Defect Diagnosis Techniques for Silicon Customer Returns

Patrick Girard, Alberto Bosio, Aymen Ladhar, Arnaud Virazel

Chapter 1
Defect Diagnosis Techniques for Silicon Customer Returns

Patrick Girard, Alberto Bosio, Aymen Ladhar, Arnaud Virazel

Abstract

This chapter provides an overview of the various approaches and techniques proposed so far for defect diagnosis in silicon customer returns. It focuses on diagnosis of defects in **logic blocks of SoCs**. After some backgrounds on test and fault diagnosis, the chapter presents the various test scenarios used in practice during customer return diagnosis. A discussion on the quality required by the test sequences used during customer return is also proposed. Then, the chapter reviews the state-of-the-art techniques existing to identify defects at the cell level (called *intra-cell* or *cell-aware diagnosis*). A summary of conventional approaches is first proposed. Then, the latest Machine Learning (ML) -based cell-aware diagnosis techniques are reviewed. Effectiveness of existing ML techniques is shown through industrial case studies and corresponding diagnosis results in terms of accuracy and resolution. The chapter ends with a discussion on the future directions in this field.

1 Introduction

Modern electronic systems are composed of complex Systems on a Chip (SoCs) which are made of heterogeneous blocks that comprise memories, digital, analog and mixed-signal parts, etc. These SoCs demand a huge amount of knowledge and expertise to be designed, fabricated and embedded on the final support with the required levels of functionality and reliability. To guarantee their correct behaviour and hence fit a given quality level required by the application standard (e.g., automotive, avionic, etc.), SoCs pass through a comprehensive test flow (functional, structural, parametric, etc.) at the end of the manufacturing process. SoCs that pass the test flow are further used in the field by the target application.

Despite the high-quality level of the test flow used during manufacturing test, SoCs may still fail in the field. Thus, in an attempt to identify the source of failures and avoid their re-occurrence in next generation of products, each defective SoC (referred to as "*customer return*") is always sent back to the manufacturer who is in charge of analyzing the device to determine the root cause of failures. This is

Patrick Girard, Arnaud Virazel
LIRMM, University of Montpellier / CNRS, 34095, Montpellier, FR
E-mail: [firstname.lastname]@lirmm.fr
Alberto Bosio
Univ. of Lyon, ECL, INSA Lyon, CNRS, UCBL, CPE Lyon, INL, 69130 Ecully, FR
E-mail: alberto.bosio@ec-lyon.fr
Aymen Ladhar
STMicroelectronics, 38920 Crolles, FR, e-mail: aymen.ladhar@st.com

particularly true for safety-critical applications. A customer return is a circuit that passed the entire manufacturing test flow but failed on the customer's side [1]. The two main causes of a customer return are test escape during manufacturing test or latent defect mechanisms during lifetime [2]. Latent defects cause two types of failures: i) early-life failures that are not exposed during manufacturing tests, but that are degraded due to electrical and thermal stress during in-field use, and ii) failures caused by various wear-out mechanisms. Wear-out or aging manifesting as progressive performance degradation is induced by various mechanisms such as Negative-Bias Temperature Instability or Hot-Carrier Injection. All these failures that occur in the field are the most critical as they may result in catastrophic consequences.

When a customer return is identified, it is important to reproduce the failure mechanism in the failure analysis lab of the manufacturer with the appropriate test conditions (temperature and voltage) and the original test flow. In case of test escape, efforts must be spent on finding new test patterns that will exhibit the failure in the same test conditions. In case of latent defect, the task will often succeed and a diagnosis program made of several routines is used to identify, step by step, the failing part and, finally, the suspected defect(s). Each routine coincides with the application of a diagnosis algorithm at a given hierarchy level (system, core and cell levels) [3].

Diagnosis is a software-based method that analyses the applied test sequences, the tester responses, and the circuit structure (possibly with layout information) to produce a list of candidates that represent the possible locations and types of defects within the defective circuit. 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 by the diagnosis. An accurate diagnosis with perfect resolution (i.e., one candidate which is the real defect) is the ideal case.

Diagnosis is usually followed by *Physical Failure Analysis* (PFA), a time-consuming process for exposing the defect physically (owing to special techniques and tools such as acoustic microscopy, x-ray imaging, transmission electron microscopy, photon emission microscopy, laser-induced voltage alteration, thermal imaging, etc.) in order to characterize the failure mechanism. Due to the high cost and destructive nature of PFA, diagnosis accuracy and resolution are of critical importance. In practice, it is very uncommon to perform PFA on any defect with more than five candidates [4]. This ensures that the likelihood for uncovering the root-cause of failure is maximized when performing PFA.

With more defects inside standard cells at leading-edge technology nodes [5], *cell-aware* (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 [6-7]. 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 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 [7].

Unfortunately, today's diagnosis resolution is typically far from ideal due to SoC complexity. Especially with the advent of very deep submicron technologies (i.e., 7 nm), a high resolution (very few or one candidate) is not always reachable by existing CA logic diagnosis tools based on conventional methods [8]. For this reason, considerable efforts have been spent on improving resolution by using machine learning techniques, mainly through the extraction of features that allow correct candidates (those that correctly represent defect locations) to be distinguished from incorrect ones [2,4,9,10,11].

Even though they are efficient, these techniques address volume diagnosis for yield ramp-up, which is a different problem than *defect diagnosis of customer returns*. Indeed, during volume diagnosis for yield improvement, a lot of 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, 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 any other similar chips used in the same conditions (application, environment, workload). For this reason, approaches existing for volume diagnosis cannot be used in a straightforward manner for customer returns.

Historically, conventional approaches based on *critical path tracing* or *fault simulation* were used in industry for defect diagnosis of customer returns. However, 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. Most of these techniques are based on supervised learning, because it naturally aligns with the common practice of training with labeled historical data and usually performs well in industrial diagnosis tasks [12].

This chapter reviews the latest developments in the field of customer return diagnosis based on ML. It is organized as follows. Section 2 gives some background on test and diagnosis in the context of customer return analysis. Section 3 first explains how customer returns are usually (re-)tested for diagnosis purpose and what are the limitations. Next, a proposal of best practices for customer return test pattern generation is done and discussed. Section 4 summarizes the state-of-the-art in the field of defect diagnosis for customer returns. Both conventional approaches and ML-based techniques are discussed. Section 5 presents some industrial case studies performed with one of the latest ML-based diagnosis techniques. Section 6 concludes the chapter and draws some conclusions.

2 Background on Test and Fault Diagnosis

During the manufacturing process of Integrated Circuits (ICs), defects can occur in the physical structure of the IC, subsequently leading to erroneous behaviors. The role of testing is to detect ICs affected by defects and discard them from the set of

ICs sent to the customers. Moreover, testing is also important to collect as much information as possible to be further exploited during fault diagnosis, aiming at understanding the root causes of the observed failures. This section provides some backgrounds on defects, test and fault diagnosis.

2.1 From Defects to Failures

Physical defects like shorts and opens may occur during any single step of the fabrication process. These defects can be randomly caused by contaminations or due to systematic process-design interactions [13]. In modern deep submicron technologies, defects appear not only in the cell interconnection (inter-cell defect), but also inside the cell itself (intra-cell defect) [14-15]. This is caused by the reduced circuit sizes, the use of new complex process technologies, new materials and the increasing number of vias and contacts. For example, Fig. 1.1 depicts a short defect affecting a four-input AOI cell made of 48 transistors in a 32 nm STMicroelectronics design [16].

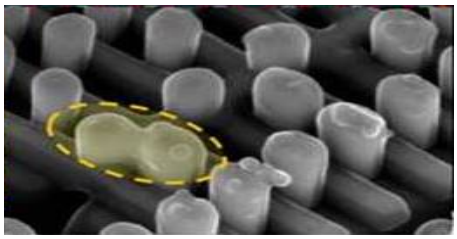

Fig. 1.1 Short defect affecting a logic gate

The representation of a defect is a *fault model*, an anomalous physical condition that may lead to an *error*. An error is the exhibition of a fault in a system that might or might not be propagated and, in this last case, give rise to *failure* [17].

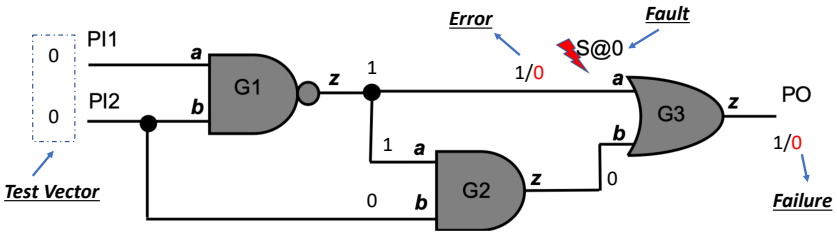

Fig. 1.2 Fault, Error, Failure example

Figure 1.2 summarizes the above concepts through a simple example. The IC is a combinational circuit composed of three gates (G1, G2 and G3), two primary inputs (PI1 and PI2) and one primary output (PO). The circuit is affected by a defect

represented by a Stuck-at-0 fault model (S@0) located at the input *a* of G3 (i.e., the behavior of the defect can be modeled as a logic value always set to '0' at input *a* of G3). The circuit is stimulated by the logical values applied to PIs ("00"). The applied values lead to have '1' at input *a* of G3. However, because of the S@0, the value is actually set at '0'. This situation is represented as '1/0', where '1' is the expected value and '0' is the wrong valued induced by the fault. Since the expected value is different from the wrong one, the fault is said to be *sensitized* and the wrong value is the error induced by the fault. In the example, the error is propagated through G3 and reaches the PO. At this point, the error becomes observable and leads to a failure. The set of applied logic values is called *test vector* since it can detect the presence of the S@0 affecting input *a* of G3. Several fault models exist and are used in industry. Among them, the most popular are the following:

- *Stuck-at Fault Model* [17]: the logic value of a given net appears to be stuck at a constant logic value ('0' or '1'), referred to as stuck-at-0 or stuck-at-1;
- *Transition Fault Model* [17]: the transition from a given logic value V to the opposite logic value V' at the output of a gate is delayed. In this case, the delay of the gate is changed, and is assumed to be large enough to prevent a passing transition from reaching any output of the circuit within the clock period. Two types of transition fault are defined: slow-to-rise (slow transition from logic '0' to logic '1') and slow-to-fall (slow transition from logic '1' to logic '0');
- *Bridging Fault Model* [17]: usually modeled at the gate or transistor level, it represents a short between a group of signals. The logic value of the shorted net can be modeled as a 1-dominant (OR bridge), 0-dominant (AND bridge), or indeterminate, depending upon the technology in which the circuit is implemented.
- *Cell-Aware Fault Model* [15]: it represents a defect inside a given logic cell. The faulty behavior depends on the logic cell transistor-level structure. This fault model has to be defined every time a new technology library is implemented.

2.2 Testing

IC Testing consists in applying a set of test vectors (forming a so-called *test sequence*) in order to detect the highest number of faults as possible. The test sequence quality is measured owing to the following metrics:

- *Fault Coverage* (FC): this is the ratio between the number of detected faults and the total number of faults. Ideally, the FC has to be equal to 1;
- *Defect Coverage* (DC)**:** similar to the FC, it gives the ration between the number of detected defects and the total number of defects. It is important to mention that a high FC does not automatically implies a high DC. For example, it

is shown in [14-15] that intra-cell defects cannot be detected by using test approaches based on classical fault models such as stuck-at or transition faults;

- *Test Coverage* (TC): this is the ratio between the number of detected faults and the total number of detectable faults. Redundant faults are omitted by this metric. This is the main metric used in industry to qualify a test sequence.
- *Test Length* (TL)**:** is the number of test vectors composing the test sequence. The higher the length, the higher the cost of test in terms of test time and test data volume.

The test sequence is generated by using a commercial EDA tool, known as Automatic Test Pattern Generator (ATPG). In short, an ATPG aims at generating a test sequence that maximizes the FC while minimizing the TL. The description of ATPG architectures and algorithms is out the scope of this chapter. The reader can refer to [17] for detailed information.

IC testing is always executed after the manufacturing process. It allows to quantify the quality of the manufacturing process itself through the *Yield* defined as:

$$Yield = \frac{\#Good}{\#Total}$$

where #Good is the number of fault-free devices over the total number of manufactured devices. Every time a new technology node and its manufacturing process is used, the yield loss can be very high (yield < 40%). The process of identifying yield losses, quantifying and improving them is referred to as *yield learning*. Testing and Fault Diagnosis play a crucial role for yield learning.

Even when the manufactured IC passes the testing phase and thus is used in-the field, testing may be still needed. This is the case of safety-critical applications, like automotive or avionic, where all the hardware components have to be continuously tested to ensure the correct behavior. Any IC that fails in the field can be considered as a *customer return*.

2.3 Fault Diagnosis

Fault diagnosis is the process applied to a failing IC to shedding light into the actual defect and then apply corrective actions to prevent failure re-occurrence in next generation products. We can identify two main types of fault diagnosis depending on the scope:

- *High-Volume Fault Diagnosis*: it is applied for yield learning. Indeed, diagnosing the sources of failures assists the designers in collecting valuable information regarding the underlying failure mechanisms, in order to enhance yield through improvement of the manufacturing process and development of new design techniques that minimize the failure rate [2].
- *Customer Return Fault Diagnosis*: it is applied on a given IC to determine the root cause of failures that have occurred in the field. In this scenario, failures

are not easy to reproduce in the manufacturer's lab as the real mission conditions and executed workload are unknown and cannot be exhaustively modeled.

In the rest of this chapter, we mainly refer to fault diagnosis of customer returns. Fault diagnosis can be applied at different levels depending on the complexity of the IC. Todays' ICs are complex devices that typically consist of independent and heterogeneous blocks, and each block may comprise memory, digital circuits, Analog and Mixed-Signal (AMS) circuits, etc. (see Fig. 1.3).

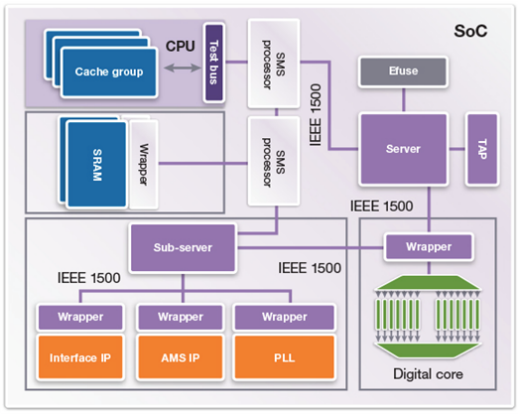

Fig. 1.3 Heterogeneous System-on-Chip (courtesy of Synopsis)

The first level of fault diagnosis is thus the system level that aims at determining the failing block. The second level of fault diagnosis is the block level. Depending on the nature of the failing block, i.e., analog, digital, memory, etc., different fault diagnosis techniques (as well as test sequences) have to be applied. This chapter focuses on *digital circuit blocks* only. Memory and Analog & Mixed-Signal (AMS) fault diagnosis is out of the scope of this chapter but the reader can refer to [18,19] for more details. The third level of fault diagnosis is the cell level, called *cell-aware fault diagnosis*. It consists in identifying defects within a logic cell.

The key metrics characterizing fault diagnosis performance are:

- *Resolution*: $R = \frac{\#C}{\#S}$ defined as the ratio of identified candidates (#C) over the total number of possible suspects (#S). The smaller the R the better the fault diagnosis. Ideally, fault diagnosis should provide a single suspect. The definition of suspects and candidates depends on the fault diagnosis level. At system-level, a candidate is a block, while at digital circuit level a candidate can be a gate, a net or even a transistor in the case of cell-aware faults;
- *Accuracy*: the fault diagnosis is accurate if the physical defect responsible for the observed failure is indeed in the list of identified candidates. Accuracy $A = \frac{\#Correct}{\#Diagnosis}$ can be defined as the number of correct diagnosis (#Correct) over a given set of experiments (#Diagnosis).

2.3.1 System-Level Fault Diagnosis

Existing effective system-level fault diagnosis techniques either apply only for boards [20] or, if they apply for SoCs, then they target the digital part of the SoC [21]. In particular, the individual blocks can be accessed and tested in isolation so as to identify the faulty block. The problem is exacerbated when the failure occurs during the mission mode. In this case, the SoC operates in functional mode, so there is no information that is logged at block level (i.e., the functional stimuli applied to each block may be unknown). A promising solution is to use machine learning. In the literature, machine learning techniques have been already exploited for system-level diagnosis, but only for boards [22].

2.3.2 Digital Block-Level Fault Diagnosis

Regarding digital blocks, there exist commercial tools offered by EDA vendors, such as Synopsys and Siemens (formerly Mentor Graphics), and many research works. All existing approaches are based on the "Cause-Effect" [23] or the "Effect-Cause" [24] paradigms. The "Cause-Effect" paradigm requires a pre-computed fault dictionary, which can be obtained by simulating targeted faults in a specific design with a given set of test vectors. During diagnosis, a search on the fault dictionary is performed to determine a set of candidates that explain the observed errors. On the other hand, the "Effect-Cause" diagnosis approach determines the set of candidates by using a back-tracing algorithm [24]. The algorithm is executed starting from each failing primary output and traces back through circuit nets to reach primary inputs. Each traced net is classified as a candidate. Compared to "Cause-Effect" diagnosis, the "Effect-Cause" approach does not require any pre-computed fault dictionary, thus it is independent of a targeted fault model. However, "Effect-Cause" diagnosis may require a very high computational time, which is proportional to the circuit complexity (e.g., number of gates and test vectors). Hence, the "Effect-Cause" approach may not be appropriate for large designs.

Irrespective of the adopted paradigm (i.e., Cause-Effect or Effect-Cause), the result is a list of nets (e.g., connections between logic cells or Flip-Flops) that are declared as candidates. Even if only one candidate is included in the list, it is often not accurate enough to isolate the defect causing the error. For example, one cell typically contains many transistors (usually more than 100), and one net could be extended to several metal layers. Without more accurate information of the defect location inside a cell, PFA may fail, that is, the root cause may not be discovered. Hence, it is very important to identify which components within a cell are more likely to be the defective ones so as to successfully perform PFA. This shift in the diagnosis accuracy level is obtained by applying transistor-level diagnosis inside a cell, referred to as *Cell-Aware fault diagnosis*.

2.3.3 Cell-Aware Fault Diagnosis

Previous works on Cell-Aware (CA) fault diagnosis focusing on logic cells can be classified into three approaches. The first approach converts a transistor-level netlist into an equivalent gate-level netlist by means of complex transformations rules [25]. Then, on the equivalent gate-level netlist, any classical fault diagnosis approach can be applied. The main drawback of this approach is that the set of transformation rules depends on the targeted defect and, thereby, the non-modeled defects may not be diagnosed. The second approach is based on the "Cause-Effect" paradigm [25-26]. The transistor-level netlist of a cell is exploited, in order to inject the targeted defects. Therefore, a defect dictionary is created by transistor-level simulations and the defect signatures of all the defects affecting the cells in the library are stored in this defect dictionary. Then, during fault diagnosis the defect signature of all defects affecting a suspected cell is compared with the observed failures to obtain a list of candidates inside the cell. These approaches can be further classified depending on the "accuracy" of the injected defects and the simulation "precision". In [26], a large number of defects are simulated at transistor-level using SPICE. For a given defect, different resistance values are simulated, in order to be as accurate as possible. This approach leads to more precise results but it requires a huge simulation time. To reduce the simulation time and the fault dictionary size while keeping a high resolution, authors in [25] propose to exploit layout information, in order to consider only realistic defects. For example, for each cell, only the realistic, potential net bridging defects and via open defects are extracted and then simulated. Then, the identified set of realistic defects is simulated at transistor-level. The third intra-cell fault diagnosis approach is based on the "Effect-Cause" paradigm [27]. All the existing diagnosis techniques depend on the targeted fault models or defects. In [27], the main goal is to achieve a resolution close to the transistor-level. However, instead of explicitly considering defects at transistor-level, the idea is to exploit the knowledge of the faulty behavior induced by the defects.

Unfortunately, Cell-Aware fault diagnosis resolution is typically far from the ideal due to circuit complexity. For this reason, considerable effort has been spent to improve resolution by using machine learning techniques, initially through the extraction of features that allow correct candidates (those that correctly represent defect locations) to be distinguished from incorrect ones [2,11]. Even though they are efficient, these techniques address volume diagnosis for yield improvement, which is a different problem than fault diagnosis of customer returns (as already mentioned and explained in Section 1). For this reason, these techniques cannot be reused for fault diagnosis of customer returns.

3 Test of Customer Returns for Diagnosis Purpose

The ultimate objective of an IC test engineer in charge of a given IC product is to reach a zero *Defective Parts Per Million* (DPPM) with a reasonable test cost (test time and test data volume). To this purpose, several test sequences made of test patterns are generated during test preparation to target all types of defects (static, dynamic) by using different (i.e., slow or fast) test clock schemes. Note that a test

pattern can be composed of only one vector to deal with static faults, e.g., stuck-at faults, or of two vectors (two-vectors test pattern) to deal with dynamic faults, e.g., transition faults. These test patterns are usually generated in an incremental manner to avoid multiple detection of the same defects and hence reduce test costs. In this section, we first give an example of a typical test scenario used in production (manufacturing) industrial test. Next, we discuss some limitations of this type of scenarios in terms of diagnosis accuracy and resolution. Finally, we suggest some of the best test practices to be applied for custom return diagnosis.

3.1 Typical Test Scenario

When a custom return occurs and is sent back to the manufacturer, the first step is to reproduce the failure mechanism by reusing the initial manufacturing test program and collect the failure files to be diagnosed. Therefore, before proceeding with fault diagnosis, it is important to understand the ATPG process as well as its implementation into the test program. This knowledge is valuable and helps to enhance the diagnosis process and accelerate the time needed to retrieve the silicon failure.

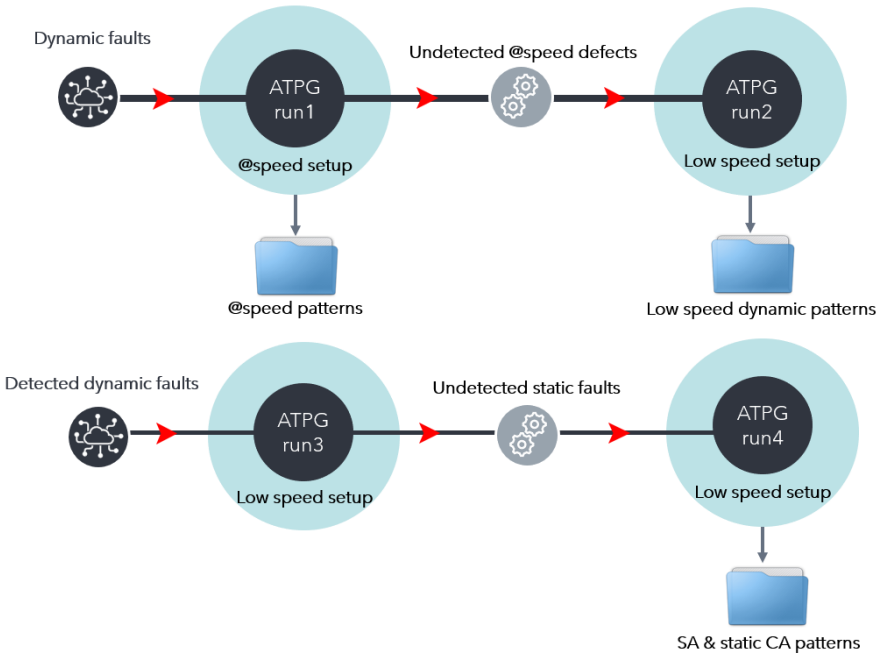

Fig. 1.4 Example of an Industrial Multi-Run ATPG flow

Figure 1.4 gives an example of a multi-run ATPG flow used in industry for screening defects in logic parts of a SoC. It can be seen that several ATPG runs are

implemented at various speeds (low-speed or at-speed) to get different test sequences targeting different types of defects (static and dynamic) with different fault models (stuck-at, transition, bridging and cell-aware). *Static defects* are defects that require one-vector test patterns, called *static test patterns*, to be detected. *Dynamic defects* are defects that require two-vector test patterns, called *dynamic test patterns*, to be detected. Dynamic defects do not modify the functional behaviour of the IC but rather induce some delays that prevent the IC to operate at the desired clock frequency. With the advent of nanometer 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, functional, environmental, etc.).

The flow sketched in Fig. 1.4 shows four ATPG runs implemented to generate three types of test patterns (at-speed, low-speed dynamic, and static test patterns). The first step in this ATPG flow is to generate at-speed test patterns for dynamic defect detection. These patterns target the detection of delay defects either at the interconnect wires or inside cells of the logic part of the SoC. At-speed test is made so that the PLL clock is used instead of the tester clock during the capture phase. This is usually performed owing to an On-chip Clock Controller (OCC). An OCC is the logic inserted into the SoC for controlling clocks during silicon testing on an ATE (Automatic test Equipment). Since at-speed testing requires two clock pulses in capture mode with a frequency equal to the functional clock frequency, these at-speed pulses would need to be provided through I/O pads without OCC. However, these pads are limited in terms of supported maximum frequency. Conversely, an OCC uses an internal PLL for generating test clock pulses and hence allows application of at-speed tests. During static testing, the OCC ensures that only one clock pulse is generated during the capture phase.

Once the dynamic patterns are generated, the undetected dynamic defects are targeted with a second ATPG run (see Fig. 1.4). In fact, it has been proven that even at low frequency some dynamic defects can be still detected. Typically, a stuck-open or a source/drain open defect results from a complete break between circuit nodes that should be connected [28, 29] and has a sequential behavior that requires two-vector test patterns to be detected. It has been shown in [30] that changing the test speed, voltage, and temperature, does not improve the test effectiveness when such type of dynamic defects is targeted. This explains why this type of ATPG runs (low speed setup) is used in industrial test flows to generate low speed dynamic test patterns.

The third ATPG run is a preprocessing step for the fourth one. In fact, the fourth run requires as input a list of all static defects not detected by the dynamic tests. This information can be identified only through fault simulation. To this end, the static faults are fault simulated with the dynamic patterns to determine the set of undetected faults.

At the end of the ATPG flow, three test sequences are generated and then applied sequentially to the Circuit Under Test (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.

3.2 Limitation of Manufacturing Test for Customer Returns

The goal of an ATPG as used for manufacturing test is to detect the maximum number of faults with the minimum number of test patterns. Unfortunately, this may not be adequate for fault diagnosis for the following main reasons:

- Distinguishing between defect candidates during fault diagnosis is achievable only when test sequences are made of patterns that each sensitizes a limited number of faults. Conversely, test patterns generated by conventional ATPG sensitize as many faults as possible to avoid long test time. These two conflicting objectives between test pattern generation for testing purpose and test pattern generation for diagnosis purpose generally leads to poor diagnosability of test sequences generated by an ATPG. Table 1 shows an example where the generation of a test pattern is needed to detect two defects internal D1 and D2 in a NOR2 gate with A and B as inputs and Z as output. As can be seen, any ATPG tool would only generate P1 as it can detect both defects on output Z (Z_{D1} and Z_{D2} both have a value different than the fault-free value on Z). However, these patterns are unable to distinguish between the two defects and hence any diagnostic tool would report D1 and D2 as fault candidates. As can be observed, P2 can detect only D2, and hence is able to distinguish between D1 and D2. The same is true for P3 which is able to distinguish between D1 and D2. Therefore, an ideal test sequence for diagnosis purpose would be made of (P2, P3),

Table 1.1: Example of distinguishing patterns between defect candidates

Pattern	AB	Z	Z_{D1}	Z_{D2}
P1	00	0	1	1
P2	01	0	0	1
P3	10	0	1	0
P4	11	1	1	1

- Generally, there is no full picture on how an actual defect behaves with regards to passing patterns (patterns declared as “pass” during test application). In fact, the purpose of a test pattern is to create a failing excitation of the defect and ensure its propagation to an observable point. This means that the behavior of the defect is known only for the failing patterns and most of the time no information is collected for the passing patterns. Let us consider again the example in Table 1.1. An ATPG tool will generate P1 and eventually P2 to detect and distinguish between D1 and D2. Consequently, P3 and P4 will not be generated However, the knowledge of this information would be important to improve the diagnosis accuracy since it would provide additional indications on how the defect behaves with a complete set of test stimulus.

- Scan chain diagnosis of ICs with an embedded test compression mechanism usually leads to low diagnostic capabilities [31]. This is the case of most ICs nowadays. In fact, compressed test patterns and compacted test responses are widely used in industry to reduce test data volume and scan input/output requirements. The use of test response compaction negatively impacts fault diagnosis since errors in responses due to defects which are captured in scan cells are not directly observed. This means that using typical chain test patterns is not enough to distinguish between failing scan chains, and thus generating additionally distinguishing patterns is needed to improve the resolution.
- Test truncation is a widely used technique in production test. Indeed, not all the test responses of a failing CUT are collected, and this is limited to a predefined number of failing patterns. Indeed, recording the failing patterns and their observation points is a time-consuming step especially when the collected failures is huge. Proceeding with test truncation can reduce the test time since not all failing patterns are recorded. However, this procedure has a negative impact on the diagnostic quality as exploiting only a subset of the failing patterns to retrieve the fault candidates limits the diagnostic tool capabilities.

The above-mentioned limitations of manufacturing test can prevent the successful failure analysis of a customer return. In the following subsection, we first explain how to adapt such a test flow for a customer return. Then, we present different techniques to generate *diagnostic patterns* (also referred to as distinguishing patterns).

3.3 Best Practices for Customer Return Test Pattern Generation

Two approaches exist to improve diagnosis quality. The first one is to improve the efficiency of diagnostic algorithms, which is usually done by CAD vendors. The second one is to improve the test sequence quality in such a way that more valuable information can be collected during test [32]. In both cases, the goal is to make so that diagnostic tools can easily retrieve the defect location with a minimum set of candidates. In the rest of the section, we suggest some of the best practices to be used during test to this purpose, as well as advice on how to generate new diagnostic test patterns for silicon costumer returns.

Before testing a customer returns for diagnosis purpose, two modifications on the test program must be performed. The first one is to include all test patterns generated by the ATPG into the test program and make so that the test will not be stopped at the first failing pattern. The second one is to remove or to increase the truncation value. Once these two modifications have been done, the test program can be started, followed by the first run of fault diagnosis. Depending on the diagnostic results, i.e., in case of low diagnostic resolution, a diagnostic pattern generation process can be launched. In the following, some scenarios that require the generation of new diagnostic patterns are detailed.

- *Diagnostic ATPG for cell internal defects*. In case of a suspected cell internal defect, information provided by the application of the entire failing and passing test patterns applied at the inputs of the defective cell is crucial to efficiently find out the cell internal defect. The best way to get this information is to generate a so-called *cell exhaustive test*. This test applies all the possible combinations at the cell inputs. These combinations can be static or dynamic. Static combinations include logic values '0' and '1', whereas dynamic combinations include rising and falling transitions. By applying such type of test, all non-equivalent cell internal defects can be distinguished. It is recommended to apply the dynamic part of the test with different frequencies (low speed, at-speed) to get information about the failure mechanism with respect to the clock frequency.
- *Diagnostic ATPG for interconnect open defects*. In case of a suspected interconnect open defect, it is important to generate test patterns targeting each segment composing this interconnection and ensure a different fault propagation through different primary outputs. Figure 1.5 shows a net with two metal layers *(M2 and M3)* and five segments *(s1 to s5)*. An open defect located on segment *s2* disconnects two cells *(C2 and C3)*. To help the diagnostic tool to report the defective location, it must be forced to select all possible fault propagation paths during test pattern generation *(C2 only, C3 only, C4 only, C2 and C3, C2 and C4, C3 and C4, all fanout cells: C2 and C3 and C4)*. With this additional information, the diagnostic tool will be able to locate the failing segment more accurately. In this example, only an open defect on s2 can propagate through C2 and C3, and not through C4.

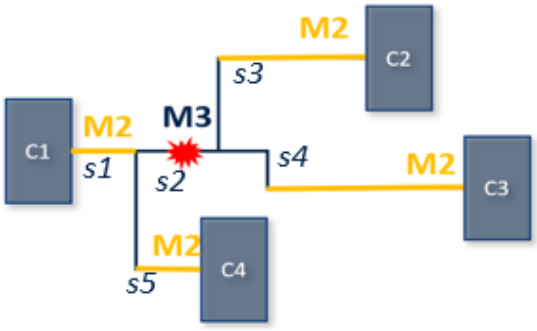

Fig. 1.5 Example of interconnect open defect

- *Diagnostic ATPG for bridging defects*. In case of a suspected bridging defect, it is important to have the list of bridging pairs, extracted from the layout database, to generate additional diagnostic patterns. The goal in this case is to test an opposite value on each bridging pair separately ('0' on one net and '1' on the other one, or vice-versa) and avoid testing several bridging pairs at the same time. In Fig 1.6, let us assume that two possible bridges

(net1-net3 and net2-net4) are reported by the diagnostic tool. A new diagnostic pattern must be generated to distinguish between these two defects. Any ATPG tool would try to target net1-net3 and net2-net4 with the same test pattern, which is not appropriate for fault diagnosis. A diagnostic test pattern generation will try to test net1-net2 by delivering an opposite value on these two nets and ensure that net2 and net4 have the same logic value. With this additional information, a diagnostic tool will be able to differentiate these two potential bridging defects and identify the actual failing one.

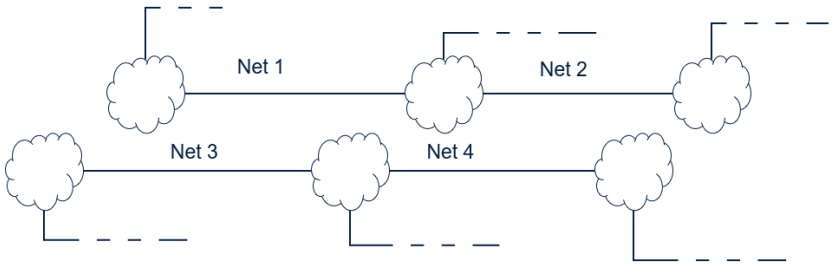

Fig. 1.6 Example of bridging defects

- *Diagnostic ATPG for chain defects*. In case of a scan chain failure in a design using test compression, it is crucial to generate additional patterns that can distinguish between two or more faults. These patterns are called *chain diagnostic patterns*. These patterns are not used during production (manufacturing) test but can be generated and applied for diagnostic purpose, especially in the case of a customer return. Figure 1.7 shows an example of a test program in which additional chain diagnostic patterns are inserted. For a faulty-free CUT, the flow starts by testing the chain integrity followed by the ATPG test. In case of a failure, additional chain diagnostic patterns are used. The main drawback of using chain diagnostic patterns, however, is the increased test time.

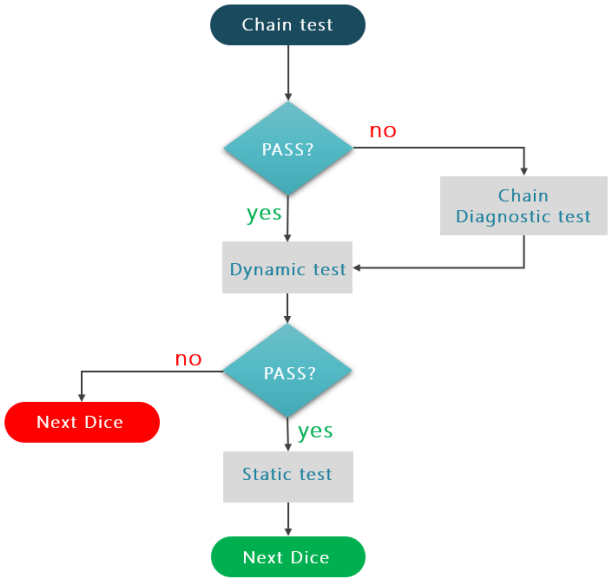

Fig. 1.7 Example of a manufacturing test program

4 Defect Diagnosis Techniques for Customer Returns

A conventional diagnosis flow of ICs is shown in Fig. 1.8. The circuit is first designed according to specifications and supported by a large set of design and verification tools. When the circuit design is completed and verified, the circuit can be manufactured. After manufacturing, all parts of the circuit (logic, memory, analog, etc.) must be tested by different methods. This step is mandatory for any manufacturer to ensure a high quality of products before delivery to the customer. If the test reveals an abnormal behavior (i.e., test fail), information from the ATE are subsequently exploited during the diagnosis step to identify the source of failure and take necessary actions to correct the design or modify the manufacturing process. The final objective of this step is to improve the yield ramp-up. Conversely, if the test is passed, the IC is sold and embedded in a system.

During the lifetime of the IC (e.g., in field), and especially during mission mode, the IC may fail by providing incorrect and unexpected responses to functional stimuli (cf. Fig. 1.8). In case such erroneous behavior is observed, the IC is sent back to the manufacturer and is considered as a customer return. It is then tested with the test sequences initially used after manufacturing process in order to reproduce the failure occurrence. In case this test phase does not reveal any error, then a test pattern enhancement step is launched to produce new test patterns able to exhibit the

erroneous behavior. Obtained test data log are then analyzed during the diagnosis step to finally identify the source of failure.

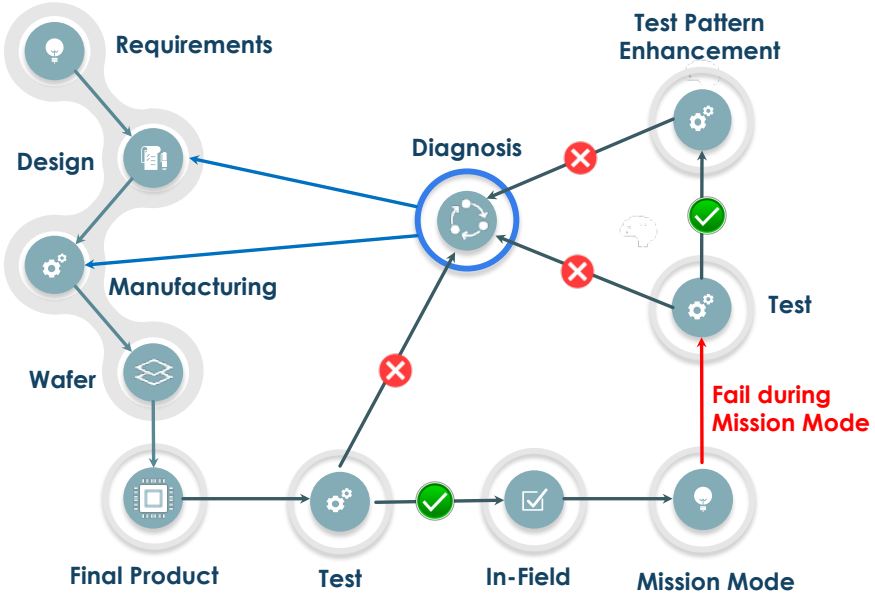

Fig. 1.8 Diagnosis flow of integrated circuits

Fault diagnosis must be able to identify the fault location and the fault type. This information is further used to guide the failure analysis process in pinpointing the defect on silicon and identifying the root cause of failure. For example, the root cause may be a problem in the physical implementation process or a misalignment in the production masks. The results of these investigations during PFA can be further used to optimize the design and manufacturing processes, and avoid reoccurrence of the failure in next generation products.

The next sub-sections give details on conventional diagnosis approaches and advanced methods based on Machine Learning.

4.1 Conventional Approaches

Conventional diagnosis approaches aim at identifying the list of suspected nets and gates (or cells) in a digital circuit. Two main paradigms can be distinguished: cause-effect and effect-cause. The "Cause-Effect" paradigm needs a pre-computed fault dictionary, which can be obtained by simulating the targeted faults in a specific design with a set of test patterns [23]. From this fault dictionary and the set of failing and passing test patterns obtained after test application, a diagnosis tool is able to identify a list of suspects (i.e., fault or defect candidates). Compared to "Cause-

Effect" diagnosis, the "Effect-Cause" approach does not require any pre-computed fault dictionary [33]. It is based on Critical Path Tracing (CPT) and proceeds by back tracing sensitive paths in the circuit from every failing output identified after test application to identify the suspected faults.

For each conventional diagnosis approach, there are two levels of diagnosis: inter-cell and intra-cell. In the case of inter-cell diagnosis, each candidate is a circuit net (i.e., a connection between cells) or a cell. On the other hand, for intra-cell diagnosis, each candidate is a net inside one cell. Figure 1.9 gives an example of diagnosis report obtained from an inter-cell diagnosis (left part of Fig. 1.9) and an intra-cell diagnosis (right part of Fig. 1.9).

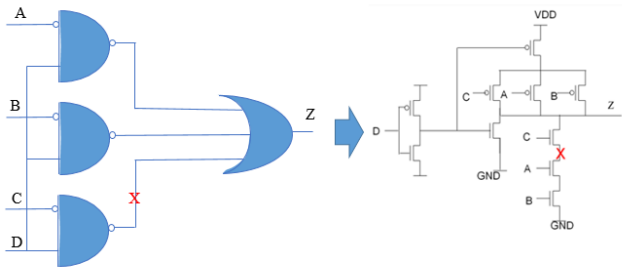

Fig. 1.9 Inter-cell (left) v.s. intra-cell (right) diagnosis results

The next sub-sections give details on "Cause-Effect" and "Cause-Effect" diagnosis state-of-the-art approaches.

4.1.1 Diagnosis Using Fault Simulation

Inter-cell diagnosis

A typical cause-effect diagnosis method is shown in Fig. 1.10.

To build a fault dictionary, a specific fault model, such as the stuck-at fault model or the transition fault model, is first assumed. A dictionary which records the responses of all test patterns for all possible faults is generated by intensively performing fault simulation. This dictionary is referred to as the fault dictionary. Once the fault dictionary is built, the failure syndrome of the failing device is examined using the fault dictionary. The fault(s) whose test response matches the observed failure during test application will be considered as fault candidate(s).

The time for constructing the fault dictionary is equal to the time for fault simulating all test patterns for all faults considered for the circuit under diagnosis, which is acceptable as it is done only once. During diagnosis, analyzing the fault dictionary to derive fault candidates is usually quite fast. However, for practical applications, the cause-effect diagnosis approach can be limited by some problems.

The first problem is the size of the fault dictionary since it requires a large amount of storage for recording all test responses for all faults against all test patterns. With

the increasing size of today's design, this method thus becomes sometimes unpractical. Some methods have been proposed to reduce the size of the fault dictionary. The pass-fail dictionary is the simple way to reduce the dictionary size by using a single pass/fail bit to replace the output response of the test vector [34]. However, this solution may negatively impact the diagnosis resolution as some faults become undistinguishable by using only pass-fail bits. Another method was proposed in [35] to build small fault dictionaries by recording only the test responses of the failing patterns instead of recording test responses of all test patterns for all faults. This can reduce the memory requirement without sacrificing resolution. Researchers in [36] proposed a technique to compress the fault dictionary by using a multiple input signature register to generate compressed fault signatures. One problem of this method is that two different test responses may be compressed and lead to the same failing signature.

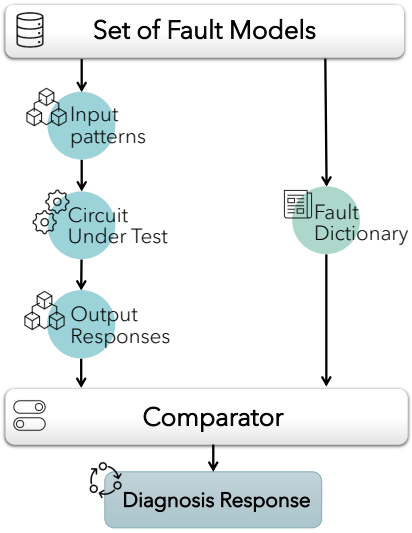

Fig. 1.10 Cause-effect diagnosis flow

Another issue that may occur when using the cause-effect diagnosis approach is the lack of accuracy (i.e., the real defect is not in the list of suspects). In fact, if the defect is not modeled by the fault models used to compute the dictionary, it cannot be identified by the diagnosis process. To solve this problem, diagnosis must be performed using several dictionaries, one for each fault model [37].

Intra-cell diagnosis

There are many research works focusing on intra-cell diagnosis. They can be classified into two categories.

The first category proposes a conversion from transistor level netlist into an equivalent gate level netlist, based on complex transformations rules [38-40]. Figure 1.11 gives some examples of these rules. Then, any classical inter-cell diagnosis

solution can be applied on the equivalent gate-level netlist. The main drawback of this approach is that the set of transformation rules depends on the targeted defects and non-modeled defects may not be diagnosed.

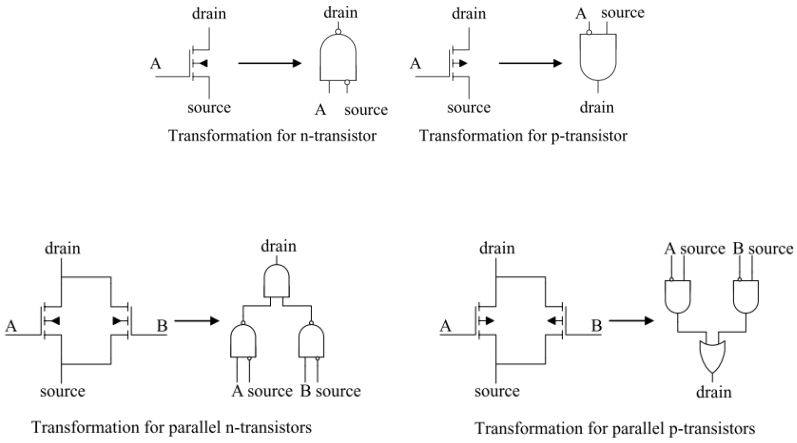

Fig. 1.11 Examples of transistor transformation to gate net-list

The second category of intra-cell diagnosis techniques is based on the "Cause-Effect" paradigm. The transistor-level netlist of a cell is exploited in order to inject the targeted defects. Therefore, a defect dictionary is created using results from transistor level simulations. The defect signature of all defects affecting a library cell is stored in this defect dictionary. Then, during the diagnosis process, the defect signature of all defects affecting a suspected cell is compared to the observed failures to obtain a list of candidates internal to the cell.

These approaches can be further classified depending on the accuracy of the injected defects and the simulation precision. In [41] targeted defects are simulated by using a switch-level simulator, thus leading to a less precise defect injection but this solution saves simulation time. In [6,42] a large number of defects is simulated at transistor level (i.e., by using SPICE simulations). For a given defect, different sizes of resistance are simulated to be as accurate as possible. This approach leads to more precise diagnosis results but it requires a huge simulation time. Moreover, the size of the defect dictionary is usually very high.

In order to reduce the simulation time and the dictionary size while keeping a high precision, authors in [43-45] propose to exploit layout information in order to consider only realistic defects. For example, for each cell, only the realistic potential bridging defects and via open defects are extracted and then simulated. By this way, only the identified set of realistic defects is simulated at transistor-level.

4.1.2 Diagnosis Using Critical Path Tracing

Inter-cell diagnosis

Unlike the cause-effect paradigm, effect-cause diagnosis approaches directly derive the fault candidates by using a CPT algorithm as illustrated in Fig. 1.12, without pre-constructing any fault dictionary.

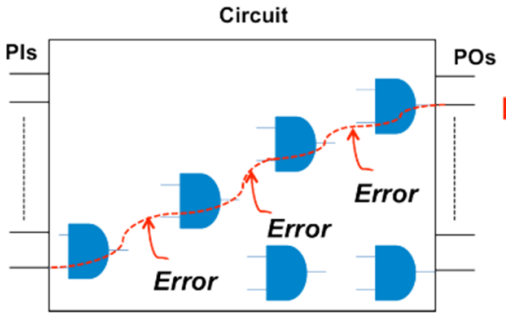

Fig. 1.12 Critical Path Tracing principle

A generic effect-cause diagnosis algorithm that makes the single fault assumption (only one fault at a time can occur in the circuit) is summarized as follows:

- Step 1 - *Initial faulty candidates identification*: In [46-47] the CPT technique is applied for logic diagnosis. In both cases, a specific fault model is considered, the stuck-at fault model for [46] and the delay fault model for [47]. The process of critical path tracing consists in starting from each faulty output and back tracing sensitive paths up to the primary inputs of the circuit. By this way, a number of critical paths containing logic gates and nets is obtained. If a single fault is assumed, then the intersection of all the critical paths traced from all failing outputs is considered as the final candidate set (that contains suspect gates and nets). Otherwise, in the case of a multiple fault assumption, the union will be the final candidate set.
- Step 2 - *Passing pattern validation*: The initial suspect set can be reduced by using passing test patterns. To this purpose, the same critical path tracing process is done from each fault-free output of the circuit. By definition, the real defect cannot produce any faulty behavior when applying passing patterns. Therefore, a candidate (suspected gate or net) will be removed from the initial suspect set if it is contained in the set of critical paths traced from the fault-free primary outputs.

Intra-cell diagnosis

In [48], the authors proposed an intra-cell diagnosis method based on the “Effect-Cause” paradigm aiming at locating the root cause of the observed failures inside a logic cell. It is based on the CPT here applied at transistor level. The main

characteristic of this approach is that it exploits the analysis of the faulty behavior induced by the actual defect. In other word, a defect is located by analyzing the effect it induces in the circuit. Moreover, since the complexity of a single cell in terms of transistor number is low, the proposed intra-cell diagnosis approach requires a negligible computation time.

Figure 1.13 sketches the overall diagnosis flow proposed in [48]. First, test patterns are applied to the CUT to distinguish between the correct circuit behavior and the faulty circuit behavior caused by defects. These defects induce failing output responses for one or more input test patterns. Input test patterns leading to observed faulty behavior (i.e., failing test patterns) are stored into a file called datalog.

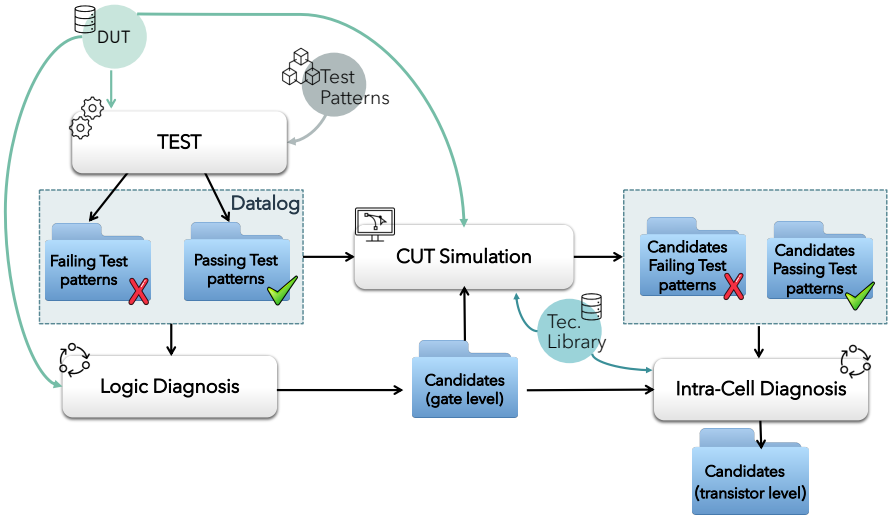

Fig. 1.13 Overall intra-cell diagnosis flow

Then, an inter-cell fault diagnosis (i.e., logic diagnosis) algorithm exploits datalog information to determine a list of suspected logic cells. Any available commercial diagnosis tool can be adopted. Then, CUT simulation aims at determining the local set of failing/passing patterns for each suspected logic cell reported by the logic diagnosis tool. Finally, the intra-cell diagnosis is executed for each suspected gate and the set of pre-determined local failing/passing test patterns. The diagnosis result is a list of candidates at transistor level. For each suspect, a set of fault models able to explain the observed failures is associated.

Compare with the cause-effect methodology, effect-cause approaches have several advantages. They require less memory storage since no fault/defect dictionary has to be constructed a priori. And they do not assume any fault model. Thus, they can be used for diagnosing more realistic faults.

4.2 Advanced Methods based on Machine Learning

Although conventional diagnosis techniques discussed in the previous section can achieve a good resolution, in some cases (e.g., complex cells, complex failure mechanisms) the number of candidates may be too high to allow an efficient PFA. This problem will be exacerbated in the future with the advent of very deep submicron (i.e., 7 nm and beyond) technologies. Improving diagnosis efficiency at the transistor level (i.e., CA diagnosis) is therefore mandatory. A mean to achieve this goal is to use supervised learning algorithms to determine suspected defects. Supervised learning is now used in several classification problems where the knowledge on some data can be used to classify a new instance of such data. In this section, we summarize the latest developments in the field of CA diagnosis based on supervised learning.

4.2.1 Preliminaries

Several learning-guided solutions for CA diagnosis of mission mode failures in customer returns have been proposed recently in [16, 49-53]. All solutions are based on a Bayesian classification method for accurately identifying defect candidates in combinational standard cells of a customer return. Choosing one solution over another depends on the test scenario (test sequence, test scheme, test conditions) considered during the diagnosis phase and selected according to the types of targeted defects and failure mechanisms.

The test scenarios in [16, 49-53] are sketched in Fig. 1.14. In [49-50], two **distinct** processes were developed to diagnose static and dynamic defects **separately**. In [49], a basic scan testing scheme used to apply static CA test sequences is considered, so that stuck-at faults plus static intra-cell defects are targeted during diagnosis. In [50], a fast sequential testing scheme used to apply dynamic CA test sequences is considered, so that transition faults plus dynamic intra-cell defects are targeted during diagnosis. Note that [51] is just a combination of [49] and [50], i.e., two testing schemes, one static and one dynamic, and two CA diagnosis flows, one static and one dynamic, are considered independently. The main limitation of the solutions in [49-51] is the required a priori knowledge of the type of targeted defects in the customer return. In other words, a test engineer needs to know what type of defects is screening before choosing between [49] or [50].

Test Sequence / Fault & Defect	Static (low speed)	Dynamic (at-speed)	Static + Dynamic
Stuck-at & Static CA	[49] [51]	[53]	[52] [16]
Transition & Dynamic CA		[50] [51] [53]	[52] [16]

Fig. 1.14 Test scenarios considered in [16, 49-53]

In an attempt to deal **concurrently** with all types of defects that may occur in customer returns, without any a priori knowledge of the targeted defect type, a new implementation of the CA diagnosis flow was proposed in [16-52]. Note that [16] is a fully extended version of [52]. Authors assume a test scenario in which **two** test sequences (static and dynamic) are used successively, each one considering 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.

To construct the comprehensive flow described in [16], a new framework was set up in which specific rules were defined 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. This method is summarized in the next subsection (4.2.2).

In [53], a new version of the CA diagnosis flow was proposed, 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 test flow depicted in Fig. 1.4, this scenario may happen 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). 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 no loss of information and hence without decreasing the quality of the training and inference phases.

4.2.2 Learning-Based Cell-Aware Diagnosis Flow

Figure 1.15 is a generic view of the learning-based CA diagnosis flow utilized in [16]. 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.

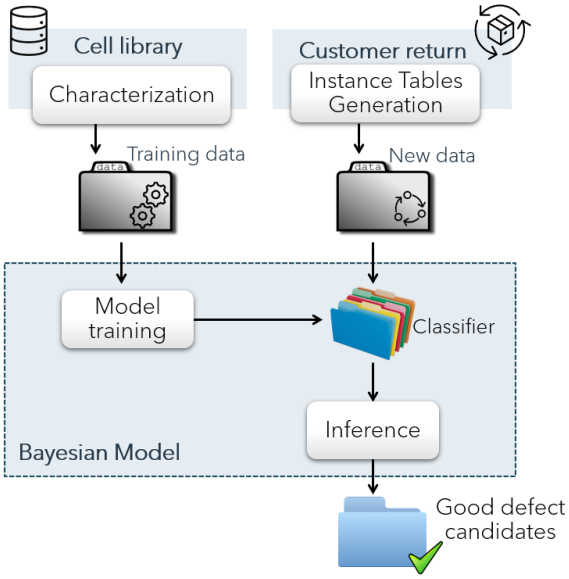

Fig. 1.15 Generic view of the cell-aware diagnosis flow used in [16]

After investigating several ML algorithms and observing their inference accuracies in [49], a Bayesian classification method has been chosen for the learning and inference phases in [16, 49-53]. So, the first main step of the 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. 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. More details about performance evaluation as done in this framework can be found in [51]. 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. The next subsections detail the various steps of the CA diagnosis flow, which is able to deal with any type of cell-internal defect (i.e., static and dynamic) that may occur in customer returns.

4.2.2.1 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 [16, 49-52] and containing six instances for an arbitrary two-input cell, is shown in Fig. 1.16. 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. 1.16) or dynamic (two input vectors - P_5 to P_{16} in Fig.1. 16 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.

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. 1.16 Example of training dataset for all defect types in a two-input cell as used in [16, 49-52]

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 [53], the representation of training data as used in [16, 49-52] 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. 1.16, 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. 1.17, 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. 1.16) 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 defects.

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. 1.17 Example of training dataset for all defect types in a two-input cell as used in [53]

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. 1.17, 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.

4.2.2.2 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. 1.18. 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, one 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 as in the case reported in [16].

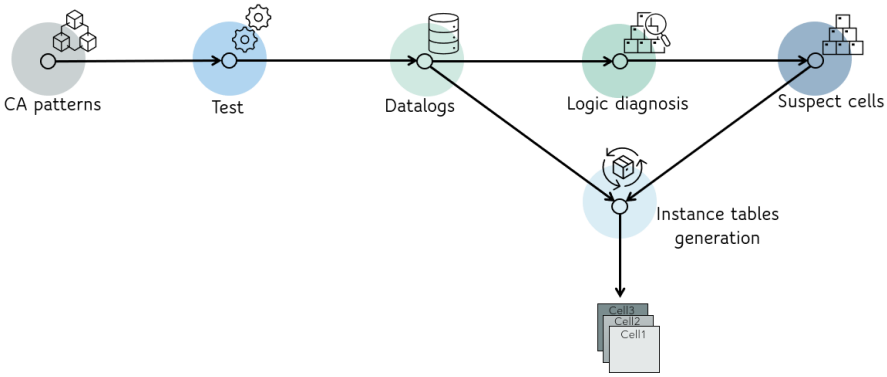

Fig. 1.18 Generation flow of instance tables

The format of a static instance table is illustrated in Fig. 1.19 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. 1.19 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 in detecting a stuck-at 1 on Z.

```
-----------------------------------------------------------------
                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. 1.19 Example of static and dynamic instance tables

4.2.2.3 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, one 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.

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. 1.20 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. 1.21 Format of a new data instance as used in [53]

The formats of a new data instance as used in [16, 49-52] and [53] are illustrated in Fig. 1.20 and Fig. 1.21 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 **actually** 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.

4.2.2.4 Diagnosis of Defects in Sequential Cells

All the work carried out in [16, 49-53] 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 [54].

The two main differences between a combinational cell and a sequential cell are that i) the latter has 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 expanded accordingly. In each tuple, the first value is either U (i.e., pUlse) or 0, depending 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. 1.22. 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. 1.22. 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, the solution consists in including static cell-patterns (e.g., P1 to P4 in Fig. 1.22) in the training data of sequential cells.

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

With the above representation of training data for sequential cells, one can see that the diagnosis flow in Fig. 1.15 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 type of manipulated standard cells.

5 Industrial Case Studies

The CA diagnosis flow described in Section 4.2.2 and targeting defects in both combinational and sequential cells of customer returns has been implemented in a Python program. For validation purpose, authors in [16, 49-54] have experimented the proposed flow in three different ways:

- First, they conducted experiments on ITC'99 benchmark circuits with defect injection campaigns targeting **combinational cells** in each circuit. Various results are reported in [16, 49-53] to show the superiority of the framework when compared to commercial diagnosis solutions.

- Next, they considered a test chip developed by STMicroelectronics and designed using a 28 nm FDSOI technology, and they conducted two defect injection campaigns targeting **sequential cells** [54]. Results are reported in subsection 5.1 and also demonstrate the effectiveness of the diagnosis framework.
- Finally, they considered **a customer return** from STMicroelectronics and performed a silicon case study with a real defect subsequently analyzed and identified during PFA. Results are reported in subsection 5.2.

5.1 Simulated Test Case Studies

Authors in [54] 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 1.2.

A first simulated case study was done with a **static** defect injection campaign. All possible static defects were successively injected into three scan flip-flops (SFF) of a single full-scan digital block. This block was tested 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 1.3. Results obtained after executing the 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. 1.23 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).

TABLE 1.2 MAIN FEATURES OF THE SILICON TEST CHIP

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

TABLE 1.3 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

A second simulated case study with another defect injection campaign was performed on the same test chip. All possible dynamic defects were successively injected into three scan flip-flops of a single full-scan digital block. This time, a dynamic CA test sequence was applied and achieved 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 the 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.

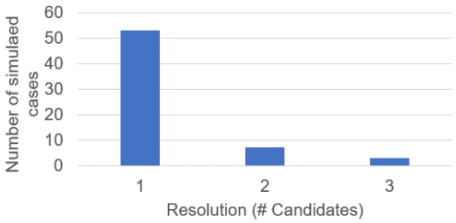

Fig. 1.23 Distribution of the resolution with respect to the simulated cases

5.2 Silicon Test Case Study

Next, a silicon case study was performed on a customer return designed with a 28 nm FDSOI technology from STMicroelectronics [54]. The test conditions used to run the experiments were as follows: 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. The CA diagnosis flow was experimented and the following results were obtained. 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 the 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. 1.24) and observing the failure analysis cross-sectional view.

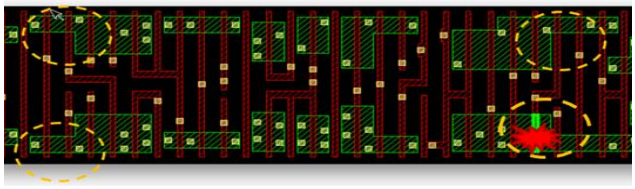

Fig. 1.24 Layout view of the suspected cell and the incriminated transistor. Yellow circles indicate defect candidates and red mark indicates actual observed defect

6 Discussion and Conclusion

This chapter has identified the key challenges in cell-aware diagnosis of silicon customer returns and has presented the solutions and corresponding results that have emerged from leading-edge research in this domain. In a comprehensive form, it proposed a compendium of solutions existing in this field. More in detail, the chapter has presented a framework for cell-aware diagnosis of customer returns based on supervised learning. The flow indistinctly deals with static and dynamic defects that may occur in combinational cells or sequential cells of real circuits. A Naive Bayes classifier is used to precisely identify defect candidates. Experiments on silicon test cases have been done to validate the flow and demonstrate its efficacy in terms of accuracy and resolution.

Results of these experiments proved the appropriateness of a learning-based method to solve the problem of customer return diagnosis, despite the small size of the training dataset used (only one sample for one defect class). If multiple defect sizes and test conditions are used, this becomes even truer. Indeed, multiple samples (one for each defect size or defect size range, one for each PVT test condition) can be associated to a given defect class, simply because the behavior of the defect differs when applying the same set of test patterns. In terms of timing and complexity, this will just slightly impact the 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), the diagnosis framework will still be easily and time-efficiently applicable.

It is worth noting that among other factors, the effectiveness of the framework can be explained by the fact that Naïve Bayes algorithm usually offers good classification performance [55]. The NB classifier requires a small amount of training data to estimate its parameters [56], which is the case in the proposed 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.

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. N. Sumikawa, D. Drmanac, Li-C. Wang, and M.S. Abadir, "Understanding Customer Returns From A Test Perspective," *in Proc. IEEE VLSI Test Symposium*, pp. 2-7, 2011.
2. R.J. Tikkanen, S. Siatkowski, Li-C. Wang, and M.S. Abadir, "Yield Optimization Using Advanced Statistical Correlation Methods," *in Proc. IEEE International Test Conference*, 2014. DOI: 10.1109/TEST.2014.7035326
3. Y. Benabboud, A. Bosio, L. Dilillo, P. Girard, A. Virazel, and O. Riewer, "A Comprehensive System-on-Chip Logic Diagnosis," *in Proc. IEEE Asian Test Symposium*, 2010. DOI: 10.1109/ATS.2010.49
4. Y. Xue, X. Li, R. D. Blanton, C. Lim, and M. Enamul Amyeen, "Diagnosis Resolution Improvement through Learning-Guided Physical Failure Analysis," *in Proc. IEEE International Test Conference*, 2016. DOI: 10.1109/TEST.2016.7805824
5. S. Pateras, "IC Test Solutions for the Automotive Market," Mentor Graphics, White Paper, May 2017. https://www.techonline.com/tech-papers/ic-test-solutions-for-the-automotive-market/
6. F. Hapke, et. al., "Cell-Aware Test," IEEE Transactions on Computer-Aided Design, vol. 33, no. 9, pp. 1396 - 1409, 2014.
7. P. Maxwell, F. Hapke, and H. Tang, "Cell-Aware Diagnosis: Defective Inmates Exposed in their Cells," *in Proc. IEEE European Test Symposium*, 2016. DOI: 10.1109/ETS.2016.7519313
8. Y. Xue, X. Li, R. D. Blanton, and C. Lim, "Improving Diagnostic Resolution of Failing ICs Through Learning," IEEE Trans. on Computer-Aided Design, vol. 37, no. 6, pp. 1288-1297, 2018.
9. X. Ren, M. Martin, and R. D. Blanton, "Improving Accuracy of On- Chip Diagnosis via Incremental Learning," *in Proc. IEEE VLSI Test Symposium*, pp. 1–6, 2015.
10. 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," *in Proc. IEEE European Test Symposium*, pp. 1–10, 2015. DOI: 10.1109/ETS.2015.7138758
11. Y. Xue, O. Poku, X. Li, and R. D. Blanton, "PADRE: Physically- Aware Diagnostic Resolution Enhancement," *in Proc. IEEE International Test Conference*, 2013. DOI: 10.1109/TEST.2013.6651899
12. R. Pan, Z. Zhang, X. Li, K. Chakrabarty and X. Gu, "Unsupervised Root-Cause Analysis for Integrated Systems," *in Proc. IEEE International Test Conference*, 2020. DOI: 10.1109/ITC44778.2020.9325268

13. B. Kruseman, A. Majhi, C. Hora, S. Eichenberger, and J. Meirlevede, “Systematic Defects in Deep Sub-micron Technologies”, *in Proc. IEEE International Test Conference*, pp. 290-299, 2005.
14. F. Hapke; R. Krenz-Baath; A. Glowatz; J. Schloeffel; H. Hashempour; S. Eichenberger; C. Hora; D. Adolfsson, “Defect-Oriented Cell-Aware ATPG and Fault Simulation for Industrial Cell Libraries and Designs”, *in Proc. IEEE International Test Conference*, pp. 1.2, 2009. DOI: 10.1109/TEST.2009.5355741
15. F. Hapke, W. Redemund, J. Schloeffel, R. Krenz-Baath, A. Glowatz, M. Wittke, H. Hashempour, and S. Eichenberger, “Defect-Oriented Cell-Internal Testing”, *in Proc. IEEE International Test Conference*, paper 10.1, 2010. DOI: 10.1109/TEST.2010.5699229
16. S. Mhamdi, P. Girard, A. Virazel, A. Bosio, and A. Ladhar, “A Learning-Based Cell-Aware Diagnosis Flow for Industrial Customer Returns,” *in Proc. IEEE International Test Conference*, 2020. DOI: 10.1109/ITC44778.2020.9325246
17. M. Bushnell and V. Agrawal, “Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits,” Springer, 2002, ISBN 978-0-7923-7991-1.
18. T. P. Ho, E. Faehn, A. Virazel, A. Bosio, and P. Girard, “An Effective Intra-Cell Diagnosis Flow for Industrial SRAMs,” *in Proc. IEEE International Test Conference*, pp. 1-8, 2018. DOI: 10.1109/TEST.2018.8624799
19. A. Pavlidis, E. Faehn, M.-M. Louërat, and H.-G. Stratigopoulos, “BIST-Assisted Analog Fault Diagnosis,” *in Proc. IEEE European Test Symposium*, pp. 1-6, 2021. DOI: 10.1109/ETS50041.2021.9465386
20. F. Ye, Z. Zhang, K. Chakrabarty and X. Gu, “Adaptive Board-Level Functional Fault Diagnosis Using Incremental Decision Trees,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 35, no. 2, pp. 323-336, 2016.
21. Y. Benabboud, A. Bosio, P. Girard, S. Pravossoudovitch, A. Virazel, L. Bouzaida, and I. Izaute, “A case study on logic diagnosis for system-on-chip,” *in Proc. IEEE International Symposium on Quality Electronic Design*, pp. 253-259, 2009.
22. F. Ye, Z. Zhang, K. Chakrabarty and X. Gu, "Board-Level Functional Fault Diagnosis Using Multikernel Support Vector Machines and Incremental Learning," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 33, no. 2, pp. 279-290, Feb. 2014.
23. J.A. Waicukauski and E. Lindbloom, “Failure Diagnosis of Structured VLSI,” IEEE Design & Test of Computers, vol. 6, no. 4, pp. 49-60, July 1989.
24. A. Bosio, P. Girard, S. Pravossoudovitch, and A. Virazel, “A Comprehensive Framework for Logic Diagnosis of Arbitrary Defects,” IEEE Transactions on Computers, vol. 59, no. 3, pp. 289-300, 2010.
25. X. Fan, W. Moore, C. Hora, and G. Gronthood, ‘‘A Novel Stuck-At Based Method for Transistor Stuck-Open Fault Diagnosis,’’ *in Proc. IEEE International Test Conference*, pp 386-395, 2005.
26. F. Hapke, M. Reese, J. Rivers, A. Over, V. Ravikumar, W. Redemund, A. Glowatz, J. Schloeffel, and J. Rajski, “Cell-Aware Production Test Results from a 32-nm Notebook Processor,” *in Proc. IEEE International Test Conference*, 2012. DOI: 10.1109/TEST.2012.6401533
27. A. Sun, A. Bosio, L. Dillilo, P. Girard, A. Virazel, S. Pravossoudovitch, and E. Auvray, “Intra-Cell Defects Diagnosis,” *Journal of Electronic Testing: Theory and Applications*, vol. 30, no. 5, pp. 541-555, 2014.

28. J. Li et al, “Diagnosis for Sequence Dependent Chips,” *in Proc VLSI Test Symposium*, pp.187-192, 2002.
29. J. Li and E. J. McCluskey, “Diagnosis of Resistive-Open and Stuck-Open Defects in Digital CMOS ICs,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 24, no. 11, pp. 1748-1759, Nov. 2005.
30. J. Li, C.W. Tsang, and E.J. McCluskey, “Testing for Resistive Opens and Stuck Opens,” *in Proc. IEEE International Test Conference*, pp 1049-1058, 2001.
31. Y. Huang, R. Guo, W.-T. Cheng, and J. Chien-Mo Li, “Survey of Scan Chain Diagnosis,” IEEE Design and Test of Computers, vol. 25, no. 3, pp. 240-248, June 2008.
32. P. Girard, C. Landrault, S. Pravossoudovitch, and B. Rodriguez, “A Diagnostic ATPG for Delay Faults Based on Genetic Algorithms,” *in Proc.* IEEE International Test Conference, 1996.
33. M. Abramovici and M. A. Breuer, “Fault Diagnosis Based on Effect-Cause Analysis: An Introduction,” *in Proc. ACM Design Automation Conference*, pp. 69–76, 1980.
34. Pomeranz and Reddy, “On the Generation of Small Dictionaries for Fault Location,” *in Proc. IEEE/ACM International Conference on Computer-Aided Design*, pp. 272–279, 1992.
35. B. Chess and T. Larrabee, “Creating Small Fault Dictionaries [logic circuit fault diagnosis],” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 3, pp. 346–356, Mar. 1999.
36. H. Tang, C. Liu, W.-T. Cheng, S. M. Reddy, and W. Zou, “Improving Performance of Effect-Cause Diagnosis with Minimal Memory Overhead,” *in Proc. IEEE Asian Test*, pp. 281–287, 2007.
37. J. Wu and E. M. Rudnick, “Bridge Fault Diagnosis Using Stuck-at Fault Simulation,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 4, pp. 489–495, Apr. 2000.
38. X. Fan, W. Moore, C. Hora, and G. Gronthoud, “A Novel Stuck-at Based Method for Transistor Stuck-open Fault Diagnosis,” *in Proc. IEEE International Test Conference*, pp. 378–386, 2005.
39. X. Fan, W. Moore, C. Hora, M. Konijnenburg, and G. Gronthoud, “A Gate-Level Method for Transistor-Level Bridging Fault Diagnosis,” *in Proc. IEEE VLSI Test Symposium*, pp. 266–271, 2006.
40. X. Fan, W. R. Moore, C. Hora, and G. Gronthoud, “Extending Gate-Level Diagnosis Tools to CMOS Intra-Gate Faults,” *IET Computer and Digital Techniques*, vol. 1, no. 6, pp. 685, 2007.
41. M. Amyeen, D. Nayak, and S. Venkataraman, “Improving Precision Using Mixed-level Fault Diagnosis,” *in Proc. IEEE International Test Conference*, pp. 1–10, 2006.
42. A. Ladhar, M. Masmoudi, and L. Bouzaida, “Efficient and Accurate Method for Intra-Gate Defect Diagnoses in Nanometer Technology and Volume Data,” *in Proc. IEEE/ACM Design, Automation & Test in Europe Conference & Exhibition*, pp. 988–993, 2009.
43. A. Ladhar, “Extraction and Diagnosis of Submicron Defects,” PhD thesis, Tunis University, 2011.
44. A. Ladhar, M. Masmoudi, and L. Bouzaida, “Extraction and Simulation of Potential Bridging Faults and Open Defects Affecting Standard Cell Libraries,” *in Proc. IEEE International Conference on Signals, Circuits and Systems*, pp. 1–6, 2008.
45. A. Ladhar and M. Masmoudi, “A Novel Algorithm to Extract Open Defects from Industrial Designs,” in *Proc. IEEE International Conference on Electronics, Circuits and Systems*, pp. 675–678, 2009.

46. M. Abramovici, P. R. Menon, and D. T. Miller, "Critical Path Tracing: An Alternative to Fault Simulation," *IEEE Design & Test of Computers*, vol. 1, no. 1, pp. 83–93, Feb. 1984.
47. P. Girard, C. Landrault, and S. Pravossoudovitch, "Delay-Fault Diagnosis by Critical-Path Tracing," *IEEE Design & Test of Computers*, vol. 9, no. 4, pp. 27–32, Dec. 1992.
48. Z. Sun, A. Bosio, L. Dilillo, P. Girard, A. Todri-Sanial, A. Virazel, and E. Auvray, "Effect-Cause Intra-Cell Diagnosis at Transistor Level*," in Proc. IEEE International Symposium on Quality Electronic Design* pp. 460–467, 2013.
49. S. Mhandi, A. Virazel, P. Girard, A. Bosio, E. Auvray, E. Faehn, and A. Ladhar, "Towards Improvement of Mission Mode Failure Diagnosis for System-on-Chip," in *Proc. IEEE International On-Line Testing Symposium*, 2019.
50. S. Mhandi, P. Girard, A. Virazel, A. Bosio and A. Ladhar, "Cell-Aware Diagnosis of Automotive Customer Returns Based on Supervised Learning," presented at *IEEE Automotive Reliability and Test Workshop*, 2019.
51. S. Mhamdi, P. Girard, A. Virazel, A. Bosio, E. Faehn, and A. Ladhar, "Cell-Aware Defect Diagnosis of Customer Returns Based on Supervised Learning," *IEEE Transactions on Device Material and Reliability*, vol. 20, no. 2, 2020.
52. S. Mhandi, P. Girard, A. Virazel, A. Bosio and A. Ladhar, "Learning-Based Cell-Aware Defect Diagnosis of Customer Returns," in *Proc. IEEE European Test Symposium*, 2020.
53. S. Mhandi, P. Girard, A. Virazel, A. Bosio and A. Ladhar, "Cell-Aware Diagnosis of Customer Returns Using Bayesian Inference," in *Proc. IEEE International Symposium on Quality Electronic Design*, 2021.
54. P. d'Hondt, S. Mhamdi, P. Girard, A. Virazel, and A. Ladhar, "A Comprehensive Framework for Cell-Aware Diagnosis of Customer Returns," to appear in *Microelectronics Reliability Journal*, October 2021.
55. H. Zhang, "The Optimality of Naive Bayes", in *Proc. 17th International Florida Artificial Intelligence Research Society Conference*, May 2004.
56. Webpage, "1.9. Naive Bayes — Scikit-Learn 0.24.1 Documentation", https://scikit-learn.org/stable/modules/naive_bayes.html