Towards Improvement of Mission Mode Failure Diagnosis for System-on-Chip

Safa Mhamdi, Arnaud Virazel, Patrick Girard, Alberto Bosio, Etienne Auvray, Aymen Ladhar, Eric Faehn

Towards Improvement of Mission Mode Failure Diagnosis for System-on-Chip

S. Mhamdi A. Virazel P. Girard
LIRMM, Univ. of Montpellier / CNRS
Montpellier, France
<lastname>@lirmm.fr

A. Bosio
INL, École Centrale de Lyon
France
alberto.bosio@ec-lyon.fr

E. Auvray E. Faehn A. Ladhar
STMicroelectronics
Crolles, France
<firstname.lastname>@st.com

***Abstract*—In critical (e.g. automotive) applications, Systems-on-Chip (SoC) failures that occurred during mission mode (in the field) are the most critical since they may lead to catastrophic effects. In this context, diagnosis is crucial in order to establish the root cause of observed failures with the best accuracy. With the advent of very deep submicron technologies (i.e. 7 nm), achieving such level of accuracy will become more and more difficult with today's intra-cell diagnosis tools based on effect-cause or cause-effect paradigms. This will compromise the success of subsequent Physical Failure Analysis (PFA) done on defective SoCs. Machine Learning (ML) is now used in numerous classification problems where the knowledge on some data can be used to classify a new instance of such data. In particular, several ML-based solutions exist to address volume diagnosis for yield improvement. These learning-guided diagnosis approaches start from an existing set of defect candidates and try to minimize this set (eliminate bad candidates) owing to the use of ML tools and numerous data collected during production test (e.g. thousands of failed chips with candidates correctly labeled). Although efficient in volume diagnosis, these approaches cannot be used to identify the root cause of failures in customer returns, since only one failed chip is investigated in this case, with no information about the defective behavior of some other similar chips used in the same conditions (environment, workload, etc.). In this paper, we propose a new learning-guided approach for diagnosis of mission mode failures in customer returns. The proposed approach directly produces a minimum set of good candidates derived from the application of the learning-guided intra-cell diagnosis flow. Results obtained on a set of benchmark circuits, and comparison with a commercial intra-cell diagnosis tool, show the feasibility, effectiveness and accuracy of the proposed approach.**

Keywords—Diagnosis, Machine Learning, SoC, Customer returns.

I. INTRODUCTION

Today's electronic systems are composed of complex SoCs that consist of heterogeneous blocks that comprise memories, digital circuits, analog and mixed-signal circuits, etc. To fit a critical application standard requirement, SoCs pass through a set of test phases at the end of the manufacturing process. The goal is to achieve near-zero defective parts per million (DPPM) so as to ensure the quality level required by the standard.

Despite the quality level (in percentage of fault coverage) of the test sequences generated by industrial or in-house tools and used during manufacturing test, SoCs may fail in mission mode due to i) occurrence of a defect not covered during the manufacturing test phase, or ii) occurrence of early-life failures or failures due to various wear-out mechanisms. Early-life failures, also called infant mortality, are caused by defects that are not exposed during manufacturing tests, but that are degraded due to electrical and thermal stress during in-field use, and lead to a failure in functionality. Wear-out, also called aging, manifesting as progressive performance degradation, is induced by various mechanisms such as, e.g., Negative-Bias Temperature Instability (NBTI) or Hot-Carrier Injection (HCI).

Such failures that occur during the mission mode are the most critical as they may result in catastrophic consequences. Thus, in an attempt to identify the source of these failures and avoid their re-occurrence in next generation products, the defective SoC is always sent back to the manufacturer (referred to as "customer returns") who is in charge of analyzing the device to determine the root cause of failures [1]. In this scenario, failures are not easy to reproduce in the company lab as the real mission conditions and executed workload are generally unknown and cannot be exhaustively modeled. Therefore, efficient diagnosis methods to locate and assess failures at different system levels are of vital importance.

Diagnosis is usually followed by PFA, a time-consuming and destructive process for exposing the defect physically in order to characterize the failure mechanism. Due to the high cost and destructive nature of PFA, diagnosis resolution is of critical importance. In practice, it is very uncommon to perform PFA on any defect with more than five candidates [2]. Ideally, resolution is one, that is, a single location is identified when a defect is diagnosed. This ensures that the likelihood for uncovering the root-cause of failure is maximized when performing PFA. However, with the advent of very deep submicron technologies, such a resolution is not always reachable by today's intra-cell logic diagnosis tools based on conventional methods (effect-cause / cause-effect) [3]. In this context, machine learning can be viewed as an efficient mean to exploit data (logical or physical) other than that used by conventional methods to improve diagnosis resolution [4].

In this paper, we present a new learning-guided approach for diagnosis of mission mode failures and demonstrate its effectiveness to identify the root cause of failures in customer returns. State-of-the-art learning-guided diagnosis approaches used for volume diagnosis start from an existing set of defect candidates (obtained from a conventional diagnosis technique) and try to minimize this set owing to the use of ML tools that exploit the knowledge of defective behavior on other similar failed chips. Conversely, the proposed learning-guided intra-cell diagnosis flow aims at **directly** producing a minimum set of candidates by exploiting only test data associated with the targeted customer return. To the best of our knowledge, this is the first time such type of diagnosis approach is proposed.

The rest of this paper is organized as follows. Section II presents a state-of-the-art on diagnosis methods and gives the motivations of this work. Section III presents the proposed learning-guided intra-cell diagnosis approach. Section IV presents results obtained on a set of benchmark circuits, as well as a comparison with a commercial tool. Section V concludes the paper and discusses future work that still need to be done.

II. State of the Art and Motivations

Diagnosis is the first analysis step for a defective SoC. This is a software-based method that analyzes the applied tests, the tester responses, and the netlist (possibly with layout information) to produce a list of candidates that represent the possible locations and types of defects (or faults) within the defective SoC [5]. The key metrics that characterize diagnosis performance are resolution, *i.e.*, the number of candidates reported by diagnosis for a given defective SoC, and accuracy, *i.e.*, the physical defect is indeed in the list of candidates.

In the case of a customer return, the first step is to re-use the original test program to check if the SoC fails again or not. If not, efforts have to be made to find new test patterns and test conditions (i.e. voltage and temperature) that will sensitize the defect and reveal the failure. Otherwise, if the SoC fails, 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. *SoC level diagnosis* is the first routine used to identify the cores or interconnections in the system that can explain the failure. *Core level diagnosis* (inter-cell diagnosis) is then used to identify the possible failing cells within a core (or block). *Intra-cell diagnosis* is finally used to pinpoint the possible defect candidates within a cell (or gate).

Except industrial in-house SoC diagnosis tools, the literature proposes very few comprehensive diagnosis approach able to deal with a full SoC and provide reliable information about fault localization. To the best of our knowledge, the only work targeting SoC-level diagnosis is reported in [6]. The key concept is that diagnosis consists in a comparison between a set of pre-computed SoC failures and the set of failures observed during test. This type of approach was formerly proposed in [7] and [8] but only for full-scan circuits. In [4], authors propose to extend it to the case of SoC. The main advantages of this approach w.r.t. the state-of-the-art are (i) the capability to manage both full-scan and sequential logic cores, (ii) to deal with several fault models at a time (both static and dynamic) and (iii) to address both single and multiple fault occurrences.

Regarding core-level diagnosis, a considerable amount of work can be found in the literature. Dedicated techniques have been proposed to target specific cores: logic cores (logic diagnosis) [7]–[9], memory cores (memory diagnosis) [10–11] and analog cores (analog diagnosis) [12–13]. Considering logic diagnosis, the result is either an interconnection between gates or a suspected gate. Faults can hence occur either in the interconnection between gates (inter-cell faults) or inside the gate (intra-cell faults). When the observed failure is inside the gate, another diagnosis approach is applied to locate the cause of this failure at the transistor level [4]. An intra-cell diagnosis method used for mission mode failures in customer returns has been proposed in [14]. It uses a CPT algorithm applied at transistor level and works as follows. First, the test determines which are the failing and passing test patterns for a given Circuit Under Test (CUT). Then, *logic diagnosis* exploits this information to determine a list of suspected gates (candidates). Any available logic diagnosis tool can be used. For each suspected gate, we have to know the logical values applied to it when failing and passing test patterns are applied to the CUT. This step amounts to determine the actual set of failing/passing test patterns at the cell level. Finally, *intra-cell diagnosis* is executed for each suspected gate and its corresponding failing /passing test patterns. The result is a list of suspected nets at transistor level with a set of fault models able to explain the observed failures. More details about this flow and results obtained on industrial circuits from ST can be found in [14].

Unfortunately, for various reasons, diagnostic resolution is typically far from ideal due to the SoC complexity. As a result, a lot of efforts have been dedicated for improving diagnosis resolution. Among several types of solutions, it has been demonstrated recently that diagnosis resolution can be improved with machine learning techniques, primarily through the derivation of characteristics that enables correct candidates (candidates that correctly represent defect locations) to be distinguished from incorrect ones (candidates that do not) [15]–[20]. In [15], authors describe an approach to identify bridge defects from a population of diagnosed defects by using a combination of effective rules and a decision-tree-based classifier. In [16], authors improve on-chip diagnosis resolution with a modified k-nearest neighbors classifier that is updated with real-time failure data. In [17], volume diagnosis resolution is improved with a Bayesian classifier that identifies the actual candidates based on their layout properties. In [18], authors present a novel yield optimization methodology based on establishing a strong correlation between a group of fails and an adjustable process parameter. The core of the methodology comprises three advanced statistical correlation methods. In [19], the authors use statistical learning methods to predict the termination of tester-data collection to ensure good resolution. In [20], a machine learning-based resolution improvement approach called PADRE (Physically-Aware Diagnostic Resolution Enhancement) is proposed. PADRE uses a classification algorithm (Support Vector Machine) to analyze easily available tester and simulation characteristics about the candidates to identify those that correspond to the actual failure locations. The capabilities of this solution have been further extended with a novel Active Learning (AL) based PFA selection approach [2]. AL-PADRE selects the most useful defects for PFA in order to improve diagnostic resolution.

Despite their efficiency, a common feature of these techniques is that they all address volume diagnosis for yield improvement, which is a different problem than fault diagnosis of customer returns. During volume diagnosis, numerous data collected during manufacturing test and subsequent diagnosis phases are available, such as, e.g. hundreds of similar failed chips with candidates correctly labeled (good or bad) obtained in a previous stage. It is therefore possible to use these data for failure diagnosis of a new failed chip. Conversely, during fault diagnosis of customer returns, 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, etc.). For this reason, learning-guided approaches used for volume diagnosis cannot be used fault diagnosis of customer returns.

III. Learning-Guided Intra-Cell Diagnosis

Despite the good resolution achievable with conventional intra-cell diagnosis technique, in some cases (e.g. complex cells, complex failure mechanisms) the number of candidates is too high to allow an efficient PFA. This problem will be exacerbated with the advent of very deep (i.e. 7 nm) submicron technologies. Improving intra-cell diagnosis efficiency is therefore mandatory. A mean to achieve this goal is to use learning algorithms based on classification to determine suspected defects. In this section, we present a new approach that uses ML techniques instead of traditional cause-effect and/or effect-cause analysis. This technique identifies defect candidates within a cell with a high resolution and accuracy.

A. Overall Diagnosis Flow

Figure 1 shows the proposed diagnosis flow. It takes as input (i) the cell test patterns, i.e., all possible static and dynamic combinations of values on the inputs of a cell, (ii) the list of all possible types of cell in a given circuit, and (iii) the cell netlists at transistor level. From these inputs, intra-cell transistor-level defect simulations using Spice are performed by iteratively injecting all possible defects into each cell type and then simulate the behavior of the cell. This part of the flow can be seen as a characterization phase as **it is done only once for each type of cell** (NAND, NOR, etc.) within a considered circuit. The output is a set of instances representing the training data. A features vector describes each instance and has the format shown in Figure 2 (for a two-input cell). This instance represents, for each defect, the possibility to be detected (value 1) or undetected (value 0) by each cell test pattern Pi at the output of the cell. Cell test patterns are static (one input vector) or dynamic (two input vectors). For an n-input cell, there exists 2^n static test patterns and $2^n.(2^n - 1)$ dynamic test patterns.

Besides training data, the Learning-Guided Intra-Cell Diagnosis (LGICD) module receives new data. Each instance of the new dataset represents a defect candidate that has to be classified as good or bad candidates. These data are obtained from a data instance generation module that uses as inputs (i) the cell failing/passing test patterns, (ii) the list of suspected cells provided by a logic diagnosis tool and ranked according to their score to be the source of failure (to contain the real defect), and (iii) the cell netlists. The cell failing/passing test patterns are obtained by performing a simple logic simulation of the CUT with the failing/passing test patterns identified by the tester. The format of a new data instance is quite similar to that of a training data instance, but has a different meaning. In each instance, the value 1 (respectively 0) is associated to a failing (respectively passing) cell test pattern Pi for a given defect candidate, meaning that the candidate is indeed detected (respectively undetected) by the cell test pattern Pi. In such instance, the value 0.5 is associated to a cell test pattern for a given defect candidate when this pattern does not exist in the list of cell failing/passing test patterns (i.e., the cell test pattern can not appear at the inputs of a suspected cell). The median value 0.5 has been chosen to avoid missing information in new data instances while not biasing the features of these data. Finally, from the training and new data, the LGICD module provides a set of good transistor-level defect candidates with the corresponding probability to be the root cause of the failure.

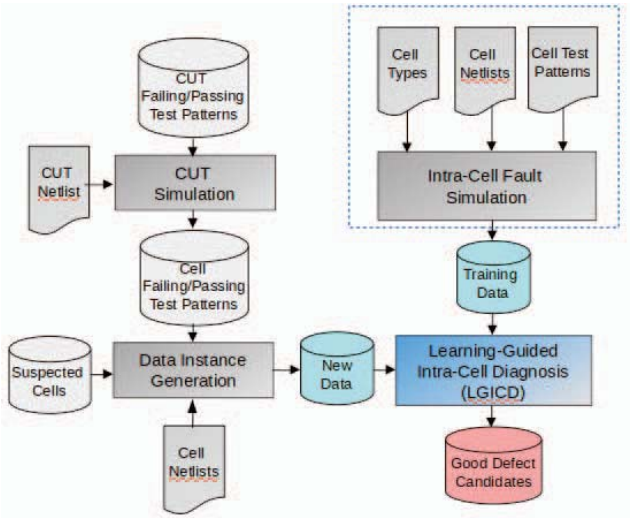

Figure 1: Learning-guided intra-cell diagnosis flow

P1	P2	P3	P4	P5	P6	····	P11	P12	P13	P14	P15	P16	Defect
1	1	1	0	1	1	····	0	1	1	0	1	1	D17

Figure 2: Format of a training data for a two-input cell

B. Details of the LGICD Module

Details of the LGICD module are illustrated in Figure 3. The first step consists in data preparation. Since learning algorithms learn from data, it is essential to use data that perfectly match the problem to be solved. Data have to be in a useful format and include meaningful features. In our case, the process for getting data ready for machine learning algorithms can be summarized in three phases [21]:

i) *Data Selection*. It consists in selecting the subset of all available data that will be used to build a model and classify new data. In our case, available data are the training data obtained from the intra-cell transistor-level fault simulation (characterization phase). Each instance of the training data is associated to a defect, and corresponds to the behavior of the cell (fault-free or faulty) in presence of such defect and for all possible combinations (static and dynamic) of the cell inputs. Some defects lead to the same cell behavior. These defects are called equivalent defects. Some others are undetectable by any cell test pattern. In our selection process, 90% of the available data were randomly selected and this operation was repeated several times to obtain training data with good randomness.

ii) *Data Preprocessing*. Once training data have been selected, we need to consider how they will be used. Training data are first stored in a CSV file. Then, they are sampled and grouped by considering equivalent defects. All equivalent defects are thus associated to a given Defect Class *i* (DCi). Training data instances of undetectable defects are removed.

iii) *Data Transformation*. The final phase is to transform the preprocessed data ready for learning in a format manageable by the classifiers (or models). In this format, each instance of a training data contains *m+1* columns, where m =

$2^n + 2^n.(2^n - 1)$ for an n-input cell (e.g. 16 for a 2-input cell). Columns 1 to *m* correspond to the exhaustive cell test patterns. Column *m+1* corresponds to each defect class. The names of each column are specified when transforming data. This will help to explore these data in a later stage of the process.

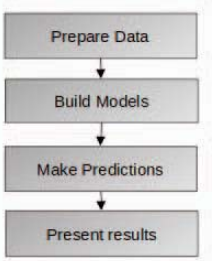

Figure 3: Main steps of the LGICD module

In the second step of the LGICD module, we build models based on different classification algorithms (called classifiers). As we preliminary do not know which algorithm will be efficient for our problem, evaluating selected algorithms is an important step. These evaluations are most often based on prediction accuracy (the percentage of correct prediction divided by the total number of predictions). There are many techniques used to calculate the accuracy of a classifier. The technique used in this work is known as cross-validation. The training data is divided into mutually exclusive and equal subsets. For each subset, the classifier is trained on the union of all the other subsets [22].

In the next two steps and once we have selected the models, we make predictions on new data instances with all of the available data. In our case, the expected results correspond to a defect's class probability of being the root-cause of failure. Each model returns the best defect's class candidate and the probability for each class has a value between 0 and 1.

C. *The Scikit-learn Library*

In this work, suspected defects are classified using a publicly available machine learning software package called Scikit-learn [23]. Scikit-learn is an integrated development environment with a suite of ML tools. Various tools of Scikit-learn with supervised learning algorithms for classification have been used in this work. The selected algorithms are the following: Logistic Regression (LR), k-nearest-neighbors (KNN), Naive Bayes (NB) Classifier, and Support Vector Machines (SVM). These algorithms represent a mixture of simple linear (LR) and nonlinear (KNN, NB, SVM) algoritms.

IV. EXPERIMENTAL RESULTS

The proposed learning-guided intra-cell diagnosis approach has been implemented in a Python program. We conducted experiments on ISCAS'85 benchmarks circuits synthesized in a 28nm FDSOI technology from STMicroelectronics. Circuits were synthesized using a commercial tool. We used an ATPG tool (commercial) to generate test patterns for each circuit. The stuck-at fault model was considered and the single fault assumption was done. Test patterns were provided to achieve 100% fault coverage. From each circuit and the corresponding test set, we simulated the behavior of the tester by performing a defect injection campaign (about 500 injections per circuit) into a number of randomly selected gates and collecting test information to build the tester data log. For the defect injection campaign, we considered each transistor of the selected gates and we targeted all possible defects affecting that transistor. These defects are shown in Figure 4 and are as follows:

- RO_i: Open defect at node *i* (*i* = [Gate, Drain, Source, Bulk])
- RS_{ij}: Short defect between nodes *i* and *j* (*i/j* = [Gate, Drain, Source, Bulk])

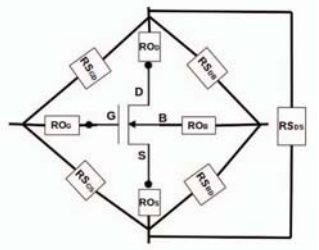

Figure 4: Considered transistor defects

For example, the number of defects for a NAND2 gate is equal to 36 defects (9 defects per transistor). However, several defects have the same impact on the logic behavior of the gate. So, these defects are logical-equivalent defects and hence are grouped in defect classes. Table I shows the equivalent defects with the corresponding defect classes for such a gate. In this table, *Dtk* refers to a defect in transistor t (*t* ranges from 1 to 4), and *k* indicates the type (*RO* or *RS*) and source node (Gate, Drain, Source, Bulk) of the defect. Labels from 1 to 4 for *k* refer to open defects. Labels from 5 to 9 refer to short defects.

TABLE I. NAND2 DEFECT CLASSES

Defect Class	Equivalent Defects
DC1	D11, D12, D13, D16, D21, D22, D23, D26, D38, D39, D48, D49
DC2	D14, D24, D28, D34, D37, D44, D47
DC3	D15, D36
DC4	D17
DC5	D18, D27, D29
DC6	D19
DC7	D25
DC8	D31, D32, D33, D35
DC9	D41, D42, D43, D45
DC10	D46

From the list of failing/passing test patterns (#TP) with the corresponding failing/passing CUT outputs, an inter-cell logic diagnosis tool based on fault simulation was used to determine a list of suspected gates ranked according to their score to be the source of failure. We used a commercial diagnosis tool to this purpose. For most of experiments, the list of suspected gates contained the gate in which the defect were injected. In few cases, about 5%, the commercial tool was unable to identify the faulty gate as suspect. Learning-guided intra-cell diagnosis was not done in such cases. The average number of suspected gates (#aSG) for each circuit is listed in Table II.

According to the flow presented in Figure 1, the list of all possible types of cell and the cell test patterns for each cell are used as input to produce the training data. For a two-input gate, 16 cell test patterns are used: 4 static patterns (0-0,0-1,1-0,1-1) and 12 dynamic patterns (00-01, 00-10, 00-11, 01-00, etc.). In these experiments, dynamic patterns are used to sensitize stuck-open faults. Later on, dynamic patterns will also be used to sensitize delay faults. Note that this number (16) increases exponentially with the number of inputs of a gate. Similarly, the length of each training data (as shown in Figure 2) as well

as the number of data increases accordingly. The third input used to produce the training data is the cell netlists, where each netlist is an electrical description of the cell with an injected defect. From all these inputs, intra-cell transistor-level defect simulations using Spice are performed for all possible defects, hence producing the training data. This characterization phase of the flow was done using a commercial tool and ST libraries.

TABLE II. RESULTS OF INTER-CELL LOGIC DIAGNOSIS

Circuit	#PIs	#POs	#Gates	#TP	#aSG
C880	60	26	383	34	2
C1355	41	32	938	85	3
C2670	233	140	945	60	3
C3540	50	22	1504	131	2
C5315	178	123	2228	75	2
C7552	207	108	3417	83	4

For generating each new data, we used the list of suspected gates and the cell netlists as input. All suspected gates are considered successively according to their score ranking. In addition, we need to use the cell (local) failing/passing test patterns of the suspected gates. For a suspected gate, the number of local test patterns can be lower than 16, especially when it was not possible to obtain some patterns at the gate inputs from the initial circuit-level test patterns. Finally, from these inputs, a simple simulation-based generation algorithm is used to produce the new data instance (defect) to be classified.

From the training and new dataset, the LGICD module proceeds as explained in Section IV.B. Regarding the second step of the LGICD module, we use a cross-validation algorithm to calculate the prediction accuracy of the initially selected learning algorithms. Table III shows the result obtained for each learning algorithm. NB, KNN and SVM have the best prediction accuracy scores, meaning that they are the best algorithms for learning data and providing defect candidates. LR has a good accuracy score but a bit lower than the others.

TABLE III. PREDICTION ACCURACY EVALUATION

Algorithm	Accuracy score
LR	60%
KNN	81%
NB	84%
SVM	79%

Table IV illustrates the results obtained by the LGICD module for a defect injection campaign in a two-input AND gate of circuit c2670 (with 54 open and short defects). The first column lists the various defect classes. The second column indicates if the defects of the corresponding class could be detected or not by the initial circuit-level test set. In the case such defects cannot be detected, this means that they have no impact on the gate output and hence cannot be the source of failure. So, they will no longer be considered in our diagnosis process. The third column shows the number of suspected gates obtained after logic diagnosis. The next four columns list the best defect's class candidate for each learning algorithm with the corresponding probability of being the root cause of failure.

From these results, the first comment is that KNN and NB identify as best candidate the real (injected) defect. This is true for all defect classes. For LR, the real defect is always identified as a candidate, but sometimes (for DC2, DC3 and DC4) in the second or third position. The second comment refers to the probability given to each best candidate. For example, for DC1, the probability given by LR to DC1 to be the best candidate is 0.11. KNN gives a probability of 0.5 (with n-neighbors=2), which is even better. NB gives a probability of 1 to DC1 to be the best candidate, hence do not providing any other candidates with lower probabilities (unlike what is done by LR and KNN). SVM is a non-probabilistic algorithm and gives the right defect class in 7 (over 9) cases. It does not provide any other candidate (inherent property). All these results clearly demonstrate the feasibility, the effectiveness (in terms of resolution) and accuracy of the proposed approach.

TABLE IV. LEARNING-GUIDED DIAGNOSIS RESULTS – C2670

Class	Det	#SG	LR	KNN	NB	SVM
DC1	Yes	2	DC1=0.11	DC1=0.5	DC1=1	DC1
DC2	Yes	3	DC6=0.11 / DC2=0.09	DC2=0.5	DC2=0.5	DC6
DC3	Yes	2	DC5=0.11 / DC3=0.10	DC3=0.5	DC3=1	DC3
DC4	Yes	7	DC9=0.14 / DC4=0.12	DC4=0.5	DC4=0.5	DC9
DC5	Yes	1	DC5=0.14	DC5=0.5	DC5=1	DC5
DC6	Yes	3	DC6=0.11	DC6=0.5	DC6=0.5	DC6
DC7	Yes	6	DC7=0.16	DC7=0.5	DC7=1	DC7
DC8	Yes	2	DC8=0.16	DC8=0.5	DC8=1	DC8
DC9	Yes	7	DC9=0.14	DC9=0.5	DC9=0.5	DC9

A fair comparison with a commercial intra-cell diagnosis tool has been performed by using the same characterization data. This tool is non-probabilistic and provides the list of all suspects obtained after diagnosis with a ranking and a matching score. Results achieved with the same defect injection campaign in the same gate of circuit c2670 are reported in Table V. The first three columns are identical to those in Table IV. The fourth column gives the number of identified defect candidates. The fifth column shows the ranking of the injected defect (when it is in the list of candidates – NA otherwise) and the matching score. The last column reports the accuracy, i.e. the injected defect is or is not in the list of candidates. From these results, the first comment is that the commercial tool was often unable to provide a ranking among the candidates, thus complicating the decision before PFA. The second (more important) comment is that, in 2 out 9 cases, the injected defect is not in the list of candidates provided by the tool (i.e. results are not accurate). Conversely, our technique with LR, KNN and NB **always** provides the right candidate. This proves the superiority of our approach.

TABLE V. DIAGNOSIS RESULTS WITH A COMMERCIAL TOOL – C2670

Class	Det	#SG	#candidates	Ranking / Matching	Accuracy
DC1	Yes	2	4	No ranking / 100%	Yes
DC2	Yes	3	3	No ranking / 100%	Yes
DC3	Yes	2	4	No ranking / 100%	Yes
DC4	Yes	7	0	NA / 100%	No
DC5	Yes	1	3	No ranking / 100%	Yes
DC6	Yes	3	3	No ranking / 100%	Yes
DC7	Yes	6	1	1 / 100%	Yes
DC8	Yes	2	1	1 / 100%	Yes
DC9	Yes	7	0	NA / 100%	No

Table VI summarizes the results obtained for a set of ISCAS'85 benchmark circuits. For each learning algorithm, we report the percentage of cases in which the injected defect was identified as the best candidate (with the highest probability) by the proposed diagnosis technique. Percentage lower than 100% means that in some cases, the injected defect was not identified as the best candidate. **However, in all these cases, the injected defect was always identified as a candidate, but in second or third position** (except for SVM). From this standpoint, we can consider that **the diagnosis accuracy of our technique when using LR, KNN and NB algorithms is**

100%. This important result leads to the conclusion that, during diagnosis of customer returns and before PFA, it will be possible to use our technique with different learning algorithms to assert the confidence level of the achieved intra-cell diagnosis results. The last column in Table VI shows results obtained for each circuit with the commercial intra-cell diagnosis tool. The first value is the average number of candidates (resolution). The second value is the percentage of cases in which accuracy is good. **This value must be compared to 100% obtained by our technique with LR, KNN and NB**. As no ranking was most of the time provided by the commercial tool, it was not possible to calculate the percentage of cases in which the injected defect was identified as the best candidate (as done in columns 2 to 5).

TABLE VI. OVERALL DIAGNOSIS RESULTS

Circuit	LR	KNN	NB	SVM	Com. Tool
C880	84.6%	100%	100%	100%	2 /100%
C1355	44.2%	100%	100%	77%	3 / 96%
C2670	46%	100%	100%	77%	2 / 84%
C3540	72%	84%	84%	62%	3 / 100%
C5315	85.1%	96%	96%	88.8%	2 / 97%
C7552	44.4%	100%	100%	77%	3 / 90%

As can be seen in Table VI, and in accordance with the results shown in Table II, we can assert that KNN and NB work perfectly. LR is still efficient but less accurate and with a higher variability in the results. SVM works efficiently (especially for c880) but is a bit less informative due to its non-probabilistic nature and inherent property. The commercial tool provides results with a much lower resolution and accuracy.

The CPU time taken by the proposed diagnosis flow to provide a list of good defect candidates is always very low (few seconds) and does not depend on the circuit size (except the CUT simulation but this phase is done only once and in just few seconds). Only the number of suspected cells obtained after logic diagnosis may have an impact on the CPU time, but in a slight manner. In fact, the most time-consuming part of the flow (few hours) is the characterization phase, but this phase is also done only once and is not correlated with the circuit size.

V. CONCLUSION AND FUTURE WORK

In this paper, we have addressed the problem of diagnosing failures that occur during mission mode of SoCs. We have presented a novel intra-cell diagnosis approach that uses supervised learning algorithms to produce a minimum set of candidates at transistor level. Results carried out on benchmark circuits have shown that this approach can lead to an accurate localization of the root cause of observed failures.

Above efficient experimental results have been achieved with a preliminary version of the learning-guided diagnosis flow which is not yet ready to be used in an industrial environment. Further developments have to be done to address several missing aspects. First, layout information has to be used to refine the list of defects that are considered during training data preparation. By this way, only realistic defects will be assumed during the whole process, thus increasing diagnosis efficiency. Next, simple defects modeled by stuck-at or stuck-open faults have been assumed in our experiments. In the case of industrial circuits, more complex (e.g. resistive or bridge) defects modeled by delay or bridging faults will need to be considered. Similarly, in-field failure mechanisms related to premature aging and that have to be considered in the context of customer returns, such as NBTI or HCI, will need to be appropriately taken into account by considering representative defect models that already exist and are used in industry (essentially resistive opens and shorts). Note that all the above aspects will probably not impact the format of data instances. Another point is that unique test conditions have been assumed in our experiments. In the context of mission mode failure diagnosis, multiple test conditions with various PVT corners will also need to be considered. Importantly, we will also need to compare our results with those obtained with industrial in-house tools [3–14]. We will also investigate additional learning algorithms and related learning parameters. Finally, we will perform experiments on large-size benchmark and industrial circuits from ST investigated as customer returns.

REFERENCES

[1] N. Sumikawa, D. Drmanac, Li-C. Wang, L. Winemberg, and M.S. Abadir, "Understanding Customer Returns From A Test Perspective", IEEE VLSI Test Symposium, Pages 2-7, 2011.

[2] Y. Xue, X. Li, R. D. Blanton, C. Lim, and M. Enamul Amyeen, "Diagnosis Resolution Improvement through Learning-Guided Physical Failure Analysis", Proc. IEEE International Test Conference, 2016.

[3] A. Ladhar and M. Masmoudi, "Efficient and Accurate Method for Intra-gate Defect Diagnoses in Nanometer Technology and Volume Data", IEEE/ACM Design, Automation & Test in Europe, pp. 988-993, 2009.

[4] Li-C. Wang, "Data Learning Based Diagnosis", ACM/IEEE Asia and South Pacific Design Automation Conference (ASP DAC), pp. 247-254, 2010.

[5] A. Bosio, P. Girard, S. Pravossoudovitch, A. Virazel, "A Comprehensive Framework for Logic Diagnosis of Arbitrary Defects", IEEE Transactions on Computers, Vol. 59, No 3, pp. 289-300, March 2010.

[6] Y. Benabboud, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, O. Riewer, "A Comprehensive System-on-Chip Logic Diagnosis", Proc. IEEE Asian Test Symposium, pp. 237-242, 2010.

[7] L. M. Huisman, "Diagnosing Arbitrary Defects in Logic Designs Using the Single Location At a Time (SLAT)", IEEE Trans. on CAD, Vol. 23, No 1, pp. 91, 2004.

[8] S. Holst and H-J. Wunderlich, "Adaptative Debug and Diagnosis Without Fault Dictionaries", Proc. IEEE European Test Symposium, pp. 7-12, 2007.

[9] S. Venkataraman and S. B. Drummonds, "Poirot: Applications of a Logic Fault Diagnosis Tool", IEEE Design & Test of Computers, Vol. 18, No 1, pp. 19, 2001.

[10] D. Appello, V. Tancorre, P. Bernardi, M. Grosso, M. Rebaudengo and M. Sonza Reorda, "Embedded Memory Diagnosis: An Industrial Workflow", Proc. IEEE International Test Conference, pp.1-9, 2006.

[11] A. Ney, A. Bosio, L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel and M. Bastian, "A History-Based Diagnosis Technique for Static and Dynamic Faults in SRAMs", Proc. IEEE International Test Conference, paper 3.2, 2008.

[12] K. Huang, H. Stratigopoulos, and S. Mir, "Fault Diagnosis of Analog Circuits Based on Machine Learning", IEEE/ACM Design, Auto. & Test in Europe, 2010.

[13] C. Zhang, Y. He, L. Yuan, and S. Xiang, "Analog Circuit Incipient Fault Diagnosis Method Using DBN Based Features Extraction", IEEE Access, Vol. 6, April 2018.

[14] Z. Sun, A. Bosio, L. Dilillo, P. Girard, A. Todri, A. Virazel, and E. Auvray, "Effect-Cause Intra-cell Diagnosis at Transistor Level", Proc. IEEE International Symposium on Quality Electronic Design, pp. 460-467, 2013.

[15] J. E. Nelson, W. C. Tam, and R. D. Blanton, "Automatic Classification of Bridge Defects", IEEE International Test Conference, pp. 1–10, 2010.

[16] X. Ren, M. Martin, and R. D. Blanton, "Improving Accuracy of On-Chip Diagnosis via Incremental Learning", Proc. IEEE VLSI Test Symp., pp. 1–6, 2015.

[17] Y. Huang, W. Yang, and W. Cheng, "Advancements in diagnosis driven yield analysis (DDYA): A survey of state-of-the-art scan diagnosis and yield analysis technologies", Proc. IEEE European Test Symposium, pp. 1–10, 2015.

[18] R.J. Tikkanen, S. Siatkowski, Li-C. Wang and M.S. Abadir, "Yield Optimization Using Advanced Statistical Correlation Methods", IEEE Int'l Test Conf., 2014.

[19] H. Wang, O. Poku, X. Yu, S. Liu, I. Komara, and R. Blanton, "Test- Data Volume Optimization for Diagnosis", Proc. ACM/IEEE Design Auto. Conf., pp. 567, 2012.

[20] Y. Xue, O. Poku, X. Li, and R. D. Blanton, "PADRE: Physically- Aware Diagnostic Resolution Enhancement", Proc. IEEE Int'l Test Conference, 2013.

[21] https://machinelearningmastery.com/

[22] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques", Informatica, Vol. 31, No 3, pp. 249-268 249, 2007.

[23] *http://scikit-learn.org/stable/user_guide.html*