Targeting different defect-oriented fault models in IC testing: an experimental approach

Targeting different defect-oriented fault models in IC testing: an experimental approach

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*— In the field of integrated circuit (IC) testing, the detection of defects is crucial to ensure the reliability and functionality of the final product. Among the variety of fault models that can be used to target the many possible defects in a circuit, delay faults (transition and path delay) have been used for many years. Lately, cell-aware testing (CAT) has been introduced as a different approach that aims to improve the detection of internal defects of standard cells: it involves using specific patterns to detect faults that could not be detected by common fault models (e.g., stuck-at and transition delay fault models). Both delay and cell-aware faults can be caused by several factors, such as manufacturing defects, environmental conditions, and aging effects. In this paper, we investigate the application of test patterns generated with the transition and path delay fault models in comparison with others developed with the cell-aware approach, in terms of fault coverage, pattern count and test generation time. Overall, the study shows that the combination of the path delay fault model and cell-aware testing can lead to improved fault coverage and lower test. The experimental results are presented over a wide range of open-source benchmarks and on a RISC-V design using a proprietary industrial technology library.**

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

I. INTRODUCTION

The continuous advancement of integrated circuits (ICs) has revolutionized the modern world, enabling the development of sophisticated electronic systems. Among various IC types, Application-Specific Integrated Circuits (ASICs) have gained prominence due to their customized functionality and high-performance capabilities. However, the escalating complexity of ASIC designs poses significant challenges in ensuring their dependable and error-free operation. Consequently, thorough testing of ASICs becomes indispensable to identify potential faults that could undermine their functionality. Delay fault models [1] can detect potential delay defects into designs when a transition signal (slow-to-rise and rise-to-fall) is run into a specific gate and exceeds the clock period. In particular, one of the aspects of ASIC testing is the accurate recognition and detection of path delay faults [2], as it directly impacts the overall system performance. Path delay faults [3][4] are particularly relevant in digital circuits, as they can lead to timing violations, signal integrity issues, and, ultimately, functional failures. The path delay fault model focuses on testing, identifying, and analyzing the delays encountered by signals as they traverse various paths within a circuit. By considering the individual delays at each stage and the cumulative effect, the path delay fault model provides valuable insights into potential timing issues.

Among the fault models that can detect physical defects in a circuit, cell-aware testing (CAT) can model several intra-cell defects [5][6][7][8]. The cell-aware faults, if detected, can increase the overall quality of the manufacturing process by reducing the number of test escapes. In this work we evaluate the approach of cell-aware testing on path delay faults, to enhance the effectiveness of path delay fault detection.

This paper aims to give a comparative evaluation of the effectiveness of test stimuli generated targeting one fault model out of those discussed before when they are evaluated with respect to a different fault model. In this way we pave the way towards a more effective test pattern generation in terms of achieved defect coverage and pattern count.

Through an in-depth examination of the path delay fault model and the potential of cell-aware testing the methodology of the test may be optimized for obtaining efficient defect coverage in ICs. In fact, CAT, if properly used, can improve fault coverage and pinpoint specific faults associated with path delays. By directly targeting the cells that contribute significantly to path delays, cell-aware testing offers a more refined and efficient testing methodology. Linking the cell-aware testing pattern set with the path delay fault model pattern set we ensure a comprehensive test coverage for critical timing issues. This integration allows designers and test engineers to focus their efforts on specific areas of concern, optimizing test resources and reducing overall test time. By addressing path delays, researchers and industry professionals can strive towards the production of ASICs with improved quality and performance.

This paper is organized as follows: Section II briefly introduces some background on delay and cell-aware testing; Section III describes in detail the methodology and the environment used for this study; Section IV presents and describes the case studies that were used to evaluate the efficacy of each fault model presented in Section II; Section V reports the obtained results. Section VI draws some conclusions.

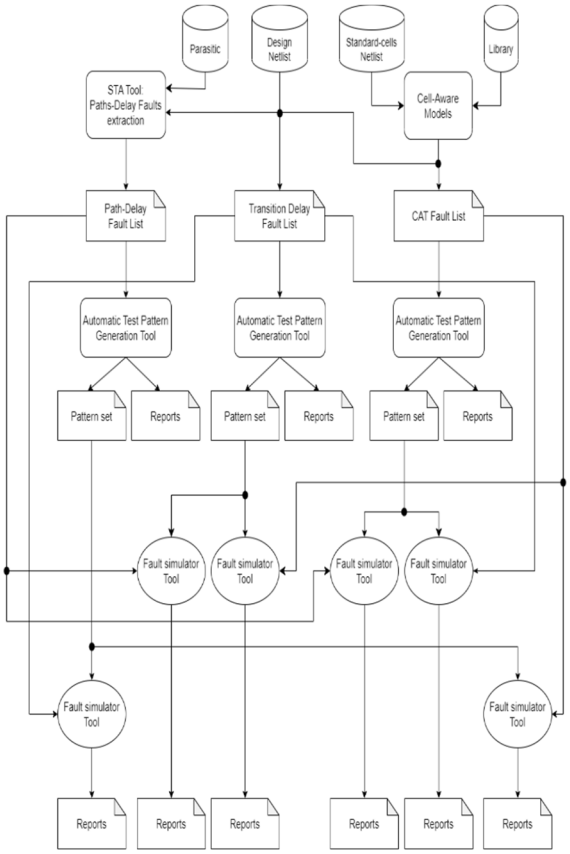

Fig. 1. Methodology flow

II. BACKGROUND

A. Delay fault models

Common fault models, such as stuck-at, assume a permanent change in the data over time, so they can be defined as infinite delay faults. On the contrary, fault models based on the timing delay can be defined, and there may be *transition delay faults (TDFs)*, if we refer to each gate of the circuit, and *path delay faults (PDFs)*, if we refer to each path of the circuit.

Considering the most common fault models, the transition delay fault model may be considered as a special case of the stuck-at fault (SAF) model, in which when a gate of a cell is considered, it fails when a transition is performed in the cell delaying the transition on its output. To test a TDF, a stimulus (slow-to-rise and rise-to-fall vector) must be applied to a circuit input (input port or scan flip-flop), propagated through the target cell and then to an observable output. The corresponding faults can be caused by numerous factors such as manufacturing defects, aging, and environmental factors. The fault detected in the cell in a circuit can cause a different behavior than expected, leading to incorrect results or circuit malfunction.

While the transition delay fault model focuses on the delay of a specific gate, the path delay fault model captures small extra delays that have cumulative effect along a path that may result in faulty behavior of the circuit. A path delay fault is activated when the delay along a single path exceeds the clock period of the circuit and hence may produce a failure. Typically, each path begins at a primary input or at the output of a flip-flop and ends at a primary output or at the input of a flip-flop.

B. Cell-aware fault model

Cell-aware testing aims at detecting intra-cell faults in standard cells. It covers an internal range of defects in a physical device such as short circuits, open circuits and transistor defects that could not be covered by common fault models.

The CAT methodology consists of a preliminary phase aiming at the cell-aware fault list generation. This one-time task considers each cell of the technology library. In this phase the characterization of each cell in the technology library is performed, based on its electrical-level representation. Once the characterization tool has extracted from each cell structure (i.e., a transistor-level netlist with parasitic information given by layout data) a list of possible faults, an analog simulation is performed to determine the complete set of cell input combinations that detect each intra-cell defect, assuming that all input and output cell signals are fully controllable and observable. Lastly, a comparative analog fault simulation is performed between the fault-free netlist and the faulty one to validate which fault is considered as detected when the voltage signal output behavior of the faulty netlist deviates by more than a certain threshold from the fault-free one.

The cell-aware faults can be labelled as *static* when a single input vector is required to detect the fault, while cell-aware faults are labeled as *dynamic* when a pair of vectors is required to detect them.

The result of this process is a defect matrix for each library cell. Each defect matrix contains the input values detecting a particular defect within the cell 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 [9]. For the purpose of this work, only dynamic CAT faults are considered, due to their similarities with the other fault models analyzed.

The outcome of the process described previously is then integrated in the standard test flow where an ATPG tool uses either all the traditional and CAT fault models or only one fault model to cover all the defects inside an IC [10], generating then the scan vectors able to detect all the considered faults of the fault list (CAT and traditional, only CAT, or only traditional).

TABLE I. DESIGN DETAILS

Designs	*Flip-Flops*	*Logic Gates*	*PIs*	*POs*	*PDFs*	*TDFs*	*DYN-CATs*
B01	5	29	4	2	37	230	397
B02	4	14	3	1	27	124	216
B03	30	62	6	4	682	678	1,721
B04	66	146	13	8	932	1,656	4,448
B05	34	265	3	36	1,342	2,194	4,986
B06	9	26	4	6	104	254	392
B07	45	151	3	8	1,752	1,480	4,158
B08	21	78	11	4	345	716	1,528
B09	28	74	3	1	749	720	1,838
B10	17	83	13	6	468	734	1,350
B11	30	164	9	6	1,218	1,544	4,653
B12	121	469	7	6	2,954	4,282	9,844
B13	51	141	12	10	618	1,392	2,854
B14	215	2,977	34	54	10,064	26,004	87,402
B15	417	2,560	38	70	20,551	24,802	72,757
B17	1,316	9,208	42	98	7,895	81,656	222,566
B18	3,062	30,805	41	24	18,363	237,320	735,970
B19	6,126	62,007	50	31	30,000	477,616	1,486,346
B20	430	7,060	37	23	2,571	55,556	216,062
B21	430	7,159	37	23	2,571	55,772	214,248
B22	645	10,694	37	23	3,857	83,696	324,465
RISC-V	2,329	29,036	222	268	5,054	195,812	376,720

III. METHODOLOGY

This section presents the methodology used for the analysis for each fault model considered in this work. The chart in Fig.1 represents the summary of the considered flow. The flow is composed of three parts, depending on which fault model is considered. Starting from the left-top side of the figure, the static-time analysis (STA) tool extracts the path delay faults of the design (the critical and less critical ones in terms of the length of each path and minimum timing slack between the starting and ending point, considering fixed the period of the clock), hence they are included in a path delay fault list and used in the ATPG tool.

The path delay faults are extracted considering all the slack ranges in which the design works, where the clock period is previously defined. Once the list is generated and imported into the ATPG tool, the pattern set is created to detect all possible path delay faults. At the end of this flow, the final reports are provided containing fault coverage and pattern count information.

The same netlist is used to run an ATPG to generate stimuli detecting as many as possible transition delay faults in the design where the reports and the pattern set containing all the generated patterns are collected to be used in the second phase of the flow.

In the right-top side of the figure, it is depicted the CAT flow. The flow starts with the characterization of each cell in the library, using the layout spice netlist of each standard-cell and their own electrical-level models. Then, the cell-aware fault list containing all the *static* and *dynamic* CAT faults of the circuit is collected by the ATPG tool through which the final reports and the pattern set file are collected.

Considering the bottom side of the figure, fault simulation is used to analyze the coverage obtained using the patterns generated targeting the different fault models when considering the fault lists of the other fault models. In particular, using the path delay fault list, two fault simulations are performed resorting to the transition delay pattern set for the first one, and dynamic CAT pattern set for the second one. The same for the other fault models: this means that, first for the transition delay patterns and then for the dynamic CAT ones, two additional fault simulations are performed in each case.

The purpose of using these mixed approaches is to evaluate the fault coverage obtained by the pattern set generated targeting different fault models. In this way we will show that in some cases the patterns generated for a different fault model may achieve relevant fault coverage figures also for the other ones.

IV. CASE STUDIES

The designs used in this work include the Open-source benchmarks presented in [11] that were synthesized with STMicroelectronics' proprietary 130-nm HCMOS technology for power applications. We also considered the PULPino 32-bit RISC-V by ETH Zurich [12]. All the standard-cell characterizations were performed by a commercial cell-aware tool. All the experiments were performed resorting to a commercial ATPG tool handling the most common fault models as well as CAT. For the extraction of the paths, a STA commercial tool was used. For the experiments, a single scan chain was used for each design. Considering other works that used open-source design to analyze path delay faults [13][14], the choice to use simple scan chains stems from the purpose of this work, i.e., targeting different defect-oriented fault models to observe the testing behavior through simple open architectures.

In TABLE I the details of the case study circuits are collected. Each row reports the circuit identifier, the number of sequential cells (flip-flops) and combinational cells (logic gates), the number of primary inputs and outputs (PIs and POs), the number of path delay faults (PDFs), the number of transition delay faults (TDFs) and dynamic CAT faults (DYN-CATs).

The number of extracted PDFs is a result of all detected paths by the ATPG tool, excluding invalid and untestable paths. In the following section the results obtained from the complete flow are presented.

V. EXPERIMENTAL RESULTS

In this section the results from the comparison of each fault model in terms of fault coverage and pattern count are reported and discussed, with respect to the effectiveness that the patterns achieved targeting every considered fault model show, with respect to the fault lists of the other fault models. Test generation time is considered negligible due to the fact that each flow requires no more than 10 minutes to be processed on a server with 16 CPU cores Intel® Xeon e-2000 series and 40Gb of RAM.

TABLE II. EFFECTS OF PATTERNS TARGETING PATH DELAY FAULTS ON DIFFERENT FAULT MODELS

Designs		Fault Coverage			Pattern Count
	PDFs	FC_{PDF}^{PDF}	FC_{PDF}^{CAT}	FC_{PDF}^{TDF}	PC^{PDF}
B01	37	54.05%	54.05%	**59.46%**	12
B02	27	**51.85%**	40.74%	**51.85%**	7
B03	682	**53.08%**	28.01%	29.77%	57
B04	932	**43.45%**	27.04%	22.42%	60
B05	1,342	**18.85%**	12.37%	11.70%	73
B06	104	**33.65%**	**33.65%**	**33.65%**	9
B07	1,752	**15.47%**	14.50%	13.81%	48
B08	345	**27.54%**	18.26%	23.48%	27
B09	749	12.42%	**18.42%**	16.69%	23
B10	468	21.58%	20.94%	**21.79%**	20
B11	1,218	**7.06%**	5.67%	4.68%	25
B12	2,954	**33.07%**	20.07%	22.51%	292
B13	618	**39.48%**	31.07%	33.82%	45
B14	10,064	**7.24%**	3.34%	3.44%	232
B15	20,551	**3.15%**	2.40%	1.89%	217
B17	7,895	**34.05%**	17.18%	16.81%	405
B18	18,363	**26.60%**	13.17%	13.17%	424
B19	30,000	**31.90%**	15.01%	13.94%	536
B20	2,571	**9.37%**	1.71%	1.52%	111
B21	2,571	**7.90%**	1.94%	2.14%	86
B22	3,857	**8.06%**	2.46%	2.00%	145
RISC-V	5,054	**12.96%**	0%	0%	129

TABLE III. EFFECTS OF PATTERNS TARGETING DYNAMIC CAT FAULTS ON DIFFERENT FAULT MODELS

Designs		Fault Coverage			Pattern Count
	DYN-CATs	FC_{CAT}^{CAT}	FC_{CAT}^{PDF}	FC_{CAT}^{TDF}	PC^{CAT}
B01	397	79.85%	**90.68%**	76.32%	20
B02	216	92.13%	87.04%	**93.06%**	13
B03	1,721	94.13%	**94.60%**	94.36%	46
B04	4,448	80.64%	**89.95%**	77.90%	56
B05	4,986	**69.53%**	57.10%	68.75%	109
B06	392	**81.38%**	79.34%	76.02%	15
B07	4,158	**94.13%**	83.31%	91.75%	105
B08	1,528	90.31%	72.05%	**90.90%**	62
B09	1,838	**93.96%**	90.86%	92.38%	58
B10	1,350	**79.41%**	75.04%	75.19%	49
B11	4,653	91.51%	71.33%	**91.66%**	116
B12	9,844	**90.61%**	86.64%	90.09%	347
B13	2,854	80.87%	76.94%	**83.50%**	60
B14	87,402	**96.39%**	54.91%	94.17%	853
B15	72,757	**94.64%**	50.24%	94.14%	747
B17	222,566	91.88%	60.85%	**92.45%**	1,108
B18	735,970	**83.34%**	55.71%	83.15%	1,371
B19	1,486,346	**82.31%**	55.80%	82.15%	1,530
B20	216,062	**97.51%**	28.70%	94.31%	1,565
B21	214,248	**97.54%**	19.94%	94.18%	1,569
B22	324,465	**97.64%**	39.74%	94.87%	1,677
RISC-V	376,720	**96.11%**	17.28%	94.99%	2,282

In TABLE II the data regarding the patterns generated targeting the path delay fault model when simulated with respect to the different fault lists are shown both for the ITC'99 benchmark and the RISC-V designs. We denoted with FC_j^i the fault coverage figure obtained with the test patterns generated by targeting the fault model i with respect to the fault list for fault model j. For example, FC_{PDF}^{TDF} is the fault coverage obtained by fault simulating the patterns generated targeting the TDFs with respect to the fault list of PDFs.

Starting from the left side of the table, in the first column all the considered designs are listed; the number of path delay faults are included in the second column. The other three columns represent the fault coverage figures achieved by the patterns targeting the 3 fault models: the first one relates to the patterns generated by the ATPG targeting the path delay fault model (FC_{PDF}^{PDF}); the second one is obtained by the fault simulation of the path delay fault list with the pattern set obtained by an ATPG flow targeting the dynamic CAT faults (FC_{PDF}^{CAT}); the third one is obtained by the fault simulation of the path delay fault list with the transition delay pattern set (FC_{PDF}^{TDF}).

The last column shows the pattern count (PC) for the considered fault model which, for the sake of showing all the data listed, is included in each table of this paper and in order: PC^{PDF} is the number of patterns that refer to FC_{PDF}^{PDF}; PC^{CAT} refers to FC_{CAT}^{CAT}; PC^{TDF} refers to FC_{TDF}^{TDF}. It can be noted that FC_{PDF}^{PDF} has the overall best results in terms of coverage and pattern count, except for B01, B09 and B10 design, due to the fact that the ATPG tool, after the pattern generation, classified many faults as undetectable and ATPG untestable, hence decreasing the fault coverage of the design; whereas FC_{PDF}^{CAT} has the best results in B09 but using more patterns with respect to the pattern sets for the other fault models (analogous situation for B06 but with the same coverage for all the fault models but with more patterns used). FC_{PDF}^{TDF} has the best results in B01 and B10 where the reason may be inferred from the number of patterns Involved in the fault simulation.

The data regarding RISC-V require particular attention. In this case only the ATPG flow FC_{PDF}^{PDF} has the best performances, able to detect a very low set of the considered path delay faults in the design. With the other considered fault models used for the comparison with the path delay fault list, even though the pattern count is much higher than the PC^{PDF}, the results are 0% for both considered fault coverages, not allowing any path delay fault detection. From the analysis done for this work, only the specific path delay ATPG flow can detect some paths, which still are very few because most of them are classified either undetectable or ATPG untestable by the ATPG tool.

TABLE III shows the results obtained by using the dynamic CAT fault list. As shown in the previous table, the designs and the number of faults are included in the first and second column from the left. The fault coverages (the other three columns) show the data collected by the ATPG and the fault simulator tool, according with the methodology explained in the previous section. In the last column, the number of patterns for the respective fault model is shown. It can be noticed that FC_{CAT}^{PDF} has the worst overall fault coverage excluding B01, B03 and B04 that do not have a large number of gates and they may be considered negligible with respect to the other circuits.

In those cases, the PDFs pattern set has been capable of detecting most of the dynamic CAT faults of the design with a comparable number of patterns implemented in the fault simulator. FC_{CAT}^{TDF} has the best

TABLE IV. EFFECT OF PATTERNS TARGETING TRANSITION DELAY FAULTS ON DIFFERENT FAULT MODELS

Designs			Fault Coverage		Pattern Count
	TDFs	FC_{TDF}^{TDF}	FC_{TDF}^{PDF}	FC_{TDF}^{CAT}	PC^{TDF}
B01	230	79.85%	**80.00%**	70.43%	16
B02	124	**92.13%**	77.42%	77.42%	11
B03	678	**94.13%**	87.61%	86.73%	32
B04	1,656	**80.64%**	68.06%	60.00%	30
B05	2,194	**69.53%**	47.93%	58.85%	78
B06	254	**81.38%**	67.32%	64.17%	12
B07	1,480	**94.13%**	65.99%	81.23%	62
B08	716	**90.31%**	57.40%	74.44%	54
B09	720	**93.96%**	82.92%	82.50%	33
B10	734	**79.41%**	62.94%	66.89%	32
B11	1,544	**91.51%**	53.70%	76.43%	96
B12	4,282	**90.61%**	76.09%	82.44%	263
B13	1,392	**80.87%**	63.51%	66.16%	61
B14	26,004	**96.39%**	44.17%	88.68%	722
B15	24,802	**94.64%**	34.26%	80.56%	589
B17	81,656	**86.05%**	41.08%	80.55%	979
B18	237,320	**79.20%**	38.47%	76.03%	1,121
B19	477,616	**78.89%**	39.67%	75.44%	1,164
B20	55,556	**93.33%**	21.40%	91.93%	986
B21	55,772	**93.99%**	12.42%	92.62%	1,022
B22	83,696	**93.61%**	30.17%	92.51%	1,080
RISC-V	195,812	**95.70%**	13.93%	14.36%	2,205

performance in B02, B08, B11, B13 and B17, but also in these cases the increased number of patterns used (especially for the bigger designs) affects the test time.

The data regarding RISC-V show a slightly different situation: FC_{CAT}^{PDF} has a very low fault coverage because of the number of patterns (PC^{PDF}) that are capable to cover most of the faults in the design. The other fault models have comparable performances.

TABLE IV shows the data gathered from the transition delay fault model. The design list and the transition delay faults (TDFs) are included in the first and second column. FC_{TDF}^{TDF} is the fault coverage obtained by the TDF-oriented ATPG flow; FC_{TDF}^{PDF} and FC_{TDF}^{CAT} are the fault coverage values obtained by the respective fault simulations and the pattern count is shown in the last column of the table. Also in this case, FC_{TDF}^{PDF} has the worst performance in terms of fault coverage, considering the difference between the pattern count of each considered fault model, excluding B01 where the number of patterns used is higher than the transition delay fault model. FC_{TDF}^{TDF} has the best performance of all the designs.

The data regarding RISC-V shows that: FC_{TDF}^{PDF} has low fault coverage because of the low number in PC^{PDF} as already shown and it is not capable to cover most of the faults in the design. Similar conditions hold true for FC_{TDF}^{CAT} , where most of the faults of the design were classified as "not detected" by the fault simulator.

Overall, considering the benchmark designs and the data from the previous tables, it can be noticed that:

- The patterns generated targeting the *path delay* fault model, compared with the other ones, have the best performances when fault simulated on their own fault lists, excluding some cases for the small designs.
- The patterns generated targeting the *dynamic* CAT fault model have the best performance when a CAT ATPG is performed, and a fault simulation is run (FC_{CAT}^{TDF}); the path delay pattern set has the worst performance in terms of fault coverage.
- The patterns generated targeting the *transition delay* fault model used with the TDF ATPG tool has the best performances in terms of coverage, followed by the fault simulation for the other ones.

Considering the overall data for RISC-V design, from the previous tables, it can be observed that:

- The patterns generated targeting the *path delay* fault model have the best performance when a PDF ATPG run is performed (FC_{PDF}^{PDF}); the same patterns cannot detect any fault considering the other models.
- Considering the patterns generated targeting the *dynamic* CAT fault model, they have the best performance when the respective fault list is used in ATPG (FC_{CAT}^{CAT}); whereas, from the comparison with the other fault models, FC_{CAT}^{PDF} has the worst performance, while FC_{CAT}^{TDF} have similar results.
- The patterns generated targeting the *transition delay* fault model have the best performance only resorting to an TDF ATPG run (FC_{TDF}^{TDF}), while for the other fault models the achieved fault coverages figures are not comparable with the first one.

VI. CONCLUSIONS

In this paper, different fault models have been used to generate different sets of defect-oriented test patterns. We then evaluated the fault coverages obtained by running the fault simulation of patterns generated for a different fault model.

The obtained results showed that the behavior of the different circuits is rather different. Clearly, the FC obtained using the ATPG tool targeting a specific fault model is generally higher than the one obtained by using patterns generated for other fault models. However, there are cases where the patterns generated for a given fault model achieve very good fault coverage figures even when simulated with different fault lists. In some cases, as shown for CAT faults in TABLE III, it can be observed that pattern sets generated targeting one fault model (TDFs) may be useful to detect other faults as well, possibly increasing the achieved fault coverage. This observation may pave the way to develop optimized test generation strategies, able to reduce the test generation time and (most importantly) to reduce the number of required test patterns while still achieving the same fault coverage.

VII. REFERENCES

[1] Z. Barzilai and B. Rosen, “Comparison of AC self-testing procedures,” in Proc. Int. Test Conf., 1983.

[2] P. Varma, “On test generation for path delay faults in ASICs,” in Proc. VLSI Test Symp., 1992.

[3] G. L. Smith, “Model for delay faults based upon paths,” in Proc. Int. Test Conf., 1985.

[4] M. H. Schulz, K. Fuchs and F. Fink, "Advanced automatic test pattern generation techniques for path delay faults," [1989] The Nineteenth International Symposium on Fault-Tolerant Computing. Digest of Papers, Chicago, IL, USA, 1989.

[5] 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

[6] J. Sudbrock, J. Raik, R. Ubar, W. Kuzmicz and W. Pleskacz, "Defect-oriented test- and layout-generation for standard-cell ASIC designs," 8th Euromicro Conference on Digital System Design (DSD'05), Porto, Portugal, 2005.

[7] 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.

[8] 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), Annecy, France, 2012

[9] N. Mirabella, A. Floridia, R. Cantoro, M. Grosso and M. Sonza Reorda, "A comparative overview of ATPG flows targeting traditional and cell-aware fault models," 2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Glasgow, United Kingdom, 2022

[10] 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.

[11] 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.

[12] M. Gautschi et al., "Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 10, pp. 2700-2713, Oct. 2017.

[13] W. Qiu and D. M. H. Walker, "Testing the path delay faults of ISCAS85 circuit c6288," Proceedings. 4th International Workshop on Microprocessor Test and Verification - Common Challenges and Solutions, Austin, TX, USA, 2003, pp. 19-24

[14] I. Pomeranz, "Path Unselection for Path Delay Fault Test Generation," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 31, no. 2, pp. 267-275, Feb. 2023