defects number (determined by the proposed tool). Columns 4, 5 and 6 show the results obtained for the three test sets. For each test set we report the achieved fault coverage (FC%), the intra-cell defect coverage (DC%), the diagnosability (Diag%) and the test length. The simulation time varies from few seconds up to some hours. Please note that the simulation time is due to the implementation and it is not related to the applied simulation algorithm (i.e., the deductive fault simulation).

The first comment about these results is that the achieved intra-cell defect coverage is lower compared to the fault coverage. This result was expected, however the gap between fault and defect coverage is quite high (up to about 35% for the

b09). In previous work [3] the gap was quite low about 5%. This difference is mainly due to the use of a different library. Thus proving once more the importance of the intra-cell defects and the need to generate meaningful test sets.

The second comment refers to the applied test set. All of them have been generated targeting a classical fault model. Even if the generation exploits the ndetect option to increase the defect coverage, the result is very low (down to 33%). No significant differences between test sets have been found, meaning that independently on the targeted fault model and ATPG option, the intra-cell coverage is not enough.

Finally, the last comment is for the diagnosability of the test sets. The reported values (in the Diag% column) correspond to the percentage of intra-cell defects that the test set is able to identify. As clearly reported, the diagnosability is very low, in the best case the 33% of detected defects can be identified. This also implies that the defect equivalent sets are very large. To the best of our knowledge this is the first analysis of a given test set concerning the diagnosability. The result proves once more the importance to target intra-cell defects especially for the diagnosis.

VI. CONCLUSIONS

In this paper we presented a defect grading tool able to simulate defects affecting the library cells. Results carried out on ITC'99 benchmark circuits show the importance of these defects from both test and diagnosis. Future works mainly focus on the analysis of the scan flip-flops in order to estimate the impact of intra-cell defects for the scan-chain test and diagnosis.

REFERENCES

[1] B. Kruseman, A. Majhi, C. Hora, S. Eichenberger and J. Meirlevede, "Systematic defects in deep sub-micron technologies", IEEE International Test Conference, 2005, pp. 290-299.

[2] S. Eichenberger, J. Geuzebroek, C. Hora, B. Kruseman, and A. Majhi, "Towards a World Without Test Escapes", IEEE International Test Conference, 2008, paper 20.1.

[3] F. Hapke, R. Krenz-Baath, A. Glowatz, J. Schloeffel, H. Hashempour, S. Eichenberger, C. Hora, and D. Adolfsson, "Defect-Oriented Cell-Aware ATPG and Fault Simulation for Industrial Cell Libraries and Designs", IEEE International Test Conference, 2009, paper 1.2.

[4] F. Hapke, M. Reese, J. Rivers, A. Over, V. Ravikumar, W. Redemund, A. Glowatz, J. Schloeffel, J. Rajski, "Cell-aware Production test results from a 32-nm notebook processor", International Test Conference, 2012, pp. 1-9.

[5] A. Ladhar and M. Masmoudi, ''Efficient and Accurate Method for Intra-gate Defect Diagnoses in Nanometer Technology and Volume Data'', Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE '09. pp 988-993

[6] Z. Sun, A. Bosio, L. Dilillo, P. Girard, A. Todri, A. Virazel and E. Auvray, **"**Effect-Cause Intra-Cell Diagnosis at Transistor Level", 14th International Symposium & Exhibits on Quality Electronic Design, ISQED'13, pp. 476-483, 2013

[7] D. B. Armstrong, "A Deductive Method for Simulating Faults in Logic Circuits", IEEE Transaction on Computer, Vol. C-21, Issue 5, May 1972, pp. 464 – 471.

[8] J. C.-M. Li, "Diagnosis of Resistive-Open and Stuck-Open Defects in Digital CMOS ICs" IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, 2005, Vol. 24, Issue. 11, pp. 1748-1759.

[9] A. Bosio and G. Di Natale, "LIFTING: a Flexible Open-Source Fault Simulator", IEEE Asian Test Symposium, 2008, pp. 35-40.

[10] A. Bosio, P. Girard, S. Pravossoudovich, P. Bernardi, M. S. Reorda, "An efficient fault simulation technique for transition faults in non-scan sequential circuits", DDECS 2009, pp. 50 – 55

[11] P. Bernardi, M. Grosso, M. Rebaudengo, M. S. Reorda, "A pattern ordering algorithm for reducing the size of fault dictionaries", IEEE VLSI Test Symposium, 2006, DOI: 10.1109/VTS.2006.9

TABLE IV. DEFECT COVERAGE & DIAGNOSABILITY

Circuit	#Faults	#Defects	SA Random				SA Deterministic				TF Deterministic			
			FC%	*DC%*	*Diag%*	*#Patt*	*FC%*	*DC%*	*Diag%*	*#Patt*	*FC%*	*DC%*	*Diag%*	*#Patt*
b01	276	406	100.00	71.92	21.23	295	100.00	78.08	21.14	173	100.00	79.80	20.06	261
b02	172	209	100.00	87.56	23.50	256	100.00	88.04	22.28	115	100.00	88.04	21.74	172
b03	934	1174	100.00	69.25	30.38	360	100.00	68.74	30.61	213	100.00	67.71	30.69	360
b04	5062	6753	89.82	33.48	26.08	608	89.82	33.56	25.42	527	89.78	33.08	25.96	931
b05	4640	6799	99.35	69.85	28.39	871	99.35	69.85	28.38	657	98.64	69.85	32.37	1162
b06	374	473	100.00	81.40	23.38	353	100.00	82.66	24.55	148	100.00	81.82	23.25	250
b07	2870	3981	100.00	39.18	31.21	834	100.00	39.71	30.04	603	100.00	39.89	30.79	1101
b08	1058	1443	99.53	60.85	31.44	396	99.76	60.91	33.53	255	99.76	63.13	30.96	521
b09	1144	1353	100.00	65.19	29.02	295	100.00	65.19	29.02	189	100.00	65.11	30.19	326
b10	1102	1431	100.00	77.43	29.06	565	100.00	79.32	29.96	356	100.00	78.33	29.97	611
b13	1986	2645	100.00	68.54	30.00	617	100.00	68.54	30.23	359	100.00	68.36	30.20	686