Deconstructing the traditional boundaries of the automotive system through the recycling of composite materials in the design and production of the future car

A comparative overview of ATPG flows targeting traditional and cell-aware fault models

N. Mirabella[1,2] , A. Floridia[1] , R. Cantoro[2] , M. Grosso[1] , M. Sonza Reorda[2]
[1]*STMicroelectronics s.r.l., Italy* , [2]*Dept. of Control and Computer Engineering, Politecnico di Torino, Italy*

***Abstract*— Recently, the adoption of Cell-Aware Testing (CAT) has become an option for an increasing number of semiconductor companies. Typically, CAT is adopted in the context of scan chain tests, and patterns are generated with an Automatic Test Pattern Generation (ATPG) tool. Moreover, past studies have extensively shown the capability of CAT to identify the microchips' physical defects that would otherwise remain undetected using traditional fault models, only. However, due to the higher number of patterns generated, an improper CAT-related ATPG flow can lead to a longer test application time. This means higher costs for semiconductor companies, thus reducing the advantages of CAT. The aim of this paper is to overview different ATPG flows supporting CAT, showing advantages and disadvantages of each approach. Each flow is evaluated together with traditional and cell-aware fault models in terms of achievable fault coverage and pattern count. The experimental results are presented through a wide range of open-source benchmarks using a proprietary industrial technology library.**

Keywords*— *ATPG, cell-aware, testing, design for testability, defect testing

I. INTRODUCTION

The continuous miniaturization of technology nodes, combined with the ever-increasing complexity of systems-on-chips (SOCs), especially for application-specific integrated circuits (ASICs), can contribute to an increase in defectiveness of silicon products, making them more difficult to test and achieve low Defective Parts Per Million (DPPM) levels. Customers, on their side, ask for higher quality levels, especially in safety- and mission-critical applications.

As a consequence, microelectronics companies constantly improve manufacturing processes to systematically increase yield. At the same time, they try to reduce the defect level for integrated circuit (IC) designs, regardless of the technology node used. To this purpose, new testing techniques are needed to achieve the least defect rate.

Physical defects can occur at any time during the manufacturing process of semiconductor devices. For this reason, Very-Large-Scale Integration (VLSI)ICs testing plays a key role in detecting possible defects. Basically, ATPG tools use fault models to represent faults into a circuit and create patterns to be used by product engineers on automatic test equipment (ATE). Although common fault models such as stuck-at faults (SAFs), bridging faults (BFs) and transition delay faults (TDFs) can detect a high percentage of defects within a circuit, they view each standard-cell as a black box by considering only the inputs and outputs of each logic gate. As a result, these fault models might not consider some type of physical defects that could possibly occur in the ICs production. ICs require higher quality tests to be applied at the end/during of the manufacturing process, evaluating not only defect-oriented tests [1] but also physical-aware tests [2]. CAT [3], unlike other fault modeling approaches, can model a considerable number of intra-cell defects. These cell-aware faults can be then tested with suitable test patterns generated by an ATPG tool, increasing the overall quality of the manufacturing process by reducing the number of test escapes.

As shown in [4][5], the adoption of CAT (supported by proper ATPG tools) produces a higher number of patterns compared with the other fault models. This in turn causes an increase in testing time on the ATE. To reduce this negative side effect, it is important to find ways to reduce the number of generated patterns and obtain the best trade-off between coverage of all faults, including traditional fault models (stuck-at and transition delay faults) and application time.

For this reason, a good strategy for test engineers is to find the most appropriate test development flow implementing CAT without compromising fault coverage of traditional fault models while maximizing at the same time the coverage of cell-aware (CA) faults addressed by CAT.

Considering the above-mentioned limitations of CAT, the aim of this paper is to propose a comparative overview of different ATPG flows that include CAT. Previous works [6][7] have demonstrated the use of certain ATPG flows supporting CAT. The purpose of the article is not to prove the effectiveness of CAT or to propose new algorithms. These were addressed in previous research that has presented results comparing different failure models [8], determining different methodologies [9][10][11] or improving the characterization of libraries [12]. Differently, in this paper a comparison between different flows (all supported by commercial tools) is performed, to allow test engineers to fully assess their advantages and disadvantages. These flows are evaluated in terms of the coverage obtained and the number of patterns (that ultimately contribute to the testing time). The experimental results were obtained using commercial tools, a proprietary technology that lends itself well to the use of CAT and using open-source benchmarks, all of them with the same scan chain architecture, to make the experiments as design-independent as possible.

This paper is organized as follows: in Section II briefly introduces some background on cell-aware testing; Section III describes in detail the ATPG flows used for this study; Section IV presents and describes the case studies that were used to evaluate the different flows presented in section III; Section V reports the obtained results. Section VI draws some conclusions.

II. Background and Motivations

A. Cell-aware testing (CAT)

Cell-aware testing is a methodology increasingly adopted for modelling and testing intra-cell defects such as short circuits, open circuits and transistor defects that may be neglected by traditional inter-cell fault models.

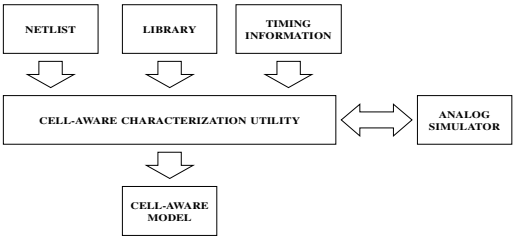

Fig. 1. Cell-aware characterization flow

The CAT methodology consists of a preliminary phase aiming at the cell-aware fault models generation. This is a one-time task performed for each cell of the technology library. This step requires the spice netlist, the technology library, and the timing information data (liberty file). This step starts with the layout extraction phase, then an analog fault simulation is performed, followed by a synthesis phase to create the Cell-Aware library models to be used during the IC-level ATPG.

During the layout extraction, the tool extracts from the cell structure (i.e., a transistor-level netlist with layout information) a list of possible defects. Then, for each of the previously extracted defects, an analog simulation is performed to determine the complete set of cell-input combinations that detect the intra-cell defect and the list of Cell Aware (CA) faults. The analog fault simulation exhaustively (i.e., considering all possible input stimuli) compares the outputs of defect-free transistor-level netlist against the outputs of each defective netlist (i.e., the netlist with the defect). A defect is labelled as detected when the defective cell output voltage deviates from the defect-free voltage by more than a specified percentage of the supply voltage for at least one input combination. A defect is labelled as static-CA fault (CAT-STAT) when there is an output difference for at least one vector in a specified time instant. Defects requiring a pair of vectors to be detected are labeled as dynamic-CA faults (CAT-DYN). This analysis requires comparing at regular intervals the output voltages of the defect-free and defective cell. When the defective cell output voltage deviates from the defect-free only for a given number of strobe instants, the defect is labelled as dynamic. The result of this process is a defect matrix for the library cell. Each matrix contains the input values detecting a particular defect and the classification of the defect as static or dynamic. The third step is the cell-aware fault model synthesis. During this step, the defect matrix is optimized in order to reduce the number of test conditions required for detecting the cell defects.

The outcome of the process described previously is then integrated in the standard test flow where the ATPG tool uses all the traditional and CAT fault models to cover all the defects inside an IC [13], generating the scan vectors able to detect all the considered faults. Fig. 1 summarizes the flow used for the characterization of an industrial library.

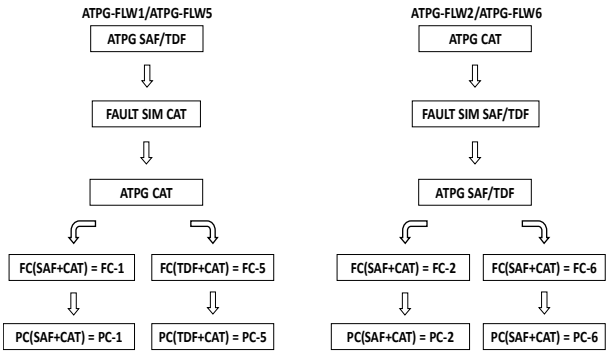

Fig. 2. ATPG-FLW1 (5) and ATPG-FLW2 (6)

III. Possible ATPG Flows

This section presents the different ATPG flows considered for the comparative analysis done in this paper. The flows were divided into two sets of four flows each, considering SAFs and static-CA faults for the first set (ATPG-FLW1 to ATPG-FLW4) and TDFs and dynamic-CA faults for the second set (ATPG-FLW5 to ATPG-FLW8). Since the ATPG tool differentiates the two types of cell-aware faults (static and dynamic), the SAFs flow is matched with the corresponding static-CA faults and the TDFs flow is matched with the corresponding dynamic-CA faults. Furthermore, each flow for a given fault model (e.g., SAFs) has its corresponding identical flow for the different one (e.g., TDFs). It is worth noting that the fault coverage for traditional fault models is ensured by the ATPG tool. In other words, the fault coverage for SAF/TDF is always kept at the maximum.

The following paragraphs detail the eight flows presented above. For every flow once the CA models for the library cells have been defined, the design gate-level netlist (already including the scan-related logic) and libraries are imported into the ATPG tool.

A. SAFs/TDFs ATPG flows incremental with CA faults

Two sets of flows are presented in this section for the SAF and static-CA fault models for ATPG-FLW1/2 and for the TDF and dynamic-CA fault models for ATPG-FLW5/6. The ATPG-FLW1 (and 5, respectively) flow implements a typical ATPG run with a SAF (TDF) model fault list that incrementally add a CAT-type flow. The ATPG-FLW2 (6) flow exploits the opposite approach, i.e., it implements a CA-ATPG run and incrementally add an ATPG flow with a fault list of the SAF (TDF) fault model. These flows exploit the ATPG tool to work in an incremental way.

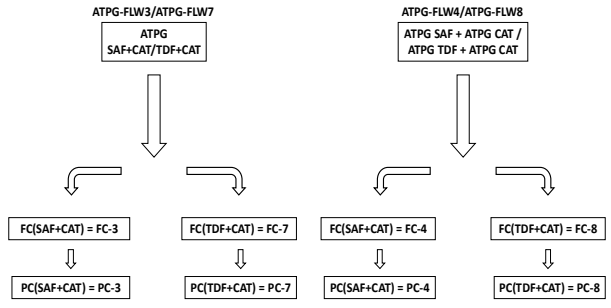

Fig. 3. ATPG-FLW3 (7) and ATPG-FLW4 (8).

The ATPG-FLW1 (5) procedure is as follows:

- The ATPG step is run on the SAFs (TDFs) and suitable patterns are generated to detect them.
- The patterns generated for the SAFs (TDFs) are then fault simulated with the static (dynamic) CAT faults.
- A new ATPG step is then executed to generate further patterns to detect the still undetected CAT faults.
- Finally, fault coverage (FC) and pattern count (PC) figures are collected.

The ATPG-FLW2 (6) is reversed with respect to traditional and CA faults. In other words, the ATPG run starts with the CA faults, then the generated patterns are obtained. A fault simulation of the generated patterns with the SAF (TDF) faults is performed. At last, the incremental ATPG run is performed to cover all the undetected SAF (TDF) faults. The pattern count and the coverage percentage are therefore collected. Fig. 2 summarizes the representation of the above-explained flows.

B. ATPG flow with both SAF/TDF and CAT lists

The other two sets of flows are presented in this section for the SAF and static-CA fault models for ATPG-FLW3/4 and for the TDF and dynamic-CA fault models for ATPG-FLW7/8, respectively. These flows exploit the use of the ATPG run for creating patterns to find the fault coverage of the considered design. Unlike the previously introduced flows, in ATPG-FLW3 (7) the fault list of SAF (TDF) model and the fault list of static (dynamic) CA fault model are merged before the ATPG is executed.

In the ATPG-FLW4 (8) flow, on the other hand, the ATPG flow are split by fault models. That is, one traditional ATPG flow is performed on a fault list with SAF (TDF) and another separate for static (dynamic) CA faults. The obtained fault lists are then merged (preserving fault detection status) within a single flow, resulting in the summary results of the two flows considered. The collected results consider the total patterns and coverage obtained from the two separate flow.

The ATPG-FLW3 (7) is as follows:

- Once the fault model is defined, all SAFs (TDFs) and static (dynamic) CA faults of the current design are added.
- The ATPG is executed and all patterns created are saved, including considering the coverage obtained for all faults of both the traditional and cell-aware fault model.

TABLE I. BENCHMARKS DETAILS

Designs	*Flip-Flops*	*Logic Gates*	*PIs*	*POs*	*SAFs*	*TDFs*	*CAT-STATs*	*CAT-DYNs*
B01	5	29	4	2	270	230	492	397
B02	4	14	3	1	158	124	282	216
B03	30	62	6	4	868	678	1,842	1,721
B04	66	146	13	8	2,062	1,656	5,029	4,448
B05	34	265	3	36	2,408	2,194	5,050	4,986
B06	9	26	4	6	318	254	544	392
B07	45	151	3	8	1,760	1,480	4,001	4,158
B08	21	78	11	4	852	716	1,717	1,528
B09	28	74	3	1	898	720	1,881	1,838
B10	17	83	13	6	846	734	1,632	1,350
B11	30	164	9	6	1,734	1,544	4,191	4,653
B12	121	469	7	6	5,018	4,282	11,033	9,844
B13	51	141	12	10	1,708	1,392	3,398	2,854
B14	215	2,977	34	54	27,408	26,004	78,237	87,402
B15	417	2,560	38	70	27,510	24,802	73,014	72,757

The ATPG-FLW4 (8) consists of two different ATPG runs separately. In practice, the ATPG for SAF (TDF) fault lists and the ATPG for static (dynamic) CA fault lists are run in parallel. Then, the two runs are implemented in a single flow in the ATPG tool to collect the overall fault coverage and pattern counts derived from the separate usage of the two procedures. Fig. 3 summarizes the explained flows.

IV. CASE STUDIES

This section presents the designs used for the purpose of this manuscript.

The Open-source benchmarks presented in [14] were synthesized with STMicroelectronics' proprietary 130-nm HCMOS technology for power applications. About five hundred standard cells were characterized for CAT models. For sake of simplicity and conciseness, only fifteen out of twenty-two benchmarks were considered.

All the experiments were performed resorting to a commercial CAT ATPG tool.

For the experiments, a single scan chain was used for each design. This choice stems from the purpose of this work, i.e., comparing different ATPG flows to observe the testing behavior through simple open architectures. Starting from these designs and approaches it is possible to observe the benefits and drawbacks of each flow in terms of coverage and testing time. In TABLE I. the main figures related to the adopted benchmark circuit designs are collected. For each column, the number of sequential (flip-flops) and combinational cells (Logic-Gates), the number of inputs and outputs (PI and PO), and the number of faults collected for each fault model are defined: stuck-at faults (SAFs), transition-delay faults (TDFs), static-CA faults (CAT-STATs), and dynamic-CA faults (CAT-DYNs). It is worth noting that for some designs the cell-aware characterization tool has generated more dynamic-CA faults than static-CA faults.

TABLE II. CONSIDERED ATPG FLOWS RESULTS

Design	*ATPG-FLW1*		*ATPG-FLW2*		*ATPG-FLW3*		*ATPG-FLW4*		*ATPG-FLW5*		*ATPG-FLW6*		*ATPG-FLW7*		*ATPG-FLW8*	
	FC-1 [%]	*PC-1*	*FC-2 [%]*	*PC-2*	*FC-3 [%]*	*PC-3*	*FC-4 [%]*	*PC-4*	*FC-5 [%]*	*PC-5*	*FC-6 [%]*	*PC-6*	*FC-7 [%]*	*PC-7*	*FC-8 [%]*	*PC-8*
B01	98.69%	24	98.69%	26	98.69%	21	98.69%	45	75.12%	18	67.46%	22	67.46%	21	67.46%	36
B02	97.27%	14	97.95%	15	97.95%	15	97.95%	26	88.24%	11	77.65%	13	77.65%	15	77.65%	23
B03	98.82%	38	98.82%	49	98.82%	46	98.82%	74	93.46%	37	59.44%	70	59.44%	65	59.44%	96
B04	98.52%	60	98.52%	79	98.52%	77	98.52%	112	76.01%	54	69.91%	70	69.96%	68	69.91%	99
B05	72.03%	87	71.71%	134	71.52%	165	71.71%	192	66.95%	104	57.24%	147	57.97%	163	57.24%	222
B06	98.03%	17	98.38%	26	98.38%	23	98.38%	40	73.84%	15	61.76%	21	61.76%	18	61.76%	33
B07	97.74%	77	97.74%	114	97.74%	111	97.74%	163	91.55%	102	73.80%	145	73.87%	158	73.80%	200
B08	99.03%	57	99.03%	82	99.03%	77	99.03%	127	88.01%	67	71.84%	92	71.84%	98	71.84%	135
B09	99.03%	50	99.03%	63	99.03%	52	99.03%	89	92.65%	53	74.00%	94	74.04%	90	74.00%	125
B10	98.75%	59	98.75%	73	98.75%	67	98.75%	116	74.57%	52	64.97%	72	64.73%	76	64.97%	101
B11	97.47%	106	97.47%	169	97.47%	213	97.47%	232	89.62%	123	74.89%	202	74.89%	272	74.87%	278
B12	99.14%	165	99.14%	225	99.14%	207	99.14%	311	89.81%	334	77.90%	516	78.10%	546	77.89%	716
B13	97.20%	56	97.20%	87	97.20%	83	97.20%	119	80.10%	78	64.74%	111	64.72%	103	64.74%	150
B14	95.95%	698	96.09%	792	96.09%	795	96.09%	1,207	95.91%	913	87.63%	1,553	87.72%	1,576	87.62%	2,077
B15	97.47%	558	97.41%	1,668	97.39%	1,940	97.41%	1,977	92.17%	709	83.99%	1,601	84.18%	1,550	83.98%	2,028

For instance, in the B14 design the synthesis tool inferred a considerable number of XOR-cells. For this kind of cell, the produced defect matrix has more CAT-DYNs than CAT-STAs. Therefore, the overall number of CAT-DYNs is higher than CAT-STAs.

V. Experimental results

This section discusses the results from the comparison of the various flows for the considered designs, as depicted in TABLE II. , where the fault coverage (FC) and the number of patterns (PC) obtained for each design are presented for each considered flow.

By comparing all the methodologies presented and dividing them by type (SAF/static-CAT and TDF/dynamic-CAT) from the results obtained it can be seen that:

- ATPG-FLW1 and ATPG-FLW5 produce on average up to 50% less patterns than the other approaches and thus resulting in a reduced test application time; they would be the best choice for CAT implementation in the test development flow.
- ATPG-FLW4 and ATPG-FLW8 yield the worst results, due to the higher number of patterns and a lower coverage value regarding the ATPG TDF/dynamic-CAT (ATPG-FLW8).

In particular, from ATPG-FLW1 to ATPG-FLW4 similar fault coverage values were obtained. Especially for B14 and B15, by comparing the number of patterns among the various flows, there is a considerable reduction in pattern count compared to the other flows. For ATPG-FLW4 to ATPG-FLW8, only the first flow gives a higher fault coverage than the others, with a lower number of patterns as well. Overall, it can be observed that ATPG-FLW1 and ATPG-FLW5 produce the best results. At the same time, they have up to 50% more pattern generation run-time than the other flows and up to 30% more patterns to detecting CA faults, unlike the other flows where they have up to 50% more patterns to detect CA faults.

Experiments were done using the tools CMGen (for cell-aware fault model generation) and TestMAX™ ATPG (for pattern generation and fault simulation) by Synopsys.

VI. Conclusions

In this paper, a comparative overview of different test flows for CAT implementation was conducted resorting to a set of open-source benchmarks. The advantages and disadvantages related to the use of CAT methodology considering different test flows were shown. The results obtained showed that the flows ATPG-FLW1 and ATPG-FLW5 (i.e., incremental CA ATPG starting from a set of patterns for traditional fault models) are the most suitable in terms of achievable fault coverage and number of total patterns. Future work will involve the adoption of such approaches within more complex designs.

References

[1] F. Hapke et al., "Defect-Oriented Test: Effectiveness in High Volume Manufacturing," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 40, no. 3, pp. 584-597, March 2021.

[2] H. M. Eissa and A. ElRahman ElMously, "Physical aware design methodology for analog & mixed signal integrated circuits," 2009 4th International Design and Test Workshop (IDT), 2009, pp. 1-5.

[3] F. Hapke et al., "Cell-Aware Test," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 33, no. 9, pp. 1396-1409, Sept. 2014.

[4] F. Hapke and J. Schloeffel, "Introduction to the defect-oriented cell-aware test methodology for significant reduction of DPPM rates," 2012 17th IEEE European Test Symposium (ETS), 2012, pp. 1-6.

[5] K. S. Das and A. Zala, "Optimizing cell-aware ATPG pattern volume to keep test cost competitive," 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), 2020, pp. 1-6.

[6] Z. Gao et al., "Optimization of Cell-Aware ATPG Results by Manipulating Library Cells' Defect Detection Matrices," 2019 IEEE International Test Conference in Asia (ITC-Asia), 2019, pp. 91-96K.

[7] F. Yang, S. Chakravarty, A. Gunda, N. Wu and J. Ning, "Silicon Evaluation of Cell-Aware ATPG Tests and Small Delay Tests," 2014 IEEE 23rd Asian Test Symposium, 2014, pp. 101-106.

[8] A. Prabhu, V. Vorisek, H. Lang and T. Schumann, "Analysis of cell-aware test pattern effectiveness — A case study using a 32-bit automotive microcontroller," 2014 19th IEEE European Test Symposium (ETS), 2014, pp. 1-2.

[9] D. -Z. Lee, Y. -Y. Chen, K. -C. Wu and M. C. . -T. Chao, "Improving Cell-Aware Test for Intra-Cell Short Defects," 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2022, pp. 436-441.

[10] F. Hapke et al., "Defect-oriented cell-aware ATPG and fault simulation for industrial cell libraries and designs," 2009 International Test Conference, 2009, pp. 1-10.

[11] P. -Y. Chuang, C. -W. Wu and H. H. Chen, "Cell-aware test generation time reduction by using switch-level ATPG," 2017 International Test Conference in Asia (ITC-Asia), 2017, pp. 27-32.

[12] H. H. Chen, S. Y. -H. Chen, P. -Y. Chuang and C. -W. Wu, "Efficient Cell-Aware Fault Modeling by Switch-Level Test Generation," 2016 IEEE 25th Asian Test Symposium (ATS), 2016, pp. 197-202.

[13] F. Hapke et al., "Defect-Oriented Test: Effectiveness in High Volume Manufacturing," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 40, no. 3, pp. 584-597, March 2021.

[14] F. Corno, M. Sonza Reorda and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results," in IEEE Design & Test of Computers, vol. 17, no. 3, pp. 44-53, July-Sept. 2000.