DATA MINING AND DIAGNOSING IC FAILS

Leendert M. Huisman

Springer

DATA MINING AND DIAGNOSING IC FAILS

FRONTIERS IN ELECTRONIC TESTING

Consulting Editor
Vishwani D. Agrawal

Books in the series:

Embedded Processor-Based Self-Test
D. Gizopoulos
ISBN: 1-4020-2785-0

Testing Static Random Access Memories
S. Hamdioui
ISBN: 1-4020-7752-1

Verification by Error Modeling
K. Radecka and Zilic
ISBN: 1-4020-7652-5

Elements of STIL: Principles and Applications of IEEE Std. 1450
G. Maston, T. Taylor, J. Villar
ISBN: 1-4020-7637-1

Fault Injection Techniques and Tools for Embedded systems Reliability Evaluation
A. Benso, P. Prinetto
ISBN: 1-4020-7589-8

High Performance Memory Memory Testing
R. Dean Adams
ISBN: 1-4020-7255-4

SOC (System-on-a-Chip) Testing for Plug and Play Test Automation
K. Chakrabarty
ISBN: 1-4020-7205-8

Test Resource Partitioning for System-on-a-Chip
K. Chakrabarty, Iyengar & Chandra
ISBN: 1-4020-7119-1

A Designers' Guide to Built-in Self-Test
C. Stroud
ISBN: 1-4020-7050-0

Boundary-Scan Interconnect Diagnosis
J. de Sousa, P.Cheung
ISBN: 0-7923-7314-6

Essentials of Electronic Testing for Digital, Memory, and Mixed Signal VLSI Circuits
M.L. Bushnell, V.D. Agrawal
ISBN: 0-7923-7991-8

Analog and Mixed-Signal Boundary-Scan: A Guide to the IEEE 1149.4 Test Standard
A. Osseiran
ISBN: 0-7923-8686-8

Design for At-Speed Test, Diagnosis and Measurement
B. Nadeau-Dosti
ISBN: 0-79-8669-8

Delay Fault Testing for VLSI Circuits
A. Krstic, K-T. Cheng
ISBN: 0-7923-8295-1

Research Perspectives and Case Studies in System Test and Diagnosis
J.W. Sheppard, W.R. Simpson
ISBN: 0-7923-8263-3

Formal Equivalence Checking and Design Debugging
S.-Y. Huang, K.-T. Cheng
ISBN: 0-7923-8184-X

Defect Oriented Testing for CMOS Analog and Digital Circuits
M. Sachdev
ISBN: 0-7923-8083-5

DATA MINING AND DIAGNOSING IC FAILS

LEENDERT M. HUISMAN
IBM Systems and Technology Group

Library of Congress Cataloging-in-Publication Data

A C.I.P. Catalogue record for this book is available
from the Library of Congress.

ISBN-10: 0-387-24993-1
ISBN-13: 9780387249933

ISBN-10: 0-387-26351-9 (e-book)
ISBN-13: 9780387263519

Printed on acid-free paper.

Printed in the United States of America.

9 8 7 6 5 4 3 2 1 SPIN 11379102

springeronline.com

To Ans and Niels

Contents

List of Figures

List of Tables

Preface

This book grew out of an attempt to describe a variety of tools that were developed over a period of years in IBM to analyze Integrated Circuit fail data. The selection presented in this book focuses on those tools that have a significant statistical or datamining component. The danger of describing statistical analysis methods is the amount of non-trivial mathematics that is involved and that tends to obscure the usually straigthforward analysis ideas. This book is, therefore, divided into two roughly equal parts. The first part contains the description of the various analysis techniques and focuses on ideas and experimental results. The second part contains all the mathematical details that are necessary to prove the validity of the analysis techniques, the existence of solutions to the problems that those techniques engender, and the correctness of several properties that were assumed in the first part. Those who are interested only in using the analysis techniques themselves can skip the second part, but that part is important, if only to understand what is being done.

Several of the analysis techniques presented here were described previously in journal and conference articles: SLAT was described in [6][1] and [30][1], Embedded Object Analysis and Commonality Analysis were presented briefly in [8][1] and [31][1], respectively, and the relationship between coverage and yield was explored in [28][1] and [29]. The treatment in this book adds many details, and corrects some errors in the previous publications. The work presented here was not the work of its author alone, as is clear from the list of contributors in the Acknowledgements. The details of the mathematical analysis and of the analysis of the experimental data, however, are. Consequently, any errors are his responsibility.

LEENDERT HUISMAN
IBM Systems and Technology Group

Acknowledgements

This book was written over a period of years, and has benefitted from many discussions with my colleagues. The chapter on Commonality analysis is based on a collaboration with Maroun Kassab and Leah Pastel, and the chapter on Instance Analysis on one with Greg Bazan, Francis Gravel, Anne Pardee, Leah Pastel and Ken Rowe. The chapter on SLAT arose from several discussions with Tom Bartenstein, Doug Heaberlin, David Sliwinski and Kevin Stanley. The explanation of why the stuck-at fault model is so successful is the result of many lunch-time discussions with Peter Wohl.

Chapter 1

Introduction

Diagnosis is the extraction of information from fail data. In this book, the things that fail are Integrated Circuits (ICs), and the failures are those that occur during manufacturing test; more precisely, during the application of specially designed series of electrical stimuli to the integrated circuits on finished wafers or in separate modules. The information that is extracted concerns the causes of the failures. It can vary from the simplest, like the average number of distinct defects on an IC, to the most detailed, for example, the exact location and nature of the anomalies that caused the failure. The information that can be extracted depends on the tests during which the IC failed, and on the amount of fail data collected.

Extracting information from fail data can take many forms. For example, the failure probabilities of identifiable units on the device, like embedded memories or PLLs, can be estimated; different fail mechanisms may be identified by comparing fails at different phases of the test, or by analyzing the patterns of passes and fails on wafers; groups of chips may be identified that seem to share an underlying fail mechanism that did not affect other chips; if enough fail data has been collected, the location or the nature of the defect that caused the fail may even be estimated.

All these forms of extracting diagnostic information help identify the nature and the causes of the defects that occur on integrated circuits. Clustering chips into groups that seem to have failed similarly provides a first estimate of what defects occur, even though the identity of the defects is not yet known. Estimating occurrence probabilities helps focusing on the most prevalent failure mechanisms. And determining the location of the real defect then makes it possible to study the actual defect mechanism: why they occur, and how they affect the electronic circuit.

What is learned during such an analysis is fed back to the process or design engineers, so design and/or manufacturing changes can be made to prevent, or, at least, decrease the possibility of future occurrences of similar defects. Such changes increase the yield, that is, lead to fewer failing ICs, but may also increase the reliability of the finished products.

There are many types of fail data that can be used in diagnostic activities. Not all of the data will unambiguously identify a cause. Some of the data can only point in certain directions, like the probability that some defect, known only through its effects, will occur, or the most likely location on the device where this defect might be situated.

Thus defined, diagnosis covers a wide range of activities.This book will be limited to only one of them, software based analysis of routinely collected fail data during manufacturing testing. Fail data collected at the tester is usually very limited, showing only which phase of the test exposed the defect, and just enough to establish that the device indeed failed the test. The manufacturing tests are sometimes enhanced, however, with additional data collection to support diagnosis. Such additional data collection can be done systematically on a fraction of the (failing) devices, or can be done automatically in response to certain types of fail events. This added information is still part of routine data collection, and will of course also be considered here. In fact, one of the purposes of this book is to determine what additional data collection gives the most diagnostic benefit for the lowest added cost.

Other diagnostic techniques, like probing the failing IC or interactively applying tests based on what has been learned already about the cause of the fail will not be addressed. Such techniques are very important, and are almost always required in the end for the final determination of the cause of the fail, but they fall outside the scope of this book.

The process of diagnosis starts with applying tests to batches of chips. The test sequence is usually the same for all devices, even though the data collection strategy may not be, and is typically divided into a number of steps. The first few steps are intended to verify that the tester probe has made good electrical contact with the chip, and that there are no gross defects that would cause a large current to flow through the pins of the probe. The observations in these initial steps, called the gross tests, consist of measurements of currents and voltages. The results of the measurements are compared to predefined ranges of allowed values, and a chip fails any of the gross tests if the corresponding measurement is outside its accompanying range. If a failure does occur, there is either no good contact between the tester and the internal electronics of the device, or the current flowing through the probe and the contacts on the chip is so large that it might damage the probe. In either case, no further testing can be done.

If the chip does not fail the gross tests, more detailed tests are applied. These are generally of two kinds. Tests of the first kind consists of further measurements, for example of IDDq currents, flush delays, or ring oscillator frequencies. Those of the second kind consists of patterns that exercise selected portions of the chip electronics. A pattern consists of the application of a series of electrical stimuli to the inputs of the chip, and the observation of the electrical responses at its outputs. It is generally digital in nature, and exercises some portion of the chip electronics directly, as in deterministic (like LSSD) tests, or causes the chip electronics to generate internal patterns,

as in Built-In Self Test (BIST). The chip can fail any of the patterns applied to it.

The tests can be enriched further by applying portions of the patterns and measurements at different voltages and temperatures. The fails occurring at one voltage or temperature should be distinguished from fails occurring at other voltages or temperatures, because they may be caused by different defect mechanisms. Also, the chip may fail some pattern at one voltage, but pass the same test at some other voltage, and this pattern of corner specific passes and fails provides more information about the defect.

The test sequence consists therefore of a large number of steps: the measurements in the initial phase, and subsequent patterns and more measurements. A chip that fails any of these steps is defective, and should be removed. The removal need not be immediate, however, unless the failing test is one of the gross tests, as it might be desirable for diagnostic purposes to obtain the response of a chip to other tests in the sequence.

The responses of the devices to the patterns and measurements are recorded in the test history, which is the complete record of which chips failed at which step. If the number of steps is large, as it typically is because of the large number of patterns, they can be grouped together into more meaningful major steps. Examples are all the deterministic patterns at a given voltage and temperature, the measurements of the oscillator frequencies, and detailed IDDq measurements. Each such major step is assigned a code, and the test history is merely a listing of the codes for each chip, indicating which major steps it failed. Each failing chip is also assigned an overall code, called the sort code, which is the code of that major test step at which the chip failed for the first time in the test sequence.

Various types of analysis can be performed on the accumulated fail data, and four different types will be discussed in this book. The first type consists of statistical analyses of the fail data of large numbers of devices that were all tested with the same test sequence. Examples of such large groups of devices are the chips on a single wafer, or on all the wafers in a lot. The second type is that of the spatial patterns of passes and fails on wafers, and the subsequent classification of those patterns into random ones and ones that are distinctly non-random. The third analysis type is that of potential commonalities between fails; that is, the attempt to identify common fail mechanisms by comparing fail syndromes. The fourth and final analysis type to be discussed in this book is using the fail syndrome of a particular device to identify the location of the defect that caused that device to fail.

Statistical analysis fits very naturally in the IC test process, because much data that is collected is statistical in nature. Examples are the fractions of good devices on individual wafers, the number of devices that were exposed for the

first time by a certain pattern, and the number of times a given object on the device, like an embedded memory, failed across all the devices on a wafer. The statistical analysis is largely descriptive, and its purpose is, broadly speaking, to establish the normal behavior of the ICs, that is, to describe how the ICs, on average, respond to the manufacturing test.

Establishing a normal behavior of the real hardware has two benefits. First, it makes it possible to compare that normal behavior to what is expected based on our understanding of the manufacturing process and test. Such a comparison can then either confirm our understanding, or point at limitations thereof; for example, holes, or, even worse, completely erroneous ideas of what goes on during manufacturing and test.

The second benefit is that, once the normal behavior has been established, it is possible to estimate normal statistical fluctuations around the expected normal values, and to separate those from the truly deviant ones. In other words, it makes it possible to identify ICs that truly deviate from the normal behavior, even when taking statistical fluctuations into account. Non-normal fluctuations often point at systematic problems, and finding them is the first step in identifying, and then removing the underlying cause.

This book uses a limited number of statistical distribution and techniques. Their main features are briefly reviewed in Chapter 2.

The most immediate statistical analysis, taken up in Chapter 3, is that of the distribution of the yields, that is, of the fractions of good devices on single wafers. Using pass/fail information only is a poor use of the test history, however, and a more detailed statistical analysis of the distribution of all the sort codes will be taken up subsequently in the same chapter.

The device yield depends on the area of the device: the larger the area, the lower the yield. It has been noted over the years that this area dependence is more complex than a simple Arrhenius factor, and the cause of this complexity is usually assumed to be some clustering of the defects over the wafer. Clustering of the defects is, of course, related to what causes the defects in the first place, and this area dependence may, therefore, provide additional information about the defect mechanisms. It is discussed in some detail in Chapter 4.

Even more detailed yield information can be obtained when the device contains identifiable embedded objects that are tested separately by specific portions of the test sequence. Examples of such units are embedded memories and scan chains. The former are tested using specific memory tests, applied from the tester or generated on chip (memory BIST). The latter are tested with specific scan tests before the beginning of scan based testing.

Such object oriented yield analysis is valuable for all the reasons mentioned above regarding device yield analysis. There is a normal rate at which

such objects are expected to fail, and any significant deviations from the normal rates point at systematic problems. In addition, however, these objects often have a size associated with them, like the number of cells in an embedded memory, and this size makes it possible to compare the failure rates of the objects not just to their counterparts on different chips, but also to different ones on the same chip that are similar in nature but have different sizes. The statistical analysis associated with embedded objects is the topic of Chapter 5.

In addition to rates, the fails of chips or objects on chips are distributed in some fashion over the wafer. This distribution is, of course, limited by the locations of the devices on the wafer, but, with enough such devices, distinct patterns can still be recognized. The primary goal of analyzing fail patterns is to identify those patterns that deviate significantly from a random one, and, therefore, may indicate some systematic problem. In addition, many process problems lead to distinct patterns of fails over the wafer that can be classified. Fairly standard pattern recognition techniques can then be used to identify the presence of such patterns, and, thereby, the potential occurrence of the associated process problem. This type of analysis is treated in Chapter 7.

The goal of commonality analysis, the third type of analysis to be considered in this book, is almost the opposite of statistical analysis: cluster the devices with similar fail syndromes into separate groups. There is no expectation of normality, or conformance to some model, but, instead, the fail data, whatever they are, are taken as signs of the underlying defect, and used to identify instances of the same or similar defects.

This clustering is important, because it attempts to catalogue the types of defects that occur and to determine their occurrence rates. If the devices can be divided into groups at least some of which are large and clearly separated, the obvious conclusion can be drawn that those larger groups correspond to unique fail mechanisms that need to be investigated further. Once such clusters have been identified, they can be selected for further, more detailed analysis. Various forms of commonality analysis are discussed in Chapter 6, with additional examples briefly mentioned in Chapter 3 and Chapter 5.

The most detailed analysis that can be done, and that is still statistical in nature, occurs when there is a notion of coverage. Coverage is a number between 0 and 1 that is attached to any initial section of the scan based patterns (that is, all patterns up to and including some selected one), and that is equal to the fraction of defects that are exposed by the patterns in that initial section. It exists, in particular, for that portion of the test that uses scan based patterns, and for them coverage is in fact routinely calculated.

Coverage clearly is related to the fraction of devices that fail during the application of the patterns in such an initial section. The form of that relationship depends on the nature of the defects, and, consequently, analyzing the

progressive fallout when scan based patterns are applied should give useful information about the defects. Chapter 8 will address both the relationship between coverage and fallout, and how to use this relationship to extract defect specific information.

Chapter 9 is devoted to the fourth type of fail data analysis, that of using the collected fail data to identify the location of the defect. This type of analysis is far more complex than the previous types, because it uses a detailed logical model of the device, in addition to the fail data. Consequently, it is far more time consuming, and places far more stringent requirements on the fail data that need to be collected for it to be applicable. On the other hand, if successful, it can locate the defect exactly within the device, and is one of the main enablers of a successful physical failure analysis.

The simplest form of logic diagnosis is that based on the single stuck-at fault model. It gives good results surprisingly often, even though many realistic defects cannot be modeled by single stuck-at faults. Chapter 9 discusses this approach in detail, even though more powerful logic diagnosis techniques are available, because it is the classical form of logic diagnosis, and because many of the issues that complicate more powerful techniques already occur here.

SLAT is a far more powerful logic diagnosis technique that relies on two assumptions. The first assumption is almost an observation, and states that any defect behaves as some set of stuck-at faults under the application of any particular pattern that detects it. The defect may, and often will behave as different sets of stuck-at faults with different detecting patterns. The second assumption is that there will be some detecting patterns that cause the defect to behave as a single stuck-at fault. This assumption is the crucial one, for it reduces logic diagnoses to the standard problem of stuck-at fault diagnosis discussed in Chapter 9. It is more complicated than the latter one, though, for each detecting pattern has to be diagnosed as if it is the only one available. SLAT is the careful simultaneous analysis of all these single pattern diagnoses, and will be described in Chapter 10.

The discussion in this book is ordered roughly according to the amount of detail and the computational effort used in the various analyses. This ordering corresponds more or less to what a diagnostic engineer might do when faced with a large volume of failing devices, and having to find the main causes of the fails. The corresponding flow is shown in Figure 1.

Design data are needed to generate the test sequence and in some of the diagnoses. Not all diagnostic techniques require design data, though. Yield analysis, for example, does not need it at all. Most diagnostic techniques do, however, and those that can be done in its absence may still increase their effectiveness when design data is available. Its importance for the various

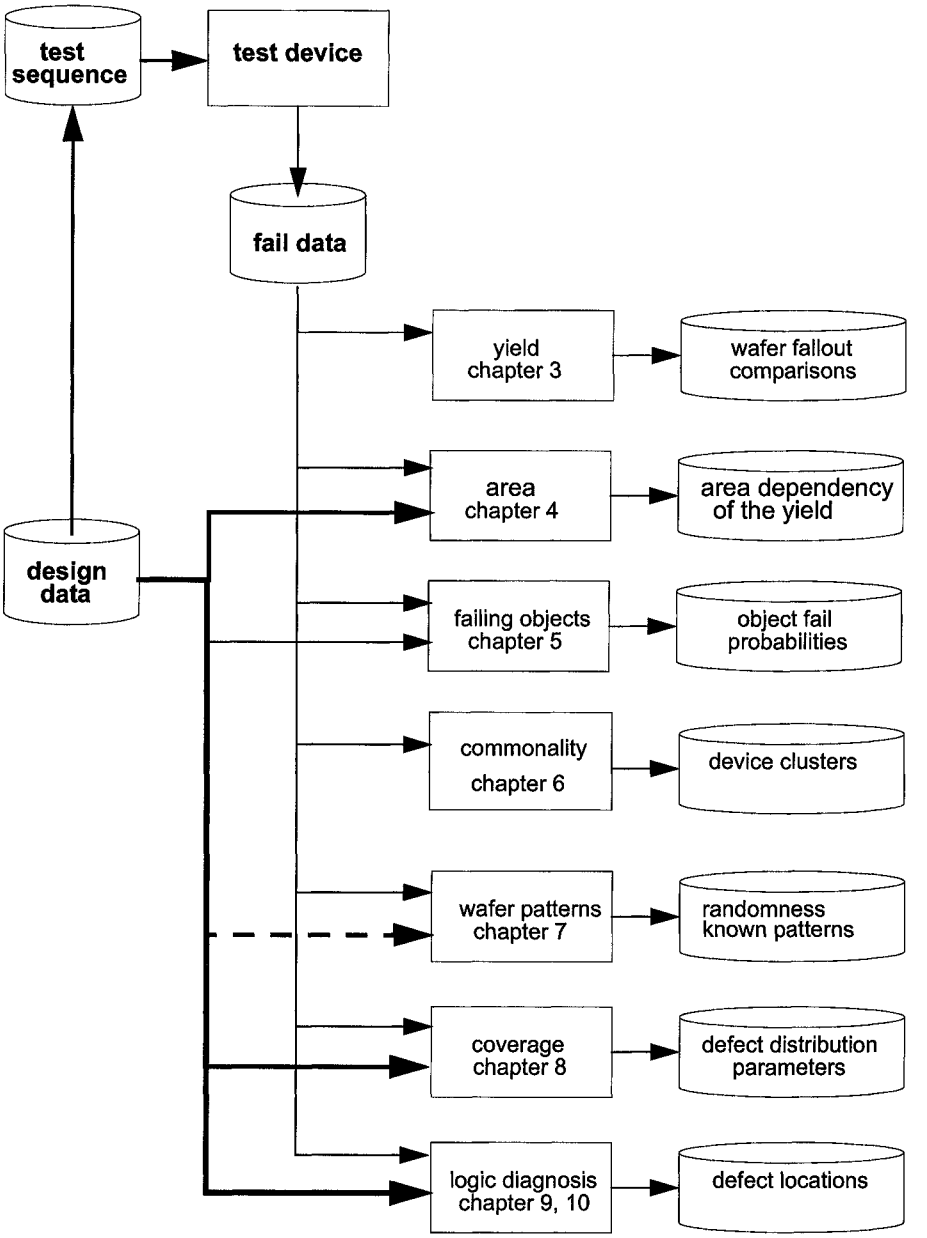


Figure 1 Schematic diagnostic flow

diagnostic analysis techniques is indicated roughly by the heaviness of the arrows from the design data icon to the analysis icons. The results of the various analyses is indicated briefly in the data icons on the right.

The diagnostic techniques to be discussed in this book have certain data collection requirements, and these requirements have repercussions for both design and test. For test, because the required data needs to be collected and made available to the diagnostic software; for design, because it has to be possible to collect the test data, and because certain design data have to be available during diagnosis. An example of the former is the latch contents after the application of a test pattern, while an example of the latter is the positions of those latches in the scan chains. It will become clear from the main text what those requirements are, but they will be briefly summarized in Chapter 11 for the sake of convenience.

Chapter 2

Statistics

In the present chapter, I will briefly review some statistical distributions that are used often in this book. I will also discuss some statistical techniques that are important in this book, but that may not be very well known. Good introductions to practically all the statistical techniques used here can be found in, for example, Lindgren [38], or Casella and Berger [10]. The group of techniques that are used most often is centered on the likelihood function, but in some instances bootstrapping will be used as well. They will be described briefly.

Many chapters in this book rely strongly on the difference between random variables and model parameters. To accentuate this difference, the general custom will be followed of labeling random variables with upper case letters, and parameters with lower case ones.

1 STATISTICAL DISTRIBUTIONS

The number of distributions used in this book is small, basically the binomial and Poisson distributions, and some variations on them.

1.1 Binomial and multinomial distributions

The binomial distribution is that of the number of fails in a given number of attempts, given the fail probability. To simplify notation, I will use Feller's one [22] for the probability density function of the binomial distribution. The probability that n fails will be observed in N tries if the fail probability is p is

$$b(n;N,p) = \binom{N}{n} p^n (1-p)^{N-n} \quad . \tag{2.1}$$

The expected value of n is Np, and its variance is Np(1 - p).

When p is very close to 0 or 1, the relationship between the expected value of n and its fluctuations becomes very simple. When p is very small, it can be neglected with respect to 1. The standard deviation of n is then roughly equal to the square root of its expected value. Likewise, when p is very close to 1, the standard deviation of N - n is roughly equal to the square root of that number. In other words, when p is either very small or very large, the typical size

of the variations in the number of the rarer events (failures with very low fail probability, passes otherwise,) is roughly equal to the square root of the number of those events, and does not depend on the number of the more common events.

The binomial distribution can be generalized by compounding [12]. In that case, the binomial parameter p is a random variable itself, with a probability distribution h(p). The expected value of p will be indicated by

$$\langle p \rangle = \int h(p) p \, dp \,, \tag{2.2}$$

and its variance by $\sigma^2(p)$.

The expected value of the number of fails in the compounded distribution equals N<p>, and its variance is equal to

$$N\langle p \rangle (1 - \langle p \rangle) + N^2 \sigma^2(p) \,. \tag{2.3}$$

The first term in this variance is the standard binomial one, the second one is the contribution from the finite width of h(p). It has the important consequence that, when N becomes large, the ratio of the standard deviation of the number of fails to its expected value does not go to 0, as in a pure binomial distribution, but, instead, to the finite ratio $\sigma(p)/\langle p \rangle$. Even with large N, therefore, the variability in the number of fails cannot be ignored, and can, in fact, be substantial.

Another extension of the binomial distribution is the multinomial [22] one, in which more than two outcomes are possible, each with their own probability of occurrence. There is no standard notation for this distribution. The one that will be used here was inspired by that for the binomial distribution. If there are k choices, with probabilities p_i for i = 1, ..., K, the probability $P(n_1, ..., n_k)$ of n_i occurrences of choice i is given by the multinomial probability

$$m(\{n_i\};N,\{p_i\}) = \frac{N!}{\prod n_i!} \prod p_i^{\,n_i} \,, \tag{2.4}$$

where n! stands for the factorial of n, and all products are from i = 1 to k. The sets $\{p_i\}$ and $\{n_i\}$ obey the obvious sum rules $\sum_i p_i = 1$, and $\sum_i n_i = N$.

By summing over all n_i except one, say n_j, we find that the probability of n_j occurrences out of N trials equals $b(n_j;N, p_j)$ Consequently, the expected value of any n_i equals Np_i.

1.2 Poisson and compound Poisson distributions

The Poisson distribution is that of the number of occurrences of some event in a given space, given the probability of an occurrence in a unit amount of space, and given that occurrences are independent. Typical examples are the number of events in a given amount of time or the number of defects in a given area. The latter example is the important one in this book.

The probability of an occurrence in a unit amount of space is also called the strength of the Poisson distribution. When the strength is ν, the probability of n occurrences in a unit amount of space equals

$$\frac{\nu^n}{n!}e^{-\nu}. \tag{2.5}$$

The expected value and variance of n are both equal to ν. The probability of no occurrence is $e^{-\nu}$.

A more general version of the Poisson distribution is the compound Poisson distribution, in which the strength ν is itself a random variable with some distribution $h(\nu)$ [12]. The probability of n occurrences is then equal to

$$\int h(\nu)\frac{\nu^n}{n!}e^{-\nu}\,d\nu. \tag{2.6}$$

It is easy to show, by interchanging integration and summation, that the expected value μ of n is now equal to $\langle\nu\rangle = \int h(\nu)\nu d\nu$, and that its variance equals $\mu + \sigma^2(\nu)$, in which $\sigma^2(\nu)$ is the variance of the Poisson strength ν. Compounding, therefore, always increases the variance of the observed yields.

Another effect of compounding is to increase the probability of no occurrences at all, at least when $\langle\nu\rangle$, the expected number of occurrences stays the same. This probability equals

$$p_0 = \int h(\nu)\,e^{-\nu}\,d\nu. \tag{2.7}$$

That compounding always increases p_0 compared to its Poisson value can be proven as follows. It is easy to see that $e^{-\nu} \geq e^{-\lambda} - (\nu - \lambda)e^{-\lambda}$, because $-(\nu - \lambda)e^{-\lambda}$ describes the tangent to $e^{-\nu}$ at $\nu = \lambda$, and because $e^{-\nu}$ curves upwards. The constant λ in the inequality can be any number, but is taken here as the mean of ν. In the compound model, we then find that $p_0 \geq e^{-\langle \nu \rangle}$.

1.3 Negative binomial distribution

An important example of a compound Poisson distribution is the negative binomial one. It emerges when the compounding function is the gamma distribution. In other words, when

$$h(\nu) = \frac{\nu^{\alpha - 1} e^{-\nu\alpha/\mu}}{\Gamma(\alpha)(\mu/\alpha)^{\alpha}}, \tag{2.8}$$

in which α is a positive parameter, called the cluster coefficient, μ is the mean of ν, and $\Gamma(x)$ is the gamma function of x. The negative binomial distribution can be generated in other ways than by compounding a Poisson distribution [12], but compounding is a very convenient one.

The probability of n occurrences in the negative binomial distribution equals

$$\frac{\Gamma(\alpha + n)}{n!\Gamma(\alpha)} \left(\frac{\mu}{\alpha}\right)^{n} \left(1 + \frac{\mu}{\alpha}\right)^{-\alpha - n}. \tag{2.9}$$

The expected value of n is μ, and its variance is $\mu + \mu^2/\alpha$. The probability of no occurrences equals

$$(1 + \mu/\alpha)^{-\alpha}. \tag{2.10}$$

The cluster coefficient functions as a sort of scale that separates the region $\alpha \gg \mu$ in which the negative binomial distribution is very similar to a Poisson distribution, from that in which the two are very different.

The cluster coefficient is related to the distributional parameters of the compounded Poisson distribution through

$$\frac{1}{\alpha} = \frac{\sigma^2(n) - \mu}{\mu^2}, \tag{2.11}$$

which suggests a rough estimate of the inverse of the cluster coefficient from actual data. Using the inverse of α rather than α itself is more meaningful, for the former vanishes in the limit of a pure Poisson distribution. It will be indicated by γ.

Equation (2.11) can also be seen as a generalized definition of a cluster coefficient, one that goes beyond its definition in the gamma function. As such, the estimate obtained from equation (2.11) need not be positive, even though α is in Equation (2.8). There is in fact no reason why the generalized cluster coefficient should always be positive, and we will find in Chapter 4 that it oftentimes is not.

Large values of γ correspond to strong clustering, and small values to little clustering. For example, when we calculate γ for the compound binomial distribution, it equals $-1/N$ in the case of no compounding, but then increases smoothly to positive values. It can become arbitrarily large when $\sigma(p)$, the width of the compounder, becomes large.

2 LIKELIHOOD

In many situations, the data that are collected have some known statistical properties, except that some parameters of the underlying distribution are not known. One of the goals of collecting the data is to estimate those parameters. An example is the passes and fails of an embedded SRAM on the chips. It is assumed to fail with a probability that may depend on the wafer column in which the chip is located. The numbers of passing and failing SRAMs per column have Binomial distributions, and one statistical analysis that can be done is estimating the fail probabilities of those distributions, and determining whether they are column dependent or not.

A standard way of constructing estimators for the parameters of a distribution is the maximum likelihood method. It relies on the so called likelihood function. This approach is described in some detail in the statistics books mentioned previously [10, 38], and in more detail in the book by Edwards [20].

The likelihood function is numerically proportional to the probability that the observed data would have been obtained, given a specific set of distributional parameters. By considering the likelihood function as a function of the parameters, with the observed data as fixed values, the probability is trans-

formed into a function of the parameters. The likelihood function is proportional to it, for factors that do not depend on the parameters turn out to be irrelevant.

In the example given above, the probability that the given numbers of passes and fails in the various columns would have been observed is equal to the product of a number of binomial probabilities, one for each column, and each one with its own fail probability. With the actual observations fixed, this product is a function of the column fail probabilities. It will vary when the fail probabilities are varied.

2.1 Maximum likelihood

The maximum likelihood method is based on the assumption that the best estimate of the physical fail probabilities, the ones that govern the actual passes and the fails on the physical wafers, is that set of probabilities that maximizes the likelihood function. It obviously depends on the observed data, because different sets of data will put the maximum of the likelihood function in different places.

The likelihood function is generally indicated by L. If we continue the example, L is function of the column fail probabilities p_i. To make the dependence on the observed data explicit, they are sometimes added to L as a condition:

$$L = L(p_1, \ldots, p_k | \text{data}) \,. \tag{2.12}$$

Given the data, the first step in the analysis is estimating the fail probabilities. As mentioned above, this is done by maximizing L, and entails two steps. First, the extrema of L have to be found, which can be done by solving

$$\frac{\partial L}{\partial p_i} = 0 \tag{2.13}$$

for each i (column in the example). Second, the maximum has to be found among the extrema. A maximum corresponds to an extremum where the matrix with elements

$$\frac{\partial \, \partial L}{\partial p_i \partial p_j} \tag{2.14}$$

is negative definite. In most cases, Equations (2.13) have only one solution, and that solution can trivially be shown to correspond to a maximum. In some cases, however, multiple solutions may have to be considered, and the negative definiteness of the matrix of second derivatives of L has to be established using numerical methods.

Strictly speaking, the found maximum should also be compared to values of L on the boundary of the range of the parameters of the distribution, for maxima on those boundaries usually do not obey Equation (2.13). In most cases encountered in this book, L trivially vanishes on this boundary, and is positive in the interior region of the range, so the question of maxima on the boundary does not occur.

There are in fact situations in which Equations (2.13) are so complex that they cannot be solved even with moderate effort. If all else fails, the maximum of L can always be found by reliable, but numerically more demanding maximization routines [44].

The estimates of the parameters are random variables, for they depend solely on the observations, and not on the parameters to the underlying distributions. These estimates, therefore, have a distribution, but that distribution is usually not known. Fortunately, for large sample - that is, large wafers in the example - the distribution of the estimates is approximately normal with a covariance matrix equal to minus the inverse of the matrix of second derivatives. The latter matrix is therefore not only important for establishing maximality of extrema, but also for gauging the accuracy of the estimates.

2.2 Likelihood ratio

The likelihood function is used not only for estimating parameters, but also for deciding whether one particular statistical model is better suited to explain the data than some other potential model. The manner in which that will be done in this book can be demonstrated with the example that we have been using in this section.

In the running example, there are two reasonable models. The first one, called the heterogeneous model, is the one that we have been using: one fail probability per column. The second one is called the homogeneous model, and is a simplification of the first: one fail probability for all columns. The heterogeneous model is always more accurate, for it has more adjustable parameters. The homogeneous one is more parsimonious, and may be preferred for that reason.

Even when the homogeneous model is correct, the numbers of fails on any given column will not always be equal to the mean, but will fluctuate around it. Small deviations of the numbers of fails around their respective means will not necessarily invalidate this model, therefore; only large deviations can do

that. The question is, "how large should the deviations be before we should discard the homogeneous model and assume the validity of the heterogeneous one ?"

This question can be answered to some extent with the likelihood ratio

$$\Lambda = \frac{L(\hat{p}\,|\,\text{data})}{L(\hat{p}_1, \ldots, \hat{p}_k\,|\,\text{data})}, \tag{2.15}$$

in which a carrot ($\hat{}$) over a variable indicates the maximum likelihood estimates of that variable, and $\hat{p}$ is the maximum likelihood estimate of the single fail probability in the homogeneous model.

Λ will never exceed 1, for both numerator and denominator are maximized, and the space of the p values is a subset of the space of the p_i values. Therefore, if $L(\hat{p})$ were larger than $L(\hat{p}_1, \ldots, \hat{p}_k)$, the latter could be increased by replacing the estimates of p_i by the estimate of p, contrary to the assumption that it is maximal.

Λ is a convenient measure of the extent to which the observed deviations match the expected ones; in other words, it is a good indicator of column similarity. If the homogeneous model reflects the true state of affairs, it will be close to 1, but not equal to it, because of statistical fluctuations. If this model is not the correct one, Λ will be much smaller than 1.

How much Λ should differ from its maximum value before the homogeneous model can be rejected depends of course on the size of the expected statistical fluctuations, which depend on the numbers of columns and chips per column through N_{DF}, the number of degrees of freedom. This number equals, in this case, $\sum(m_i - 1)$, in which the sum is over all the columns, and m_i is the number of chips in column i.

Under the null hypothesis that all columns have the same fail probability, $-2\ln\Lambda$ has approximately the chi-squared distribution with N_{DF} degrees of freedom [10]. Consequently, under the null hypothesis, the expected value of $-2\ln\Lambda$ equals N_{DF}, and its variance $2N_{DF}$.

If the null hypothesis is correct, the actual value of $-2\ln\Lambda$ is expected to be within a few standard deviations of its mean. A more convenient measure of column similarity, therefore, is the ratio

$$\rho = \frac{-2\ln\Lambda - N_{DF}}{\sqrt{2N_{DF}}}. \tag{2.16}$$

Any significant deviation of Λ from its mean leads to a large value of ρ, and indicates that one or more columns differ significantly from the others. Moreover, when the number of degrees of freedom is large, as it typically is, the chi-square distribution can be replaced by a normal one with the same mean and variance.

3 BOOTSTRAPPING

When estimating the values of distributional parameters or other distribution related quantities, we often would like to the know the accuracy of those estimates, in addition to the estimates themselves. When the statistical distribution of the estimator is known, the accuracy of the estimate can be obtained from the variance of the estimator. Oftentimes, however, the distribution is not known, or, if known, is valid only in the limit of very large samples. In such cases, other means have to be employed to get a sense of the accuracy of the estimators.

The variance of an estimator could also be estimated, and trivially so, if many samples were available. For then we could estimate whatever quantity we are interested in in each sample, and compare the results. Unfortunately, there is only one sample. It is possible, however, to create artificial samples, with many of the same statistical properties as real samples, and use these artificial samples as substitutes for the latter. This technique is called bootstrapping [41].

In bootstrapping, a large number of secondary samples are generated from the original one, called the primary sample. The secondary samples have the same size as the primary one, and are formed by randomly selecting the units of the sample (embedded SRAMs in our running example) from the original sample. The selection is done sequentially, and with replacement (so the same unit can be selected multiple times.)

The bootstrap assumption is that the statistical properties of primary samples are approximately the same as those of the secondary samples, based on a single primary one. For example, a single fail probability for the embedded SRAMs, valid for all columns, can be calculated for each secondary sample, and the distribution of these fail probabilities is assumed to approximate that of the maximum likelihood estimate of the fail probability in the primary sample.

Chapter 3

Yield Statistics

Testing of electronic devices consists of applying a sequence of test operations to those devices. Each test operation, or test for short, may cause one or more devices to fail. As the device failures are caused by defects introduced by, or at least during the manufacturing processes, the progressive increase in chip fallout during testing may provide us with information about these defects and about the processes that caused them.

Extracting such information is made difficult, however, by the inevitable statistical fluctuations in the real fallout data. After all, whether a chip has the defects that will make it fail at or before a certain test is a matter of chance: on average, the same fraction of chips will fail, but the actual number will fluctuate around this average, depending on the particular batch of chips that entered the test process. The size of these fluctuations will of course decrease when the size of the batch of chips increases, but the batch has to become very large for the yield fluctuations to become negligible.

The chapter will address several of these statistical issues, and is divided into roughly three equal parts. In the first two sections, the statistical aspects of the yield, and in particular of the defect level, are studied. The predictions of the theory are compared with experimental data, and it turns out that the distribution of wafer yields is much wider than can be explained by assuming that the fail probability of the devices is the same on all wafers.

In Section 3, the statistical analysis is generalized to include the partial yields at all the test steps, not just the overall yields. It will be shown that, under very general assumptions, the statistical properties of the fallout data are of a rather simple kind, and can easily be estimated from the observed data. Any comparison between theory and practice can therefore be done using standard statistical techniques.

The final sections are devoted to an analysis of the distributions of sort codes for different wafers. This analysis is important in its own right, because it focuses on which wafers have the same fallout behavior, and, presumably, the same process histories, but it also provides more insight into the excessively wide wafer yield distributions.

1 YIELD AND DEFECT LEVEL

The simplest information that can be obtained from the results of applying a test sequence to N chips is N_{pass}, the number of chips that passed all tests. The same information is often presented as the perceived yield Y, which equals N_{pass}/N. This section will focus on Y and some of its statistical properties

1.1 Final yield

Y is a random variable whose value depends on the particular batch of chips that was submitted to test. Its expectation value will be indicated by y. Because chips either pass or fail, y is also equal to the probability of a chip passing the test. The variance of Y is $y(1-y)/N$. The distribution of Y is assumed to be independent of the particular batch of chips being tested. Consequently, Y is likely to be in the range

$$y \pm \sqrt{y(1-y)/N}, \tag{3.1}$$

and, if the square root term is small compared to the yield, Y is a good estimate of Y, in the sense that y is in the range $Y \pm \gamma\sqrt{Y(1-Y)/N}$, where γ is some number that depends on the required confidence, and is typically equal to three.

It is also often desirable to focus on a subset of M devices from the batch. For example, this subset could be a single wafer from a lot of many wafers, in which case M is the number of devices on the wafer, and N is the number of devices in the lot. But the subset need not be as obvious as a wafer; it could be smaller, like a region on a wafer, or larger, like a set of wafers that are known to have been processed through the same tools. If N is large, Equation (3.1) also implies that this subset should have a yield in the range of

$$Y \pm \sqrt{Y(1-Y)/M}. \tag{3.2}$$

1.2 Defect Level

Y may not be the same as the fraction of chips that are truly defect free, and usually isn't. This latter fraction will be indicated by Y_0. Y_0 is a random variable, in the sense that it depends on the batch that is being tested. It is always unknown, however, and it may even be very hard to define. For example, consider a chip that is good in the sense that it would pass all tests that anyone might want to apply, but that has a weak defect that will grow rapidly

when the chip is being used and that will cause a failure after only a few hours of use. Is this chip defect free and contributes to Y_0, or not ? I will ignore these subtleties and assume that Y_0 is well defined but unknown.

Y_0 is a random variable, just like Y. Its expectation value will be indicated by y_0. The existence of chips that pass all tests even though they have defects means that, in fact, there are three kinds of chips: those that are shown to be defective by the test, those that are defective but pass all tests, and those that are defect free. The probabilities for these three categories are 1 - y, y - y_0 and y_0, respectively.

The difference between Y and Y_0 is the fraction of chips that have defects but passed all tests. The goal of test, of course, is to make this fraction as small as possible. A more standard measure of goodness of test is the ratio $DL = (Y - Y_0)/Y$, called the defect level.

The actual number of defective chips that pass all tests will be indicated by N_{def}. It is related to the usual defect level DL by

$$DL = \frac{Y - Y_0}{Y} = \frac{N_{def}}{N_{pass}}. \tag{3.3}$$

Even though DL is the standard measure for test escapes, it is easier to work with N_{def} than with DL. Its statistical properties can easily be determined from the definition (see Appendix A)

The most interesting statistical properties of N_{def} are those with N and N_{pass} known. In that case, N_{pass} chips passed all tests, but N_{def} of them are still defective. It is shown in Appendix A that N_{def} has the binomial distribution with expected value

$$\langle N_{Def} \rangle = N_{pass}\left(1 - \frac{y_0}{y}\right), \tag{3.4}$$

and variance

$$\sigma^2(N_{def}) = \langle N_{def} \rangle \frac{y_0}{y}, \tag{3.5}$$

which is approximately equal to $<N_{def}>$ when y_0 and y are close.

These results depend on the actual value of N_{pass}, which varies from batch to batch, and on y_0. As the latter is generally unknown - in fact, all chips that

pass the test may be defective, in which case Y_0 vanishes - *no reliable estimate can be made of the expected number of field failures without either knowing how complete the test is, or assuming that the test is almost complete.* The question of how to estimate DL when some measure of test completeness is available will be taken up in Chapter 8.

Even when the test completeness is known, no good estimate can be obtained if N_{def} is small (as it should be,) because of the unavoidable statistical fluctuations. For Equation (3.5) shows that the standard deviation of N_{def} is approximately equal to the square root of its expectation value. The fluctuations are therefore not important if that expectation value is much larger than 1, and N_{def} is roughly equal to it. When the expectation value is of order 1, however, $\sigma(N_{def})$ will be comparable to, if not larger than N_{def}, and the fluctuations will determine the value of the latter.

2 EXAMPLE: EXPERIMENTAL WAFER YIELDS

In a recent experiment, one ASICs part was tested extensively. The goal of this experiment was to gauge various test methods, according to their effectiveness in detecting defects, as well as to gauge the availability, efficiency and accuracy of existing diagnostic methods in determining the locations of the defects that caused ICs to fail. More detailed information about this experiment is given in Chapter 9.5. Here, only the wafer yields will be considered.

Each wafer contained 329 devices. The experiment looked at 147 wafers, divided over nine lots, with varying numbers of wafers per lot. The yield results are shown in Figure 2 using box charts.

The box charts summarize the yield distributions within a lot, shown on the X-axis. Lot_6 and Lot_7 consisted of a single wafer each, and no boxes are shown for them, only a a single + mark, indicating the yield of the single wafer. The widths of the boxes indicate the number of wafers in the lot. The tops and bottoms of the boxes correspond to the 25th and 75th percentiles of the yield distributions, respectively, and the lines dividing the boxes show the positions of the medians. The average yields are indicated by the '+' symbols. The thin lines extending from the tops and bottoms of the boxes are called whiskers, and show the ranges of the yields, but they are restricted in length to one and half times the height of the associated boxes. Wafers with yields outside this range are considered to be outliers, and are indicated by the black dots. Lot_9, for example, has four outliers with excessively low yields.

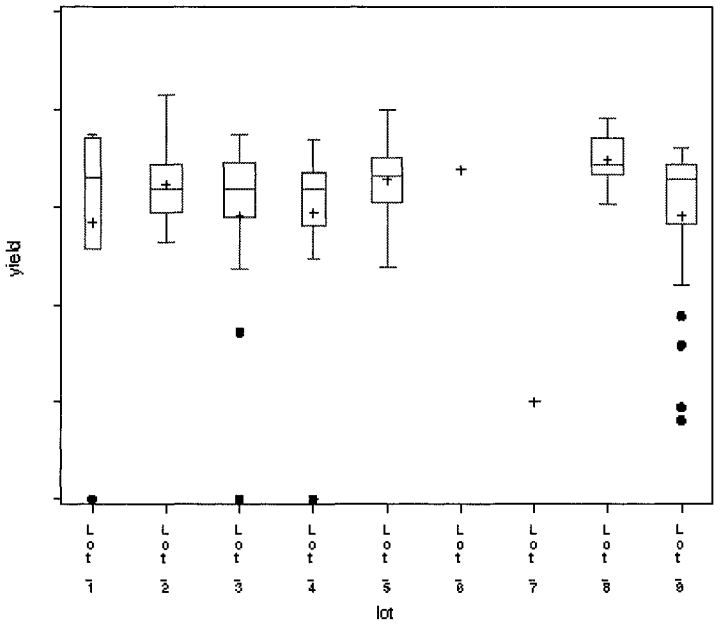


Figure 2 Distribution of individual wafer yields for several ASIC lots

The figure shows a reasonably stable distribution of lot yields, in that the yields of the different lots are more or less equal (with the exception of the singular Lot_7, which consisted of a single, low yield wafer.)

The observed distributions of the lot yields are much wider than expected, however. If the distributions were purely binomial, the height of the boxes, called the interquartile range, would be roughly 1.35 standard deviations in the normal approximation [15], which is considerably less than observed.

One possible explanation for such a larger than expected variation is that the probability y of a device passing all tests is not a constant, but is itself a random variable with some distribution. This may occur if, for example, different wafers were operated upon by different tools, each one of which performs the same function, but with different tolerances and characteristics. Slight tool differences may lead to slightly different values of y, resulting in a yield distribution that is wider than when y is the same for all wafers. Other causes are discussed in, for example, [47].

The distribution of y values is often called a compounding distribution (see Chapter 2), because an individual wafer yield is the compound result of random statistical variations around y, and the random variations of y due to the random choice of tools used to perform required process operations. That compounding widens the yield distribution was proven in Chapter 2, where it

was shown that a non-trivial yield distribution increases the variance of the number of failing devices by the variance of that yield distribution.

Even though compounding explains the wide yield distribution, it is not an attractive approach to yield analysis, for any yield distribution can be explained by choosing a suitable compounder. Unless the compounding function can be related to the tool characteristics, no useful information can be obtained by finding one that merely reproduces the observed yield distribution. A different, and more productive approach will be developed, among other things, in the next sections.

3 TEST FALLOUT

The experimental results in the previous section show that individual wafer yields can differ substantially from the overall lot yield. This excessive yield variation indicates that not all wafers were subject to the same defect producing mechanisms. To locate the source of these differences, a more detailed analysis has to be made of the fallout history, the record of how many chips failed for the first time during the various steps in the test sequence.

3.1 First fail probabilities

The overall test sequence was described in some detail in Chapter 1. It consists of a, potentially large number of steps, many of which can be combined in useful major steps, like IDDq, IO leakage, and deterministic tests at nominal voltage and temperature. Every chip is assigned a sort code, indicating in which major step it failed first during the application of the test sequence. In its crudest form, the fallout history is merely the record of the sort codes.

There is no need, however, to restrict ourselves to the major test steps. Many major steps can be subdivided into smaller steps, maybe as small as the application of a single test pattern. Usually, no standard sort codes are available for such smaller steps, but it is clear that a fallout record can still be maintained. This section will focus on the statistical aspects of the fallout history, regardless of the step size. The test steps will be labeled by an index k, running from 1 through k_f. k_f will be large if the test steps are chosen to be very small.

To perform a statistical analysis of the fallout data, each chip needs to be assigned to one of many, mutually exclusive test result buckets. In the previous section, there were only two buckets, one for chips that passed all the tests, and one for the ones that did not. A far more sophisticated choice of

buckets is the complete set of test steps that showed the chip to be defective. It is rarely known at what test steps a given device fails, however, for the cost of applying the complete test to every chip is prohibitive. A better choice is to label each failing chip with the test step at which it failed first, that is, by the sort code, if such a code exists. There are then $k_f + 1$ different buckets, instead of only two as in the previous section.

The probability of a chip failing for the first time at a particular test step depends on the manufacturing process and on the test sequence, but is assumed to be stable (that is, time invariant.) It is therefore well defined, and will be indicated by d_k, with k being an index that labels the test steps 1 through k_f. d_k is the probability of a rather complex event, namely that of failing in test k and passing all preceding test steps. d_{k_f+1} is the probability that the device will not fail at all.

There is, therefore, a set $\{d_k\}$ of first failure probabilities associated with the manufacturing process that produced the chips. As a chip does not fail at all, or, if it does, does so for the first time at some test step,

$$\sum_{k=1}^{k_f+1} d_k = 1 . \tag{3.6}$$

First failures of different chips are independent events, and, if the process is stable and all devices have the same process histories, all chips are subject to identical sets of such first fail probabilities.

d_k is obviously related to the yield y_k at the completion of the k^{th} test. Because the latter is the probability that the chip does not fail any of the first k tests in the test sequence,

$$y_k = 1 - \sum_{i=1}^{k} d_i . \tag{3.7}$$

The yield at the completion of all the tests has, until now, been indicated by y. It is, of course, the same as y_{k_f}, but I will continue to use the y symbol for the final yield, rather than the technically more correct y_{k_f}. Combining Equations (3.6) and (3.7), or its definition, shows that d_{k_f+1} equals y.

3.2 Statistical distribution of fails

The fate of a chip will take one of $k_f + 1$ forms: either it fails one of the k_f tests, or it passes all tests. Consequently, the joint distribution of the numbers of chips first failing in any of the k_f tests is multinomial (Chapter 2.1.1,) with parameters d_k, $k = 1, ..., k_f$, and y. This observation forms the basis of the statistical treatment of the yield curve. Of immediate interest are some special cases, the mathematical details of which can be found in Appendix A.

The number of chips failing for the first time at the k^{th} test has the binomial distribution with average Nd_k and variance $Nd_k(1 - d_k)$. A related special case is that of all tests from test 1 to k grouped into one test. This case is of great interest for it is related to the yield curve as it is usually shown. The fraction Y_k of chips that pass all tests through the k^{th} one is the perceived yield after the k^{th} test. With the probability y_k of passing these tests as given in Equation (3.7), the number of chips passing all of them has the binomial distribution with expected value Ny_k.

These results are of course entirely expected, but they ignore - or average out - what happens in the tests preceding any particular test step. The number of chips that first fail a specific test does depend on the outcome of the previous tests, however. After all, when the tests preceding the k^{th} one find all chips to be defective, the number of chips failing first at test k is zero with probability one.

It is easy to show that what happens at test k depends on the previous tests only through the number of chips that passed all the preceding tests. Let K be the number of devices that did not fail any of the tests preceding test k. It is another random variable, and has the binomial distribution with expected value Ny_{k-1}. The number of chips failing test k is N_k, and has again the binomial distribution, but, when K is given, with expected value Kd_k/y_{k-1}.

The statistical properties of N_k depend on d_k, y_{k-1}, and K. This is awkward if we need to compare results from different batches with different values of K, even if y_{k-1} is known to be the same in all batches: the differences in N_k could be caused by differences in d_k, which is interesting, or by differences in K, which is trivial. K will vary because of normal statistical fluctuations, but those variations can be ignored if their effect on N_k is small compared to the normal statistical fluctuations in N_k with a fixed K. It is shown in Appendix B that the fluctuations in K can be ignored when y_k is not too small compared to y_{k-1}. It is therefore usually legitimate to ignore the

dependencies between different tests, and to approximate the distribution of N_k by Equation (A.6), using the observed, the averaged, or the expected value of K.

4 MEASURING FIRST FAIL PROBABILITIES

In the previous section, all wafers or lots share the same process histories, and their $\{d_k\}$ sets should be the same. On the other hand, if these first fail probabilities differ significantly, then it can safely be assumed that the process histories of the batches are not the same either.

This question was alluded to in Section 2, where it was noted that different wafers in the same lot can have very different yields. Clearly, when the yields of two wafers are very different, the wafers must have had different process histories, and this was modeled in Section 2 by giving y, the probability that a device on a wafer will pass all tests, a non-trivial probability distribution. On the other hand, giving y a probability distribution does not provide any more insight than what was already available from the yield distribution.

The multinomial machinery is a different approach to the same problem. It makes it possible to study differences between wafers at a much more detailed level than the overall yield, and even when the final yields are similar. In particular, it can identify those specific test steps where the differences are most pronounced. Knowing the identity of those test steps can then provide information about the possible defects that gave rise to the differences, because different test steps are typically sensitive to different defect types.

In this section, a statistical method will be described that uses the fallout history to estimate the first fail probabilities for any batch of devices, and that can gauge the extent to which the histories of different wafers differ.

4.1 Fallout histories

The wafer by wafer fallout histories can be presented as a matrix, in which the rows correspond to wafers, and the columns to the different steps in the test sequence (see Table 1.). The number of rows in the matrix is I, the number of wafers, while the number of columns equals $k_f + 1$, corresponding to the number of outcomes of the test sequence. The final column has the virtual test step index $k_f + 1$, and contains the data for the devices that passed all tests.

The row/column entries n_{ij} surrounded by the heavy line in Table 1. are the number of chips on wafer i failing test step j, while not failing any of the

	step 1	step 2	...	step k_f	pass
wafer1	n_{11}	n_{12}	...	n_{1k_f}	$M - \sum_j n_{1j}$
wafer2	n_{21}	n_{22}	...	n_{2k_f}	$M - \sum_j n_{2j}$
...	...	...	...	...	...
wafer 1	n_{l1}	n_{l2}	...	n_{lk_f}	$M - \sum_j n_{lj}$
Total	N_1	N_2		N_{k_f}	$N - \sum_j N_j$

Table 1. Wafer by wafer test histories

test steps preceding j. The final column, with matrix elements $n_{i(k_f+1)}$, indicates the number of chips on wafer i that did not fail any of the tests. The column totals N_j correspond to the test history used in Section 3, and N_{k_f+1} equals $N - \sum_j N_j$. The row totals equal M, the number of chips on a wafer, but are not indicated in the table for they are typically all the same (except when there are test or data integrity problems). The total number of chips on all the wafers equals N, as before.

The n_{ij} are usually not the same for all wafers i, but they are similar when all chips on all wafers are subject to the same defect mechanisms, that is, when they all have the same multinomial parameters $\{d_k\}$. In matrix terminology, this means that a constant d_k is associated with each column. This case will be referred to as the homogenous model. The alternative is the heterogeneous model, in which different wafers may have been subject to different defect mechanisms. The homogeneous model has been assumed so far, but the wide yield distributions displayed in Figure 2 may force us to consider the heterogeneous one.

4.2 Maximum likelihood estimation

In the homogeneous model, the multinomial parameters are the same for the same test step, and the different wafers can be viewed as different, inde-

pendent random samples from the same population. The resulting distribution of the n_{ij} values is the product over all wafers of identical multinomial distributions. The d_k parameters can then be estimated from the fallout data using the maximum likelihood method (see Chapter 2.2.1).

In this case, the likelihood function is indicated by $L(d_k)$, and is the product of identical multinomial distributions, with the random variables replaced by their measured values. Estimates for d_k are obtained by finding those values of the parameters for which $L(d_k)$ is maximum. The result of the estimation is indicated by $\widehat{d_k}$, and

$$\widehat{d_k} = \frac{1}{N} N_k . \tag{3.8}$$

If different wafers are subject to different defect mechanisms - the heterogeneous model - their first fail probabilities will differ, but the distribution of the n_{ij} is still a product of multinomial distributions, be it with different multinomial parameters. The parameter set for wafer i will be indicated by the set $\{d_{ik}\}$, and can still be estimated from the fail data using the maximum likelihood method. The likelihood function for this scenario will be indicated by $L(d_{ik})$. Maximizing it leads to the estimates

$$\widehat{d_{ik}} = \frac{1}{M} n_{ik} . \tag{3.1}$$

The problem is to recognize when to use identical multinomial parameters, and when to use different ones. This can be done using the likelihood ratio (see also Chapter 2.2.2.) This ratio will be indicated by Λ, and equals $L(d_k)/L(d_{ik})$, with the multinomial parameters evaluated at their respective maximum likelihood estimates:

$$\Lambda = \prod_i \prod_{j=1}^{k_f+1} \left(\frac{N_i M}{N n_{ij}} \right)^{n_{ij}} . \tag{3.9}$$

Λ is a good indicator of wafer similarity. Under the null hypothesis that all wafers have the same multinomial parameters, $-2 \ln \Lambda$ approximately has the chi-squared distribution with k_f degrees of freedom. A more convenient measure of wafer similarity, therefore, is the ratio ρ, defined in the same chapter.

Any significant deviation of Λ from its mean leads to a large value of ρ, and indicates that one or more wafers differ significantly from the others.

5 COMPARING WAFERS

Figure 3 shows the distributions of the sort codes for each wafer in one

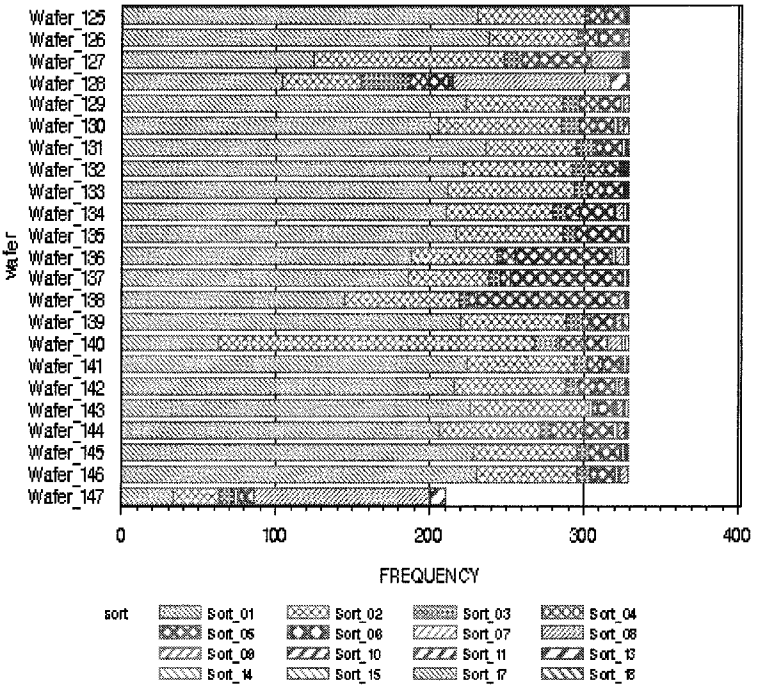


Figure 3 Distribution of test sorts by wafer

particular lot - Lot_9 - of the design introduced in Section 2. The bars refer to different wafers, and the different patterns correspond to the different sort codes. The sizes of the patterned sections in each bar constitute in fact the fall-out history of the associated wafer. It is not relevant what tests the various codes refer to, but, as an example, some refer to different environmental stresses applied when the LSSD test patterns were applied.

The information contained in such fallout histories is quite extensive. The fallout history is dominated by the Sort_1 and Sort_2 categories, although there are some exceptions like Wafer_128, and Wafer_147. The outliers mentioned in Section 2, are easily visible in the chart. It is also obvious from the

chart that one of them, the last one, is abnormal, because of its excessively large number of Sort_2 fails.

This figure demonstrates clearly that the homogeneous model is not realistic: different wafers can have very different mixes of sort codes, and, therefore, must have had very different process histories. To progress, however, it is more productive to identify those wafers that share similar fallout histories; in other words, to group the set of wafers into clusters, each cluster containing those wafers that seem to have failed more or less similarly.

A general clustering algorithm will be discussed in Chapter 6.4. What is required for this algorithm is a measure for the degree of commonality between two clusters. A natural choice for such a commonality measure is the ratio ρ, defined in Chapter 2.2.2, for the union of the two clusters, with a small value of ρ indicating a high degree of commonality.

When the clustering process with this commonality measure is applied to the experimental test results from Section 2 with a threshold of three, the set of 147 wafers breaks up into twenty seven clusters. The largest cluster has thirty eight members, while the smallest ones consist of a single wafer each. Clustering goes across lots, as is shown in Figure 5, which shows the yields of the individual wafers, grouped by cluster. Whatever causes the process differences, therefore, is active in different lots, even though there is some tendency for wafers from the same lot to be in the same cluster.

Clustering groups wafers together that seem to suffer from similar defect mechanisms. Therefore, the fail probability y of a device is likely to be much more constant within one cluster than across all the lots, and, consequently, Equation (3.2) is likely to be a much better description of reality. In other words, we expect the yield distribution to adhere much more closely to our binomial expectations than in Figure 2. The yield distributions within each cluster are shown in Figure 4. Within each cluster, the yield distributions are obviously much more narrow than the overall yield distribution.

It is also of obvious interest to know what makes a cluster a cluster; in other words, what sort combinations define the various clusters. After all, the point of clustering is to group together those wafers that seem to have failed similarly, and differences between clusters, therefore, reflect essential differences in process history. The sort compositions of the twenty seven clusters are shown in Figure 6. Because the Sort_1 and Sort_2 sorts dominate the fallout, and because their contributions seem to be more or less stable between wafers, and, therefore, are not very indicative of cluster differences, the figure shows only the contributions of the other sorts. There are clearly large differences between the clusters, and they can now be used to further unravel the exact differences in process histories.

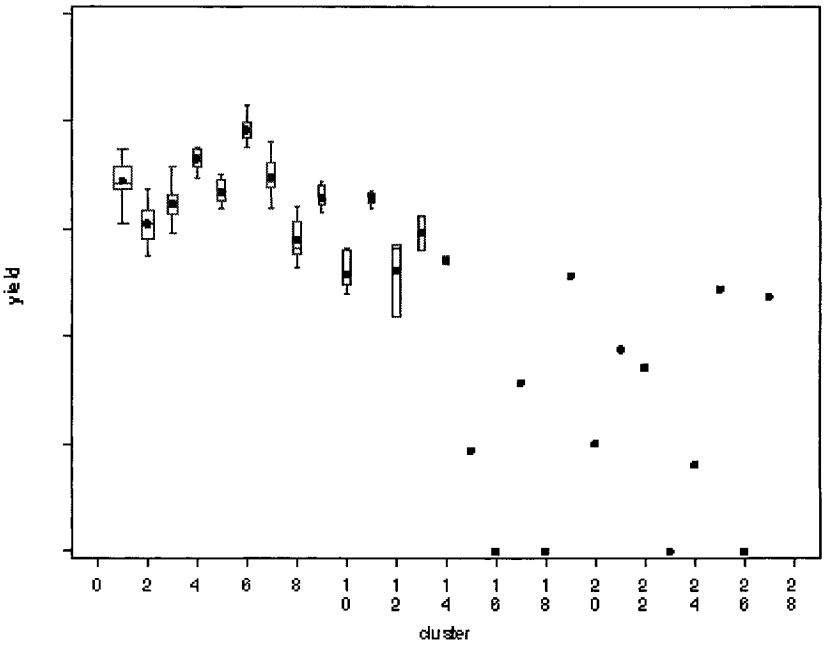


Figure 4 Yield distributions of wafer clusters

Such additional analyses will not be discussed here since they go beyond what can be done with tester fail data. Clustering has identified common sets of wafers. Actual manufacturing histories now need to be consulted to determine why and how different clusters differ.

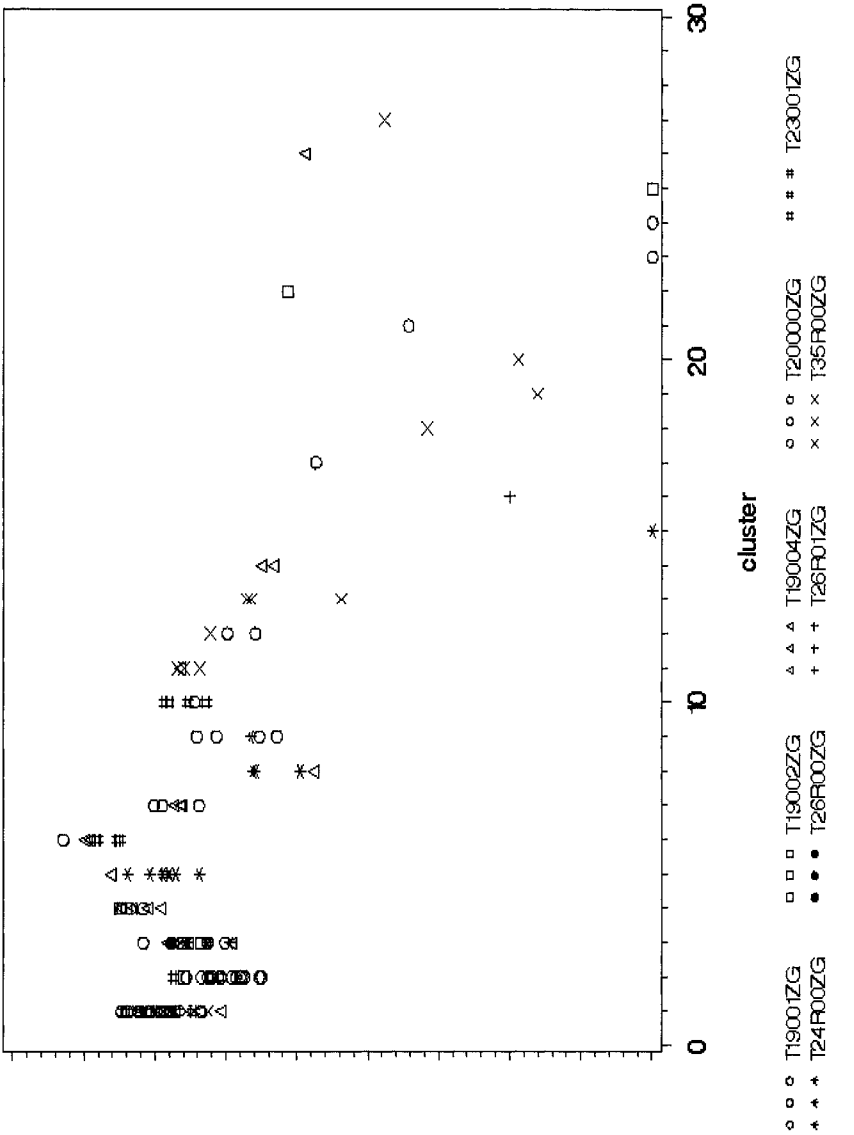


Figure 5 Individual wafer yields grouped by cluster

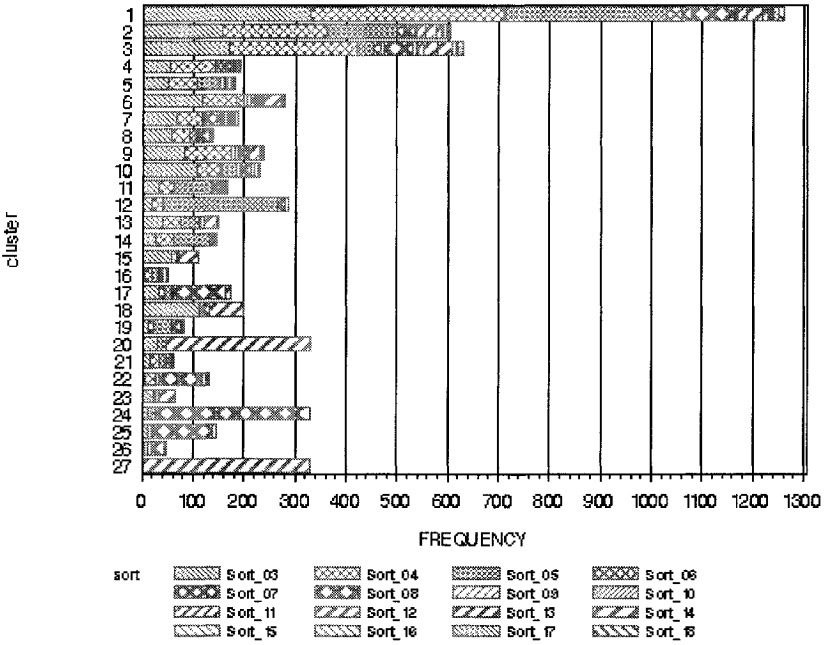


Figure 6 Sort compositions of the wafers clusters

Chapter 4

Area Dependence of the Yield

In Chapter 3, the observed yield Y and the real yield Y_0 were discussed. In this chapter, I will focus on the latter and on its area dependence. Understanding this area dependence is important for many reasons. First, it determines how large a chip can be fabricated, and, therefore, how much function can be put on a single chip. Putting additional function on a chip rather than on separate chips does away with the chip to chip delays and, therefore, improves the speed with which this function can be performed. It also obviates the need for having to package those additional chips and put them on the second level package, typically a board or a MCM.

Increasing the area of the chip, however, inevitably reduces its yield. Given a known (we hope) process quality, one might want to increase the size of chips until the yield drops below some economic threshold. To predict, however, at what chip size this threshold will be crossed requires a detailed understanding of how the yield depends on the process quality and the area.

The area dependence determines also how testable the chip can be made. Adding test features reduces the number of defective chips that are sent to the customer, but at the cost of lowering the yield and lowering the number of chips that can be put on a single wafer. Reducing the number of shipped defective chips reduces the cost associated with returned products. Increasing the area, however, increases the manufacturing cost per sold chip. At what point adding further test features becomes uneconomical depends of course on the various costs involved. One of the main components of the economic analysis, however, is, again, the area increase incurred by adding such features and its associated drop in the yield.

Similar economic considerations apply to adding redundant features to an integrated circuit [34, 51]. When adding redundant logic, other considerations than the yield become important, however. Typically, redundancy is added only when the probability of finding no defects is too low to be economically acceptable.

Second, it has been known for a long time [42] that the area dependence of the yield depends strongly on the average number of defects per chip and, to a lesser extent, on the variance. This is best described in term of P_n, the probability of finding exactly n defects on the chip. The overall shape of the set $\{P_n\}$ is captured succinctly by

$$\mu = \sum nP_n , \tag{4.1}$$

the expected number of defects on the chip, and the variance

$$\sigma^2(n) = \sum (n-\mu)^2 P_n \tag{4.2}$$

of this number.

When the defects are uniformly distributed, the number of defects on a chip has the Poisson distribution, $\sigma^2(n)$ equals μ, and the logarithm of the yield is a linear function of the area with slope $-\mu$. In reality, the logarithm of Y_0 is not a linear function of the area, but curves slightly upwards. This deviation from a straight line behavior can be attributed to the distribution of the defects not being uniform.

Analyzing the area dependence of the yield can, therefore, give us important information about $\{P_n\}$. It will allow us to estimate, for example, μ and $\sigma^2(n)$. Fallout during testing is caused by defects, and it can, therefore, give information about $\{P_n\}$ as well. A generally valid relation between fallout and defect coverage will be obtained in Chapter 8.1.2. Using that approach to obtain information about $\{P_n\}$ requires, however, that one knows the defect coverage, and is suitable only when the test sequence covers most of the defects of interest. In fact, the estimates based on area dependence could be used to predict the cumulative fallout, given a known defect coverage.

Many models have been proposed in the past (see, for example, [42, 55]) to reproduce the observed area dependence. One group of models assumes that the defects are uniformly distributed over the chip, but with a density that may vary from chip to chip. The resulting defect distributions are generically known as compounded Poisson distributions discussed in Chapter 2.1.2.

All compounded Poisson models have several drawbacks, however. First of all, their physical background is rather obscure. Their only justification is that they seem to reproduce qualitatively the observed area dependence of the yield, and that defect densities do vary from wafer to wafer, and even from chip to chip on the same wafer. This justification obviously does not explain the physical cause of any particular compounding function: some merely work better than others. A physical explanation of the negative binomial model to describe wafer to wafer variations has been given in [54], but has not been extended to chip to chip variations (see also [23]).

Another problem with compounded Poisson distributions is that they depend on a very small number of parameters that are independent of the area and occur in a fixed relationship. The negative binomial model, for example, has only two parameters, which makes it, therefore, somewhat rigid. In fact, to explain the area dependence quantitatively, one has to assume that at least

one of these parameters has an area dependence of its own: an area dependence that is merely observed and not explained.

An alternative approach is to start from some assumed local distribution of defects and then to deduce the consequences of this distribution for the yield [27, 24, 39]. This approach is able to handle localized clusters of defects, even clusters whose size is comparable to that of the chip, and that may partially overlap the chip. The most developed one of these models is the center-satellite model [51, 39]. The disadvantage of these models are the resulting mathematical complexities. They can in principle describe the whole range of cluster sizes, from negligibly small to very large, but only limited results have been obtained so far.

In this chapter, I will analyze the area dependence of the yield, starting from *as general a defect distribution as possible.* The main questions to be answered are, first, what is the overall shape of $\ln y_0$ when plotted as a function of the area, and, second, how are μ and $\sigma^2(n)$ related to the quantitative aspects of this shape ?

These questions will be answered independently of the details of the distribution of defects. The basic physical starting point of the analysis is that the defect producers, whatever they are, may have a wide variety of spatial characteristics and strengths, but the defects they produce almost never interact physically. This observation will form the basis of the approach followed in this chapter: *defects may be correlated, but only if they have the same cause. Once produced, they can be treated independently of all other defects.*

In the first section, a general model will be described of how defects are distributed between chips and inside a chip. This model is based on the lack of physical interaction between chips. Despite the generality of the model, fairly detailed results can be derived that connect the behavior of the yield as a function of the area of the chip with the moments of the defect distribution. The main conclusions reached in this section is that, under very general assumptions, the logarithm of the yield divided by the area is an non-decreasing function of the area.

To get a better feel for how clustering can affect the yield, a restricted version of this model will be analyzed in detail in the second section. This specialization is in fact a simplified center-satellite model. Some additional approximations are made that do not affect the basic physics of the defect distribution, but that do make it possible to derive closed form expressions for the yield and for the first two moments of the defect distribution. Numerical evaluation of these expressions then gives a quantitative picture of the behavior of the yield. This model also allows the explicit calculation of the area dependence of μ and $\sigma^2(n)$, and of the cluster coefficient.

The main problem in applying the general results to real designs is how to get yield data at different areas. Barring the production of special test chips, only yield data on actual production chips are available. Fortunately, this includes wafer maps that describe which chips on each wafer were free of defects and which were not. These wafer maps can be used to simulated chips of different size. The method employed is that of quadrats [53, 17, 12]. In the final section, an systematic formulation of the quadrat method will be presented. This method will be applied to a large data set of over a thousand wafers.

1 GENERAL MODEL

Defects can be produced in a large variety of ways, and different defect production mechanisms will have very different spatial characteristics and strengths. The defects that they produce, however, almost never interact physically. They may be correlated, but only if they have the same cause. Once produced, they can be treated independently of all other defects.

In this section, a general model will be developed of how defects are distributed between chips and inside a chip that is based on the approximation that the defects themselves do not interact. The main purpose of this model is to allow the derivation of general expressions for the yield and for the first few moments of $\{P_n\}$. Despite the generality of the model, fairly detailed results can be derived that connect the behavior of the yield as a function of the area of the chip with the moments of the defect distribution.

1.1 Primitive Polluters

The defect production mechanisms that affect integrated circuits are approximated here by more simple defect producers that, for lack of a better name, will be called primitive polluters. Each primitive polluter is characterized by a strength, and by a region on the chip that it affects. The strength of a primitive polluter is a measure of how many defects it produces on average in its associated region. Regions can have arbitrary shapes and sizes. The regions of different primitive polluters can overlap and one can even be completely included in another.

The defects produced by a specific primitive polluter are randomly distributed over its associated area. Consequently, if the size of this area is C, and the strength of the primitive polluter is ν, then the number of defects it produces has the Poisson distribution [15] with mean value νC. The assumption that defects are produced randomly is admittedly an over-simplification: mis-

registration of a mask for example will introduce all kinds of defects on the chip that are very definitely not random. We don't expect such defects to occur in a stable production environment, however, and we can, therefore, safely ignore such highly non-random defects.

Different primitive polluters can be arbitrarily correlated, in the sense that both their strengths and their regions can be correlated. All defects, however, whether they are produced by different primitive polluters or by the same one, are independent. In other words, all apparent correlations between defects are assumed to be caused by correlations between the primitive polluters, and not by physical interactions between the actual defects. This is an idealization, as the primitive polluters can produce arbitrarily shaped defects, including defects that have a non-negligible size. When the strength is low, this idealization is justified, but of course, when the strength increases, deviations may occur.

Note that the model does not exclude any correlation between the primitive polluters, and, therefore, that more complex defect production mechanisms can be approximated by a number of strongly correlated primitive polluters. For example, some defect producers affect a more or less circular area, and introduce defects randomly within this area with an intensity that depends on the radial distance from the center of the circle. A sophisticated approach to this type of defect producers is the center-satellite model [39], but that model is not easily extended to arbitrarily shaped regions. In the approach used here, this defect producer can be approximated by a number of primitive polluters with ring-shaped regions that are concentric with the circle, but whose strengths diminish with the distance from the center of the circle. Within each concentric ring, defects are produced randomly with constant strength. By adjusting the widths of the bands, the approximation can be made arbitrarily accurate.

Next we assume that each defect, whatever its shape, has a definite location; in other words we can assign to the defect a point on the chip that represents its location. I will generally indicate such a point by $\vec{r}$, where the arrow over r indicates that it represents a location in two dimensions. When the defect has no spatial extent, $\vec{r}$ coincides with the physical location of the defect. Even when the defect has a spatial extent, however, it is often possible to describe a defect by a point location plus some other information. For example, a circular spot defect is described by a center and a radius.

1.2 Yield and Moments

For a specific chip, we can, at least in principle, list the defect producers that affected it. This list can then be transformed into a list of approximating primitive polluters. Of course, different chips have different lists of primitive polluters associated with them. Each small area on this given chip is affected by a number of primitive polluters, each of which randomly produces defects with a strength that depends on the specific primitive polluter. The number of defects produced by a given primitive polluter i with strength ν_i in the small area $\vec{dr}$ has the Poisson distribution with mean $\nu_i \vec{dr}$.

As the defects introduced by different primitive polluters are independent, even though their strengths and associated areas may not be, the number of all defects introduced in the area $\vec{dr}$, therefore, also has the Poisson distribution, but now with mean

$$\nu(\vec{r})\vec{dr} = \sum_i \nu_i \vec{dr}. \tag{4.3}$$

In this equation, the sum is over all the primitive polluters that affect the area $\vec{dr}$.

Finally, the number of defects on the chip has the Poisson distribution, with mean $\int \nu(\vec{r})\vec{dr}$, with the sum over all the areas of the chip replaced by an integral. Consequently, the probability that the chip is free of defects equals

$$e^{-\int \nu(\vec{r})\vec{dr}}. \tag{4.4}$$

As mentioned above, different chips will have different sets of primitive polluters affecting it. To obtain the probability that a randomly chosen chip is free of defects, the average over all configurations of primitive polluters has to be taken. In general, however, even the actual configuration of primitive polluters for a single chip is unknown. Averaging over different configurations requires, in addition, a knowledge of the distributions of the areas and strengths of the primitive polluters, and a knowledge of their statistical dependencies. Fortunately, we will not actually have to do the averaging in the general case. We will assume, however, that, although unknown, it is well defined.

The configuration average of any function will be indicated by <f>. The expected yield y_0 of an integrated circuit is the probability that a randomly

chosen manufactured chip is free of defects. It depends on the design and the production technique of the product. Its value depends, among other things, on the area A of the chip, and is obtained by taking the configuration average of the probability that a randomly chosen chip is free of defects

$$y_0(A) = \langle e^{-\int \nu(\vec{r})d\vec{r}} \rangle . \tag{4.5}$$

The consequences of this equation will form the central part of this section. It has been proposed previously [27], and some of its properties are well known. It has to be stressed, however, that it has been derived here making only very weak assumptions: *defects are produced by a large variety of mechanisms, but are independent, even though the mechanisms themselves may not be.*

The moments of $\{P_n\}$ are most easily obtained from the generating function of this distribution. This generating function is derived in Appendix B. The expected value μ of the number of defects on a chip is found to be $\langle \int \nu(\vec{r})d\vec{r} \rangle$, and the variance of n equals

$$\sum_n n^2 P_n - \mu^2 = \mu + \langle \int_A \left(\nu(\vec{r}) - \frac{\mu}{A} \right)^2 \rangle , \tag{4.6}$$

where the second term on the right can be interpreted as $\sigma^2(\nu)$, the variance of $\nu(\vec{r})$ (compare Chapter 2.1.2).

When $\nu(\vec{r})$ is constant, say ν_0, μ equals $\nu_0 A$ and $\sigma^2(n)$ equals μ. The difference $\sigma^2(n)$ - μ is, therefore, a measure of the degree of non-uniformity of $\nu(\vec{r})$. In the literature, the cluster coefficient α (Equation (2.11)) is often used to gauge the degree of non-uniformity of $\nu(\vec{r})$. In the present context, it equals $\mu^2/\sigma^2(\nu)$. For the same μ, the smaller α the wider the distribution of the number of defects on the chip, and the more common large values of n. α is clearly always positive in this model.

1.3 Examples

The simplest case is the Poisson model, in which $\nu(\vec{r})$ is constant for all $\vec{r}$ and all chips. The next simplest case is that of $\nu(\vec{r})$ constant over a chip, but not necessarily the same for all chips. This occurs when the area of the chip is small compared with the scale over which $\nu(\vec{r})$ changes, and will be called the small area approximation. This case is of particular interest, because now the configuration average is simply the average over different defect densities. The defect strength on a randomly chosen chip is a random variable with some probability density function, and we can rewrite Equation (4.5) as

$$y_0(A) = \int e^{-\nu A} f(\nu) d\nu, \tag{4.7}$$

which is the well studied compound distribution model, discussed in Chapter 2.1.2.

1.4 General Properties

Experimentally, it is known that the real yield exceeds the Poisson one, and that the excess increases with increasing chip area (even though the yield itself decreases with area.) That this is not a coincidence is shown in Appendix B. The main conclusions reached in that Appendix are illustrated in Figure 7.

This figure shows the yield as a function of area. The yield is plotted on a logarithmic scale. The dashed line represents the yield in the case of the Poisson model with μ equal to 0.3. Because of the logarithmic scale, it is a straight line with slope $-\mu$. The full line represents a general yield, although for the purpose of generating the figure, a negative binomial yield was taken with νA equal to 0.3 and α equal to 2.0. At very small values of the area A, it approaches the yield of a Poisson distribution with the same value of νA, but is everywhere larger than this Poisson yield, with the excess increasing with area.

The dotted line in the figure is the tangent to the general yield curve, taken at an area of 2.0, where the yield equals 0.625. The yield as well as the logarithm of the yield obviously always decrease with area. In addition, however, the logarithm of the negative binomial yield is a convex function of the area, which means that any tangent to it, taken at any area, never exceeds it. It is not clear that a general yield, using some arbitrary distribution of primitive polluters, will also have these properties: that its logarithm is a convex function of

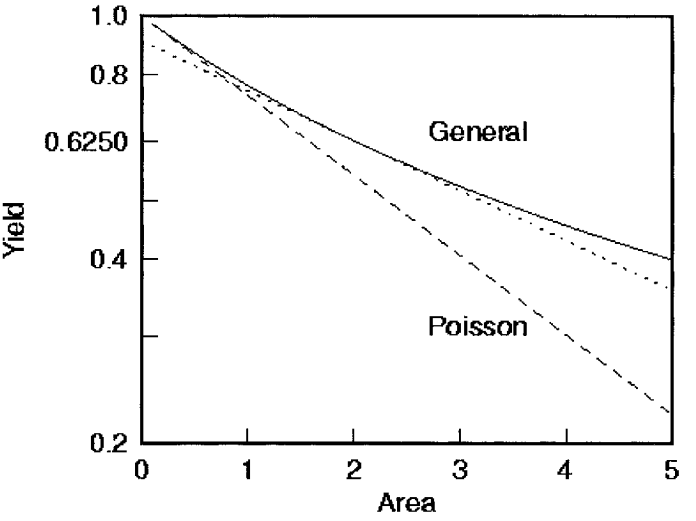


Figure 7 Logarithm of the yield as a function of area

the area and that for very small areas it will become a linear function of the area.

As is shown in Appendix B, the logarithm of the yield becomes a linear function of the area in the limit of very small areas, and is always negative and increasing at any area. Pictorially, a line tangent to $\ln Y_0$ always slopes downwards, for all yields and at all areas, but becomes more horizontal at larger areas. *The logarithm of the yield, therefore, is a convex function of the area, no matter what the distribution of primitive polluters.*

This convexity property is of great importance for a powerful way of analyzing yield data. To show this, let us write f(A) for $\ln y_0(A)$, and let us indicate derivatives with respect to A by a '. f(A) is negative, except when A vanishes, for then f(A) goes to zero. In addition, f'(A) is negative, and for very small A goes to the limit -νA.

Now, let us define

$$g(A) = A^{-1} \ln y_0(A). \tag{4.8}$$

Clearly, g(A) is negative and goes to -ν when A goes to zero. In fact, for the Poisson yield, g(A) is constant and equal to -ν for all values of A. Deviations from the Poisson approximation will show up by g(A) not being

constant, and in fact by having a non-zero slope when A is small. g(A) is a better tool for analyzing clustering than f(A), because, when there is clustering, f(A) is not a straight line and g(A) has a non-zero slope. Non-zero slopes are easier to spot, however, than non-zero curvatures, and, more importantly, slopes are easier to measure than curvatures.

Figure 8. shows g(A) for the two yields discussed above. The Poisson

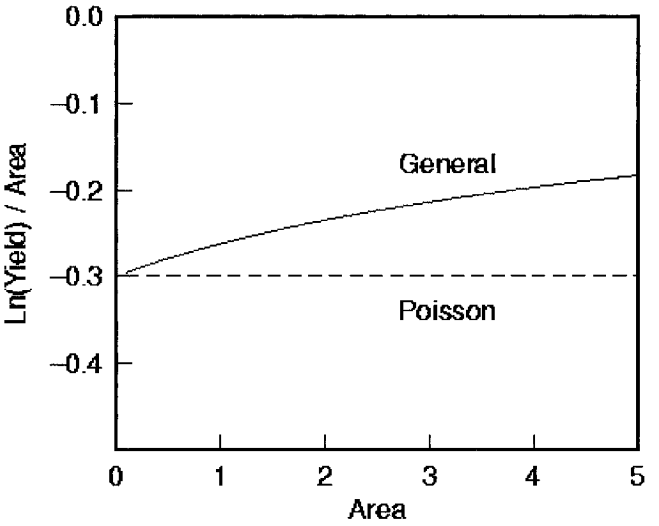


Figure 8 Logarithm of the yield divided by the area

yield is indicated by the dashed line. The general yield, exemplified by the negative binomial yield in this case, is indicated by the full line. It increases (becomes less negative) with area. Because $y_0(A) = e^{g(A)A}$, this figure gives a visual proof of the well known fact that, for the same average number of defects per chip, clustering improves the yield.

That g(A) is always non-decreasing follows from the definitions and the convexity of f(A). Taking the derivative with respect to A, we find

$$g'(A) = \frac{1}{A}(f'(A) - g(A)). \tag{4.9}$$

As f(A) is linear in A for very small A, g(0) has some finite, negative value (-μ in the case of the Poisson yield.) When we draw a line L through f(A) at A equal to 0 and A equal to some area S, this line will be above f(A) at all areas between 0 and S. Its slope is equal to g(S), for f(0) equals 0. On the other

hand, this same line also cuts the tangent to f(A) at A equal to S. Because of the convexity of f(A), the tangent is below the line L for areas smaller than S and above L for areas larger than S, showing that the slope of L is more negative than that of the tangent. Consequently, g(S) is more negative than f'(S). As S was arbitrary, g(A) is, therefore, more negative hat f'(A) for all A, and g'(A) is always positive.

As g(A) has a non-zero slope in the presence of clustering, this slope can be used to estimate the degree of clustering. The relationship between this slope and the degree of clustering is obtained by expanding the exponential Equation (4.5) in powers of $\int \nu(\vec{r}) n d\vec{r}$. This gives

$$y_0(A) = 1 - \bar{\nu}A + \frac{1}{2}\left\langle \left(\int \nu(\vec{r}) d\vec{r} \right)^2 \right\rangle + \ldots , \tag{4.10}$$

in which $\bar{\nu}$ is the mean value of or ν. Taking the logarithm and simplifying,

$$\ln y_0(A) = -\bar{\nu}A + \frac{1}{2}(\sigma^2(n) - \mu) + \ldots , \tag{4.11}$$

and

$$g(A) = -\bar{\nu} + \frac{1}{2}\bar{\nu}^2 \frac{1}{\alpha} A . \tag{4.12}$$

$\bar{\nu}$ is obtained from the intercept of g(A) with the A = 0 axis, and α is obtained from the slope of g(A) and $\bar{\nu}$.

2 CENTER-SATELLITE MODEL

In the previous chapter, an example was given of a non-uniform defect distribution. The non-uniformity was rather restricted though, in the sense that the defect density could vary between chips but not over the area of a single chip. The opposite of this example would be one in which the clusters were so small that the probability that a cluster only partially overlapped a chip could be ignored. In such a case, however, the clusters can just as well be treated as single defects, with a small adjustment due to the fact that the cluster may not produce a defect at all. At both ends of the size spectrum, therefore, we expect an approximately Poisson like behavior, although with very different strengths.

The transition from very large clusters to very small clusters is much more difficult. It is important though to understand this transition, because many defect producing mechanisms give rise to clusters that have a spatial extent that is comparable to the area of a single chip. Quantities of interest are of course the yield, but also the first few moments of $\{P_n\}$. In particular, the expected value and the variance, or, equivalently, the cluster coefficient of this distribution are of importance. They, after all, determine how many defects can be expected to occur on a chip, and roughly the range of this number.

To see the effects of defect distributions that do vary over the area of the chip, a more complex model will have to be considered than the simple compounded one. A very convenient one for studying the effects of clustering is the center-satellite model.

In this model there is a uniform background of defects with strength ν_0. Superimposed on this uniform background is a uniform distribution of circular defect clusters. The strength of this distribution is λ, meaning that there are on average λ cluster centers per unit area. The radii of these clusters is the same for all clusters and equal to ρ. The area of the cluster, $\pi\rho^2$, will be indicated by C. The defects produced by a cluster are distributed uniformly within the area of the cluster, with distribution strength ν. Each cluster, therefore, produces on average $n_C = \nu C$ defects per chip. Each cluster is in fact a primitive polluter, as introduced in the previous chapter.

The effects of varying the chip size in comparison with the cluster size can now be studied easily by putting a chip somewhere in the plane and varying its size. To facilitate the calculations, I will approximate the shape of a chip by a circle with radius R. This will introduce a small error in the results, due to the difference in geometry between a real chip and a circle. The size of this error will be small though, and can be ignored compared with the simplifications that have been made in approximating the real defect distribution by a single primitive polluter and a uniform background. By varying R, we can then follow in detail what happens to the yield, the cluster coefficient, etc.

This model, with a square chip rather than a circular one, has been analyzed in some detail by Meyer and Pradhan [39]. The essentials of their analysis can be found in Appendix C.

2.1 Center-Satellite Yield

Using the results derived in the appendix, we find

$$y_0 = e^{-\lambda S(1-Q(0))-\nu_0 A}, \tag{4.13}$$

with

$$Q(0) = \frac{1}{S}\int_S e^{-n_C \alpha_A(\vec{r})} d\vec{r}, \tag{4.14}$$

and $\alpha_A(\vec{r})$ the probability that a defect produced by a cluster centered at $\vec{r}$ will fall within the area of the chip centered at the origin.

In the limit of a very small chip, we find

$$\ln y_0 \approx -\lambda A n_C - \nu_0 A. \tag{4.15}$$

In this limit, it looks, therefore, as if the defects produced by the clusters, λn_C per unit area, are smeared out uniformly, and are merely added to the already existing background.

In the limit of very large chips, we find

$$\ln y_0 \approx -\lambda A(1 - e^{-n_C}) - \nu_0 A. \tag{4.16}$$

This result shows that in the limit of very large chips, the size of the clusters can be ignored. Instead, a cluster acts as a single super-defect, the factor $1 - e^{-n_C}$ being the probability that this super-defect actually produces a fault.

For very large chips, therefore, there seems to be a uniform background of regular defects with strength ν_0, and another uniform background of super-defects with strength λ. This is rather different from the negative binomial yield, which is not linear in A at all for large areas. This difference between the center-satellite model and the negative binomial model becomes even more pronounced when considering $g(A) = A^{-1}\ln y_0(A)$. In the center-satellite model, g(A) becomes constant when A becomes very large, while it goes to zero as $A^{-1}\ln A$ in the negative binomial model.

2.2 Center-Satellite Moments

Moments of $\{P_n\}$ can be obtained from the generating function of $\{P_n\}$ [15]. This function is given in Appendix C. From it, we immediately obtain

$$\begin{aligned} \mu &= \lambda S n_C E(\alpha_A) + \nu_0 A \\ \sigma^2(n) &= \lambda S n_C^2 E(\alpha_A^2) + \mu \end{aligned}, \tag{4.17}$$

where $E(\alpha_A)$ is short for $S^{-1}\int_S \alpha_A(\vec{r})d\vec{r}$, and $E(\alpha_A^2)$ is defined analogously. As μ is obviously also equal to $\lambda n_C A + \nu_0 A$, we find

$$E(\alpha_A) = \frac{A}{S}. \tag{4.18}$$

The cluster coefficient α describes the degree of non-uniformity of the defect distribution. When ν_0 is zero, it equals

$$\alpha = \lambda S \frac{E(\alpha_A)^2}{E(\alpha_A^2)}. \tag{4.19}$$

$E(\alpha_A)$ increases smoothly from its small chip limit R^2/ρ^2 to its large chip limit 1. Likewise, $E(\alpha_A{}^2)$ varies smoothly from R^4/ρ^4 to 1. This shows that, when the chips are small, the cluster coefficient goes to the non-zero constant λC. When the chips are large, it is proportional to the area A of the chip, the proportionality constant being λ. It will, therefore, go to infinity when the chip size increases.

This does not mean, however, that the variance and expected value of n will become equal in the limit of very large chip areas, for the cluster coefficient becomes large too when the expected number of defects on the chip becomes large. This is obvious from equation 32. In fact, when the area of the chip becomes large, $\sigma^2(n) / \mu$ goes to $1 + n_C$.

2.3 Numerical results

The general behavior of the yield and the cluster coefficient can be obtained by straightforward numerical integrations. Specific calculations were done for a zero background density ($\nu_0 = 0$,) a cluster with unit area ($C = 1$,) and a total defect density of 0.1 per unit area ($\lambda n_C = 0.1$.) $g(A) = A^{-1}\ln y_0(A)$ is shown for several choices of μ in Figure 9. The general shape of g(A) is the same as shown in figure 1.2: all g(A) converge on the same con-

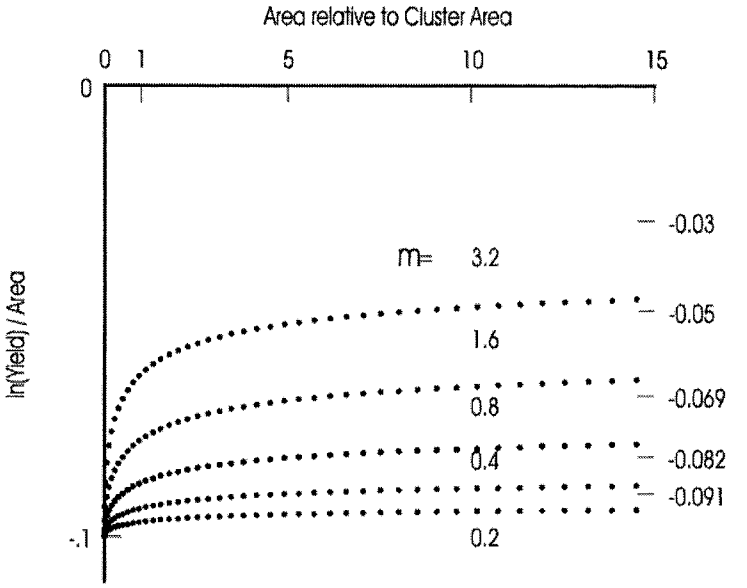


Figure 9 Logarithm of the yield divided by the area in the center-satellite model

stant $-\lambda n_C$ at small values of A, and they are all non-decreasing. Unlike the negative binomial result, however, g(A) does not go to zero in the center-satellite model when A becomes very large, but, instead, goes to the constant $-\lambda(1 - e^{-nC})$, indicated on the right of the figure. These limits are drawn at their proper positions on the vertical axis, and are, therefore, slightly above their corresponding g(A) curves. When A is large compared with C, g(A) depends only weakly on A, and using some Poisson approximation seems to be justified. As mentioned above, this is not strictly correct, as will become clear when we consider the cluster coefficient.

Near A = 0, the slopes of the g(A) shown in Figure 9 are inversely proportional to the cluster coefficients α. The g(A) curves, therefore, clearly indicate that, when the product $\nu\lambda C$ is constant, clustering increases when ν increases. This is also what one would expect intuitively, because λ must decrease when ν increases to keep the average number of defects constant. When we increase ν, therefore, we create fewer clusters, but more defects per cluster; in other words, stronger clustering.

This is also confirmed by calculating the cluster coefficients directly. They are shown in Figure 10 for the same set of ν values as in Figure 9. What is plotted are in fact the inverses γ of the cluster coefficients, so the large area

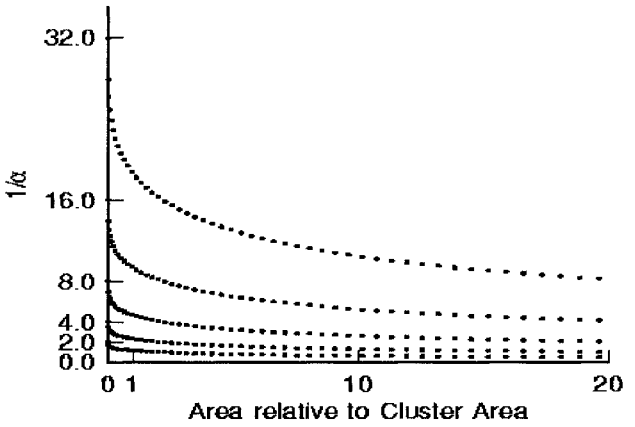


Figure 10 Inverse of the cluster coefficient

behavior of α can be followed as well. γ starts at the high value of λ^{-1} which in this case equals 10ν, and decreases monotonically to zero. At large values of the chip area A it in fact behaves as $1/\lambda A$. Even when A is large compared with C, however, γ is still not small compared with 1, indicating that the distribution of the number of defects per chip is far from Poisson, even though the yield may safely be approximated by the Poisson result.

In the literature, the possibility of local minima in the cluster coefficient have been discussed [55, 39, 57], but no such phenomenon is observed here. That that is true in general for the simplified center-satellite model is obvious from equation 34 and Figure 10. Equation (4.19) shows that α is equal to λ times a geometric factor that depends only on the sizes of the clusters and the chip. Consequently, each α curve shown in Figure 10 is representative for all possible α curves, and can be made equal to any one of them simply by redefining the unit of length and by multiplying it by the ratio of two λs. As none of the αs in Figure 10 shows a minimum, no α function will have a minimum for any combination of λ, ν, ρ or R. A more likely explanation for the observed minima is, therefore, normal statistical variations.

3 ESTIMATING THE AREA DEPENDENCE

In normal circumstances, the distribution of defect producers is partially known at best. The only way of estimating the area dependence of the yield is, therefore, to measure it experimentally. Some possible approaches are counting defects on blank wafers [17, 55], or putting specific test structures on the wafers and testing these structures for defects [13]. The drawback of both these methods is that they are sensitive to only a limited set of defects, in particular to the ones that occur naturally on wafers, and not necessarily to defects that are introduced during the fabrication of the actual chip. They have the additional disadvantage of requiring separate processing steps that are not part of normal chip production.

A different way to measure the distribution of defect producers is using the results of testing real chips, in particular the distribution of passes and fails on the wafer. This approach assumes that the quality of the test is high, and that we can trust that the chips that fail the test are bad, and that the chips that pass the test are good. In this section, techniques will be discussed for using the test results to estimate the area dependence of the yield.

I will follow the general practice [23, 64] of simplifying the analysis by dividing the defects into two classes. One class consists of gross defects that are so pervasive, and so detrimental to the operation of the chip that not only will they make a chip fail, but they will make every portion of the chip fail as well. The defects in this class are assumed to contaminate some portion of the wafer, and to kill every chip in the affected region regardless of the size of the chip. Consequently, the yield due to these defects is area independent. It will be indicated by y_S. The second class consists of defects whose distribution, if not Poisson, is still relatively simple.

In other words, the yield is modeled as the product $y_S y(A)$, in which A is the area of the chip and y_0 is the gross yield; that is, the yield due to the gross defects. All the area dependence of the yield is concentrated in the factor $y(A)$, which, in this chapter, will be taken to have the negative binomial form. It could easily have some other form, but the negative binomial one is the usual choice. The goal of the spatial clustering analysis is then to estimate the three parameters y_S, α (or $\gamma = 1/\alpha$)and ν (or $\mu = \nu A$)[1].

The area independence of y_S followed from the assumed severity of the gross defects: they kill every part of the chip. The converse is true too: an area independent yield factor implies that there are gross defects that make every portion of the chip inoperative. This can be seen by analyzing what it means

1. And I suspect that part of the reason for including the area independent factor is to increase the number of free parameters from two to three, and, thereby, increase the accuracy of the fit.

for a yield factor to be independent of area. Chips that are killed by whatever gives rise to y_S are colloquially called wipe outs.

Assume that there are N chips on a wafer, and that a fraction $1 - y_S$ of the chips are wipe outs. This fraction covers an area on the wafer equal to $(1 - y_S)NA$. Next, consider the same wafer, but much smaller chips. In fact, assume that there are kN chips on the wafer, each with area A/k, obtained by dividing each original chip into k smaller ones. If the same fraction of chips are wipe outs, the smaller chips cover an area on the wafer equal to

$$(1 - y_S)kN(A/k) = (1 - y_S)NA. \tag{4.20}$$

Consequently, the two regions containing the wipe outs must be the same area on the wafer, for they are on the same wafer and are equal in size. We conclude then that each small chip in a large wipe out is a wipe out, and no small chip is a wipe out if it is not in a large wipe out. Another way of saying the same is that each part of a subdivision of a wipe out is still a wipe out.

This addition of the concept of a gross, area independent yield is a simplification, and is defensible only when there are manifestly multiple defect sources: one that is more or less random, with, perhaps, some spatial clustering, and another one that is systematic, in the sense that it affects some region of the wafer, and, in that region, kills all the chips. A better way of handling any obvious mixtures of multiple defect mechanisms, however, is first to identify wafers with non-random fail patterns (see Chapter 7), and then to use spatial clustering methods only on wafers that are not so identified.

3.1 Comparing different products

There are at least two ways for using the test results to estimate the area dependence of the yield. Both techniques are simple to implement, but suffer from the disadvantage that yields are compared for chips of very different sizes. This is a problem, for, as we have seen in the previous section, the degree of clustering is relative to the size of the chips. Using different sizes means that we attempt to get one measure of clustering by looking at the defect distribution at different levels of granularity.

The first technique is to merge yield data for different integrated circuits of roughly the same complexity but of different areas [17]. These merged data can then provide a composite yield versus area curve. This approach suffers from having to define, and measure, complexity, and from having to compare areas for different integrated circuits. The latter is more difficult than it may

seem, because not all regions on the chip are equally used and allowances have to be made for heavily used regions and sparsely used ones [25].

This technique was applied to a CMOS2 data set [43]. The results are presented in Figure 11. The yield data were obtained for CMOS2 chips, ranging

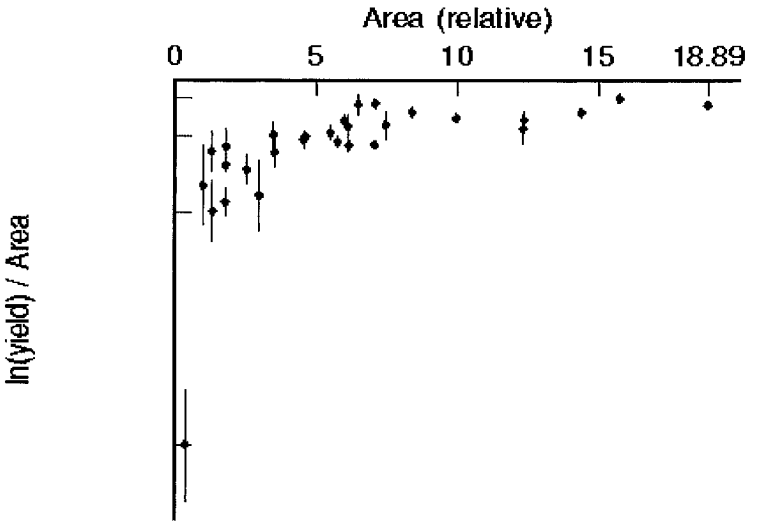


Figure 11 g(A) for CMOS2 chips

in size by more than a factor 20. The yield for all these chips was very high; for almost all of them over 90%. The test coverage was typically 99% or higher, and the difference between the observed yield and the real one was ignored. The areas were estimated using cell counts. They were divided by that of one of the chips to get relative areas.

The figure shows both the actually observed $\ln(y_0(A))/A$ values and their three sigma error ranges. The latter were obtained by estimating the standard deviation of a yield Y as $\sqrt{(Y(1-Y))/N}$, where N is the number of chips involved, and then adding and subtracting three times the resulting standard deviation to or from Y.

The data point with g(A) equal to -0.067 seems to be anomalous. Ignoring this data point, the results show a weak trend to become less negative when the area increases, indicating a certain amount of clustering. If we believe the data as they are, then g(A) goes up and down and up again at larger areas. As this is not possible, these data can clearly not be trusted. The obvious weak

point is the area estimation. When we cannot trust the area estimates, however, then clearly any estimation of either μ or α cannot be trusted either.

3.2 Quadrat method

The second technique for estimating the area dependence of the yield relies on the quadrat method [53, 12]. In this method, the wafer is divided into sections of two by two chips, and into sections of two by one chips. Larger sections, for example of two by three or three by three chips, can also be considered, but are not necessary. Despite its name, therefore, a quadrat need not consist of four chips; the name is used for all section sizes. By considering each two by two section as a chip of size 4A, and each two by one section as one of size 2A, yield data can be obtained for areas A, 2A and 4A. This technique bypasses the problem of having to define complexity, or having to measure area in a consistent fashion.

The data used in the calculations are subject to normal statistical fluctuations. A necessary part of the analysis is, therefore, the estimation of the expected size of these fluctuations, and of their effect on our estimates of the quantities of interest, like the cluster coefficient and the average number of defects per chip. It is straightforward to enhance the quadrat analysis such that these variations can be obtained from the same yield data that were used to estimate the distribution parameters in the first place. Before the enhanced quadrat analysis can be described, however, some difficulties with the technique in general have to be mentioned.

3.2.1 Problems with the quadrat method

The main problem with the quadrat method is the choice of the quadrats. The assumption underlying all calculations based on this method is that the quadrats that are constructed on the wafers form a sample of randomly and independently selected quadrats from the space of all possible quadrats. There are a number of problems with this assumption.

First, the quadrats are not randomly placed on the wafer surface, but their locations and orientations are dictated by the grid of chips on the wafer. Second, when there is clustering, quadrats on the wafer surface, in particular, when they are right next to each other, or even overlap each other, can hardly be considered to be independent, as the clustering impresses a landscape of varying defect densities on the wafer that crosses chip and quadrat boundaries. After all, fail probabilities of neighboring chips are independent only when the defect distribution is Poisson.

The assumption of an independent sample may introduce severe statistical errors. The effects of such statistical errors can be gauged using bootstrapping, however.

The third problem with the quadrat method is more procedural than fundamental. Given a wafer surface, there may be several ways to define quadrats, but no clear way to define a best choice. For example, when two by one quadrats are desired, we could choose horizontal ones, vertical ones, or some mixture of both. Unfortunately, different selections of quadrats may lead to different results.

One possible way to proceed is to ask for the average over all possible ways to select quadrats. If we look at all those ways as a form of bootstrapping, the average values of the parameters can be approximated by choosing all possible quadrats. We might still be interested in knowing how different choices would affect the results, but that variability in the parameter estimates can be estimated using standard bootstrapping techniques (see Chapter 2.3).

Finally, the quadrat method is inherently restricted to clustering whose spatial scale is large compared to the dimensions of the largest quadrat, for otherwise the parameters of the distribution would vary between quadrats of different sizes, and, consequently, could not meaningfully be determined using this method.

3.2.2 Numerical technique

Despite all the problems with the quadrat method, it is a popular way to estimate the degree of clustering, and we now turn to the actual calculations.

Once the quadrats of various sizes have been selected, the parameters of the negative binomial distribution can be estimated. The standard estimation technique uses least squares regression of the yields on the areas (for example, A, 2A and 4A). As the yields are non-linear functions of the areas, this leads to a set of non-linear equations that have to be solved numerically.

This approach suffers from a minor methodological problem, namely that, when quadrats of different sizes are employed, the smaller quadrats could be selected from the corners and edges of the wafer where the large quadrats would not fit. That problem could be solved by first selecting the largest quadrats, and then selecting the smaller ones using the chips in those largest quadrats. Once that restriction is made, however, there is little reason to pretend to use smaller quadrats, because all possible information should already be contained in the largest ones.

In this chapter, I will use a different approach. The mathematical details of the calculations are described in Appendix D. The calculations differ somewhat from the usual ones, since only quadrats of size four are used. Given

quadrats of any size, say k, the relevant statistics are the numbers $N_i^{(k)}$ of quadrats of that size having i passing chips, with i running from 0 to k. For any k, there are only k independent such numbers, for the sum over all the numbers should be the total number of quadrats.

The distribution of these statistics is multinomial, and the probabilities of them having particular values are functions of the parameters of the underlying defect distribution. The parameters can, therefore, be obtained from these statistics using maximum likelihood. That method leads to very complex equations, however, and it is doubtful that the benefit of getting accurate estimates is worth the cost of handling these complexities when the validity of the quadrat method itself is already in question.

If need be, the parameters can be obtained numerically by maximizing the likelihood function. In the remainder of this section, a simplified approach will be developed that leads to much simpler equations without deviating too much from statistical correctness

When we divide $N_i^{(k)}$ by the total number of quadrats, we obtain a ratio $P_i^{(k)}$. The latter's expectation value is also of interest, and will be indicated by $p_i^{(k)}$. As is shown in Appendix D, this expectation value can be written as a linear combination of the expectation values $p_i^{(i)}$, with i going from 1 to k. Conversely, $p_i^{(i)}$ can be written as a linear combination of $p_i^{(k)}$, with i not exceeding k,.

$p_i^{(i)}$ is also the expectation value of the yield of a quadrat of size i. If we now use those same linear equations to obtain ratios $P_i^{(i)}$ from the $P_i^{(k)}$, we obtain quantities that are analogous to the observed yields obtained for quadrats of size i. I will refer to them as pseudo yields.

The calculations described in Appendix D use the ratios for k = 4 to estimate the distribution parameters. The distribution is assumed to be negative binomial, and only three parameters will have to be estimated: a gross yield y_S, the cluster parameter α, and the defect density μ. The calculations are simplified in the sense that only three statistics are used, instead of the available four. The advantage of that simplification is that no regression needs to be done, for the three distribution parameters are uniquely determined by the three chosen statistics. Consequently, the calculations are considerably simpler than either regression or maximum likelihood.

The disadvantage is that not all available information is used, which will lead to a loss of accuracy. A further disadvantage is that different choices of the three statistics will lead to different estimates.

The calculations stay close to the standard approach by using for the three statistics the three pseudo yields $P_1^{(1)}$, $P_2^{(2)}$ and $P_4^{(4)}$. The resulting equations still involve one non-linear one, which, however, depends on only one variable and can be solved using standard numerical techniques.

3.2.3 Numerical results

This approach was applied to a medium sized microprocessor. There are 580 devices per wafer and 1339 wafers were included in the analysis. The pass/fail status for each chip on each wafer was determined, and γ, μ and y_S were determined as described above. Wafers for which $P_2^{(2)}$ turned out to be 0 were removed, for reasons explained in Appendix D, after which 1336 wafers were left. In addition, three more wafers were removed because they gave problems with bootstrapping (to be explained later). The final sample, therefore, consisted of 1333 wafers.

There will be inevitable statistical fluctuations, and it may be difficult to distinguish those from true non-Poisson behavior. To judge the size of these fluctuations, a preliminary experiment was performed in which the same wafer layout was used, with a yield of 65%, and with a strictly random distribution of the passes and fails. The resulting distribution of γ and μ values is shown in Figure 12. γ is centered around 0 with a spread of $(\pm 0.2)w$, and μ is centered on 0.43.

Note that γ can be as easily negative as positive, or, in other words, that the cluster coefficient can be as easily negative as positive. This is purely the effect of the finite sample size, as is, in fact, any γ not equal to 0, and does not indicate a violation of the assumptions made in Section 1. Both γ and μ are centered on their expected values: γ on 0.0 and μ on 0.43, the natural logarithm of 0.65.

The figure does not show the y_S values. In the experiment, y_S was distributed more or less symmetrically around 1.0 with a standard deviation of 0.13. This is unavoidable. Within the scope of the simplified calculations, there is no natural way to force y_S not to exceed 1.0. If we would use maximum likelihood, we could restrict all parameters to whatever ranges would be appropriate, and, if necessary, look for a maximum on the boundaries of those ranges. Because we have chosen not to use that method, however, this route is not open to us.

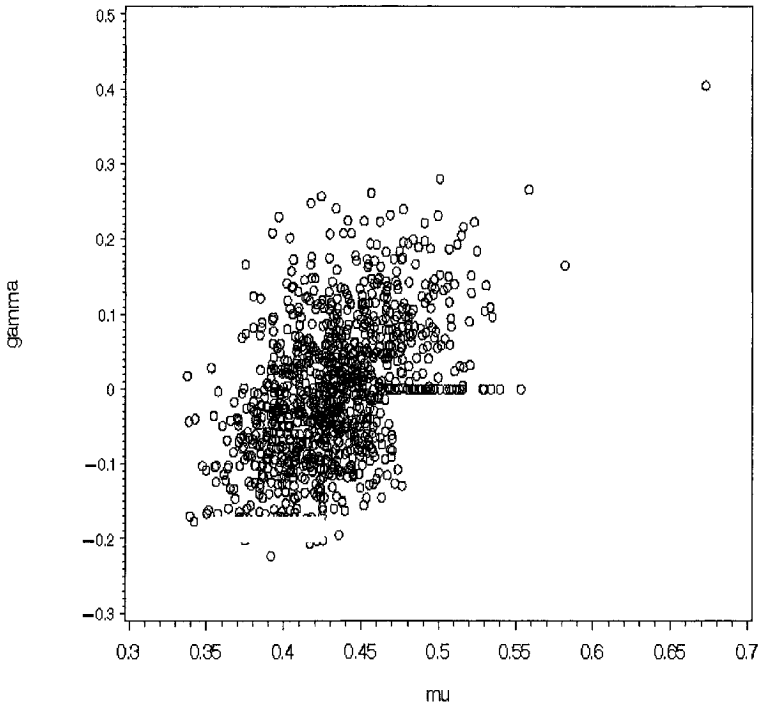


Figure 12 μ and γ for random defects

The same yield parameters were calculated for the 1398 microprocessor wafers. The results for those wafers that had a yield between 60% and 70% will be discussed first.

The results for γ and μ are shown in Figure 13. μ is again in the 0.3 to 0.5 range, as in Figure 12. γ, however, has a far wider distribution than in the Poisson experiment. It is negative on many wafers, which may be the result of the finite wafer size. It can also be much larger than in the comparable random experiment, however, which can be explained only as a manifestation of true clustering. As the data show no clear transition from random to clustered behavior, a more detailed analysis will have to be done to separate true clustering from mere statistical fluctuations. This will be done in Chapter 7.

γ and μ results for all yields are shown in Figure 14. The wafers were classified according to whether $Y_4 = P_4^{(4)}$ was zero (quadratrc = y4zero) or not (quadratrc = regul4.) The arc of data points in the bottom of chart, between μ values of 1 and 2, corresponds to y4zero wafers, for which $\gamma\mu = -0.25$. The μ in Figure 14 are now much more varied, in accord with the large variations

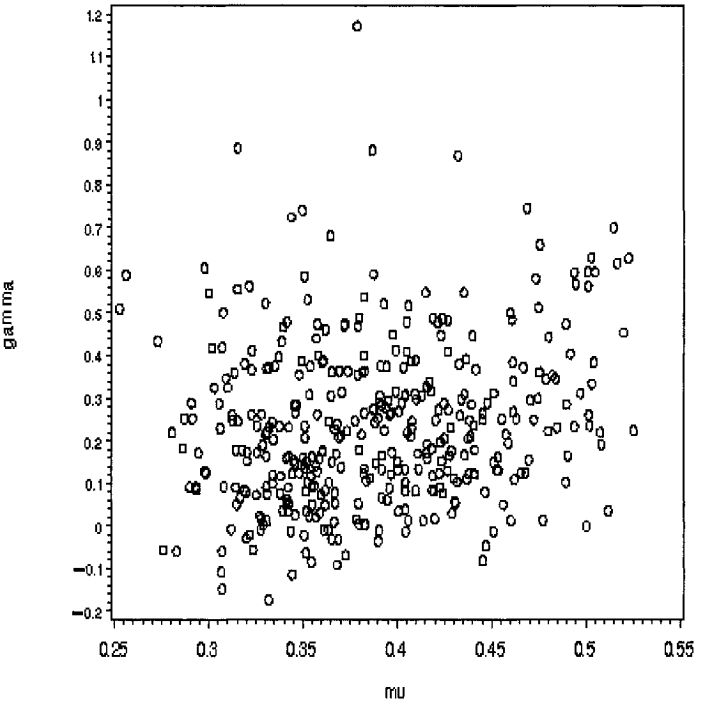


Figure 13 μ and γ for wafers with yield between 0.6 and 0.7

in yield that are seen in this sample of wafers. There is in fact good correlation between μ and yield, as shown in Figure 15. The range of γ values is a little bit larger than in Figure 13, but not by much, showing that clustering, if any, is more or less common at all yields.

Comparing the results from real wafers with simulated ones on which the defects are explicitly distributed with a Poisson distribution is one way to judge the size of statistical fluctuations. We could also attempt to estimate the variances of the estimates directly. This would be straightforward, although not necessarily easy, if we had used the maximum likelihood approach, for the set of $N_i^{(4)}$ has the multinomial distribution. As we are not using this method, however, the variances have to be determined differently. The distribution of the estimates is complex because of the non-linear equations that need to be solved, and determining them even approximately is not feasible. Consequently, the best we can do is using the bootstrap method.

The original wafer will be called the primary wafer, and the parameters, estimated with the techniques described above, will be referred to as the primary parameters. On each of the 1336 primary wafers, a bootstrap experiment

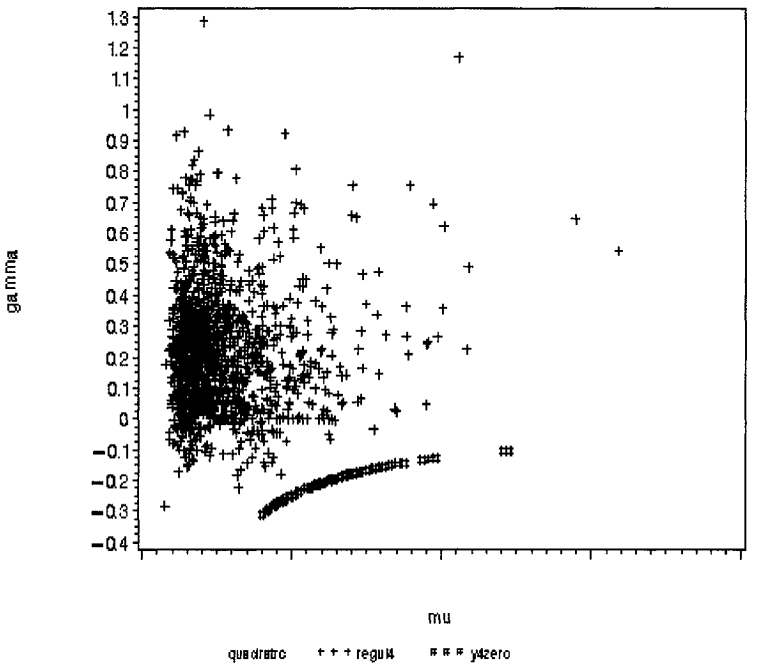


Figure 14 μ and γ for all wafers

was done in which 100 bootstrap samples were drawn from the set of available quadrats. The results from the 100 sample wafers were then combined to obtain a mean and a standard deviation for each primary wafer. The standard deviation is used as an estimate of the standard deviation of the primary parameter.

As pointed out in the appendix, several anomalies are possible when calculating negative binomial parameters from the data, be they primary or bootstrap. Wafers with $P_2^{(2)} = 0$ were already removed, but problems associated with the calculation of the gross yield y_S still have to be dealt with. Two possible anomalies were pointed out. y_S could be complex, or it could be large - larger than 2 in this experiment. No primary wafers were removed directly because of this anomaly, but several bootstrap samples were. Primary wafers are considered to be anomalous, however, if the number of bootstrap samples on which y_S is either complex or large exceeds 100. In this experiment, three primary wafers were found to be anomalous, and were removed from further consideration.

The results from the remaining 1333 wafers are shown in the next figures. The results for γ are shown in Figure 16. Each point in the figure correspond

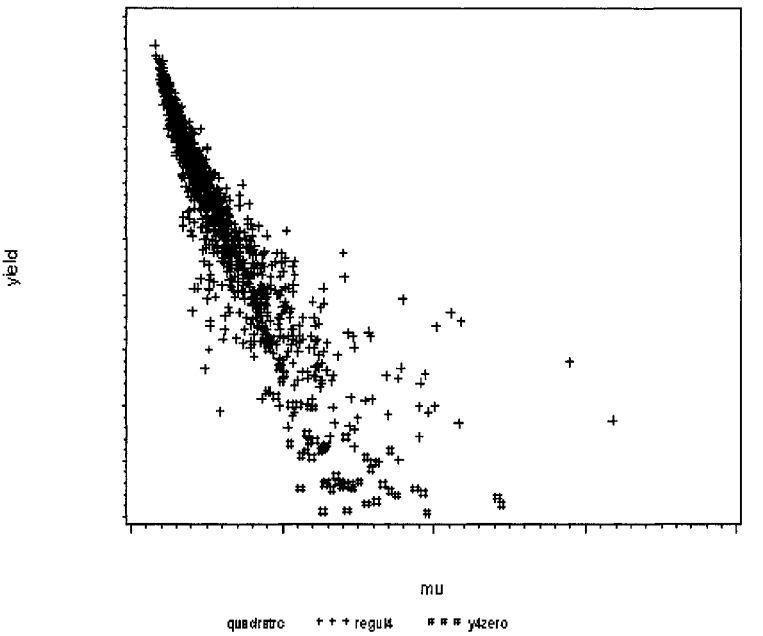


Figure 15 μ and yield for all wafers

to the bootstrap experiment for one primary wafer, and is labeled by the primary γ and the corresponding bootstrap standard deviation. The correlation between γ and the bootstrap mean is over 95%, indicating the general reliability of the bootstrap estimation. The data are seen to be fairly regular, with a regression line of

$$\gamma_{\text{stdev}} = 0.09 + 0.12\gamma_{\text{mean}}. \tag{4.21}$$

The small cluster of data points at the bottom of the chart, near $\gamma = -0.3$, correspond to the y4zero wafers.

Unfortunately, the analogous spreads in μ or y_S are not nearly as compact. The resulting standard deviations can be very large, as is shown clearly in Figure 17 for y_S. Most of the wafers seem to behave regularly, but there are some exceptions in which the standard deviations are comparable to the primary values. The y_S standard deviations seem to be smallest near $y_S = 1$, as expected, for that is the physically most meaningful value for the gross yield.

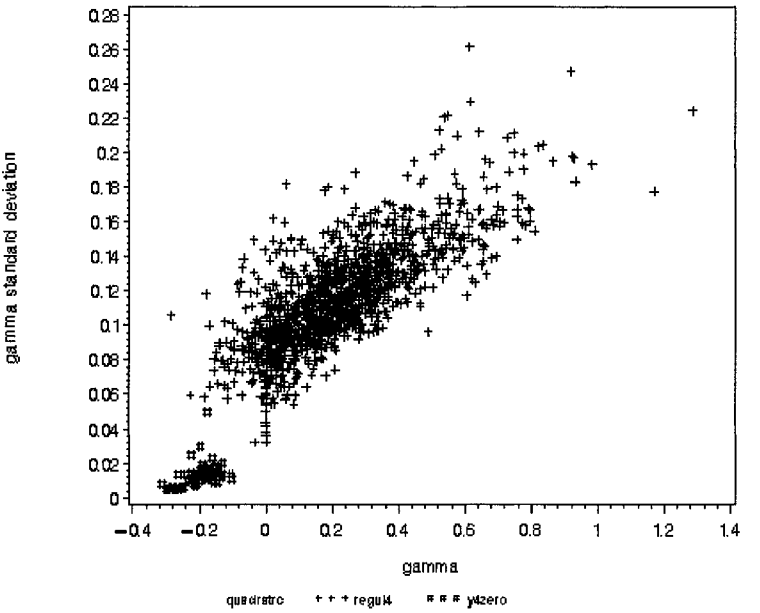


Figure 16 Ranges of γ values

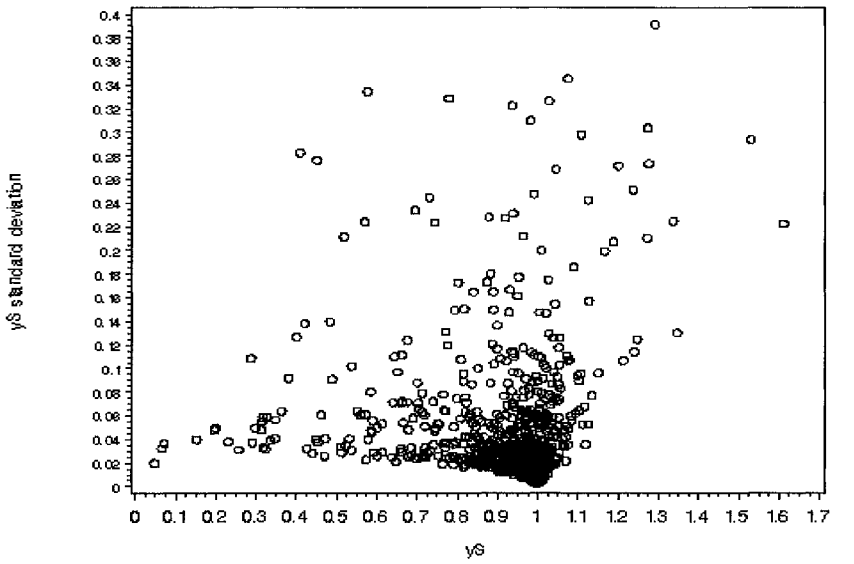


Figure 17 Ranges of y_S values

Chapter 5

Statistics of Embedded Object Fails

Integrated circuits can contain many objects of a more or less similar nature. Examples are embedded memories and scan chains. Some of the tests in the test sequence are intended to verify that such objects are defect free. When the objects are scan chains, these tests are the scan tests, and, if the clock control of the chains has been designed appropriately, the flush test. When the objects are embedded memories, the tests are specialized memory tests. In the latter case, the test patterns are often generated on the chip itself by a so-called ABIST (Array Built-In Self Test) engine. The memory test step applied from the tester then consists of initializing the ABIST engine, and providing the proper number of clock pulses.

The statistical properties of embedded object fails, such as the expected values and variances of the fail probability of a single object, and the correlations between fails of different objects, provide another route to learning about defects, and, in particular of course, about defects that are important for the objects in question. In practice, most of the defect mechanisms that can affect the electronic circuitry on the chip will also affect scan chains or embedded memories, and oftentimes both. As the objects are simpler and more regular than the logic circuitry, using them to learn about what causes chips to fail is usually more expedient than attempting to diagnose fails caused by defects in the logic itself.

An additional consideration for being interested in object yields is that there are usually many of them on today's integrated circuits. Because of their relative diagnostic simplicity, they can conveniently be used as embedded, and free, defect monitors.

In the total sequence of test steps, the tests that target similar objects often occur together in one step. This step has some index, say k, as explained in Chapter 3. The total population of integrated circuits that enter test can then be divided into three groups. The first group consists of those devices that fail tests preceding the object tests. No information about the objects is available in this group, as the appropriate tests were never applied. This leaves NY_{k-1} chips that were tested with the object tests.

The second group, called the fail group, consists of those devices that did fail the object tests, and the third group of those that passed them. There are N_k chips in the second group and NY_k in the third one.

In practice both first and third groups are not represented in the fail data, although they can often be reconstructed with some effort. There is very little

one can do with the first group, because fails in that group typically are caused by gross problems, like excessive IO leakage or shorts between Vdd and Ground. It is better to consider the second and third groups as constituting the total population. The statistical properties mentioned above are, therefore, related to the devices that were tested with the object tests.

The statistical analysis is simplest if all devices that were tested with the object tests are available, whether they passed those tests or not. Detailed information can then be extracted from the data. In particular, of course, the fail probabilities of all the objects can be estimated, as well as the relationships of those probabilities to the sizes of the objects. In addition, suspected correlations between the fails of different objects can be verified, and their strengths determined.

Collecting such detailed fail information is costly, however, and various strategies are employed to reduce this cost. Examples are terminating test once one failing object has been found, and collecting complete fail data on only a small sample of devices. All these test time reduction strategies diminish the quality of the information extracted from the fail data, Fortunately, they rarely make extracting meaningful information impossible.

In this chapter, the statistics of object fails will be discussed: the distribution of the number of times an object fails, and the relationship of this distribution to the sizes of the objects. In addition, some measures of correlation will be discussed between objects, based on their fail statistics. A related problem, how to use the object fails to identify systematic defect mechanisms, will be postponed to Chapter 6, where a general approach to commonality and clustering will be developed.

If the objects are collections of the same basic cell, the fail probability of such a cell can be estimated from the fail data, but the result depends on whether cells in different objects can be treated as identical. In sections 4 and 4.4, two different models will be discussed, and cell fail probabilities will be derived from experimental data using these two models.

In sections 5 and 6, the problem of reduced data collection is addressed. It will be shown that meaningful estimates of the object fail and cell fail probabilities can still be obtained.

When the objects are more complex than a mere repetition of the same basic cell, but still are constructed from a small number of different components, the fail probabilities of those components can be estimated. The statistical analysis is considerably more complex than when the objects are constructed from a single cell, however. It will be discussed in Section 7.

1 GENERAL DEFINITIONS

For simplicity, NY_{k-1}, the number of devices that saw the objects tests, will be indicated by K. K, of course, is a random variable, but only in the context of all devices being tested. In this chapter, I will focus on only those devices that were tested with the object tests, and, therefore, in this chapter, K can be treated as a constant.

The number of objects on a device will be indicated by I, and the objects are labeled by an index i = 1, ..., I. The result of applying the object tests to object i is a random variable Z_i. Its value is 1, when object i failed one or more of the object tests, and 0 otherwise. Z_i has the Bernoulli distribution, with some expectation value u_i, and variance $u_i(1-u_i)$. u_i is also the fail probability of the object.

It is often more convenient to work with $O_i = KZ_i$, the number of devices on which object i failed the object tests. O_i has the binomial distribution with expected value Ku_i, and variance $Ku_i(1-u_i)$. Random variables Z_{ij} describe the events that both objects i and j fail simultaneously. The corresponding probabilities of these events are u_{ij}. Similarly, O_{ij} is the number of devices on which both objects i and j fail. Its expected value is Ku_{ij}.

The fail probabilities u_i are not completely arbitrary, because the objects considered here typically have a size s_i, which indicates the number of cells in the object. If the object is a RAM, the cells are real RAM cells; if it is a scan chain, the cells are scan latches. When such a size exists, and if the cells fail independently,

$$u_i = 1-(1-t_i)^{s_i}, \tag{5.1}$$

in which t_i is the probability that a single object cell fails.

Considering cells rather than the objects themselves is helpful, if the object can realistically be considered to be a set of identical cells, for it allows us to focus on the size independent aspects of the fails, and, in particular, to compare fail probabilities of objects with very different sizes.

If the objects are similar, for example if they are all SRAMs or all scan chains, the fail probabilities of the cells are expected to be equal to some global cell fail probability. This global cell fail probability will be indicated by t. Even if the cells are all of the same type, however, their fail probabilities may still not be equal because of design or manufacturing differences. I will refer to the homogeneous model as the one in which all cells have the same fail

probability, and to the heterogeneous model as the one in which the cells may have different fail probabilities.

A more sophisticated model is the one in which the objects belong to one of a number of groups of objects, such that all the cells within one group have the same cell fail probability, while objects in different groups may have different cell fail probabilities. This model makes the statistical analysis very complex, in particular when the group compositions need to be determined from the data. I will not address this model here.

It is of course possible to consider other, even more complex scenarios. For example, the homogeneous model may be valid, but on a wafer by wafer basis only. In other words, all the cells in the different objects are assumed to have the same fail probability on any given wafer, but the fail probability may vary from wafer to wafer. Or, the wafers may be clustered together into groups, for example using the clustering technique outlined in Chapter 3.5, and the single cell fail probability may be assumed to be the same for all wafers within one group, but to vary between groups. All these more complex scenarios can be treated with the same statistical technique that will be used for the simple homogeneous and heterogeneous models, but will not be addressed explicitly in this chapter.

2 CORRELATIONS AND CLUSTERING

In later sections in this chapter, it will often be necessary to assume that objects fail independently. This assumption may be justified because of previous observations, or because of our knowledge of the manufacturing process. It is safer, however, to verify its validity whenever possible using the collected fail data.

Objects can be correlated, for example, because they fail or pass the same tests together significantly more than they would if they had been independent. They can also seem to be correlated when they have cell fail probabilities that are significantly larger than those of other objects. The first type of correlation can be studied by estimating, for example, the classical correlation coefficient between the pass and fail events of the objects, the second one by estimating individual cell fail probabilities.

The correlation coefficient $\rho(i,j)$ between two objects i and j equals

$$\frac{u_{ij} - u_i u_j}{\sigma_i \sigma_j}, \tag{5.2}$$

in which σ_i equals $\sqrt{u_i(1-u_i)}$. $\rho(i,j)$ can be estimated by

$$\frac{KO_{ij} - O_iO_j}{\sqrt{O_i(K-O_i)O_j(K-O_j)}}. \tag{5.3}$$

Rather than estimate the correlation between two distinct objects, it is often useful to analyze the correlation between two devices using the pattern of passing and failing objects on both. Such commonality analysis will be addressed in detail in Chapter 6.

3 EXAMPLE OF EMBEDDED OBJECT FAILS

SRAM fail data were collected for a large ASIC design. This design has seventy three embedded SRAMs, ranging in size from about 7K to over 1M bits. The sample consisted of 101 devices on which at least one of the memories failed at least one of the memory tests. The experimentally determined fail probabilities are shown in Figure 18.

The fail probabilities clearly depend on the size of the objects, in this case the number of bits in the SRAM. In addition, there is a substantial spread in fail probabilities at any given size. The size dependence can be removed by plotting cell fail probabilities, using Equation (5.1), and the result is shown in Figure 19. Interestingly, the spread at high memory sizes has been replaced by a large spread at very small memories. The latter spread can be explained as resulting from statistical fluctuations (see Section 4.4.) Some spread at large sizes is still present, though, and will be discussed in Section 4.4 as well.

Correlation analysis using Equation (5.3) shows that most of the RAMs fail independently, although there are some that are perfectly correlated (always fail together). The picture that emerges from this initial analysis, therefore, is seventy three largely independent embedded memories of various sizes that fail with a more or less constant cell fail probability. Further analysis of the fail data will be done in Section 4.4.

4 OBJECT AND CELL FAIL PROBABILITIES

The result of applying the object tests to a set of K devices is the set $\{O_i\}$ of numbers of devices on which the various objects were observed to fail. The likelihood function L is the probability that this particular outcome is obtained, and, if the objects are independent, equals

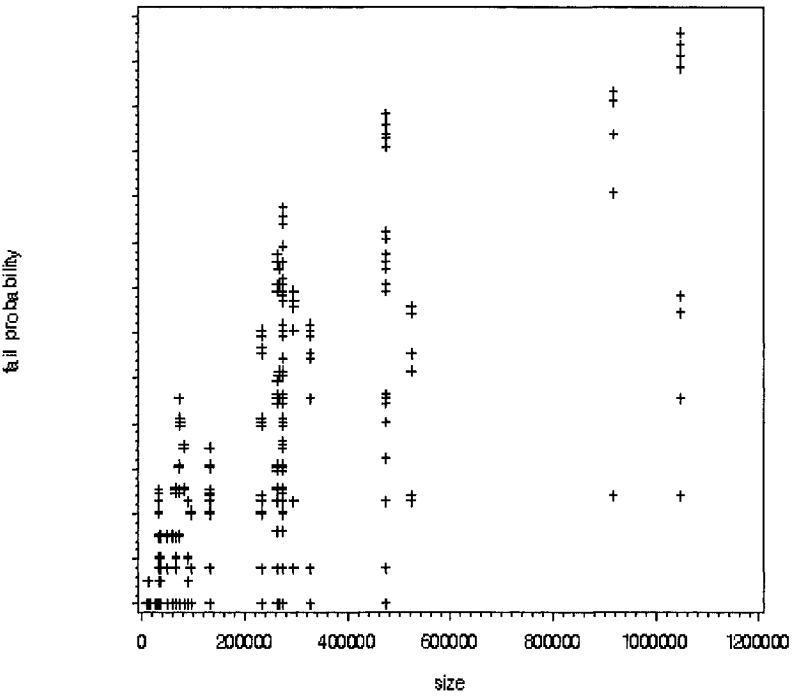


Figure 18 Fail probabilities of embedded SRAMs, as a function of SRAM size

$$\prod_i u_i^{O_i}(1-u_i)^{(K-O_i)}. \tag{5.4}$$

When the objects are independent, the fail probabilities of the objects can be estimated directly from the fail data using the maximum likelihood method (see Chapter 2.2). This approach can also be used to verify the hypothesis that the cells in all the objects fail with the same probability.

4.1 Estimating cell fail probabilities

The parameters in the statistical model are the fail probabilities u_i, which should be estimated from the fail data. In fact, if the objects are constructed from single cells and have a definite size, it would be even better to estimate the cell fail probabilities, defined in Equation (5.1). I will assume here that that is the case, and focus on the cells rather than the objects.

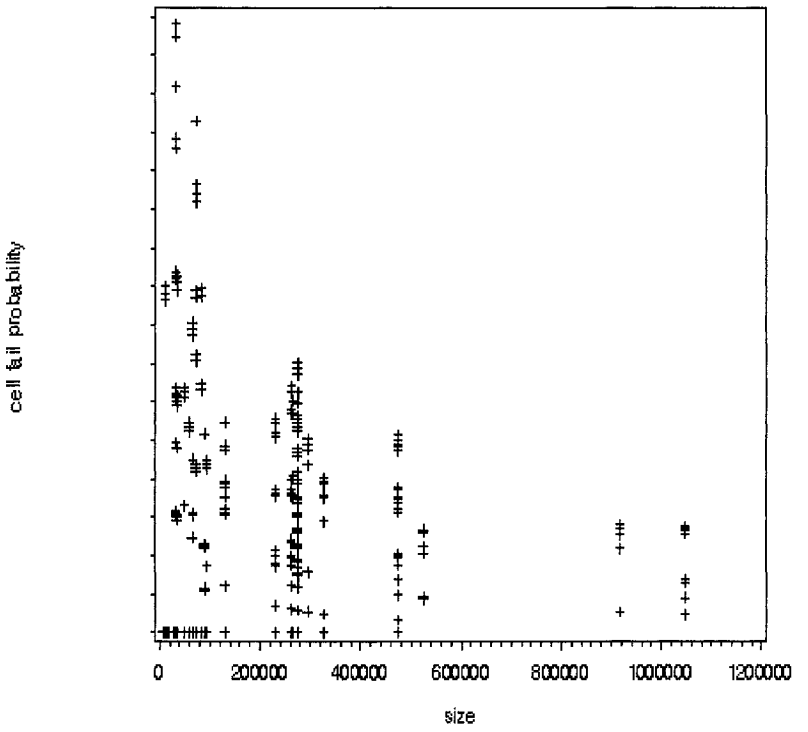


Figure 19 Cell fail probabilities of embedded SRAMs as a function of SRAM size

A general approach to estimating the cell fail probabilities is the maximum likelihood method (see also Chapter 2.2,) which requires that L, as a function of the fail probabilities, is maximized. In the heterogeneous model, L is a function of the set $\{t_i\}$, and in the homogeneous model of the global fail probability t. The results are estimates $\hat{t}_i$ and $\hat{t}$, respectively, of the true cell fail probabilities.

As is shown in Appendix E, the maximum likelihood equation has, as expected, the solution

$$\hat{u}_i = 1 - (1 - \hat{t}_i)^{s_i} = O_i/K \tag{5.5}$$

in the heterogeneous case, and

$$\sum_i \frac{O_i s_i}{1-(1-\hat{t})^{s_i}} = K\sum_i s_i, \tag{5.6}$$

in the homogeneous one.

The solutions of the maximum likelihood equations are random variables with certain statistical distributions. When K is large, these distributions are approximately normal. The means of these normal distributions are the true values of the cell fail probabilities, and their covariance matrices are minus the inverses of the matrices of second derivatives of ln(L).

The variances are calculated in Appendix E. The matrix of second derivatives turns out to be diagonal in the heterogeneous model, and the variance of $\hat{t}_i$ is

$$\frac{\widehat{u_i}(1-\widehat{u_i})}{K}\left(\frac{\partial u_i}{\partial t_i}\right)^{-2}. \tag{5.7}$$

In the homogeneous model, the matrix is a scalar, for there is only one variable, and the variance of $\hat{t}$ equals

$$\left(\sum_i \frac{O_i}{\widehat{u_i}^2(1-\widehat{u_i})}\left(\frac{\partial u_i}{\partial t}\right)^2\right)^{-1}. \tag{5.8}$$

The derivatives in these expressions are evaluated at the solutions of the maximum likelihood equations.

The derivatives are functions of s_i and $\hat{t}_i$ (or $\hat{t}$,) but Equations (5.7) and (5.8) are written in their particular forms to highlight their underlying simple structures. For example, the factor $\widehat{u_i}(1-\widehat{u_i})/K$ in Equation (5.7) is the standard binomial variance, while the factor containing the derivative simply changes the scale due to the change of variables from u_i to t_i. Equation (5.8) is of course more complicated, but the terms in the sum have the same structure, for $O_i/\widehat{u_i}$ is approximately equal to K.

4.2 Comparing different models

The obvious question is, of course, which model is better: the heterogeneous model or the homogeneous one ? The heterogeneous model has more parameters than the homogeneous one, and is, therefore, more complex, but always gives a better fit of the observed data. This question was addressed in Chapter 2.2.2: the heterogeneous model never has worse agreement with experiment than the homogeneous one, even when the latter one is correct, because of statistical fluctuations. Of course, a certain amount of fluctuations is always expected, so the real question is whether the fluctuations are excessive, or are more or less what they are expected to be statistically. If the former, the heterogeneous model should be used, otherwise the homogeneous one.

To determine whether the fluctuations are excessive, the likelihood ratio Λ can be used. In Chapter 2.2.2, the indicator

$$\rho = \frac{-2\ln\Lambda - N_{DF}}{\sqrt{2N_{DF}}} \tag{5.9}$$

was suggested, in which $N_{DF} = I - 1$ is the number of degrees of freedom. The result was that, if ρ is large, the homogeneous model cannot explain all the variability in the data, and needs to be rejected.

4.3 Small fail probabilities

Equations (5.5) through (5.9) become much simpler when the fail probabilities of the objects are small, for then $u_i \approx t_i s_i$ in the heterogeneous model, and $u_i \approx t s_i$ in the homogeneous one. The maximum likelihood estimators become

$$\hat{t}_i \approx \frac{1}{K}\frac{O_i}{s_i} \quad \text{and} \quad \hat{t} \approx \frac{1}{K}\frac{\langle O\rangle}{\langle s\rangle}, \tag{5.10}$$

respectively. In Equation (5.10), <O> and <s> are the averages of the O_i and s_i, respectively. The corresponding variances are

$$\frac{1}{K^2}\frac{O_i}{s_i^2} \quad \text{and} \quad \frac{1}{K^2 I}\frac{\langle O\rangle}{\langle s\rangle^2}, \tag{5.11}$$

in which I is the number of embedded objects.

If we now abbreviate the standard deviation of $\hat{t}$ by δ, we find

$$\hat{t}_i \approx \frac{O_i}{s_i}\,\delta\sqrt{I}\frac{\langle s\rangle}{\sqrt{\langle O\rangle}} \quad \text{and} \quad \hat{t} \approx \delta\sqrt{I}\sqrt{\langle O\rangle}\,, \tag{5.12}$$

while the standard deviation of $\hat{t}_i$ can be written as

$$\delta\sqrt{I}\sqrt{\frac{O_i}{\langle O\rangle}}\frac{\langle s\rangle}{s_i}\,.$$

The importance of the last three equations is that the various quantities, in particular $\hat{t}_i$ and $\hat{t}$, can be compared with each other and with the standard deviations of the former without having to know K, because they are all K independent multiples of δ.

For example, the likelihood ratio Λ can be written as

$$\frac{\langle O\rangle}{\langle s\rangle}\prod\frac{s_i}{O_i}\left(1-\frac{I}{K}\left(\frac{\langle O\rangle}{\langle s\rangle}\mathrm{Cov}(O,s)-\sigma^2(O)\right)\right)\,, \tag{5.13}$$

in which Cov (O,s) is the covariance of the O and s vectors, and $\sigma^2(O)$ is the variance of the former. The covariance will be small if the objects have all roughly the same size. If we also assume that the variance of the O vector is small compared to K, Λ depends solely on ratios of s and O, and can easily be computed without having to know K. Equation (5.9) can then be used to gauge whether the homogeneous model is appropriate, or needs to be replaced by the heterogeneous one.

4.4 Example of cell fail probabilities

The analyses described in the previous section were applied to the fail data described in Section 3. As was pointed out in that section, the individual cell fail probabilities are more or less the same for all seventy three embedded SRAMs. The spread in the probabilities is large when the SRAMs are small, as expected, considering the variance of the cell fail probability, Equations (5.7) and (E.17), which show that the standard deviation is inversely proportional to the size of the memory. For example, several of the smaller memories have a size of about 7K bits, and for these RAMs the standard deviation is about the same as the fail probabilities themselves. Hence the large statistical fluctuations.

The homogeneous and heterogeneous models were compared using Equation (5.9), with the result that $\rho = 7$. This is larger than the cutoff of three that was mentioned in Section 4.2, so the homogeneous model should be rejected. In fact, Figure 19 shows that the spread of the fail probabilities of the larger memories, although much smaller than that of the smaller ones, is still substantial. The likelihood ratio test merely confirms that not all the variability is due to normal statistical fluctuations.

5 PARTIAL DATA COLLECTION

When testing embedded objects, it is often desirable to stop as soon as some object has been shown to be defective. In the simplest case, the objects are tested sequentially, in which case stopping on first fail is a natural way to reduce test time. Often, however, may objects are tested in parallel, and the pass/fail status of the various objects are observed more or less simultaneously. For example, the embedded objects might be associated with pass/fail latches, one latch per embedded object, whose values are set when the embedded objects are tested. The status of the various objects are then determined when the values in those pass/fail latches are scanned out.

Stopping when the first failing object is observed is universally called the "Stop On First Error" protocol, or SOFE. Under SOFE, the status of objects whose pass/fail status is observed after that of the failing one(s) are not known. In this section, I will show that, despite the absence of complete information, reliable fail probabilities can still be obtained. We only need to assume that the order in which the fails are observed is known and fixed for all objects.

The pass/fail status of the embedded objects are observed in some order. There are three possibilities for any pair of objects i and j:

1. i precedes j; that is, it is observed before j.
2. i and j are simultaneous, meaning that they are observed at the same time. Simultaneity occurs, for example, when associated pass/fail latches are in different scan chains, but at the same relative location with respect to the scanout sides of their chains.
3. i succeeds j; that is, it is observed after j. Alternatively, one can say that j precedes i.

When a given IC is tested, the output syndrome $\mathcal{A}$ is the set of objects that failed the tests. $\mathcal{A}$ may be empty, indicating that all objects passed their tests.

When $\mathcal{A}$ is not empty, all the objects in $\mathcal{A}$ should be mutually simultaneous under SOFE.

If a more complex stopping protocol is followed, the contents of $\mathcal{A}$ should be consistent with that protocol. For example, if the protocol is "stop on n^{th} error", there should be a subset of at most n objects among those in $\mathcal{A}$ that can be linearly ordered according to observational precedence, and such that all other objects in $\mathcal{A}$ are simultaneous with some object in the subset.

The probability $P_{\mathcal{A}}$ of a given syndrome $\mathcal{A}$ equals 0 when $\mathcal{A}$ is not consistent with the stopping rule, and otherwise

$$\prod_{j \notin \mathcal{A}, j \leq \mathcal{A}} (1 - u_j) \prod_{i \in \mathcal{A}} u_i , \tag{5.14}$$

where $j \leq \mathcal{A}$ indicates that j does not succeed any of the objects in $\mathcal{A}$.

The first product is over those objects that are not in $\mathcal{A}$, but that are observed before or at the same time as any of the objects in $\mathcal{A}$, and that are, therefore, known to have passed the tests. Note that there is factor $1 - u$ for each such object. Likewise, there is a factor u for each object in the syndrome $\mathcal{A}$, that is, for each object that failed the test. When $\mathcal{A}$ is empty, the first product is over all objects, while the second one equals 1.

The likelihood function L of an observed set of syndromes, one for each tested device, is the product of all such syndrome probabilities. Each object i gives rise to a factor of the form $(1 - u_i)^{n_i} u_i^{m_i}$, with n_i the number of ICs on which i is known to have passed the tests, and m_i the number of times it was observed to have failed. Consequently,

$$\ln L = \sum_i (n_i \ln(1 - u_i) + m_i \ln u_i) . \tag{5.15}$$

The resulting maximum likelihood estimate of u_i is

$$\widehat{u_i} = \frac{m_i}{n_i + m_i} , \tag{5.16}$$

while the covariance matrix is diagonal, with diagonal elements

$$\frac{\widehat{u_i}(1-\widehat{u_i})}{n_i+m_i}. \tag{5.17}$$

This result is entirely what one would expect if the only available information is that there are n_i ICs on which object i is known to have passed the tests, and m_i ICs on which it was observed to have failed.

Those objects that are observed rather late, and that are, therefore, often obscured by fails of earlier objects, will tend to have small values of both n_i and m_i. The corresponding fail probability can still be estimated, but with a severely reduced accuracy compared to when all objects are always observed (in which case $n_i + m_i$ in Equation (5.17) needs to be replaced by the number of ICs.) The only exception is when both n_i and m_i equal 0, which happens only if there is some object that always fails, and that precedes i. In that case, there truly is no information about i, but a lot of information about the always failing one.

6 SAMPLING DEFECTIVE DEVICES

When testing is completed, it is of course known which devices failed the object tests. It may not always be known, however, which particular objects on a given failing device failed the tests, because collecting the detailed fail information about all the embedded objects can be costly, and is not always done on all devices.

In this section, I will consider the case that only a number of failing devices is selected for complete fail data collection. This can be all failing devices, if their number is small enough, a fixed fraction of failing devices, or some predetermined maximum number of them. This group of devices will be called the characterization group.

The fail information now consists of three categories. First, there is detailed pass/fail information about all the embedded objects in the characterization group. This group contains M devices, and is such that each device in it failed the object tests. Second, there is the incomplete information for the failing devices that are not in the characterization group. The total number of devices in the characterization group and in this second group is known, and, in conformance with earlier usage, will be indicated by N_k. Third, there is the implicit information that all the objects passed the object tests in those devices that did not fail the object tests. The problem at hand is how to estimate the object fail probabilities from these disparate pieces of information.

The information contained in the detailed fail data from the set of M devices in the characterization group may not allow us to estimate the fail probabilities directly. What can always be obtained, however, are the probabilities p_i that an object i fails, given that the device belongs to the characterization group. It can be approximated as

$$p_i \approx O_i/M, \tag{5.18}$$

where O_i is the number of times object i fails in the characterization group. The usefulness of p_i lies in its relationship to u_i, which will now be derived.

Let R be the probability that a given device being tested by the object tests fails those tests. If all the objects fail independently, R is a simple function of the fail probabilities of the objects (see Equation (F.1)). Sometimes, however, different objects are not independent, and R is some fixed but unknown quantity. It can be approximated by N_k/K.

As the characterization group is a random selection of M devices from the failing ones, the probability p_i that object i on a given device in the characterization group fails the object tests is the same as that in the group of all failing devices. The probability that this object, say i, fails the object tests in all devices, failing or not, equals u_i, and the probability that a device fails the object tests equals R. Consequently,

$$p_i = u_i/R. \tag{5.19}$$

As in Section 3, we want to be able to assume that the objects fail independently. No correlation coefficient as in Equation (5.3) can be calculated because no complete information is available. The approximate value of R still allows us, however, to estimate the correlations between object fails.

p_{ij}, the probability within the characterization group that i and j fail, equals u_{ij}/R, similarly to Equation (5.19). If objects i and j fail independently, u_{ij} equals the product u_iu_j, and, consequently, p_{ij} equals p_ip_jR. Clearly, independence between i and j does not imply that p_{ij} equals p_ip_j. Equality holds only when R equals 1.

Using Equation (5.18) and

$$p_{ij} \approx O_{ij}/M. \tag{5.20}$$

Equation (5.3) can now be rewritten as

$$\frac{MO_{ij} - O_i O_j R}{\sqrt{O_i(M - O_i R)O_j(M - O_j R)}}, \tag{5.21}$$

in which O_i and O_{ij} refer to the numbers of fails in the characterization group only.

Note that Equation (5.21) requires knowledge of R. So far, the only estimate available for R is N_k/K, and, therefore, this equation needs K. If K is not known, for example because nothing is known about devices that did not fail the object tests, correlations between the fails of different objects cannot be estimated this way. In practice, one proceeds by assuming no correlation, and verifies later that the assumption was justified.

If the correlation coefficients (5.21) show that the objects fail approximately independently, or if one simply assumes that they do, p_i can be estimated using the maximum likelihood method, as shown in Appendix F. In the heterogeneous model, $\widehat{p_i} = O_i/M$, as expected, and in the homogeneous model equation (5.6) is still valid, with K replaced by M, and u by p. Both R and the desired probabilities u_i can then be estimated as well, as shown in the same appendix.

In both models, R is between 0 and 1. It equals 0 only in the exceptional circumstance that exactly one object fails on every chip in the characterization group. This may seem self contradictory for a failing object ipse facto shows that the chip can fail, and, therefore, that R is not 0. On the other hand, the circumstance in which R is 0 seems to violate the assumption of independence between the objects, and the result of any calculations based on this independence should not be taken too seriously. R equal to 1 is less questionable as a result. In the heterogeneous model, it occurs when any object fails on all chips in the characterization group, but in the homogeneous model only when all embedded objects fail on all chips.

The choice between the two models can now be made as outlined in Section 4.2. With the estimated value of R, Equation (5.21) can be used to verify that the objects did indeed fail independently. A large value of any element in the correlation matrix indicates that the assumptions that underlie the calculations may not be valid.

Some final comments. Using the characterization group only gives nontrivial results when there is more than one object, as discussed in the appendix. On the other hand, if there is more than one object, not only the quantities mentioned above can be estimated, but also K, the number of devices that

were tested with the object tests. For K is approximately equal to N_k/R, where N_k is the number of failing devices.

7 FAIL PROBABILITIES OF OBJECT COMPONENTS

In many cases, embedded objects are not complex chunks of random digital logic, but, instead, are simple collections of components from a small set of component types. Examples are SRAMs, which are rectangular arrays of one-bit cells, and scan chains, which are linear strings of flip-flops or master-slave latch pairs. Embedded objects, however, need not be simple one or two dimensional repetitions of the same components, but may contain many different ones. Scan chains, in particular, can contain many latch types that may differ in the number of data ports, output drive strength, noise immunity, etc.

In this section, I will address the problem of estimating the fail probabilities of these different component types, using, as input, only the passes and fails of the embedded objects. Estimating the component fail probabilities, as it turns out, is considerably more complex than estimating the fail probabilities of the embedded objects themselves.

The appropriate statistical model will be described in the next section. It relates the probability u_i that object i passes all the object tests to the numbers of components of each type there are in the object, and to the probabilities that those component will pass the tests. The general equation is

$$\ln(1-u_i) = \sum_j \beta_j n_{ij}, \tag{5.22}$$

in which the independent variable n_{ij} is the numbers of components of type j in the object, and β_j is the logarithm of the expected yield of that type. This is similar to the well-known logistic regression equation [10]

$$\ln\frac{u_i}{1-u_i} = \alpha + \sum_j \varepsilon_j n_{ij}, \tag{5.23}$$

where n_{ij} has the same meaning as above. α and ε_j are the logistic parameters to be estimated, but are harder to interpret than β_j in Equation (5.22). The mathematical techniques for estimating the parameters are the same in both models.

7.1 Component fail estimates

Let us assume that there are k different types of components out of which the objects are constructed. The probability that a component of type j, $j = 1, \ldots, k$, is defect-free is q_j, and we want to estimate these probabilities. The fail probability of the component is, of course $1 - q_j$, but we rather use the probability of passing the tests, as it is more convenient. In a more general model, these probabilities depend on the objects in which the components are located, but we will not use that more general model here.

Object i has n_{ij} units of type j. The object has a size, which, however, is more general than the one discussed in Section 1, for it now is built from more than one type of component. The size of the object is defined as $s_i = \sum_j n_{ij}$. The objects can pass or fail the object tests, but the number of devices for which test results are known can be different for different objects. The actual number of devices in which object i is tested is M_i, and there are O_i passes and $M_i - O_i$ fails.

Assuming that components fail independently, the probability that object i fails the object tests is

$$u_i = 1 - \prod_j q_j^{n_{ij}}, \tag{5.24}$$

which, after taking logarithms, becomes Equation (5.22). The outcome of an experiment is the numbers of passes and fails of the various objects. The corresponding likelihood function is

$$L = \prod_i u_i^{O_i} (1 - u_i)^{M_i - O_i}. \tag{5.25}$$

Finding the maximum of L, and, thereby, the best values of the component fail probabilities, is described in Appendix G. No clean, analytic expressions are available for the estimates of the component yields. Here, we will only discuss some special cases.

7.2 Special cases

One potentially anomalous case is that of some component having zero probability of passing the tests. If some component always fails at least some of the object tests, all objects that contain this component will also always

fail. In other words, $q_j = 0$ implies $u_i = 0$ for all objects for which $n_{ij} \neq 0$. It, therefore, also implies that the corresponding O_i should be 0.

If we now invert these implications, it turns out that the probability that some component will pass the test cannot be 0 if there are some objects that contain that component and that sometimes pass the tests. In fact, if some objects always fail the tests, and if there are one or more units that are contained in only those objects and not in any other objects, it is better to remove those objects and those units from consideration, because some of the units might have a zero probability of passing the object tests.

Another special case is when all objects have the same relative contents; that is, when n_{ij}/s_i is a constant ν_j that depends only on j. In that case, we can write the occupancy numbers as $n_{ij} = \nu_j s_i$, and Equation (5.24) becomes

$$u_i = 1 - \left(\prod_j q_j^{\nu_j}\right)^{s_i}, \tag{5.26}$$

or, more meaningfully, $u_i = 1 - \pi^{s_i}$, with π the average component yield. L is then a function of π alone, and only π can be obtained from the test data.

Incidentally, Equation (5.26) is the same as Equation (5.1) in the homogeneous model, with t equal to 1 - π. The connection with the heterogeneous model will be made below.

The derivation of Equation (5.26) shows that, to get information about the individual component probabilities, different objects should have different relative contents. Obviously, the more the contents of the various objects differ, the more accurate those probabilities can be estimated. In the limit of each object consisting of only one component type, the accuracy is maximized.

In this limit case, for each component type j, n_{ij} equals 0 for some objects and s_i for all others. The maximum likelihood equations (G.3) then split up into k different groups, one for each component type, and the equations for different component types are independent. This is in fact the heterogeneous model of Section 4, with the potential refinement that some objects are made to have identical cell fail probabilities when the components out of which those objects are constructed are the same.

The resulting equations are very much like those for designed experiments, if we consider each object as an experiment for a particular component type. There may be multiple experiments per component, and the numbers of experiments for different components may differ.

Chapter 6

Fail Commonalities

The focus of the discussions in the previous chapters were various types of fail probabilities: first fail probabilities at the different steps in the test sequence, or fail probabilities of embedded objects and of object cells. These probabilities can be estimated using the fail data, if the latter were collected in sufficient volume, and with sufficient detail.

Fail data can also be used in a less quantitative way, however. They can be used as a signature, so to speak, of the underlying defect, the one that caused the fail. Such signatures can be defined for each failing device using the raw fail data or some kind of summarization of the fail data.

The importance of fail signatures is that they can be used to compare different devices, or the same device under different test conditions, and to determine whether the fails of those devices could have been caused by the same defect mechanism. Once a way is found to reliably compare different devices, the failing devices can be clustered into groups of devices that seem to have failed because of the same or similar defects.

One reason for attempting such clustering is that, if all the devices in a single group did fail because of similar defects, information like occurrence probabilities of such defects is available immediately. In addition, diagnosis can then be targeted to the more frequent defects.

More importantly, clustering failing devices focuses on the presence of systematic defects. The Null hypothesis of manufacturing is that there are no systematic defects: all defect producers are assumed to act with equal strengths on all wafers, and to produce defects randomly, and independently of each other. If true, commonality analysis would find no clusters, at least not any large ones. On the other hand, if large clusters are found, it shows evidence of some systematic defect. The goal of comparing fail signatures, and of clustering failing devices on the basis of such comparisons, is to uncover the presence of systematic defects, if any exist. Once identified, further diagnosis can provide more information about them.

One type of commonality was discussed in Chapter 3.5. In that chapter, commonality between wafers was studied on the basis of detailed fallout histories. In the present chapter, commonality between devices, whether on the same wafer or in the same lot, will be treated. The fail data that will be employed to define signatures are either the lists of failing embedded objects on the devices, or the lists of failing latches. Because the latter lists may be

very large, various ways to summarize them without losing essential information will also be studied.

The basic steps to be taken are fairly straightforward. First, define a measure of commonality between the fail data of two failing devices; second, identify clusters based on the commonality measure. Commonality analysis can be performed for all devices failing any of the deterministic tests, or can be done on, for example, a corner by corner basis. Finally, develop suitable cluster signatures to identify future occurrences of the same underlying defects.

All the techniques described here, other than perhaps the actual commonality measures, are standard data mining techniques [19]. A similar analysis was for example applied to Iddq data in [32].

This type of analysis will be performed on large numbers of devices. Consequently, the analysis has to be fast. This requirement limits the amount of work that can be performed on the fail data. It is the opposite of logic diagnosis, in which accuracy of the result is the driving factor and performance can be sacrificed to it.

This chapter is divided into three parts. First, abstract commonality measures will be defined for pairs of devices, as well as groups of more than two devices. Then, such measures will be described in detail for commonly collected fail data. Finally, a clustering technique will be presented, and results of applying such clustering on some selected sets of fail data will be discussed.

1 COMMONALITY MEASURES

In all cases considered in this chapter, the fail signatures that need to be compared are collections of pairs (n, v), in which n refers to an identifiable object, like an embedded array or a logic book, and v is a number. n can be an actual string, but it can also be a number, like a net index. v is always a number, but it can have arbitrary (non-negative) values, or it can be restricted to 0 and 1. Obviously, each signature has at most one (n, v) pair for any given n. By convention, if a given signature does not have a (n, v) pair for some n, it implicitly contains the pair (n, 0).

A simple example is the signature based on embedded objects. For any given device, the signature contains explicitly pairs (n, 1) for those objects that did fail on the device, with n being the name of the object, while it implicitly contains pairs (n, 0) for those objects that did not fail. If the signature is that for a wafer, the pairs would be of the form (n, v) (or, implicitly, (n, 0))

where n has the same meaning as above, and v is the number of times the object n failed on the wafer.

1.1 Pairwise commonality

There are several ways to compare two signatures. The general approach is to compare the (n,v) pairs in the two signatures one by one by comparing the v values of pairs with corresponding n values. This comparison should end with some number that, for the sake of convenience, will be between 0 and 1.

The two most relevant comparisons to this book are $\cos\theta_{IJ}$ and h(I,J), where I and J are two different signatures. The first one is most suitable when v can have arbitrary values. It starts from the notion of a vector $V = \{v_l\}$. The vector is that of the v values with the n values in some arbitrary but definite order that is the same for all the signatures. If n is a name, it could, for example, be their alphabetical order. v_l is then the v value of the l^{th} pair according to that order.

Next, a cross product $V^I \otimes V^J$ is defined between the vectors I and J by

$$V^I \otimes V^J = \sum_i v^I{}_i v^J{}_i, \tag{6.1}$$

where $v^I{}_i$ is the i^{th} element of vector V^I. Using this cross product, a length $L^I = \sqrt{V^I \otimes V^I}$ is assigned to each vector. Finally, the commonality between two signatures is defined as

$$\cos\theta_{ij} = (V^I \otimes V^J)/(L^I L^J), \tag{6.2}$$

in which θ_{IJ} is the angle between the two vectors in k-dimensional signature space, where the dimensions correspond to the various values n can have, and k is the number of all such possible values.

It follows immediately from this definition that $\cos\theta_{IJ}$ is between 0 and 1. When signatures I and J are strongly correlated, that is, when corresponding v values are almost equal, $\cos\theta_{IJ}$ is close to 1. If I and J are uncorrelated, $\cos\theta_{IJ}$ is expected to be small because corresponding v values will be very dissimilar, and often one or the other will be zero

When the v values are restricted to 1 and 0, this commonality measure can still be used, but in that situation it may be more natural to use h(I,J). The latter is obtained by considering only pairs where the corresponding v values differ. In other words,

$$h(I, J) = 1 - \frac{\sum_i v^I{}_i(1 - v^J{}_i) + v^J{}_i(1 - v^I{}_i)}{\sum_i (v^I{}_i + v^J{}_i - v^I{}_i v^J{}_i)} \tag{6.3}$$

$$= \frac{\sum_i v^I{}_i v^J{}_i}{\sum_i (v^I{}_i + v^J{}_i - v^I{}_i v^J{}_i)}, \tag{6.4}$$

in which the sums are over all the n occurring in either I or J. h(I,J) is a number between 0 and 1. It equals 1 when corresponding v values are the same. When the signatures have little in common, it is expected to be small, because, for each n, one or the other v value is likely to be zero.

This measure can be generalized to a weighted h(I,J) by

$$h(I, J) = \frac{\sum_i w_i v^I{}_i v^J{}_i}{\sum_i w_i (v^I{}_i + v^J{}_i - v^I{}_i v^J{}_i)}, \tag{6.5}$$

in which the weights w_i are arbitrary. Such weights are sometimes useful if some property of the objects make some objects more important than others in gauging similarity. The definition (6.5) is such that h(I,J) still equals 1 when the two signatures match exactly, regardless of the weights.

If signatures are random, their lack of commonality should be reflected in the commonality measures being small. The average values of $\cos\theta_{IJ}$ and h(I,J), however, can be calculated only in exceptional circumstances, and we will essentially trust our intuition that both of them are small if the signatures having little or nothing in common.

1.2 Commonality of sets of signatures

Next, the commonality measure needs to be extended to arbitrary sets of signatures, not just pairs. This can be done in a number of ways [19], but in this chapter I will use only one. Another one was discussed in Chapter 3.5.

Given a set of signatures, each pair of them will have a commonality measure, be it $\cos\theta_{IJ}$ or h(I,J). This measure will be indicated by the generic form o(I, J). Let now {I} be the set of signatures. O({I}), the commonality of the signatures in the set {I}, is then defined as

$$O(\{I\}) = \min_{I, J \in \{I\}} o(I, J). \tag{6.6}$$

As a result, O({I}) will be close to 1 if the signatures in the set are all mutually highly correlated, while it will be close to zero if at least two signatures in the set have low commonality.

It is often useful to have a measure for how little two different sets, say {I} and {J}, have in common. The measure to be used in this book will be indicated by D({I}, {J}) and is the opposite of that in Equation (6.6):

$$D(\{I\}, \{J\}) = \max_{I \in \{I\}, J \in \{J\}} o(I, J). \tag{6.7}$$

Finally, as a warning, notice that O({I}) and D({I}, {J}) depend on the particular choice made for o(I, J). Most of the time the differences will be minor, but there may be cases in which $\cos\theta_{IJ}$ is large and h(I,J) is small or vice versa.

2 EMBEDDED OBJECTS

Fails of embedded objects were treated extensively in Chapter 5. The set of objects that fail on some device provide a convenient signature for that device. As the objects either fail or not, the natural commonality measure is h(I,J). This measure, however, ignores the information contained in the sizes of the objects. This size dependence can be accommodated by using the weighted commonality measure, defined in Equation (6.5). The weights should be such that larger objects have smaller weights than smaller objects.

Possible choices are $1 - O_i/K$, or $1/s_i$, where the various variables were defined in Chapter 5.

3 LOGIC FAILS

The main test of the internal logic circuitry on a device is performed by the deterministic tests. Arrays, scan chains, IOs, all are tested by other tests, but defects that affect the functional operation of the logic on the chip can be tested only with the deterministic test step. As such defects constitute one of the main reasons of why ICs fail, good commonality measures that can identify systematic problems will be of great value.

The result of the deterministic tests is symbolically shown in Table 2.. The

	latch 1	latch 2	...	latch n-2	latch n-1	latch n	# latches	pass /fail
pattern 1	☒		...	☒			l_1	1
pattern 2			...				l_2	0
...	...	...	...	...	...	...	...	...
pattern m-2		☒	...	☒			l_{m-2}	1
pattern m-1	☒		...	☒		☒	l_{m-1}	1
pattern m			...				l_m	0
# patterns	p_1	p_2	...	p_{n-2}	p_{n-1}	p_n	N	
distinct fail	1	1	...	1	0	1		

Table 2. Deterministic Test Results

rows in this table correspond to the patterns that were applied during the test, and the columns to the latches whose values were inspected as part of the test procedure. A latch that, after a given pattern was applied, was found to have an incorrect value is marked by the symbol ☒ in the cell corresponding to that pattern and that latch.

The table contains two auxiliary columns and two auxiliary rows. The column labeled # latches contains, for each pattern, the number of latches that were found to have incorrect values. This number being 0 indicates that the

pattern did not fail. The column labeled pass/fail merely shows whether the value in the previous column is 0 or not.

The two auxiliary rows have similar functions. The one labeled # patterns contain the number of patterns that caused a particular latch to have an incorrect value, while the row labeled distinct fail shows whether this number of failing patterns is 0 or not.

Finally, the total number of failing values that were observed in any of the failing patterns is N, and is the column sum of the values in the # latches column, and is also equal to the row sum of the values in the # patterns row.

The contents of Table 2. represent all the available fail data, without omissions, but in a way that is more suitable to discuss usable signatures. Various types of signatures can now be defined using this table. The most obvious one is to keep the complete table. This is of course the best possible signature, but has the drawback of being rather voluminous.

Two approaches will be discussed in this book. One uses only information in the table, the other one uses the table, and, in particular the distinct fails row, as a starting point for further analysis. I will consider both in turn.

3.1 Signatures based on fail data only

A better choice than all the data in Table 2. is to summarize the fail data using one or more of the auxiliary rows and columns. In fact, the column labeled pass/fail (see Table 3.) is a well known summarization, used in dictio-

	latch 1	latch 2	...	latch n-2	latch n-1	latch n	# latches	pass/ fail
pattern 1	☒		...	☒			l_1	1
pattern 2			...				l_2	0
...	...	...	...	...	...	...	...	...
pattern m-2		☒	...	☒			l_{m-2}	1
pattern m-1	☒		...	☒		☒	l_{m-1}	1
pattern m			...				l_m	0
# patterns	p_1	p_2	...	p_{n-2}	p_{n-1}	p_n	N	
distinct fail	1	1	...	1	0	1		

Table 3. Signature Based on Failing Patterns

nary based diagnosis. The assumption in that type of diagnosis is that the vector of values in that column can be used to nearly uniquely identify the underlying defect. Upon further experimentation it has usually become clear that this vector gives a very poor diagnostic resolution, and is not used in practice. For the same reason, it is not a good signature to use in commonality analysis.

A considerably better choice, as it turns out, is the vector of ones and zeros in the distinct fail row (see Table 4.,) and this vector will be called the unique

	latch 1	latch 2	...	latch n-2	latch n-1	latch n	# latches	pass/ fail
pattern 1	☒		...	☒			l_1	1
pattern 2			...				l_2	0
...	...	...	...	...	...	...	...	...
pattern m-2		☒	...	☒			l_{m-2}	1
pattern m-1	☒		...	☒		☒	l_{m-1}	1
pattern m			...				l_m	0
# patterns	p_1	p_2	...	p_{n-2}	p_{n-1}	p_n	N	
distinct fail	1	1	...	1	0	1		

Table 4. Signature Based on Unique Fails

fails signature. A clearly even better signature is the vector of values in the # patterns row, possibly augmented with the vector of values in the # latches column (Table 5.) It cannot be worse than the unique fails signature, and it requires only marginally more storage. This vector, or the combination of the two vectors, will be called the marginals signature.

With these choices for signatures, choosing the appropriate commonality measures is straightforward. The unique fails signature consists of ones and zeros only, and its appropriate measure is h(I, J). On the other hand, the marginals signature consists of arbitrary value and its appropriate commonality measure is $\cos\theta_{IJ}$. Note that, in all cases, patterns that did not fail and latches that never contained fail data in either signature do not contribute to the commonality measure. In other words, the commonality measures are based solely on fails that occur in at least one of the two signatures.

	latch 1	latch 2	...	latch n-2	latch n-1	latch n	# latches	pass/ fail
pattern 1	☒		...	☒			l_1	1
pattern 2			...				l_2	0
...	...	...	...	...	...	...	...	...
pattern m-2		☒	...	☒			l_{m-2}	1
pattern m-1	☒		...	☒		☒	l_{m-1}	1
pattern m			...				l_m	0
# patterns	p_1	p_2	...	p_{n-2}	p_{n-1}	p_n	N	
distinct fail	1	1	...	1	0	1		

Table 5. Signature Based on Marginals

Further refinements can be made as appropriate. For example, it might be desirable not to use all the failing patterns, but only a suitable chosen subset of them. This can, of course, easily be accommodated by removing rows that correspond to rejected failing patterns from the various tables, and by updating the values in the # patterns row and in the distinct fails row.

3.2 Signatures based on backtracing

The unique fails signature works rather well in practice, in that a high commonality value based on this signature often indicates a common defect cause. The reason for this is probably that the defect that caused the fail is likely to be close to the latches that contained the incorrect values, because fault effects caused by the defect flow along signal wires that do not commonly cross large distances over the chip. Consequently, when fault effects propagate away from the location of the defect, they will usually not travel far before they are stored into latches.

If a fault effect is stored in a latch, this latch will, upon inspection at the end of the test pattern, contain an incorrect value. Such a latch is commonly called a failing latch, although, usually, the latch as a logic circuit is defect free.

If this is true, then it makes sense to follow the fault effects in the opposite direction: start from the failing latches and trace through the logic backwards

until primary inputs, embedded memories, or other latches are encountered, and store all the nodes that were encountered during the tracing in a list.

Backtracing through combinational logic is straightforward, because all the logic elements are unidirectional, and the backtrace always arrives at the output of a logic gate and needs to continue backwards from the inputs. It stops at primary inputs.

Most test patterns are such that embedded memories provide a constant set of logic values on their outputs, and no further tracing needs to be performed through them when they are encountered during the backtrace. Latches, when encountered however, require more consideration.

For example, in Figure 20, the tracing starts at failing latch A. The cone

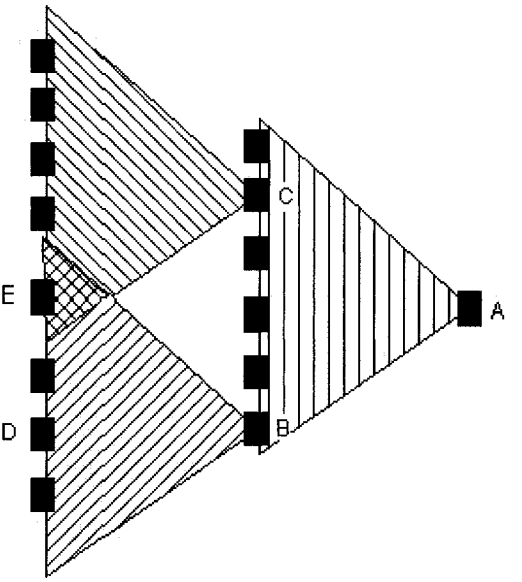


Figure 20 Tracing through backcones

rooted at A contains combinational logic, and is bordered by several latches. Tracing may have to be continued through these latches if they are clocked with a clock pulse that precedes the one that clocks latch A. The figure shows the case that latches B and C are clocked during the application of the test pattern before A is clocked. Consequently, tracing continues through these latches and through more combinational logic, as shown by the additional two combinational cones.

When latches D or E are reached, a decision has to be made whether or not to continue the tracing. At D this decision depends on whether its clock pulse, if any, occurs before the one at B. At E, the choice is more complicated, and the clock pulses at E, B and C all have to be taken into account. Tracing continues at E if its clock pulse precedes either the one at B or the one at C, for fault effects through E could continue through B or through C.

If a latch has multiple ports, as many do, then the tracing should continue only from the port that was clocked. Furthermore, there is a choice between tracing through the data input or through the clock input of a clocked port. Tracing needs to follow only clock inputs when it is clear that the defect does affect the clock lines. In the spirit of single fault diagnosis, the data lines need not be traced in that case. Usually, however, it is not clear whether the defect affects clock lines or not, and tracing has to continue from both clock inputs and data inputs.

Each trace starts from a failing latch and defines a backcone to that latch. This backcone is the sum of the combinational cones and latches encountered during the tracing. Figure 20 shows a complex backcone consisting of three distinct combinational logic cones. Note that backcones from the same latch may be different for different patterns, because the sequences of clock pulses in the patterns may differ. During the backtrace, all the encountered nodes are stored in a list. The backtrace is repeated for each failing latch, and each time a node is encountered the corresponding entry in the list of encountered nodes is incremented by 1 (or by some other value depending perhaps on the backcone or the number of failing latches.)

The resulting signature is a list of (node, v) pairs, in which v is the number of times this node was encountered during the backtraces. A high v value shows that the corresponding node is in the backcones of many failing latches. The v values, therefore, form a rough estimation of the likelihood that the defect is located on or near any of the nodes in any of the backcones.

It is useful to compare this signature with the result of a crude form of diagnosis that is sometimes employed, called intersection. In intersection, backcones are obtained as above, but instead of incrementing counters, the backcones are kept as sets and the intersection is taken of all these sets. The result is a set of nodes that are in all the backcones.

The theory behind this form of diagnosis is that only nodes in the intersection can be the location of the defect, because otherwise fault effects from the defect could not have propagated to all the failing latches. Unfortunately, not all defects affect single nodes. Bridges, for example, affect at least two, and the latches downstream from one leg of the bridge may not be the same as the ones downstream from the other leg. Consequently, intersecting backcones from the failing latches is likely to result in an empty set.

Using the backcones based signature, however, circumvents this problem. It lists all the nodes ever encountered in any of the backtraces, but it ranks them according to how often they were encountered. The group of nodes most often encountered form then a generalization of the intersection, one that does not suffer from the problem of potentially being empty.

As the v component in the (n, v) pairs is an arbitrary number, the most appropriate commonality measure is $\cos\theta_{IJ}$.

3.3 Signatures based on cells

Instead of keeping track of which nodes or blocks are encountered during backtracing, one can also notice their functional properties. These can include the logic function, drive strength, power level, and delay times. All these details are encoded in the cell name of the block, which is a reference to a specific book in the design library of which this block is an instance. All the physical and layout details of the block can be found in the description of the library book.

Monitoring cells rather than nodes during backtracing is sometimes useful when the defect is not one that impacts a specific instance of a library book, but, instead, one that impacts the library book itself; perhaps a defect prone layout style, an underpowered driver, or any other design flaw that will affect all instances of that book.

The resulting signature is very similar to the one discussed in the previous section, except that now the n component of the (n, v) pairs is not the name of a node or a book instance, but the name of the book itself. Signatures that have high counts of certain books hint at problems with that book, rather than at some point defect somewhere on the device.

As with the previous signature, the most appropriate commonality measure is $\cos\theta_{IJ}$.

3.4 Signatures based on diagnosis

An even more complex analysis than backtracing is diagnosis. The details of such an analysis will be presented later in this book, but the results are rather simple: a set of nets or pins, generically called nodes, at least one of which is affected by the defect. The signature is then this set of nodes, or, more precisely, a set of (node, 1) pairs. As the v component of this signature only has the values 0 and 1, the most appropriate commonality measure is h(I, J) (Equation (6.4)).

4 CLUSTERING

Now that means have been defined to measure commonality between the fails of different devices, we can turn our attention to clustering together those devices that seem to share the same failing mechanisms. Intuitively, this is easy: just put those devices in the same cluster that have high values of the chosen commonality measure for each pair of devices. This, however, immediately runs into problems. Consider for examples, A, B and C. A and B have a high commonality value, and so do A and C. B and C, however, have an medium commonality value, and not enough to qualify them for membership in the same cluster. Should the core cluster now be A and B, or A and C ? Either choice would be arbitrary.

To handle such conundrums, an algorithm is needed that reduces the number of arbitrary decisions to a minimum. The chosen algorithm is well known in the literature as the furthest neighbor method [19]. A simplified program that implements this algorithm in a brute force way is shown in Figure 21.

The algorithm starts with as many clusters as there are signatures. At each step of the algorithm, the number of clusters is reduced by 1 by merging two clusters. The selection of the clusters uses the cluster commonality measure $O(\{I\})$, defined above in Equation (6.6), where $\{I\}$ is the set of signatures in the cluster. The two clusters selected for merging are such that, after merging, the commonality measure of the resulting cluster is larger than that of any other pair of clusters. If there is a choice between several pairs of clusters, each pair leading to the same commonality measure of the resulting cluster, then an arbitrary choice needs to be made to pick one of the pairs. The process of merging stops if there is no pair such that the commonality measure of the result of the merger is larger than some threshold t, a number between 0 and 1.

The result of the algorithm is a set of clusters for each one of which $O(\{I\})$ is larger than t. The resulting clusters can be compared to how tight they are, using the $O(\{I\})$ measure (see Equation (6.6)), or to how different they are using the $D(\{I\}, \{J\})$ measure (see Equation (6.7)).

5 EXAMPLES

The first example, Figure 22, is that of a repeater based cluster. The design was a large ASIC with many embedded SRAMs. The commonality was calculated using the pass/fails of these memories (see Section 2.) The largest cluster found using the commonality matrix is shown in Figure 22. This figure is a composite over several wafers. The wafer locations of the devices in the cluster are indicated by the shaded cells. The numbers in the cells, and the

```
// S is set of clusters
S = ();

// put all signatures in S
for each I, put I in S;

// best_pair (S) finds pair of clusters in S that
// produce the highest combined commonality
// measure among all pairs in S.
// It returns a pair of clusters, or an empty set
// if no commonality measure exceeds the
// threshold t.

best_pair (S, t) {
  best = t;
  P = ();
  for all (c1 in S) {
    for all (c2 in S and c2 ne c1) {
      if ((m = measure (merge (c1, c2))) > best) {
        best = m;
        P = (c1, c2);
      }
    }
  }
return (P);
}

// merge (c1, c2)  returns the union of the signatures
merge (c1, c2) {
  return (c1 ∪ c2 );
}


// Main routine
while (P = best_pair (S, t) not empty) {
  C = merge (P);
  remove (S, P);    // remove clusters in P from S
  add (S, C);       // add merged cluster to S
  }

exit;
```

Figure 21 Clustering algorithm

corresponding shade intensities, indicate the number of wafers that contribute devices at this location to the cluster. The empty cells show those wafer loca-

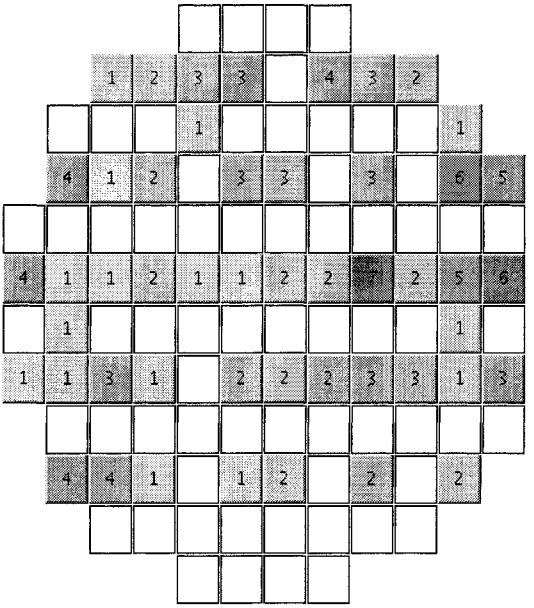


Figure 22 Repeater based cluster

tions that have failing devices on some wafers, none of which were in the largest cluster.

The first sign of a repeater is the clear striping in the largest cluster found using this commonality matrix. As the design is printed from a 1 by 2 reticle, a repeater, probably due to a mask fail, was suspected. Further analysis of the contents of this cluster revealed that one particular memory on all devices in the cluster had failed, and that there was no other common fail among the devices in the cluster. At the time of this writing, no failure analysis had been performed yet to confirm the diagnosis, but the signature is so strong, that no other explanation is likely.

The second example is that of a defect based cluster. The design was another large ASIC. The reticle in this case had size 2 by 2, with three A chips and one B chip (different versions of the same design.) Almost all A chips failed. Commonality analysis using latches (see Table 4.,) used to cluster the failing devices, and three medium sized clusters were found, one of which is shown in Figure 23 Further diagnosis, indicated a unique failing location in the design, which then, after failure analysis was traced back to a physical design problem (misaligned via.)

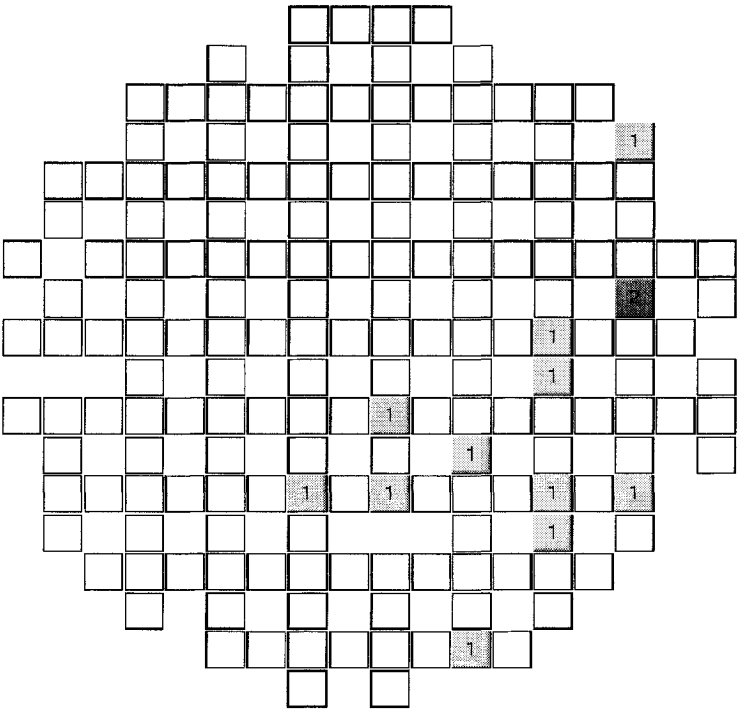


Figure 23 Defect based cluster

Chapter 7

Spatial Patterns

Many defects are randomly distributed over the wafer surface, and cause the die to fail at random locations. Some defects, however, have spatial distributions that can be used for their identification.

One example of such defects are particles left on the wafer after an incomplete rinsing, but left preferentially downstream of the direction of the rinse. Another example is bad or incomplete polishing caused by sagging or bowing of the wafer, which causes the local plane of the wafer not to be parallel to the plane of the polishing pad, and may cause the outer regions of the wafer to yield less than the central one. A final example is a mask defect in one of the die images in a reticle, which produce a clearly recognizable repetitive pattern of fails of those sites that correspond to the defective die image in the mask.

Such defects can be recognized by the distribution of failing die over the wafer not being random, but, instead, having some recognizable spatial pattern. The fact of a pattern of fails not being random may, of course, be recognized without identifying an underlying defect. A repetitive pattern whose pitch matches that of the reticle, for example, is a clear indication of a mask defect, even though the defect itself has not yet been identified.

In this chapter, I will focus on the recognition of spatially non-random patterns of fails. Once recognized, those die that seem to be part of the pattern can be subjected to other forms of diagnosis to identify the actual defect mechanism. This second step, however, the actual defect identification, will not be part of this chapter.

Before getting into the details of pattern recognition, two questions need to be answered. First, what wafer maps will be analyzed ? Maps showing the passing and failing die is one obvious example. Wafer maps need not be limited to just final sort codes, however. It is often also useful to analyze maps of passing and failing embedded objects, the ones described in Chapter 5. The main reason for being interested in wafer maps of such objects is that they show the die in much more detail than sort codes do, and that different objects may have different sensitivities to those systematic defects that give rise to spatial patterns. I will focus on wafer maps resulting from embedded object tests, with the understanding that a similar analysis can easily be applied to other types of wafer maps.

The second question is which passes and fails to use. For example, in the case of instance fails, not all passes will be known in general, for the fail data contain information only for those die on which at least one embedded object

failed. A die for which there are no fail data for any of a certain type of embedded objects may either have passed all the relevant object tests, or have failed some test preceding them. Consulting the final sort codes allows us to distinguish these two cases, but the pass/fail status of the objects is inherently unknown for those die to which the object tests were not applied.

Some information about unknown passes can often be retrieved from the fail data, as a fail of an object of some type on some die shows that other objects of the same type on the same die were tested with the object tests for that type. Consequently, we can assume that those other objects passed the tests if no fail data were collected for them. This assignment of a pass to those other objects is made more complex when the objects are tested sequentially, rather than in parallel, but passes can still be assigned in many cases following the general strategy described in Chapter 5.5.

We will assume that pass information has been determined as far as is possible from the available fail data, but that it can still be incomplete. That not all pass information may be available need not be an impediment to spatial pattern analysis. Some patterns will be missed, however. For example, assume that the die that would fail the object tests, if applied, would have some distinctive pattern, but that the die that were thus tested do not. The distinctive pattern would be missed. This is a consequence of not always applying the object tests, and cannot be mitigated by more sophisticated spatial analyses. On the other hand, the die that were not even tested with the object tests may now have some recognizable spatial pattern on their own that can be used for defect identification.

Once a - potentially partial - map of passes and fails is available, we can proceed systematically to recognize and classify non-random patters. The recognition phase comes first, and is intended to establish that a particular pattern is non-random. This phase is useful, as the classification of a non-random pattern may be time consuming. The recognition phase functions as a quick and inexpensive screen that removes wafer maps that do no seem to have any distinguishable pattern. This phase will be discussed in the first two sections of this chapter. The second phase will be discussed in the remaining sections, and is the classification of non-random patterns.

1 NON-RANDOM PATTERNS

The first task at hand is to recognize wafers on which the pattern of fails is not completely random. I will discuss several different approaches to this problem. The method to be used in this book is based on the Spatial Log Odds

Ratio (SLOR [56, 14]). A detailed study of a related method is presented in [26].

1.1 Clustering parameter

A sophisticated approach to identifying non-random patterns is to assume that the pattern is governed by some underlying distribution that differs from the Poisson one, and whose parameters can be estimated using standard statistical estimation techniques. This underlying distribution should be flexible enough to contain the random one as a special case, in which case some measure of statistical significance can be used to decide if the estimated parameters are sufficiently distinct from the random ones to allow us to label the pattern of fails non-random.

Such a measure was explored in detail in Chapter 4, where we studied the behavior of γ, the inverse of the cluster coefficient. This measure was obtained for a large number of wafers, and we saw that it could differ significantly from the same measure obtained for a known Poisson distribution of defects. γ seems to be particularly attractive, for its defining equation, Equation (D.7), was shown in Appendix D to be approximately valid for a large range of distributions, not just the negative binomial one. On the other hand, γ does not exist when Y(2A) equals zero.

1.2 Geometric properties of the pattern

Another approach to identifying non-randomness is to use geometric aspects of the pattern of passes and fails. Because any specific pattern of fails can be produced by a random defect producer, and because any particular pattern is as likely as any other pattern, the pattern by itself cannot show itself to be non-random. Instead, some geometric property of the pattern has to be identified whose possible values have different probabilities, even when the defects are produced at random. Non-random patterns are then indicated when the likelihood of the selected property for the wafer pattern at hand is, in some sense, low.

1.2.1 Geometric centers

The simplest property available for the die represented in the fail data is their geometric center $\mathcal{X}_T$. This center is a point on the wafer whose coordinates are $\sum_i x_i / \sum_i 1$ and $\sum_i y_i / \sum_i 1$, in which x_i and y_i are the coordinates of the i^{th} tested die, and the sums are taken over the tested die. When the fail data contain information only for the die tested with some part

of the test sequence, and not for all die, $\mathcal{X}_T$ may differ from $\mathcal{X}_W$, the geometric center of the wafer. The latter is defined similarly to $\mathcal{X}_T$, but with its coordinates such that the sums are over all the die on the wafer, not just over all the tested die.

The geometric center of the failing die is of separate interest, and is indicated by $\mathcal{X}_F$. It is defined similarly to $\mathcal{X}_T$, but with the sums in the definitions of the coordinates over all the failing die rather than over all the tested ones.

An example of an otherwise unremarkable pattern is shown in Figure 24,

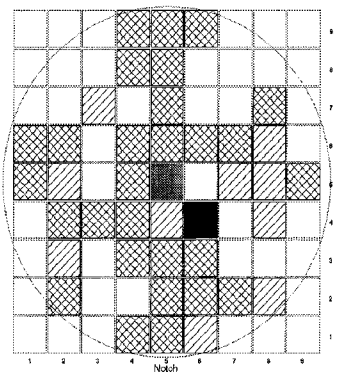


Figure 24 Geometric centers

with the positions of $\mathcal{X}_T$ and $\mathcal{X}_F$ indicated approximately, that is, after having been rounded off to the closest die position. The row and column numbers are shown on the left and at the bottom, respectively. The singly hashed sites correspond to die that failed, while the doubly hashed ones correspond to passing die. $\mathcal{X}_T$ is indicated by the black site, and $\mathcal{X}_F$ by the grey one. No attempt was made to obtain $\mathcal{X}_W$.

If the failing die are distributed randomly among the tested ones, $\mathcal{X}_F$ is likely to be close to $\mathcal{X}_T$, while large differences indicate that the failing die are not so distributed. Differences between these two can therefore be used to gauge how non-random a particular pattern is.

Unfortunately, the statistical distribution of the center coordinates, be that of $\mathcal{X}_F$ or of $\mathcal{X}_T$, is too complicated to be useful, and likelihoods can be calculated only with great effort. The relative positions of $\mathcal{X}_F$ or of $\mathcal{X}_T$ are also not enough to discriminate many obvious non-random patterns from random ones, and the ones that are recognized can also be identified as easily by other means. The relative positions will therefore not be used, but the two geometric centers will in other contexts.

1.2.2 SLOR

The measure that seems to be generally useful, and that has been studied extensively in the literature [12], is based on the numbers of neighboring pass-pass, pass-fail, and fail-fail pairs. This measure has been studied in detail in [56, 14], where it is called SLOR. The rationale for this measure is that a non-random pattern requires some correlation between different die on the same wafer, and that this correlation changes the numbers of pass-pass, pass-fail and fail-fail pairs. The actual form of the measure is then obtained by making specific assumptions on what die can be correlated, and how.

Let us indicate the event of a pass or fail of die i by the random variable X_i, with X_i being 1 when die i passes, and 0 otherwise. The event of die i and j both passing is then X_iX_j, that of both die failing is $(1-X_i)(1-X_j)$, and that of one passing and the other one failing is $(1-X_i)X_j + X_i(1-X_j)$. Assuming that the probability of a die passing is p, and that it is the same for all die on the wafer, the expectation values of the latter three events are p^2, $(1-p)^2$ and $2p(1-p)$, respectively, when the fates of two different die are independent.

When there are dependencies, however, these expectation values may change. In the simplified SLOR model, each die has a well defined neighborhood, and two die are correlated only when they are neighbors. In this chapter, two die are neighbors when one is immediately to the north, to the east, to the south or to the west of the other.

For two neighboring die, the expectation values become η, $1-2p+\eta$, and $2(p-\eta)$, respectively, in which η is the expectation value of X_iX_j, and i and j are neighboring die. When two die are neighbors, the odds of one particular one passing, given that the other one has passed equals $\eta/(p-\eta)$, while the odds of that same die passing, given that the second one has failed equals $(p-\eta)/(1-2p+\eta)$. If the fates of the two die were not correlated, the two odds would be equal. The ratio of the two odds, therefore, seems to be a reasonable measure of randomness. It equals

$$\frac{\eta(1-2p+\eta)}{(p-\eta)^2} = 1 + \frac{\eta - p^2}{(p-\eta)^2}. \tag{7.1}$$

The second form shows that the ratio equals 1 when η equals p^2, that is, when there is no correlation between the passes and fails of the two die.

This ratio, therefore, is some number between zero and infinity, and is equal to one when there is no correlation. A more convenient measure is the logarithm of the ratio, since that leads to a number between minus and plus infinity, and equal to 0 in the absence of correlations. This logarithm is called the Spatial Log Odds Ratio, or SLOR.

To estimate it, we simply count the numbers of neighboring pairs of different types. Let the number of pairs be J, the number of pairs with two passing die J_{++}, the number of pairs with two failing die J_{--} and the number of pairs with one failing and one passing die J_{+-}. The expectation values of these numbers are $J\eta$, $J(1-2p+\eta)$, and $2J(p-\eta)$, respectively. As in the quadrat method (Chapter 4.3.2), there may be come uncertainty on which pairs to use. As in that chapter, I use here all possible pairs.

The obvious estimator of the SLOR is then

$$\ln\left(\frac{J_{++}J_{--}}{(J_{+-}/2)^2}\right). \tag{7.2}$$

There are some possible anomalies, for example when $J_{++} = 0$, or when $J_{+-} = 0$. Such anomalies can be handled easily by replacing the estimator by a suitably chosen large positive or negative value.

In the case of a Poisson distribution of defects, the SLOR is expected to be close to zero. This can be verified using the same set of Poisson wafers used in Chapter 4. Figure 25 shows the correlation between γ (see also Figure 12) and the SLOR. The latter values are gratifyingly small, and there is an equally gratifying correlation between them and the corresponding γ values.

In Chapter 4, it was shown that γ has a far larger spread in values on some set of real wafers than one would expect if the defects on those wafers had been randomly distributed. Those results are shown again in Figure 26, where they are compared with the corresponding SLOR values. All data points with anomalous SLOR values (values corresponding to plus or minus infinity) were removed. These include all the wafers for with Y(2A) = 0, and several of the wafers with Y(4A) = 0.

There is again a pleasing correlation between the two, and the SLOR values have a much larger spread than in Figure 25, confirming the conclusions that there is a fair amount of non-randomness. The group of data on the left of the chart, at small values of γ, and labeled separately, correspond to the case that Y(4A) equals zero. The data in this chart will be analyzed in much more detail later in this chapter.

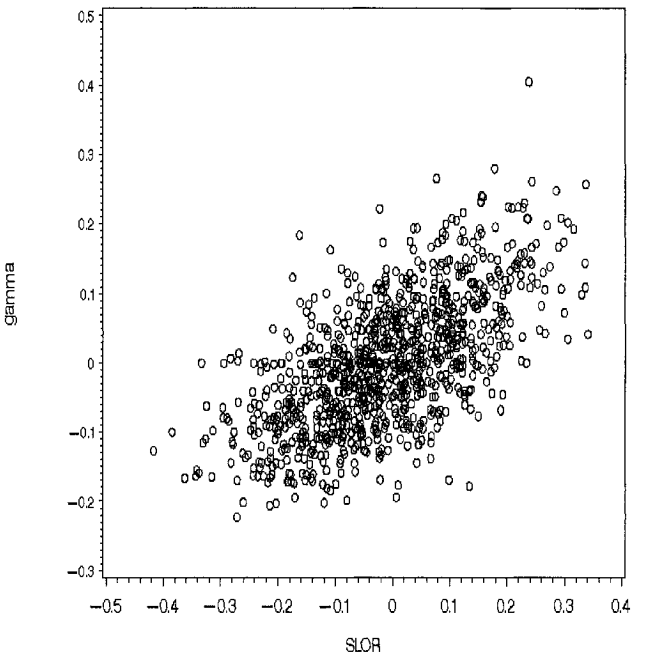


Figure 25 Correlation between SLOR and γ
Poisson defect distribution

2 CLASSIFYING PATTERNS

The measures discussed so far are intended to gauge the degree of non-randomness of a given pattern. They give no information, however, on the shape of a pattern found to be non-random. Recognizing such spatial patterns has been the focus of a recent Sematech study [33]. In this book, I will discuss two simplified approaches that are still reasonably effective in recognizing and classifying non-random patterns.

One group of techniques that can identify various types of non-randomness will be discussed in the present section. They all start from the set of fail probabilities in suitably chosen partitions of the wafer. The particular partitions used here are columns and rows on the wafer, or circular segments around $\mathcal{X}_T$, the center of the known die, and angular sectors emanating from that center.

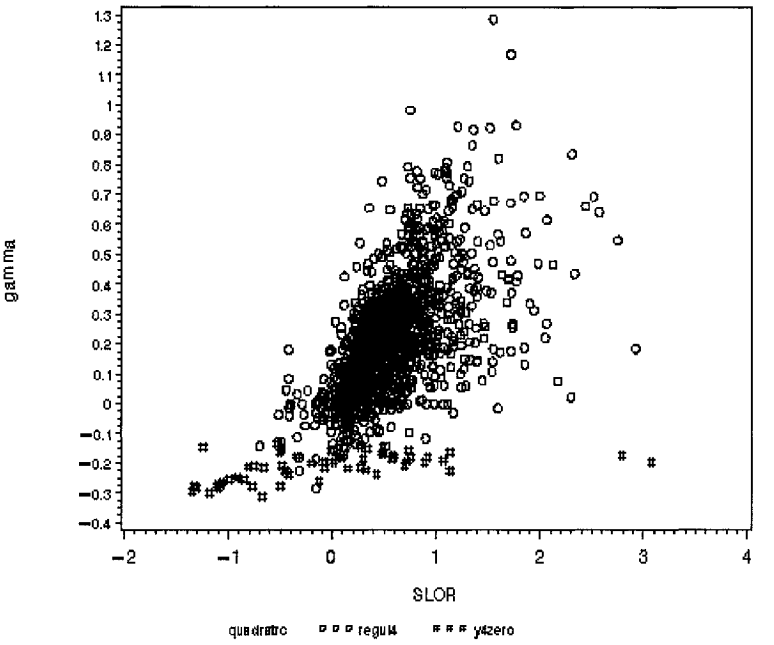


Figure 26 Correlation between SLOR and γ; Real wafer

2.1 Marginal probabilities

A wafer can be divided into several large areas, on the order of ten or so, like the rows or columns of the wafer. There are many ways the wafer can be divided into small regions. Each way will be generically called a partition. The partitions considered here are rows, columns, circular segments and angular sectors. There is very little flexibility in the regions when rows or columns are chosen, but more so in the case of circular and angular regions. I will always use twelve angular sectors, corresponding to triangles with 30^0 degree angles at their apices, and at most ten circular segments.

Let us indicate the regions of a given partition by the subscript i, and let there be I such regions. In each region, some die will pass and some will fail. The expected proportion of passing die is the expected yield, but will be called yield for short. If the passes and fails were truly random, the yield within any region would be more or less that of the wafer as a whole; if not, each region might have its own yield.Let us indicate the yield specific to region i by p_i, and that for the wafer as a whole by p. Likewise, let the number of die in region i be indicated by n_i, and the number of failing die by N_i.

Finally let n and N be the numbers of die and of failing die on the wafer, respectively.

The type of fail pattern can now be identified by determining for which partition the regional yields differ most from the overall yield. A partition is called implicated when it is thus identified. If no partition can be implicated, we have an alternative indication that the pattern is random. Ideally, exactly one type of region is implicated. Fail patterns that can thus be recognized include rings, left half versus right half, horizontal or vertical stripes, and repeaters. Multiple partitions may also be implicated, however. One often occurring example is a partial ring where both angular sectors and circular rings are implicated.

Distinguishing "more or less equal" from "clearly distinct" can be done using the likelihood ratio, described in Chapter 2.2.2. The maximum likelihood estimators of p_i is N_i/n_i, and the resulting value of the likelihood function at its maximum equals

$$\prod_i \hat{p}_i^{N_i} (1 - \hat{p}_i)^{n_i - N_i} . \tag{7.3}$$

Likewise, the maximum likelihood estimator of p is N/n, while the corresponding likelihood is obtained by replacing all $\hat{p}_i$ in Equation (7.3) by $\hat{p}$.

The measure of randomness then becomes the likelihood ratio

$$\Lambda = \frac{N^N (n-N)^{n-N}}{\prod_i N_i^{N_i} (n_i - N_i)^{n_i - N_i}} . \tag{7.4}$$

A small value of Λ, or, alternatively, a large value of $-\ln\Lambda$ can be used as an indicator of non-randomness. To find the threshold separating random from non-random patterns, we use the results presented in Chapter 2.2.2. A threshold of a small number of standard deviations beyond the expected value of $-\ln\Lambda$, therefore, seems appropriate. In the present experiment, a value of 5 was used for ρ.

2.2 Experimental results

All the wafers used to generate Figure 26 were analyzed for specific patterns, using the flow diagram shown in Figure 27. First, a number of trivial checks are performed on the data to make sure that they are usable. In particu-

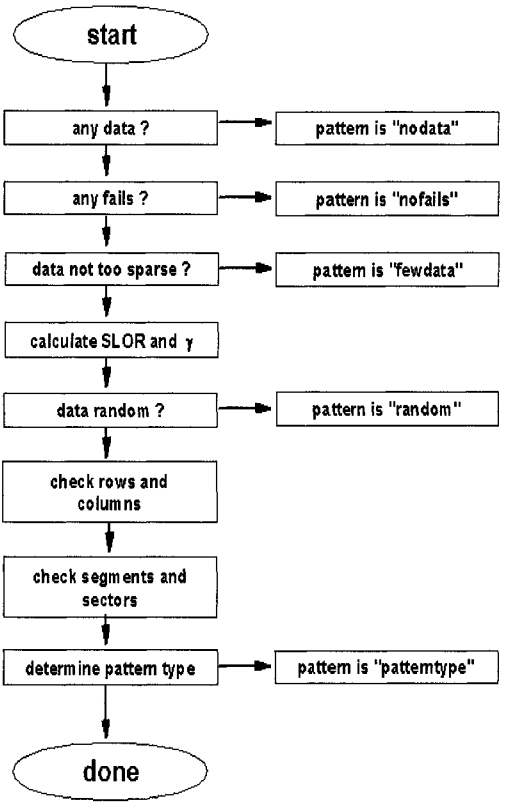


Figure 27 First wafer pattern analysis flow

lar, when not the full wafer, but only some chips on the wafer are available, there is a risk that the data is so sparse that no statistically meaningful analysis can be done. Sparseness is a vague concept. The heuristic definition used here is that the data is sparse when there are too few rows and columns having sufficiently many chips for which pass/fail information is available. Too few rows and columns again is ill-defined, but is set to less than three in this experiment. Likewise, sufficiently many chips means at least three chips.

After the data has been validated, the SLOR and γ are calculated, as described previously in this chapter and in Chapter 4.3.2.2. A wafer is then labeled random when the absolute value of γ is less then 2.5 and that of the SLOR less then 0.7. Further pattern analysis is done only on wafers that are not declared random. These threshold are somewhat arbitrary, and better screens can easily be created. The purpose of this screen, however, is to

remove all patterns that seem to be unlikely to be anything else than random, and the present, although crude screen meets that purpose.

The actual pattern analysis is, by necessity, complex, and will not be described here in detail. It needs to be flexible, to be able to handle all kinds of spatial patterns, and it will inevitably evolve in time, for new, meaningful patterns that need to be recognized will occasionally be identified.

I will briefly describe one particular algorithm here. The analysis starts with the statistical calculations outlined in Section 2.1 for various partitions. Once individual fail probabilities for the regions of a particular partition have been determined, they are compared with the overall yield of the wafer using Equation (7.4). If the former differ significantly from the latter, the low yielding regions of are marked, and that partition is implicated. There are now three different possibilities. All the marked regions bunch together in one superregion, several disconnected marked regions can be identified, or the marked regions occur at regular distances from each other. The latter possibility is considered only when the regions are rows or columns, because it may indicate a repeater, that is, a problem with the mask.

The loglikelihood ratios measure the degree to which the partitions show the deviation of the pattern from a random one. It is of course possible that none of the loglikelihood ratios shows a significant difference from a random pattern, that is, that no partition is implicated. This may mean that the pattern is not random after all, or that the algorithm is not looking at the pattern in the right way. As there is no way of proceeding at this point, a verdict of no_pattern is returned.

If at least one partition is implicated, the one with the largest loglikelihood ratio is assumed to describe the pattern best. The results are shown in Figure 28. The square region corresponding to the random patterns is clearly visible. Examples of each of the pattern types shown in Figure 28 are presented in Figure 29, in which failing die are indicated by the black squares, and the passing ones by the light grey ones.

Ring patterns are very common. These are patterns in which the center of the wafer is relatively high yielding, while the yields in different circular regions differ significantly from the overall wafer yield. If there is no significant angular effect in the yield, the ring is called complete; otherwise, it is called partial or fragmented. Each fragment may consist of several, neighboring sectors. A partial ring is a ring fragment with only one fragment.

If the angular sectors show the strongest deviation from randomness, the pattern is supposed to have one or more bad sectors. If there is also a radial effect, the pattern is classified as a partial or fragmented ring; otherwise as one or many bad sectors.

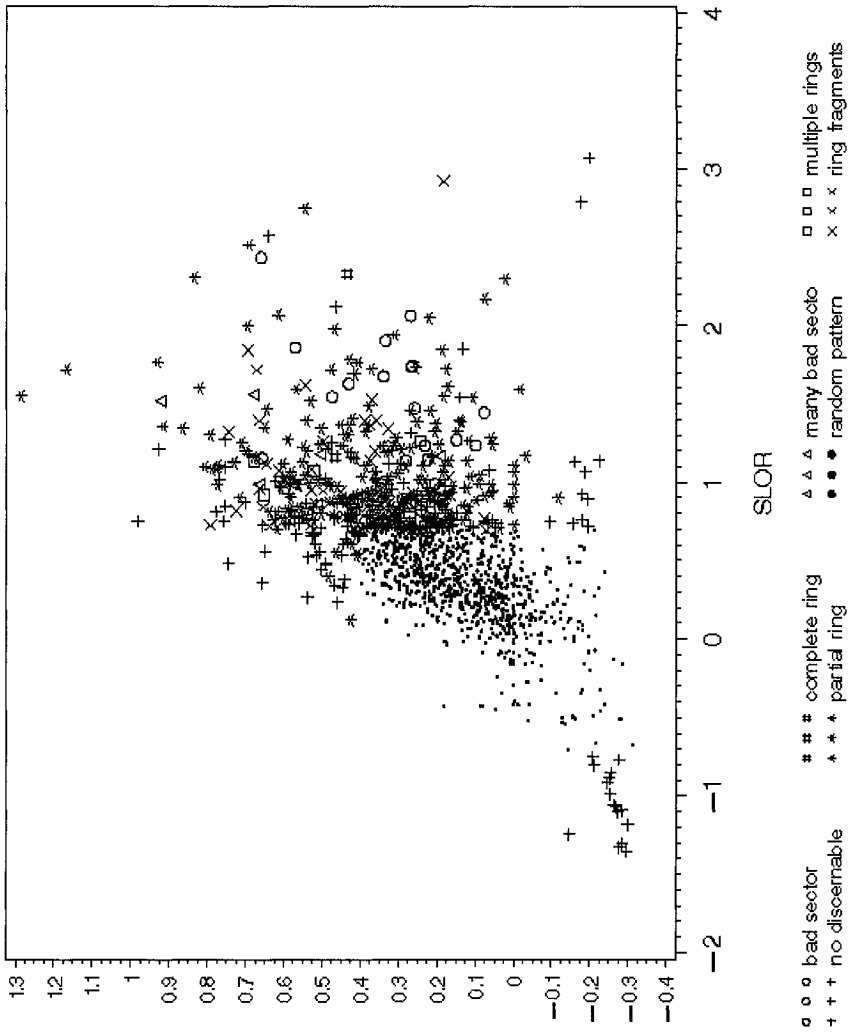
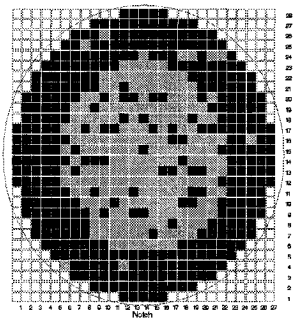
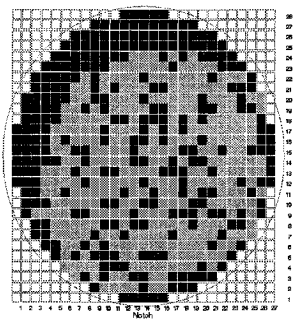


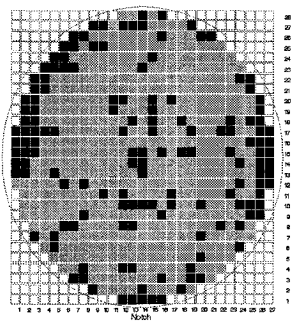
Figure 28 Correlation between SLOR and γ labeled by the wafer pass/fail pattern



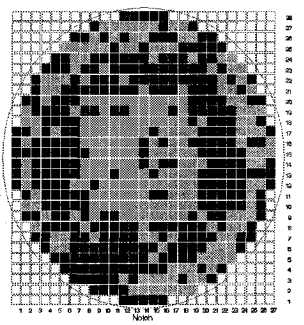
a: complete ring



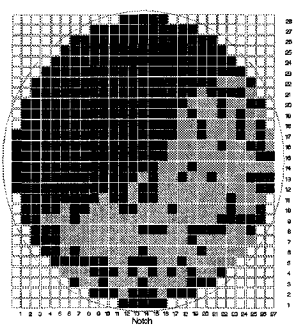
b: partial ring



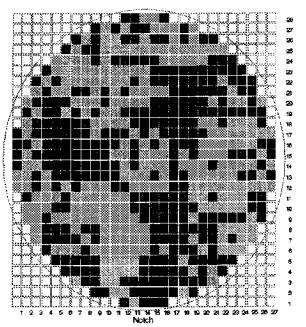
c: ring fragments



b: multiple rings



c: bad sector



d: many bad sectors

Figure 29 Several examples of pass/fail wafer patterns

A large group of patterns is classified as "no discernible pattern", which corresponds to the no-pattern label introduced above. The classification algorithms used here is not strong enough to recognize such patterns. More sophisticated algorithms could be devised to properly classify them, but a more profitable approach will be described in the next section.

2.3 Clustering patterns

The pattern recognition strategy outlined in the previous section has the advantage of being fairly straightforward, and able to recognize many common patterns. Its disadvantage is that it is designed to recognize only a small number of specific patterns. In other words, it may be blind towards important, but non-standard patterns that are now labeled as "no discernible pattern". The only way to enrich the spectrum of patterns that can be recognized is by explicitly writing new algorithms for finding those additional patterns. This limitation will obviously not be addressed by using large numbers of training sets to a spatial pattern analyzers, ad was done in the Sematech study [33].

The goal of recognizing spatial patterns, however, is to find important systematic yield detractors, and those would manifest themselves on multiple wafers, not just on a singe one. In other words, non-random patterns that are caused by some systematic problem that occurs on only one wafer are interesting, but, within the context of yield learning, not important. If the goal is to recognize non-random patterns that do not occur just once, but many times, other strategies for identifying such patterns can be followed.

A very different strategy from the one described in the previous section is cluster analysis. In such an analysis, all spatial patterns are compared, and clustered into groups, such that all the wafers in a group show more or less the same pattern of passes and fails. Large groups then indicate some systematic problem, even though the spatial pattern may not have any particular, easily recognizable features. This approach is the same as was followed in Chapter 6.

The signature of a wafer pattern is the vector of pass/fail statuses of the die on the wafer, for example 1 for a pass and 0 for a fail. In the terminology of Chapter 6, the label n is the xy coordinate of a given die, and the value v is the pass/fail status of that die. The similarity measure is (see also Equation (6.4))

$$\frac{\sum_i v^I{}_i v^J{}_i}{\sum_i (v^I{}_i + v^J{}_i - v^I{}_i v^J{}_i)}, \tag{7.5}$$

in which I and J refer to the signatures of the two wafers and v_j^I is the pass/fail status of die j in signature I. This measure equals 1 when the two signatures are identical, and is less than 1 otherwise.

If the fail data is incomplete, and not all die on one wafer have a counterpart on the other wafer, there is no corresponding (n,v) pair in the signature of the wafer. Such absentee pairs are treated different here than in Chapter 6, because their absence indicates that the pass/fail status of the die is not known, rather than that it is 0. The sum in Equation (7.5), therefore, is taken over all die that have known pass/fail statuses on both wafers.

Once the similarity measures have been obtained for all pairs of wafer patterns, the latter can be clustered, using the algorithm described in Chapter 6.4. Some results are shown in Figure 30. This figure shows two clusters, and the fail patterns of some of the wafers that make up those clusters. The cluster maps were produced by averaging over the fail maps of the individual wafers in the cluster. A black square means that the chips at that position on the wafers in the cluster were considerably more likely to fail than on average, while the light grey squares indicate that the corresponding chips were considerably less so. There is a separate category for chip positions where fail probability of the corresponding chips is more or less equal to the average fail probability over all the wafers in the cluster, but neither of the cluster maps in Figure 30 use that category.

All the individual wafer maps were classified as "no discernible pattern". The power of clustering is that very clear patterns can emerge nevertheless. The first cluster shows the spokes that were also seen in Figure 29, which is the wafer in the cluster that was classified as "many bad sectors". The second cluster in Figure 30 has a less obvious, but still clear pattern. The patterns from two of the wafers in the cluster are shown in frames c and d. The last one in particular is barely distinguishable from a random pattern, but turns out to have a systematic problem anyhow.

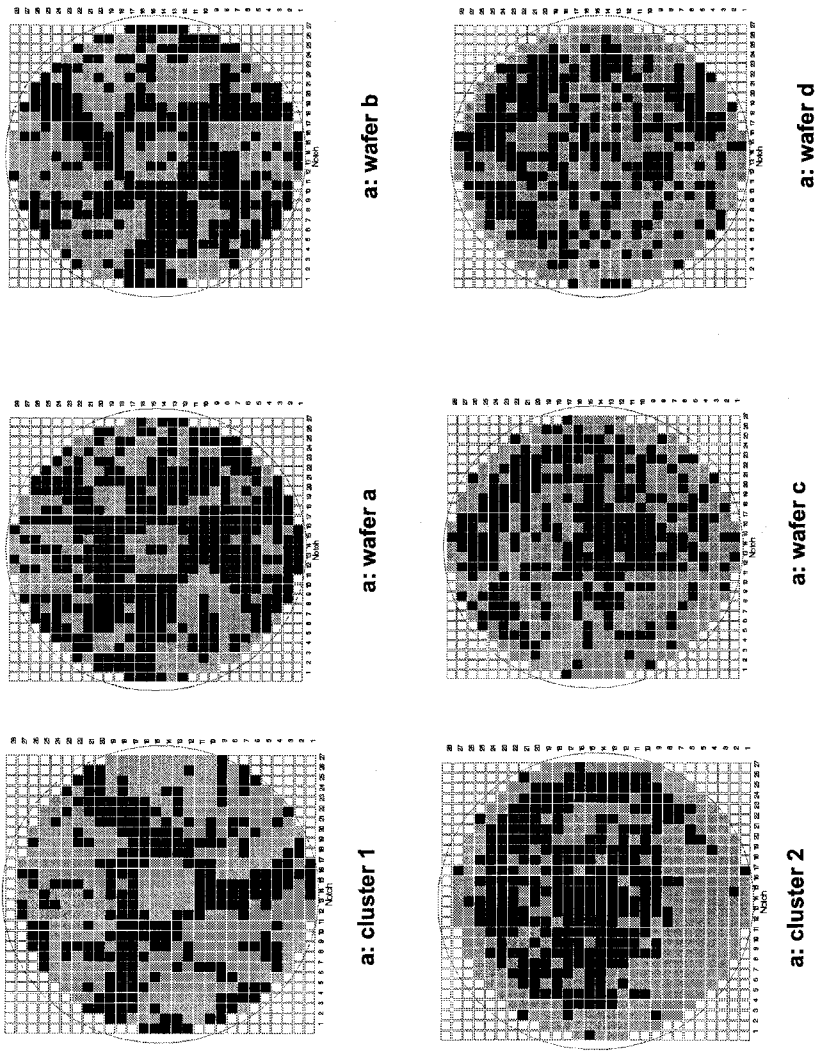


Figure 30 Cluster fail maps

Chapter 8

Test Coverage and Test Fallout

In Chapter 3, generally valid statistical properties of fallout and yield were discussed. If we want to relate these quantities to physical defects, some relation has to be found between defects and fallout.

Fallout information can be studied in more detail for tests that target the internal logic of the Integrated Circuits, the logic gates and memory elements, and their connections, because of the large number of patterns that make up the tests. Examples of such tests are scan based tests, like LSSD and LBIST. They will be referred to as structural tests.

The starting point of this chapter is the set of devices that passed all gross tests, the ones preceding the structural tests. The fallout statistics during these tests was discussed in Chapter 3, and in particular in its Appendix A. It is related to the fallout statistics over all the tests through a simple rescaling of the multinomial parameters by the yield at the completion of the gross tests. In this chapter, the rescaled parameters will be indicated by d_i, and the corresponding yields by y_i, and we will assume that K devices have passed the tests preceding the structural ones.

In this chapter, I will assume that the fallout has been recorded at some level of granularity. The level of granularity is typically not that of individual patterns, but more likely the level of groups of such patterns. Patterns in a single procedure all have the same clocking sequence, and their number is typically on the order of 30.

For structural tests there is a notion of coverage, and this coverage can be related to the fallout. In this chapter, a general relationship between yield and defect coverage will be derived. This relationship will then be used to obtain information about the distribution of the number of defects on the devices.

1 YIELD AND COVERAGE

The first step in the detailed fallout analysis is to develop a model of the relationship between defect coverage and fallout. It requires a better understanding of the distribution of the number of defects on a device, and how the presence of multiple defects affect the yield.

1.1 Defect model

The defect model to be used here has F defects that can be present on a chip, with F very large. Each chip has either no defects, or contains a subset of them. This subset is generally small and will be indicated by S. S is different for different chips, and indicates the complete set of defects on a given chip. It can be empty, indicating a good chip, it can have one member, or it can have several members. In the latter case, we say that the chip has a multi-defect. |S| is the number of single defects in S, and indicates its size.

Before the start of diagnosis, nothing is known about the defects. I will assume, however, that there are well defined probabilities p_S that a randomly chosen chip will have (multi-)defect S. When the single defects are independent, p_S is the product of the occurrence probabilities of its members. As the defects in general are not independent, however, I will not assume that p_S has that form when S has more than one member.

It is reasonable to assume that such p_S exist for a mature process, that is, once the manufacturing process parameters have become stable. The p_S are unknown, and, as it turns out, it is far more convenient to use certain combinations of them instead. These combinations are the probabilities

$$P_n = \sum_{|S| = n} p_S \tag{8.1}$$

that a randomly chosen chip will have a defect of size n. These are therefore also the only quantities pertaining to the defect distribution that can be obtained from the yield data.

The P_n depend on the defect model and on the distribution of defects over the chips. Because they are probabilities, $P_n \geq 0$, and $\sum P_n = 1$. The average number of defects per chip will be indicated by

$$\mu = \sum n P_n, \tag{8.2}$$

and the variance in the number of defects per chip by

$$\sigma^2(n) = \sum_{n = 0} n^2 P_n - \mu^2 . \tag{8.3}$$

Finally, as the maximum reachable yield is equal to the probability that no defects are present, y_0 equals P_0.

Using such an arbitrary set of P_n makes it possible to handle arbitrary degrees of spatial clustering, and arbitrary dependencies between defects. It provides for the most detailed analysis of the spatial distribution of defects available, one that completely includes all clustering effects. It needs to be noted, however, that spatial clustering is not the only way to get a distribution of numbers of defects per chip that differs from the standard Poisson one. One example is given in Section 1.1.3.

1.1.1 Poisson and negative binomial models

The general model includes many used in the literature. For example, in the negative binomial distribution discussed in Chapter 2.1.3,

$$P_n = \frac{\Gamma(\alpha + n)}{n!\Gamma(\alpha)} \left(\frac{\mu}{\alpha}\right)^n \left(1 + \frac{\mu}{\alpha}\right)^{-\alpha - n} . \tag{8.4}$$

The case of no spatial clustering, that is, of independent defects, corresponds to α going to infinity. In that case,

$$P_n = \frac{\mu^n}{n!} e^{-\mu} . \tag{8.5}$$

In the case of no clustering, $\mu = \sigma^2(n)$. The difference between the first and second moments of the P_n, as estimated from the yield data, therefore gives an indication of the actual degree of clustering. To be consistent with the literature [17], it is better to estimate $(\sigma^2(n) - \mu)/\mu^2$, which equals $1/\alpha$ in the negative binomial distribution.

1.1.2 Compound model

A more general way of incorporating clustering is to assume that on a single device the number of defects is distributed according to Equation (8.5), but with a parameter ν that varies from chip to chip. Such a compound model suffers from the same problems as the one in Chapter 3.2: it merely reproduces observed P_n values. It can be made useful if the compounding distribution of ν can be related to device process histories.

The compound Poisson model was described in Chapter 2.1.2. Compounding always increases the variance of the observed yields, in the same way that compounding did in Chapter 3. The clustering is called weak when

the cluster parameter is large, which occurs when $\sigma^2(v)$ is small, that is, when the compounding function is concentrated narrowly around its mean μ. The Poisson factor in Equation (2.6) can then be expanded in a Taylor series around $\mu = v$, resulting in

$$P_n \approx \frac{\mu^n}{n!} e^{-\mu} \left(1 + \frac{\sigma^2(v)}{2\mu^2}((n-\mu)^2 - n)\right). \tag{8.6}$$

Notice that, compared to the pure Poisson distribution, P_n is depressed when n is roughly equal to μ, and elevated otherwise (to be precise, when $\mu + 1/2 \pm \sqrt{\mu + 1/4}$).

Another effect of compounding is to increase the probability of finding a device with no defects at all, as was demonstrated in Chapter 2.1.2. This probability was referred to as y_0 in Chapter 3, and equals

$$P_0 = \int h(v)\, e^{-v}\, dv. \tag{8.7}$$

In the weak clustering limit,

$$P_0 = e^{-\mu} \left(1 + \frac{1}{2}\sigma^2(v)\right), \tag{8.8}$$

which is clearly larger than the Poisson value $e^{-\mu}$.

1.1.3 Independent defect model

A very different model from the previous one is obtained when the defects are independent, but have different occurrence probabilities [52]. No simple equations exist for the y(c), but this model is important anyhow, because it shows that spatial clustering is not the only way to get deviations from the simple Poisson model.

The probability of a complex defect S occurring is equal to the probability of all the individual defects in S occurring and no others. In other words, in the independent defect model,

$$p_S = \prod_{j \notin S}(1 - \pi_j) \prod_{i \in S} \pi_i = P_0 \prod_{i \in S} \frac{\pi_i}{1 - \pi_i}, \tag{8.9}$$

in which π_i is the occurrence probability of defect i, and

$$P_0 = \prod_{1 \le i \le F} (1 - \pi_i). \tag{8.10}$$

Consequently,

$$P_n = P_0 \sum_{|S| = n} \prod_{i \in S} \frac{\pi_i}{1 - \pi_i}, \tag{8.11}$$

which obviously can take on values that are very different from those in Equation (8.5).

These results become particularly simple when all the occurrence probabilities are the same [62], because then

$$P_0 = (1 - \pi_i)^F, \tag{8.12}$$

and

$$P_n = \binom{F}{n} (1 - \pi_i)^{F-n} \pi_i^n \tag{8.13}$$

1.2 Coverage and yield

The equations presented above are very general. They assume only the existence of some well defined defect model that contains all the defects that can occur on the chips, and the existence of p_S. The yield clearly depends on p_S, but not exclusively so. It depends also on the tests that are being applied.

Rather than label the tests by some index k, it is customary to label them by a more meaningful parameter, like the coverage c. In order to make c well defined, we assume that each one of the F defects is definitely detected or definitely not detected by the test sequence. Determining whether or not the defect is detected may be extremely impractical, but we will assume that it can be done.

The coverage c is then defined as the fraction of defects covered by the test sequence. To be precise, c_k is the fraction of defects detected by at least one test between 1 and k, and $y(c_k) = y_k$ is the yield, now written as a function of c_k.

The most general relationship between the yield, the test coverage and the probability of having a particular size defect on the chip is

$$y(c_k) = \sum_{n=0} P_n Q_{n;k}, \tag{8.14}$$

where $Q_{n;k}$ is the conditional probability that the chip will pass tests 1 through k, given that there is a defect of size n on the chip. The upper limit in this sum is the total number of possible defects F. In practice, however, n is not very large and certainly much smaller than F.

Equation (8.14) is so general as to be meaningless. All the details about the test effectiveness are hidden in the conditional probabilities $Q_{n;k}$. These conditional probabilities depend on the coverage c_k, but in an as yet unknown fashion. To determine this relationship, some assumptions have to be made.

The first assumption that is usually made, and one that will be made here too, is that a multi-defect will be detected when any of its members is. What this means is that defects do not mask each other or unmask each other. Masking occurs when two defects are exposed by a pattern if they are present on a chip by themselves, but are not exposed when they occur together. Unmasking is the opposite of masking. It occurs when two single defects are not exposed by a given pattern but, when together, cooperate to produce a fault effect. Both masking and unmasking can occur, but are rare, and this assumption does not seem to be a severe one (see however [1]).

Secondly, we assume that the defects are in some sense normal, meaning that the yield is not determined by a small subset of very likely defects, but instead by a large number (of order F) of them, none of them being very likely by themselves. The specific criterion for normality is discussed in Appendix H. This is not a very strong assumption, as it almost never happens that a few defects dominate the yield. For if it did, redesign of either the logic or the manufacturing process would almost certainly have eliminated those defects.

Finally, we will assume that the occurrence probabilities of the defects are independent of whether or not they are detected. This assumption is necessary, because otherwise, for example, any large number of defects with zero occurrence probability, and therefore without relevance to testing or yield, could artificially raise the coverage if they were all tested by the test sequence (or, likewise, lower the coverage if none of them were tested). The independence assumption consists of two parts, as explained in Appendix H.

Another way of phrasing this assumption is that there is no correlation between the occurrence probability of a defect and its detection probability. For a particular test sequence, this assumption may not be valid, as a defect is detected or not, and the correlation coefficient has some value, usually differ-

ent from zero. The detection probability, however, is the probability that a randomly chosen test generation procedure generates a test pattern that exposes the defect. It is this probability that is assumed to be uncorrelated to the occurrence probability of the defect.

A randomly chosen test generation procedure is admittedly a somewhat vague concept, but can usually be defined in practice. For example when the test sequence is obtained using a standard Automatic Test Pattern Generation (ATPG) package, target defects for test generation could be picked at random from the defect list. A randomly selected sequence of defects then constitutes an randomly chosen test generation process.

As another example, when the test generation process consists of fault simulating random patterns until the defect coverage is c, then different sequences of random patterns constitute different runs of the test generation process. Defects that have very large detection probabilities will almost always be detected, while random pattern resistant defects will almost never be detected. These random pattern detection probabilities are then assumed not to be correlated with the occurrence probabilities.

Using these assumptions, it is shown in Appendix H that all test sequences with the same defect coverage have roughly the same yield. Different test sequences may have slightly different yields, but these differences are of order $1/\sqrt{F}$, and can, therefore, be ignored for very large designs.

To be precise, $Q_{n;k}$ is related to c_k by

$$Q_{n;k} = (1 - c_k)^n, \tag{8.15}$$

plus terms of order $1/\sqrt{F}$, and [50]

$$y(c_k) \approx \sum_{n=0} (1 - c_k)^n P_n. \tag{8.16}$$

From now on, yield is understood to be the average yield over all test sequences with the same coverage. The benefit of this averaging is that it simplifies the relationship between coverage and yield, while introducing only negligible errors.

Equations (8.15) and (8.16) are more surprising than it may seem. At the least, one would expect that $Q_{n;k}$ depends on the test generation method used to obtain the test sequence. As it turns out, however, it is independent of how the test sequence was obtained, as long as the coverage is c_k: *the relationship between defects, coverage and yield is the same, whether we use random pat-*

terns, minimized sets of deterministically generated patterns or sequences of functional code to test the chip.

Equation (8.16) also has another practical use, as it relates the yield y(c) to the characteristic function $\Phi(t)$ of P_n. $\Phi(t)$ is the expected value of e^{itN}, and

$$y(c) = \Phi(-i\ln(1-c)). \tag{8.17}$$

Consequently, if the distribution of the number of defects on a chip is known, calculating y(c) is reduced to looking up the corresponding characteristic function in a table. In particular, for the negative binomial distribution,

$$y(c) = \left(1 + c\,\frac{\mu}{\alpha}\right)^{-\alpha}, \tag{8.18}$$

and for the Poisson distribution

$$y(c) = e^{-c\mu}. \tag{8.19}$$

1.3 Properties of the yield curve

Let us now consider some of the information that can be obtained from Equation (8.16). First of all, y(c) is a smoothly varying function of c. Its first derivative is negative, at least for c between 0 and 1. This is obvious, for y(c) is the perceived yield and, per definition, cannot increase as the test progresses. Less obviously, the second derivative is everywhere positive. This indicates that y(c), even though it continues to decrease, will do so less and less rapidly.

Obviously, y(0) equals 1 because there is no fallout if no testing is done. Also, y(1) equals p_0, the probability of finding a chip with no defects on it. Various derivatives of y(c) with respect to c are related to other properties of the defect distribution. In particular,

$$\begin{aligned} \frac{d}{dc}y(c=0) &= -\mu \\ \frac{d}{dc}y(c=1) &= -P_1 \\ \frac{d^2}{dc^2}y(c=0) &= \sigma^2(n) - \mu(1-\mu) \end{aligned} \tag{8.20}$$

The first and third relations in Equation (8.20) make it possible to use the yield data to estimate the average number of defects per chip and the variance of that number. Those two data are particularly important for, in the Poisson model, $\sigma^2(n)$ equals μ. Therefore, if the defects are independently distributed, we should find that

$$\frac{d^2}{dc^2}y(c = 0) \approx \left(\frac{d}{dc}y(c = 0)\right)^2 . \tag{8.21}$$

Any significant deviation from this relationship then indicates a breakdown in the independent defect model.

The second relationship, $\frac{d}{dc}y(c = 1) = -P_1$, is important because it is related to the defect level DL (see also Chapter 3.3.) Let c_f be the coverage after the last test has been applied, and consider the case that it is close to 1. From Equations (3.3) and (3.4), $\langle DL \rangle = 1 - y_0/y_f$. If c_f is close to 1, Taylor series expansion of y_f around $c = 1$ gives

$$y_f \approx y_0 + (c_f - 1)\frac{d}{dc}y(c = 1) , \tag{8.22}$$

from which we find

$$\langle DL \rangle \approx \frac{(1 - c_f)P_1}{y_0} . \tag{8.23}$$

To estimate the defect level and its variance, we can attempt to find the ratio P_1/y_0 by fitting a equation with a small number of free parameters to the yield data (for example [18].) An alternative method is to approximate y_0 by Y_f, which can be done when the defect level is small, and then estimate P_1 directly using an independent defect analysis. A third method will be described in Section 2. All methods, however, will be influenced by statistical variations in the yield data, and no estimation method will succeed when the expected number of field failures is of order 1 or less, as explained at the end of Chapter 3.

Because of the importance of the negative binomial and Poisson distributions, it is useful to summarize the results of the previous section for these two cases, which is done in Table 6. This table also contains the compound model

	Negative Binomial	Poisson	Compound	Independent Faults
P_n	$\frac{\Gamma(\alpha+n)}{n!\Gamma(\alpha)}\left(\frac{\mu}{\alpha}\right)^n\left(1+\frac{\mu}{\alpha}\right)^{-\alpha-n}$	$\frac{\mu^n}{n!}e^{-\mu}$	$\int h(\nu)\frac{\nu^n}{n!}e^{-\nu}\,d\nu$	$P_0\sum_{\lvert S\rvert=n}\prod_{i\in S}\frac{\pi_i}{1-\pi_i}$
y(c)	$\left(1+c\,\frac{\mu}{\alpha}\right)^{-\alpha}$	$e^{-c\mu}$	$\int h(\nu)\,e^{-c\nu}\,d\nu$	
mean = expected n	μ	μ	$\mu \equiv \int h(\nu)\nu d\nu$	
variance $\sigma^2(n)$	$\mu+\frac{\mu^2}{\alpha}$	μ	$\mu+\sigma^2(\nu)$	
cluster parameter	α	∞	$\mu^2/\sigma^2(\nu)$	
$y_0 = P_0 = y(1)$	$\left(1+\frac{\mu}{\alpha}\right)^{-\alpha}$	$e^{-\mu}$	$\int h(\nu)\,e^{-\nu}\,d\nu \approx e^{-\mu}\left(1+\frac{1}{2}\sigma^2(\nu)\right)$	$\prod_{1\le i\le F}(1-\pi_i)$
$P_1 = -\frac{d}{dc}y(c=1)$	$\mu\left(1+\frac{\mu}{\alpha}\right)^{-\alpha-1}$	$\mu e^{-\mu}$	Apr..: $\mu e^{-\mu}\left(1+\sigma^2(\nu)\left(\frac{1}{2}-\frac{1}{\mu}\right)\right)$	$P_0\sum_i\frac{\pi_i}{1-\pi_i}$
<DL(c)>	$(1-c)\mu\left(1+\frac{\mu}{\alpha}\right)^{-1}$	$(1-c)\mu$	Apr..: $(1-c)\mu\left(1-\frac{\sigma^2(\nu)}{\mu}\right)$	$(1-c)\sum_i\frac{\pi_i}{1-\pi_i}$

*Table 6.*Comparison of Compound, Poisson and Negative Binomial models

results for further comparison, as well as some relevant quantities from the independent fault model.

2 OBSERVED YIELD CURVE

In addition to the information obtained using equations (8.20), all the fallout distribution parameters can be obtained from the overall y(c) curve using maximum likelihood estimation (see also Chapter 3.4.2 and [37].) Such estimation uses all the available fallout data, not just the ones near c = 0 or c = 1. It will be developed in this section, assuming that the coverages are the Negative Binomial ones, and both μ and α will be estimated. The Negative Binomial model is taken as an example, because it is close to, but more general than the Poisson model, and because it is very popular. Once the estimates are available, they can be used, in conjunction with the known final coverage c, to estimate DL using

$$DL(c) = (1-c)\mu\left(1+\frac{\mu}{\alpha}\right)^{-1}. \tag{8.24}$$

When K chips are tested with scan based tests 1 through f, some number, say N_i, will fail test i and N_{pass} will not fail at all. The distribution of N_{pass} and the N_i is multinomial, as in Equation (A.4). The yield at the completion of the i^{th} test is related to the multinomial parameters d_i by

$$y_i = 1 - \sum_{1 \le k \le i} d_k, \tag{8.25}$$

or, equivalently,

$$d_i = y_{i-1} - y_i. \tag{8.26}$$

y_i is a function of the coverage reached at the end of the test, and of the negative binomial parameters μ and α. Because of Equation (8.26), d_i is function of the coverage and these parameters as well.

As the yield is a function of μ and α, the latter two can be estimated from the observed fallout data taken as a function of the coverage. Their are several ways to obtain such estimates. One is the maximum likelihood method, and another one is regression. More details for the maximum likelihood method can be found in Appendix H. Both methods lead to multidimensional minimi-

zation (or maximization) problems, that have to be tackled with fairly standard numerical optimization routines. These calculations will not be addressed any further in this book.

Chapter 9

Logic Diagnosis

Logic diagnosis is the process of using fail data to deduce the location, and, if possible, the logic nature of the defect that caused a fail. The fail data used in logic diagnosis are collected when those tests are applied that most directly exercise the internal logic of the Integrated Circuit (IC). Typically, they are scan based tests (see, for example, [2], Chapter 9, [21], Chapter 1, and [16], Chapter 3). They can be deterministic tests, but they can also be randomly generated by a Built-In Self Test Engine.

The logic nature of a defect is its behavior during the application of any of those scan based tests. It is often useful to determine, or at least estimate this logical behavior, but the true purpose of logic diagnosis is determining the location of the defect. That information, after all, is needed by physical failure analysis to find the defect on the integrated circuit, observe it, and determine its physical nature.

In principal, logic diagnosis is straightforward (see Figure 31). The process starts from a logic design description of the Integrated Circuit. This is usually the same as what was used when generating the scan based tests. A more detailed description, like a transistor level model, would lead to a more accurate diagnosis, at the cost, however, of greatly increasing the diagnostic turn-around time.

The first step in diagnosis is to obtain a list of possible defects. These defects are used, one at a time, to modify the logic model of the Integrated Circuit. Each modification uses the original logic model and one selected defect from the list, and builds a logic model of a defective IC, one that differs from the defect free IC only by the presence of the defect on the device. Such a modified logic model is called a fault machine, and there are as many fault machines as there are defects in the defect list.

The second, main step in logic diagnosis is to simulate the scan based test patterns on each one of these fault machines, and to collect simulated fail data, i.e. miscompares at the scannable latches between the behavior of the defect-free logical model and the fault machine.

Finally, the simulated fails are compared with the ones collected on the tester, and some measure of agreement between the two sets of fail data, called a score, is calculated. If the score is sufficiently high, the defect that was used to construct the fault machine, as well as the simulated fails are stored in a file for later use. The high scoring defects are the ones whose

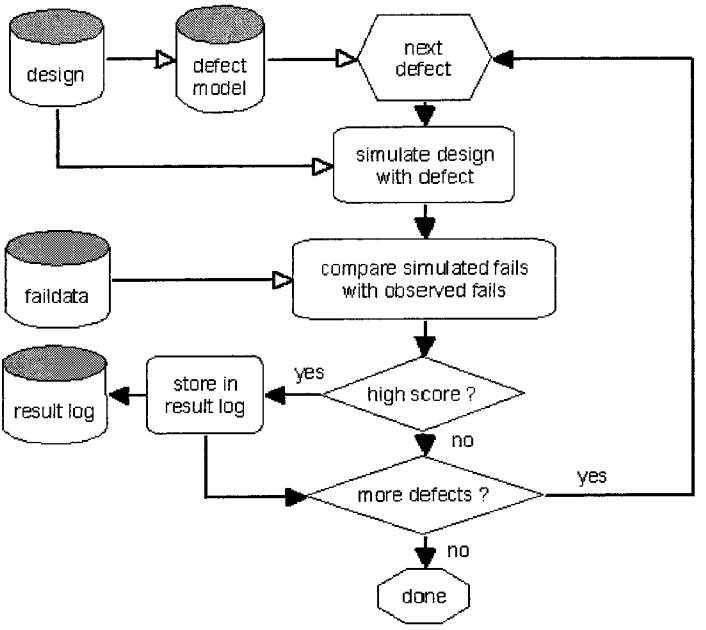


Figure 31 Basic logic diagnostic flow

behavior seems to match best the behavior of the defective device, and they will be the ones called by logic diagnosis as the best candidates for the defect.

This simplistic description of the diagnostic flow hides a multitude of practical problems. What potential defects should be in the list, and how should their logical behavior be determined and described ? The simulation step will be very time consuming if the defect list is large. Is it possible to decrease the turn-around time of this step, for example by selecting only those defects for simulation that are likely to have high scores ? Finally, how should matches between simulated and collected fail data be measured, and how should the resulting scores be interpreted ?

These questions will be addressed in this and the next chapter. Here, we will focus on a simplified diagnostic strategy that only uses defects that create errors on single nets in the defective device. It is the standard diagnostic support available in most commercial diagnostic packages. A more sophisticated form of logic diagnosis, called SLAT, will be discussed in the next chapter.

Many of the issues discussed here, however, will be relevant for SLAT as well.

1 DEFECT MODEL

The defects to be employed in logic diagnosis need to fulfill a host of often conflicting requirements. First and foremost, they should be able to mimic the behavior of real defects on real devices. On the other hand, it should be possible to describe their behavior logically, for, otherwise, it would not be possible to construct fault machines, which are not more than logic models of defective chips. Finally, the logic behavior of the defects should not be so complicated, nor their number so large that the diagnostic turn-around time becomes unacceptable.

The defects used in logic diagnosis are models of the real defects, and are called faults. When activated, they produce errors on the nets or pins where they are located, and we say that they affect those nets or pin. Errors are logical deviations from the defect-free behavior of the design. They may produce fault effects on nets and at pins in the cone of influence of the fault, and, eventually, may cause fault effects in observable latches or at observable Primary Outputs.

Before continuing, some explanation needs to be given of the usage of nets and pins in the discussions. The difference between nets and pins is not as large as it may seem. Figure 32 shows a simple logic diagram with some

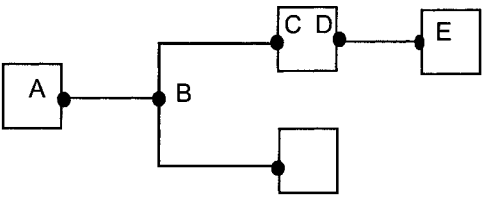


Figure 32 Relation between nets and pins

gates, drawn as boxes, some nets, and the pins A through E where the nets are attached to the gates.

Pins can be identified with nets or portions of nets. For example, output pin A can be identified with the net that attaches to it, or, at least, with the portion of the net between A and the first fanout point (B). If the net attached to an output pin does not fan out, like DE, the output pin can be identified with

the full net. Likewise, an input pin like C can be identified with the portion of the net that attaches to C, starting from the last fanout point (B), or, if absent, with the full net (as in DE). Nets and pins can, therefore, be used interchangeably, when this identification is kept in mind.

Choosing the proper defects to be put in the defect list is the same as choosing the proper faults to be put in, what is now called, a fault list. A good starting point is the fault model used in test generation. This is usually the single stuck-at fault model [2], in which faults are located on pins, and defects are modeled by a single faults. The logic behavior of a stuck-at fault is such that it forces the logic value on the pin to which it is attached to be either a logic 1 or a logic 0.

For example, if a stuck-1 fault (s@1) is present on an input pin, the gate to which the pin is attached experiences a logic 1 and not a logic 0, even when a logic 0 is applied to the net attached to the input pin. Likewise, a net connected to an output pin that carries a s@0 fault experiences a logic 0, no matter what logic value is produced by the gate on which the output pin resides.

Stuck-at faults are easy to model logically, and the modifications required to transform the defect-free model into a fault machine are straightforward. This fault model has another, less obvious advantage. As its faults reside on pins, their number is at most twice the number of pins in the design; in other words, the size of the fault model grows linearly with the size of the design.

It is important, however, that the fault list contains faults on all the pins in the design, for a defect that creates an error on some pin is modeled best by a fault on that pin. Even more importantly, diagnostic precision would be lost in the absence of such a fault, even if other faults could reproduce the behavior of this defect in a reasonable way, because those other faults would indicate the pins where they reside as the most likely candidates for the location of the defect, rather than the defect's actual location.

For example, the s@0 faults on the input and output pins of an AND gate are equivalent. Consequently, if the defect is such that it can be modeled accurately as a s@0 on an input pin, it can be modeled equally well by the s@0 on the output pin. But, if the defect list does not contain the input s@0 faults, the diagnosis will be misleading because, even though the list has a fault that exactly reproduces the fail behavior observed on the tester, it points at the wrong pin.

Various extensions of the stuck-at fault model are sometimes used, like transition faults ([21], Chapter 13), or pattern faults. Transition faults attempt to capture the effects of excessive delays when transitioning from one logic value to another, and pattern faults have more complex activation conditions than stuck-at faults but are otherwise like the latter in that they are static and

produce errors on single nets only. Both fault types share many of the drawbacks of stuck-at faults.

The cost of the required simulations, however, rapidly becomes prohibitively expensive with increasing model complexity. In addition, such more complex fault models may lose the linear relationship between the size of the fault model and the size of the design, and consequently, for sufficiently large designs, violate the requirement that the number of faults should not be too large.

In practice, stuck-at faults are the ones used, with transition faults optionally added when the defect is obviously timing sensitive. Stuck-at faults internal to complex gates can be replaced by pattern faults to reduce simulation complexity.

Stuck-at faults, transition faults and pattern faults are easy to model, and their number grows only linearly with the size of the circuit. The main problem is that most defects do not behave as stuck-at faults, or even as transition faults or pattern faults, and, therefore, that their ability to mimic realistic defects is in doubt. A more sophisticated use of the fail data than the one described in this chapter, however, circumvents this problem. This more sophisticated form of logic diagnosis will be described in Chapter 10, in which also further theoretical reasons will be given for why stuck-at faults should be able to mimic realistic defects.

2 FAULT SELECTION

For very large design, the list of faults may be correspondingly large, and may be too large for rapid diagnostic turn-around. It is therefore important to reduce the size of this list as much as possible before fault simulation begins.

The size reduction that is usually done is based on the observation that there has to be a functional path between the pins physically affected by the defect and the latches that have incorrect logic values at the completion of the test pattern (the failing latches.) The functional paths, however, may not be restricted to combinational logic, but may cross one or more latch boundaries, depending on the clock pulses issued during the application of the test pattern.

The strategy for reducing the fault list is to trace backwards from a failing latch through combinational logic till a Primary Input, an embedded memory, or another latch is encountered, and to store all the pins that were encountered during the tracing in a list. Such a backwards trace was described in Chapter 6.3.2. The logic encountered in such a trace contains the functional path(s) from the defect to the failing latch, if, in fact, this defect was responsible for this latch having an incorrect logic value.

Each trace starts from a failing latch and defines a backcone to that latch. When the tracing is done to reduce the fault list, the backcone consists of a set of pins. The set of faults on the pins will be indicated by $\mathcal{B}_{lp}$, with l indicating the failing latch, and p the pattern that caused this latch to have an incorrect value.

The pin where the defect can produce errors has to be in one or more of these backcones. If only a single pin can have an error, then the intersection of all the backcones should contain this pin. The faults on the pins in the intersection then form the intersection fault set

$$\mathcal{F}_{\text{intersection}} = \bigcap_{l,p} \mathcal{B}_{lp}, \tag{9.1}$$

which is likely to be small.

On the other hand, real defects need not cause errors on a single pin, and not all the pins affected by the defect may be in this intersection; in fact, they may be in non-overlapping backcones. Defects residing in non-overlapping backcones goes somewhat beyond the single fault assumption. It is a crude attempt to correct for that assumption's shortcomings, but it has become such a standard part of logic diagnosis that this extension will be discussed here rather than in the next chapter. It leads to a number of less restrictive fault selection strategies. The most liberal one is the union fault set

$$\mathcal{F}_{\text{union}} = \bigcup_{l,p} \mathcal{B}_{lp}, \tag{9.2}$$

which is obtained by taking the union of all the backcones. $\mathcal{F}_{\text{union}}$ is considerably larger than $\mathcal{F}_{\text{intersection}}$, but almost always still much smaller than the full fault list. It is the safest way of selecting faults, because there is no functional path from any pin outside the union of the backcones to any of the failing latches.

An intermediate selection strategy is used in SLAT (see Chapter 10,) but it can also be used here. It starts from the observation that explaining all the fails collected during test should start with explaining the fails collected when any particular test pattern was applied (say pattern p). If there is a single stuck-at fault that can explain the fails collected when p was applied, then that fault should be in the intersection

$$\mathcal{J}_p = \bigcap_l \mathcal{B}_{lp}, \tag{9.3}$$

in which the intersection is now taken over all the failing latches at the completion of pattern p.

If $\mathcal{J}_p$ is empty, no single fault can explain the fails of p, and no single fault diagnostic strategy will succeed, at least not for p. On the other hand, if some single faults can explain all the fails of p, they should be among the ones in the intersection $\mathcal{J}_p$.

Taking the intersection of all the $\mathcal{J}_p$ leads to $\mathcal{F}_{\text{intersection}}$, which may be too restrictive. A more liberal approach is to take the union over all the $\mathcal{J}_p$, which leads to the SLAT fault set

$$\mathcal{F}_{\text{SLAT}} = \bigcup_p \mathcal{J}_p. \tag{9.4}$$

The characteristic of $\mathcal{F}_{\text{SLAT}}$ is that every one of its faults can explain all the fails of at least one pattern, and, vice versa, any single fault that can explain all the fails of any single test patterns will be contained in $\mathcal{F}_{\text{SLAT}}$.

3 ALTERNATIVES TO SIMULATION

Even with the fault selection strategies described above, the simulation required for the diagnosis may still be formidable, and may still lead to large turn-around times. In addition to searching for techniques to increase the performance of the simulators - an ongoing effort - some other approaches have been explored to reduce the turn-around time.

The most important one of these is the dictionary approach ([2], Chapter 12, [48]). In this approach all the faults and all the patterns are simulated before any testing has started, and not when the faults are required during diagnosis, using only those patterns that actually failed during test. Once the dictionary has been built, diagnosis is reduced to a mere lookup in a, admittedly, large table.

This approach is obviously not suitable for products that will rarely be diagnosed, because of the cost of building the dictionary. On the other hand, it seems ideal for products that are likely to be manufactured in large quantities, and that may even be used as line or reliability vehicles, for the effort to con-

struct the table can be amortized over all the diagnoses that will be performed during the lifetime of the product (and, therefore, of the dictionary.)

The main problem with the dictionary approach, however, is not the cost of constructing the dictionary, but its size. Many designs have millions of faults and require thousands of patterns to test. Each failing latch requires on the order of thirty bits to be described: between ten and fifteen to name the pattern, and the remainder to name the latch (of which there can be several hundred thousand, even in medium sized designs.) Even if, on average, only a few latches (say ten) fail when a pattern is applied, storing this fail information requires three hundred bits per fault per failing pattern. With ten million faults and on average three hundred failing patterns per fault, this translates into a table containing one trillion bits of information.

This size makes the dictionary approach in its crudest form rather impractical. There have been several attempts to reduce the size of the table [11], mostly by reducing the amount of fail information stored for each fault and each pattern. Reducing this information also reduces the diagnostic resolution, however, i.e. the number of faults that are called as likely candidates for the real defect. No practical solution has been found yet that has both a practical dictionary size and an acceptable resolution.

4 SCORING MATCHES

The score is a measure of the agreement between the fails produced by a simulated fault, and the ones collected on the tester. This score is traditionally some number between 0 and 100, with 100 indicating perfect agreement. Of course, the score depends not only on the agreement between the defect and the fault, but also on the patterns that are used to gauge this agreement. When only some failing patterns are used, more faults may produce the same fault effects as the defect than when all failing patterns, or even all patterns, failing or not, are taken into account.

Gauging agreement between the fault model and the actual defect is very similar to measuring commonality between two different devices, and the scoring methods that are used for the former, as a result, are very similar to the commonality measured discussed in Chapter 6. As an example, I will briefly describe the scoring method used in Encounter Test.

What is known, after a set of patterns have been simulated on the fault machine, is the list of failing latches. This list needs to be compared with a similar list collected on the tester when the same patterns were applied to the real device. A failing latch is an (l, p) pair, in which the latch l contained an incorrect value after the pattern p was applied to the device or the fault

machine. The lists of failing latches are lists of such (l, p) pairs, and they can be compared one pair at a time. The section of the matrix in Table 2. bordered by the heavy line is one such list.

To compare two such lists quantitatively, a number of counts are defined:

- Tester Pass, Simulator Pass.
  Number of (l, p) pairs where the latch had the correct value after p was applied to both the fault machine and the device.
- Tester Pass, Simulator Fail.
  Number of (l, p) pairs where the latch had the correct value in the physical device but the incorrect value in the fault machine.
- Tester Fail, Simulator Pass.
  Number of (l, p) pairs where the latch had the incorrect value in the device but the correct value in the fault machine.
- Tester Fail, Simulator Fail.
  Number of (l, p) pairs where both device and fault machine had incorrect values after p was applied.

Clearly, the score should be an increasing function of TFSF, and a decreasing one of TFSP and TPSF, for TFSF measures how often both device and fault machine agreed on observable fault effects, while TFSP and TPSF measure how often they disagree. The score based on the commonality measure in Equation (6.4) is

$$100\frac{\mathrm{TFSF}}{\mathrm{TFSP}+\mathrm{TPSF}+\mathrm{TFSF}}, \tag{9.5}$$

and has the desired features. It does not contain TPSP, which is correct as it is very easy to get arbitrarily high TPSP counts, for example by applying patterns that do not exercise either the fault in the fault machine or the defect in the device.

Encounter Test uses a somewhat modified form

$$100\frac{\mathrm{TFSF}}{\mathrm{TFSP}+\alpha\mathrm{TPSF}+\mathrm{TFSF}}, \tag{9.6}$$

with α equal to 0.1. The purpose of this alteration is to reduce the relative importance of latches that fail during the simulation but not on the tester. Such latches are considered less important, because, for example, the fail data collection on the tester might have been incomplete, and it is actually not known whether this latch failed or not.

IF TFSF equals 0, the score is 0 regardless of TFSP and TPSF. On the other hand, if TFSF is not 0 and TFSP and TPSF are, the score is 100, and that is the only scenario in which the score can be 100. In other words, a 100 score indicates perfect agreement between the fault machine and the device for both the failing and the passing patterns. Notice however, that this agreement has only been established for the patterns that were used in the diagnosis, not for all possible patterns.

If the score is 0, TFSF is 0 and the simulated fault did not reproduce any of the failing latches observed on the device. Such faults cannot explain any of the fail behavior, and many of them are in fact already removed by the fault selection techniques described in Section 2. Any score between the two extremes indicates a fault that explains some of the observed fails, but not all.

It often happens that this form of diagnosis does not find a fault with a 100 score. The obvious reason for this is that the defect does not behave as a single stuck-at fault. The fault(s) with the highest score may still be useful, however, even though they are not perfect models of the defect, because they seem to capture some aspects of the defect.

The following discussion goes beyond the single stuck-at fault assumption, but is relevant here for it shows how scores are being used advantageously in logic diagnosis.

One of the causes of a score not being either 100 or 0 is that the defect creates errors on multiple pins in the design, and that the selected fault happens to model one of those manifestations of the defect. An example of this phenomenon is a bridging fault ([2], Chapter 7), in which the defect can alter the logical behavior on two nets (the two legs of the bridge), but only one at a time. The faults on the two legs then reproduce some, but not all of the observed fault behavior.

A wired-AND bridge, for example, behaves as two s@0 faults, with the added condition that neither fault is activated if both legs of the bridge have the same logical value. Both faults will be found by the present diagnostic strategy, if enough failing patterns are observed. All observed fails will be explained by a combination of the simulated fails of both faults [9], and the match is perfect when only failing patterns are used during diagnosis. The two faults will, however, make some patterns fail during simulation that do not fail on the tester (passing patterns.)

The defect may also create an error on a single pin, but with different polarities during different patterns. The best example of this type of defect is a dominant bridge, in which there is a short between two nets, but with the driving strength on one (the dominant net) being much larger than the driving strength on the other (the victim net.) Fault effects will emanate only from the victim net, but with the polarity depending on the logical value on the domi-

nant net. The diagnostic strategy developed here will find the two stuck-at faults on the victim net, both with intermediate scores, but such that their combination explains all the observed fails perfectly.

Finding such pairs can be done by analyzing the results of the diagnosis, and in particular of the complete (l, p) lists. Such a post-diagnosis analysis of the diagnostic results is often done, and can complete the diagnosis not only of the two examples mentioned above, but of several others as well, depending on the inventiveness of the diagnostic engineer. The diagnostic strategy to be described in the next chapter, however, implicitly does all these analyses, making further discussion of non-100 scores unnecessary.

5 EXPERIMENTAL RESULTS

The main measures of success of software based diagnosis are efficiency, resolution and accuracy. The first one is essentially the fraction of failing devices for which a high confidence diagnosis could be made. The second one is the number of faults with that high score, and the third one describes how well the location of the defect, predicted by the diagnosis, agrees with the actual location of the defect. In this section some experimental efficiency and resolution results will be presented. Accuracy cannot be measured as easily because of the cost of doing physical failure analysis, and will not be addressed here.

In 1998, an experiment was conducted in which one ASIC part was tested extensively. The goal of this experiment was to gauge various test methods, according to their effectiveness in detecting defects, as well as to gauge the availability, efficiency and accuracy of existing diagnostic methods in determining the locations of the defects that caused ICs to fail.

The vehicle chosen for this experiment was a SA12 ASIC part. It contains five levels of metal, 17 scan chains, the longest one being 1392 latches long, 15624 SRLs, including the latches in the LPRAs, and 4 SRAMs. The logic contains about 300K blocks. The fault list contains about 900K fault equivalence classes.

For each failing device, not more than 256 failing cycles were collected. Failing cycles is a technical term, and indicates the number of scan out clock events at which a failing bit was observed. During scan out, the latch contents become successively available at the scan out pins. Each time the latch content miscompares with the expected value, a record is made in a tester fail buffer. In this experiment not more than 256 such records were made. Because 17 failing bits could be logged at each scan-out event into that many different fail buffers, however, 3840 miscomparing bits could be collected,

although that maximum was never reached. On the other hand, the total number of failing bits can be substantially smaller than 256, if the defect is such that only few patterns will observe it. The lot that will be reported on here had 1062 failing devices. Only stuck-at and pattern faults, and only failing patterns were used during diagnosis.

The results of this experiment were also used in Chapter 3, Sections 2 and 5. In that chapter, the relative yields of all the lots in the experiment were discussed. Here, more detailed results for Lot_2 will be presented.

For each device, the highest score was determined, indicating roughly the success of the diagnosis. The distribution of the resulting highest scores is shown in Figure 33. In this histogram the scores have been grouped into vari-

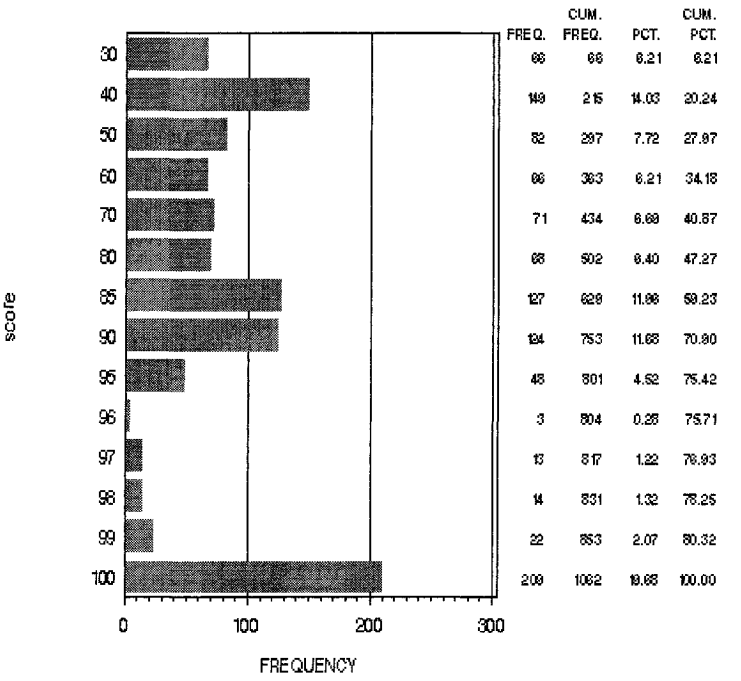


Figure 33 Distribution of high diagnostic scores

ous buckets. The buckets are labeled by their midpoints. For example, the bucket labeled 90 contains all the scores between 88 and 92, inclusively. The bucket labeled 30 contains all score up to and including 34

This lot was typical for the score distribution. About 50% have a score over 80, and, consequently, about 50% have a score below 80. A somewhat large group of 20 % has the highest score possible (100).

As the meaning of non-100 scores is not well defined, it is not clear at what score there is a separation between usable and non-usable diagnoses, and there may not even be such a separation. In practice, therefore, the scores are used to guide the diagnostic engineer towards those faults that seem to be most closely related to the actual defect, regardless of the actual value of the score.

To measure efficiency, we can arbitrarily set the separation at 90, in which case the efficiency for this design is about 30%.

Unfortunately, not only the highest score is important, but also the number of faults that have that score, or a score not substantially different from it. That there may be several faults with a 100 score is partially due to fault equivalence, and is unavoidable. Logic diagnosis cannot be more precise than an equivalence class. Faults that are not equivalent, but still have identical scores also occur, and lower the diagnostic resolution. This phenomenon will be addressed in more detail in the next section. Here, only some experimental results will be displayed.

To measure resolution, it is not useful to consider diagnoses with low scores. We therefore concentrate on those devices with efficient diagnoses, that is, with scores at least 90. As diagnostic resolution is limited to equivalence classes, only equivalence classes will be used in the next discussion. High scoring equivalence classes are of most interest, and they are defined as equivalence classes the faults of which have a score of at least 90.

Figure 34 shows the distribution of the number of high scoring equivalence classes among the efficient diagnoses, with the different scores differentiated by the hashing patterns. The bulk of efficient diagnoses has a resolution of at most 5, meaning that the number of high scoring equivalence classes is not more than 5. Note, however, that this can still translate into a large number of faults (or pins). Also note that about 10% of the efficient diagnoses has a resolution with more than 5 high scoring equivalence classes, and, in some case, over 50 of such classes. Such diagnoses should be considered failures, even though the scores were high. In fact, having a score of 100 is no guarantee that the diagnosis will have high resolution, as is shown in the figure by the number of diagnoses with a score of 100 that still have a large number of high scoring equivalence classes.

That the diagnostic approach often does not distinguish between different equivalence classes is a result of using only failing patterns. Section 6 will discuss this point in detail, while Section 7 will address the use of passing patterns to alleviate the problem.

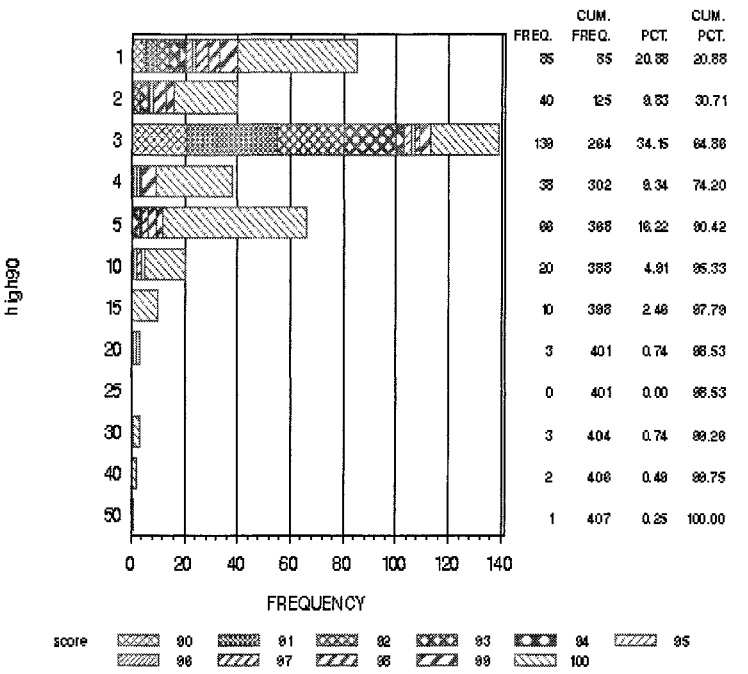


Figure 34 Distribution of the number of high scoring equivalence classes

6 RESOLUTION OF LOGIC DIAGNOSIS

For a diagnosis to be called successful, a number of conditions have to be met. There should be at least one fault that got a very high score, one that is close to, if not equal to 100, and the number of faults with high scores should not be large. Of course, there is no way to distinguish between faults of the same equivalence class, so, in reality, all we can demand is that the number of high scoring equivalence classes is not large. Preferably, it should be 1, but in some cases a low number of equivalence classes, like two or three is still acceptable.

It is very easy to end up with several distinct equivalence classes when only failing patterns are used. This is best illustrated with a realistic example. Figure 35 shows the result of a diagnosis on one of the devices of the experimental design. This schematic shows only that portion of the design where faults were found with a 100 score. The up- and down-arrows show the locations and polarities of those faults, an up-arrow meaning a s@1 fault and a

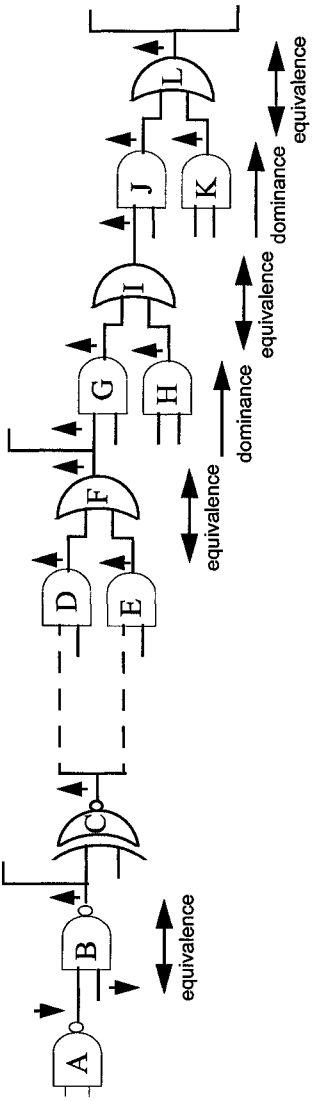


Figure 35 Example of a diagnosis with multiple equivalence classes

down-arrow a s@0 fault. The dashed lines between gate C and gates D and E indicate logic paths on which no faults were found. These two paths fanout from the output of the XNOR gate, but then completely reconverge at gate F. There are also fanouts after gates B, F and L, that were not explored any further.

The total number of faults with 100 scores is fourteen. There are five equivalence classes: four consisting of the input and output faults on gates B, F, I and L, and one on the output of the XNOR gate C. The equivalence groups are indicated by the doubly arrowed lines below the respective gates.

Such a low resolution diagnosis can easily arise when only failing patterns are used. For example, assume that the real defect is the s@1 fault on the output of B. As this fault is equivalent to the s@0 faults on the inputs of B, the latter ones will necessarily be found by the diagnosis as well.

Furthermore, failing patterns are, by definition, those patterns during which the defect was observed. Consequently, fault effects from the defect site had to have propagated to some observable output through one of the branches of the fanout after gate B. It seems, however, that all branches except the one through the XNOR gate were blocked, at least with the failing patterns used in the diagnosis. In addition, all the input values to the lower input of the XNOR gate were at logic 1 when the failing patterns were applied As a result, the s@1 fault on the output of C would have produced the same fault effects further downstream as the actual defect, and, consequently, was also found to have a 100 score.

The remainder of the faults with 100 scores follows now easily from equivalence and dominance [2]. A fault p dominates another fault q if all the patterns that test q also test p. For example, all the test patterns for a s@1 fault on the input of a OR gate are also test patterns for the s@1 fault on the output of that gate. It is then clear that the s@1 faults on the outputs of gates D, G and J dominate the s@1 faults on their respective inputs, and diagnosis will give them a 100 score as well, at least if only failing patterns are used. The dominance relationships are indicated in the figure by the single-arrowed lines below the OR gates.

The fanout after gate F does not change the flow of equivalences and dominances, presumably because fault effects propagating along the other branches of this fanout are blocked further downstream from F. That not more faults are found further downstream from L is due to the presence of the fanout after L. From there on, fault effects propagate into two different directions, and no fault on either branch can explain the fails observed at the latches at the end of the other branch.

This explanation of why there are so many faults with 100 scores uses the accidental facts that all the failing patterns used in the diagnosis put a logic 1

on the lower input of the XNOR gate, and that all those patterns also block fault effect propagation down the other branches of the fanout after gate B.

More complicated accidents are required if, for example, the real defect is a s@1 on the top input of F. Fault equivalence and dominance will guarantee that diagnosis will also give a 100 score to all the faults downstream from the defect, up to gate L. That diagnosis also gives a 100 score to the top input of D, however, can be explained only by assuming that the failing patterns put a logic 1 on the bottom input of that gate. Likewise, a constant 1 on the bottom input of C will make diagnosis find the s@1 fault on the top input of that gate, which is equivalent to the s@1 faults on the inputs of B.

The existence of these accidents indicate that it might be possible to increase resolution by finding more failing patterns. In fact, these accidents open up the possibility of exploiting them and generating special diagnostic patterns that intentionally put "wrong" values on the various pins [5].

For example, in the first scenario, having failing patterns with different logic values on the bottom input of C would remove all the faults downstream from C from the 100 list. Likewise, in the second scenario, all the faults upstream from the s@1 on the top input of F would get lower than 100 scores if at least one failing patterns would put a logic 0 on the bottom input of D.

The negative effect of dominance on the diagnostic resolution cannot be conquered, however, by using more failing patterns. Only passing ones will be able to distinguish between a defect and the faults that dominate the defect.

7 USING PASSING PATTERNS

The benefit of using passing patterns was studied by rediagnosing all devices with a highest score of at least 90, but now with many passing patterns added. The test patterns are divided into groups of around thirty patterns, and the selection of passing patterns was such that all the patterns in a group were used, failing or passing, if there was at least one failing pattern in that group.

The results are summarized in Figure 36. The effect of using passing patterns is the reduction of the scores of those faults that cause fails during the simulation of such patterns. It could, for example, lower the scores of faults that dominate the real defect without being equivalent to it, but only when the right passing patterns are applied.

The expected effect is a lowering of the number of high scoring equivalence classes. That this does occur is clearly shown in the figure, which shows the distribution of high scoring equivalence classes after diagnosis with passing patterns. Compared to the results in Figure 34, the average number of

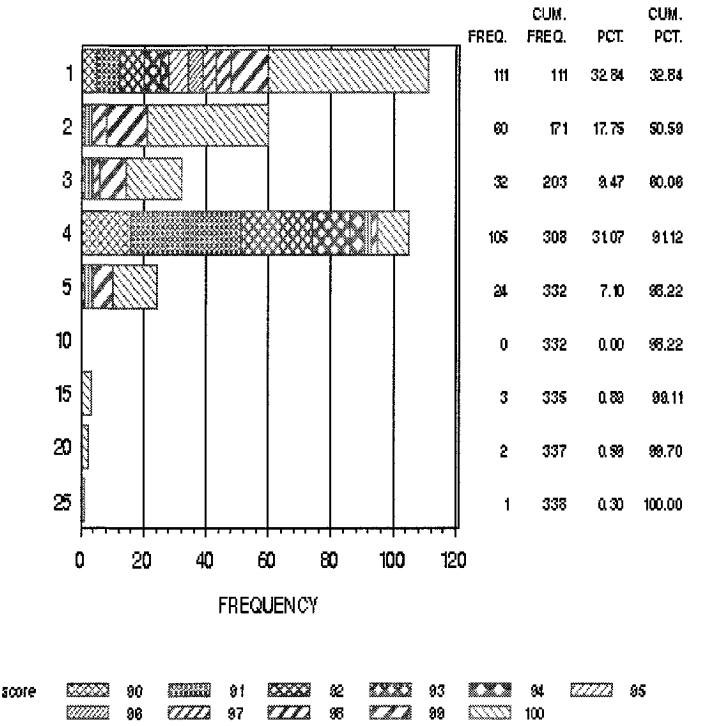


Figure 36 Distribution of the number of equivalence classes, with the use of passing patterns

equivalence classes per diagnosis has clearly decreased. The 50% point, which was around three, is now below two. Likewise, the 90% point moved from five down to four.

It is also interesting to see what passing patterns do to the diagnostic result shown in Figure 35. The diagnostic result with passing patterns is shown in Figure 37. The numbers above the faults indicate the scores that were obtained. Faults without score indicators have scores below 70.

Surprisingly, there is no 100 score. None of the candidates found previously, with failing patterns only, turned out to be perfect models of the real defect. The best candidate is the one with score 94, on the input to gate G. Note also that, with the use of passing pattern, dominance lowers the score, while equivalence keeps them the same.

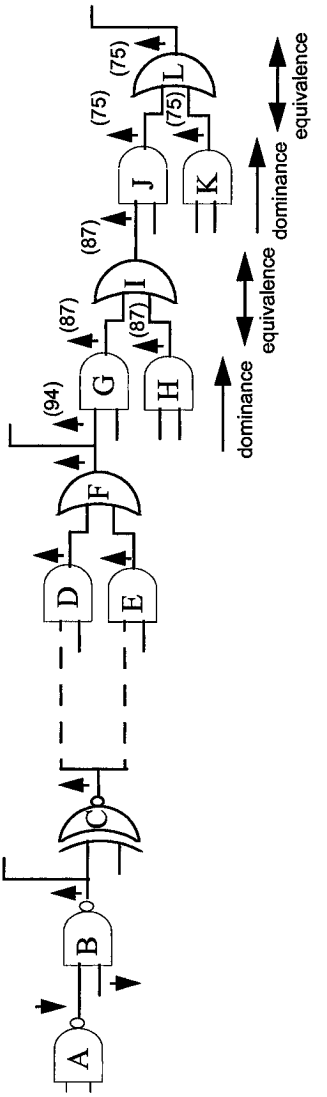


Figure 37 Example of a diagnosis with multiple equivalence classes and passing patterns

Chapter 10

SLAT based Diagnosis

1 INTRODUCTION

Even though the diagnostic approach outlined in the previous chapter has been fairly successful, there are several problems that are rapidly becoming more apparent with decreasing feature sizes, and with increasingly aggressive design styles that deviate more and more from the robustly digital behavior assumed by logic simulators. The most serious of those problems is that only single stuck-at faults are used.

Stuck-at faults are very restrictive as models of defects, because they allow the defect to influence only one net, force the defect to be active all the time, and assume that the defect behaves in one specific way. Using combinations of stuck-at faults removes the first of these drawbacks, but does little to alleviate the others: bridging and intermittent defects, for example, cannot be modeled by any combination of stuck-at faults.

The advocated solution to the inadequacy of stuck-at faults for diagnostic purposes has always been more complex faults (see, for example, [2] and [49].) One group of such approaches centers around the notion of composite signatures [40, 35, 36, 58]. The signature of a single stuck-at fault is the set of failing patterns, possibly augmented with, for each such pattern, the set of latches that contain incorrect data after the application of that pattern. A composite signature is the union of a suitable set of single stuck-at fault signatures. What particular set of stuck-at faults is chosen depends, of course, on the defect that the composite signature is intended to model. For simple bridging faults, they are the stuck-at 1 and stuck-0 faults on the two legs of the bridge.

The solution of more complex fault models, however, has problems of its own. First of all, the cost of the required simulations rapidly becomes prohibitively high with increasing model complexity. In addition, such more complex fault models may loose the linear relation ship between the size of the fault model and the size of the design, and consequently, for sufficiently large designs, violate the requirement that the number of faults should not be too large.

Secondly, no matter how sophisticated the arsenal of logic faults that is at the disposal of the diagnostic software, there will always be defects that do not correspond to any one of them. For example, typical bridging faults that are available with some simulators are of the wired AND or wired OR variety

([2], Chapter 7). There are, however, also dominant bridging faults of various flavors [36, 49], and the activation conditions of any bridging fault may be more complex than that the two legs of the bridge have opposite logic values [35].

These problems with the use of more complex fault models also highlight the second general problem with the previous chapter's diagnostic technique. A physical defect has two components. The first one is its location, or locations if the defect is a compound one. The second component is its logic behavior when test or functional patterns are applied. All of today's diagnostic approaches attempt to address these two components simultaneously. This, however, leads either to overly simplistic fault models, like the stuck-at one, or to a gross inadequacy of more complex fault models, like the bridging faults mentioned above.

The central problem in logic diagnosis is to model realistic defects by logic abstractions that faithfully mimic the logic behavior of the defect, but that can also be simulated efficiently by logic simulators. In this chapter, a solution to this diagnostic problem will be offered that approaches it in a drastically different way from the standard technique of the previous chapter. The main idea is not to model the logic behavior of the defect, but, instead, to focus on its location. Determining the location is much simpler, and can be done in a piecemeal fashion by analyzing failing patterns and building up a composite picture of the defect's whereabouts.

This diagnostic technique is called SLAT [6, 30], which stands for Single Location At a Time, because it uses only those patterns during which the defect affected only a single location, be that a pin or a net. In the first section of this chapter, modeling realistic defects will be revisited. It will be shown that defects can still be modeled by stuck-at faults, but in a decidedly non-standard way. Some of the ideas in this and subsequent sections, in particular the idea of using only those patterns during the application of which the defect affected only a single net, were anticipated more than fifteen years ago [60], but not pushed as far as here.

In the first section of this chapter, modeling realistic defects will be revisited. I will show that defects can still be modeled by stuck-at faults, but in a decidedly non-standard way. The remaining sections will describe how this new defect model can be used to perform sophisticated and powerful logic diagnoses. The detailed approach to logic diagnosis will be outlined in Section 3. The basic output of such a diagnosis will be discussed in Section 4, where more insightful interpretations of the diagnostic results will be developed. A faster, but approximate greedy version of SLAT will be presented in Section 5. Some final comments regarding the SLAT output will be made in Section 6, and results obtained on real ICs will be presented in Section 7.

2 LOGIC DEFECT MODEL

The defect model that underlies SLAT will be discussed in this section. SLAT does not assume that the defects are stuck-at faults, even though it employs them to obtain information about the defects. A more accurate description of a SLAT defect is that it is a set of locations. This section will define the SLAT defects in detail, and show why stuck-at faults can be used to obtain information about them.

2.1 Physical justification

Empirically, we know that the stuck-at fault model is very efficient in driving test generation towards tests that can give a high assurance of a low defect level. Likewise, standard logic diagnosis, as described in the Introduction, has shown that even the single stuck-at fault model can be very effective in diagnosing defective ICs. What is not clear is why stuck-at faults are so successful.

The real surprise, however, is not that stuck-at faults are effective, but, instead, that the logic model itself can be used even in the presence of defects; i.e., that the logic model of a defective IC can be obtained by making small modifications to the logic model of the defect-free device. This is surprising, for the function of the defect-free logic model is to represent the device in the absence of defects, and there are no requirements on how it should behave in their presence.

What allows the logic model to function, even in the presence of defects, seems to be the strong digital behavior of integrated circuits. Because ICs have to emulate logic designs, as supposed to analog designs, they have to be immune against small disturbances like electrical noise and temperature fluctuations. Likewise, even stronger disturbances that are caused by real defects are quickly brought back to digital behavior, be it sometimes incorrect digital behavior.

Consider for example the voltage at the input of a logic gate. When it is sufficiently close to Vdd or GND, its logic value is well defined. In other words, there is a region δ between Vdd and GND outside of which the voltage clearly defines a logic value. Logic gates have to be designed such that, in normal circumstances, and in the presence of normal fluctuations, all voltages stay away from this uncertainty region δ. This is accomplished by having the gate produce an output voltage that is far away from δ, even when some of the input voltages are close to δ.

A minor extension of this property can then be defined that, it seems, is present in today's technologies, and, if not, should be present in future ones.

For a logic gate to produce an output voltage in $\mathcal{D}$, one or more of the input voltages have to be in an even smaller range $\mathcal{d}$. $\mathcal{d}$ is completely included in $\mathcal{D}$, and smaller, because otherwise the gate would not have the correcting property mentioned above. A logic design now has strong digital behavior when $\mathcal{d} << \mathcal{D}$. In other words, a design has strong digital behavior when the logic gates almost always produce output voltages outside $\mathcal{D}$, even when the input voltages suffer strong disturbances that pull them within the region $\mathcal{D}$.

Of course, the logic gate may not redigitize the input disturbances correctly, and, if not, a logical error results. The important point, however, is that the logic model remains valid, and that the disturbance can be represented by some logic fault. We can go even further, though. As the physical disturbance gets redigitized at the first logic gate it encounters, it can be represented by a set of stuck-at faults on all the input nets that were affected by the disturbance. This set of nets, and the polarities of the stuck-at faults, may be different for different test patterns. The number of nets in the set, however, will be small, because physical defects can affect only a small number of nets, and most nets have small fanouts.

2.2 Logic defects

The starting point of the present diagnostic strategy, therefore, is the realization that, during the application of any test pattern, any defect behaves as a set of stuck-at faults on some set of nets. It need not behave as the same set of stuck-at faults on every pattern. During different patterns, the defect may behave as different sets of stuck-at faults, or even as in the defect-free device. For example, a defect that creates an error on a single pin can behave as a stuck-at 0 (s@0) on some patterns, a stuck-at 1 (s@1) on others, and even not have any faulty behavior at all on yet other patterns.

This leads to a new concept of a logic defect, one that lies at the root of SLAT. A logic defect is a model of a physical defect that is suitable for logic simulators and test generators. Logic defects mimic the behavior of the physical ones, but need not duplicate the details of the latter. They typically cannot model the detailed electrical behavior of physical defects, and may not be able to represent faithfully their spatial properties either.

For example, it may happen that a physical defect creates an error on some net, while the logical consequences of that influence can be described properly only some distance away from the defect because of electrical reasons. The most obvious example is the Byzantine bridges studied in [35], when the stem of a fanout tree is bridged to some other net, but the strengths of the various upstream and downstream transistors are such that some leafs of the fanout tree seem not to be affected by the bridge, while some others are. The

logic defect that describes such a bridge has as its basic components those leaf nets that can be influenced by the bridge, but not the other leaf nets, and ignores the stem of the tree.

A logic defect, then, is merely a set of nets that can be affected by the defect. Of course, this set of nets should be as parsimonious as possible in its explanation of the defect, but need not consist solely of the nets touched by the physical defect. *The goal of SLAT diagnosis is the identification of this set of nets.* A logic defect does have a logic behavior, but this logic behavior is not the primary target of the diagnostic efforts. It will become important when further refinements of the diagnostic calls are required, but plays no role in the initial phase of the diagnosis.

One important assumption regarding the logic behavior has to be made, however, to make it possible to build a diagnostic strategy around logic defects. It concerns the use of multi-clock test patterns. The application of a test pattern can be described as a multi-phase process. During the first phase, the circuit is brought into an appropriate state, typically using some form of scan-in. During each subsequent phase, the logical evaluations resulting from the preceding phase are clocked into various memory elements, and a new circuit state is generated. In the final phase, the values in the memory elements are observed at the tester, for example using scan-out.

If the application of the test pattern is a three phase process, as it is in simple scan designs, the logic evaluations are performed only in the second phase. In that case, the defect behaves as a set of stuck-at faults in the circuit state resulting from the application of the first phase of that particular pattern.

If the test pattern is more complex, and the circuit cycles through different states during the application of the pattern, we have to assume that the defect behaves the same way in all the states occurring during that pattern. This is a very strong assumption, and suggests that SLAT is not easily applicable to sequential tests.

The latter assumption can be weakened considerably, however, for the purpose of the assumption is to guarantee that all errors produced by the defect are caused at the same time, and not at different times, when the defect might behave in different ways. But the unique evaluation is guaranteed, and the purpose of the assumption satisfied, when the logic evaluations at any net are performed only once during the application of the pattern, even when the values in different observable latches and at different POs are set in different phases of the pattern. In fact, all that is necessary is that, regardless of how many times the logic value on some net is reevaluated during the application of the test pattern, only one evaluation of the logic value on that net can contribute to the logic values in observable latches and POs, which leaves room for a considerably amount of sequentiality in the pattern.

2.3 SLAT patterns

A second assumption needs to be made to make the present diagnostic strategy efficient. It deals with the problems stemming from complex defects that can affect multiple nets.

If a defect can produce errors on multiple nets during some failing patterns, it can do so during the application of all failing patterns, and this, as will become clear later, would make the defect undiagnosable. The second assumption that underlies SLAT is that there will be circuit states in which only a single net is affected (or, at most, only one net is affected from which fault effects propagate to some observable output.)

The patterns whose circuit states are such that only a single net is affected will be called SLAT patterns. In practice, a slightly different definition of SLAT patterns needs to be used, because it is of course not known for any particular pattern whether an error was produced on a single net or not. In particular, SLAT patterns that produce no fails can never be recognized as such.

For this more procedural definition, it is useful first to define the SLAT property. This is a property that is attached to failing patterns, and indicates that all the observed fails for that pattern can be explained exactly by at least one stuck-at fault, or more generally, by at least one single fault that, regardless of its activation conditions, can affect only a single pin. More descriptively, these are patterns during which the defect seems to have been activated in such a way that only one fault effect (a D or a $\overline{D}$) was generated at some pin, and during which this fault effect was propagated to one or more observable outputs. The SLAT property is similar to vectorwise intersection ([59]), except that the latter applies to all kinds of faults, and not just to stuck-at ones.

From now on, SLAT patterns are defined as failing patterns with the SLAT property. This may not always agree with the original definition of SLAT patterns, because it is conceivable that some pattern causes the defect to produce errors on several pins, propagating those errors to observable outputs in such a way that the observed fails can be explained by some single stuck-at fault. This is an unavoidable problem in all forms of diagnosis: that a complex problem manifests itself with the symptoms of a simpler one, and is confused with it. The risk of it happening seems slight, and I will assume that the procedural and the original definition of SLAT patterns are equivalent.

The restriction to SLAT patterns of course reduces the information that is available about the defect. The most serious risk is that no failing pattern has the SLAT property. In that case, SLAT diagnosis fails. Usually, however, there are enough patterns with the SLAT property. Obvious examples of defects for which all failing patterns have this property are stuck-at faults,

node faults like opens, and regular bridging faults like the wired AND and dominant ones.

The importance of the second assumption, and of its validity for large classes of defects, cannot be overestimated (and has been recognized before [3].) It makes it possible to apply all our accumulated knowledge of stuck-at faults to a much wider and much more realistic class of defects, and to treat those defects with even higher diagnostic accuracy and success than what we are accustomed to with regular stuck-at faults. If valid, standard stuck-at faults will suffice for logic diagnosis, be it in a rather nonstandard manner.

3 SLAT BASED DIAGNOSIS

The basic strategy that will be used to diagnose failing ICs is derived from the observations made in the previous sections. It is called Single Location At a Time (SLAT) to emphasize its reliance on the assumption that there are some failing patterns that produce errors on a single pin only.

Failing patterns with the SLAT property are called SLAT patterns. It is not required that all failing patterns have this property, nor that the defect is always active. SLAT will work equally well with intermittent faults as with hard faults (although it does rely on the availability of a sufficient supply of SLAT patterns, which may be hard to come by for intermittent faults.)

The result of the diagnosis will be logic defects, which are sets of pins, and a list of those failing patterns during the application of which the defect caused an error on one of those pins. All patterns whose failing latches are reproduced exactly by some fault on some pin will be said to be explained by that pin.

All logic defects consist of input and output pins on logic blocks, like ANDs and Buffers, as well as chip inputs and outputs. Standard logic diagnosis uses the concept of fault equivalence, which is tied to the polarity of the fault. As SLAT does not use those polarities, fault equivalence has no meaning, and all pins have to be identified that can explain the failing pattern, not just the ones that are the locations of the representatives of fault equivalence classes.

It is important that all the pins in the design are used as potential fault locations, for a defect that creates an error on some pin is modeled best by a fault on that pin. Even more importantly, diagnostic precision would be lost in the absence of that pin, even if faults on other pins could reproduce the behavior of this defect in a reasonable way, because those other faults would indicate the pins where they reside as the most likely candidates for the location of the defect, rather than the defect's actual location.

For example, the s@0 faults on the input and output pins of an AND gate are equivalent. Consequently, if the defect is such that it can be modeled accurately as a s@0 on an input pin, it can be modeled equally well by the s@0 on the output pin. But, if the defect list does not contain the input pins, the diagnosis will be misleading, for it points at the wrong pin, even though the list has a fault that exactly reproduces the fail behavior observed on the tester.

SLAT diagnosis recognizes three types of patterns: those that have the SLAT property, called SLAT patterns, failing patterns that do not have the SLAT property, and non failing patterns. The latter two groups will be discussed briefly once all the available information has been extracted from the first one. The starting point of the diagnosis is the set of SLAT patterns. SLAT assumes that there will be enough of such patterns to do a meaningful diagnosis.

3.1 Initial Diagnosis

SLAT diagnosis proceeds in three phases. In the first phase, shown in Figure 38, SLAT patterns are identified and pertinent information is stored in a table. It consists of a double loop, the outer one over all the failing patterns, the inner one over all the faults in the fault list (or, at least, over other faults than those that can be easily excluded, for example after tracing through the logic model, because they cannot possibly explain all the fails observed in the present failing pattern). Standard stuck-at fault diagnosis is performed on each failing pattern separately, using the diagnostic technique described in the Introduction.

The diagnostic step identifies the SLAT patterns, because they are the ones for which there is at least one stuck-at fault that completely explains all the fails collected for that pattern. It also identifies all the faults that can explain all the observed fails for each such pattern. For each fault, it notes the pin where that fault is located.

All pin-pattern pairs that are found in the diagnostic step are stored in a table, called the explain fails table, a small example of which is shown in Table 7.. The pins that explain failing patterns are indicated by the ☒ symbol. Additional information, like the fault id and fault polarity, are stored as well, but will not be used immediately. The polarity in particular is useful to store, even though it is not used in this phase of SLAT, as it may help refine the SLAT diagnosis after the present phase has finished.

Once the explain fails table has been created, the second phase of SLAT is entered, in which small sets of pins are identified such that each SLAT pattern is explained by at least one pin in each set. This search can be restricted to pins in the table, for other pins do not explain any failing pattern at all. The search is done simply by first checking whether any single pin can explain all

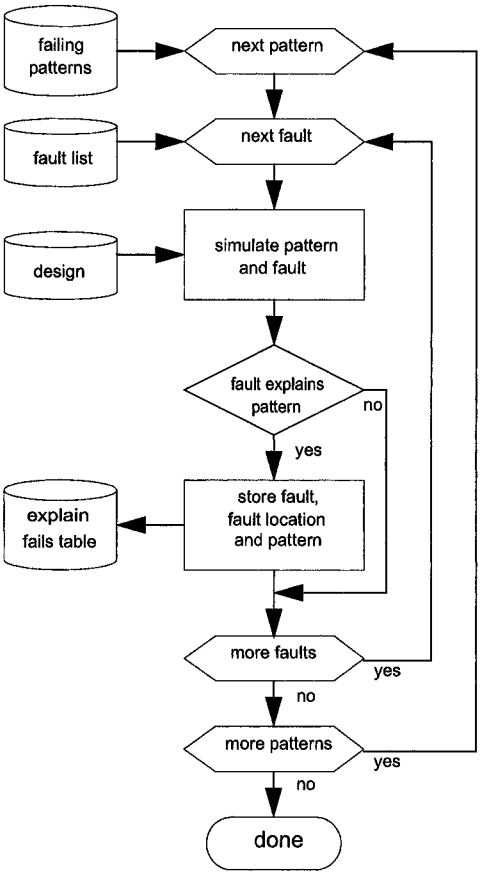


Figure 38 SLAT diagnostic flow

the fails, then if any pair of pins can explain all the fails, etc., until a suitably sized multiplet is found. A multiplet is a set of pins, and its size is the number of pins it contains. A detailed pseudo program for identifying multiplets can be found in [6]. Each multiplet is said to explain all the SLAT patterns, since each SLAT pattern can be explained by at least one pin in the set.

Once a multiplet is found that can explain all the fails, all multiplets of the same size that can also explain all the fails are identified, and no multiplets of larger size are considered. Consequently, only sets of the same, minimal size are found by SLAT, and no multiplet of smaller size can explain all the fails.

Usually, there is more than one multiplet. In the example shown in Table 7., suitable multiplets are (1,3), (1,4), (1,9), (7,3), (7,4) and (7,9). There are undoubtedly larger multiplets that can also explain all the fails, but they are

	pin 1	pin 2	pin 3	pin 4	pin 5	pin 6	pin 7	pin 8	pin 9
pattern 1	☒	☒					☒		
pattern 2			☒	☒	☒				☒
pattern 3	☒					☒	☒		
pattern 4			☒	☒				☒	☒

Table 7. Example explain fails table

ignored. This introduces a small risk of missing important information, and will be discussed further in Section 3.3

A different and more complex example of an initial SLAT output is shown in Figure 39. It contains a list of multiplets, as well as, for each multiplet, a

```
19 patterns failed but were not SLAT patterns.
37 SLAT patterns were found.
The returned multiplet size is 3.
40 multiplets were found.

Multiplet 1:
pin index 1.
  pattern 1      fault 1 ISA1
  Pattern 2      fault 1 ISA1
    .
    .
pin index 3.
  pattern 10     fault 2 OSA1
  pattern 11     fault 2 OSA1
    .
    .
pin index 7.
  pattern 80     fault 3 ISA1
  pattern 81     fault 3 ISA1
    .
    .

Multiplet 2:
  .
```

Figure 39 Initial SLAT output

description of how each SLAT pattern was explained (which pin in the multiplet and which particular fault.) It is not the final SLAT output, but represents an intermediate state of the diagnosis. The figure shows, in order, the number of failing patterns that did not have the SLAT property, the num-

ber of those that did, the size of the multiplets, the number of multiplets, and a complete list of all multiplets (only one of which, multiplet (1,3,7), is shown.) Each multiplet is listed as a pin followed by a list of the SLAT patterns that are explained by that pin (only some of which are shown.) The explanation has the form of SLAT pattern, fault index, fault polarity, with the latter two referring to the fault that explained that pattern perfectly. ISA stands for Input Stuck At and OSA stands for Output Stuck At.

The figure shows only the minimal amount of information necessary to explain the initial diagnostic output of SLAT. In practice, other information can be added, like the net and the gate where the pin resides, the function of the gate, or the stuck-at faults that were employed to locate the pin.

The defect indicated by SLAT has size three, which means that it affects at least three pins. There is no indication here, nor will there be after a more detailed analysis, whether this means a single physical defect that affects three pins (or nets feeding those pins,) or three distinct physical defects. It may be possible to guess the nature of the defect from the SLAT diagnosis, and have this guess be verified using the passing patterns or the failing ones that do not have the SLAT property, but the accuracy of the guess is not guaranteed.

3.2 Comparison with stuck-at fault diagnosis

SLAT is clearly more powerful than single stuck-at fault diagnosis, because it can diagnose all defects that have at least some SLAT patterns, while single stuck-at fault diagnosis requires defects to behave as single stuck-at faults all the time. Also, whenever the latter is able to explain all the observed fails perfectly with single stuck-at faults, SLAT will find multiplets of size 1, and all the faults found by stuck-at fault diagnosis will be among the multiplets.

SLAT, however, pays a price for this increased diagnostic power, in that it may incur some loss of resolution compared to stuck-at fault diagnosis whenever the defect really is a stuck-at fault. An example is shown in Figure 40.

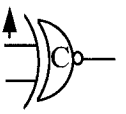


Figure 40 Example stuck-at fault

The XNOR gate shown in the figure has a s@1 fault on its top input. Stuck-at fault diagnosis will of course find this fault, as will SLAT. If both inputs have constant logic values in all the failing patterns, stuck-at fault diagnosis will also put a fault on the bottom pin on the list of candidate faults, and

SLAT will find a multiplet consisting of that pin. SLAT, however, will always have a multiplet with the bottom pin, regardless of the logic values on that pin, for either a s@1 or a s@0 on that pin can explain all the fails on any failing pattern.

This extra multiplet may be considered a disadvantage, but is unavoidable in SLAT. It is only a disadvantage if the defect is a stuck-at fault, however, something that is not known at the start of the diagnosis. If the defect were a bridge on the bottom pin, for example, stuck-at fault diagnosis might not find it at all.

Moreover, in the scenario sketched here, SLAT would find one pin, the top one, that has a defect with the same polarity, and another pin, the bottom one, that has a defect with varying polarity. This information can be used with, for example, physical design details to further reduce the list of possible defects. Such more detailed analysis falls outside the scope of SLAT, however (and also outside the scope of stuck-at fault diagnosis.)

3.3 Potential accuracy risks

In principle, SLAT should be able to diagnose most defects, the only obvious exception being those defects for which no failing pattern can be explained by a single stuck-at fault. The procedures outlined in this chapter, however, may lead us in some cases to an incorrect diagnostic call. These risks all entail accepting a simpler, but incorrect explanation of a fail over the correct but more complex one. This is a problem that all diagnostic strategies face: the correct diagnosis may be more complex that the simplest one, but the simplest one is the one we go with if we want to make any diagnostic call at all. There is nothing wrong with listing potential alternate diagnoses, and sometimes personal experience may lead us to focus on one of them rather than on the simplest one, but, with no other information, the simplest diagnosis should be selected.

The procedural problems have been mentioned before. The first one is caused by the decision to call a failing pattern a SLAT pattern when all the fails observed with that pattern can be explained by a single stuck-at fault (or, more generally, by a single pin fault.) As was pointed out at the time (see Section II-2.3), this may identify some failing patterns as SLAT patterns, even though multiple errors were produced and more than one of them caused fails at observable outputs.

The core of the problem is of course that a simple but incorrect explanation is available for the fails observed with that particular pattern, and that we are bound to accept that explanation. The risk may be reduced by using many failing patterns. For, if there are many true SLAT patterns, there is a good chance that the one pattern that was labeled a SLAT pattern but isn't will

stand out by its explanations being incompatible with the ones found for the other failing patterns.

The second procedural problem springs from our decision to look for minimal multiplets only. An example of how this may lead to an incorrect diagnosis is provided by an XNOR gate in Figure 40 (see also [2]). Assume that the defect is a bridge between the inputs of the gate, instead of a stuck-at fault. Such a bridge will manifest itself as a s@1 on the output of the gate, and SLAT will find it, since it requires a single pin to explain all the fails, rather than the real defect which requires two pins. Again, this problem is hard to avoid, because there is a simple explanation of the fails that hides the more complex correct one.

It is possible to modify SLAT, and have it look also for multiplets that have only a few more pins than the minimal ones. This would greatly increase the complexity of the SLAT output, and seriously decrease its accuracy, even though it might catch the occasional hidden defect. Even with this more complex SLAT procedure, however, there is still no strategy for recognizing the true defects. In the absence of additional information, simplicity is the only guide we have, and minimal multiplets are the ones that will be selected.

3.4 Non SLAT patterns

Each multiplet can explain all the SLAT patterns, and is, in some sense, a representation of the defect. Combined with the list of patterns that were explained by each pin, as well as the polarity of the fault that was activated, it gives a detailed description of how the defect behaved when the SLAT patterns were applied.

SLAT diagnosis, therefore, gives a picture of the defect in certain restricted circumstances (only for SLAT patterns.) It is not necessarily a complete picture, however, and there is no guarantee that a consistent logic defect can be found that will mimic the logic behavior of the defect in all circumstances. In particular, it may not be possible to extrapolate the behavior found for SLAT patterns to failing patterns that do not have the SLAT property, or to passing patterns.

The non SLAT patterns are important, because they can sometimes be used to obtain a more detailed picture of the defect. Failing patterns that do not have the SLAT property are patterns where a single stuck-at fault does not model the defect correctly. Instead, errors were caused by the defect at multiple locations, such that two or more of them gave rise to incorrect data at observable outputs.

It is tempting to assume that those multiple locations can be found among the pins in the multiplets that SLAT diagnosis produced initially, i.e. that SLAT diagnosis has found all the locations (pins) where the defect can pro-

duce errors. This can be verified by simulating all combinations of all pins found in any multiplet, and comparing the results against the fail data that were collected at the tester. In the example shown in Figure 39, this amounts to 320 simulations, easily within reach of today's simulators.

Passing patterns are very different. It may be that the defect was not activated, or that it was activated but no fault effect managed to propagate to an observable output. It is not possible to verify either scenario without a more detailed model of the defect. Such a model will have to be extracted from the immediate output of the SLAT diagnosis, augmented perhaps with an analysis of the failing patterns without the SLAT property.

It is sometimes possible to extract such a realistic model from the initial SLAT results [7]. For example, if all failing patterns are SLAT patterns, and if the multiplets have size one, and if the polarities of the stuck-at faults that correspond to the multiplets are always the same, then a stuck-at fault as a model for the real defect may seem to be appropriate. The confidence with which we arrive at that conclusion depends on the number of failing patterns that were employed in the diagnosis, and maybe on other factors, like known peculiarities of the manufacturing process, or details of the design near the multiplet pins.

If we do trust this model extraction, then it can be used in fairly standard ways to improve the accuracy of the diagnosis: follow the standard diagnostic strategy, as outlined in the Introduction, using only the stuck-at faults of the right polarities attached to the pins in the multiplets, and remove those stuck-at faults, and corresponding multiplets, that caused fails during the simulation of any passing pattern.

Like any extrapolation, however, this model extraction has an element of risk, because it is based on a limited and imperfect amount of information. If, in the example above, the real defect is not a stuck-at fault, but only appears to be one based on the information available from the failing patterns, simulating passing patterns in the presence of stuck-at faults is clearly inappropriate. The best that can happen is that the simulation removes all candidate stuck-at faults, thereby demonstrating the incorrectness of the extraction. On the other hand, if some of the stuck-at faults are left by the simulation of the passing patterns, the best a diagnostic engineer can do is to accept the model extraction, for it explains in the most simple manner all the data collected during test.

4 MULTIPLET ANALYSIS AND SPLATS

In general, diagnosis will find more than one multiplet, just as in the case of stuck-at fault diagnosis there is usually more than one fault that can explain all the fails. In its crudest form, the multiplets produced by SLAT are separate and independent. Each one of them stands on its own as a representation of the defect.

The pins in the multiplets are the ones that seem most directly related to the defect that caused the fails. This set of pins will be referred to quite often, and, for brevity's sake, will be indicated by *M*. The pins in *M* are the SLAT equivalent of faults in the standard diagnostic strategy that match the fails perfectly. We expect some structure in *M* if the size of the multiplets is larger than one, because the fails that are being diagnosed were caused by a possibly complex defect, and this provenance of the fails will undoubtedly be reflected in the diagnostic output. This additional structure will be explored in this section.

The basis of the diagnosis is the set of SLAT patterns. This set could be partitioned according to the single pins where the errors occur, if we knew for each SLAT pattern on which pin the defect produced an error. The diagnosis for the patterns in each subset would then basically be the same as in standard logic diagnosis, except that the latter is fault oriented, while SLAT is pin oriented. A set of pins would be found, each one of which explains all the fails collected for the patterns in that subset. Each pin in the set might or might not explain patterns in other subsets.

This set of pins will be called a splat.The pin that is affected by the defect is likely to be among the ones in the splat, but its identity cannot be established any more accurately than that. There can be as many splats as there are pins that are affected by the defect, or fewer, if no error occurred on some of those pins during the failing patterns used for diagnosis.

Finding splats is easy, once the SLAT patterns are partitioned according to the pins where errors were produced. They are very valuable, because *they are the best estimates diagnosis can provide of the pins affected by the defect, and, therefore, of the defect itself.* There is no obvious way, however, to partition the SLAT patterns. To find an effective algorithm, a different definition of a splat has to be used, one that makes the relationship to multiplets explicit. The third phase of SLAT is concerned with finding and applying this algorithm.

4.1 Splat structure

Partitioning SLAT patterns induces a partition of at least some of the pins in M, the set of pins in the multiplets, as each pin in M can be associated with a particular subset of patterns if it explains all the fails observed with the patterns in the subset. Alternatively, it might be possible to start from M, and search there for a splat structure.

An example of how this might be accomplished is shown in Table 8.. This

	pin 1	pin 2	pin 3	pin 4	pin 5	pin 6	pin 7	pin 8	pin 9
pattern 1	☒	☒					☒		
pattern 2			☒	☒	☒				☒
pattern 3	☒					☒	☒		
pattern 4			☒	☒				☒	☒

Table 8. Initial SLAT diagnostic results

table is the same as Table 7., but with the pins not in M indicated by the additional shading. Pins 1 and 7 explain patterns 1 and 3, while pins 3, 4 and 9 explain the remaining patterns. Why pins 1 and 7 explain the same patterns is of course not clear from this output. Perhaps, the fault on one pin dominates a fault on the other pin.

This table shows six multiplets: (1,3), (1,4), (1,9), (7,3), (7,4), and (7,9). On the other hand, the figure also suggests a partitioning into splats: (1,7) and (3,4,9). This partitioning into splats becomes obvious once the rows and columns are reordered (see Table 9.), and only the multiplet pins are kept.

	pin 1	pin 7	pin 3	pin 4	pin 9
pattern 1	☒	☒			
pattern 3	☒	☒			
pattern 2			☒	☒	☒
pattern 4			☒	☒	☒
	splat 1		splat 2		

Table 9. Splat structure of multiplet pins

Reordering puts the matrix in block-diagonal form, with the splats corresponding to the different blocks. This observation forms the basis of the splat analysis.

In the following discussion, italic capital letters like A and B will indicate subsets of SLAT patterns, and S_A will indicate the pins in the splat corresponding to subset A. The set of all the pins in splats will be indicated by S.

4.1.1 Completely separated splats

Let us start from the simplest, but also the most common case, in which the splats are completely separated, meaning that no pin in any splat explains any of the failing patterns associated with another splat. In that case, S and M are identical, as is shown in Appendix J. Multiplets and splats are then just different ways of partitioning M (or S). The number of splats is the same as the size of the multiplets, because each multiplet must have a pin from each splat, and no two multiplet pins can be in the same splat. Each multiplet can be obtained from the splats by choosing one pin from each splat. Likewise, splats can be obtained by putting the matrix formed from the SLAT patterns and the multiplet pins into block-diagonal form, as in Table 9..

A more elaborate example of a diagnosis with completely separated splats is shown in Figure 41, and refers in fact to the same device that was used in Figure 39. This Figure shows a final SLAT output, minus some additional information like the identity of the pin or the block on which it is located. The multiplet shown in Figure 39 is constructed from the first pins in splats 1, 2 and 3, respectively. That SLAT diagnosis finds 40 multiplets is now seen to be a consequence of the fact that there are splats of size 2, 4 and 5. How the various pins explain the observed fails is not indicated in the figure. It proceeds roughly in the same fashion as in Figure 39, except that the list of patterns explained by each pin has to be given only once for that pin, rather than repeated each time the pin occurs in a multiplet.

4.1.2 General case

The identity of S and M has been shown only for the case of completely separated splats. This case may seen rather special, but is in fact very likely to occur if the defect affects nets that are logically unrelated, because then a pin that explains all the failing patterns in one subset is very unlikely to explain also the patterns in another subset.

There is no guarantee, however, that splats will always be completely separated. They need not be, as a pin in a splat S_A may accidentally explain one or more patterns in subset B. If this happens, as in Table 10., the matrix of patterns and multiplet pins cannot be brought into block-diagonal form anymore.

```
19 patterns were skipped.
37 good patterns were found.
The returned multiplet size is 3.
Found 40 multiplets.

The 2 pins in splat 1 are:
pin index 1
pin index 2

The 4 pins in splat 2 are:
pin index 3
pin index 4
pin index 5
pin index 6

The 5 pins in splat 3 are:
pin index 7
pin index 8
pin index 9
pin index 10
pin index 11
```

Figure 41 Final SLAT output

	pin 1	pin 7	pin 3	pin 4	pin 9
pattern 1	☒	☒			
pattern 3	☒	☒		●	
pattern 2	●		☒	☒	☒
pattern 4			☒	☒	☒
	splat 1		splat 2		

Table 10. Non-completely separated splats

The ● symbols in Table 10. indicate such additional, nuisance explains. Despite their presence, the matrix is approximately block-diagonal; it is still obvious how to partition the SLAT patterns, and, consequently, what the splats are. The problem is how to transform this observation into a reliable algorithm.

The more general case can almost always be handled by generalizing some of the properties of the completely separated case. This generalization leads to the following rules for finding splats:

1. Only pins from M can be used;
2. no two pins from the same multiplet are in the same splat;
3. the sets of SLAT patterns explained by pins in the same splat have a non-zero intersection.

The first rule merely states the obvious: realistically speaking, all we know is M, and we have no good way of including pins not in M. The second rule describes the central relation between splats and multiplets, that multiplets are sets of pins, one from each splat. Consequently, two pins in the same multiplet cannot be in the same splat. The final rule defines the main property of splats, that its pins all explain some core set of patterns, and, therefore, that each set of patterns explained by any of those pins must at least contain that core set.

Returning now to how to find splats in M, the rules listed above are a guide for how to group the pins in M into splats. Finding splats is, therefore, a form of clustering pins in M, with the criterion being the degree of matching between the sets of patterns that the pins explain, and rule 2 providing an additional, negative criterion.

Let us define the pin commonality

$$h(I, J) = -1, \tag{10.1}$$

if the pins I and J occur together in some multiplet, and

$$h(I, J) = \frac{\sum_k v^I_k v^J_k}{\sum_k (v^I_k + v^J_k - v^I_k v^J_k)} \tag{10.2}$$

otherwise. In this equation, the sums are over all SLAT patterns, and v^I_k equals 1 if pin I explains pattern k, and 0 otherwise. h(I,J) is set to -1 rather than to 0 when I and J occur in the same multiplet to distinguish between that case and the case of two pins that don't have any failing patterns in common (see rule 3.) In both cases, the pins should not be put in the same splat, but it is convenient to label them differently. It has no consequences for the clustering

algorithm. In the completely separated case, h(I,J) equals 1 when I and J are in the same splat, and -1 when they are not.

The technique for finding splats in the general case is then to use some agglomerative clustering method ([19], Chapter 5), for example the one described in Chapter 6.4, which proceeds by first putting each pin in its own cluster, and then reducing the number of clusters by merging at each step those clusters that have the greatest degree of commonality. Merging stops when the number of clusters has been reduced to n, the size of the multiplets, or when the only clusters that can still be merged have commonality 0 or -1. Of course there is no need for clustering when the size of the multiplets is 1, because then the looked-for splat is *M*.

This approach will obviously work in the completely separated case, and will work in more general cases as well. For example, for the SLAT results shown in Table 10., the resulting commonality matrix is shown in Table 11..

pin 1				
2/3	pin 7			
-1	-1	pin 3		
-1	-1	2/3	pin 4	
-1	-1	1	2/3	pin 9

Table 11. Commonality matrix for Table 10.

Only the lower half of the matrix is shown, as h(I,J) is symmetric. The (1,7), (3,4,9) splat structure indicated in Table 10. will clearly be retrieved from this commonality matrix.

In almost all cases, the clustering that is found is unique, in which case the clusters can safely be considered to be the looked-for splats. The criterion for uniqueness is that the pins in the same cluster (now called a splat as well), are closer to each other than they are to pins in other clusters. To be precise, clustering is unique, and the clusters can be interpreted as splats, if, for all S_A and S_B, and for all pins I and J belonging to S_A, and all pins K belonging to S_B

$$h(I, J) > h(I, K). \tag{10.3}$$

One very important example of unique splats will be discussed in the next section.

4.1.3 Complete set of multiplets

Very often it is possible to group the pins in M by mere inspection such that each multiplet can be obtained by taking one pin from each group, and, vice versa, each set formed by taking one pin from each group is a valid multiplet. If the pins in M can be grouped in this fashion, M is called complete. Table 10. shows an example of a complete, but not a completely separated M. Completeness is similar to complete separateness, but it does not require that pins in one group do not explain any of the patterns associated with a different group. It was used in [6] to find splats.

If such a grouping exists, it is unique, as follows almost immediately from the definition of completeness. As the multiplets are formed by taking one pin from each group, there are as many groups as there are pins in the multiplets. This also implies that no two pins from the same multiplet can be in the same group. Assume now the existence of two different groupings G and H, both complete. As G and H are assumed to be different, there should be two pins that are in the same group in G, and in different groups in H. Because they are in different groups in H, they should occur together in some multiplet, by the definition of completeness. As they are in the same group in G, on the other hand, they cannot occur in the same multiplet. Consequently, G and H cannot be different, and the grouping that demonstrates completeness is unique.

The groups have all the desired properties of splats. Rules 1 and 2 are trivially satisfied, and rule 3 is too, for, if there were two pins I and J in some group that have no explained patterns in common, the patterns explained by I would not be explained by J, and vice versa. Each multiplet m that contains J, however, needs to explain the patterns explained by I, and, as J does not explain them, they would have to be explained by pins in m other than J. Now consider another multiplet m' that is equal to m, except that J is replaced by I. The pins in m' other then I explain all the SLAT patterns, because they explain all the patterns explained by I. m', therefore, does not need I, and would not be minimal, contrary to what is done in the second phase of SLAT. Consequently, I and J have to have some explained patterns in common, and rule 3 is satisfied.

In addition, Equation (10.3) is satisfied, for each pair of pins from two different groups occur together in some multiplet and have commonality -1. The grouping, therefore, coincides with the unique splat partitioning found by clustering.

Completeness has an alternate definition that is more useful when using the general clustering technique. If the clustering succeeds in finding n splats, with n the size of the multiplets, then each multiplet consists of one pin from each splat. The reverse may not be true: not every combination of one pin from each splat needs to be a valid multiplet. If the reverse is true, M is com-

plete. Consequently, M is complete if and only if clustering finds n splats, and if the product of the sizes of the splats equals the number of multiplets. Using this property of completeness, it is found that almost all SLAT diagnoses are complete.

4.1.4 Risks

There are some potential risks with the clustering approach to finding splats, similar to the risks mentioned in Section 3.3. In the first place, the starting point is the set of multiplet pins, and the assumption that S and M are identical. This assumption may be invalid, for M may have pins that are not in S when not all failing patterns that can be explained by single stuck-at faults are true SLAT patterns. More importantly, not all pins in S need be in M, because our restriction to minimal multiplets may hide the true defect. This is further illustrated in Table 12., which shows a variant of Table 10., with pin 4

	pin 1	pin 7	pin 3	pin 4	pin 9
pattern 1	☒	☒		●	
pattern 3	☒	☒		●	
pattern 2	●		☒	☒	☒
pattern 4			☒	☒	☒
	splat 1		splat 2		

Table 12. Example of hidden splat pins

having so many nuisance explains that it can explain all failing patterns. SLAT's multiplet search, for this table, would find pin 4, and stop, for all SLAT patterns are explained by it. All other splat pins would, in fact, be hidden by pin 4 and by the SLAT technique of only looking for minimal multiplets.

The second rule, that no two pins from the same splat can occur in the same multiplet, can also be violated, an example of which is shown in Table 13.. This table shows an example of nuisance fails in which two pins, pins 4 and 9, can together explain all the SLAT patterns, even though they belong to the same splat. Such a multiplet will be called abnormal. SLAT would find this abnormal multiplet, in addition to the usual ones, and rule 2 prevents it then from identifying the otherwise obvious splats.

These potential violations may decrease the confidence one has in SLAT diagnosis, but it is not clear how to avoid them. The problem of hidden splat

	pin 1	pin 7	pin 3	pin 4	pin 9
pattern 1	☒	☒			●
pattern 3	☒	☒		●	
pattern 2	●		☒	☒	☒
pattern 4			☒	☒	☒
	splat 1		splat 2		

Table 13. Example of an abnormal multiplet

pins is one that, in one form or another, all diagnostic approaches face, and was discussed in Section 3.3. Abnormal splats are peculiar to SLAT, but can be recognized when no more clusters can be merged without violating rule 2, and clustering stops prematurely. The example shown in Table 13., in fact, will force clustering to terminate when three clusters are found, rather than the required two.

4.2 *M* incomplete

Even though most SLAT diagnoses have complete sets if multiplet pins, there are exceptions. In this section, I will discuss some of the observed cases.

Several theoretical possibilities were indicated in the preceding sections for non-complete *M*s, among them abnormal multiplets and non-unique clusterings. I have not yet found an example of non-unique clustering. Abnormal multiplets and true non-complete *M*s are not very common, but do occur. An example of the latter is shown in Table 14.. The table is an explain fails table, with the rows and columns interchanged compared to previous explain fails tables. It is the result of a SLAT diagnosis of a medium sized ASIC, with some of the rows and columns removed that merely duplicate other rows or columns. The calculated splats are indicated by the alternate shading.

As this table, and its splat structure, were constructed from *M*, all multiplets can be obtained by taking one pin from each splat. On the other hand, the pin set (1, 2, 5) is not a multiplet, because it does not explain patterns d and e. Clearly, each multiplet has to have either pin 3 from the second splat, or pin 4 from the third splat to explain pattern e. If pin 3 occurs in the multiplet, any pin from the third splat will do; if pin 4, any pin from the second splat.

This example also shows how abnormal multiplets could happen. If patterns c and d had not been applied, for example, either pin 3 or pin 4 still would have to be in any valid multiplet, but the multiplet consisting of pins 1, 4 and 5 would now also be able to explain all the failing patterns. On the other

	pattern					
pin	a	b	c	d	e	f
1	☒					
2			☒			☒
3			☒	☒	☒	☒
4		☒		☒	☒	
5		☒				☒
6		☒				
7		☒				☒
8		☒		☒		☒

Table 14. Non-complete M

hand, this same example shows that abnormal multiplets might disappear if enough failing patterns were collected.

5 GREEDY SEARCH FOR SPLATS

The full SLAT process makes maximum use of the information available in the fail data. Its result is a set of splats, which is the most accurate estimate logic diagnosis can provide of the whereabouts of the defect that caused the fails. Its cost is that of simulating all the failing patterns.

An alternate, and less costly way of finding splats is to bypass the construction of multiplets, and to organize single pattern fault simulation in a greedy fashion. The flow is shown in Figure 42, and is an adaptation of a comparable figure in [21]. The adaptation consists of replacing faults by pins, because SLAT uses pins as the basic explanatory mechanism, not faults. A second adaptation is that no use is made of reduction modes, as they are unnecessary complications. No change was required, however, in the treatment of failing patterns that cannot be explained by single stuck-at faults, for both the diagnostic strategy of [21] and SLAT ignore them.

The search proceeds by attempting to diagnose an as yet unexplained failing pattern using single stuck-at faults. This step follows, and can benefit from any performance improvements that have been developed for standard logic diagnostic strategy. If successful, the pins that explain the fails for this

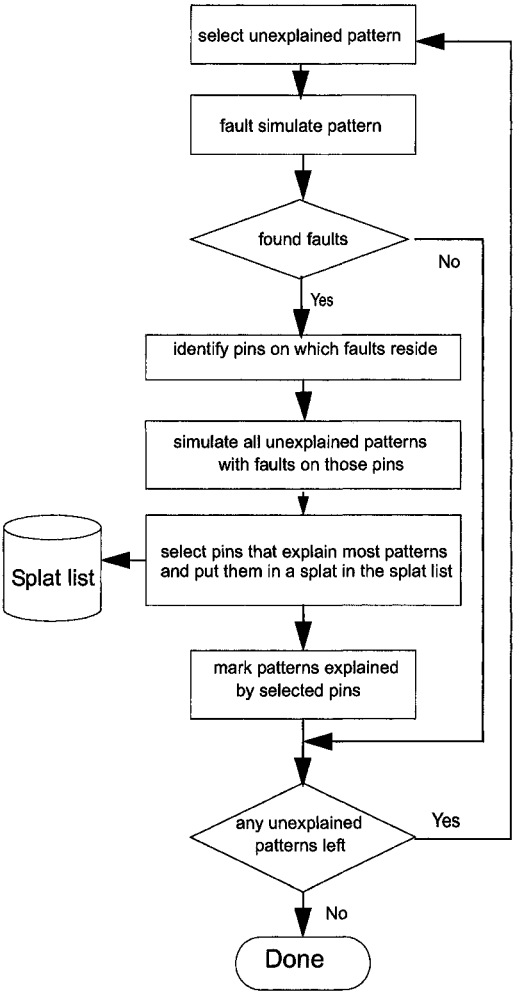


Figure 42 Greedy splat search

pattern are noted, and all not yet explained failing patterns are simulated with stuck-at faults on these pins. The number of patterns that each pin explains is calculated, and those pins are kept that explain the most patterns. Those patterns are marked as explained, and, if any unexplained pattern remains, the process starts over. If the diagnosis of the selected pattern is not successful, a non-SLAT pattern has been found, and, as in SLAT proper, it is not used further in the diagnosis.

Each set of pins kept at the end of a diagnostic phase corresponds to a splat. If the splats are completely separated, the sets of pins produced by the greedy algorithm are identical to the splats. In all other cases, there is the potential for misdiagnosis. The most important one is that one pin has a substantial number of nuisance explains, which add to the number of patterns explained by that pin. The greedy algorithm will then choose that pin, but not other pins in the splat with fewer nuisance explains. As a result, many of the pins in the splat, including the pin that is actually affected by the defect, may be missing. A second, related problem is when two pins explain many, but differing sets of patterns. When that happens, an arbitrary decision has to be made which one to keep, and which one to discard, with the obvious potential of making the wrong choice.

The main advantage of the greedy algorithm is its speed. It needs to simulate only few patterns with large sets of faults, most patterns being simulated with the much smaller sets of those faults that reside on pins that are already known to explain at least one pattern. The other advantage is that it is often successful, for many defects have completely separated splats. For a rough diagnosis, the greedy algorithm is the correct choice. In fact, no further work is required if its result is a single splat, because SLAT will not improve it.

When more than one splat is found, and when a more secure diagnosis is desired, however, for example, when the diagnostic call is to be used in physical failure analysis, SLAT should be used, for it makes maximum use of the information in the fail data.

6 INTERPRETATION

Using the notions of splats, we arrive at the following appealingly simple picture of a SLAT diagnosis. In the simplest but very common case, there are several splats, say n. We interpret this as a single defect that can affect n pins. It may or may not affect those pins simultaneously, but, when SLAT patterns are applied, only one of them is affected. The splat analysis of the diagnostic output has produced a large simplification. For example, in the design used in Figure 41, there are 40 multiplets, but only 1 defect if we consider pins belonging to the same splat as coming from the same defect.

When all failing patterns have the SLAT property, it is tempting to hypothesize that the pins in different splats are affected by the defect in a mutually exclusive fashion, but that conclusion does not follow necessarily from the SLAT diagnosis.

On the other hand, if there are failing patterns that do not have the SLAT property, it is obvious that the defect can affect multiple pins simultaneously.

Whether those pins are among the ones found by using SLAT patterns only cannot be decided by the SLAT diagnosis alone, although it is probably a good starting assumption.

The actual pins that can be affected by the defect are not known exactly, but each one is localized within its particular splat. The size of a splat depends on the structure of the logic around the failing pin. No further refinement in its identification can be made without using non SLAT patterns. In order to do that, however, a more detailed logical model of the defect is required (see Section 3.4).

The splat structure also provides guidance for subsequent failure analysis. As each splat is an estimate of one of the pins that can be affected by the real defect, finding (part of) the real defect can be done by inspecting the pins, and nets connected to those pins, of one splat only. This should of course be the smallest splat.

Once the defect has been located, one can verify that other nets affected by the defect are indeed connected to pins listed in the other splats. There are now three possibilities. First, all the affected nets are accounted for by the splats identified during diagnosis. This is the preferred outcome.

Second, some affected nets are not accounted for by the splats. This means only that the patterns that were applied were not enough to probe the defect in all its manifestations. But, as the defect was found anyhow, the diagnosis can still be called successful.

Finally, the affected nets cannot account for all the splats. This indicates that there are other defects on the device than the one just found. Another defect could be located by using one of the as yet unaccounted for splats, but finding the first defect may have removed the others. In that case, failure analysis is incomplete, even though the diagnosis is still a (partial) success.

7 EXPERIMENTAL RESULTS

The SLAT technique was compared previously [6] with the standard diagnostic technique described in the introduction, and the essential results will be reviewed briefly here. A recent publication [37] shows the results of applying SLAT to a large variety of simulated defects, with near perfect success.

The purpose of this section is to compare the efficiency of SLAT with the classical diagnosis based on stuck-at faults, and to demonstrate the success of SLAT in diagnosing real-life complex defects.

7.1 Comparison with stuck-at fault diagnosis

The first set of experimental results compare the overall efficiency of SLAT diagnosis with that of standard stuck-at fault based diagnosis. The vehicle was an ASIC design, described in more detail in the previous chapter and in Chapter 3. In this section, the results for Lot_3 will be used.

A total of 437 failing devices was used. SLAT diagnosis was also applied to all failing devices. To reduce excessive run times, the size of the multiplets was restricted to 7.

As SLAT does not use passing patterns, and not even those failing patterns that do not have the SLAT property, it is important to know how many failing patterns there were initially, and what fraction of those patterns had the SLAT property.

Figure 43 shows the distribution of the ratios of SLAT patterns to all fail-

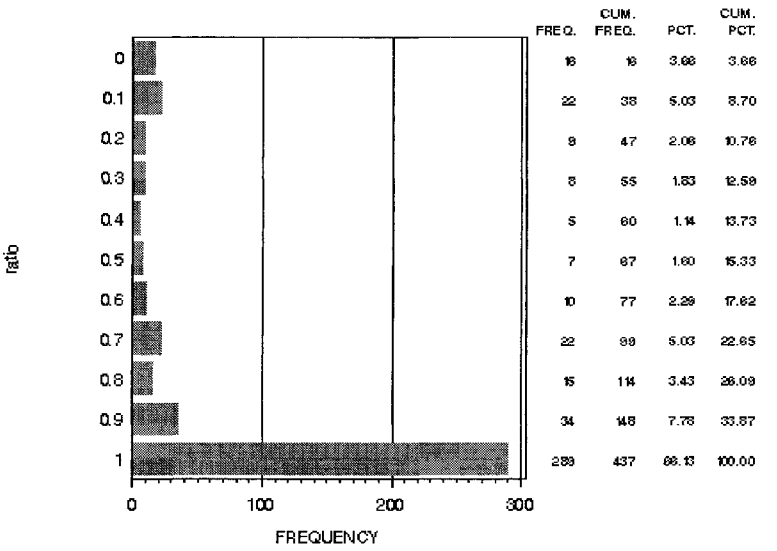


Figure 43 Distribution of fractions of SLAT patterns

ing patterns observed among the 437 devices. Finding a high ratio does not always indicate success. The diagnosis is only a borderline success when no multiplet of size not exceeding 7 is found, or when the size of the multiplets found by SLAT is the same as the number of SLAT patterns used to do the diagnosis. When that happens, we cannot have great confidence in the diagnosis, other than that SLAT found some SLAT patterns. Fortunately, in the majority of the cases, SLAT finds multiplets with sizes substantially less than

the number of SLAT patterns, with the excess SLAT patterns providing added confirmation that SLAT indeed found the correct defect.

Figure 43 clearly shows that in the majority of cases most of the failing patterns have the SLAT property. The figure does not show the absolute numbers of SLAT patterns, but for almost all devices this number ranges from well over ten to several hundred. It is also important to realize that even a small ratio does not doom diagnosis, because SLAT only requires a sufficient supply of SLAT patterns, not a large supply, or even that the majority of failing patterns have the SLAT property.

The actual distribution of multiplet sizes is shown in Figure 44. This figure

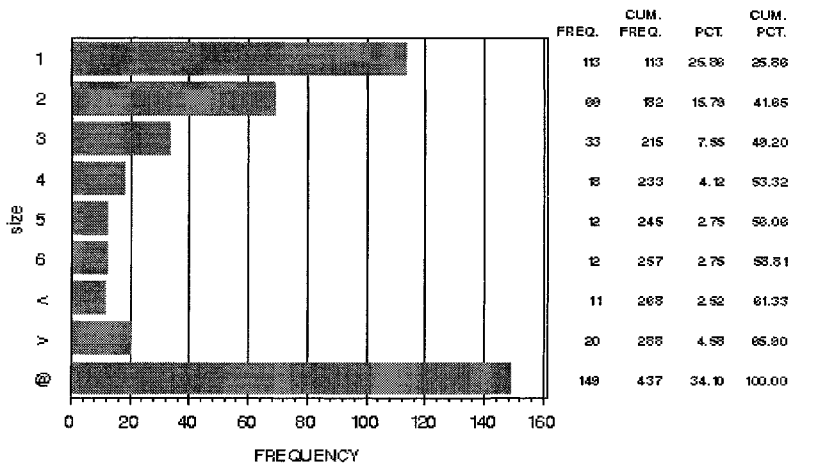


Figure 44 Distribution of multiplet sizes

demonstrates the much larger efficiency of SLAT compared to regular diagnosis. The symbols < and > indicate no SLAT patterns found and no multiplet of size less than 7 found, respectively. The group of size 1 diagnoses is divided into a group labeled @, indicating those devices for which vanilla diagnosis obtained a 100 score, and a remainder, indicated by 1.

The number of failing devices for which SLAT diagnosis found at least one multiplet is about 94% of the total number of failing devices, compared to an efficiency of about 34% for regular diagnosis (this being the fraction of devices for which regular diagnosis found at least one stuck-at faults that explained all the observed fails).

More strikingly, the number of failing devices for which SLAT diagnosis found multiplets of size 1 is about 60% of the total. Part of this 60% are the

devices for which regular diagnosis was successful as well, because a single stuck-at fault that explains all failing patterns will obviously be found by SLAT too. The remainder, about 26% of the total, are those devices in which the defect did affect a single node, but not in a consistent manner.

The relationship between vanilla scores and SLAT results is shown in Figure 45. The fraction of failing devices on which regular diagnosis was successful is somewhat larger than the fraction of devices for which SLAT found size 1, single polarity multiplets, since some of the devices for which SLAT found other size 1 multiplets may have pins that need only a single polarity stuck-at faults to explain all the SLAT patterns. Such devices would be counted among those for which regular diagnosis found at least one stuck-at fault, if all the failing pattern had the SLAT property. But regular diagnosis would ignore other single pins that can also explain all the failing patterns, be it with stuck-at faults of varying polarity.

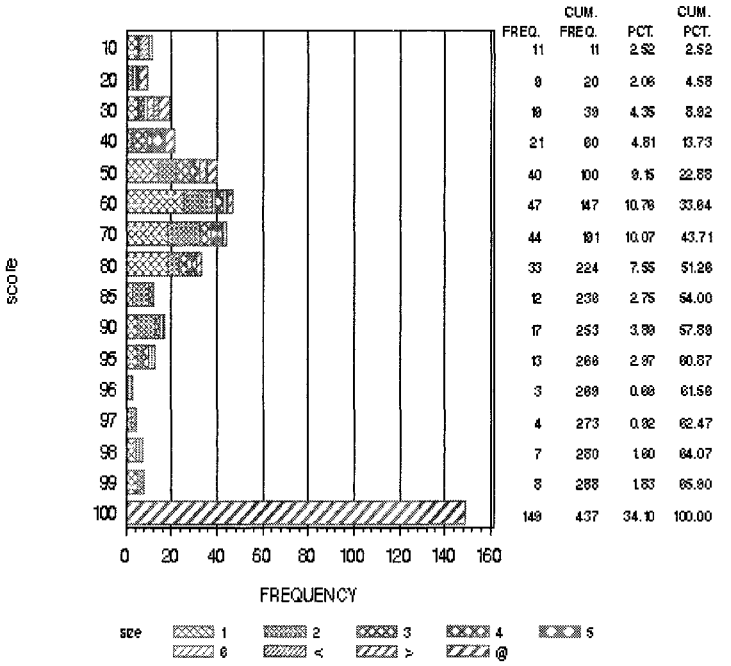


Figure 45 Correlation between vanilla diagnostic scores and SLAT sizes

The remaining 33% are devices in which the defect affected more than one pin. It is likely that there is a large number of bridges among those defects, but SLAT cannot show definitively that a two pin defect is in fact a bridge. There is some hint of bridges in Figure 45 which shows a distinct peak when the scores are around 50. This is expected to happen with bridges, because the

faults on the two legs of the bridge are will each explain about half of the failing patterns

As mentioned above, completeness is the norm. This is shown by the simple statistic that of the 437 devices for which SLAT diagnosis was successful, 91% had complete diagnoses. Of all the incomplete cases, one was caused by an abnormal multiplet, and the remainder had true incomplete multiplets.

7.2 Specific diagnoses

The diagnoses mentioned in the preceding section did not include actual verification by failure analysis. It is a comparison with standard single stuck-at fault diagnosis, and, to the extent that the latter leads to a successful root cause analysis when it is successful, SLAT will too. The comparison, however, does not show how well SLAT performs when the standard diagnosis is not successful, for example when SLAT finds more than one splat and at least of them small. Two examples of such SLAT diagnoses will be presented here.

The first one was done on a medium sized ASIC design, with about 1.2M gates, over 100K latches, and about 3M faults. SLAT found two splats, one with 2 pins, and one with 16 pins. The diagnosis was trustworthy because all failing patterns, 33 of them, were SLAT patterns. As there were two splats and no non-SLAT patterns, a bridge was expected. Failure analysis indeed found a bridge between two nets, one of the nets being connected to a pin in one splat, and the other net to a pin in the second splat. The failure analysis photograph is shown in Figure 46.

The second example was a small microprocessor, with about 1.5M gates, 76K latches and 1.9M faults. Ten failing patterns were available for diagnosis, but only four of them were SLAT patterns. This lack of SLAT patterns does not prevent diagnosis - after all, four SLAT patterns were still left - but indicates that the defect is a complex one. Based on these four, SLAT found a single splat, consisting of four pins. The logical behavior at these pins was not that of a stuck-at fault - some patterns needed a s@0 other a s@1 - reinforcing the observation that the defect is complex.

Failure analysis found that the defect affected three distinct nets, one of them connected to one of the pins in the splat, the other two being immediately upstream from those pins. No further logical analysis was done of the failing patterns and the logical portion of the design involved in the defect and the splat, but, given the small number of SLAT patterns, it is not surprising that not all nets affected by the defect were found by SLAT. Given the grossness of the defect, it is surprising that there were any SLAT patterns at all to enable SLAT diagnosis.

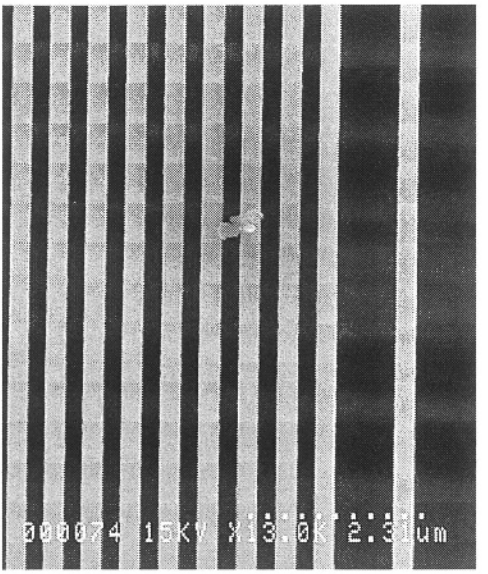


Figure 46 Root cause analysis of a bridge

Chapter 11

Data Collection Requirements

In the preceding chapters, various ways were described of using fail data to obtain information about the causes of the fails. The details of what data to collect, or how to collect it, were left out. These details are important, however, as the success of any analysis technique depends on the availability of the appropriate data. Such details will be addressed in this chapter.

Because the details of data collection require the design and the test sequence to have certain attributes, these requirements will be addressed first. Table 15. presents in abbreviated form the main conclusions from this chapter. The columns cover design, test, and data collection requirements as dictated by the various analysis techniques discussed in preceding chapters. The final column shows the main results that can be obtained from each type of analysis (compare with Figure 1). These results are the benefits that should outweigh the cost incurred when meeting the requirements in the earlier columns.

1 DESIGN REQUIREMENTS

The first diagnostic requirement on Integrated Circuits is that the state of the design can be observed immediately after that state has changed due to the application of some clock pulse. The design is assumed to be digital, so the state of the design is equivalent to the contents of the memory elements, and only clock pulses can alter that state. It is not required that the clock pulses can always be controlled externally, but it is required that, once a clock pulse has been issued, the state of the design can be frozen and observed before further clock pulses are applied. The easiest way to meet this requirement is to make the clock pulses externally controllable, and to design the circuit such that its state can be observed at any one time.

It is the responsibility of the designers to make sure that their designs support this diagnostic observability, and to generate the necessary observational test sequences (protocols). In the remainder of this section, I will describe some techniques that will simplify the observation of the state of a digital design.

In a digital design, the memory elements are generally of two kinds: latches and true memory elements, like the cells in embedded RAMs. The standard technique for making the latches observable is to connect them

Analysis	Design information	Test sequence	Data collection	Diagnostic result
Yield		Order of test steps that have individual sort codes	First fail codes for all devices	• Wafer fallout comparisons • Clusters of wafers with similar distributions of sort codes
Failing memories	• Connections between memories and their pass/fail and nofix bits • RAM attributes, like size and number of ports		All pass/fail and nofix bits	• Memory fail probabilities • Memory cell fail probabilities
Failing chains	• Lengths of chains • Types of latches in the chains		Pass/fail results for all chains	• Chain fail probabilities • Fail probabilities of different latch types
Commonality	Logic model (for some forms of commonality)		All failing latches for each pattern for which fails are observed	Clusters of similarly failing devices
Coverage		• Coverage at as many points in the sequence as possible • s@0 and s@1 on all pins	First failing pattern for each device that has not failed the tests preceding the LSSD tests	Defect distribution parameters
Logic diagnosis	Logic model	• Detailed test sequence • s@0 and s@1 on all pins	• All failing latches for each pattern for which fails are observed • If not all latches, indication of final latch that was observed (pass or fail) • At least ten failing patterns	Defect location(s)

Table 15. Summary of Data collection requirements

together into one or more scan chains [16, Chapter 3], and to make the inputs of the chains externally controllable and the outputs observable. If there are latches that are not in scan chains, provisions will have to be made to make them observable in some other way.

Embedded memories can sometimes be tested and diagnosed by mapping their inputs and outputs to chip Primary Inputs (PIs) and Primary Outputs (POs), and then applying special memory tests directly to the memory. If such is the case, there is no diagnostic problem. However, if there are many embedded memories, as there often are in today's designs, mapping the inputs and outputs of all the arrays to PIs and POs becomes cumbersome because of the added wiring and multiplexing. In that case, the memories need to be tested by special test sequences that are generated on the device itself. The observability requirements then imply that it has to be possible to interrupt testing immediately after any write clock has been applied, and to observe the contents of the array, or, at least, the contents of the address to which data were written last before any further write clocks are applied.

The easiest way to observe the contents of a memory word is to read them into scannable latches and then scan out their contents, but complex protocols may be required to accomplish such read operations. If such a strategy is followed, the chains should be designed such that, on switching from memory test mode to scan out mode, the contents of the relevant latches are guaranteed not to be affected.

It should not only be possible to observe the state of a digital design, but the time required to do so should not be excessive either, lest the collection of diagnostic data, even if feasible, is impractical. Scanning out the latch data can usually be done at a reasonable cost, since many testers are designed to handle scan designs. Collecting embedded memory data, however, is far more time consuming. The data collection time can be reduced if diagnostic data collection is taken into account when designing the RAMs, the BIST engines used to test the RAMs, and the manner in which they are imbedded in the surrounding logic.

Two important features have emerged in practice that greatly reduce the complexity of observing RAM contents. Both assume that the memory tests are generated by on-chip BIST engines.

1. When scanning out the contents of a RAM word, also scan out the address of that word, and, if possible, the state of the BIST engine. This additional information simplifies the interpretation of the data, for otherwise both the address and the particulars of the ABIST test sequence (forward or backward through address space, the type of

test, data or inverse data, ...) have to be extracted from the number of ABIST clock cycles that were applied since the beginning of the test.

2. Design the BIST engine such that the state of the engine is preserved when data (and address) are scanned out. This makes it possible to continue applying BIST patterns after scan out has completed, without having to restart the BIST engine.

Observing the state of the device is not the only activity that has strong design implications. Making sense of the observed state adds additional requirements on the design. In particular, as embedded memories and scan chains can be treated as objects that can pass or fail appropriate tests, all the theory presented in Chapter 5 can be used to extract, for example, cell fail probabilities. To do so, however, one needs to be able to determine unambiguously whether a given object passes or fails its associated tests.

For scan chains, this is usually not a problem if the outputs of the chains can be observed from the tester. If, as may be the case in future designs [4, 46, 63], these outputs feed on-chip compactors, like MISRs, alternative methods will have to be found to determine whether any specific scan chain passes or fails the chain tests. One potential method is a diagnostic test mode in which the same chains are connected to POs rather than to compactor inputs. If there are too many chains for the available number of POs, a sequence of test modes may have to be defined such that each chain output is connected to some PO in at least one of these test modes.

Embedded memories, on the other hand, are rarely made observable at POs. Instead, their passes and fails are monitored by pass/fail bits stored in latches. These pass/fail bits can be calculated by the BIST engine itself during the application of the memory tests. When the memory contains redundant rows, and sometime even columns, an additional bit indicates whether the memory, even though defective, can be repaired. Clearly, to know which memories passed the memory tests, the relation between the pass/fail bits and the memories should be unambiguous. This means that each pass/fail bit should be related to only one embedded memory. Likewise, the relation between the so-called nofix bits and the memories with redundant elements, should be unambiguous. *It is the responsibility of the designers to report this relation to the diagnostic engineers*, but establishing the connection between a pass/fail bit and its associated memory can be made easier by clearly labeling that memory block in the logic model of the design.

The state of the design at various points in the test sequence is undoubtedly the most important diagnostic quantity to be obtained during data collection. Nevertheless, there are various design attributes that can also help when interpreting the collected data. For example, for instance analysis it may

be useful to know the sizes of the objects, and, if different cell designs are used, to know what types of cells are used in the various objects. Furthermore, different RAM architectural features, like the number of ports, may have a noticeable impact on the yield, and should be known as well. All these design level attributes should be available to the diagnostic engineer to make maximum use of the available diagnostic data.

2 TEST REQUIREMENTS

First, the test program that was applied to any failing device should be known to the diagnostic tools. It has to be known in detail, down to the precise sequence of test patterns used in the deterministic portion of the test. This requirement remains in force even if the patterns are generated on-chip by some BIST engine, but is then easier to satisfy, because only the structure of the pattern generator and the initial seed need to be known.

This knowledge is required as the diagnostic engineer needs to know which parts of the tests did not uncover a defect, in addition to what parts did. In logic diagnosis, for example, passing patterns can be used in some cases to increase the accuracy of the diagnostic call (see Chapter 9). Another example is commonality analysis, in which both passes and fails of particular tests may be compared.

The collected fail data will tell us about non-fails, but only if the latter can be deduced from the former by implication: a device did not fail a particular test if the test sequence is followed exactly, and the device failed a test later in the sequence. Changes in the test sequence are frequently made, however, for example to reduce the test application time, and relying on implication is not a safe practice. Instead, the test sequence that was actually applied should be known explicitly. Alternatively, the notion of fail data can be enlarged to include passing tests as well. In that case, fail data become test results, and include for each part of the test sequence whether any genuine fail data were collected for that part. A record of no fail data for any particular part of the test then indicates a pass for that part of the test sequence.

Even though many of the analysis techniques described in this book do not depend on the details of the tests, only on whether or not a device failed the tests, coverage analysis (Chapter 8) and logic diagnosis (Chapter 9 and Chapter 10) engender additional requirements on how the tests were generated. To explain these requirements, I will assume here that stuck-at faults were used for test generation, as this is common practice. If other faults were used, similar comments apply to them.

Underlying test generation is a description of the design, called the logic model. This description is a logic abstraction of the design that mimics, as faithfully as possible, the actual design details, like its wires and basic functional blocks. The faults, used in test generation and diagnosis, are objects in this logic model, and need to be attached to other objects in the model. For test and diagnosis to be effective, both s@0 and s@1 faults should be attached to all identifiable pins in the model, in which pins are the points where nets are connected to logic blocks or IOs. If a net has a complex fanout structure, the fanout points should be replaced by fanout boxes and the branches of the net connected to either these fanout boxes or to the source and sinks of the original net. Faults can then be attached to the pins on the fanout boxes or to the pins on the source and sinks of the net. Of course, this only makes sense if the actual fanout structure is known. If not, the best one can do is to attach faults to the pins where the net is connected to its source and sinks.

For SLAT - Chapter 10 - only stuck-at faults are required. For test generation, or regular logic diagnosis - Chapter 9 - other faults can be used, like shorts. Requirements for shorts are much less clear-cut than for stuck-at faults, but they should include at least those between nets that run parallel at minimum distance for a distance that exceeds some predetermined threshold.

The final requirement on the test sequence is that the coverage be known at as fine a granularity as possible; preferably for each test pattern separately. Because the coverage of a sequence depends on what faults are not yet uncovered, given the other sequences that are applied prior to it, it should be calculated for the detailed test sequence that was applied on the tester.

3 DATA COLLECTION REQUIREMENTS

The fail data collected for diagnostic purposes have to meet various requirements. They should be appropriate for the task at hand, that is, they should be adequate, in type and volume, for the desired form of analysis. To do diagnosis of failing objects, for example, pass/fail information is required for all the embedded objects, and that information has to be collected for a large sample of failing devices to make statistical analysis of the fails meaningful. On the other hand, the results of a scan chain integrity test are not appropriate for logic diagnosis.

The collected data should also be complete, meaning that all the data, assumed by the diagnostic technique, is collected. In logic diagnosis, for example, it means that all the failing latches for the patterns employed by the diagnosis are known, not just the first n, where n is some number that is determined in practice by tester limitations or by test time requirements. Of course,

if the data is not complete for one type of analysis, it may still be complete for another. Logic diagnosis can still be performed if the number of n were known, be it with greatly reduced accuracy. For such a diluted form of diagnosis, completeness has changed meaning, and indicates now that indeed all the failing latches or n failing latches, whichever is smaller, are available.

Finally, the collected data should be predictable, or, in other words there should be a clear, well understood protocol for collecting fail data, one that meets some agreed upon appropriateness and completeness requirements, and such that the analysts at the receiving end of the fail data collection can be assured that the data they use is collected in accordance with the protocol.

Table 15. attempts to summarize the requirements on fail data collection that are appropriate, complete and predictable. For example, pass/fail results need to be collected for all embedded objects for the sake of completeness. Likewise, failing latches should be completely collected for each pattern for which fail data are collected; in other words, no partial scan-outs. Another requirement is that the test step at which a device failed for the first time, given the test sequence, be known for every device. For some devices, this test step will be known only in a post-test disposition step; for example, when the device failed no specific test, but when, instead, some combination of performance tests was outside a predetermined acceptance region.

Finally, for some fraction of devices, all the failing test steps need to be known. In practice, no further testing can be performed if the device fails one of the initial gross test steps, like contact, leakage or probe-melt, but the response of the device to all subsequent tests can be known. Of course, there should also be some indication in the fail data that tells whether, for a given device, such extended test data collection was performed.

Appendix A

Distribution of IC Fails

1 GENERAL DEFINITION

The binomial and multinomial distributions were introduced in Chapter 2.1.1. Given the d_k for k = 1, ..., k_f, the probability $P(N_1, ..., N_{k_f})$ of N_i chips failing test i is given by the multinomial form

$$\frac{N!}{N_{pass}! \prod N_i!} \left(1 - \sum d_i\right)^{N_{pass}} \prod d_i^{N_i}, \tag{A.1}$$

where n! stands for the factorial of n and all sums and products are from i = 1 to k_f. By summing over all values N_i for all i except one, say j, we find that the probability that test k_j fails N_j chips out of a total of N chips equals $b(N_j;N,d_j)$.

A related special case is that of all tests from test 1 to some test k grouped into one test. The probability y_k of passing all tests through the k^{th} one is

$$y_k = 1 - \sum_{i=1}^{k} d_i \tag{A.2}$$

Therefore, K, the number of chips passing all tests 1 through k, as well as N-K, the number of chips failing one of those tests, have the probability density function

$$b(K;N,\ y_k) = b(N-K;N,\ 1-y_k)\ . \tag{A.3}$$

The number of chips that fail at test k does depend on the outcome of the previous tests. Consider the case that the tests 1 through j-1 found N-K chips to be defective. There are therefore K chips left to be tested by tests j through k_f. Summing Equation (A.1) over all values of N_i, $1 <= i < j$, with the condition that the sum over all those N_i equals N-K, shows that the conditional probability of N_i chips failing the i^{th} test, for all $j <= i <= k_f$ and given that exactly K chips passed the first j-1 tests, equals

$$\frac{K!}{\left(K-\sum N_i\right)!\prod N_i!}\left(1-\sum \bar{d}_i\right)^{K-\sum N_i}\prod d_i^{N_i}, \tag{A.4}$$

where now all the sums and products are from i = j to k_f, and the new detection probabilities $\bar{d}_i$ are related to the original d_i by

$$\bar{d}_i = d_i/y_{i-1}\,. \tag{A.5}$$

This probability has again the multinomial form, and, consequently, N_i, the number of chips failing test i for any particular $i >= j$, has again the binomial distribution with probability density function

$$\text{Prob}(N_i|K) = b(N_i;K,\ \bar{d}_i)\ . \tag{A.6}$$

Focusing on the fails starting with the j^{th} test and ignoring the preceding ones, therefore amounts to nothing more than a rescaling of the first fail probabilities. The corresponding yields are rescaled as well according to

$$\bar{y}_i = y_i/y_{j-1}\,. \tag{A.7}$$

1.1 Fallout fluctuations

N_i can be written as the sum

$$\bar{d}_i K + \delta N_i(K) = \bar{d}_i \langle K \rangle + \bar{d}_i \delta K + \delta N_i(K), \tag{A.8}$$

in which the three terms on the right represent, respectively, the expected value of N_i averaged over all values of K, fluctuations in N_i due to fluctuations in K, and fluctuations in N_i around $\bar{d}_i K$. In other words, fluctuations in the value of N_i have two causes: the center of the distribution, $\bar{d}_i K$, fluctuates because K fluctuates, and the actual value of N_i fluctuates around this center.

The latter fluctuations are usually much larger, however, than the former, and the fluctuations in K can generally be ignored. This can be shown using equations (A.3) and (A.6). The expected values of the two fluctuation terms are, of course, zero. Their variances are $\bar{d}_i^2 N y_{i-1}(1 - y_{i-1})$ and $N y_{i-1} \bar{d}_i (1 - \bar{d}_i)$, respectively. The fluctuations in K can then be ignored if $\bar{d}_i (1 - y_{i-1}) \ll (1 - \bar{d}_i)$, or if $d_i (1 - y_{i-1}) \ll y_i$. The identity $d_i = y_{i-1} - y_i$ and some algebra reduce the inequality to

$$\frac{y_{i-1}(1 - y_{i-1})}{2 - y_{i-1}} \ll y_i . \tag{A.9}$$

If y_{i-1} is close to 0, this inequality becomes $y_{i-1} \ll 2y_i$, while near 1 it becomes $1 - y_{i-1} \ll y_i$. The maximum of the left hand side of Equation (A.9) occurs at $y_{i-1} = 2 - \sqrt{2}$, where y_i should be much larger than $3 - 2\sqrt{2} \approx 0.2$ for inequality (A.9) to hold.

In general, the inequality will hold if y_i is not too small compared to y_{i-1}. How small is too small depends on y_{i-1}. When y_{i-1} is near 1, y_i can have almost any value. For other values of y_{i-1}, as long as it does not differ too much from y_i, fluctuations in K can be ignored.

1.2 Defect Level

The final issue to be considered in this Appendix is that of the distribution of the defective chips that pass all tests. All chips fall into three buckets: the chips that are defective but are caught by the tests, the chips that are defect free, and the chips that have defects but pass the tests. The probability that a defective chip will be caught equals 1 - y, where y is the expected yield from the test. The probability d_n that a defective chip is not caught equals y - y_0, where y_0 is the probability of the chip being defect free.

Using Equation (A.6), we find that the probability $P(N_{def}|N_{pass})$ of not catching N_{def} defective chips, given N_{pass}, equals $b(N_{def};N_{pass},\bar{d}_n)$, with $\bar{d}_n = d_n/y$. The expected value of N_{def} is therefore

$$\langle N_{def} \rangle = N_{pass}\bar{d}_n = N_{pass}\left(1 - \frac{y_0}{y}\right), \tag{A.10}$$

while its variance is given by:

$$\sigma^2(N_{def}) = N_{pass}\bar{d}_n(1 - \bar{d}_n), \tag{A.11}$$

which is approximately equal to $<N_{def}>$ when $\bar{d}_n$ is small.

Appendix B

General Yield Model

The easiest way to derive the properties of the general yield model is by calculating first the generating function G(z). Let A be the area of the chip and P_n the probability that the chip contains n defects. G(z) is defined as the expectation value of z^N, where the random variable N is the number of defects on a chip. Or,

$$G(z) = \sum_n z^n P_n . \tag{B.1}$$

The actual value of N has two contributors: first, the distribution of primitive polluters, and, second, the number of defects produced by each polluter. The generating function can, therefore, be written as

$$G(z) = \langle \sum_n z^n p(N = n | \{\nu(\vec{r})\}) \rangle , \tag{B.2}$$

in which identity p is the probability that there are n defects, given the specific distribution of primitive polluters, indicated by $\nu(\vec{r})$, and < ... > indicates averaging over all those distributions. As the defects produced by the primitive polluters are independent and random, the expectation value of z^N, given $\nu(\vec{r})$, equals

$$e^{(z-1)\int \nu(\vec{r}) d\vec{r}} . \tag{B.3}$$

Taking the expectation value of this with respect to all distributions of primitive polluters leads to the central result that

$$G(z) = \langle e^{(z-1)\int \nu(\vec{r})d\vec{r}} \rangle . \tag{B.4}$$

Moments of the distribution of N can be obtained by differentiating equation (B.4) with respect to z at z = 1 [15]:

$$G(z) = 1 + \mu(z-1) + \frac{1}{2}\langle n(n-1)\rangle (z-1)^2 + \ldots . \tag{B.5}$$

The expectation value of n is then found to be $\mu = \langle \int \nu(\vec{r})d\vec{r} \rangle$, and the variance of n equals

$$\sigma^2(n) = \mu + \langle \int \left(\nu(\vec{r}) - \frac{\mu}{A} \right)^2 d\vec{r} \rangle . \tag{B.6}$$

Let us now derive some more detailed properties of y_0. As it is obviously equal to p_0, we find that

$$y_0 = \langle e^{-\int \nu(\vec{r})d\vec{r}} \rangle . \tag{B.7}$$

To simplify the equations, define

$$E(f) = \frac{1}{y_0} \langle e^{-\int \nu(\vec{r})d\vec{r}} f \rangle \tag{B.8}$$

for any expression f that depends on the actual distribution of primitive polluters. Let us also define

$$\Sigma^2(f) = E((f - E(f))^2) . \tag{B.9}$$

We study changes in the area by changing A by a small amount δA. Two important examples are first that δA is a small narrow band of vanishing width around the periphery of the chip. This example is important when we

consider chips with slightly larger (positive δA) or smaller (negative δA) areas. The second example is more specialized, but is important when analyzing the effect of non-uniform defect coverages. In that case, δA is a small area, like a circle or rectangle, of vanishing area, somewhere inside the chip. It will turn out that different ways of changing the area of the chip will have different consequences for the yield.

Let us now write the integral over the area of the chip as the sum of two integrals: one over A and one over δA. For convenience, we write

$$\delta\mu = \int_{\delta A} \nu(\vec{r})d\vec{r}, \tag{B.10}$$

and expand in powers of it:

$$\begin{aligned} y_0(A+\delta A) &= y_0(A)E(e^{-\delta\mu}) \\ &= y_0(A)\left(1 - E(\delta\mu) + \frac{1}{2}E(\delta\mu^2) + \ldots\right) \end{aligned} \tag{B.11}$$

From this, we easily obtain

$$\ln y_0(A+\delta A) = \ln y_0(A) - E(\delta\mu) + \frac{1}{2}\Sigma^2(\delta\mu) + \ldots \tag{B.12}$$

Now, assume that δA shrinks in some fashion, such that $\frac{1}{\delta A}\int_{\delta A} \nu(\vec{r})d\vec{r}$ is finite and well defined. Let us call this limit $\nu_{\delta A}$. Then, in the limit of small δA, $\delta\mu \approx \delta A \cdot \nu_{\delta A}$, and $\ln y_0(A+\delta A)$ equals

$$\left(\ln y_0(A) - \delta A \cdot E(\nu_{\delta A}) + \frac{1}{2}(\delta A)^2\Sigma^2(\nu_{\delta A}) + \ldots\right) \tag{B.13}$$

It is important to realize that $\nu_{\delta A}$ depends on how δA goes to zero. When it is well defined, we find

$$\frac{\partial \ln y_0(A)}{\partial \delta A} = -E(\mu_{\delta A}), \tag{B.14}$$

and

$$\frac{\tilde{d}^2 \ln y_0(A)}{\tilde{d}\delta A^2} = \Sigma^2(\mu_{\delta A}), \tag{B.15}$$

where the ~ over the d indicate that the derivatives are meaningful only for specific changes in the area.

From these equations one can immediately conclude that the logarithm of the yield never increases, because its first derivative is negative, but that its rate of decrease diminishes, because its second derivative is positive.

For very small A, $\delta\mu \approx \nu(\vec{mer} = \vec{0})\delta A$, and, setting A equal to 0,

$$\ln y_0(\delta A) \approx -E(\delta\mu) + O(\delta\mu^2) \approx -\langle \nu(\vec{0}) \rangle \delta A, \tag{B.16}$$

for $y_0(0)$ equals 1. This is the same as for a Poisson distribution with the same average number of defects per chip. In general, therefore, the logarithm of the yield starts out as a linear function of A, as in the Poisson case, but then starts deviating from this linear function, such that it is larger than the Poisson result, with the difference growing with A.

Appendix C

Simplified Center-Satellite Model

The center-satellite model has been treated in fairly great detail by Meyer and Pradhan [39], and I will closely follow their technique for averaging over all cluster configurations. Their more general dependencies on time t and wafer quality w will be ignored, however. Their locations x, y will be indicated by $\vec{r}$.

The central quantity to be calculated is the generating function of the number of defects on the chip, called G(z). The contribution to G(z) from the clusters is called $E[z^{K_A}]$ by Meyer and Pradhan. E[f] in their notation indicates the expectation value of f, and K_A is the number of defects in area A.

To calculate the generating function of K_A, we first need to calculate the probability that a defect produced by a cluster centered at $\vec{r}$ falls within the area of the chip. This probability is

$$\alpha_A(\vec{r}) = M \int_A I_A(\vec{r}') f_D(\vec{r}'|\vec{r}) d\vec{r}' . \qquad \text{(C.1)}$$

I_A equals 1 when $\vec{r}'$ is within the area of the chip and equals 0 otherwise. $f_D d\vec{r}'$ is the probability that a defect produced by a cluster whose center is at $\vec{r}$ is found in $d\vec{r}'$. In the simplified center-satellite model studied here, the defects are distributed uniformly within the area of the cluster, and, therefore,

$f_D(\vec{r}'|\vec{r}) = C^{-1} I_C(\vec{r}')$ as the area of the cluster is C. I_C equals 1 when $\vec{r}'$ is within the area of the cluster and equals 0 otherwise. The essential simplification obtained by considering circular chips is that the integration required to get $\alpha_A(\vec{r})$ can now easily be done numerically, as there are no problems with awkward shapes and equally awkward orientations.

Next, let $D_A^1(\vec{r})$ be the number of defects in A given a single defect produced by a cluster at $\vec{r}$. This number is either 0 or 1. Likewise, let $D_A(\vec{r})$ be the number of defects in A caused by the same cluster. Meyer and Pradhan then show that

$$E\left[z^{D_A^1(\vec{r})}\right] = 1 + (z-1)\alpha_A(\vec{r}), \tag{C.2}$$

and

$$E\left[z^{D_A(\vec{r})}\right] = \sum_{d=0}^{\infty} F_D(d|\vec{r})\left(E\left[z^{D_A^1(\vec{r})}\right]\right)^d, \tag{C.3}$$

in which $F_D(d|\vec{r})$ is the probability that the cluster has d defects. Let us assume that the defects are uniformly distributed within the area of the cluster, and with strength ν. Consequently,

$$F_D(d|\vec{r}) = \frac{n_C^d}{d!} e^{-n_C}, \tag{C.4}$$

and

$$E\left[z^{D_A(\vec{r})}\right] = e^{(z-1)n_C \alpha_{mpA}(\vec{r})} \tag{C.5}$$

As the cluster can be anywhere with equal probability, we should average the cluster location over all space to get K_A^1, the number of defects in A caused by a single cluster. The cluster, however, can clearly not influence the chip when its center is more than ρ+R away from the center of the chip. We,

therefore, have to average only over an area S, which is a circle with the same center as that of the chip, and with radius ρ+R. Consequently,

$$E\left[z^{K_A^1}\right] = \frac{1}{S}\int_S e^{(z-1)n_C\alpha_A(\vec{r})}d\vec{r}, \tag{C.6}$$

which will be abbreviated to Q(z). As the clusters are uniformly distributed within S with strength λ, we finally get

$$E[z^{K_A}] = e^{-\lambda S(1-Q(z))}. \tag{C.7}$$

Finally, G(z) is equal to

$$e^{-\lambda S(1-Q(z))-\nu_0 A}, \tag{C.8}$$

obtained by multiplying Equation (C.7) by the generating function for the uniform background

The moments of $\{P_n\}$ are most easily obtained by expanding G(z) in powers of (z - 1):

$$1+\mu(z-1)-\frac{1}{2}\langle n(n-1)\rangle(z-1)^2+\ldots. \tag{C.9}$$

Let E(f) for any function f of $\vec{r}$ be short for

$$E(f) = \frac{1}{S}\int_S f(\vec{r})d\vec{r}. \tag{C.10}$$

Expanding Q(z) in powers of (z - 1) gives

$$1+(z-1)n_C E(\alpha_A)+\frac{1}{2}(z-1)^2 n_C{}^2 E(\alpha_A^2)+\ldots. \tag{C.11}$$

Using this expansion, we find for G(z)

$$1+(z-1)\Lambda+\frac{1}{2}(z-1)^2(\lambda S n_C^2 E(\alpha_A^2)+\Lambda^2)+\ldots, \tag{C.12}$$

with

$$\Lambda = \lambda S n_C E(\alpha_A) + \mu_0 A. \tag{C.13}$$

For very small chips, we need $\alpha_A(\vec{r})$ when R goes to zero. Clearly, when the center of the cluster is at a distance of more than ρ+R from the center of the chip, $\alpha_A(\vec{r})$ equals 0. On the other hand, when this distance is less than ρ-R, $\alpha_A(\vec{r})$ equals R^2/ρ^2, the ratio of the size of the chip and the size of the cluster. As the change from R^2/ρ^2 to 0 occurs over a very small distance range between ρ-R and ρ+R, we can ignore this smooth variation and, instead, approximate $\alpha_A(\vec{r})$ by a step function, with the step occurring at distance ρ. Similarly, we can approximate the area of the region S by C.

For very large chips, a similar calculation can be made. Now, however, S equals A and $\alpha_A(\vec{r})$ equals 1 when the cluster is within distance R-ρ of the chip. We again ignore the smooth variation between R-ρ and R+ρ and approximate $\alpha_A(\vec{r})$ by a unit step function.

Appendix D

Quadrat Analysis

Let us start the analysis by considering a contiguous group of k chips on the wafer. Such groups are usually called quadrats, although they need not consist of four chips. These groups can have any number of passing devices between zero, when all devices fail the tests, and k, when all devices pass the tests.

We will consider the general case of the gross yield y_S not equal to 1. We need to assume that the distribution of defects over that portion of the wafer that is not affected by gross defects can have some clustering, but that the range over which this clustering happens is large compared with the size of the groups. Consequently, the distribution of defects over the area of a group is Poisson with some strength ν, and the clustering can be modeled by compounding a Poisson distribution. This assumption on the one hand restricts the range of applicability of the quadrat analysis, but, on the other hand, creates many relations between the various quantities that can be observed, as will become clear in the sequel.

1 ESTIMATION

The number of quadrats is M, and a quadrat has k chips, each having area A. The probability that a chip in a given quadrat that is not a wipe out is good will be indicated by $p = e^{-\nu A}$, with ν the strength of the Poisson distribution in that quadrat. The dependence of p on the Poisson strength ν will be assumed.

The clustering parameters will be estimated using the observed fractions $P_n^{(k)}$, which are the ratios of the numbers of quadrats with n good chips and M. Their expectation values are $p_n^{(k)}$, and equal

$$y_S\binom{k}{n}\int h(\nu)p^n(1-p)^{k-n}d\nu + (1-y_S)\delta_{0n}, \tag{D.1}$$

with h(ν) the compounder of the Poisson distribution, and δ_{0n} the Kronecker delta function. The first term in Equation (D.1) corresponds to the quadrats that are not wipe outs, and the second term to those that are. The latter term only contributes to $p_0^{(k)}$, as explained in the main text. By expanding the 1-p factor in Equation (D.1), the first term can be written as

$$y_S\binom{k}{n}\sum_{j=n}^{k}\binom{k-n}{j-n}(-1)^{j-n}\int h(\nu)e^{-j\nu A}d\nu \quad . \tag{D.2}$$

Note that

$$p_j^{(j)} = y_S\int h(\nu)e^{-jA\nu}d\nu. \tag{D.3}$$

Because $p_j^{(j)}$ is the yield of a quadrat of size j, it will also be indicated by y(jA). Despite appearances to the contrary, y_S and h(ν) appear in Equation (D.1) only in combinations of the form y(jA), with j larger than 0. In fact, as $p_i^{(k)}$ for i not equal to k is a linear combination of $p_j^{(j)}$ for j = 1, ... k, it suffices to consider only the latter.

Conversely, we can easily deduce from Equation (D.2) that

$$(y(kA), \ldots, y(A))^T = \bar{Q}(p_k^{(k)}, \ldots, p_1^{(1)})^T, \tag{D.4}$$

with $(a, \ldots, b)^T$ the transpose of $(a, \ldots, b)$, and $\bar{Q}$ the transformation matrix. For example, for k = 4, $\bar{Q}$ equals

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1/4 & 0 & 0 \\ 1 & 1/2 & 1/6 & 0 \\ 1 & 3/4 & 1/2 & 1/4 \end{bmatrix}. \tag{D.5}$$

Note that all the entries in the matrix are positive, and that, for the same column, they increase with row number. Consequently, when y(jA) is obtained from the $p_i^{(k)}$, it is guaranteed to be non-negative, and not to increase with j.

We now use the same matrix to obtain statistics $P_j^{(j)}$ from the observed $P_n^{(k)}$. These statistics are analogous to the observed yields of quadrats of size j, and are referred to as pseudo yields. Like real yields, they will also be indicated by Y(jA). The expectation values of the pseudo yields are still y(jA).

In order to use these equations with experimental data, we have to make a choice for the compounder. The Gamma function (Equation (2.8)) is the usual one. For this compounder,

$$\int h(\nu) e^{-jA\nu} d\nu = \left(1 + \frac{j\nu A}{\alpha}\right)^{-\alpha}. \tag{D.6}$$

α^{-1} is a more useful quantity to work with than α, and will be indicated by γ. It is also useful to replace the combination of variables $\nu A\gamma$ by β. This change of variables will not cause problems, for the corresponding Jacobian and its inverse are singular only at $\gamma = 0, \infty$ and $\beta = 0, \infty$, which can easily be avoided.

The proper way to estimate the parameters of the distribution is to use maximum likelihood on, for example the observed values $P_j^{(4)}$, for j running from 1 through 4. This, however, leads to complex non-linear equations, and is not usually done. The simplified way of estimating the parameters is to choose three of the four pseudo yields, and then calculate the parameters from them. The traditional choice for these three yields is Y(A), Y(2A) and Y(4A).

The gross yield y_S, the inverse cluster coefficient γ and β can now be obtained as follows. From Equations (D.3) and (D.6), and some simple algebra, we find that

$$y(4A)^{-\gamma} - 3y(2A)^{-\gamma} + 2y(A)^{-\gamma} = 0. \tag{D.7}$$

When we replace the yields y by the observed pseudo yields, Equation (D.7) becomes an implicit equation for γ. It has a trivial solution at $\gamma = 0$, which cannot be used, however, for that is a singular point of the Jacobian. To find the real solution, let us write the left hand side of Equation (D.7) as $f(\gamma)$, and let us consider what happens if γ goes to $\pm\infty$. Of the three quadrat yields, we expect Y(4A) to be the smallest and Y(A) to be the largest. Consequently, when γ goes to ∞, the contribution from Y(4A) will dominate, and, when γ goes to $-\infty$, the contribution from Y(A) will dominate. In both cases, $f(\gamma)$ will be positive. At $\gamma = 0$, however, $f(\gamma)$ changes sign. There must, therefore, be at least one other solution to get $f(\gamma)$ back to the proper sign.

We can determine on which side of $\gamma = 0$ this solution is situated by evaluating the derivative of $f(\gamma)$ at $\gamma = 0$, where it equals

$$-\ln Y(4A) + 3\ln Y(2A) - 2\ln Y(A). \tag{D.8}$$

The value of this derivative can be determined easily. When negative, a solution exists for positive γ. Otherwise, a solution will have to be found for negative γ.

Given γ, the gross yield y_S can now be estimated with

$$y_S = (2Y(A)^{-\gamma} - Y(2A)^{-\gamma})^{-1/\gamma}, \tag{D.9}$$

and β with

$$\beta = -(Y(A)^{-\gamma} - Y(2A)^{-\gamma})y_S{}^{\gamma}. \tag{D.10}$$

The desired standard parameters are

$$\alpha = \gamma^{-1} \quad \text{and} \quad \nu A = \alpha\beta. \tag{D.11}$$

Unless A is known, ν cannot be determined separately.

As the pseudo yields are in general not equal to their expectation values, it may not be possible to fit them with a negative binomial function, even assuming that the corresponding expectation values can. Several anomalies are possible. First, when Y(4A) is zero, there is a unique solution with $\beta = -0.25$, and

$$\gamma = \ln(1.5)/\ln(Y(2A)/Y(A)), \tag{D.12}$$

which is always negative. When Y(2A) is zero as well, however, there is in general no solution. Wafers for which Y(2A) vanishes should, therefore, be removed from consideration.

Anomalies with the calculation of y_S are harder to deal with. Clearly, when $Y(A)^{-\gamma} - Y(2A)^{-\gamma}$ is negative, no meaningful solution is possible. On the other hand, there is no limit on how large y_S can become when $Y(A)^{-\gamma} - Y(2A)^{-\gamma}$ is positive. Because y_S is a yield, it should not be larger than 1.0. Due to the statistical fluctuations in the observed pseudo yields, some level of violation should be allowed, but there is no natural threshold that y_S should not be allowed to exceed. In practice, I have set the threshold at 2.0.

Equation (D.10) shows that, when y_S is positive, β has the same sign as γ. This is exactly what is needed to make the ratio β/γ positive in all circumstances. Consequently, there are no anomalies associated with the calculation of β that have not yet been addressed in the calculation of y_S.

2 GENERAL EQUATION FOR THE CLUSTER COEFFICIENT

That γ is a solution of Equation (D.7) was derived for the negative binomial distribution, but happens to be approximately valid for a much wider class of distributions. In particular, when the clustering is weak, solutions of this equation correspond to the inverse of the cluster coefficient $\sigma^2(\nu)/\nu_0^2$ discussed in Chapter 2.1.2 for compound Poisson distributions. This result will be demonstrated in this appendix.

Let us assume the general equation

$$ay(4A)^{-\gamma} + by(2A)^{-\gamma} + cy(A)^{-\gamma} = 0, \tag{D.13}$$

and determine the coefficients a, b and c such that one solution of this equation in γ corresponds to the cluster coefficient. We start from the general equation

$$y(jA) = y_S \int h(\nu) e^{-jA\nu} d\nu, \tag{D.14}$$

and use the assumption that clustering is weak, that is, that $h(\nu)$ is narrowly concentrated around its mean ν_0. In fact, we assume that $h(\nu)$ can be approximated by a normal distribution with mean ν_0 and standard deviation $\sigma(\nu)$, and that the latter is small. We then find

$$y(jA) \approx y_S e^{-jA\nu_0} e^{j^2A^2\sigma^2(\nu)/2}. \tag{D.15}$$

When the variance $\sigma^2(\nu)$ is zero, the compound distribution is the regular Poisson one, and the only solution of Equation (D.13) should be $\gamma = 0$. This leads immediately to the requirement $a + b + c = 0$. The left hand side of the equation would change sign, however, if its derivative at $\gamma = 0$ were not zero too, which would force the existence of a second solution, similarly to what was observed with Equation (D.7). Consequently, we also require $4a + 2b + c = 0$. The solution to these two equations is $b = -3a$ and $c = 2a$. As global multiplicative constants do not matter, the simplest solution is $a = 1$, $b = -3$ and $c = 2$, as in Equation (D.7).

When we now insert Equation (D.15) in Equation (D.13), with the coefficients as established above, and expand the exponentials, we find

$$3A^2\gamma(\nu_0^2\gamma - \sigma^2(\nu)) = 0, \tag{D.16}$$

which leads to the desired conclusion that the non-trivial solution to Equation (D.13) is

$$\gamma = \sigma^2(\nu)/\nu_0^2, \tag{D.17}$$

that is, the inverse of the cluster coefficient. This result is independent of the details of $h(\nu)$ when the latter is narrowly concentrated around $\nu = \nu_0$.

Appendix E

Cell Fail Probabilities

The likelihood function L equals

$$\prod_i u_i^{O_i}(1-u_i)^{(K-O_i)}, \tag{E.1}$$

and has to be minimized with respect to its parameters u_i, or with respect to the cell fail probabilities, using Equation (5.1). In this appendix, the latter strategy will be followed.

There are two possibilities. Each object has its own cell fail probability t_i, or all cell fail probabilities are equal to some global probability t. The maximizing $\hat{t}_i$ or $\hat{t}$ are found more easily by maximizing the logarithm of L rather than L itself. The resulting equations are

$$(\forall i) \qquad \left(\frac{O_i}{u_i} - \frac{K-O_i}{1-u_i}\right)\frac{\partial u_i}{\partial t_i} = 0, \tag{E.2}$$

and

$$\sum_i \left(\frac{O_i}{u_i} - \frac{K-O_i}{1-u_i}\right)\frac{du_i}{dt} = 0, \tag{E.3}$$

respectively. $\hat{t}_i$ or $\hat{t}$ can be obtained from these equations using equation (5.1), which shows that

$$\frac{du_i}{dt} = s_i \frac{1-u_i}{1-t}, \tag{E.4}$$

and a similar equation for t_i. These first derivatives are always positive, because the object fail probability increases when the corresponding cell fail probability increases. Using these equalities, we easily find that the maximum likelihood equation for different cell fail probabilities is

$$\hat{u}_i = 1-(1-\hat{t}_i)^{s_i} = O_i/K, \tag{E.5}$$

while, for a single cell fail probability, it is

$$\sum_i \frac{O_i s_i}{1-(1-\hat{t})^{s_i}} = K\sum_i s_i. \tag{E.6}$$

The latter equation is an implicit equation in the global cell fail probability $\hat{t}$. It has always a solution, for $1-(1-\hat{t})^{s_i}$ is a monotonically increasing function of the global cell fail probability. The sum on the left hand side is infinite when the latter is zero, and decreases monotonically to $\sum_i O_i s_i$, when the cell fail probability equals 1. It will, therefore, be equal to the sum on the right hand side for some value of $\hat{t}$ between 0 and 1.

For the solutions of these equations to be true estimates of the respective cell fail probabilities, they have to correspond to maxima of the likelihood function L. Maximality can be established by showing that the second derivative of ln(L) is negative. This second derivative will be calculated explicitly, because it is also related to the possible statistical variations in the maximum likelihood estimates $\hat{t}_i$ and $\hat{t}$. The homogeneous case is the most difficult one, and will be treated first.

Differentiating ln(L) twice, we find

$$\frac{d^2}{dt^2}\ln(L) = A+B, \tag{E.7}$$

in which

$$A = -\sum_i \left(\frac{O_i}{u_i^2} + \frac{K - O_i}{(1-u_i)^2}\right)\left(\frac{du_i}{dt}\right)^2 , \tag{E.8}$$

and

$$B = \sum_i \left(\frac{O_i}{u_i} - \frac{K - O_i}{1-u_i}\right)\frac{d^2 u_i}{dt^2} . \tag{E.9}$$

B can be rewritten, using Equation (E.4) and

$$\frac{d^2 u_i}{dt^2} = -s_i(s_i - 1)\frac{1-u_i}{(1-t)^2} = -\frac{s_i - 1}{1-t}\frac{du_i}{dt} , \tag{E.10}$$

as

$$-\sum_i \frac{s_i - 1}{1-t}\left(\frac{O_i}{u_i} - \frac{K - O_i}{1-u_i}\right)\frac{du_i}{dt} . \tag{E.11}$$

The final step in calculating the second derivative of ln(L) is to combine the terms in A and B that are proportional to O_i, and the terms proportional to $K - O_i$. These combinations involve the factors

$$-\frac{1}{u_i}\frac{du_i}{dt} - \frac{s_i - 1}{1-t} = -\frac{1}{u_i}\frac{s_i}{1-t} + \frac{1}{1-t} , \tag{E.12}$$

and

$$-\frac{1}{1-u_i}\frac{du_i}{dt} + \frac{s_i - 1}{1-t} = -\frac{1}{1-t} . \tag{E.13}$$

Using the results above, the second derivative of ln(L) is seen to be equal to

$$-\sum_i \frac{O_i}{u_i^2}\frac{s_i}{1-t}\frac{du_i}{dt} + \frac{1}{1-t}\sum_i\left(\frac{O_i}{u_i} - \frac{K-O_i}{1-u_i}\right)\frac{du_i}{dt}, \tag{E.14}$$

which is negative when $t = \hat{t}$, because the first term on the right hand side is negative, and the second term vanishes because of Equation (E.3).

The heterogeneous case is similar, but more straightforward, for ln(L) is a sum of terms, each one of which depends on the cell fail probability of a single object only. Consequently,

$$\frac{\partial^2}{\partial t_i t_j}\ln(L) = 0 \qquad i \neq j. \tag{E.15}$$

Furthermore, when i = j, all equations for the homogeneous case, starting with the definitions of A and B, remain valid after removing all sums over i, and replacing t by t_i, and all derivatives with respect to t by partial derivatives with respect to t_i. The second partial derivative of ln(L) equals

$$-\frac{O_i}{u_i^2}\frac{s_i}{1-t_i}\frac{\partial u_i}{\partial t_i} + \frac{1}{1-t_i}\left(\frac{O_i}{u_i} - \frac{K-O_i}{1-u_i}\right)\frac{\partial u_i}{\partial t_i}, \tag{E.16}$$

which is negative when $t = \hat{t}$, because the first term on the right is, while the second term vanishes according to Equation (E.2).

At the solutions of the maximum likelihood equations, the second derivatives can also be written as

$$\frac{\partial^2}{\partial t_i^2}\ln(L) = -\frac{K}{\hat{u}_i(1-\hat{u}_i)}\left(\frac{\partial u_i}{\partial t_i}\right)^2 \tag{E.17}$$

in the heterogeneous model, and

$$\frac{d^2}{dt^2}\ln(L) = -\sum_i \frac{O_i}{\hat{u}_i^2(1-\hat{u}_i)}\left(\frac{\partial u_i}{\partial t}\right)^2 \tag{E.18}$$

in the homogeneous one. The derivatives on the right in these equations are given in Equation (E.4), and need to be evaluated at $\hat{t}_i$ and $\hat{t}$, respectively. When the sample size K is large, the variances of the maximum likelihood estimators are roughly equal to minus the inverses of these second derivatives. In the heterogeneous model, the variances actually form a covariance matrix, which is the inverse of the matrix of second derivatives. The latter matrix is diagonal, however, because of Equation (E.15), and inverting it amounts to nothing more than inverting the individual diagonal terms.

Appendix F

Characterization Group

The characterization group contains M devices, and it is known of each device in the characterization group whether it passed or failed the object tests. The number of times object i failed among the devices in the characterization group equals O_i, and it will be assumed that the objects fail independently. R, the probability that a device fails one or more of the object tests, is then given by

$$R = 1 - \prod (1 - u_i) \tag{F.1}$$

Even if all the objects fail independently, this independence is lost within the characterization group, for then the probability that at least one of the objects fails is 1. To handle this dependency, all possible fail patterns need to be considered explicitly.

1 LIKELIHOOD EQUATIONS

Let Ξ_k be a particular set of objects such that object i has failed if $i \in \Xi_k$, and passed if $i \notin \Xi_k$. Each set Ξ_k is labeled by an integer k, running from 0 to 2^I-1. $k = 0$ corresponds to the empty set. To simplify the notation, sums over Ξ_k, $k \geq 1$ will be indicated by sums over Ξ, with the restriction to $k \geq 1$ understood.

The probability p_Ξ of a particular set equals 0 when Ξ is the empty set, and otherwise

$$p_\Xi = \frac{1}{R} \prod_{i \in \Xi} u_i \prod_{j \notin \Xi} (1 - u_j), \tag{F.2}$$

similar to equation (5.18). Note that p_Ξ is a probability with respect to the characterization group. From equation (F.2),

$$\sum_\Xi p_\Xi = 1, \tag{F.3}$$

as it should, because every device in the characterization group has at least one failing object, and because the set of Ξ_k, $k \geq 1$, covers all possible combinations of passing and failing objects.

Equation (F.3) has an interesting consequence. The p_i, introduced in Chapter 5.7, are related to the p_Ξ through

$$p_i = \sum_{\Xi,\, i \in \Xi} p_\Xi. \tag{F.4}$$

Using Equations (F.4) and (F.3), we find

$$\sum_i p_i \geq 1, \tag{F.5}$$

for every p_Ξ in the sum in Equation (F.3) occurs in at least one of the sums in Equations (F.4).

The number of devices for which the set of failing objects is Ξ, is M_Ξ. By definition of the characterization group, M_{Ξ_0} equals 0, and, by definition of M_Ξ, $\sum_\Xi M_\Xi = M$. The resulting likelihood function L equals

$$\prod_\Xi p_\Xi^{M_\Xi} = R^{-M} \prod_i u_i^{O_i} (1 - u_i)^{M - O_i}. \tag{F.6}$$

Note that, if there is only one object, M equals O_1, R equals u_1, and L equals the constant 1. This becomes obvious once we realize that, with only one object, the characterization group is merely a random selection from the failing devices, and no more statistical variability is possible within this group. Only with more than one object can any non-trivial information be obtained from the characterization group.

L has to be maximized with either individual t_i or all t_i equal to a global t. In the former case, maximizing the logarithm of L with respect to the u_i will work just as well. The likelihood equation is

$$\frac{O_i}{u_i} - \frac{M - O_i}{1 - u_i} - \frac{M}{R}\frac{\partial R}{\partial u_i} = 0. \tag{F.7}$$

Using Equation (F.1), we find

$$\frac{\partial R}{\partial u_i} = \frac{1 - R}{1 - u_i}, \tag{F.8}$$

which leads to the maximum likelihood estimator $\hat{u}_i / \hat{R} = O_i/M$, and shows that p_i can be estimated by O_i/M, despite the dependencies between the object fails in the characterization group. Likewise, when a single cell fail probability t is assumed, the $\hat{p}_i$ obey the equation

$$\sum_i \frac{O_i s_i}{\hat{p}_i} = M \sum_i s_i. \tag{F.9}$$

2 HETEROGENEOUS MODEL

In the heterogeneous case, p_i is estimated by O_i/M, but it is not immediately obvious that the desired cell fail probabilities $\hat{t}_i$ can be extracted from $\hat{p}_i$. In the following discussion, the ^ mark over the random variables will be left out, and all random variables will be understood to be maximum likelihood estimators.

Clearly, if the object fail probabilities u_i can be obtained, so can the cell fail probabilities, and, as the former equal Rp_i, and as R is related to the object fail probabilities through Equation (F.1), the latter can be obtained if the equation

$$R = 1 - \prod_i (1 - Rp_i) \tag{F.10}$$

can be solved. With the definition

$$f(R) = 1 - R - \prod_i (1 - Rp_i), \tag{F.11}$$

Equation (F.10) is equivalent to f(R) = 0, and the question is whether f(R) has zeros in [0, 1].

Equation (F.11) always has the solution R = 0, but that is not an acceptable solution, for the existence of failing objects shows that the probability of a device failing the objects tests is not zero. It has the solution R = 1 if, and only if, at least one p_i equals 1. In all other cases, a solution between 0 and 1 should be found.

It is easy to see that f(0) is 0 and f(1) is negative. Also, the slope of f(R) equals

$$-1 + \sum_i p_i \prod_{j \neq i} (1 - Rp_j) \tag{F.12}$$

It equals $-1 + \sum p_i$ at R = 0, and is non-negative there, according to Equation (F.5). Furthermore, the slope is a continuous function of R, and is monotonically decreasing when R goes to 1. In fact, at R equal to 1, the slope is negative, as f(R) decreases between R = 0 and R = 1. The slope, therefore, must have a single zero in [0, 1].

The behavior of f(R) on [0, 1] is now as follows. It starts out at 0 and increases. It then has a maximum, after which it continuously decreases until it reaches f(1), where its value is negative. It, therefore, has a unique zero on [0, 1], in addition to the one at R = 0.

The exception to this qualitative behavior occurs when $\sum p_i = 1$, because then the two zeros coincide at R = 0. On the other hand, this anoma-

lous case occurs only when each device has exactly one failing object. This indicates a strong negative correlation between the object fails, and it is not surprising then that Equation (F.11) fails to produce a meaningful result.

3 HOMOGENEOUS MODEL

Equation (F.9), like Equation (5.6), is a complex implicit equation in t. It always has a solution, for the same reason that Equation (E.6) always has a solution. The proof of the existence of a solution relies on the monotonic increase of p_i as a function of t

In the homogeneous model, p_i is estimated using Equation (5.19), with u_i and R being functions of the estimate $\hat{t}$. $\hat{p_i}$ is never equal to 0, even when $\hat{t} = 0$, because then

$$\hat{p_i} = \frac{s_i}{\sum_j s_j}. \tag{F.13}$$

On the other hand, when $\hat{t} = 1$, $\hat{p_i}$ is equal to 1 too. That $\hat{p_i}$ is an increasing function of $\hat{t}$ is intuitively obvious, and will now be demonstrated.

We use the abbreviation

$$Q = (1-t)^s, \tag{F.14}$$

and find

$$p = \frac{1-Q}{1-Q^\alpha}, \tag{0.1}$$

in which $\alpha = \Sigma/s$. s is the size of the object in question and Σ is the sum over the sizes of all the objects.

As Q is a decreasing function of t, p is an increasing function of t if it is a decreasing function of Q. It equals 1 when Q is 0, and $1/\alpha$ when Q is 1. Consequently, p is monotonically decreasing for $Q \in [0, 1]$, if it has no extremum in that range.

To show the absence of an extremum, we use

$$\frac{dp}{dQ} = \frac{Q^{\alpha-1}(Q(1-\alpha)+\alpha)-1}{(1-Q^{\alpha})^2}. \tag{F.15}$$

This derivative equals $-(\alpha-1)^2/\alpha^2$ when Q is 1. It has a zero when

$$Q^{\alpha-1}(Q(1-\alpha)+\alpha) = 1. \tag{F.16}$$

The left hand side of Equation (F.16) equals 0 when Q is 0, and equals 1 when Q is 1. It has extrema at Q equal to 0 and 1, and, therefore, is equal to 1 only when Q is 1. That, however, does not correspond to an extremum of p, and we conclude that p has no extremum when $Q \in [0, 1]$; or, in other words, that p is an increasing function of t.

In the homogeneous model, the primary estimate is that of $\hat{t}$. This estimate, and, therefore, also the estimate of R, equals 1 only if all objects fail on all devices in the characterization group. It equals 0 in the same situation in which the heterogeneous estimate of R equals 0, namely when $\sum_i O_i = M$.

4 VALIDITY OF THE LIKELIHOOD ESTIMATES

Even if a solution of the maximum likelihood equation is found, it is not acceptable unless it corresponds to a maximum of L (or ln(L).) This issue will now be investigated, but only for the heterogeneous model. The starting point is the logarithm of the likelihood function

$$\sum O_i \ln u_i + \sum (M - O_i)\ln(1-u_i) - M\ln R. \tag{F.17}$$

The subsequent notation will be simplified by defining

$$R_i \equiv \frac{\partial R}{\partial u_i} = \frac{1-R}{1-u_i}, \tag{F.18}$$

and

$$R_{ij} \equiv \frac{\partial}{\partial u_i}\frac{\partial R}{\partial u_j} = \frac{-1}{1-R} R_i R_j (1-\delta_{ij}), \tag{F.19}$$

where δ_{ij} equals 1 when i = j, and 0 otherwise. u_i and R_i obey an important inequality. R_i equals $\prod_{j \neq i}(1-u_j)$, and is the probability that all objects other than i do not fail. Therefore, $u_i R_i$ is the probability that object i fails, and no other objects. Consequently, the sum over these terms is the probability that exactly one object fails. R on the other hand is the probability that at least one object fails, that is, exactly one, or more than one. This implies

$$\sum u_i R_i \leq R \tag{F.20}$$

The maximality of the solution of Equation (F.7) can be established by evaluating the second derivative of ln(L), which equals

$$\left(\left(-\frac{O_i}{u_i^2} - \frac{M-O_i}{(1-u_i)^2}\right)\delta_{ij} + \frac{M}{R^2}R_i R_j - \frac{M}{R}R_{ij}\right)\frac{du_i}{dt_i}\frac{du_j}{dt_j} \quad . \tag{F.21}$$

Unlike the situation in Appendix E, this matrix is not diagonal, since the fails in the characterization group are correlated. At the solution of (F.7), the coefficient of the product of the derivatives of u_i and u_j equals

$$-\frac{M}{R}\left(\left(\frac{1}{u_i} + \frac{1}{1-u_i}\right)\delta_{ij} - \frac{R_i R_j}{R(1-R)}\right) \quad . \tag{F.22}$$

For the solution of the maximum likelihood equations to be valid, its matrix of second derivatives has to be negative definite. This translates into the requirement that the matrix of the coefficients of $-\frac{M}{R}$ in equation (F.22) needs to be positive definite. In other words,

$$\sum_{ij} \alpha_i \alpha_j \left(\left(\frac{1}{u_i} + \frac{1}{1-u_i}\right)\delta_{ij} - \frac{R_i R_j}{R(1-R)}\right) \tag{F.23}$$

has to be positive for all values of the coefficients α_i.

(F.23) is now easily shown to be positive, as it can be written as

$$\sum_i \alpha_i^2 \frac{1}{u_i(1-u_i)} - \frac{1}{R(1-R)}\left(\sum_i \alpha_i \frac{\sqrt{R_i}}{\sqrt{u_i}}\sqrt{u_i R_i}\right)^2. \tag{F.24}$$

The reason for writing the second term in this curious fashion is that we can now use Cauchy's inequality to show that (F.23) is larger than

$$\frac{1}{1-R}\sum_i \alpha_i^2 \frac{R_i}{u_i}\left(1 - \frac{\sum_j u_j R_j}{R}\right), \tag{F.25}$$

which is positive, as all factors preceding the one in parentheses are positive, and as that final factor is positive because of Equation (F.20).

Appendix G

Component Fail Probabilities

1 MAXIMUM LIKELIHOOD ESTIMATION

With

$$L = \prod_i u_i^{O_i} (1 - u_i)^{M_i - O_i}. \tag{G.1}$$

and

$$u_i = 1 - \prod_j q_j^{n_{ij}}, \tag{G.2}$$

the maximum likelihood equations for the probabilities $\widehat{q_i}$ that the components are defect free are

$$\frac{\partial}{\partial q_j} \ln L = \sum_i \left(\frac{O_i}{u_i} - \frac{M_i - O_i}{1 - u_i} \right) \frac{\partial u_i}{\partial q_j} = 0, \tag{G.3}$$

with

$$\frac{\partial u_i}{\partial q_j} = \frac{n_{ij}}{q_j} u_i \ . \tag{G.4}$$

With the use of Equation (G.4), Equations (G.3) can be rewritten as

$$\sum_i \left(M_i - \frac{M_i - O_i}{1 - u_i} \right) n_{ij} = 0 , \tag{G.5}$$

assuming that $q_j \neq 0$. This set of equations has to be solved numerically.

Before continuing with these equations, we need to determine when a solution corresponds to a maximum. Even though the likelihood function is defined for all values of the component probabilities, we need to restrict solutions to the hypercube $\mathcal{H}$ defined by

$$(\forall j)(0 \leq q_j \leq 1) , \tag{G.6}$$

for the solutions to be physically meaningful. It is shown in that there is a single solution of Equations (G.5) in a region that includes $\mathcal{H}$, and that this solution is a maximum. If the solution is outside $\mathcal{H}$, the likelihood function has to be maximized on its boundary, which is a much more complicated problem than finding solutions to Equations (G.5), and will not be addressed here.

The remaining question is how to find the unique maximum of L. As L is a non-linear function of many variables, and Equations (G.5) likewise are non-linear, no easy solution is available. The maximum, therefore, has to be found with a laborious search algorithm (see [44] for example). It is useful, however, to first make a rough estimate of where the solution will likely end up.

To be meaningful, the maximum has to be located within the hypercube $\mathcal{H}$ (Equation (G.6).) In addition, we expect the objects to have reasonable yields, which implies that the component yields should all be close to 1. This is unfortunate for numerical reasons, for an estimate close to 1 may end up as 1 because of machine round off. In other words, the computer used to do the numerical calculations may not be able to express the difference between q_j and 1 if 1 - q_j is small enough

This numerical problem can be solved by changing variables. Instead of q_j, we use $r_j = \ln(q_j)$. When q_j is close to 1, r_j is small, and when it goes to 0, r_j goes to $-\infty$. The hypercube $\mathcal{H}$ is mapped onto the hyperquadrant $\mathcal{H}'$, defined as

$$(\forall j)(r_j \leq 0). \tag{G.7}$$

u_i equals $e^{\sum_j n_{ij} r_j}$, and is positive, except when any r_j goes to $-\infty$, in which case it goes to 0. The other interesting value for u_i is 1, which it attains, in $\mathcal{H}$, when $\sum_j n_{ij} r_j = 0$, that is when $r_j = 0$ for each j for which $n_{ij} \neq 0$. L is positive in $\mathcal{H}$, except when some u_i equals 0, or some u_i equals 1.

All algorithms that attempt to find maxima need a good starting point, and one convenient one is the maximum of L along $r_j = r$ for all j. The corresponding maximum likelihood equation is

$$\sum_i \frac{O_i - M_i u_i}{1 - u_i} s_i = 0, \tag{G.8}$$

with $u_i = e^{s_i r}$. This equation is still non-linear, but now of only one variable. Moreover, we only need a solution in the range $r < 0$, and standard techniques are available for such a simplified problem. There obviously is a solution, for the sum on the left hand side of Equation (G.8) is positive when r goes to $-\infty$, and is negative when r goes to 0.

2 LOCATION OF THE MAXIMA OF THE LIKELIHOOD FUNCTION

The question of whether the solutions to the maximum likelihood equations correspond to a maximum can be settled by determining whether the matrix of second derivatives of ln(L) is negative definite at those solutions. This matrix is described by the general element

$$\frac{\partial}{\partial q_r}\frac{\partial}{\partial q_t}\ln L = -\sum_i E_i \frac{n_{ri} n_{ti}}{q_r q_t} - \frac{1}{q_r^2} F \delta_{rt}$$

$$E_i = \frac{u_i}{(1-u_i)^2}(M_i - O_i) \tag{G.9}$$

$$F = \sum_i \left(M_j - \frac{M_i - O_i}{1-u_j} \right) n_{ij}$$

At the solution of the maximum likelihood equations, $F = 0$, according to Equation (G.5).

Consequently, for an arbitrary set of numbers $\{\alpha_r\}$,

$$\sum_{r,t} \alpha_r \alpha_t \frac{\partial}{\partial q_r}\frac{\partial}{\partial q_t}\ln L = -\sum_i E_i \left(\sum_r \alpha_r \frac{n_{rj}}{q_r} \right)^2, \tag{G.10}$$

which is manifestly negative for all choices of $\{\alpha_r\}$. The matrix of second derivatives is, therefore, negative definite, and any solution of the maximum likelihood equations corresponds to a maximum.

The remaining question is whether this solution exists within the hypercube $\mathcal{H}$ defined in Equation (G.6), and whether this solution is unique. I will show the existence and uniqueness of the solution within a region $\mathcal{I}$ that is somewhat larger $\mathcal{H}$.

L is a polynomial in the component probabilities q_i, and is therefore defined for all finite values of those probabilities. It vanishes whenever any of these probabilities equals 0, and also whenever any of the object probabilities u_i equals 1. The latter are also polynomials in the q_j.

Let $\mathcal{S}$ be the region in which all q_j are positive. In $\mathcal{S}$, u_i is a positive and non-decreasing function of the q_j. We can easily find the region $\mathcal{R}_i \subset \mathcal{S}$ in which u_i is less than 1. As u_i is trivially less than 1 in $\mathcal{H}$, $\mathcal{H} \subset \mathcal{R}_i$.

The extended region $\mathcal{I}$ is now defined as the intersection of all regions $\mathcal{R}_i$. Each $\mathcal{R}_i$ contains at least $\mathcal{H}$, and $\mathcal{I}$, therefore, contains the hypercube as well. In addition, L is positive inside $\mathcal{I}$, and goes to 0 on its boundary. It defines an extended region in which the likelihood function has a single maximum, and

no other extrema (because all extrema are maxima.) The position of the maximum need not be in $\mathcal{H}$, though.

Appendix H

Yield and Coverage

Assume that there are F defects. Let the fraction of tested single defects be c, and let Γ be the set of defects not exposed by some arbitrary but fixed test sequence. Γ depends on the test, but its size $\gamma = (1\text{-}c)F$ is independent of the test.

let X_i be a random variable that equals 1 when defect i is tested by the test sequence with defect coverage c, and 0 otherwise. The X_i are in general not mutually independent. Because of its definition,

$$\begin{aligned} X_i X_i &= X_i \\ (1 - X_i)(1 - X_i) &= 1 - X_i \end{aligned} \quad . \tag{H.1}$$

Each chip contains a set S of the defects. S can be empty, indicating a good chip. The probability of finding a chip with this set of S defects will be indicated by p_S. S is assumed to be tested when any of its members is. The corresponding random variable X_S equals $1 - \prod_{j \in S} (1 - X_j)$, which equals 1 when any of the members of S is tested and 0 otherwise.

It is more convenient to work with $1 - X_S$ than with X_S. $1 - X_S$ depends on Γ, for it equals 1 when $S \subseteq \Gamma$, and is 0 otherwise. To make this dependency explicit, it will be indicated by $\eta_{S\Gamma}$. Because the test sequences considered at this point all have coverage c, $\eta_{S\Gamma}$ obeys the useful sum rule

$$\sum_{|S| = n} \eta_{S\Gamma} = \sum_{(|S| = n),\, S \subset \Gamma} 1 = \binom{\gamma}{n}, \tag{H.2}$$

independent of the actual test sequence. The average over all test procedures of $\eta_{S\Gamma}$ will be indicated by

$$b_S = \langle \eta_{S\Gamma} \rangle , \tag{H.3}$$

and, because of Equation (H.2),

$$\sum_{|S| = n} b_S = \binom{\gamma}{n}. \tag{H.4}$$

The average

$$a_n = \frac{1}{\binom{F}{n}} \sum_{|S| = n} b_S , \tag{H.5}$$

will also be needed. Equation (H.4) leads to the result that

$$a_n = \frac{\binom{\gamma}{n}}{\binom{F}{n}}. \tag{H.6}$$

When $n > \gamma$, a_n equals 0. a_0 equals 1, and a_1 equals 1 - c. Otherwise, when $F \gg n$ and $n > 1$,

$$a_n \approx (1-c)^n - \frac{c}{2F}(1-c)^{n-1} n(n-1) . \tag{H.7}$$

1 AVERAGE YIELD

We are now ready to consider the yield Y. Clearly, Y corresponds to the event that either the chip is not defective, or that any existing defects are not detected by the test sequence; in other words, when the defect S is empty, or when all the members of the defect S are in Γ.

It is more convenient to consider y, the expected value of Y given the test sequence. Even though y is an expected value, it still depends on the test sequence, or, equivalently, on Γ. As we will study the dependence of y on Γ, and as Γ depends on the randomly selected test sequence, y really should be treated as a random variable. In this section, it will continue to be indicated by a lower case y, but a subscript Γ will be added to make the dependence of y on Γ clear.

The yield is given by

$$y_\Gamma = \sum_n \sum_{|S| = n} p_S \eta_{S\Gamma} \ , \tag{H.8}$$

and the average over different test generations by

$$\langle y_\Gamma \rangle = \sum_n \sum_{S = n} p_S b_S \ . \tag{H.9}$$

Let us now use the all-important assumption that p_S and b_S are uncorrelated, at least for sets S of the same size. Or, in other words, that

$$\sum_{|S| = n} \Delta p_S (b_S - a_n) = 0 \ , \tag{H.10}$$

where Equation (8.1) was used as well as the abbreviation

$$\Delta p_S = p_S - \frac{1}{\binom{F}{n}} P_n \ . \tag{H.11}$$

Equation (H.10) leads to the equality

$$\sum_{|S| = n} \Delta p_S b_S = 0 \ , \tag{H.12}$$

which will turn out to be useful later. It can also be rewritten in the more useful form

$$\sum_{|S| = n} p_S b_S = P_n a_n \ . \tag{H.13}$$

In other words, combining Equations (H.13) and (H.9),

$$\langle y_\Gamma \rangle = \sum_n P_n a_n \approx \sum_n P_n (1-c)^n . \tag{H.14}$$

To get an impression of the difference between a_n and $(1\text{-}c)^n$ on the average yield, consider the independent defect case with all the occurrence probabilities equal to π. In that case, μ, the mean number of defects per device, equals πF, and

$$P_n = P_0 \binom{F}{n} \left(\frac{\pi}{1-\pi}\right)^n . \tag{H.15}$$

The yield using Equation (H.6) is then equal to $(1-\pi)^{F-\gamma} = (1-\pi)^{cF}$, while it equals $(1-c\pi)^F$ when the approximate value of a_n is used. Because the approximate value of a_n is valid only when F is very large, and because P_n decreases rapidly when n increases, these two results for the yield are in fact both equal to $e^{-c\mu}$, which is the standard Poisson result.

2 VARIANCE OF THE YIELD

The actual value of y_Γ depends on the test sequence, as indicated by the subscript Γ, and will differ from $\langle y_\Gamma \rangle$ by some amount. The typical size of this difference will now be estimated. The conclusion will be that it is proportional to $F^{-1/2}$, and therefore negligibly small for large designs.

Consider first the difference

$$y_\Gamma - \langle y_\Gamma \rangle = \sum_n \sum_{|S| = n} p_S (\eta_{S\Gamma} - b_S) . \tag{H.16}$$

If p_S is a function only of n, but otherwise independent of S, Equations (H.2) and (H.4) show that $y_\Gamma - \langle y_\Gamma \rangle$ vanishes, or, in other words, that y_Γ is a constant, depending only on c and on the constants p_S. This suggests that $y_\Gamma - \langle y_\Gamma \rangle$ depends on the deviation of p_S from its average value. And indeed, Equation (H.16) can be rewritten with the help of Equations (H.2) and (H.4) as

$$y_\Gamma - \langle y_\Gamma \rangle = \sum_n \sum_{|S| = n} \Delta p_S (\eta_{S\Gamma} - b_S) \quad . \tag{H.17}$$

Unlike the left hand side of Equation (H.10), the right hand side of Equation (H.17) does not vanish, for it still depends on Γ, and Γ could be chosen such that those sets S are favored that have, for example, especially large p_S. On the other hand, Equation (H.12) can be used to simplify Equation (H.17) even further to

$$y_\Gamma - \langle y_\Gamma \rangle = \sum_n \sum_{|S| = n} \Delta p_S \eta_{S\Gamma} \quad . \tag{H.18}$$

Note that the sum over n runs from $n = 1$ to $n = \gamma$, because Δp_S equals 0 when $n = 0$, and $\eta_{S\Gamma}$ equals 0 when n is larger than γ.

To estimate the size of $y_\Gamma - \langle y_\Gamma \rangle$, we use the inequality

$$|y_\Gamma - \langle y_\Gamma \rangle| \leq \sum_n \left| \sum_{|S| = n} \Delta p_S \eta_{S\Gamma} \right| , \tag{H.19}$$

and concentrate on the absolute value of the sum over |S| equal to n. In fact, we will estimate the expected value of its square

$$\sum_{|S| = n,\, |T| = n} \Delta p_S \Delta p_T \langle \eta_{S\Gamma} \eta_{T\Gamma} \rangle \quad . \tag{H.20}$$

The first step is an estimate of the typical size of $|\Delta p_S|$, which we take to be the standard deviation of p_S. As there are $\binom{F}{n}$ different sets S of size n, the variance of p_S equals

$$\frac{1}{\binom{F}{n}} \sum_{|S| = n} {p_S}^2 - \left(\frac{P_n}{\binom{F}{n}} \right)^2 \quad . \tag{H.21}$$

The p_S can have all kinds of values, but we will assume that its distribution is not dominated by a very small number of very likely defects. I.e.:

$$p_S \leq \lambda_n \frac{1}{\binom{F}{n}} P_n , \tag{H.22}$$

with λ_n a constant that is much smaller than $\binom{F}{n}$ and independent of F. The standard deviation of p_S is then bounded by

$$\frac{1}{\binom{F}{n}} P_n \sqrt{\lambda_n^2 - 1} . \tag{H.23}$$

The next step is the actual estimation of the size of the double sum in Equation (H.20). This double sum seems to be a sum of $\binom{F}{n}^2$ terms, each of order $\lambda_n^2 P_n^2 / \binom{F}{n}^2$ as $\langle \eta_{S\Gamma} \eta_{T\Gamma} \rangle$ is at most of order 1. $|y_\Gamma - \langle y_\Gamma \rangle|$, therefore, seems to be of order $\sum_n \lambda_n P_n$. This, however, is not correct, for the Δp_S can be both positive and negative, and a fair amount of cancellation is expected to occur. In fact, I will show below that these cancellations lead to an effective reduction in the number of terms, and that the actual number is of order $\binom{F}{n}^2 \frac{n^2}{F}$. The final result is then that $|y_\Gamma - \langle y_\Gamma \rangle|$ is of order $\frac{1}{\sqrt{F}} \sum_n \lambda_n n P_n$, i.e. that y_Γ differs from its average by an amount proportional to $F^{-1/2}$.

What remains to be demonstrated is the reduction in the number of terms in Equation (H.20). For any given S, the sum over T can be divided into several parts, depending on the degree of overlap of S and T. Symbolically:

$$\sum_{|T| = n} \ldots = \sum_{n \leq m \leq 2n} \sum_{(|T| = n), (|S \cup T| = m)} \ldots , \tag{H.24}$$

where m indicates the size of the union of S and T.

I will now show that almost all terms in the sum on the left have the maximal value of m. The number of terms in the second sum on the right equals $\binom{n}{2n-m}\binom{F-n}{m-n}$, for 2n - m is the number of elements in $T \cap S$, and

$\binom{n}{2n-m}$ is the number of ways of selecting subsets of that size from S. Likewise, m - n is the number of elements of T not in S, and $\binom{F-n}{m-n}$ is the number of ways one can select those elements from F. The number of terms with $m = 2n$, therefore, equals $\binom{F-n}{n}$, while the total number of terms in the sum on the left of Equation (H.24) equals $\binom{F}{n}$. As the ratio between the two is approximately $1 - n^2/F$, we can conclude that the fraction of terms in the sum with M not equal to 2n equals n^2/F, which is small for large F and n not too large.

To be able to use this result, we will have to make one more assumption. The terms with m equal to 2n contribute an amount

$$\sum_{(|T| = n),\,(|S \cup T| = 2n)} \Delta p_T \langle \eta_{S\Gamma} \eta_{T\Gamma} \rangle \tag{H.25}$$

to (H.24). This sum can in general not be simplified, because $\langle \eta_{S\Gamma} \eta_{T\Gamma} \rangle$ depends on T. Let us assume then that $\langle \eta_{S\Gamma} \eta_{T\Gamma} \rangle$ and Δp_T are uncorrelated for T that are completely outside S:

$$\sum_{|S \cup T| = 2n} \Delta p_T (\langle \eta_{S\Gamma} \eta_{T\Gamma} \rangle - a_{nS}) = 0, \tag{H.26}$$

in which the sum over T is assumed, and

$$a_{nS} = \frac{1}{\binom{F-n}{n}} \sum_{|S \cup T| = 2n} \langle \eta_{S\Gamma} \eta_{T\Gamma} \rangle, \tag{H.27}$$

depends only on n and S. Equation (H.26) allows us to rewrite Equation (H.25) as

$$\sum \Delta p_T \langle \eta_{S\Gamma} \eta_{T\Gamma} \rangle = a_{nS} \sum_{|S \cup T| = 2n} \Delta p_T, \tag{H.28}$$

which, using the definition of Δp_T, becomes

$$\sum \Delta p_T \langle \eta_{S\Gamma} \eta_{T\Gamma} \rangle = -a_{nS} \sum_{|S \cup T| < 2n} \Delta p_T . \tag{H.29}$$

The result of all these manipulations is that the right hand side of Equation (H.20) can be rewritten as

$$\sum_{|S| = n} \Delta p_S \sum_{m = n}^{2n-1} \sum_{\substack{|T| = n \\ |S \cup T| = m}} \Delta p_T (\langle \eta_{S\Gamma} \eta_{T\Gamma} \rangle - a_{nS}) , \tag{H.30}$$

in which the term with $m = 2n$ is now absent. Consequently, the coefficient of Δp_S in this complex sum has about $\binom{F}{n} n^2 F^{-1}$ terms, as was to be proven.

Appendix I

Estimating First Fail Probabilities from the Fallout

The coverage at the completion of the i^{th} test is c_i, and the corresponding yield is y_i. The coverage and the yield at the completion of scan based testing are indicated by c_f and y_f, respectively. For completeness, I also define $c_0 = 0$ and $y_0 = 1$. Note that in this appendix, and in this appendix only, y_0 is the perceived yield at the beginning of scan based testing, and not the true yield.

The multinomial parameters are related to the yield by $d_i = y_{i-1} - y_i$, and, vice versa,

$$y_f = 1 - \sum_{i=1}^{f} d_i . \tag{I.1}$$

The negative binomial parameters μ and α are now estimated from the experimental fallout data by maximizing the multinomial distribution, taken as functions of μ and α. Because of the form of the negative binomial yield equation, it is convenient to take as its parameters the new variables $\nu = \mu/\alpha$ and $\beta = \alpha$. With these new parameters,

$$y_i = (1 + c_i \nu)^{-\beta} . \tag{I.2}$$

The likelihood function L equals

$$\prod_i d_i^{N_i} y_f^{N_{pass}}. \tag{I.3}$$

This function can be maximized using standard optimization routines. L does have a maximum, as I will now show.

The range of values for the parameters ν and β is $\nu > 0$ and $(\beta > 0)$. It is easy to see that L vanishes on the boundary of this range, for $\nu = 0$ or $\beta = 0$ implies that $d_i = 0$ is zero for every i, while $\nu = \infty$ or $\beta = \infty$ implies that y_f is zero. L, therefore, is a continuous, non-negative function of ν and β, and equals 0 on the boundaries of the range of ν and β. Consequently, it has a maximum in the interior of the range.

Appendix J

Identity of M and S.

If the splats are completely separated, each pin in each splat explains all the patterns associated with that splat, and no other ones. M is the set of pins in the multiplets, S is the set of pins in the splats, and M equals S. This equality will be proven in this appendix.

First, each pin in S_A has to be in M. To see that this is true, consider the hypothetical case that some pin in S_A, say p, is not in M. Some pin in S_A has to be in M, because otherwise the patterns associated with A are not explained by pins in M. Let this other pin be q. q can safely be replaced by p in each multiplet in which it occurs, and still have the multiplet explain all SLAT patterns, since p and q explain the same patterns. This replacement would create multiplets of the same size as the original ones, which would have been found already if the process outlined in Chapter 10.3.1 is correct. The replacement, therefore, does not create new multiplets, and p is in M.

Secondly, each pin in M is in S. For consider the case that some pin p in M explains some failing patterns in subset A, but not all. p, therefore, is not in S, because it does not explain the patterns in subset A, by assumption, and cannot explain the patterns in any other subset as the splats are completely separated. The other patterns in A still have to be explained by pins in the multiplets in which p occurs. Because the splats are completely separated, the patterns not explained by p can only be explained by some pin q in S_A. q, however, explains all the patterns in A, including the one explained by p. Therefore, p is not necessary to explain all the SLAT patterns, and cannot occur in M, since the multiplets are minimal.

Notice that the last argument also shows that the pins in the same multiplet all have to come from different splats, and, therefore, that the number of splats

is not smaller than the number of pins in the multiplets. On the other hand, each multiplet must have at least one pin from each splat, for the patterns in some subset A would not be explained by a multiplet if none of the pins in S_A were in that multiplet. Consequently, there are as many pins in the multiplets as there are splats.

References

1 E. J. Aas and V. T. Minh, "Defect Level calculation: The importance of accurate models for Defect Distribution and Multiple Fault Coverage in low yield situations," *Proceedings ISCAS '89*, pp. 939-944, 1989.

2 M. Abramovici, M. A. Breuer and A. D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, 1990.

3 Y. Arzoumanian and J. Waicukauski, "Fault Diagnosis in an LSSD Environment", *Proceedings International Test Conference*, 1981, pp. 86-88.

4 Carl Barnhart, Vanessa Brunkhorst, Frank Distler, Owen Farnsworth, Brion Keller, and Bernd Koenemann, "OPMISR: the foundation for compressed ATPG vectors", *Proceedings International Test Conference,* pp. 748 - 757, 2001.

5 Tom Bartenstein, "Fault distinguishing pattern generation", *Proceedings International Test Conference,* pp. 820 - 828, 2000.

6 Thomas Bartenstein, Douglas Heaberlin, Leendert Huisman and David Sliwinski, "Diagnosing Combinational Logic Designs Using the Single Locations At-a-Time (SLAT) Paradigm", *Proceedings International Test Conference*, 2001, pp. 287-296.

7 T. Bartenstein and J. Bhawnami, "SLATPlus: Work in Progress, "*2nd International IEEE Workshop on Yield Optimization and Test*", Nov. 1-2, 2001.

8 Greg Bazan, Francis Gravel, Leendert Huisman, Anne Pardee, Leah Pastel and Ken Rowe, "Using Embedded Objects for Yield Monitoring", *Proceedings of the IEEE/SEMI Advanced Semiconductor Manufacturing Conference and Workshop*, pp. 124-128, 2004.

9 Vamsi Boppana and Masahiro Fujita, "Modeling the Unknown! Towards Model-Independent Fault and Error Diagnosis, *Proceedings International Test Conference*, pp. 1094-1101, 1998.

10 George Casella and Roger L. Berger, *Statistical Inference*, Duxbury, 2002.

11 Brian Chess and Tracy Larrabee, "Creating Small Fault Dictionaries", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 346-356, 1999.

12 A. D. Cliff and J. K. Ord, *Spatial Processes Models & Applications*, Pion Limited, 1981.

13 Randall S. Collica, "The Effect of the Number of Defect Mechanisms on Fault Clustering and its Detection Using Yield Model Parameters", *IEEE Transactions on Semiconductor Manufacturing*, vol 5. pp. 189-195, 1992.

14 Randall S. Collica, Jose G. Ramirez and Winsom Taam, "Process Monitoring in Integrated Circuit Fabrication using both Yield and Spatial Statistics", *Quality and Reliability Engineering International*, pp.195-202, 1996.

15 Harald Cramer, *Mathematical Methods of Statistics*, section 15.6, Princeton University Press, 1974.

16 Alfred L. Crouch, "Design-for-Test for Digital IC's and Embedded Core Systems", Prentice Hall, New Jersey, 1999.

17 J. A. Cunningham, "The Use and Evaluation of Yield Models in Integrated Circuit Manufacturing," *IEEE Transactions on Semiconductor Manufacturing*, vol. 3, pp. 60-71, May 1990.

18 D. V. Das, S. C. Seth, P. T. Wagner, J. C. Anderson and V. D. Agrawal, "An Experimental Study on Reject Ratio Prediction for VLSI Circuits: Kokomo Revisited," *Proceedings IEEE International Test Conference*, pp. 712-720, September 1990.

19 William R. Dillon and Matthew Goldstein, "*Multivariate Analysis Methods and Applications*", John Wiley & Sons, 1984.

20 A. W. F. Edwards, *Likelihood*, The Johns Hopkins University Press, 1992.

21 Edward Eichelberger, Eric Lindbloom, John A. Waicukauski and Thomas W. Williams, "*Structured Logic Testing*," Prentice Hall, New Jersey, 1991.

22 W. Feller, *An Introduction to Probability Theory and Its Applications*, John Wiley & Sons, 1967.

23 Albert V. Ferris-Prabhu, "On the Assumptios Contained in Semiconductor Yield Models", *IEEE Transactions on Computer-Aided Design*, vol 11. pp. 966-975.

24 Virginia Foard Flack, "Introducing Dependency into IC Yield Modesl", *Solid State Electronics*, vol 28, pp. 555-559, 1985.

25 Sophie Gandemer, Bernard C. Tremintin and Jean-Jacques Charlot, "Critical Area and Critical Levels Calculation in I. C. Yield Modeling", *IEEE Transaction on Electron Devices*, vol 35., pp. 158-166, 1988.

26 Mark H. Hansen, Vijayan N. Nair and David J. Friedman, "Monitoring Wafer Map Data From Integrated Circuit Fabrication Processes for Spatially Clusterd Defects", *Technometrics*, vol. 39, pp. 241-253, 1997.

27 S. M. Hu, "Some Considerations in the Formulation of IC Yield Statistics", *Solid-State Electronics*, vol 22., pp.205-211, 1979.

28 Leendert M. Huisman, " Fault Coverage and Yield Predictions: Do we need more than 100 % Coverage ?", *Proceedings European Test Conference*, pp. 180-187, 1993.

29 Leendert M. Huisman, "Yield Fluctuations and Defect Models", *Journal of Electronic Testing: Theory and Applications*, pp. 241-254, 1995.

30 Leendert M. Huisman, "Diagnosing Arbitrary Defects in Logic Designs Using Single Location at a Time (SLAT)", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp 91-101, 2004.

31 Leendert M. Huisman, Maroun Kassab and Leah Pastel, "Data Mining Integrated Circuit Fails with Fail Commonalities", *Proceedings International Test Conference*, pp. 661-668, 2004.

32 Sri Jandhyala, Hari Balachandran, Manidip Sengupta and Anura P. Jayasumana, "Clustering Based Evaluation of IDDQ Measurements: Applications in Testing and Classification of ICs", *VLSI Test Symposium, pp. 444-449*, 2000.

33 Thomas P. Karnowski, Kenneth W. Tobin, Shaun S. Gleason and Fred Lakhani, "The Application of Spatial Signature Analysis to Electrical Test Data: Validation Study", *Proceedings of SPIE*, vol 3677, pp. 530-541, 1999.

34 Israel Koren and Dhiraj Pradhan, "Yield and Performance Enhancement Through Redundancy in VLSI and WSI Multiprocessor Systems", *Proceedings of the IEEE*, vol 74, pp. 699-711, 1986.

35 David B. Lavo, Tracy Larrabee and Brian Chess, "Beyond the Byzantine Generals: Unexpected Behavior and Bridging Fault Diagnosis," *Proceedings International Test Conference*, 1996, pp. 611-619.

36 David B. Lavo, Brian Chess, Tracy Larrabee and F. Joel Ferguson, "Diagnosing Realistic Bridging Faults with Single Stuck-At Information," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 3, pp. 255-267, March 1998.

37 David B. Lavo, Ishmed Hartanto and Tracy Larrabee, "Multiplets, Models, and the search for Meaning: Improving Per-Test Fault Diagnosis", *Proceedings International Test Conference*, 2002, pp. 250-259.

38 B. W. Lindgren, *Statistical Theory*, MacMillan Publishing Co., Inc., 1976.

39 Fred J. Meyer and Dhiraj K. Pradhan, "Modeling Defect Spatial Distributions", *IEEE Transactions on Computers*, vol. 38, pp. 538-546, 1989.

40 Steven D. Millman, Edward J.McCluskey and John M. Acken, "Diagnosing CMOS bridging faults with stuck-at fault dictionaries," *International Test Conference*, 1990, pp. 860-870.

41 Christopher Z. Mooney and Robert D. Duval, *Bootstrapping A Nonparametric Approach to Statistical Inference,* Sage Publications, 1993.

42 B. T. Murphy, "Cost-size Optima of monolithic integrated circuits", *Proceedings IEEE*, vol 52, pp. 1537-1545, 1964.

43 Jon Patrick, private conversation.

44 William H. Press, Brian P. Flannery, Saul A. Teukolsky and William T. Vetterling, "*Numerical Recipes in C*", Cambridge University Press, 1988.

45 J. E. Price, "A New Look at Yield of Integrated Circuits," *Proceedings of the IEEE*, vol. 58, pp. 1290-1291, August 1970.

46 Janusz Rajski and Jerzy Tyszer, "Test Data Compression and Compaction for Embedded Test of Nanometer Technology Designs", Proceedings International Conference om Computer Design, 2003.

47 Jose G. Ramirez and Brenda Cantell, "An Analysis of a Semiconductor Eperiment Using Yield and Spatial Information", *Quality and Reliability Engineering International*, pp.35-46, 1997.

48 J. Richman and K. R. Bowden, "The Modern Fault Dictionary", *Proceedings International Test Conference,* pp. 696-702, 1985.

49 Jayashree Saxena, Kenneth M. Butler, Hari Balachandran, David B. Lavo, Brian Chess, Tracy Larrabee and F. Joel Ferguson, "On Applying Non-Classical Defect Models to Automated Diagnosis," *Proceedings International Test Conference*, 1998, pp. 748-757.

50 S. C. Seth and V. D. Agrawal, "Characterizing the LSI Yield Equation from Wafer Test Data," *IEEE Transactions on Computer-Aided Design*, vol. CAD-3, pp. 123-126, April 1984.

51 Debendra Das Sharma, Fred J. Meyer and Dhiraj K. Pradhan, "Yield Optimization of Modular and Redundant Multimegabit Rams: A Study of Effectiveness of Coding Versus Static Redundancy Using the Center-Satellite Model", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol 1, pp. 546-558, 1993.

52 J.J.T. Sousa and J.P. Teixeira, "Defect Level Estimation for Digital ICs", *Proceedings International Wprkshop on Defect and Fault Tolerance in VLSI Systems*, pp. 32-41, 1992.

53 C. H. Stapper, Jr., "On a Composite Model to the IC Yield Problem," *IEEE Journal of Solid-State Circuits*, pp. 537-539, December 1975.

54 Charles H. Stapper, Frederick M. Armstrong and Kiyotaki Saji, "Integrated Circuit Yield Statistics", *Proceedings of the IEEE*, vol 71, pp. 453-470, 1983.

55 C. H. Stapper, "On Yield, fault distributions, and clustering of particles", *IBM Journal of Research and Development*, pp. 326-339 1986.

56 Winsom Taam and NMichael Hamada, "Detecting Spatial Effects From Factorial Experiments: An Application From Integrated Circuit Manufacturing", *Technometrics*, pp. 149-160, 1993.

57 Aakash Tyagi and Magdy A. Bayoumi, "Defect Clustering Viewed Through Generalized Poisson Distribution", *IEEE Transactions on Semiconductor Manufacturing*, vol 5. pp. 196-206, 1992.

58 Srikanth Venkataraman, Scott B. Drummonds, " A Technique for Logic Fault Diagnosis of Interconnect Open Defects," *Proceedings VLSI Test Symposium*, April 2000, pp. 313-318.

59 Srikanth Venkataraman, Scott B. Drummonds, "POIROT: A Logic Fault Diagnosis Tool and Its Applications," *International Test Conference*, 2000, pp. 253-262.

60 J.A. Waicukauski and E. Lindbloom, "Failure diagnosis of structured VLSI", in IEEE Design & Test of Computers, Volume 6, pp. 49-60, Aug. 1989.

61 Colin A. Warwick and Abbas Ourmazd, "Trends and Limits in Monolithic Integration by Increasing the Die Area", *IEEE Transactions on Semiconductor Manufacturing*, vol. 6, pp.284-289, 1993.

62 T. W. Williams and N. C. Brown, "Defect Level as a Function of Fault Coverage," *IEEE Transactions on Computers*, vol. C-30, pp. 987-988, December 1981.

63 Peter Wohl, John A. Waicukauski, Sanjay Patel and Minesh B. Amin, "X-Tolerant Compression and Application of Scan-ATPG Patterns in a BIST Architecture", *Proceedings International Test Conference"*, pp 727-736, 2003.

64 Allan Y. Wong, "A Systematic Approach to Identify Critical Yield Sensitive Parametric Parameters", *2nd International Workshop on Statistical Metrology*, pp. 56-61, 1997.

Symbols and Abbreviations

A	Area of chip
$\mathcal{A}$	Set of objects on a device that were observed to have failed
α	Cluster parameter
$\alpha_A(\vec{r})$	Probability that a defect produced by a cluster centered at $\vec{r}$ will fall within the area of the chip centered at the origin
A, B	Partitions of SLAT patterns
BIST	Built In Self Test
β_j	Logarithm of the expected yield of component of type j
b(K;N,p)	Binomial distribution of K fails given N tries and failure probability p
$\mathcal{B}_{lp}$	Set of faults in the backcone defined by latch l and pattern p
b_S	Probability that a randomly generated test sequence detects S
c	(Defect) coverage
C	Area of cluster
c_k	(Defect) coverage after the completion of the k^{th} test
γ	Inverse cluster parameter
D	Fault machine value is 0, good machine value is 1
$\overline{D}$	Fault machine value is 1, good machine value is 0.
$\mathcal{d}$	Input voltage range that will produce an output in $\mathcal{S}$
$\mathcal{S}$	Range of logically ambiguous voltages
d_{ik}	Probability that a chip on wafer i will fail first on the k^{th} test step
d_k	Probability that a chip will fail first on the k^{th} test step

$\bar{d}_k$	d_k/y_{k-1}
DL	Fraction of chips that pass all tests, but are still defective; equal to $(Y-Y_0)/Y$
η	Expectation value of X_iX_j, with i and j neighbors
F	Number of defects in the defect list
f(A)	$\ln Y_0(A)$
g(A)	$A^{-1}\ln Y_0(A)$
G(z)	Generating function of P_n
$\mathcal{H}$	$(\forall j)(0 \le q_j \le 1)$
h(i,j)	Measure of fail similarity between objects i and j
I	Number of wafers
J	Number of pairs of die on a wafer
I	Intersection of all regions $\mathcal{R}_i$
$\mathcal{J}_{\text{intersection}}$	Intersection of faults in backcones
$\mathcal{J}_p$	Intersection of faults in backcones corresponding to pattern p
$\mathcal{J}_{\text{SLAT}}$	SLAT based combination of faults in backcones
$\mathcal{J}_{\text{union}}$	Union of faults in backcones
J_{++}	Number of pairs of die with both die passing
J_{+-}	Number of pairs of die with one die passing and one failing
J_{--}	Number of pairs of die with both die failing
k	Test step label
K	Number of devices having passed some initial portion of the test sequence
κ	Probability that a failing device is also in the characterization group
k_f	Label of final test step
λ	Distribution strength of clusters
L	Likelihood ratio
Λ	Likelihood ratio
LLR	LogLikelihoodRatio; equal to $-\ln\Lambda$
M	Number of chips on a wafer
M	Set of multiplet pins
$m(\{n_i\}; N; \{p_i\})$	Multinomial distribution given N tries and n_i fails with probability p_i in group i

μ	Average number of defects on a chip
μ	Mean
m_i	Number of ICs on which object i is known to have failed the tests
M_i	Number of devices on which object i was tested
MISR	Multiple Input Shift Register
N	Number of chips to be tested
ν	Distribution strength of defects in a cluster
n_C	Average number of defects in cluster equals νC
N_{def}	Number of defective chips that pass all tests
N_{DF}	Number of degrees of freedom
n_i	Number of ICs on which object i is known to have passed the tests
ν_i	Strength of primitive polluter i
n_{ij}	Number of units of type j in the object i
n_{ij}	Number of chips on wafer i failing test step j
N_j	Number of chips failing the j^{th} test step
ν_j	n_{ij}/s_i
ν_0	Distribution strength of defects in uniform background
N_{pass}	Number of chips passing all tests.
$\nu(\vec{r})$	Aggregate strength of primitive polluters at $\vec{r}$; equal to $\sum \nu_i d\vec{r}$
Ξ	Set of all of all failing objects on a device.
O_i	Number of devices on which object i failed the object tests
O_{ij}	Number of devices on which both objects i and j failed the object tests
p_0	Overall wafer yield
p_i	Yield in region i
PI	Primary Input
π_i	Occurrence probability of defect i
P_n	Probability that a chip has a defect complex of size n.
PO	Primary Output
p_S	Probability that chip has defect complex S
q_j	Probability that component j is defect free

Q_n	Probability that a chip with defect complex of size n will pass the first k tests
Q(z)	Generating function of the distribution of the number of defects on the chip produced by a single cluster
r	measure of deviation $-2\ln\Lambda$ from its expected value
$\vec{r}$	point in two dimensional space, for example on a wafer
R	Radius of circular chip
R	Probability that at least one of the objects fails the object tests.
ρ	Radius of cluster
ρ	Measure of commonality; equal to $\frac{-2\ln\Lambda - N_{DF}}{\sqrt{2N_{DF}}}$
$\mathcal{R}_i$	Region in which u_i is less than 1
r_j	$\ln(q_j)$
S	Circle with radius $R + \rho$
S	Defect complex on a chip
$\mathcal{S}$	Region in which all q_j are positive
S	Set of splat pins
$S_{\mathcal{A}}$	Set of pins in splat corresponding to partition $\mathcal{A}$.
s@1	Stuck-at 1 fault.
s@0	Stuck-at 0 fault.
s_i	Size of object i.
SLAT	Single Location At a Time
SLOR	Spatial Log Odds Ratio
t_i	Probability that a single cell in object i will fail
TFSF	Number of latches that fail on both the tester and the simulator
TFSP	Number of latches that fail on the tester but pass on the simulator
TPSF	Number of latches that fail on the tester and pass on the simulator
TPSP	Number of latches that fail on both the tester and the simulator
u_i	Probability that object i will fail the object tests
y	Probability that a chip will pass all tests; equal to <Y>
Y	Yield; equal to N_{pass}/N
y(c)	Probability that a chip will pass the tests having coverage c
y_j	Probability that a chip will not fail the first j test steps

Y_j	Yield after completion of the j^{th} test step
y_0	Probability that a chip is defect free; equal to $\langle Y_0 \rangle$
Y_0	Fraction of chips that are defect free
y_S	Gross yield
$\sigma^2(H)$	Variance of random variable H
$\langle H \rangle$	Expected value of random variable H
$\widehat{H}$	Maximum likelihood estimate of random variable H

Index

E

F

G

H

I

L

U

W

Y