A comprehensive framework for cell-aware diagnosis of customer returns

Pierre D'hondt, Safa Mhamdi, Patrick Girard, Arnaud Virazel, Aymen Ladhar

Microelectronics Reliability xxx (xxxx) 114595

Contents lists available at ScienceDirect

Microelectronics Reliability

journal homepage: www.elsevier.com/locate/microrel

A comprehensive framework for cell-aware diagnosis of customer returns

P. d'Hondt [a, b], S. Mhamdi [b], P. Girard [b, *], A. Virazel [b], A. Ladhar [a]

[a] STMicroelectronics, Crolles, France
[b] LIRMM – Univ. of Montpellier/CNRS, Montpellier, France

ARTICLE INFO

Keywords:
Diagnosis
Customer returns
Bayesian classification

ABSTRACT

The first step when a customer return occurs in industry is to reproduce the failure mechanism with any original test and test conditions (voltage, temperature, etc.) used during initial post-fabrication test. Depending on the test scenario (test sequence, test scheme, test conditions), different types of defects and failure mechanisms are targeted, thus leading to different diagnosis outcomes. Based on our previous work, this paper presents a comprehensible framework for cell-aware diagnosis of customer returns. The first part of the paper summarizes our previous work in which a generic cell-aware diagnosis flow using a Bayesian classification method was proposed to precisely identify defect candidates in combinational cells of a defective circuit according to a given test scenario. The second part of the paper extends the previous work by dealing with defects in sequential cells of defective circuits. New experiments done on a silicon test chip and a customer return with a given test scenario have proven the efficacy of our flow in terms of diagnosis accuracy and resolution.

1. Introduction

One technology that has received a lot of attention recently in the field of digital IC design and quality assessment is Cell-aware (CA) test and failure diagnosis [1]. With more defects inside cells at leading-edge technology nodes, CA test, which deterministically targets defect locations inside standard cells, and CA diagnosis, which can identify the location and type of cell-internal defects at the transistor level, are quality assessment solutions widely adopted today in industry [2].

CA diagnosis is valuable in the context of large, complex cells such as adders, multipliers, and multi-bit sequential elements. Even when a defect is known to be within a cell, finding defects in such a complex cell during Physical Failure Analysis (PFA) can be challenging and time-consuming, especially if the defect is exhibited for a specific test condition and undetected for some others. CA diagnosis shortens the lengthy investigation process by pinpointing a small subsection of the suspected cell. It improves results for diagnosis scenarios such as customer return analysis as well as volume diagnosis applications like yield analysis [3].

Customer returns are defective ICs that passed all functional, structural and parametric tests after manufacturing but failed in the field. Industrial experiences show that they are mostly due to latent defects [4]. This is especially true for critical products (e.g., automotive) where a comprehensive test flow has been applied to ensure zero test escape after manufacturing. In this context, the first step when a customer return occurs is to reproduce the failure mechanism with any original test and test conditions. Next, a diagnosis program made of several routines is used to identify, step by step, the failing part and, finally, the suspected defects. Each routine corresponds to the application of a diagnosis algorithm at a given hierarchy level (system, core and cell levels) [5–10].

The quality of a diagnosis outcome is usually evaluated owing to two metrics: accuracy and resolution. A diagnosis is accurate if the actual defect is included in the reported list of candidates. Resolution refers to the total number of candidates reported for each actual defect. An accurate diagnosis with perfect resolution (i.e., one) is the ideal case. Achieving such a perfect resolution with conventional diagnosis approaches based on cause-effect and/or effect-cause analysis [11–13] becomes increasingly difficult. The study of a commercial chip in [14] demonstrates that just over 30 % of the diagnosed defects exhibit ideal resolution.

With the fast development and vast application of Machine Learning (ML) in recent years, ML-based techniques have been shown to be quite valuable for diagnosis, especially in the volume diagnosis scenario [15–26]. Most of these techniques focuses on supervised root-cause analysis, because it naturally aligns with the common practice of training with labeled historical data and usually performs well in industrial diagnosis tasks [27]. However, though efficient, these techniques address volume diagnosis for yield ramp-up, which is a different problem than fault diagnosis of customer returns. Indeed, during volume diagnosis, a lot of data collected during manufacturing test and subsequent diagnosis phases are available, such as, e.g., hundreds of similar failed

[*] Corresponding author.
E-mail addresses: pierre.dhondt@st.com (P. d'Hondt), girard@lirmm.fr (P. Girard), virazel@lirmm.fr (A. Virazel), aymen.ladhar@st.com (A. Ladhar).

https://doi.org/10.1016/j.microrel.2022.114595
Received 20 March 2021; Received in revised form 13 October 2021; Accepted 12 June 2022
0026-2714/© 20XX

chips with candidates correctly labeled (good, bad). It is therefore possible to use these data for failure diagnosis of a new failed chip. Conversely, during fault diagnosis of a customer return, only one failed chip is investigated, with no information about the defective behavior of some other similar chips used in the same conditions (application, environment, workload). For this reason, learning-guided approaches existing for volume diagnosis cannot be reused for diagnosis of customer returns.

In [28–33], we proposed several learning-guided solutions for CA diagnosis of mission mode failures in customer returns. All solutions are based on a Bayesian classification method for accurately identifying defect candidates in combinational standard cells of a defective IC. The selection of one solution over another depends on the considered test scenario (test sequence, test scheme, test conditions), thus leading to different types of targeted defects and failure mechanisms. Experiments on benchmark circuits and industrial customer returns as well as comparison with a commercial CA diagnosis tool have shown the effectiveness of all solutions.

In this paper, we first summarize the work presented in [28–33] and explain how these CA diagnosis solutions are used depending on the failing test scenario. Next, we extend the previous work by dealing with sequential cells and diagnosis of related defects in customer returns. To this end, each cell-pattern for a sequential cell is considered as a tuple in which the first value represents the input clock signal, the second value is associated to the main input of the cell, and the third value is associated to a virtual input pin representing the previous value of the output pin of the cell. Owing to this cell-pattern representation, sequential cells can be handled in the same manner than combinational cells by the learning-based CA diagnosis flow utilized in [28–33] and summarized in Section 3. New experiments done on a silicon test chip and a customer return have proven the efficacy of our flow in terms of diagnosis accuracy and resolution for a given test scenario.

The rest of this paper is organized as follows. Section 2 summarizes our previous work. Section 3 presents the generic CA diagnosis flow used in [28–33] and summarizes the various contributions. Section 4 shows how defects in sequential cells can be dealt with the CA diagnosis flow. Section 5 presents some results obtained on benchmark circuits as well as on a silicon test chip. Section 6 proposes a discussion on how diagnosis efficiency can be improved depending on the test scenario envisaged during customer return diagnosis. Section 7 concludes the paper and draws some perspectives.

2. Summary of previous work

When conducting the diagnosis of a customer return in industry, different test scenarios may be considered, most of the time (or in a first attempt) reproducing the test scenarios used initially during manufacturing test. Fig. 1 gives an example of a multi-run ATPG flow used in industry for screening defects in logic parts of a System-on-Chip (SoC). As can be seen, several ATPG runs are executed at various speeds (low or at-speed) in order to get different test sequences targeting different defect types (static or dynamic cell-internal defects) and fault models (SA – stuck-at, or TF – transition). Static defects are defects that require one-vector test patterns to be detected. Dynamic defects are defects that require two-vector test patterns to be detected. With the advent of nanometric technologies, the occurrence of dynamic defects is constantly increasing, not only during the manufacturing process of ICs, but also during the lifetime of the circuit where latent or wear-out defects may appear due to various stress conditions (operational, environmental, etc.).

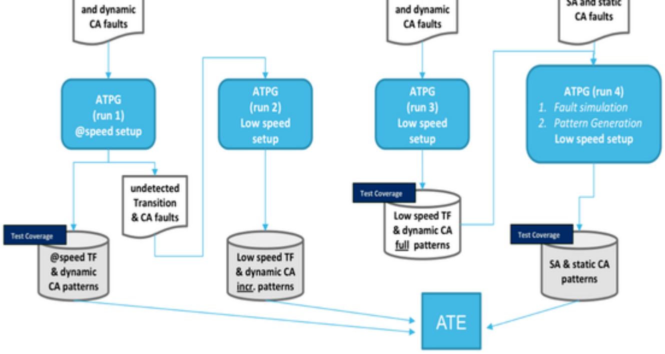

Transition and dynamic CA faults
ATPG (run 1) @speed setup
undetected Transition & CA faults
Test Coverage
@speed TF & dynamic CA patterns
ATPG (run 2) Low speed setup
Low speed TF & dynamic CA incr. patterns
Transition and dynamic CA faults
ATPG (run 3) Low speed setup
Test Coverage
Low speed TF & dynamic CA full patterns
SA and static CA faults
ATPG (run 4)
1. Fault simulation
2. Pattern Generation
Low speed setup
Test Coverage
SA & static CA patterns
ATE

Fig. 1. Example of a multi-run industrial ATPG flow.

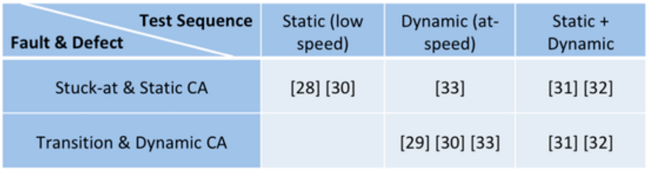
From the ATPG flow illustrated in Fig. 1, three test sequences are obtained and can be applied sequentially to a CUT to achieve a targeted test coverage. During diagnosis of a customer return, the same test sequences can be reused to exhibit the failure observed during mission mode.

In [28–33], we considered various test scenarios for CA diagnosis and described the corresponding diagnosis algorithm implementations. The test scenarios are sketched in Fig. 2. In [28,29], two **distinct** processes were developed to diagnose static and dynamic defects **separately**. In [28], we assumed a basic scan testing scheme for static CA test sequences application, thus targeting stuck-at faults plus static intra-cell defects during diagnosis. In [29], we assumed a fast sequential testing scheme for dynamic CA test sequences application, thus targeting transition faults plus dynamic intra-cell defects during diagnosis. *Note that* [30] *presents a combination of* [28,29] *from a test scenario point of view*. The main limitation of the solutions in [28–30] is the required a priori knowledge of the targeted defects type in the Circuit-Under-Diagnosis (CUD). In other words, a test engineer needs to know what type of defect is screening before choosing between [28] or [29].

In an attempt to deal **concurrently** with all types of defect that may occur in customer returns, without any a priori knowledge of the targeted defect type, we proposed a new implementation of the CA diagnosis flow in [31,32]. *Note that* [32] *is a fully extended version of* [31]. We assumed a test scenario in which **two** test sequences (static and dynamic) are used successively, each one assuming a dedicated testing scheme, i.e., basic scan and fast sequential. First, a static CA test sequence generated by a commercial cell-aware ATPG tool is applied to the CUD. This sequence targets all cell-level stuck-at faults plus cell-internal static defects, considering that these defects are not covered by a standard stuck-at fault ATPG. A standard (low speed) scan-based testing scheme is used to this purpose. Next, another option of the cell-aware ATPG is used to generate a dynamic CA test sequence that targets cell-level transition faults plus intra-cell dynamic defects not covered by a standard transition fault ATPG. In this case, an at-speed Launch-On-Capture (LOC) scheme (also called fast sequential) is used during test application.

Constructing such a comprehensive flow imposed setting up a new framework with specific rules to achieve a high level of effectiveness in terms of diagnosis accuracy and resolution. The proposed method was based on a Gaussian Naive Bayes trained model to predict good defect candidates. The flow was experimented and validated owing to an industrial test chip and a STMicroelectronics customer return.

In [33], we proposed a new version of the CA diagnosis flow assuming a test scenario in which **both** static and dynamic defects can be diagnosed owing to a **single** dynamic CA test sequence applied at-speed. According to the flow depicted in Fig. 1, this scenario may happen



Test Sequence Fault & Defect	Static (low speed)	Dynamic (at-speed)	Static + Dynamic
Stuck-at & Static CA	[28] [30]	[33]	[31] [32]
Transition & Dynamic CA		[29] [30] [33]	[31] [32]

Fig. 2. Test scenarios considered in [28–33].

when such a test sequence has been generated to target transition faults plus cell-internal dynamic defects, and appears to also cover the required percentage of stuck-at faults plus cell-internal static defects (or, more generally, satisfies the test coverage specifications) after the first run of ATPG in Fig. 1. In this case, note that only one (dynamic) datalog is generated after test application and can further be used for diagnosis purpose. Nevertheless, both static and dynamic defects are taken into account in this scenario. As only dynamic instance tables are manipulated, the representation of training and new data is simplified, i.e., a single type of feature vector is used, without loosing information and hence without decreasing the quality of the training and inference phases. Experimental results gathered on a silicon test chip have shown the benefit of using such a new version of the CA diagnosis flow.

3. Learning-based cell-aware diagnosis flow

Fig. 3 is a generic view of the learning-based CA diagnosis flow utilized in [28–33]. It is based on supervised learning that takes a known set of input data and known responses (*labeled data*) used as training data, trains a model, and then implement a classifier based on this model to make predictions (*inferences*) for the response to new data.

After investigating several machine learning algorithms and observing their inference accuracies in [28], a Bayesian classification method has been chosen for the learning and inference phases. So, the first main step of our CA diagnosis flow consists in generating a Naive Bayes (NB) model and to train it by using a training dataset. In this step, training data are used to incrementally improve the model's ability to make inference. Following the k-fold cross-validation methodology, the training dataset is divided into mutually exclusive and equal subsets. For each subset, the model is trained on the union of all other subsets. Some manipulations, such as grouping data by considering equivalent defects or removing data instances of undetectable defects, are also done during this phase. Once training is complete, the performance (accuracy) of the model is evaluated by using a part of the dataset initially set aside (validation dataset) [34]. More details about performance evaluation as done in our framework can be found in [30]. The second main step consists in implementing the NB classifier by using a Gaussian distribution to model the *likelihood* probability functions, and use this classifier to make prediction when a new data instance has to be evaluated. In the next subsections, we detail the various steps of the CA diagnosis flow able to deal with any type of cell-internal defect (i.e., static and dynamic) that may occur in customer returns.

3.1. Generation of training data

Training data are generated for each type of standard cell existing in the CUD during an off-line characterization process done only once for a given cell library. These data are extracted from CA views provided by a commercial CAD tool that contain all characterization results for a given cell type. These results are provided in the form of a fault dictionary containing, for each defect within a cell, the cell input patterns detecting (or not) this defect. An example of training dataset, as used in [28–32] and containing six instances for an arbitrary two-input cell, is shown in Fig. 4. Each instance is associated to a static defect (D_1, D_2, D_3) or a dynamic defect (D_{11}, D_{12}, D_{13}). A 1 (0) indicates that defect D_i is detectable (not detectable) at the output of the cell when the cell-level test pattern P_j is applied at the inputs of the cell. Cell-level test patterns (called *cell-patterns* in the sequel) are static (one input vector - P_1 to P_4 in Fig. 4) or dynamic (two input vectors - P_5 to P_{16} in Fig. 4 in which R (F) indicates a rising (falling) transition at the cell input). For an n-input cell, there exists 2^n static cell-patterns and $2^n.(2^n{-}1)$ dynamic cell-patterns.

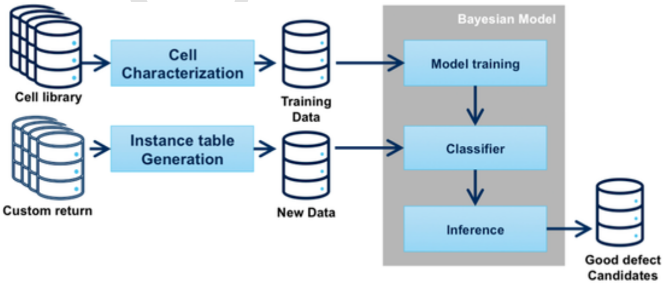
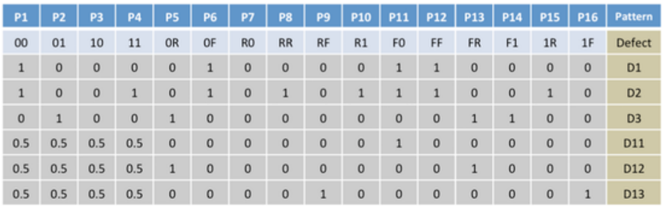


Fig. 3. Generic view of the cell-aware diagnosis flow used in [28–33].

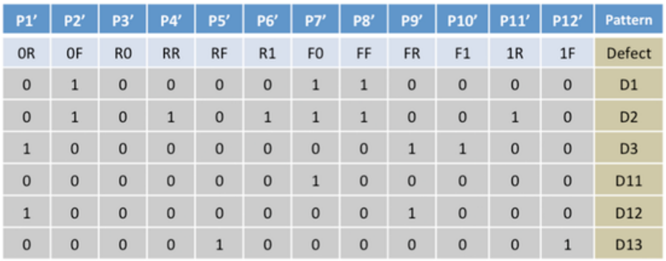
Dynamic defects can be detected not only by dynamic patterns, but also by static patterns applied using a basic scan testing scheme, provided that i) at least one transition has been generated at the cell inputs between the next-to-last scan shift cycle and the launch cycle, and ii) the delay induced by the defect is large enough to be detected (*these are the detection conditions of a dynamic defect modeled by a stuck-open or a gross delay fault*). For this reason, the value ‘0.5’ is assigned to each dynamic defect (D_{11}, D_{12}, D_{13}) for all related static cell-patterns, meaning that such a defect is detectable or not depending on whether or not the above conditions are satisfied.

As only dynamic test sequences are considered in [33], the representation of training data as used in [28–32] could be simplified without losing information and decreasing the quality of the training phase. This comes from the observation that a static defect is a particular case of dynamic defect (e.g., a full open is a resistive open with an infinite value of the resistance), and that all static cell-patterns for a given defect are embedded in its whole set of dynamic cell-patterns. Indeed, a dynamic defect requires a two-vector test pattern (V_1V_2) in which the values of V_1 and V_2 have to be properly defined for the defect to be detected. Conversely, only the value of V_2 is significant for a static defect to be detected by such pattern, irrespective of the value taken by V_1. When looking at Fig. 4, one can see that $P_1 = \{00\}$ is embedded in $P_6 = \{0F\}$, $P_{11} = \{F0\}$ and $P_{12} = \{FF\}$, and the same for P_2, P_3 and P_4. Similarly, we can see that static defect D_2 is detectable by P_1 and P_4, and hence by P_6, P_8, P_{10}, P_{11}, P_{12}, and P_{15}. So, by “compacting” a training dataset as shown in Fig. 5, in which only dynamic cell-patterns are considered, one can see that all meaningful information is still contained in this set, while redundant (‘0’ and ‘1’ values in the first four columns of Fig. 4) or insignificant (‘0.5’ values in the same columns for dynamic defects) information is removed. More generally, such compact format for training data makes so that only one type of feature vector (dynamic) is used for both types of defect.



P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	Pattern
00	01	10	11	0R	0F	R0	RR	RF	R1	F0	FF	FR	F1	1R	1F	Defect
1	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	D1
1	0	0	1	0	1	0	1	0	1	1	1	0	0	1	0	D2
0	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	D3
0.5	0.5	0.5	0.5	0	0	0	0	0	0	1	0	0	0	0	0	D11
0.5	0.5	0.5	0.5	1	0	0	0	0	0	0	0	1	0	0	0	D12
0.5	0.5	0.5	0.5	0	0	0	0	1	0	0	0	0	0	0	1	D13

Fig. 4. Example of training dataset for all defect types (static and dynamic) in a two-input cell as used in [28–32].



P1′	P2′	P3′	P4′	P5′	P6′	P7′	P8′	P9′	P10′	P11′	P12′	Pattern
0R	0F	R0	RR	RF	R1	F0	FF	FR	F1	1R	1F	Defect
0	1	0	0	0	0	1	1	0	0	0	0	D1
0	1	0	1	0	1	1	1	0	0	1	0	D2
1	0	0	0	0	0	0	0	1	1	0	0	D3
0	0	0	0	0	0	1	0	0	0	0	0	D11
1	0	0	0	0	0	0	0	1	0	0	0	D12
0	0	0	0	1	0	0	0	0	0	0	1	D13

Fig. 5. Example of training dataset for all defect types (static and dynamic) in a two-input cell as used in [33].

As the goal with training data is to provide a distinct feature vector for each data (defect), it is important to be able to distinguish between static and dynamic defects with such a new format of the training dataset. Let us consider two defects D_1 and D_{11} where D_1 is static and detectable by {00} and D_{11} is dynamic and detectable by {F0} (note that {00} is the second vector of {F0}). As can be seen in Fig. 5, these two defects can easily be distinguished since their training data instances (or *feature vectors*) are different. The consequence of using such a new format for training data (and hence for new data as will be shown later on) is not an improved accuracy or resolution, but rather a simplified manipulation of feature vectors.

3.2. Generation of instance tables

An instance table is a failure mapping file generated for each suspected cell by using information contained in the tester datalog. It describes the behavior (pass/fail) of the cell for each cell-pattern occurring on its inputs during test of the CUD. The generation process of instance tables is sketched in Fig. 6. First, CA test patterns are applied to the CUD. These test patterns are obtained from a commercial CA test pattern generation tool that targets intra-cell defects. Next, a datalog containing information on the failing test patterns and corresponding failing primary outputs is obtained. From this datalog and the circuit netlist, a logic diagnosis is carried out (still using a commercial tool) and gives the list of suspected cells. From this list and the datalog information, we can finally generate an instance table for each suspected cell. Note that in case several test sequences, e.g., one static and one dynamic, are used for diagnosis of the CUD, the generation process is repeated so as to produce static and dynamic instance tables for all suspected cells. This is the case in [31,32].

The format of a static instance table is illustrated in Fig. 7 for a given two-input NOR cell and two static cell-patterns. In this example, the first part of the file gives information on how the cell is linked to other cells in the circuit, while the second part represents, respectively, the pattern number, the pattern status (failing, passing), and the cell output Z with the associated fault model for which exercising conditions are reported. These conditions shown right below each cell-pattern in Fig. 7 represent the stimulus arriving at the cell inputs during the shift phase (before '-') and applied during the launch cycle (after '-'). For example, cell-pattern 2 consists in applying a 1 on input A and B, and failing thus detecting a stuck-at 1 on Z.

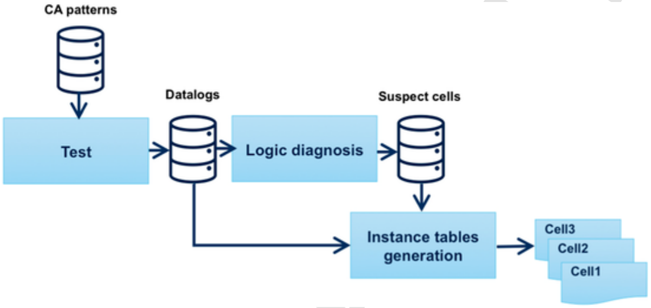
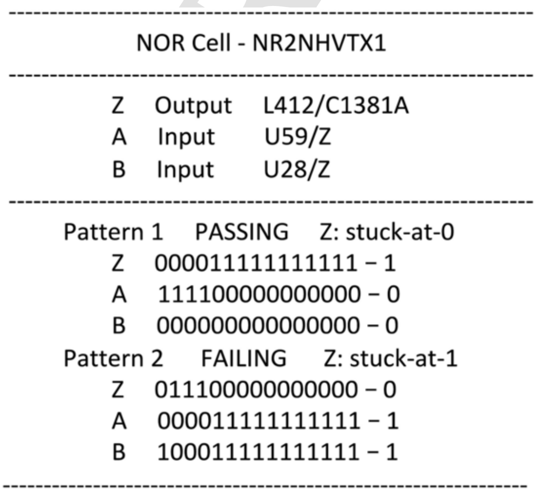


Fig. 6. Generation flow of instance tables.



```
-------------------------------------------------------------
              NOR Cell - NR2NHVTX1
-------------------------------------------------------------
          Z   Output    L412/C1381A
          A   Input     U59/Z
          B   Input     U28/Z
-------------------------------------------------------------
      Pattern 1    PASSING    Z: stuck-at-0
          Z   000011111111111 – 1
          A   111100000000000 – 0
          B   000000000000000 – 0
      Pattern 2    FAILING    Z: stuck-at-1
          Z   011100000000000 – 0
          A   000011111111111 – 1
          B   100011111111111 – 1
-------------------------------------------------------------
```

Fig. 7. Example of static and dynamic instance tables.

3.3. Generation of new data

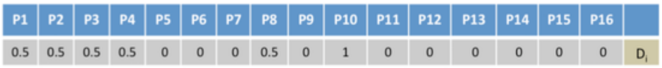
New data are generated after post-processing of instance tables. They are composed of various instances, each of them being associated to one suspected cell in the CUD, and represent a feature vector that characterizes the real behavior of the cell during test application. From each new data instance, we can extract one or more defect candidates that have to be classified as good or bad candidate with a corresponding probability to be the root cause of failure. This classification is done by comparing the new data instance with the training data of the corresponding suspected cell, and identify those training data instances that match (or not) with the new data instance.

The formats of a new data instance as used in [28–32] and [33] are illustrated in Figs. 8 and 9 respectively. This format is quite close to the format of a training data instance, but has a different meaning. In each instance, the value '1' (resp. '0') is associated to a failing (resp. passing) cell-pattern P_i for a given defect candidate, meaning that the candidate is **indeed** detectable (resp. undetectable) by the cell-pattern P_i at the output of the cell during test of the CUD, and hence can (cannot) be the real defect. In such instance, the value '0.5' is associated to a cell-pattern for a given defect candidate when this pattern cannot appear at the inputs of a suspected cell during real test application with an ATE. The median value '0.5' was chosen to avoid missing information in new data instances while not biasing the features of these data.

4. Diagnosis of defects in sequential cells

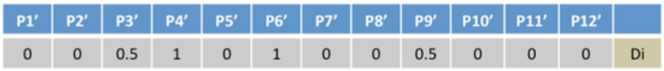
All the work carried out in [28–33] was about diagnosis of defects occurring in combinational standard cells of a customer returns. However, defects in SoCs may also occur in sequential standard cells of logic blocks. In this section, we show how the previous diagnosis flow can handle sequential cells and related defects by adding new information to the training dataset.

The two main differences between combinational cells and sequential cells are that i) the latter have a clock input pin and ii) the fact that the previous logic value of a sequential cell output can affect the current output value of the cell. To take this difference into account, each cell-pattern for a sequential cell is considered as a tuple in which the first value represents the input clock signal (pulsing or not), the second value is associated to the main input of the cell (e.g., D), and the third value is associated to a virtual input pin representing the previous value of the output pin of the cell (e.g., Q). *Note that in case of sequential cells with multiple real inputs* (e.g., *D flip-flop with a D, Scan-In, Scan-Enable and Clock input signals*), *the cell-pattern representation is extended accordingly*. In each tuple, the first value is either U (i.e., pUlse) or 0, depend-



P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	
0.5	0.5	0.5	0.5	0	0	0	0.5	0	1	0	0	0	0	0	0	D_i

Fig. 8. Format of a new data instance for a two-input cell.



P1′	P2′	P3′	P4′	P5′	P6′	P7′	P8′	P9′	P10′	P11′	P12′	
0	0	0.5	1	0	1	0	0	0.5	0	0	0	Di

Fig. 9. Format of a new data instance as used in [33].

ing on whether or not there is an active clock signal. The second value can be 0, 1, R or F. The third value can only be static (i.e., 0 or 1). An example of training dataset for all defect types (static and dynamic) that may occur in a sequential cell is shown in Fig. 10. Note that the CA views used during the generation of training data do not contain information about cell-patterns with non-pulsing clock signals (i.e., none of the cell internal defects can be detected at the cell output without clock pulse). Consequently, the training data do not include such cell-patterns as can be observed in the example of Fig. 10. Note also that instance tables of sequential cells may contain cell-patterns with no transition on the main inputs of the cell. To allow the ML algorithm understanding this information, we include static cell-patterns (e.g., P1 to P4 in Fig. 10) in the training data of sequential cells.

With the above representation of training data for sequential cells, one can see that the diagnosis flow in Fig. 3 can be used in a straightforward manner without any change. The two main steps (model training by using a training dataset, implementation of the NB classifier to make inference) remain the same irrespective of the manipulated standard cell type.

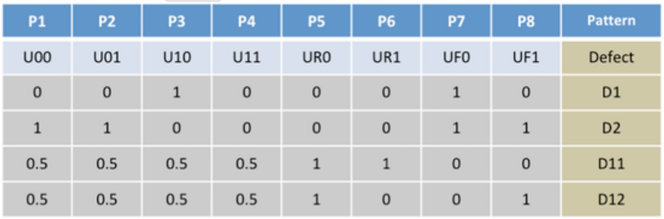
5. Experimental results

This new CA diagnosis flow targeting defects in both combinational and sequential cells of customer returns has been implemented in a Python program. For validation purpose, we have experimented the proposed flow in three different ways:

- First, we conducted experiments on **ITC'99 benchmark circuits** with defect injection campaigns targeting **combinational cells** in each circuit. Various results are reported in [28–33] to show the superiority of our framework when compared to commercial diagnosis solutions. A summary of these results is given in Subsection 5.1.
- Next, we considered a **test chip** developed by STMicroelectronics and designed using a 28 nm FDSOI technology, and we conducted two defect injection campaigns targeting **sequential cells**. Results are reported in Subsection 5.2 and also demonstrate the effectiveness of our diagnosis framework.
- Finally, we considered **a customer return** from STMicroelectronics and we performed a silicon case study with a real defect subsequently analyzed and identified during PFA. Results are reported in Subsection 5.3.

5.1. *Summary of results reported in* [32]

In [32], we conducted experiments on ITC'99 benchmark circuits synthesized in a full scan manner using a 28 nm FDSOI technology from STMicroelectronics. A commercial CA ATPG tool was used to generate static and dynamic CA test sequences targeting maximum fault coverage for each circuit. The behavior of the tester was simulated by performing a defect injection campaign in each circuit (about 2000 injections per circuit). Defects were injected into a number of randomly selected combinational cells by considering each cell transistor and targeting all possible static and dynamic defects affecting that transistor. As several defects have the same impact on the logic behavior of the cell, and hence are logical-equivalent defects, they were grouped in defect classes. After test sequences application, test information were collected to build the tester data log. Afterwards, we used a commercial logic diagnosis tool to determine the lists of suspected cells after static and dynamic test sequences application. Main circuit characteristics (number of cells and scan flip-flops) and test parameters (number of static and dynamic test patterns, stuck-at and static CA fault coverage, transition and dynamic CA fault coverage) can be found in [32].



P1	P2	P3	P4	P5	P6	P7	P8	Pattern
U00	U01	U10	U11	UR0	UR1	UF0	UF1	Defect
0	0	1	0	0	0	1	0	D1
1	1	0	0	0	0	1	1	D2
0.5	0.5	0.5	0.5	1	1	0	0	D11
0.5	0.5	0.5	0.5	1	0	0	1	D12

Fig. 10. Example of training dataset for all defect types (static and dynamic) in a sequential cell. The pin order is clock-data-previous output.

For generating training data, we used characterization data provided by a commercial tool and ST libraries. For generating new data instances, we performed post-processing of instance tables obtained as shown in Fig. 6. From the training data and the Gaussian NB model, we make predictions on new data instances. Results obtained are a list of defect candidates with the highest probability to be the root cause of failure.

Fig. 11 summarizes the results obtained on the biggest ITC'99 benchmark circuits. The curves report the accuracy and give, for each circuit, the percentage of cases in which the injected defect was reported in the list of suspects provided by the proposed CA diagnosis and the commercial CA diagnosis tool, respectively. The commercial tool is non-probabilistic and provides the list of all suspects obtained after CA diagnosis with a ranking and a matching score. The same characterization data and test scenario were used in both cases. These results show that the real (injected) defect is **always** identified by the proposed diagnosis flow. Sometimes, it is the only candidate and has a probability of 1 to be the best candidate. Sometimes, it is reported with some other candidates identified in one or more suspected cells. Conversely, we can observe that the commercial tool is **not always** able to report the injected defect as candidate. This proves the superiority of our proposed framework in terms of accuracy (always 100 %), which is not the case of the commercial tool that sometimes provides inaccurate results (at least for 5 out of 6 circuits). The reason of these misdiagnosis cases with the commercial tool are explained in [32].

The bars in Fig. 11 report the resolution and give, for each circuit and considering all injection campaigns, the average number of suspects reported by the proposed method and the commercial tool, respectively. As can be seen, the resolution achieved with our method is always better. So, overall, these results confirm the superiority of our approach.

In our experiments, suspected defects were classified using a publicly available machine learning software package called Scikit-learn, which is an integrated development environment with a suite of ML tools [35]. The single defect assumption was considered, although the proposed framework is able to manage situations where multiple defects have occurred, provided that those defects are not in the same cell. This significant feature comes from the fact that our diagnosis flow considers all suspected cells one at a time, and then incrementally constructs a list of suspected defects identified in each of these cells. Finally, in-field failure mechanisms related to premature aging, such as NBTI or HCI, essentially lead to resistive opens and shorts. These mech-

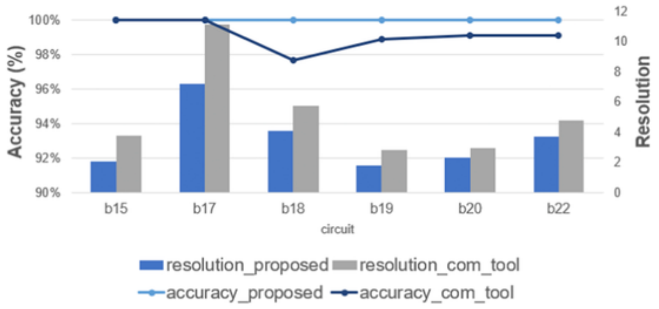


Fig. 11. Overall cell-aware diagnosis results [32].

anisms, that need to be considered in the context of customer returns, can be appropriately taken into account with our CA diagnosis flow.

The CPU time taken by the proposed flow to provide a list of defect candidates is always very low (few seconds) and does not depend on the circuit size. Only the number of suspected cells obtained after logic diagnosis may have an impact on the CPU time (to generate instances tables) but in a very light way (as this number is always very low). The most time-consuming part of the flow (few hours) is the characterization phase, but it is done only once and is not correlated with the circuit size.

5.2. Simulated test case studies

We also conducted experiments on a silicon test chip developed by STMicroelectronics and designed with a 28 nm FDSOI technology. The test chip is only composed of digital and memory blocks, and one PLL. The digital blocks are made of 3.8 million cells. Other features (number of primary inputs, primary outputs and scan flip-flops) are given in Table I.

We performed a first simulated case study with a **static** defect injection campaign. We successively injected all possible static defects into three scan flip-flops (SFF) of a single full-scan digital block. We tested this block with a static CA test sequence achieving a stuck-at + static CA fault coverage of 100 %. The average numbers of passing and failing test patterns are given in Table II. Results obtained after executing our CA diagnosis flow and averaged over all defect injections have shown an accuracy of **100 %** (the injected defect was always reported in the list of suspects) and a resolution of **1.25**. The resolution ranges between 1 and 3, and Fig. 12 shows the distribution of this resolution with respect to the total number of simulated cases. As can be seen, in most of the cases, the number of suspects is equal to 1 (perfect resolution).

We then performed a second simulated case study with another defect injection campaign on the same test chip. We successively injected all possible **dynamic** defects into three scan flip-flops of a single full-scan digital block. This time, we applied a dynamic CA test sequence achieving a transition + dynamic CA fault coverage of 89.8 %. The average number of failing test patterns was 7.9. Again, the results obtained after executing our CA diagnosis flow and averaged over all defect injections have shown an accuracy of **100 %**. The average resolution obtained for dynamic defect injection experiments was **1.37**. Again, the resolution ranged between 1 and 3, and in most of the case, the number of suspects was equal to 1.

Table I
Main features of the silicon test chip.

#cells	#PIs	#POs	#SFF
3.8 M	97	32	17.5 k

Table II
Average pattern count in instance tables of the first simulated case study.

#passing patterns	#unique passing patterns	#failing patterns	#unique failing patterns
43.4	24.0	15.5	8.6

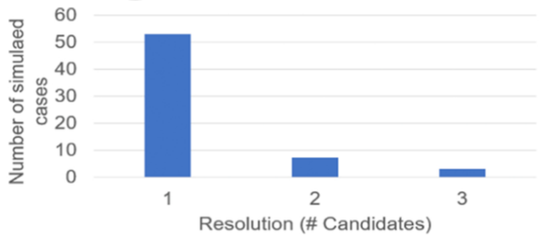


Fig. 12. Distribution of the resolution w.r.t. the simulated cases.

5.3. Silicon test case study

Finally, we performed a silicon case study on a customer return designed with a 28 nm FDSOI technology from STMicroelectronics. The test conditions used to run the experiments were the following: a nominal supply voltage of 0.83 V, a scan test frequency of 10 MHz, a launch-to-capture clock speed (for the dynamic CA test sequence application) adjusted with respect to the nominal clock frequency of the circuit, and a temperature of 25 °C. The process was considered as typical. We experimented our CA diagnosis flow, and we obtained the following results:

- Initially, the circuit failed on the tester after application of the static CA test sequence when applied at the nominal voltage. This information was stored in a "static" datalog.
- Then, a logic diagnosis gave a short list of suspected cells among which a six-input SFF cell made of 56 transistors and having a Reset, an Enable, a Test-Input and Test-Enable input pins. The cell contains 758 potential short or open defects. A static instance table was then generated for this suspected cell, and contained 5 failing and 75 passing cell-level test patterns.
- From the new data generated after post-processing of this instance table, the NB classifier identified four suspected defects among which defect D62 (a short between the gate and source of NMOS 19).

The above diagnosis results were provided to the Failure Analysis team of STMicroelectronics, who made a PFA in the past on this customer return based on the results found by their in-house intra-cell diagnosis tool. The result obtained with our CA diagnosis flow was validated as defect D62 was found to be the real defect. This was done after performing a polysilicon level inspection on the layout of the cell (cf. Fig. 13) and observing the failure analysis cross-sectional view.

6. Improvement of diagnosis resolution

Each test scenario as exemplified in Fig. 2 may lead to the occurrence of one or more failing patterns producing failure files to be diagnosed. The knowledge of failing and passing test scenarios can be used to improve the diagnosis resolution. To this end, it is crucial to have perfect knowledge about which defect type is detected by each test sequence and related test patterns. Let us clarify this point as follows:

Static defects require only one test pattern to be detected and they are usually targeted by static test patterns. However, these defects can also be detected by low speed and @speed dynamic test patterns. In fact, dynamic test patterns are two-vector test patterns, and it may happen that the second one exercises a stimulus detecting static defects.

Low speed dynamic defects require two-vector test patterns to be detected. The test frequency used to catch these defects is not relevant as the delay they induce is so huge that even a very low speed test can catch them. Therefore, low speed dynamic defects can be easily detected either by a low speed test or by a dynamic at-speed test. It is also worth mentioning that for scan designs, low speed defects can be acci-

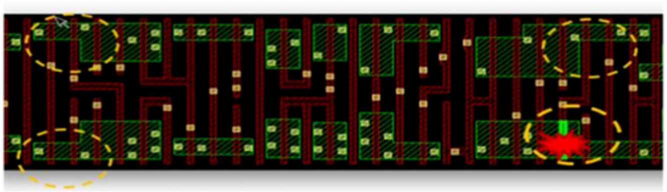
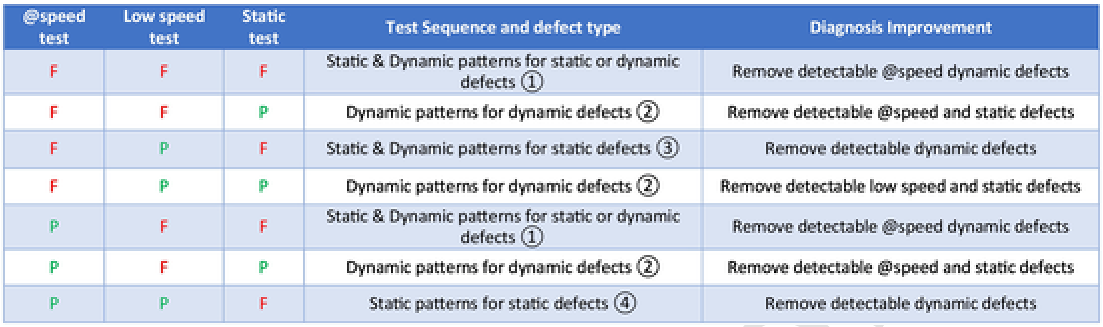


Fig. 13. Layout view of the suspected cell and the incriminated transistor. Yellow circles indicate defect candidates and red mark indicates actual observed defect. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)



@speed test	Low speed test	Static test	Test Sequence and defect type	Diagnosis Improvement
F	F	F	Static & Dynamic patterns for static or dynamic defects ①	Remove detectable @speed dynamic defects
F	F	P	Dynamic patterns for dynamic defects ②	Remove detectable @speed and static defects
F	P	F	Static & Dynamic patterns for static defects ③	Remove detectable dynamic defects
F	P	P	Dynamic patterns for dynamic defects ②	Remove detectable low speed and static defects
P	F	F	Static & Dynamic patterns for static or dynamic defects ①	Remove detectable @speed dynamic defects
P	F	P	Dynamic patterns for dynamic defects ②	Remove detectable @speed and static defects
P	P	F	Static patterns for static defects ④	Remove detectable dynamic defects

Fig. 14. Failing and passing test scenarios and corresponding analysis for diagnosis improvement.

dentally detected by static test patterns. Indeed, it happens that during the last scan shift cycle plus the capture cycle, a two-vector test pattern that exercises a dynamic defect is created.

@speed dynamic defects require two-vector test patterns applied at high frequency to be detected. They induce a delay that can be measured during the cell-aware model creation. Only @speed tests can detect these defects. The knowledge of the design max frequency can help to identify if the delay caused by the defect is large enough to be detected or not.

The ATPG flow as illustrated in Fig. 1 is an incremental flow, which means that only undetected defects are targeted by each new test pattern. However, it may happen that a defect targeted by, e.g., run 1, is also detected by run 2 or 3 unintentionally. Therefore, in a preprocessing step, we fault simulate the entire set of CA defects by each test pattern and identify detectable and undetectable defects. This information is further used to improve diagnosis resolution.

According to the above discussion, seven test scenarios may occur as depicted in Fig. 14. Each scenario is specified by Columns 1, 2 and 3. The letter ‘F’ indicates that the test is failing, ‘P’ indicates that the test is passing. Column 4 lists the test sequences to consider in each test scenario as well as the types of defect that can be detected with such scenario. The last column proposes an additional analysis that can be performed after the CA diagnosis to improve the diagnostic resolution. For example, in the first scenario the test sequence is composed of static and dynamic patterns since all test are failing. The defect type can be either static or dynamic but it cannot be @speed since low speed and static test are failing too. In order to improve the diagnosis, in this case we can remove any @speed candidates reported by the ML tool from the diagnosis report. Fig. 14 shows that the seven scenarios can be addressed by four methods (1), (2), (3) and (4) when the three types of test (@speed, low speed and static) are available.

7. Conclusion, discussion and future work

In this paper, we have presented a framework for cell-aware diagnosis of customer returns based on supervised learning. The proposed flow indistinctly deals with static and dynamic defects that may occur in combinational cells or sequential cells of real circuits. A Naive Bayes classifier was used to precisely identify defect candidates. A large set of experiments on both benchmark circuits and silicon test cases has been done to validate the proposed flow and demonstrate its efficacy in terms of accuracy and resolution.

Results of these experiments prove the appropriateness of a learning-based method to solve our problem, despite the small size of the training dataset used (only one sample for one defect class). When multiple defect sizes and test conditions will be used, this will be even truer. Indeed, multiple samples (one for each defect size or defect size range, one for each PVT test condition) will be associated to a given defect class, simply because the behavior of the defect will differ when applying the same set of test patterns. In terms of timing and complexity, this will just slightly impact our method, since training dataset is extracted from characterized cell libraries that are generated anyway for test and diagnosis purpose. So, even with large cell libraries with a huge number of defects to be simulated (e.g., 631 cells in a library, each with 4 to 6 inputs, 951 shorts and 749 opens on average – typical example of an ST library), our framework will still be easily and time-efficiently applicable.

It is worth noting that among other factors, the effectiveness of our framework can be explained by the fact that Naïve Bayes algorithm usually offers good classification performance [36].

The NB classifier requires a small amount of training data to estimate its parameters [37], which is the case in our method, as only one instance per class (i.e., CA defect) is available. On the other hand, other popular ML classification algorithms, such as K-Nearest Neighbors (KNN) classifier or classifiers based on a Support Vector Machine (SVM), which estimate the class of a new sample by analyzing the classes of similar training samples, cannot properly work when only one sample per class is available.

In our simulated case studies, all injected defects for evaluation purposes were present in the training dataset. Similarly, in our silicon case studies, the actual defects were also present in the training dataset. However, in customer returns, actual defect behavior may not perfectly match the fault models that are used to train the NB classifier. Further work will be done to see how well the proposed method works in that scenario. Finally, by exploiting the ranking of suspected cells usually provided after logic diagnosis by commercial tools, which was not done in the current work, our flow will be able to provide a similar ranking among defect candidates, thus giving additional useful information to be used during PFA.

Declaration of competing interest

No conflict of interest.

Acknowledgements

This work has been funded by the French National Research Agency (ANR) under the framework of the ANR-17-CE24-0014-01 EDITSoC (Electrical Diagnosis for IoT SoCs in automotive) project.

References

[1] F. Hapke, et al., Cell-aware test, IEEE Trans. Comput.-Aided Des. 33 (9) (2014) 1396–1409.
[2] S. Pateras, IC test solutions for the automotive market, in: Mentor Graphics, White Paper, May 2017.
[3] P. Maxwell, F. Hapke, H. Tang, Cell-aware diagnosis: defective inmates exposed in their cells, in: Proc. IEEE European Test Symposium, 2016.
[4] J. Tikkanen, N. Sumikawa, M.S. Abadir, Li-C. Wang, Multivariate outlier modeling for capturing customer returns – how simple it can be, in: Proc. IEEE On-Line Test Symposium, 2014.
[5] A. Ladhar, M. Masmoudi, L. Bouzaida, Efficient and accurate method for intra-gate defect diagnoses in nanometer technology, in: Proc. IEEE/ACM Design Automation and Test in Europe, 2009.
[6] A. Bosio, P. Girard, S. Pravossoudovitch, A. Virazel, A comprehensive framework for logic diagnosis of arbitrary defects, IEEE Trans. Comput. 59 (3) (2010)

289–300.
[7] Y. Benabboud, A. Bosio, L. Dilillo, P. Girard, A. Virazel, O. Riewer, A comprehensive system-on-chip logic diagnosis, in: Proc. IEEE Asian Test Symposium, 2010.
[8] Z. Sun, A. Bosio, L. Dilillo, P. Girard, A. Virazel, E. Auvray, Effect-cause intra-cell diagnosis at transistor level, in: Proc. IEEE International Symposium on Quality Electronic Design, 2013.
[9] L.M. Huisman, Diagnosing arbitrary defects in logic designs using the single location at a time (SLAT), IEEE Trans. Comput.-Aided Des. 23 (1) (2004) 91.
[10] S. Holst, H.-J. Wunderlich, Adaptative debug and diagnosis without fault dictionaries, in: Proc. IEEE European Test Symposium, 2007.
[11] J.A. Waicukauski, E. Lindbloom, Failure diagnosis of structured VLSI, IEEE Des. Test. Comput. 6 (4) (1989) 49–60.
[12] M. Abramovici, P.R. Menon, D.T. Miller, Critical path tracing-an alternative to fault simulation, in: Proc. ACM Design Automation Conference, 1983.
[13] P. Girard, C. Landrault, S. Pravossoudovitch, Delay-fault diagnosis by critical-path tracing, IEEE Des. Test. Comput. 9 (4) (1992) 27–32.
[14] Y. Xue, X. Li, R.D. Blanton, C. Lim, Improving diagnostic resolution of failing ICs through learning, IEEE Trans. Comput.-Aided Des. 37 (6) (2018) 1288–1297.
[15] F. Liu, P.K. Nikolov, S. Ozev, Parametric fault diagnosis for analog circuits using a Bayesian framework, in: Proc. IEEE VLSI Test Symposium, 2006.
[16] S. Wang, W. Wei, Machine learning-based volume diagnosis, in: Proc. IEEE/ACM Design Automation and Test in Europe, 2009.
[17] J.E. Nelson, W.C. Tam, R.D. Blanton, Automatic classification of bridge defects, in: Proc. IEEE International Test Conference, 2010.
[18] Li-C. Wang, Data learning based diagnosis, in: Proc. ACM/IEEE Asia and South Pacific Design Automation Conference, 2010.
[19] R.D. Blanton, et al., DREAMS: DFM rule evaluation using manufactured silicon, in: Proc. ACM/IEEE International Conference on Computer-Aided Design, 2013.
[20] Y. Xue, O. Poku, X. Li, R.D. Blanton, PADRE: physically-aware diagnostic resolution enhancement, in: Proc. IEEE International Test Conference, 2013.
[21] R.J. Tikkanen, S. Siatkowski, M.S. Abadir, Li-C. Wang, Yield optimization using advanced statistical correlation methods, in: Proc. IEEE International Test Conference, 2014.
[22] X. Ren, M. Martin, R.D. Blanton, Improving accuracy of on-chip diagnosis via incremental learning, in: Proc. IEEE VLSI Test Symposium, 2015.
[23] Y. Huang, W. Yang, W. Cheng, Advancements in diagnosis driven yield analysis: a survey of state-of-the-art scan diagnosis and yield analysis technologies, in: Proc. IEEE European Test Symposium, 2015.
[24] Y. Xue, X. Li, R.D. Blanton, C. Lim, Diagnosis resolution improvement through learning-guided physical failure analysis, in: Proc. IEEE International Test Conference, 2016.
[25] Q. Huang, C. Fang, R.D. Blanton, Diagnosis outcome prediction on limited data via transferred random forest, in: Proc. IEEE International Test Conference - Asia, 2020.
[26] Q. Huang, C. Fang, R.D. Blanton, LAIDAR: learning for accuracy and ideal diagnostic resolution, in: Proc. IEEE International Test Conference, 2020.
[27] R. Pan, Z. Zhang, X. Li, K. Chakrabarty, X. Gu, Unsupervised root-cause analysis for integrated systems, in: Proc. IEEE International Test Conference, 2020.
[28] S. Mhandi, A. Virazel, P. Girard, A. Bosio, E. Auvray, E. Faehn, A. Ladhar, Towards Improvement of Mission Mode Failure Diagnosis for System-on-Chip, in: Proc. IEEE International On-Line Testing Symposium, 2019.
[29] S. Mhandi, P. Girard, A. Virazel, A. Bosio, A. Ladhar, Cell-aware diagnosis of automotive customer returns based on supervised learning, in: Presented at IEEE Automotive Reliability and Test Workshop, 2019.
[30] S. Mhamdi, P. Girard, A. Virazel, A. Bosio, E. Faehn, A. Ladhar, Cell-aware defect diagnosis of customer returns based on supervised learning, IEEE Trans. on Device Material and Reliability 20 (2) (2020).
[31] S. Mhamdi, P. Girard, A. Virazel, A. Bosio, A. Ladhar, Learning-based cell-aware defect diagnosis of customer returns, in: Proc. IEEE European Test Symposium, 2020.
[32] S. Mhamdi, P. Girard, A. Virazel, A. Bosio, A. Ladhar, A learning-based cell-aware diagnosis flow for industrial customer returns, in: Proc. IEEE International Test Conference, 2020.
[33] S. Mhamdi, P. Girard, A. Virazel, A. Bosio, A. Ladhar, Cell-aware diagnosis of customer returns using Bayesian inference, in: Proc. IEEE International Symposium on Quality Electronic Design, 2021.
[34] S.B. Kotsiantis, Supervised machine learning: a review of classification techniques, Informatica 31 (3) (2007).
[35] Webpage n.d. "User Guide: Contents — Scikit-Learn 0.24.1 Documentation", http://scikit-learn.org/stable/user_guide.html.
[36] H. Zhang, The optimality of naive Bayes, in: Proc. 17th International Florida Artificial Intelligence Research Society Conference, May, 2004.
[37] Webpage n.d. "1.9. Naive Bayes — Scikit-Learn 0.24.1 Documentation", https://scikit-learn.org/stable/modules/naive_bayes.html.

P. d'Hondt received his Engineering degree from the Institut Supérieur de l'Electronique et du Numérique (ISEN), Lille, France, in 2019. He is currently a PhD student at the University of Montpellier, and works with STMicroelectronics, Crolles, France. Her main focus of research lies in Characterization for Test and Diagnosis of Digital circuits.

S. Mhamdi received her Master degree in Microelectronics from the University of Montpellier, France, in 2017. She is currently a PhD student at the University of Montpellier, and works in the Microelectronics Department of the LIRMM (Laboratory of Informatics, Robotics and Microelectronics of Montpellier – France). Her main focus of research lies in Test and Diagnosis of Digital circuits and systems and Reliability

P. Girard received a Ph.D. degree in Microelectronics from the University of Montpellier, France, in 1992. He is currently Research Director at CNRS (French National Center for Scientific Research) and works in the Microelectronics Department of the Laboratory of Computer Science, Robotics and Microelectronics of Montpellier (LIRMM) - France. He is Director of the International Associated Laboratory « LAFISI » (French-Italian Research Laboratory on Hardware-Software Integrated Systems). He is deputy director of the French scientific network dedicated to research in the fields of System-on-Chip, Embedded Systems and Connected Objects (SOC2). His research interests include all aspects of digital and memory testing, with emphasis on critical constraints such as timing and power. Robust design of neuromorphic circuits as well as machine learning for test and diagnosis are also part of his new research activities. He has supervised 40 PhD dissertations and has published 7 books or book chapters, 80 journal papers, and more than 250 conference and symposium papers on these fields. Patrick Girard is a Fellow of the IEEE.

A. Virazel received the Ph.D. degree in Microelectronics from the University of Montpellier, France, in 2001. He is currently Associate Professor at the University of Montpellier, and works in the Microelectronics Department of the LIRMM (Laboratory of Informatics, Robotics and Microelectronics of Montpellier - France) where he is leading the TEST ("Test and dEpendability of microelectronic integrated SysTems") team. He has published 3 books or book chapters, 40 journal papers, and more than 140 conference and symposium papers spanning diverse disciplines, including DfT, BIST, diagnosis, reliability, delay testing, power-aware testing and memory testing. He is deputy head of the electrical engineering Master program (about 200 students) in charge of the first year and of the "Integrated Electronic Systems" specialization at the University of Montpellier.

A. Ladhar received the PhD degree in Electrical Engineering from the University of Sfax, Tunisia, in 2010. He is currently a senior test & yield engineer at STMicroelectronics Crolles, France. His research interests include VLSI testing, fault diagnosis, layout analysis, defect extraction and simulation.