Spatial Yield Modeling for Semiconductor Wafers

by

Agha Irtaza Mirza

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May, 1995

Author .....
Department of Electrical Engineering and Computer Science
May, 1995

Certified by .....
Alvin W. Drake, Ford Professor of Electrical Engineering
Department of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by .....
Stephen C. Graves, Professor of Management Science
Co-Director, Leaders For Manufacturing Program
Thesis Supervisor

Certified by .....
Geoffrey O'Donoghue, Company Supervisor

Accepted by .....
F. R. Morgenthaler
Chair, Department Committee on Graduate Students
Department of Electrical Engineering and Computer Science

Spatial Yield Modeling for Semiconductor Wafers

by

Agha I. Mirza

Submitted to the Department of Electrical Engineering and Computer Science on May, 1995, in partial fulfillment of the requirements for the degree of Master of Science

Abstract

The distribution of good and bad chips on a semiconductor wafer typically results in two types of regions, one that contains both good and bad chips distributed in a random fashion, called a "non-zero yield region", and the other that contains almost all bad chips, called a "zero yield region". The yield of a non-zero yield region is modeled by well understood expressions derived from Poisson or negative binomial statistics. To account for yield loss associated with zero yield regions, the yield expression for non-zero yield regions is multiplied by Y_o, the fraction of the wafer occupied by non-zero yield regions. The presence, extent, and nature of zero yield regions on a given wafer provide information about yield loss mechanisms responsible for causing them.

Two statistical methods are developed in this work, treating the wafer map with locations of good and bad chips as a two dimensional binary lattice. The first method detects the presence of zero yield regions on a given wafer map by finding the presence and type of any spatial dependencies in the distribution of bad chips on the wafer map. This method uses a set-theoretic statistical image analysis tool, called the Aura Framework which is based on ideas similar to Gibbs/Markov Random Fields. The second method is a statistical method to measure Y_o for a given wafer and is based on hypothesis testing on nearest neighbors of bad chips. Both methods developed are applied to test data and to actual wafer map data. Results show that the modeling of the distribution of good and bad chips on wafers in terms of zero and non-zero yield regions is highly accurate on the test set of wafers. This provides evidence to support the validity of employing yield models for yield prediction and manufacturing planning.

Thesis Supervisor: Alvin W. Drake
Title: Professor of Electrical Engineering

Thesis Supervisor: Stephen C. Graves
Title: Professor of Management Science

Thesis Supervisor: Geoffrey O' Donoghue
Title: Company Supervisor

Acknowledgements

Many people deserve credit for providing support in one way or the other to make this work possible. I would like to take this opportunity to acknowledge my advisors, my family, and some others whose contributions are more prominent.

I owe many thanks to Al Drake for being very caring and always guiding me in my best interest and to Steve Graves for his patience, advice, and constant support. Al Drake's and Steve Graves' enthusiasm and encouragement made this thesis one of my best experiences at MIT despite the challenges that I faced in accomplishing the thesis goals.

I thank Rosalind Picard and Sanjoy Mitter for helpful discussions, Gary Cheek for taking time out of his busy schedule to provide me with insightful discussions and meticulously reviewing parts of this thesis, Geoff O' Donoghue for initiating and devoting time to this project at the company, and Stacy Ho for helping me in several ways at the company.

I thank all my friends for making my years at MIT good fun.

I thank my beloved and supportive family. I particularly wish to thank my parents for their selfless love and always providing, above and beyond their means, the best for me.

I dedicate this work and both my degrees from MIT to my parents.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Electronic integrated circuit (IC) chips typically consist of a few hundred to several million devices interconnected to provide specific electrical functions. The ratio of the number of chips that are functional after completion of production to the number at start of production that have the potential to be functional is defined as Yield. This yield is almost always less than 100% because deformations occur on the wafer during the process of manufacturing IC's. Some of the deformations may cause the chips to become nonfunctional giving a yield level of less than 100%.

Yield of IC chips is one of the most important factors responsible for semiconductor business profitability. Although rapid technological advancements in electronic devices and circuits have been expanding the semiconductor business by enhancing the functional potential of IC chips, a side effect has been an increased difficulty in obtaining high yields. The progress in the semiconductor industry has been primarily caused by integration of a larger number of devices and the use of smaller dimension devices in IC's. The increase in integration and decrease in device dimensions has resulted in an overwhelming increase in the complexity of semiconductor manufacturing and sensitivity of IC performance to process fluctuations. This has enhanced the ever present challenge of manufacturing functional IC's, thereby making the task of obtaining high yields more difficult.

The IC chips, also called dice, are fabricated on semiconductor wafers. The cost of producing each wafer is constant for a given product on a particular manufacturing process. The number of good chips obtained from a given wafer represents the revenue to be generated from sales from the wafer. Hence, a minimum yield level is required for the

manufacturer to stay in business and greater yield is equivalent to increase in profits. Since yield loss and yield variability represent a direct loss of profits, yield modeling and analysis form an important topic for semiconductor manufacturing.

The occurrence of deformations on wafers and the associated chip failures have been modeled as various random processes [1, 2, 4]. Several of these yield models are found useful in the planning, optimization, and control of IC manufacturing [1, 2, 4, 6]. The IC manufacturing yield can be classified into several categories; examples include line yield, probe yield, and package yield.

$$Y = Yline \times Yprobe \times Ypackage, \tag{1.1}$$

where Y is the final product or chip yield. Line yield, *Yline*, is defined as the ratio of wafers properly fabricated to wafers started from the fab line. Line yield loss may be caused by fab line misprocessing or wafer breakage. Probe yield, *Yprobe*, is defined as the ratio of the number of good dice to the total number of potentially good dice on a given wafer. The fraction of chips that are tested to be functional after the dice on the wafer are packaged is defined as package yield, *Ypackage*. In most modern IC manufacturing foundries probe yield is found to dominate the overall product yield. Hence, we focus on probe yield only, i. e. we make the following assumption.

$$Yline = Ypackage = 1, Y = Yprobe.$$

The probe yield can be considered as a product of two types of yields, gross yield, denoted by Y_o, and defect limited yield, denoted by Y_r, [1, 2, 4, 13].

$$Y = Y_o \times Y_r. \tag{1.2}$$

Gross yield loss is caused by one of the two broad classes of deformation mechanisms called gross random deformation mechanisms. The other category of deformation mechanisms cause defect limited yield loss and is called local random deformation mechanisms.

The gross deformation mechanisms cause continuous regions of wafers to have no or very few functioning chips. The chips that become bad due to the local deformation mechanisms are found randomly distributed on regions of wafers that do not encounter gross deformation mechanisms. Figure 1.1 shows an example of a wafer that has a region with almost random distribution of good and bad chips and a continuous region with almost all bad chips. We represent a bad chip by a black site and a good chip by a white site to reflect the fact that chips tested bad on a wafer are marked with a red ink dot and chips tested functional on the wafer are left blank.

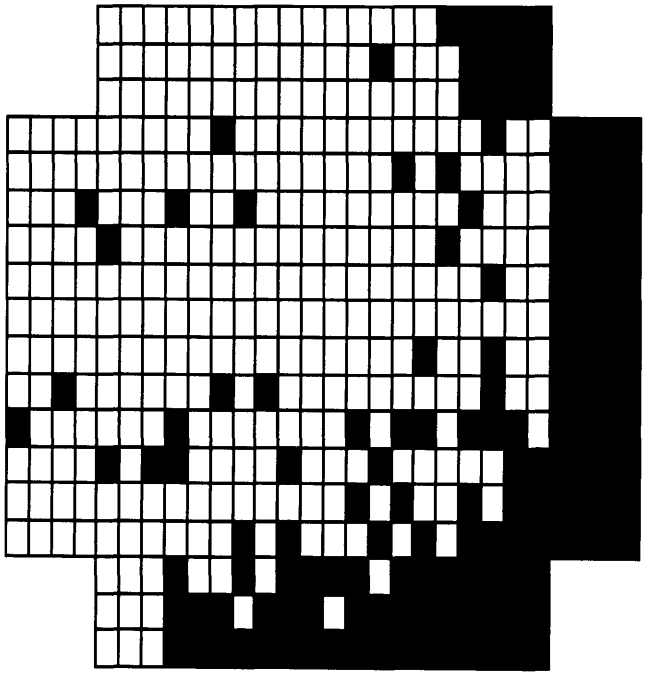


Figure 1.1: A wafer of a commercial IC product that has a region with both good and bad chips mixed in a random fashion and a region with almost all bad chips.

1.2 Problem Description

This thesis addresses the problem of finding the regions of almost all bad chips, caused by gross deformation mechanisms, on a given wafer. Two statistical methods are developed and investigated. The first method takes as input a wafer map, a two dimensional map that shows locations of good and bad chips on a wafer, and determines whether the wafer con-

tains regions of almost all bad chips. The second method takes the wafer map and the output of the first method as inputs and statistically finds those bad chips that belong to regions of all bad chips. Statistical methods, as opposed to algorithmic methods, are needed to detect and find the regions of all bad chips caused by gross deformation mechanisms because the presence, size, location, and shape of these regions and wafer yield all vary from wafer to wafer even for wafers of the same product processed together.

Detecting and finding regions of all bad chips caused by gross deformation mechanisms have both yield analysis and yield modeling value. Detection of these regions indicates the presence of gross deformation mechanisms in the manufacturing process. A fab is usually considered mature when it has eliminated all gross deformation mechanisms and has local deformation mechanisms well under control. The local deformation mechanisms are considered somewhat inherent to the manufacturing environment. They are present even in a well-controlled and well-understood process environment. That is, control can be exercised to limit the local deformation mechanisms but these cannot be eliminated perfectly. Gross deformation mechanisms, on the other hand, can be eliminated almost completely as the process becomes better understood. Hence, identifying the presence of gross deformation mechanisms reflects the opportunity for yield improvement. This opportunity for yield improvement can be materialized by taking measures to find the specific nature of gross deformation mechanisms responsible for the gross yield loss present. The second method which finds those bad chips that belong to regions of all bad chips enables calculation of percentage of yield loss that is caused by the gross deformation mechanisms. This gives a measure of the severity of gross yield loss or, equivalently, a measure of the opportunity for yield improvement that exists in elimination of gross deformation mechanisms.

The modeling value of the proposed problem lies in the second method enabling the calculation of gross yield, Y_o. In fact, Y_o is defined to be the fraction of the wafer that is occupied by regions which did not encounter gross deformation mechanisms, i. e. regions which were available for contribution to final yield. Hence, Y_o is a multiplicative factor that scales the yield predicted by defect limited yield models, which are mathematical expressions, to account for the gross yield loss. The measured wafer yield and the calculated Y_o together with relation 1.2 give a measure of the defect limited yield, Y_r. This defect limited yield number is the one that should be compared with yield predicted by defect limited yield models to establish the success of these models.

Another important contribution of this thesis is to verify the model for the spatial distribution of deformations, and therefore good and bad chips, that is commonly assumed in successful yield models. This model assumes that the distribution of good and bad chips on a semiconductor wafer typically results in two types of regions, one that contains both good and bad chips distributed in a random fashion and the other that contains almost all bad chips. Analysis supporting this model is carried out by simultaneous application of the two statistical methods developed in this work to wafer map data of commercial products.

1.3 Thesis Overview

The next chapter provides background on semiconductor manufacturing and yield modeling to motivate the problem addressed in this thesis. The attempt is to provide a coherent and well-connected picture starting from a description of basic manufacturing steps, through examples of deformation mechanisms, to presenting yield models that have been used most widely. It is the two components of yield models, gross and defect limited, and their relation to the two categories of deformation mechanisms, gross and local, that motivates the problem addressed in this thesis. Chapter 3 contains the descriptions and supporting theories of the two statistical methods developed to solve the problem addressed.

This chapter also examines the effectiveness of the methods developed by applying them to a test data set. Chapter 4 analyzes wafer map data for actual semiconductor IC products by first using common methods of analysis and then by applying the methods developed in chapter 3. It is found that wafers indeed encounter gross yield loss. Also, this analysis verifies the model for spatial distribution of bad chips on wafers. We provide conclusions and recommendations for yield management in chapter 5.

Chapter 2

Semiconductor Manufacturing and Yield Modeling

2.1 Introduction

The semiconductor IC manufacturing process involves modifying physical composition and geometry of semiconducting material in complex patterns and in extremely small dimensions to achieve desired electrical behavior from the IC. Semiconductor manufacturing consists of three components [6]; the fabrication process, the geometrical description for fabrication or IC layout, and fluctuations and disturbances in the manufacturing environment. The IC layout defines patterns in which geometry of the semiconducting material is changed and regions in which impurities are added to precisely alter the electrical behavior of an IC. Fluctuations and disturbances are inherent to most manufacturing

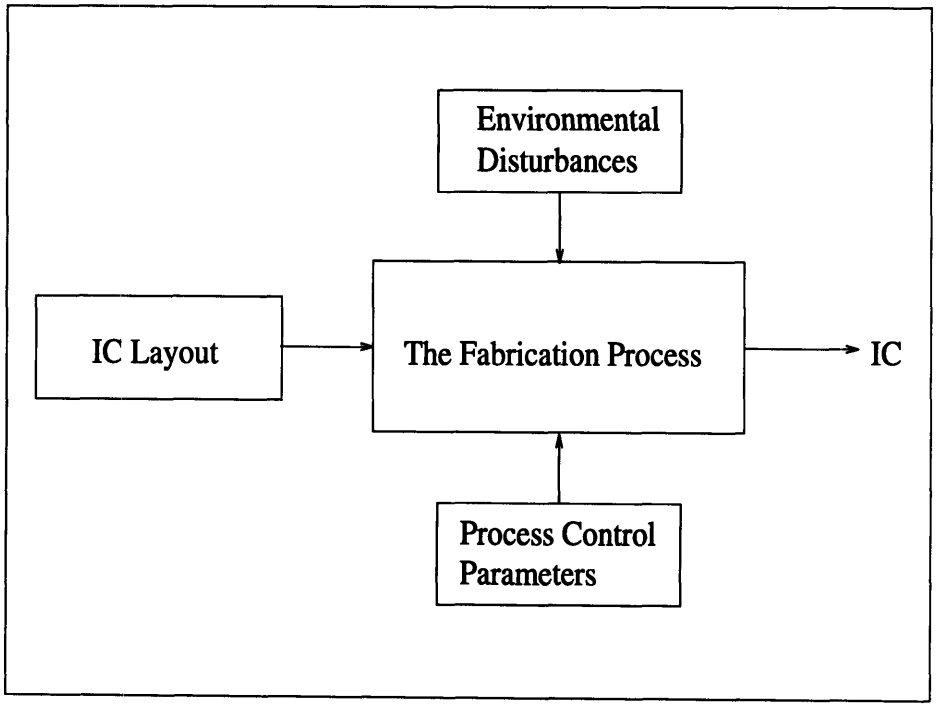


Figure 2.1: A simplified model of IC manufacturing.

environments and arise due to human errors, machine and material instabilities, and ran-

domly changing environmental factors. As a result of these disturbances the outcome of the IC fabrication process is stochastic. Figure 2.1 indicates that a manufactured IC is the output of a fabrication process whose inputs are IC layout and normal environmental disturbances or process perturbations.

This chapter discusses the concepts and issues involved in accurate yield modeling after providing background on the semiconductor manufacturing process. To develop an understanding of the various yield models requires a basic understanding of the IC fabrication process and digressions from desired fabrication parameters that may be caused by the disturbances and fluctuations in the manufacturing environment. The first section of this chapter describes the basic steps that constitute an IC manufacturing process. This section is followed by a discussion of deformation mechanisms that arise due to environmental disturbances and manufacturing errors. The next section discusses the concept and use of 'critical area'. The critical area is an important component of yield modeling. Critical area facilitates obtaining a measure of the probability that a particular deformation mechanism will cause a circuit failure by simulating the interaction of IC layout with the deformation mechanism. The subsequent section provides a review of yield models that have been widely used in semiconductor manufacturing and are also rather intuitive from a mathematical modeling point of view. However, these models account for yield loss due to one of the two broad classes of deformation mechanisms, namely local random deformation mechanisms. A discussion on modeling of yield loss due to the other class of deformation mechanisms called gross random deformation mechanisms is provided in the last section. This final section also defines the need for the statistical methods that are developed as the main contribution of this work and provided in the next chapter.

2.2 The IC Fabrication Process

Electronic IC chips consist of a large number, from a few hundred to several hundreds of

thousands, of electronic devices interconnected to provide a specific electrical function. The electrical characteristics of an individual electronic device depend upon the three dimensional impurity distribution in the region of the semiconducting material occupied by the device and its interaction with other devices. Impurities and other elements when controllably added to a semiconducting material change its electrical properties in a desired manner.

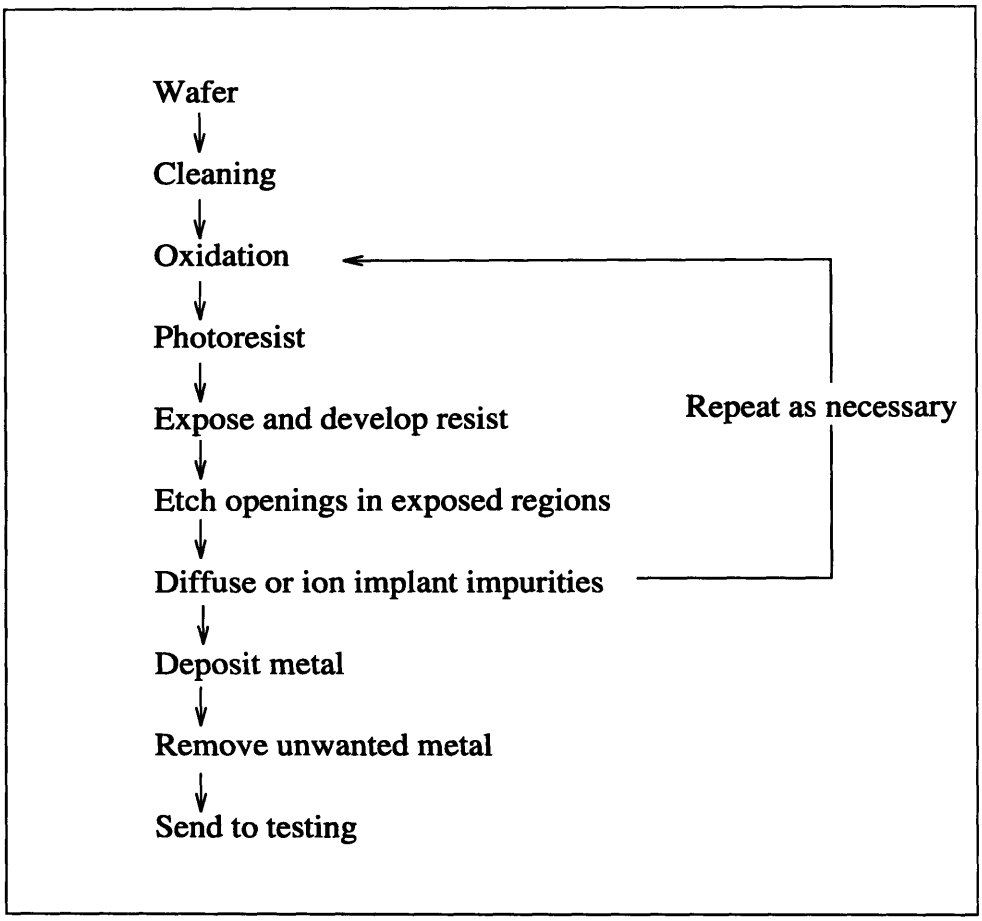


Figure 2.2: Basic processing steps in wafer fabrication.

The process of IC manufacturing involves a specific sequence of basic processing steps performed on a semiconducting material wafer. Silicon is widely used as the starting semiconducting material in most modern fabs because it usually offers cost and performance advantages compared to other semiconducting materials. The objective of semi-

conductor fabrication is to successively create precisely defined regions in the wafer that contain controlled impurity materials. This process is achieved by repeatedly subjecting the wafer to a set of processing sequences [5]. Such a set of sequences consists of covering the wafer surface with a material impervious to the penetration of certain impurities, opening selected areas on the wafer using photo sensitive masks, depositing the desired impurities in the exposed regions, covering the opened regions and applying heat treatment at some of the steps in the sequence. After layers containing the desired impurities are formed, metal and other conducting layers such as polycrystalline silicon are deposited to connect devices to each other. Contacts are provided for electrical connection of the IC to the outside world [5]. A summary of basic wafer fabrication steps is shown in Figure 2.2.

The first step is to clean the wafer using carefully chosen chemicals to eliminate any contaminants or dust particles that may be present on the wafer surface. Cleaning the wafer and its surroundings for dust and foriegn materials is crucial to minimize defects in the IC structure. The cleaned wafer surface is covered with a barrier layer of an electrically insulating material, silicon dioxide or silicon nitride, by heating the wafer in the presence of an appropriate gas thereby growing an appropriate layer.

The next few steps are typical photo processes which aims to remove the barrier layer from selected regions to expose the silicon surface underneath. The wafer is held on a vacuum chuck and spun at high speed while photoresist, a light sensitive material, is applied to it in a liquid form. A photoresist layer of uniform thickness is thus formed on the entire wafer surface. The wafer is then heated at controlled temperatures in an atmosphere of air or nitrogen to improve adhesion of the photoresist to the wafer surface. This step of heating is called soft-baking. The wafer is then aligned with a glass plate of its own size called a mask. The mask is coated with emulsion or metal in a desired pattern of opaque and transparent regions. After the mask has been aligned with the wafer it is exposed to ultra-

violet light which weakens the exposed regions of the photoresist. The mask is removed and the wafer is treated with chemicals to wash away the exposed photoresist. This is referred to as a positive photoresist process.

The photoresist on the unexposed parts of the wafer is made harder and more chemically resilient by heating the wafer at controlled temperatures for specific lengths of time. This step is called hard baking which is followed by an etching step. Etching consists of treating the wafer with chemicals that dissolve the underlying silicon dioxide layer in the exposed regions and do not affect the photoresist material, thus exposing the silicon surface in the uncovered regions only. The next step consists of adding desired impurities in controlled amounts to the exposed regions of silicon. Impurity addition may be done either through diffusion or ion implantation. Diffusion is the process of adding an impurity to silicon by heating the wafer in presence of the impurity in gaseous form. Ion implantation consists of adding an impurity to silicon by bombarding it with accelerated impurity ions in a high electric field. The choice between diffusion and ion implantation depends on the type of impurity to be added and the particular processing step. After depositing the required impurity, the openings are closed by covering them with an insulator, e. g. silicon dioxide, layer.

The above sequence of steps is repeated until predefined regions of impurities are formed in successive layers of the wafer. The last few mask steps consist of depositing metal or polysilicon (a conducting material) layers to provide electrical connections among various regions on the wafer. The conducting layers are formed by treating the wafer with metal or polysilicon vapors and letting them settle only on the exposed regions of the wafer. The entire wafer is finally covered with a silicon dioxide protective layer with small openings made in selected regions. These openings are used to provide metal contacts to serve as the IC's electrical communication with the outside world.

2.3 Deformation Mechanisms

Fluctuations in control parameters of the fabrication process, presence of foriegn material in the manufacturing environment and other sources of unplanned processing may lead to deformations in the desired IC layout. The resultant deformations can be either geometrical or electrical in nature. A geometrical deformation is a physical, usually visible, anamoly in the planned geometrical structure of the IC while an electrical deformation is a change in the planned electrical properties. An electrical deformation could arise due to defects in the crystal lattice of silicon causing a leakage current or an alteration in type or an amount of impurities added to silicon resulting in a modification of sheet resistance or depth of a key layer.

As the complexity of semiconductor manufacturing is increased, the IC performance becomes increasingly sensitive to deformations. A discussion of various deformation mechanisms is important for developing an understanding of the IC performance sensitivity to deformation mechanisms. A comprehensive account of all possible ways in which deformations may arise is beyond the scope of this section and would be difficult to provide due to the large number of deformation sources. We describe commonly occurring deformation mechanisms by considering disturbances and unplanned processing that may take place at each processing step discussed in the previous section. Each chip on a wafer may encounter deformations of some of the types described below and perhaps also others. Since all deformations interact with the electronic circuit differently, some may become fatal and cause the chip to malfunction.

One of the most common sources of deformations or "defects" in the IC result from particles of dust and foreign material that are present in the manufacturing environment. A device failure may occur due to the presence of a defect on the silicon surface caused by unwanted particulate matter. A particle present on the mask or trapped between the barrier

layer and the photoresist may cover an exposed region or may expose a covered region. For instance, if a particle is present on the transparent region of the mask, it may avoid exposure of a spot on the wafer. The incorrectly exposed spot is a defect and may result in a device failure. Also, particles are often responsible for low yield at wafer edges due to their increased density in this area. Wafers are often carried between processing steps in plastic carriers called boats. The wafers are held between grooves in boats such that only wafer edges touch the grooves. Particles can enter the boats from sides only and are electrostatically attracted to the nearest edge of the wafer. Thus particles, clustered at the wafer edges, are likely to result in the formation of defects at the edges with a higher density than on any other location on the wafer.

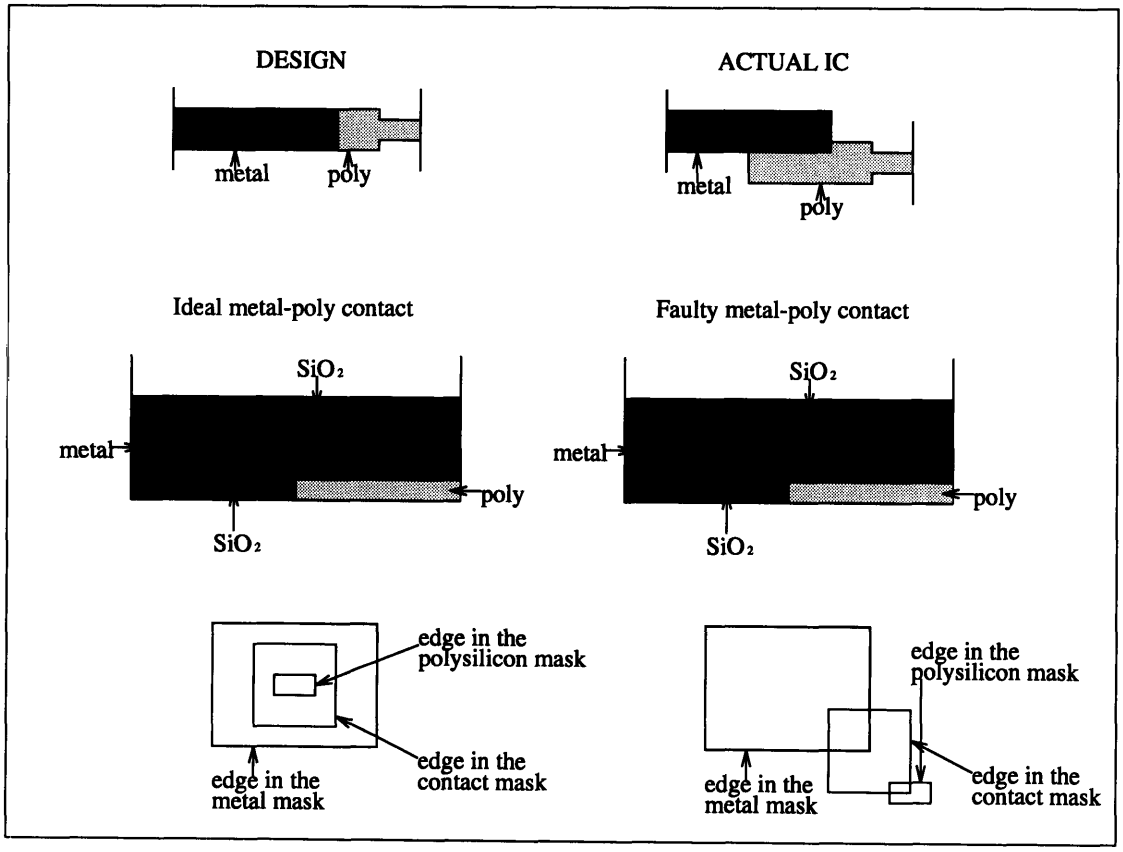


Figure 2.3: An illustration of mask alignment errors.

Scratches on or nonuniformities in the thickness of the photoresist layer are likely to result in the formation of defects. A thin region may be washed away when treated with chemicals even though it was not exposed or a thick region may not dissolve in chemicals even though it was exposed to ultraviolet light and was intended to be washed away in the photolithographic step. Such a processing deformation will result in over- or under-etching of the underlying barrier layer.

The mask alignment and exposure steps of photolithography are particularly sensitive to defect formation. Misalignment between the mask and the wafer, which may occur due to mechanical vibrations in the processing equipment or limited optical resolution, causes pattern boundries to be displaced. Mask misalignments in more than one step may cause electrically isolated regions to overlap and electrically connected regions to become isolated. The mask to mask alignment is known as registration. Such registration errors are illustrated in Figure 2.3.

The most common photolithography related defect is caused by 'misprints' or dust particles on the mask surface. This type of defect results in the printing of extra or missing pattern which in turn may cause a device malfunction, short circuit, or open circuit. Figure 2.4 indicates examples of short and open circuits caused by extra and missing patterns in a metal layer. Also, the photo mask, if in contact with the wafer surface, may damage points or regions of the wafer surface. For this reason, the mask in some processes is aligned with the wafer without making contact with the wafer surface. Another photolithography defect arises due to diffraction caused by mask edges of the ultraviolet light used for exposure. This defect may cause parts of the wafer to remain unexposed at the edges. This is often one of the causes for low yield at the wafer edges [6].

Errors in the application of etching chemicals may also lead to shifts in pattern boundries. Under-etching will result if the barrier layer in all the exposed regions is not removed

with the etching chemical. Over-etching may occur if the etching chemical attacks or undercuts the photoresist material.

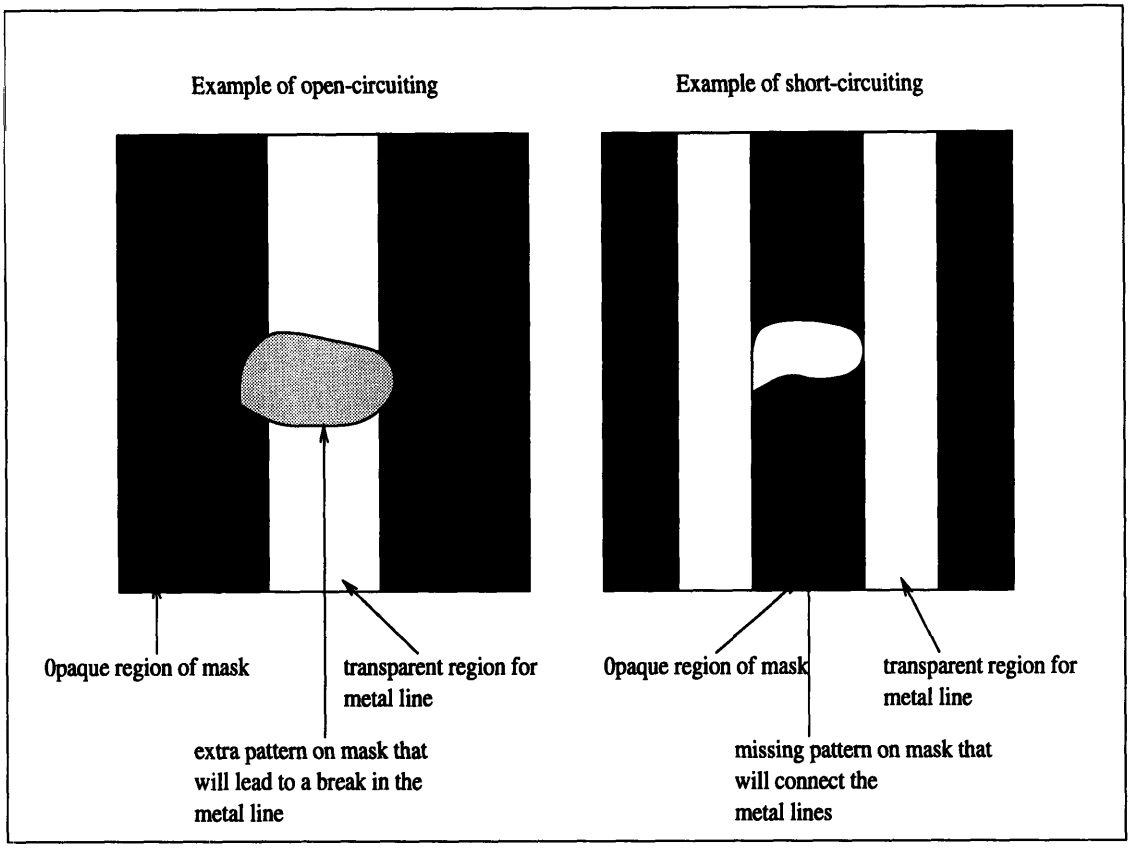


Figure 2.4: Mechanisms that introduce spot defects into the metal layer.

Errors in addition of impurities through diffusion or ion implantation may cause electrical or parametric failures. Variations in the amount or type of the required impurity will cause device characteristics to differ from the intended design value. The result may be a change in the designed characteristics of the IC leading to an electrical deformation or parametric failure of the IC.

The wafer is subjected to thermal cycles during diffusion, oxidation, hard baking, and other various steps. Variations in temperature levels and duration of heat cycles may cause defects in the crystal lattice structure of silicon. Crystal defects may result in charge leakage across junction of regions with different impurity types. Charge leakage defects are

commonly referred to as 'pipes' [13]. Very excessive heating may result in physical deformation of the wafer itself which may tend to be problematic for a subsequent processing step. For example, a wafer deformation due to excessive heating may become cause for mask misalignment.

Conducting layers of metal and polysilicon are electrically separated by silicon dioxide layers. Openings in such a silicon dioxide layer, called pinholes, will permit charge leakage between conducting layers [1]. A pinhole, depending on its size, may result in a short circuit failure of the IC.

The deformation mechanisms discussed above can be broadly classified into two categories, local and gross [6]. If a deformation mechanism affects all chips in a region of the wafer or the entire wafer in a similar fashion , it is called a gross deformation mechanism. However, if a deformation mechanism affects small areas of isolated chips independent of other chips on the wafer, it is called a local deformation mechanism. Examples of local deformations include pinholes, pipes, and photolithography spot defects. Manufacturing errors such as mask misalignments, variations in process temperatures and heating durations, and nonuniform impurity concentrations across the wafer form some examples of gross deformation mechanisms. Both local and gross disturbances may result in either geometrical or electrical deformations, although local disturbances more usually cause geometrical deformations while gross disturbances more usually result in electrical or parametric deformations of the IC.

All the process induced deformations, both gross and local, are random. The chips that fail due to local random disturbances should be found randomly distributed on the wafer. Gross disturbances will affect chips that are located in large groups or clusters on the wafer owing to the fact that gross disturbances affect regions of wafers or an entire wafer. The nature of these disturbances is consistent with the observation that the distribution of

failing chips on wafers can be broadly classified into two types of regions. One of these types of regions consists of both functional and nonfunctional chips distributed in some random fashion and the other consists of almost all failing chips. We term the former a "non-zero yield region" and the latter a "zero yield region". It can be surmised that the zero-yield regions on a wafer encountered one or more gross deformation mechanisms and the non-zero yield regions encountered one or more local disturbances. Visual inspection shows that the presence of, location, size, and shape, of zero yield regions all vary from wafer to wafer even for wafers in the same lot. Hence gross deformation mechanisms are random in nature. Yield loss due to local random disturbances is referred to as "defect limited yield loss" and the yield loss due to gross random disturbances is referred to as "gross yield loss".

As a fab matures and the process technologies become better understood, there is a reduction in gross yield loss due to a decrease in gross random disturbances. In such cases the yield loss is dominated by defect limited yield loss. The defect limited yield loss is usually the dominant yield loss even in the presence of gross yield loss. We discuss defect limited yield modeling by first describing one of its essential components called the critical area.

2.4 The Concept and Use of Critical Area

There are two main elements of defect limited yield modeling. The first element involves estimating the expected number of faults on a chip due to various local random deformation mechanisms. The second element, which is described in the next section, involves the statistical distribution of the number of faults per chip. It is the estimation of the expected number of faults per chip that requires the use of critical area.

A local random deformation or defect is a physical anamoly in the planned geometrical pattern of an IC or an isolated defect in the crystal structure of silicon. These deforma-

tions include open circuits in or short circuits between the patterns of an IC conducting layer, pinholes in the insulator between two conducting layers, and pipes. Those defects that result in circuit failures are called fatal defects or faults. Whether a defect becomes a fault depends on the IC's sensitivity to the particular type of defect mechanism. The defect sensitivity of the IC is a function of the defect size and location on chip.

The concept of critical area accounts for the fact that all areas of a layout are not equally sensitive to a defect, i. e. the probability that a specific size defect will result in a circuit failure is not equal on all parts of the layout. Intuitively, the concept of critical area determines the interaction of chip layout with a local defect mechanism to estimate the probability that a defect of a given size from the particular mechanism will result in a circuit fault. This probability is used to calculate the fraction of the total chip area, called the critical area, that is sensitive to defects of a particular size [9, 10, 11]. The average number of faults per chip is given by the product of critical area and average defect density instead of the product of total die area and average defect density.

Each layer or level of the IC has a unique critical area which is determined by the specific layout and deformation mechanisms that are likely to affect the particular level. The determination of critical areas for individual levels is, however, complicated by the fact that a given level may encounter a single defect or multiple defects and defects may affect a single level or multiple levels, i. e. critical area for a given level is not entirely independent of other levels owing to the three dimensional aspect of certain defects [7]. Therefore, calculation of critical areas for real ICs is a very complex task and needs sophisticated software tools that utilize chip layout and knowledge of defect mechanisms as input. These types of tools often use Monte Carlo simulation methods [7].

A simple single layer, single defect example is provided here to demonstrate the concepts involved in the calculation of critical area and its use in estimating the average num-

ber of faults per chip. This example, along with its notation, is adopted from [10]. Consider a metal line of width w and length L ($L >> w$) deposited on an insulator substrate of length L and width X, as shown in Figure 2.5. A missing region in the metal line which may arise because of a photo defect, metal deposition error or liftoff error, may cause an open circuit in the IC. It is assumed for simplicity that the defects that may arise are circular with diameter x and that an open circuit will occur if the metal pattern is completely broken. In reality, the defects are irregularly shaped splotches and open circuits will take place if the width of the metal is reduced to less than some minimum size depending on the technology.

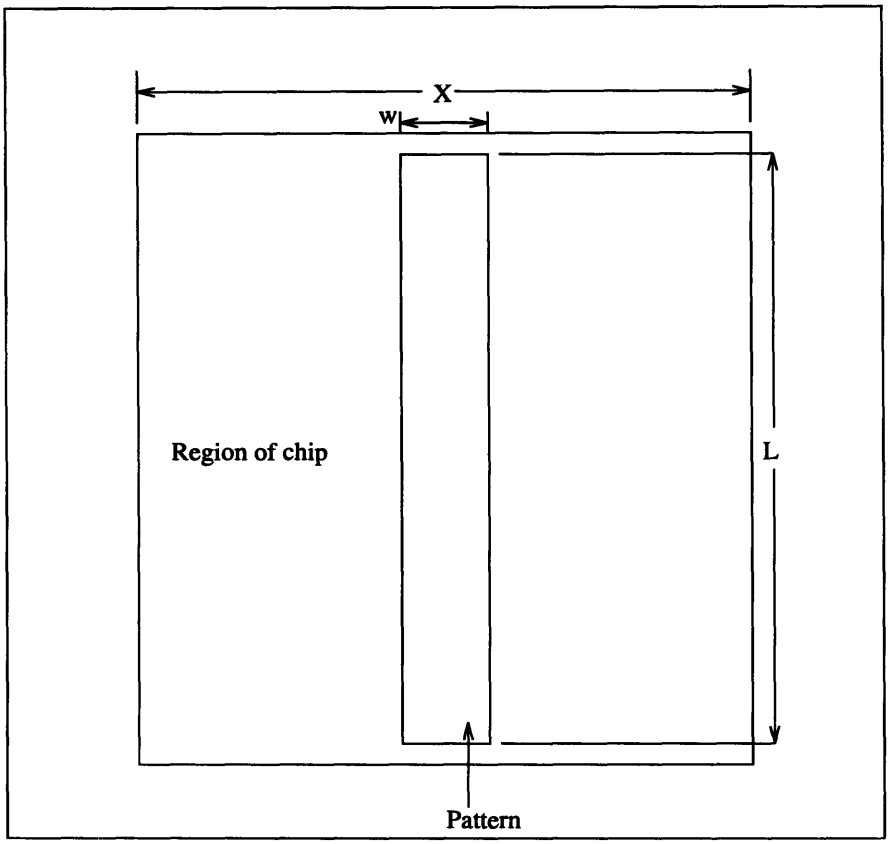


Figure 2.5: A metal line of length L and width w in a region of width X.

For an open circuit to occur due to a complete breakage of the metal line, two conditions must be met. The first condition is that the diameter of the defect must be at least as

large as the width of the metal. The second condition requires that the center of the defect must lie in a critical interval which is determined, in this case, by the metal width and the defect size. The critical interval is defined as the interval for a defect sensitive region within which a defect must fall to cause a chip failure [9, 10]. The critical interval, x_c, for the example considered, is shown in Figure 2.6 and given by,

$$x_c = \begin{array}{ll} 0 & 0 < x \le w \\ (x-w) & w \le x \le w+X \\ w+X & w+X \le x \end{array} \tag{2.1}$$

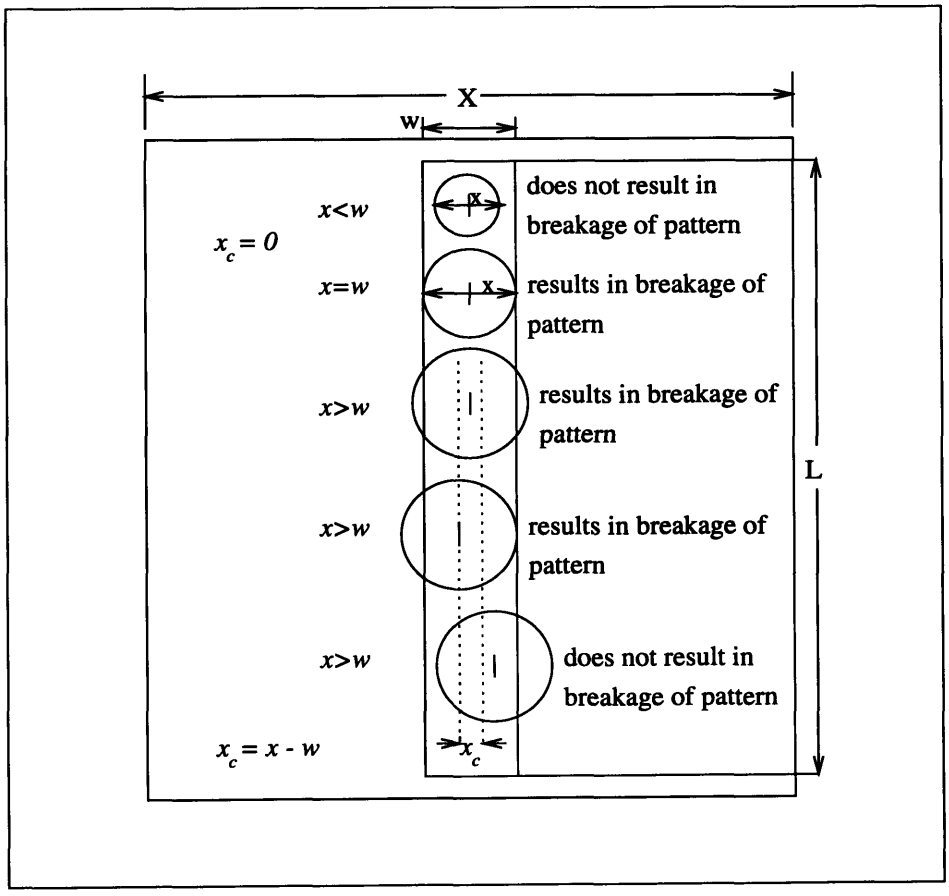


Figure 2.6: Illustration of critical interval x_c for defect of size $0 < x < w+X$.

The ratio of the critical interval and the total width on which a defect can fall is a function of defect size called probability of failure, *POF(x)* [1]. The *POF(x)* is a likelihood

function that gives a measure of the probability that a defect of a given size becomes a fault. For the single pattern example considered, *POF(x)* is shown graphically in Figure 2.7 and is given by,

$$POF(x) = \begin{cases} 0 & 0 < x \leq w \\ \dfrac{(x-w)}{X} & w \leq x \leq w+X \\ 1 & w+X \leq x \end{cases} \tag{2.2}$$

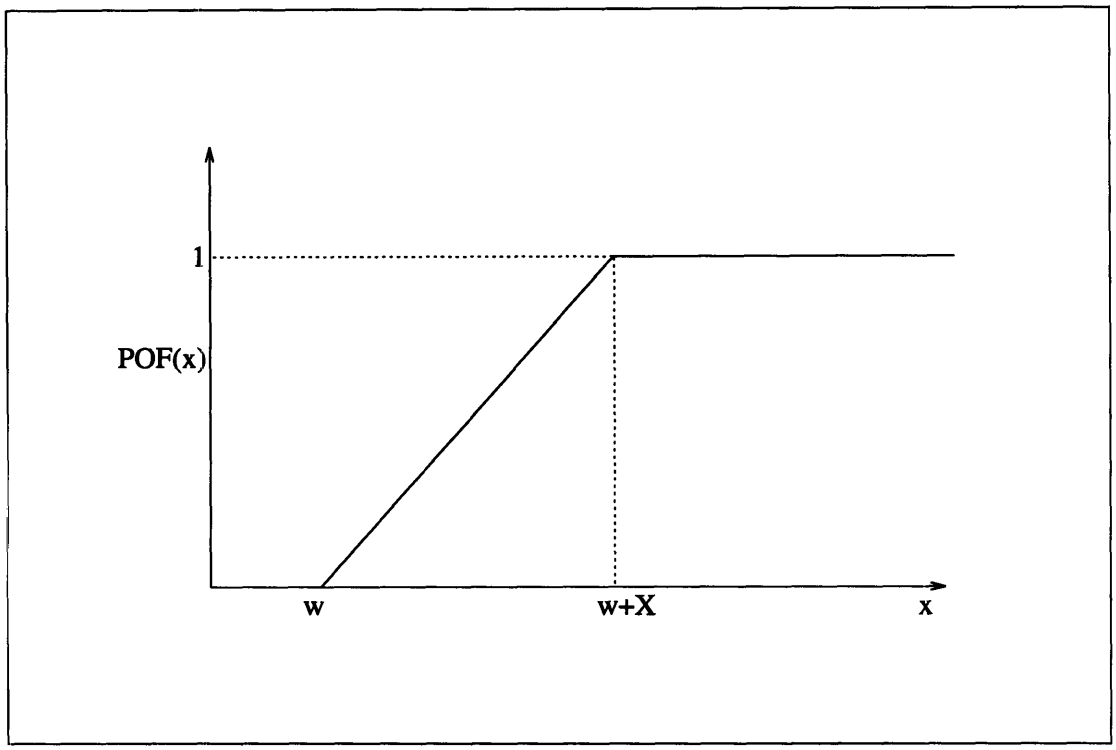


Figure 2.7: Probability of failure for the single pattern example.

The *POF(x)* for any level of a real IC is likely to be different from the simple piecewise linear curve derived for this single level single defect example. However, the *POF(x)* for any given level will follow the general shape illustrated in Figure 2.7. The shape indicates that defects of very small sizes have a small likelihood of causing a failure and the likelihood of failure increases with defect size. Defects of sizes greater than a particular size will always cause chip failure, i. e. have a 100% probability of resulting in a chip fail-

ure.

The product of *POF(x)* and the defect size distribution, *S(x)*, is used to account for the fact that defects of different sizes occur with different likelihood. The product of *POF(x)* and *S(x)* is a function called defect sensitivity and is denoted by $\phi(x)$.

$$\phi(x) = POF(x) \times S(x) . \tag{2.3}$$

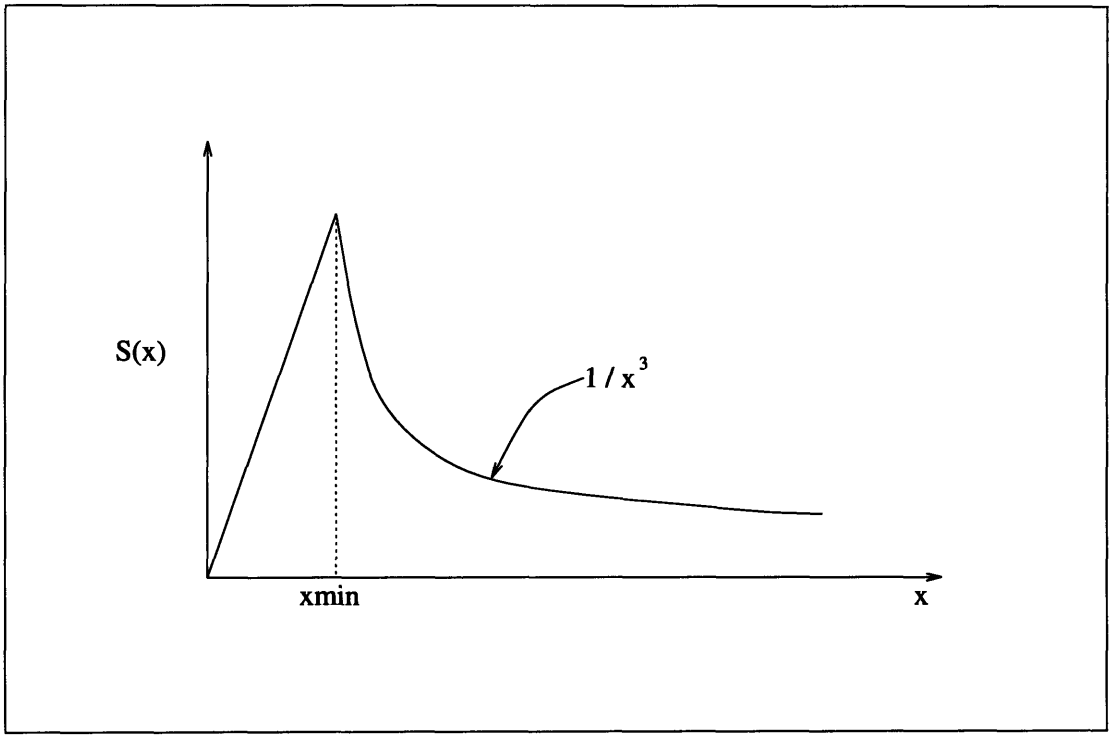


Figure 2.8: Commonly assumed defect size distribution.

The defect size distribution, *S(x)*, and the average defect density, D, which is the average number of defects per unit of chip area, are obtained using observations from defect monitors. Defect monitors are regions in scribe lanes or test dice at selected sites on the wafer that are used to investigate the size distribution and density of defects. There has been reported empirical evidence suggesting that *S(x)* varies inversely as cube of the defect size, x [2]. Figure 2.8 shows shape of defect size distribution typically derived from reported fab data [2]. For sizes less than the minimum resolution of test equipment, x_{min}, a

linear relation is usually assumed. The exact shape of defect size distribution depends on the fab, the process, and the processing equipment. Also, the shape is not constant in time for a given wafer and changes continuously.

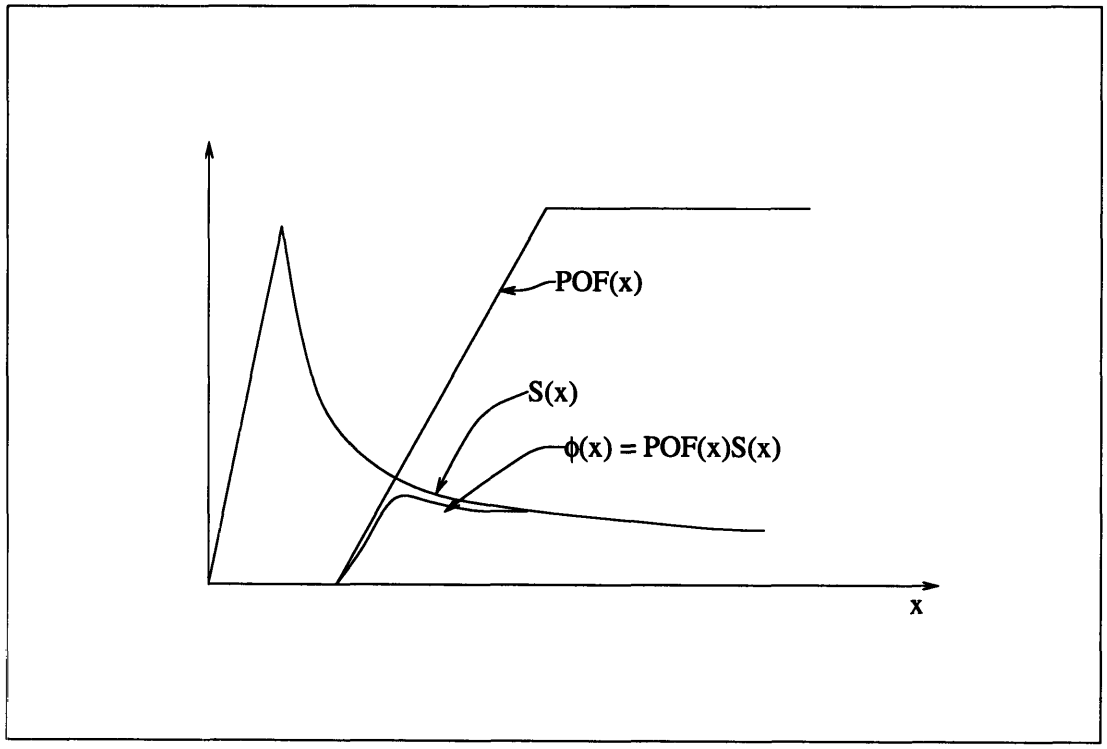


Figure 2.9: The defect sensitivity curve for the single pattern example.

The critical area is calculated by multiplying the die area with a scale factor given by the area under the defect-sensitivity curve which is shown in Figure 2.9.

$$A_c = A_t \times \int_0^{x_{max}} \phi(x)\, dx. \tag{2.4}$$

where A_c is the critical area, A_t is the total die area, and x_{max} is the largest defect size for a given level of the IC. The product of average defect density, D^i, and the critical area, $A_c{}^i$, for a particular level i in the IC structure gives the average number of faults, denoted by λ^i, for the level i in the IC.

$$\lambda^i = D^i \times A_c^i. \tag{2.5}$$

The use of critical area in yield modeling determines the expected number of faults per chip. The critical area constitutes the first component of defect limited yield modeling. The second component involves statistical distribution of the number of faults on the wafer. This component of defect limited yield modeling is addressed in the next section.

2.5 Defect Limited Yield Models

There are two defect limited yield models that are most intuitive and have been widely used to estimate yields that are limited by random defects. These models are based on Poisson and generalized negative binomial statistics.

Local deformations and any resulting faults can be assumed to be independently and randomly distributed on the wafer since the manufacturing process attempts to replicate each chip identically. When the probability of fault occurrence is independent and identical for all locations on the wafer, then the number of faults on a die, N, is given by a Poisson distribution with an average number of faults per chip of λ [1, 3].

$$P(N = n) = \frac{e^{-\lambda}\lambda^n}{n!}, \tag{2.6}$$

Since the defect limited yield, Y_r, is equal to the probability of having no faults on any given chip, Eq. 2.6 gives,

$$Yr = P(N = 0) = e^{-\lambda}. \tag{2.7}$$

λ is the sum of all λ^i, the number of faults per chip for individual layers of the IC since Y_r, the overall defect limited yield, is the product of all Y_r^i, the defect limited yield for individual layers. That is,

$$Y_r^i = e^{-D^i A_c^i}, \tag{2.8}$$

$$Y_r = \prod_i e^{-D^i A_c^i}, \tag{2.9}$$

$$\lambda = \sum_i \lambda^i. \tag{2.10}$$

The application of the Poisson model in early practices of yield modeling resulted in severe underprediction of yield [1, 2]. This underprediction owed itself partly to the use of die area instead of critical area since the concept of critical area is fairly recent in yield modeling and partly to the simplified assumptions made in deriving the Poisson model. The assumption that faults are independently and identically distributed over the wafers is a simplification of reality. Murphy [8] suggested improvement of the Poisson model by proposing a compound Poisson model given as follows:

$$Y_r^i = \int e^{-D^i A_c^i} g\left(D^i\right) dD^i. \tag{2.11}$$

$g(D^i)$, a probability density function for the defect density D^i, is called a compounder and accounts for wafer to wafer variations in D^i. Several functions including delta, uniform, and triangular functions [8] have been suggested as the compounder in attempting to match measured yield with predicted yield.

It has been shown [1] that a useful compounder comes from Erlang and gamma distributions. This results in a model that accounts for the fact that local random faults are not distributed uniformly over the wafer but show a tendency of forming small clusters. The tendency of local random faults to form small clusters is a physical phenomena and is termed "local clustering of faults". The model that is obtained by using Erlang and gamma distribution as the compounder is named a "clustered" defect limited yield model. The clustered random distribution of faults results in a number of failing chips smaller than the number of chips that fail due to a uniform random distribution of faults at the same level of defect density. The clustered defect limited yield model takes acount of this.

The local clustering of faults can be described mathematically as the fact that the probability of finding a fault in the vicinity of other faults is greater than the probability of finding an isolated fault. This makes physical sense since a perturbation that causes a fault at a particular point on the wafer is also likely to affect neighboring locations. Consider, for example, defects or faults caused by particles in the manufacturing environment. Particles present in gases, liquids, or the atmosphere in the manufacturing environment are known to form aggregates. When these particles reach the wafer surface, they will be clustered and hence any faults that arise due to the presence of these particles will also be clustered.

The fact that a fault is more likely to exist in the vicinity of another fault than exist independently of all other faults can be taken into account by assuming that the probability density function of fault formation, $f(n, t)$, where $f(n, t)\,dt$ is the probability that a fault will occur during time t to $t+dt$ on a chip which already has n faults, is independent of time t but is directly proportional to the number of faults n already present on the chip,

$$f(n, t) = c + an \tag{2.12}$$

It can be shown [1] that the following probability distribution for the number of faults per chip follows both directly from the above $f(n, t)$ and λ, the average number of faults per chip, or by using an Erlang and gamma distribution in Eq. 2.11.

$$P(N = n) = \frac{\Gamma(\alpha + n)}{n!\Gamma(\alpha)} \times \frac{(\lambda/\alpha)^n}{(1 + \lambda/\alpha)^{n+\alpha}} \tag{2.13}$$

This is a generalized negative binomial distribution. Since the expected yield is given by the probability of obtaining zero faults per chip, i. e. $n = 0$, the statistical yield expression obtained under the local clustering assumption is given by

$$Y_r = \left(1 + \frac{\lambda}{\alpha}\right)^{-\alpha} \tag{2.14}$$

where α is called the cluster parameter and is given by the ratio of c and a in Eq. 2.12.

Intuitively, α is a measure of the extent to which local defects cluster. It is always determined empirically, i. e. it is a curve fitting parameter [4]. The limit $\alpha \rightarrow \infty$ corresponds to no clustering or uniform and random distribution of faults. Therefore, in the limit $\alpha \rightarrow \infty$, the negative binomial model is identical to the Poisson model. Similarly, the limit $\alpha \rightarrow 0$ corresponds to highly clustered distribution of local faults.

The highest level of accuracy in yield model predictions is obtained when the yield is calculated as a product of yields for individual levels using Eq. 2.14, similar to Eqs. 2.8-2.10 for the Poisson model. Of course, the use of Eq. 2.14 requires correctly calculated critical area and measured defect density for individual levels. Both the Poisson and generalized negative binomial models are found to be equally fundamental to defect limited yield modeling and quite accurate if carefully used [1] [2].

The Poisson model is accurate and the generalized negative binomial model unnecessary if no or little local clustering exists for a particular fab. The only error that arises for the Poisson model (when correctly calculated critical areas and measured defect densities for individual levels are used) is underprediction, since the Poisson model is unable to account for any local clustering. The Poisson model is claimed to be simpler and more accurate for smaller chips and the generalized negative binomial model better predicts the yield for larger VLSI chips [4]. This claim appeals to reason because the effect of local clustering, i.e. underprediction by the Poisson model, will be smaller if chip areas are smaller or comparable to average local cluster size and will become more pronounced for larger chip areas. Hence, the Poisson model can be used for smaller chips, e. g. analog and mixed signal ICs, without sacrificing too much accuracy. Although the negative binomial model accounts for local clustering, it has an extra curve fit parameter, α, and therefore runs the risk of being less accurate statistically. However, when α is calculated carefully, the negative binomial model has been shown to have good results for chips of all sizes [1,

6].

2.6 Gross Yield Modeling

The deformation mechanisms described in section 2.4 belong to two broad categories, local and gross. The local deformations result in yield loss that is modeled by defect limited yield models. Yield loss that results due to gross deformations is termed gross yield loss and is modeled by a multiplicative factor called Y_o [1, 4, 12, 13].

Y_o is the fraction of wafer area that is occupied by non-zero yield regions, i. e. the regions that encountered only local deformation mechanisms. It is given as follows:

$$Y_o = 1 - \frac{S_o}{S}, \tag{2.15}$$

where S_o is the area occupied by all the zero yield regions, areas on wafer that encountered gross deformation mechanisms, and S is the total wafer area. It follows from the definition of Y_o that the fraction of the wafer occupied by zero yield regions is given by $(1\text{-}Y_o)$. If a wafer contains good and bad chips mixed in a random fashion over the entire wafer, i. e. contains no zero yield region, Y_o equals 1.

Y_o appropriately models gross yield [12, 13] because it gives the fraction of the wafer that is not prevented by the gross deformation mechanisms from having any good chips. The overall wafer yield, Y, is given by scaling the defect limited yield, Y_r, by the Y_o factor to account for the fact that only Y_o fraction of the wafer is available for defect limited yield.

$$Y = Y_o \times Y_r. \tag{2.16}$$

A desirable situation in semiconductor manufacturing is to eliminate all gross yield loss mechanisms from the fab or, equivalently, drive the value of Y_o towards one. The modeling of gross yield losses has often been overlooked in semiconductor yield model-

ing and emphasis has been on defect limited yield modeling since only defect limited component contributes to overall yield. Often the Y_o value in the yield models is simply set to a constant value, e. g. 0.9, to account for edge die loss and gross loss mechanisms that are commonly observed. However, such practices make yield model predictions unreliable. Along with other key inputs to yield models such as critical area, measured defect density and correct choice between Poisson and the generalized negative binomial models, the Y_o gross yield factor is a necessary element for correct yield modeling. Consider, for example, an erroneous and simplistic application of yield models in which the Y_o factor is neglected, or implicitly assumed to be 1, and die area is used instead of critical area. The use of die area instead of critical area will underpredict defect limited yield and neglect of the Y_o factor will overpredict yield. Although the two effects may somewhat cancel each other and measured yield may resemble the model yield, such practice of yield modeling is not physically based and provides little insight into deformation mechanisms and the interaction of the layout with various deformation mechanisms that result in yield loss.

Although the concept of Y_o is well understood, a difficulty in its application is lack of a method that calculates Y_o for a given wafer. The calculation of Y_o is equivalent to separating the yield of the wafer into gross and defect limited components. The work in this thesis addresses the problems of calculating Y_o and detecting whether a given wafer encountered any gross yield loss. The first problem is to examine the spatial distribution of faults on the wafer and determine whether the wafer encountered only local deformation mechanisms or both local and gross deformation mechanisms. If the wafer contains any continuous faulty regions, zero yield regions, then it is claimed to have been subjected to gross deformation mechanisms or gross yield loss. The second problem is to measure the extent or severity of gross yield loss by calculating the Y_o for the wafer.

It is proposed to investigate the spatial distribution of faults using the spatial distribu-

tion of the failing dice. Since each fault results in a failing die, the location of each failing die gives the area within which one or more faults occurred. Hence the failing dice can be considered to have the same approximate spatial distribution as faults on the wafer. This argument is further supported by the fact that the task here is not to distinguish between uniform and local clustered distributions. The objective is to recognize if a continuous region of faults or a zero yield region exists or not. By representing each functional die as a '1' and each nonfunctional die as a '0', the problem stated above simplifies to the characterization for random distribution of a two dimensional binary data.

There are obvious advantages in using the chip functionality data to investigate the distribution of faults on wafers. Actual production data can be used that is available with minimal acquisition effort. After completion of wafer level fabrication, each die on the wafer is tested for functionality and is marked with a red ink dot if found nonfunctional. The chip functionality data is also stored in a computer in two dimensional arrays as 1's, for functional chips, and 0's, for nonfunctional chips, at the respective locations. This data is very suitable for analysis of spatial distribution. Another important advantage is that this data represents the true effects of the total process on the functionality of chips outcome, i. e. makes no assumption about the sources of deformations that cause faults and accounts for deformations that occur at each level of fabrication [12]. That is, Y_o does not give level by level information.

The next chapter develops two statistical methods. The first method is named "a spatial randomness test." This method takes as input the two dimensional wafer map showing distribution of good and bad chips on the wafer and finds whether the wafer map contains one or more zero yield regions. Wafers are classified into two classes; class L for wafers that encountered only local deformation mechanisms and class G for wafers that encountered both gross and local deformation mechanisms. If a wafer map contains no zero yield

region, it is assigned to class L and if the wafer contains one or more zero yield regions, it is assigned to class G. The spatial randomness test also indicates the type of dominant dependence present in class G wafers. That is, if the regions of bad chips that constitute zero yield regions have a tendency of forming horizontal, vertical, right diagonal or left diagonal stripes, the test indicates the dominant direction. If zero yield regions exist without any clear directional dependency, the test indicates accordingly. If any wafers are determined to belong to class G, this is information about existence of gross yield loss or gross deformation mechanisms and is useful in determining direction for yield enhancement. If any dominant spatial dependencies are detected, it may lead to detection of root cause for gross yield loss.

The second method developed is named "a method to calculate Y_o." This method separates yield into its gross and defect limited components by statistically calculating Y_o for a given wafer. The value of measured Y_o is necessary to match measured yield data with yield predicted by defect limited yield models. It also provides a measure of the severity of gross yield loss encountered by the wafer.

It may appear that the first method, the spatial randomness test, is somewhat redundant since whenever Y_o is found to be close to one it follows that no zero yield region exists on the wafer and the wafer belongs to class L. However, the spatial randomness test is necessary because it is able to account for the fact that bad chips in non-zero yield regions may not be distributed uniformly but may exhibit a tendency of forming local clusters. The method to calculate Y_o does not allow for small local clustering and hence may provide slightly misleading results if applied independently. Moreover, the method to calculate Y_o works in conjunction with the spatial randomness test because it uses knowledge from the outcome of the spatial randomness test about any directional dependence present in zero yield regions.

Chapter 3

A Spatial Randomness Test and a Cluster Recognition Method for Semiconductor Wafers

3.1 Introduction

The distribution of good and bad chips on a semiconductor wafer typically results in two types of regions, one that contains both good and bad chips distributed in a random fashion and the other that contains almost all bad chips. The presence, size, shape, and location of regions that contain almost all bad chips constitute information useful for prediction, control, and improvement of semiconductor yield.

The coordinates on a two dimensional rectangular grid are used to represent the locations of individual chips on the silicon wafer. We refer to this grid or wafer map as a two dimensional lattice or simply lattice. A binary (0, 1) random variable x is associated with each point s on a lattice S: $x_s = 1$ if the chip at $s \in S$ is good, and $x_s = 0$ if the chip at $s \in S$ is bad. A non-zero yield region, denoted by $NZ \subseteq S$, is defined as a region on the lattice S that contains both 1's and 0's mixed in a random fashion. For simplicity, we model each site in NZ as having an independent and identical probability P for the site to take the value 1. Thus each site in NZ is represented by an independent and identically distributed (i.i.d.) Bernoulli random variable with P as its parameter. A zero yield region, denoted by $Z \subseteq S$, is defined as a region on lattice S that contains all 0's or a negligible number of 1's, e.g. only none to five 1's in a region of 100 sites. Intuitively, a zero yield region is an aggregate of enough 0's that is unlikely to arise in a non-zero yield region as a random event. Although other types of regions such as regions with all 1's or regions which contain both 1's and 0's mixed in a regular or nonrandom fashion can occur on an arbitrary

lattice, these regions are of no interest to us and hence we do not define these.

The challenge in exploration of zero yield regions on a given lattice arises from the fact that no assumptions are made about the zero and non-zero yield regions. No knowledge is assumed about the presence or number, location, size and shape of the zero yield regions and the relative number of 1's and 0's in the non-zero yield regions. Visual inspection of real wafer maps has shown that the presence, location, size, and shape of the zero yield regions and yield level in the non-zero yield regions all vary from wafer to wafer even for the same lot. Hence each wafer has to be examined individually for zero yield regions without assuming any information from other wafers. Therefore, the detection of zero yield regions is typical of clustering problems encountered in the field of pattern recognition and machine intelligence.

We develop two statistical methods in this chapter. The first method, named "a spatial randomness test", detects whether or not a region of almost all bad chips is present on a given wafer. The second method, named "a cluster recognition method" finds those bad chips on a given wafer that belong to regions of almost all bad chips.

We first expound on the problem of detecting zero yield regions on a given lattice, i. e. the spatial randomness test. A solution to this problem based on Gibbs/Markov Random Fields (G/MRF) [21, 22, 23] is provided in the following section. Although the spatial randomness test based on G/MRF is a sound theoretical solution, it is difficult to implement in practice because of computational complexity. In the subsequent section, we present a spatial randomness test based on the aura framework as a solution that has both good theoretical basis and is straightforward to implement. The aura framework [24, 25, 26, 27] is a recently developed statistical image tool that uses set theory. The spatial randomness test based on G/MRF is provided for its instructive value since the aura framework has been shown to be a reformulation of the G/MRF [26, 28]. The section that discusses a spatial

randomness test based on G/MRF can be omitted without significant loss in continuity. The implementation and performance evaluation of the aura framework based test is discussed next. The subsequent sections similarly provide the problem statement of the cluster recognition problem, a solution to this problem, and finally implementation and performance evaluation of the cluster recognition method.

3.2 The Spatial Randomness Problem Statement

The first component of the problem addressed in this work is to develop an unambiguous method to determine if the failing chips are spatially correlated (large areas of failing chips) or uncorrelated (isolated failing chips). Representing each good chip by a '1' and each bad chip by a '0', this problem can be stated as investigation of a given two dimensional binary lattice of any size to find if 1's and 0's on the lattice are distributed in a random or noisy fashion or contain a gradient, structure, or trend in the distribution. The question here is to characterize how the site values on the lattice relate (or covary) locally with other site values and not to probabilistically characterize the site values generally. The need for such a "spatial randomness test" arises in several technological, biological, geological, and geographical applications [15].

The spatial randomness test checks for the null hypothesis of randomness that each location on the binary lattice can take one of its two possible values independent of the rest of the lattice and is represented by an independent and identical Bernoulli trial. The alternative hypothesis is true if there are some locations on the lattice whose values depend on the values of other sites on the lattice. This corresponds to clusters of 1's or 0's, i. e. a tendency of like elements to attract or aggregate, and regular patterns, i. e. a tendency of like elements to repel or disperse. However, it should be noted that the acceptance of the alternative hypothesis on our lattices of interest will imply the presence of only clusters of 0's and not clusters of 1's nor dispersion between 1's and 0's. This is

because no clustering of only good chips and no regular dispersion between good and bad chips are usually found on semiconductor wafers.

The problem of testing for spatial randomness or dependencies is recognized as a difficult problem in mathematical statistics [15, 16]. In [15] the problem of detecting spatial randomness is addressed to find homogeneity in materials. It proposes three tests based on cluster test statistics, nonparametric analysis of variance of different regions on the lattice, and a graph theoretic model. In [16] several solutions to this same problem are suggested. We discuss a spatial randomness test based on Gibbs/Markov Random Fields, one which faces computational constraints, and a spatial randomness test based on the aura framework, one which is very modest in its computational requirements. Both these solutions are found to possess stronger mathematical basis and are more intuitive than the methods provided in [15, 16]. The notation, definitions, and proofs for the aura framework and the G/MRF employed here are taken from [23, 24, 25, 26, 27].

3.3 A Spatial Randomness Test Based on G/MRF

A well known and theoretically powerful statistical model that accounts for spatial dependencies is the Gibbs/Markov Random Field model. [16] suggests the use of this model as a good theoretical solution to the problem of finding spatial dependencies in a given data but rejects it as an impractical idea due to computational limitations.

3.3.1 The Gibbs/Markov Random Field Model

The famous Ising model of statistical mechanics that characterizes the mixing of opposite spin particles in materials is a special case of the binary G/MRF model. From a historical perspective, the MRF has simultaneous roots in the Gibbs distribution of statistical mechanics and Markov models of probability [25]. Most of the development of G/MRF in the field of statistics and probability is due to Besag [21, 22].

The MRF model is a probability model for distribution of values on a lattice and takes

into account the effect of the lattice values on each other. It should be noted that the G/MRF is a model for spatial interaction only and is translation invariant. It does not capture variations by location on the lattice. Investigating variations by location is better defined and easier to address than finding dependence of site values on each other.

The G/MRF model captures spatial interaction among values at different locations through the probability notions of Markov dependence. The idea of Markov dependence is illustrated best by first examining dependence in a one dimensional Markov chain. Consider a stationary Markov chain [18] that is constituted by a system that assumes different states at discrete values of time. The state of the system at time n is represented by S_n, where S_n is assumed to be translation invariant. If the distribution, f, of S_n depends only upon the value of S_{n-1}, the system is said to constitute a Markov chain. That is, S_n interacts with rest of the chain through S_{n-1} only.

$$f(S_n|S_{n-1}, S_{n-2}, S_{n-3}, \ldots) = f(S_n|S_{n-1}) . \tag{3.1}$$

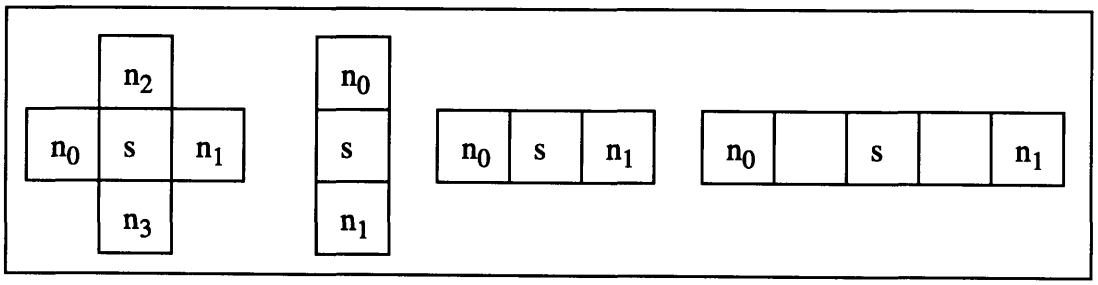


Figure 3.1: Examples of neighborhood structures.

For a two dimensional finite rectangular MxN lattice S, let the value at location s be represented by a random variable x_s. Although these ideas are equally valid for any possible number of values for x_s, we restrict ourselves to a binary lattice and hence x_s can take only one of the two possible values, 0 or 1. Let X be the vector $(x_s, 1 \le s \le |S|)$ of site values and Ω be the set of all configurations taken by X. Also a neighborhood structure associated with S is defined as $N = \{N_s, s \in S\}$ where $N_s \subseteq S$ is the set of sites which are

neighbors of the site $s \in S$. A neighborhood structure is said to be symmetric if $\forall s, r \in S, s \in N_r$ if and only if $r \in N_s$. A few examples of different symmetric neighborhood structures are shown in Figure 3.1.

The dependence in a Markov chain is extended to more than one dimension and made noncausal in G/MRF. The probability distribution for X, the lattice values, is said to be given by a MRF if the conditional probability distribution of x_s depends only upon a defined neighborhood and is independent of rest of the lattice.

$$f(x_s|X\backslash x_s) = f(x_s|N_s) . \tag{3.2}$$

That is, each location s interacts with the rest of the lattice through its neighbors only.

Besag showed in [21] that there is only one conditional distribution for a binary MRF that is self-consistent. This is shown by the equation below:

$P(x_s \mid \text{rest of lattice}) = P(x_s|N_s)$

$$= \frac{exp\left(x_s\left(\alpha_s + \sum_{r \in N_s} \beta_{sr} x_r\right)\right)}{1 + exp\left(\alpha_s + \sum_{r \in N_s} \beta_{sr} x_r\right)}, \tag{3.3}$$

where x_s and x_r can only take values 0 and 1. The above equation leads to a joint probability distribution for the entire lattice S which is same as the Gibbs distribution. The Gibbs distribution that defines a random field is given by

$$P(X) = \frac{1}{Z} exp\left(\sum_{s \in S}\left(x_s\left(\alpha_s + \sum_{r \in N_s} \beta_{sr} x_r\right)\right)\right), \tag{3.4}$$

where

$$Z = \sum_{X} exp\left(\sum_{s \in S}\left(x_s\left(\alpha_s + \sum_{r \in N_s} \beta_{sr} x_r\right)\right)\right). \tag{3.5}$$

Z is a normalizing constant known in statistical mechanics as the partition function. For the lattice considered, Z is a summation of 2^{MN} terms, one term for each of the 2^{MN} pos-

sible realizations of the binary random vector *X*. The terms in *P(X)* which do not contain any neighborhood values are called single site potentials. The terms that contain both x_s and its neighbors are called two-site potentials. The single-site potentials are responsible for the noisy component of the distribution and the two-site potentials are responsible for the presence of structure between site values in the lattice.

The parameters β_{sr} are called bonding parameters and measure the extent of dependence between lattice values. We name α_s, the coefficient of single-site potential, as a noise parameter. If the lattice is assumed homogeneous, i.e. does not vary with position, $\alpha_s = \alpha$ (site-independent), and the bonding parameters $\beta_{sr} = \beta_r$ (site-independent but possibly dependent on the directional position of the neighbor). If the strength of spatial interaction between site values is equal in all directions, the field is called isotropic and the bonding parameters are equal for all neighbors $\beta_r = \beta$, $\forall r \in N_s$.

3.3.2 The G/MRF Based Test

A good test to find whether or not the distribution of site values on a given lattice is random is to fit the two dimensional data to the homogeneous G/MRF model. If all the bonding parameters are statistically determined to be zero, the site values are found to be independently and identically distributed on the lattice. In this case, the above G/MRF model implies,

$$P(x_s | \text{rest of lattice}) = \frac{exp(\alpha x_s)}{1 + exp(\alpha)},$$

that is, we have a simple Bernoulli model with Bernoulli parameter

$$p = \frac{exp(\alpha)}{1 + exp(\alpha)}.$$

The joint M/GRF distribution for this case is given by

$$P(X) = \frac{1}{Z} exp\left(\alpha \sum_{s \in S} x_s\right)$$

$$= \frac{1}{Z}\left(\frac{exp(\alpha)}{1+exp(\alpha)}\right)^{\sum_{s\in S} x_s}\left(\frac{1}{1+exp(\alpha)}\right)^{-\sum_{s\in S} x_s}$$

$$= \frac{1}{Z}p^{\sum_{s\in S} x_s}(1-p)^{-\sum_{s\in S} x_s},$$

where

$$\begin{aligned} Z &= \sum_{X} exp\left(\alpha \sum_{s\in S} x_s\right) \\ &= (1+exp(\alpha))^{MN} \\ &= \left(\frac{1}{1-p}\right)^{MN}. \end{aligned}$$

Therefore, if all β_r are zero, *P(X)* reduces to the joint probability mass function for MN i.i.d. Bernoulli variables. However, if the parameter estimates for the bonding parameters are significantly different from zero then the data is not uniformly random and exhibits its spatial dependence. If $\beta_r > 0$ the data shows clustering or aggregation of like elements in the direction of neighbor *r*. If $\beta_r < 0$ like elements repel each other and the data shows dispersion in the direction of neighhbor *r*.

The partition function is essentially uncomputable because it contains a large number of terms. Optimal or Maximum Likelihood Estimation of G/MRF is, therefore, not possible for other than very small lattices. Simplifying assumptions can be made leading to suboptimal parameter estimation, e. g. Maximum Pseudo-Likelihood Estimation [23]. However, we treat optimal parameter estimation of the G/MRF model as a conceptual randomness test only and relate it to our next described method based on the aura framework.

3.4 A Spatial Randomness Test Based on the Aura Framework

The aura framework makes use of the neighborhood dependency employed in G/MRF. The aura framework has been shown to be an equivalent linear reformulation of G/MRF. The use of aura framework to solve the spatial randomness problem involves calculation

of statistics that pose no computational constraints. The aura statistics have similar interpretation and validity as optimally estimated parameters of the G/MRF.

3.4.1 The Aura Framework

The aura framework is a set-theoretic image analysis tool. Aura can be intuitively described as "the presence of set B in the neighborhood of set A" [24]. The aura can be formulated using simple set theory or using dilations from mathematical morphology [24, 25]. The aura framework can be related to different texture analysis tools, such as the co-occurrence statistics [28] and the G/MRF model. We define the aura framework in terms of set theory only and relate it to G/MRF only. The aura formulation will lead us to interpret the distribution of 1's and 0's on a given binary lattice as physical notions of separation/mixing and minimum/maximum boundry lengths between 0's and 1's or the sets A and B.

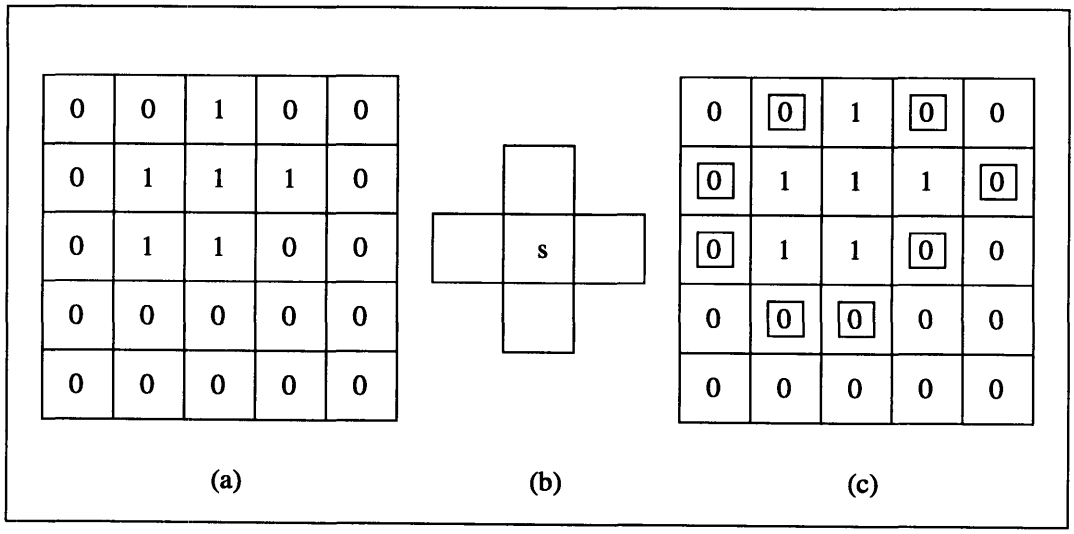


Figure 3.2: Example of an aura on a binary lattice. (a) A binary lattice. (b) The four nearest neighborhood structure. (c) The aura of the set of all 1's with respect to the set of all 0's based on four nearest neighbors is shown by the set of double-boxed elements.

Definition: Let A and B be two subsets $\subseteq S$. Then the aura of A with respect to B for the neighborhood structure $\{N_s, s \in S\}$ is a subset of S defined by

$$\bigcup_{s \in A} (N_s \cap B) . \tag{3.6}$$

The aura of a set A with respect to a set B is denoted by $O_B(A, N)$.

The aura of a set with respect to itself is called the self-aura.

It is important to note that all the results and interpretations from auras are neighborhood dependent and are valid for any arbitrarily defined neighborhood structure, N. Several properties of auras for a given neighborhood can be derived from set theory. These and the mathematical morphology formulation of aura are omitted here and the interested reader is referred to [24, 25].

The number of elements in the aura of a set A with respect to a set B, $\left|O_B(A, N)\right|$, provides a convenient measure of the size of the aura of set A with respect to set B. We note from the definition of aura and set properties that,

$$\left|O_B(A, N)\right| \leq \sum_{s \in A} \left|N_s \cap B\right| . \tag{3.7}$$

The right hand side in the above expression provides a measure of discontinuity in the boundary of set A and enables the treatment of the aura framework as a linear reformulation of the G/MRF. The aura measure, denoted by $m(A, B)$, is defined by

$$m(A, B) = \sum_{s \in A} \left|N_s \cap B\right| . \tag{3.8}$$

The usefulness of the aura measure can be illustrated by considering its physical interpretations. For the four nearest neighborhood structure, shown in Figure 3.2(b), the aura measure of two sets A and B becomes the boundary length (computed in the number of sites) between A and B. When like elements on the lattice attract $m(A, A)$ and $m(B, B)$ are large and m(A, B) and m(B, A) are small, corresponding to the minimization of boundry length between the sets A and B. The boundry length is maximized and $m(A, B)$ and $m(B, A)$ are large and $m(A, A)$ and $m(B, B)$ are small when unlike elements attract

each other. Also, we can represent the miscibility between two fluids A and B by $m(A, B)$. If A and B are miscible $m(A, B)$ is maximized, and if they are immiscible $m(A, B)$ is minimized.

Since most lattices of interest are finite in size, the aura set and aura measures for a given finite lattice can be calculated by considering the lattice to be periodic about its boundaries or by reflecting the lattice about its edges. Several aura measure properties follow from its definition and set theory. Two of these properties are relevant to developing the spatial randomness test and are stated below.

Property 1: If $\{N_s, s \in S\}$ is a symmetric neighborhood structure, then

$$m(A, B) = m(B, A), \forall A, B \subseteq S. \tag{3.9}$$

The definitions for neighborhood structure and symmetric neighborhoods are the same as defined for G/MRF. Note that the above property is true only for symmetric neighborhoods and not true in general. The proof of this property, which is rather intuitive, along with other properties of the aura measure is given in [25].

Property 2: If $A \cup B = S$ and $A \cap B = 0$ then the following are true:

$$m(A, A) + m(A, B) = |A||N|, \tag{3.10}$$

$$m(B, A) + m(B, B) = |B||N|. \tag{3.11}$$

We provide a proof for property 2. Only the first equality will be proved. The second follows in a similar way. From the definition,

$$\begin{aligned} m(A, A) + m(A, B) &= \sum_{s \in A} |N_s \cap A| + \sum_{s \in A} |N_s \cap B| \\ &= \sum_{s \in A} |N_s \cap (A \cup B)| \text{ (because } A \text{ and } B \text{ are disjoint)} \\ &= \sum_{s \in A} |N_s \cap S| \\ &= \sum_{s \in A} |N_s| \\ &= |A||N|. \end{aligned}$$

If N represents a symmetric neighborhood then it follows from the above two properties that

$$m(A,A) + m(B,B) + 2m(A,B) = |S||N|. \tag{3.12}$$

This equality shows that as the self measures increase, the cross measures decrease and vice versa. This makes intuitive sense. As the elements of set A aggregate, they separate from elements of set B, causing elements of set B to aggregate as well. This will increase the self measures and decrease the cross measures. The opposite follows for the case when like elements separate from each other. We are considering connected lattices only and hence geometrical considerations will impose some lower bound on $m(A,B)$ and upper bound on $m(A,A)$ and $m(B,B)$ if both A and B are present in the lattice.

The aura measures can be used to define a tool, called the aura matrix, important for image processing applications. The aura matrix is a representation of the aura measures for all sets of interest on the lattice and forms a partition of the set S.

Definition: Let $S_i \subseteq S$, $i = 0, 1, ..., n-1$, $\bigcup_{i=0}^{n-1} S_i = S$ and $S_i \cap S_j = \varnothing$ unless $i = j$. Then the aura matrix, A_M, is the n x n integer matrix defined by $A_M = [a_{ij}]$ where $a_{ij} = m(S_i, S_j)$, $0 \le i, j \le n-1$. For binary lattices, the aura matrix A_M is given as

$$A_M = \begin{bmatrix} m(A,A) & m(A,B) \\ m(B,A) & m(B,B) \end{bmatrix}. \tag{3.13}$$

The intuition provided above for the behavior of aura measures at various degrees of "clumpiness" and "dispersion" between sets A and B suggests that the aura matrix becomes diagonal for an attractive field between like elements when suitable constraints exist on the histogram, neighborhood, and lattice size. The aura matrix will become antidiagonal when the field is repulsive instead of attractive. Also, if the lattice follows a uniform histogram constraint, $|A| = |B|$, all the entries of the aura matrix become constant when distribution of the elements of A and B on the lattice is noisy or random with respect

to the neighborhood structure used. We recall that the bonding parameters of the G/MRF are positive when like elements aggregate or the field is attractive, negative when the field is repulsive, and zero when the field is noisy. Hence, we are led to believe that the aura matrix can be used to provide similar information about the spatial dependencies in the lattice as the bonding parameters of the G/MRF model.

In G/MRF model the varying spatial dependencies in different directions are found by assuming an anisotropic field for the lattice and using different bonding parameters for different directions. We deal with anisotropy in the aura framework by using the notion of subneighborhood and finding aura matrices for subneighborhoods in particular direction.

For a given neighborhood structure divided in K subneighborhoods, $N_s^k \subseteq N_s$, the following result from set theory, aura and aura measure definitions.

$$N_s = \bigcup_{k=1}^{K} N_s^k, \forall s \in S, \tag{3.14}$$

$$O_B^k\left(A, N_s^k\right) = \bigcup_{s \in A}\left(B \cap N_s^k\right), \tag{3.15}$$

$$O_B(A, N_s) = \bigcup_{k=1}^{K} O_B^k(A), \tag{3.16}$$

$$m^k(A, B) = \sum_{s \in A}\left|N_s^k \cap B\right|, \tag{3.17}$$

$$m(A, B) = \sum_{k=1}^{K} m^k(A, B). \tag{3.18}$$

Hence, an aura matrix with respect to a given neighborhood will be the sum of the aura matrices with respect to the subneighborhoods. For example, if the dependencies in the lattice and hence the bonding parameters of the G/MRF are different for horizontal and vertical directions, then the aura matrix for the four nearest neighborhood may not give any meaningful information about spatial bonding since different dependencies in the two

directions may obscure each other's effect. However, when the subneighborhood aura matrices are examined individually the existence and extent of any biases in the vertical and the horizontal directions can be found. If the field is isotropic, the aura matrices for the overall neighborhood and the subneighborhoods will indicate the same spatial bias.

3.4.2 The Aura Framework Based Test

In the preceeding section we discussed intuition of how aura matrices behave differently for different or lack of spatial dependencies. However, a quantitative measure of capturing this difference in aura matrix behavior for various spatial dependencies has not yet been given. Such measures exist and effectively constitute the spatial randomness test sought. Also, these measures do not require a uniform histogram constraint for the lattice. We suggest the spatial randomness test as follows:

Given a two dimensional binary lattice, find the aura matrices for a symmetric neighborhood and its isotropic subneighborhoods. The aura matrix for a particular neighborhood is a feature or statistic for the given binary lattice that contains information with respect to the neighborhood structure on spatial dependencies in the lattice. The spatial bias information contained in the aura matrix can be quantitatively extracted by defining a normalization of the aura matrix to calculate a correlation coefficient between site values. The correlation coefficient, derived below, constitutes the statistics of interest and provides a measure of the extent and type of spatial dependence in the distribution of site values on the lattice.

The normalized aura matrix which is referred to as the Miscibility matrix [27], M, has a probabilistic interpretation and is defined by

$$\mathrm{M} = \begin{bmatrix} \dfrac{m(A,A)}{|A||N|} & \dfrac{m(A,B)}{|A||N|} \\ \dfrac{m(B,A)}{|B||N|} & \dfrac{m(B,B)}{|B||N|} \end{bmatrix}, \tag{3.19}$$

or

$$\mathrm{M} = \begin{bmatrix} p_{AA} & p_{AB} \\ p_{BA} & p_{BB} \end{bmatrix}.$$

Referring to property 2 of aura measures defined earlier, we notice that the rows of the miscibility matrix M sum to 1. Hence, M is a stochastic matrix. The elements of M, called the miscibility measures, $p_{AB} = m(A, B)/|A||N|$, have a probabilistic interpretation similar to the elements of the transition matrix of a Markov chain. The coefficient p_{AB} can be interpreted as the conditional probability that a given site of value A has in its neighborhood, any allowed neighbor, a site of value B. The probabilistic interpretation enables us to write the correlation, $R(\nu)$, covariance, $C(\nu)$, and correlation coefficient, $\rho(\nu)$, of the lattice site values where a symmetric neighborhood structure N that has ν members is assumed. For a binary lattice in which each site can assume one of the two possible values a or b

$$R(\nu) = aap_{AA}p_A + abp_{AB}p_A + bap_{BA}p_B + bbp_{BB}p_B \tag{3.20}$$

$$C(\nu) = (a-m)(a-m)p_{AA}p_A + (a-m)(b-m)p_{AB}p_A + (b-m)(a-m)p_{BA}p_B + (b-m)(b-m)p_{BB}p_B \tag{3.21}$$

$$\rho(\nu) = \frac{C(\nu)}{\sigma^2}, \tag{3.22}$$

where

$p_A = \dfrac{|A|}{|A| + |B|}$,

$p_B = \dfrac{|B|}{|A| + |B|}$,

$m = ap_A + bp_B$, and

$\sigma^2 = (a-m)^2 p_A + (b-m)^2 p_B$.

The magnitude and sign of the correlation coefficient are information on the extent and type of spatial dependence in the distribution of site values on the lattice. If the sites values are independently and identically distributed over the lattice the correlation coefficient

will be close to 0. It will be greater than zero if like values tend to occur together and less than zero if like values separate from each other and attract the elements of the other set. Also, there can be a case that the correlation coefficient with respect to a particular subneighborhood indicates a spatial dependency, attractive or repulsive, and correlation coefficients with respect to all other subneighborhoods indicate no spatial dependencies. In this case, the spatial dependence in the lattice is in the direction of the subneighborhood that indicated the spatial dependency. If the spatial dependency present is in no particular direction then the correlation coefficients with respect to more than one subneighborhood as well as with respect to the main neighborhood structure will indicate this spatial dependency.

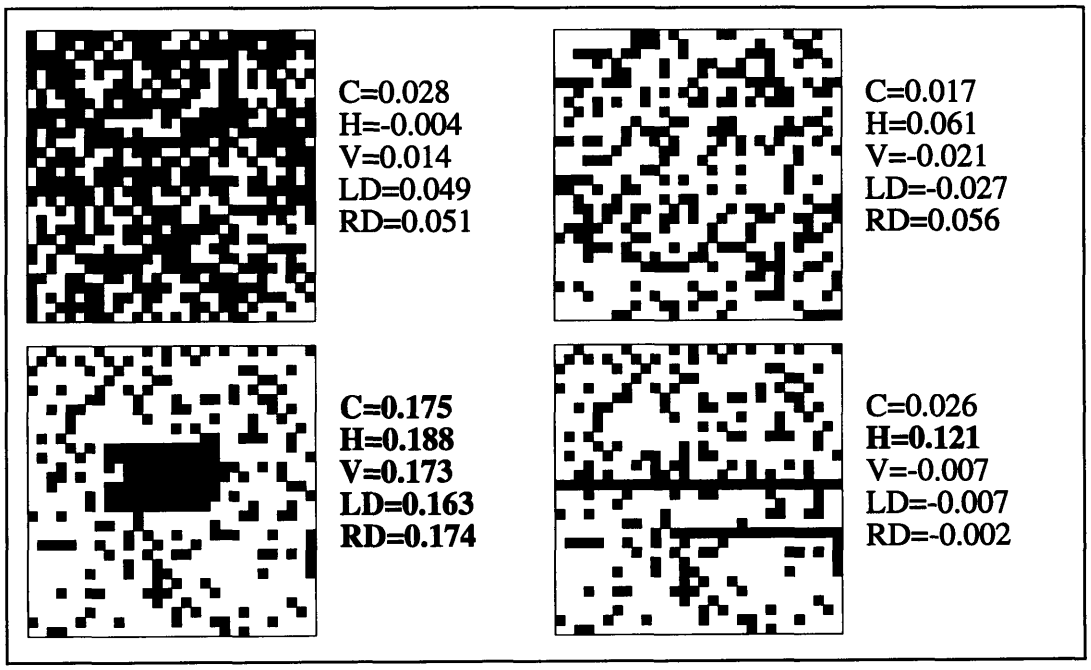


Figure 3.3: Examples of binary lattices with aura based correlation coefficients.

Figure 3.3 shows correlation coefficients for lattices with random distributions and with clusters of 0's added. In these lattices 1's are represented by white pixels and 0's are represented by black pixels. Also C, H, V, LD, and RD represent correlation coefficients

calculated using neighborhoods that are 8-nearest, 2-nearest in the horizontal direction, 2-nearest in the vertical direction, 2-nearest in the left diagonal direction, and 2-nearest in the right diagonal direction respectively. Any correlation coefficients that differ from zero by an amount significant enough to indicate a spatial dependency, see section 3.4.3, are shown in bold prints.

It can be shown that the correlation coefficient is related to the trace of the miscibility matrix, the normalized aura matrix, in a very simple manner.

$$tr[M(\nu)] = \rho(\nu) + 1. \tag{3.23}$$

We show this below only for the case when the two possible site values are represented by 1 and 0.

Let $p_A = P$ and $p_B = 1 - P$,

$$\Longrightarrow m = P,\ \sigma^2 = P(1-P),$$

$$\begin{aligned} C(\nu) &= (1-P)^2 p_{AA} P + (1-P)(-P) p_{AB} P + (-P)(1-P) p_{BA} (1-P) \\ &\quad + (-P)^2 p_{BB} (1-P) \\ &= P(1-P)\left[(1-P) p_{AA} + P p_{BB} - P p_{AB} - (1-P) p_{BA}\right] \\ &= P(1-P)\left[p_{AA} + p_{BB} - P p_{AA} - (1-P) p_{BB} - P p_{AB} - (1-P) p_{BA}\right] \\ &= P(1-P)\left[p_{AA} + p_{BB} - P(p_{AA} + p_{AB}) - (1-P)(p_{BA} + p_{BB})\right] \\ &= P(1-P)\left[p_{AA} + p_{BB} - P - 1 + P\right] \\ &= P(1-P)\left(tr[M(\nu)] - 1\right) \end{aligned}$$

$$\Longrightarrow tr[M(\nu)] = \frac{C(\nu)}{P(1-P)} + 1$$

$$\Longrightarrow tr[M(\nu)] = \rho(\nu) + 1$$

The trace is centered at 1 and can vary between 0 and 2. This follows both from the stochastic nature of the Miscibility matrix and the fact that the correlation coefficient is cen-

tered at 0 and varies between -1 and 1. Having shown the relationship between the correlation coefficient and the trace of the Miscibility matrix, we use the trace of the Miscibility matrix instead of the correlation coefficient as the statistic of interest because the trace is much simpler to compute.

A finite lattice with random distribution of 1's and 0's will result in a trace value close to 1. We find limits on trace of the normalized aura matrix to constitute a band of values centered at 1 called the randomness band. The trace values for lattices with random distributions will fall in the randomness band. A trace value smaller than the lower bound on the randomness band will indicate dispersion in the lattice and a trace value greater than the upper bound on the randomness band will indicate clustering in the lattice. Also, the randomness band becomes smaller as the the lattice size increases. The issues involved in the choice of the randomness band for a particular size lattice are treated in the next section using concepts from the field of pattern recognition and machine intelligence.

3.4.3 Training and Testing the Spatial Randomness Test

The choice of the randomness band and performance evaluation of the spatial randomness test can be facilitated by treating it as a pattern recognition system. A typical pattern recognition system seeks to classify a given image or lattice to one of a number of known classes. There are two basic components to a pattern recognition system. These components, shown in Figure 3.4, are called feature extraction and classification [19].

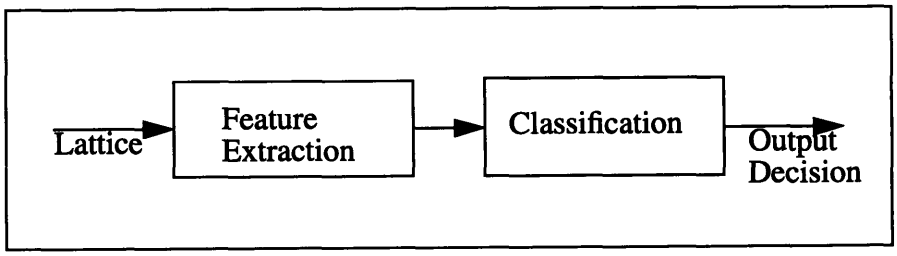


Figure 3.4: The main components of a pattern recognition system.

The feature extraction procedure is a transformation that calculates statistics using the lattice data such that information relevant to classify the lattice to its correct class is retained. The feature extraction algorithm is designed based on characteristics of the problem addressed or on intuition while ensuring that the information in the lattice distinct for different classes is preserved. The statistics with relevant information are used in a decision making scheme that classifies the lattice into one of a number of known classes. This contitutes the second component of the pattern recognition system.

The classification algorithm usually involves training or determination of some parameters that determine performance of the algorithm. The performance evaluation for the chosen parameters is called the testing procedure. The training and testing procedures start with a set of lattices of known classes. The labeled lattices are divided into two portions. The first portion is called a training data set whose features are used to determine the parameters of the classification algorithm. The other portion, called a test data set, is used to evaluate the performance of the classifier and perhaps change its parameters to improve the performance.

The trace values of the normalized aura matrix for a defined neighborhood structure and its subneighborhoods constitute components of a feature vector that contains relevant information on spatial dependencies contained in the lattice. Thresholds can be found for the trace values that act as a decision boundary for classification of the lattice into one of the three classes, dispersed, random, or clustered with respect to the particular neighborhood structure used.

The 8-nearest, first order nearest, neighborhood structure and its subneighborhoods in each direction, horizontal, vertical, right diagonal and left diagonal, form a set of neighborhoods that can suffice for use in the spatial randomness test. However, we also use a second set of neighborhoods formed by a 16-nonnearest, first order nonnearest, neighbor-

hood structure and its subneighborhoods in each direction on the training and test data sets. The need for the nonnearest neighborhood structures will become clear when we apply the spatial randomness test to actual wafer map data in the next chapter. Both the first order nearest and nonnearest neighborhood structures and their subneighborhoods are shown in Figure 3.5.

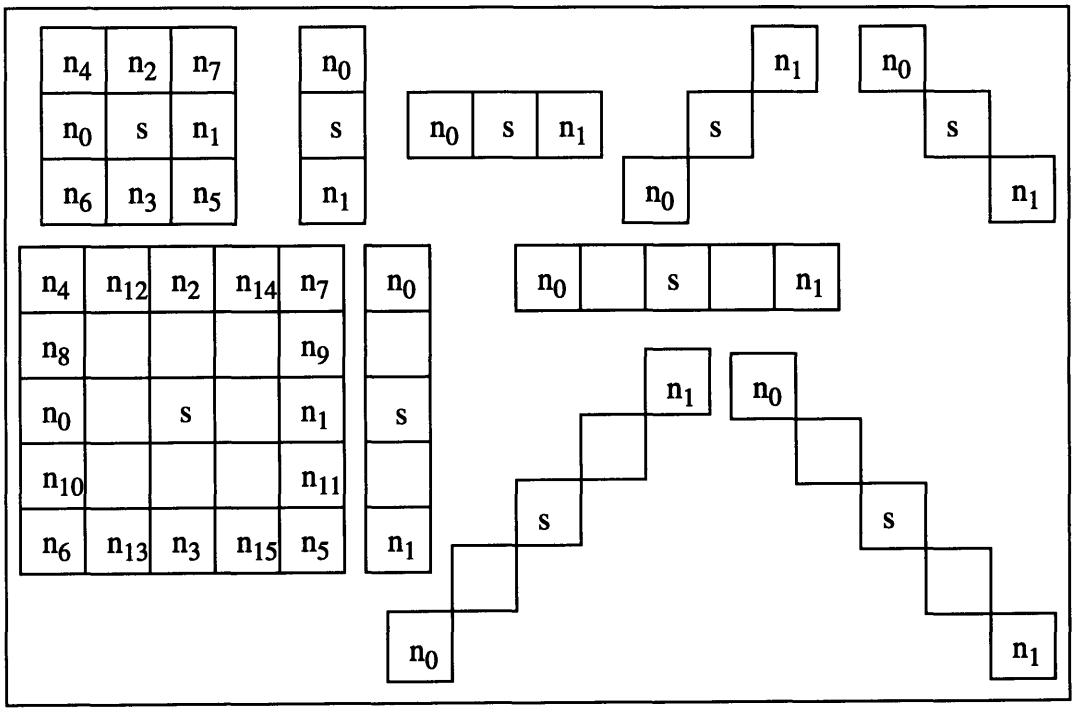


Figure 3.5: First order nearest and nonnearest neighborhoods and subneighborhoods.

The required training data set to determine the randomness band is any reasonably large number of simulated binary lattices with pseudorandom distributions (computer simulation cannot result in purely random distributions) of 1's and 0's on the lattice. The size and shape of these lattices should be the same as the lattices that need to be classified. To demonstrate the working of the spatial randomness test, we use 30x30 square lattices. For this example, the training data set consists of 90 30x30 simulated pseudorandom lattices. These lattices were generated by using an independent and identical Bernoulli trial for each location on the lattice, i. e. the value of the Bernoulli parameter P for each trial was

kept constant for a given lattice. To generate a representative training data set, the value of P was varied from 0.1 to 0.9 in steps of 0.1 for different lattices. Ten wafers were generated for each value of P, resulting in a total of 90 lattices.

The traces of the normalized aura matrix for each of these 90 lattices were found to lie between 0.96 and 1.04 for the 8-nearest and the 16-nonnearest neighbourhood structures. The trace values were found to lie between approximately 0.9 and 1.1 for the 2-nearest and the 2-nonnearest subneighborhoods in various directions. The decision rule for the randomness test was formed by using the minimum and maximum of the trace values of the 90 lattices as the lower and upper bounds, respectively, on the randomness band for each neighborhood structure.

The second data set consists of 90 other simulated pseudorandom binary lattices and 20 simulated pseudorandom distributions with patterns or spatial biases of different types, big clusters, horizontal stripes, vertical stripes, and diagonal, of 0's added. Only one out of the 90 simulated pseudorandom lattices was misclassified to nonrandom category and one cluster added lattice containing a vertical stripe of 0's was misclassified as random while all the other lattices in the test data set were classified correctly to their respective classes. Appendix A shows some member lattices of the test data set, their trace values and the spatial dependency that was attributed to them by the spatial randomness test.

3.5 The Y_o Problem Statement

The second component of the problem addressed in this work seeks to calculate the fractions of a given wafer that are occupied by zero and non-zero yield regions. The fraction of a wafer occupied by non-zero yield regions is denoted by Y_o.

Let S = total number of chips on the wafer, and

S_o = total number of chips in zero yield regions.

$$Y_o = 1 - \frac{S_o}{S} \tag{3.24}$$

The problem of calculating Y_o for a given binary lattice can be solved by identifying 0's that belong to zero yield regions or clusters of 0's. The identification of clusters of 0's constitutes an unsupervised clustering problem since no assumptions are made about the presence, number, location, size, and shape of these clusters of 0's on the lattice. The recognition of clusters of 0's can be facilitated by assuming a lattice X which consists of regions of only 1's and regions of only 0's with some unknown boundaries between regions of 1's and regions of 0's. Also consider a lattice N that consists of a binary noise process such that the noise value at a given site on the lattice is 1 with probability P and 0 with probability $1\text{-}P$, i. e. each location of the lattice N is given by an i. i. d. Bernoulli trial. A degraded image or noisy lattice Z can be formed by multiplying the lattice X with the noise process N.

$$Z = X \times N \tag{3.25}$$

The lattice Z represents distribution of good and bad chips on a wafer with each good chip represented by a '1' and each bad chip represented by a '0'. The lattice X with regions of 1's and 0's can be modeled by a ground state Gibbs/Markov Random Field. S. Geman and D. Geman showed in [29] that the posterior distribution Z is a G/MRF with a structure similar to the image model X if the noise process N is independent of the original image X. A Bayesian MAP (maximum a posterior) estimate of the lattice X [29, 30, 31, 25] can be found using a combination of procedures known as stochastic relaxation and simulated annealing on the given noisy lattice Z. The objective of Bayesian MAP estimation is to find an estimate $\hat{X}$ of the original lattice X that maximizes $P(X|Z)$, the conditional probability distribution of X given the noisy lattice Z. The estimate $\hat{X}$ will be noise filtered, i. e. will consist of regions of only 1's and regions of only 0's thus identifying clus-

ters of 0's. However, an approach that employs the G/MRF explicitly poses severe computational limitations. A simple, but novel, method is proposed here.

3.6 A Statistical Method to Calculate Y_0

The non-zero yield regions in the lattice Z are assumed to have an independent and identical distribution (i.i.d.) of 1's and 0's. Hence, the value of the Bernoulli parameter *P* for the non-zero yield regions is given by the ratio of the total number of 1's to the total number of sites in these regions.

We form an initial estimate of *P* as the ratio of the total number of 1's in the lattice to the total number of sites in the lattice. This estimate is used to assign each 0 on the lattice to one of the two classes, "n" for non-zero yield regions and "z" for zero yield regions or clusters of 0's. A new estimate of *P* is formed by discounting 0 sites assigned to clusters of 0's which is then used to reassign all the 0's to one of the two classes, "n" or "z". This procedure is iterated until the classification of 0's no longer changes. The decision of assigning a 0 to "n" or "z" is made by finding yield (ratio of 1's to total number of sites) in a defined neighborhood of the 0 and comparing this neighborhood yield with a threshold *T*. The procedure of calculating Y_o is as follows:

1. Start: $P = Pold$ = (total number of 1's on the lattice) / $|S|$.

2. Visit each 0 and find yield, *yn*, for a defined neighborhood structure *N*.

yn = (number of 1's in the neighborhood) / $|N|$.

If ($yn > T$), 0 --> "n".

If ($yn < T$), 0 --> "z".

where $|N|$ is the total number of sites in the neighborhood structure *N* and *T* is a threshold which is calculated as a function of the current value of *P*.

3. Update *P*. *P* = (total number of 1's on the lattice) / (*S* - total number of "z" 0's).

4. If $(P - Pold) > \varepsilon$, $Pold = P$ and go to step 2. Else go to step 5.

ε is a small number used as a criterion for convergence of the iterations.

5. Y_o = 1 - (total number of "n" 0's) / $|S|$. Stop.

The threshold T, described in the next section, is a positive function of the Bernoulli parameter P for the non-zero yield regions. A high value of P, therefore, will lead to more 0's classified to clusters of 0's. The method starts with an underestimation of P since it assumes at the onset that all 0's belong to the non-zero yield regions. Hence, at each iteration 0's are classified conservatively to clusters of 0's. This ensures that no redundant 0's are classified to clusters of 0's which would result in a higher value of P risking further misclassification of 0's. The method concludes when the value of P does not change significantly which implies that the classification of 0's stayed almost the same. At this stage, the value of the parameter P for the non-zero yield regions and the classification of 0's between clusters of 0's and the non-zero yield regions become consistent with the distribution of 1's and 0's on the lattice.

3.6.1 Choice of the Threshold T and the Neighborhood Structure N

The essence of the method described above is to determine for each 0 whether it belongs to a cluster of 0's. This class assignment decision is made by testing for the hypothesis that a given 0 on the lattice belongs to a cluster of 0's. A 0 well within a cluster of 0's will be surrounded by all 0's and a 0 in a non-zero yield region is likely to be surrounded by both 1's and 0's such that it will have a yn value close to P. The correctness of 0 class assignment, which determines success of the method proposed, depends on the choice of the neighborhood structure N and the threshold T.

A cluster of 0's is probabilistically defined as a continuous region of 0's which is unlikely to be drawn from the random distribution of 1's and 0's in a non-zero yield region given the Bernoulli parameter P. If P is large, a fairly small area consisting of only 0's will

be treated as a cluster of 0's. However, if P is small the occurrence of a continuous region of 0's in a non-zero yield region is likely . Thus the ability to identify clusters of 0's is in some sense fundamentally limited by their size and the value of the Bernoulli parameter P for the non-zero yield regions. This makes it necessary to make the threshold T a function of the parameter P. The threshold T is defined as follows:

$$T = P - \mu. \tag{3.26}$$

μ is a positive number smaller than P. It is given as

$$\mu = n \times \sqrt{\frac{P(1-P)}{|N|}}, \tag{3.27}$$

where n is a positive number determined experimentally.

The form of μ accounts for the variance of the neighborhood yield, *yn*, in the non-zero yield regions. The *yn* value for the non-zero yield regions is a binomial random variable given as follows:

$$E(yn) = P, \sigma^2(yn) = \frac{P(1-P)}{|N|}.$$

The choice of μ as n standard deviations of *yn* accounts for the variability of *yn* from its expected value of P in the non-zero yield regions. The threshold T acts as a lower limit on the *yn* value for the noisy regions. When the calculated neighborhood yield for a given 0 is greater than T, the zero is assigned to the non-zero yield regions. When the calculated neighborhood yield is less than T, the 0 is classified to the clusters of 0's or the zero yield regions.

The fact that T is greater than zero will help in identification of 0's on edges of the clusters of 0's. The edge members of a cluster of 0's will have both 0's in the cluster of 0's and 1's and 0's in the bordering non-zero yield region as their neighbors. This will result in a *yn* value greater than zero but less than the *yn* value for the non-zero yield regions.

The fact that T, a lower limit on the value of yn for the non-zero yield regions, is greater than zero makes correct classification of edge members of clusters of 0's highly likely. The value of n should be chosen to optimize the tradeoff between keeping the value of T high enough for detection of edge members of clusters of 0's and low enough to take account of variance of yn in the non-zero yield regions. We treat n as a parameter of the method developed and determine it using a training data set.

A symmetric neighborhood structure is considered to be an appropriate choice since no knowledge is assumed about the shape of cluster boundaries. A small neighborhood structure will prevent wrong class assignments of 0's on the boundaries of clusters of 0's because few members of the bordering non-zero yield regions will be treated as neighbors of these 0's. However, a small neighborhood structure may result in an incorrect class assignment for a 0 in a non-zero yield region because of high variance of the neighborhood yield. The neighborhood yield, yn, for the non-zero yield regions has a variance that is inversely proportional to the neighborhood size $|N|$. Similarly to the choice of n, a tradeoff between detection of edge members of clusters of 0's and accounting for variance of yn in the non-zero yield regions also exists in choice of the neighborhood size. A second order nearest neighborhood structure, shown in Figure 3.6, optimizes this tradeoff fairly well if the the noise level in the non-zero yield regions is not too high. Specifically, we assume $P \geq 0.2$, where P is the Bernoulli parameter for the non-zero yield regions.

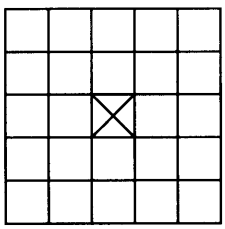


Figure 3.6: A second order nearest neighborhood structure.

The clusters of 0's may exhibit directional preference, for example the 0's may form horizontal or vertical stripes. In such cases choice of the neighborhood structure should depend on the type of dependency present in the lattice, i. e. neighborhood structures should be chosen in the same direction as the direction of spatial dependency present in the lattice. The outcome of the aura framework based spatial randomness test which gives information on presence and type of spatial dependence for a binary lattice is used to select the neighborhood structure. For example, if the spatial randomness test indicates a vertical dependence then a second order nearest subneighborhood that contains neighbors in the vertical direction only is used. The second order nearest neighborhood structure shown in Figure 3.6 is used if no specific directional dependence is indicated by the spatial randomness test. Figure 3.7 shows both calculated and actual Y_o values for four example lattices. The value calculated by the method is denoted by $Y_o c$ and the actual value, known since either none or one cluster of 0s of known size is added to each lattice, is denoted by $Y_o a$.

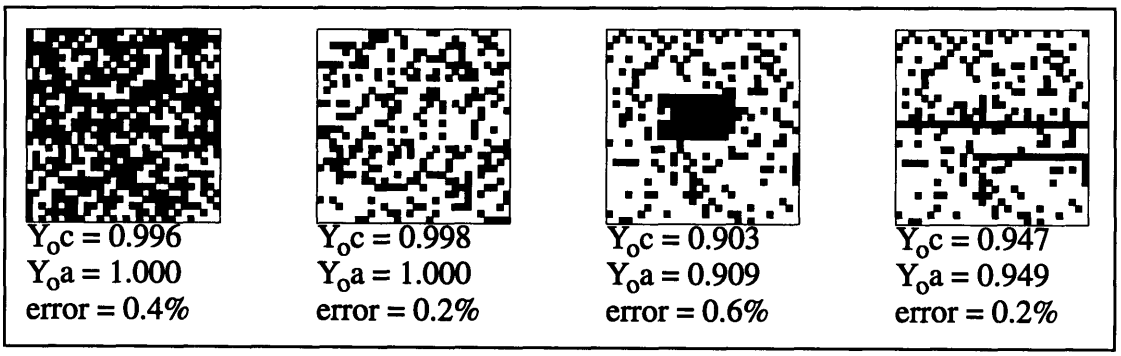


Figure 3.7: Calculated and actual Y_o values for example lattices.

3.6.2 Training and Testing the Method to Calculate Y_o

The method to calculate Y_o is parameterized similarly to the spatial randomness test. The parameters are n and ε where n is the number of standard deviations of yn for the non-zero yield regions subtracted from P to give the threshold T and ε is a small number that deter-

mines convergence of the procedure. Both the training and test data sets required are same as those used for the spatial randomness test, i. e. should consist of a number of two dimensional binary lattices with simulated pseudorandom distribution of 1's and 0's and with known patterns (clusters of 0's) added to the pseudorandom distributions.

The value of Y_o for simulated pseudorandom lattices is 1 by definition since they do not contain any clusters of 0's. The Y_o values for other lattices are also known because a cluster of known size is added to each of these lattices. The method is applied to the training data set repeatedly with parameters altered each time. The set of parameter values that result in calculated Y_o values closest to the actual Y_o values are chosen. The method is found to be reasonably robust with respect to the values of n and ε and independent of the overall yield of the lattice.

A performance estimate for the method can be formed by comparing the Y_o values calculated using the method with the actual Y_o values for the lattices in the test data set. The simulated pseudorandom and cluster added lattices of the test data set are all different from the lattices contained in the training data set. Training and testing a system on the same data set would result in an overestimation of favorable performance. The actual and calculated values of Y_o for all lattices in the test data set, shown in Appendix A, are within 3% of each other which is reasonable accuracy given the unsupervised clustering nature of the problem. The 3% error bound on the calculation of Y_o is claimed for all lattices with non-zero yield region parameter $P \geq 0.2$. The high level of noise in non-zero yield regions with $P < 0.2$ will result in errors greater than 3% in the calculation of Y_o. Appendix A also shows for each original lattice an additional lattice that shows only those 0's that are determined by the method to calculate Y_o to belong to a cluster of 0's.

Chapter 4

Analysis of Wafer Map Data

4.1 Data Description

A wafer map is a two dimensional picture that depicts locations of good and bad chips on wafers. The data analyzed in this work consists of wafer maps for two different IC products of a leading semiconductor manufacturing company. The two products were chosen at random from a choice of wafer map data for several different products manufactured and marketed in the recent past. We name the selected products, product A and product B. A total of 606 wafer maps of product A and 284 wafer maps of product B were available for analysis.

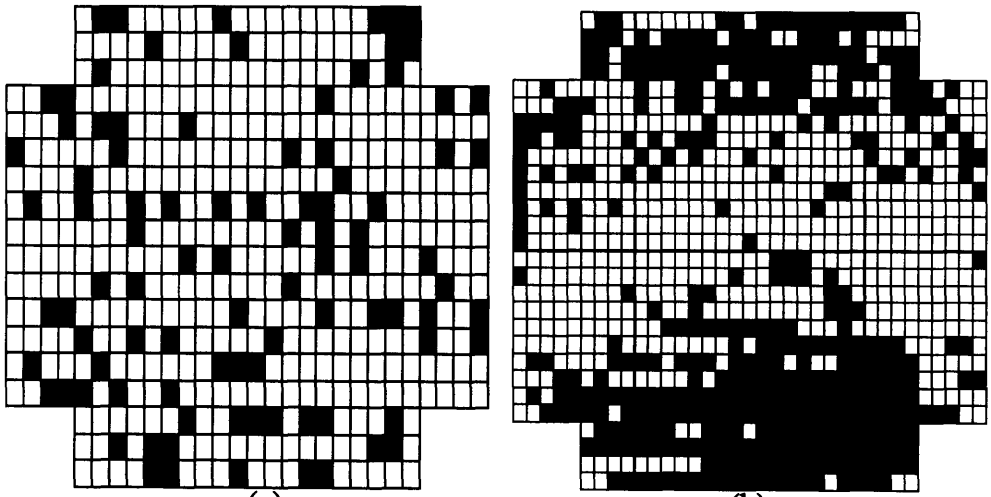


Figure 4.1: (a) Example of a product A wafer. (b) Example of a product B wafer.

Both the products were fabricated on 4-inch silicon wafers. Owing to difference in chip sizes of the two products, product A has a total of 456 chips per wafer and product B has a total of 900 chips per wafer. The wafer map for a given product does not usually have a rectangular boundry; the specific shape of the boundry results from fitting the largest possible number of chips on the circular wafer. A larger number of chips per wafer is

likely to produce higher profits since each wafer of a particular product has a cost of production that does not depend on the number of chips per wafer. Also, since the chips are rectangles and not necessarily squares, there can be more chips along one dimension than the other of the wafer map. Figure 4.1 shows one example each for the wafer maps of the two products considered.

Interestingly enough, the primary reason for collecting wafer map data in most cases is not to perform spatial yield analysis but to help in the packaging of chips. After completion of fabrication, each chip on the wafer map is tested for functionality based on electrical tests specifically designed to determine chip performance. Those chips that are deemed nonfunctional are marked with a red ink dot and the rest are left blank. The wafer is cut into individual chips in the subsequent step whereupon the chips marked with the ink dot are discarded and the remaining chips are packaged into protective cover and sent for further testing. In fabs in which the entire packaging step is automated, the wafer map data is also stored in a computer as a two dimensional array marking the location of good and bad chips. An important recommendation made as a result of this work is that all fabs interested in yield enhancement should make arrangements to store the wafer map data electronically. The wafer map data is a by product of streamlined manufacturing and does not require any extra investment in terms of fab manufacturing time and other resources. The wafer map data which requires minimum acquisition effort contains information in the distribution of good and bad chips on the wafer that can provide important insight into the yield loss mechanism using very low cost analysis.

The yield of different products, lots of wafers of a particular product, and individual wafers of a particular product all show variations. Interpreting and understanding yield variability is crucial for the planning and effective operation of semiconductor manufacturing processes. In this chapter we first carry out a preliminary analysis of the wafer maps

by forming statistical collections of wafers and plotting yield per wafer histograms. We then explore the spatial model for the distribution of good and bad chips on wafers proposed in chapter 2 by applying the methods developed in chapter 3 to the wafer map data.

4.2 Preliminary Data Analysis

Yield models that describe the probability of each chip on a wafer being good by an independent and identical Bernoulli parameter P are overly simplistic. Significant deviations can be found from the independent and identical distribution, i. i. d., assumption, also called the Bernoulli model, by straightforward analysis of the wafer maps. In this section, we investigate yield variations for individual wafers by wafer location across several wafers and with radial distance from the center of the wafer.

4.2.1 Yield Variation by Wafer

The yield of individual wafers varies significantly even for wafers of the same product fabricated on the same manufacturing process. Figure 4.2 shows histogram plots of yield per wafer for products A and B. These graphs serve to provide a visual representation of yield variation for individual wafers. [23] uses the yield variation by wafer to deduce deviations from the Bernoulli yield model. If each chip being good or bad is given by a Bernoulli random variable, the number of all good chips on the wafer is given by a binomial random variable with paremeters P and N, the total number of both good and bad chips on the wafer. It is shown in [23] that the variance of yield of individual wafers is much higher than $\frac{P(1-P)}{N}$ which is the yield variance under the Bernoulli model. We omit this analysis here since in the next section we will apply a test to the wafers that explicitly tests for spatial randomness or, equivalently, the i. i. d. assumption of the Bernoulli model.

4.2.2 Yield Variation by Location

Yield variation by location on wafers investigates systematic yield loss. For instance, one may find a particular region on the wafer that always have failing chips for all the

wafers processed during the same time period. Such yield loss will exist if a machine has a systematic problem causing it to malfunction only at particular locations of the wafers. The orientation of wafers is maintained during processing since each wafer has a flat edge and processing steps are performed with respect to the flat edge. Hence comparison of wafers for locations with consistently low yield will help detect systematic problems in the processing.

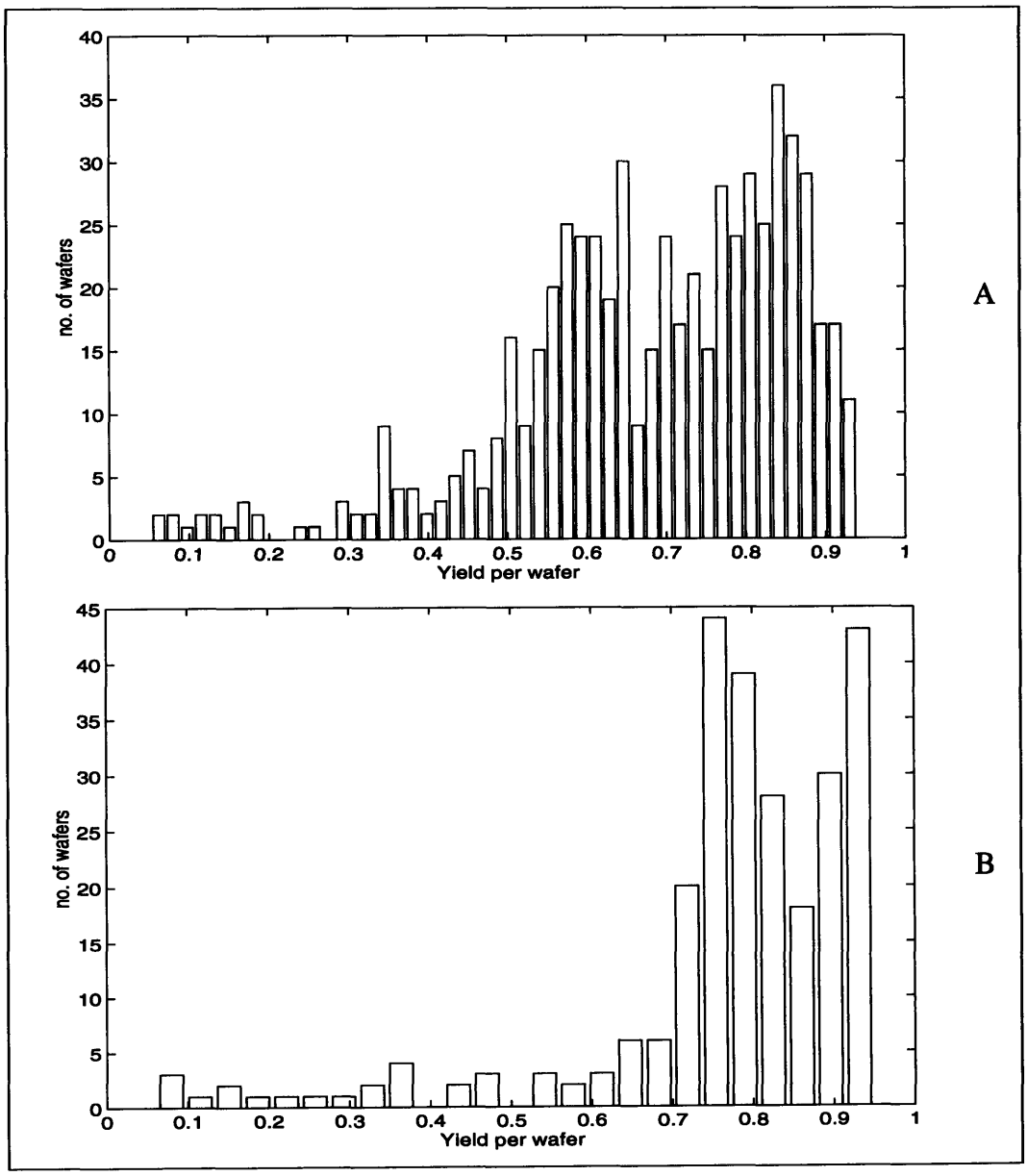


Figure 4.2: Yield distributions for product A and product B wafers.

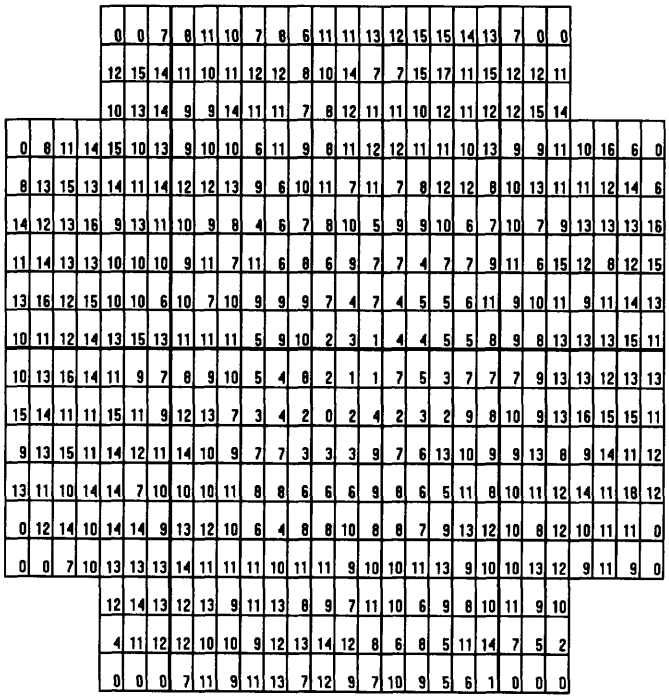


Figure 4.3: A stack map for a product A lot of 18 wafers.

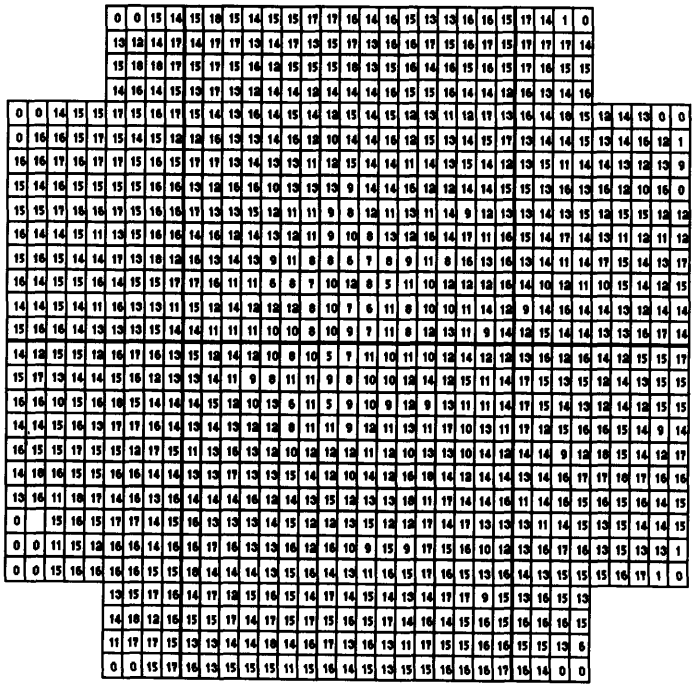


Figure 4.4: A stack map for a product B lot of 18 wafers.

We form statistical collections for each lot by forming a different type of wafer map, called stack map, whose each location is given by the number of wafer maps in the lot that have a good chip at that location. Figure 4.3 shows a stack map for a randomly chosen lot of product A that has a total of 18 wafers and Figure 4.4 shows a stack map for a randomly chosen lot of product B that also has 18 wafers. Both the stack maps have some locations on edges at which no wafer had a good chip, indicating misprocessing at these edge locations for all wafers in the lot. The consistency of chip failures on edges is easily explained by well understood causes known as edge effects, discussed in chapter 2. Neglecting the few zero locations on edges, the number of wafers with good chip on a particular location should be randomly distributed over the stack map if there were no systematic dependencies. However, the stack map examples in Figures 4.3 and 4.4 show that certain locations have significantly lower yield for these lots.

If there is no yield dependency by location in a given lot then each location on its stack map can be represented by a binomial random variable Q defined by

$$E(Q) = P \times N,\ Var(Q) = N^2 P(1-P),$$

where P is the average yield for the lot and N is the number of wafers in it. We form a binary wafer using the stack map such that any location on the stack map that has yield greater than E(Q) is represented by a '1' and a location that has yield less than E(Q) is represented by a '0'. These binary representations of stack maps of Figures 4.3 and 4.4 are shown in Figure 4.5.

It follows that this binary representation of the stack map should have a random distribution of 1's and 0's if no yield variation by location exists. Application of the spatial randomness test developed in chapter 3 to these binary wafers shows strong clustering of like elements. This verifies that yield tends to be high in some regions and low in other regions on wafers in the same lot. However, this analysis in itself is inconclusive because the

lower yield regions on the stack maps have non-zero yields. It cannot be deduced that some machines always malfunction in regions of low yield if these regions have non-zero yield. We will be better able to provide reasons for the existence of regions of high and low yields on stack maps after we apply the spatial randomness test to individual wafers in the next section.

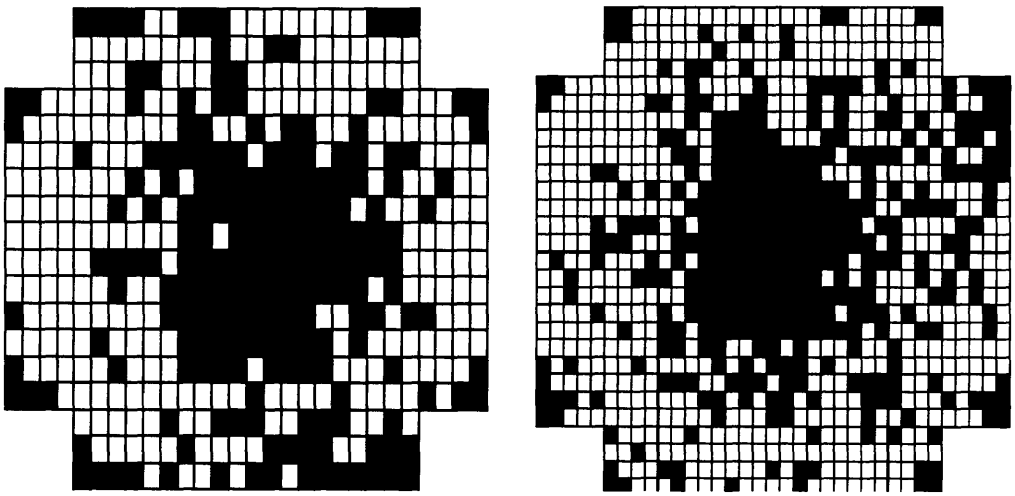


Figure 4.5: Binary representation of stack maps in Figures 4.3 and 4.4.

4.2.3 Yield Variation by Radial Distance

Yield is known to exhibit radial dependency on any particular wafer. A commonly observed radial yield variation on wafers is high yield in the center of the wafer and decrease in yield with radial distance from the center. On the contrary, donut yield patterns, i. e. low yield in the center and high yield on the outer regions, are also occasionally seen on wafers. It should be noted that almost in all cases yield on the very outer edges of wafers will be low because of the previously discussed edge effects that arise due to handling of wafers and are present on almost all wafers. Radial dependency other than yield loss on edges is likely to be caused by errors in processing steps in which the wafer is spun on a vacuum chuck and a chemical is applied to the center so as to be spread uniformly over the wafer (see chapter 2). A well known method of exploring radial dependency of

yield is to plot yield in equal area annuli on the wafer versus radial distance of the annuli from the center. If no radial dependency exists, each annulus will have the same average yield. The yield in equal area annuli versus radial distance plot constitutes important analysis tool for a technician analyzing individual wafers in real time.

4.3 Application of the Spatial Randomness Test and the Method to Calculate Y_0

Yield variation by wafer establishes the fact that semiconductor yield shows high variability. Yield variations, if conclusive, by location and radial distance from the center of the wafer indicate existence of systematic yield loss. Although exploring yield variation by location and radial distance constitutes an obvious analysis tool, these dependencies are rarely clear enough to be useful as a yield diagnostic. In this section we investigate the spatial yield model for distribution of good and bad chips on wafers proposed in chapter 2 using the spatial randomness test and the method to calculate Y_0 developed in chapter 3.

It is proposed, based on an investigation of IC fabrication and visual inspection of wafers, that two types of regions exist on wafers; the non-zero yield regions which contain both good and bad chips mixed in a random fashion and the zero yield regions which contain almost all bad chips. The presence, location, size, and shape of zero yield regions all vary from wafer to wafer even for wafers in the same lot. Although the yield of non-zero yield regions also varies from wafer to wafer, the variation in size of zero yield regions and their absence from some wafers account for part of the variability observed in number of wafers versus yield per wafer plots in Figure 4.1. No zero yield regions appear on stack maps in Figures 4.3 and 4.4 because the zero yield regions may occur at different locations and have different shapes and sizes on different wafers. The existence of regions of low and high yields on the stack maps can be explained by limited, instead of complete, overlap among the locations of zero yield regions on individual wafers of a given lot. It will be

shown in this section that the most informative spatial dependency on individual wafers exists in the form of zero yield regions.

The presence of a zero yield region is detected by applying the spatial randomness test which gives a binary output indicating the presence or absence of zero yield regions on the wafer. The method to calculate Y_o finds the zero yield regions and calculates Y_o where (1-Y_o) is the fraction of the wafer occupied by the zero yield regions on the wafer. The available wafers for each product are divided into two groups, group L, for local random deformation mechanisms, and group G, for gross random deformation mechanisms. Based on the outcome of the spatial randomness test, the wafers with zero yield regions are classified to group G while those with only random distribution of failing chips are classified to group L. A few issues need to be taken care of in applying the spatial randomness test and the method to calculate Y_o to wafer maps of products A and B to avoid misleading results. We state the problems that may arise and suggest solutions based mostly on qualitative reasoning.

4.3.1 Edge effects

The preliminary analysis carried out on wafers has shown that certain locations on the extreme edges of wafers always contain failing chips. This is called the 'edge effect' and is attributed to handling of wafers between process steps and their storage in slots in a box that contain an entire lot of wafers. This is a well understood and mostly insignificant source of yield loss. However, the edge effects can give misleading results for wafers with high yield and random distribution of failing chips. The small clusters of bad chips on the edges of a high yielding wafer with no zero yield regions may be significant enough for the wafer to be classified to group G. We address this problem by turning from bad to good the permanently failing chip sites on extreme edges of wafers of a lot before the application of the spatial randomness test.

4.3.2 Choice of the neighborhood structure for the spatial randomness test

The faults that cause chips to become bad in non-zero yield regions may be distributed independently or may exhibit limited clustering previously named local clustering. In the case of local clustering the likelihood of finding a local fault in the vicinity of other local faults is greater than the likelihood of finding isolated faults. The local clustering of faults will cause the bad chips in non-zero yield regions to exhibit similar local clustering. This local clustering of bad chips may be large enough in certain cases for the nearest neighbor randomness test to classify a wafer to group G which actually belongs to group L. To avoid such an error, we use a first order nonnearest neighborhood structure instead of the first order nearest neighborhood structure in the aura framework based spatial randomness test. This was the reason for demonstrating the training and testing procedures of the spatial randomness test on both the first order nearest and nonnearest neighborhood structures in section 3.4.3. Using the first order nonnearest neighborhood structure amounts to using close by neighbors to find spatial dependencies while ignoring very small range spatial dependencies.

Product	L(N)	G(N)	L(NN)	G(NN)	Total
A	245	361	358	248	606
B	132	152	171	113	284

Table 4.1: Classification of wafer maps based on nearest and nonnearest neighbors

Local clustering will cause more wafers to be assigned to group G than actually belong to this group if nearest neighbors are used. This is confirmed by the results shown in Table 4.1. G(NN), the number of wafers assigned to group G based on the use of nonnearest neighbors, is less than G(N), the number of wafers assigned to group G using nearest neighbors. Similarly, L(N), the number of wafers assigned to group L when nearest neigh-

bors are used, is less than L(NN), the number of wafers assigned to group L based on the use of nonnearest neighbors. This comparison supports that local clustering exists and improved classification to groups G and L takes place if a nonnearest neighborhood structure is used. Hence, all further analysis will use classification based on the use of nonnearest neighbors.

4.3.3 Model Validation

The method to calculate Y_o when applied to individual wafers of group L gave Y_o values very close to 1. It calculated a value of Y_o smaller than 1 for the wafers of group G. Since it is cumbersome to state Y_o values for individual wafers, we give in Table 4.2 the average value of Y_o for the two groups of each product where the wafers are classified in groups G and L based on the use of nonnearest neighbors in the spatial randomness test.

Product	Group L	Group G
A	0.975	0.851
B	0.974	0.762

Table 4.2: Average Y_o values for the L and G group wafers

The group L wafers possess no zero yield regions as determined by the spatial randomness test and hence should have a Y_o value close to 1. Table 4.2 shows that the group L wafers indeed have an average Y_o value close to one for both the products. The approximately 2.5% difference of these values from 1 is attributed to the local clustering present in non-zero yield regions and the fact that the method to calculate Y_o was empirically determined to have an upper error bound of 3% (see chapter 3). The Y_o values for group G wafers show that on average 14.9% of each product A wafer is occupied by the zero yield regions while on average 23.8% of each product B wafer is occupied by zero yield regions.

The wafers in group L encounter only local deformation mechanisms while wafers in group G are subject to both local and gross random deformation mechanisms. Hence, wafers in group G should, on average, have yield lower than yield of group L wafers. It is important to note that this correlation of wafer yield to its category will hold only for yield averaged over a sufficiently large number of wafers in each group since both the size of zero yield regions and the yield of non-zero yield regions are independent of each other and vary from wafer to wafer. Table 4.3 shows that average yield of group L wafers is higher than average yield of group G wafers for both the products A and B.

Product	Group L	Group G
A	0.723	0.615
B	0.828	0.627

Table 4.3: Average yield values for the L and G group wafers

The gross random deformation mechanisms can be treated to be randomly incident on some of the wafers all of which are subject to the same local deformation mechanisms. A wafer in group G can therefore be treated as having a zero yield region superimposed on a wafer with random distribution of failing chips. If this model of deformation mechanisms is correct, the two groups encounter the same process of local deformation mechanisms. The defect limited yield, the yield for non-zero yield regions, for the two groups should therefore be equal. We use the relation [1]

$$Y = Y_o \times Y_r, \tag{4.1}$$

that holds for any given wafer to calculate the average defect limited yield of a group of wafers. We propose two relations in equations 4.2 and 4.3 both of which can be used to calculate the average defect limited yield.

$$\tilde{Y}_r = \frac{1}{N}\sum_{i=1}^{N}\frac{Y^i}{Y_o^i} \tag{4.2}$$

$$\overline{Y}_r = \frac{\frac{1}{N}\sum_{i=1}^{N} Y^i}{\frac{1}{N}\sum_{i=1} Y_o^i} \tag{4.3}$$

where Y^i is the measured yield and Y_o^i is the calculated gross yield of wafer i in the group of N wafers. Equation 4.2 follows straightforwardly from Equation 4.1 while Equation 4.3 holds because Y_o and Y_r in Equation 4.1 are independent of each other. Since the method to calculate Y_o gives only an estimate of the true Y_o value of the wafer, i. e. the Y_o value calculated by the method contains statistical error, we expect Equations 4.2 and 4.3 to give different estimates of average defect limited yield. For this reason, the average defect limited yield values calculated using Equations 4.2 and 4.3 are denoted by $\tilde{Y}_r$ and $\overline{Y}_r$, respectively.

The Y_o values in Table 4.2 and comments in the beginning of this subsection provide us with legitimate reason to assume a value of unity for Y_o of wafers in group L, i. e.

$Y_o(L) = 1, ==> Y(L) = Y_r(L)$

Using this assumption and the Y_o values calculated for individual wafers of group G, We show in Tables 4.4 and 4.5 $\tilde{Y}_r$ and $\overline{Y}_r$ values, respectively, for groups L and G of wafers of products A and B.

Product	No. of Wafers	Group L	Group G	Mismatch
A	606	0.723	0.714	1.24%
B	284	0.828	0.751	9.30%

Table 4.4: $\tilde{Y}_r$ values for the L and G group wafers

Product	No. of Wafers	Group L	Group G	Mismatch
A	606	0.723	0.7227	0.04%
B	284	0.828	0.823	0.60%

Table 4.5: $\overline{Y}_r$ values for the L and G group wafers

Clearly, the average defect limited yield of wafers in group G extracted using Equation 4.3 matches better with the average defect limited yield of wafers of group L than does the average defect limited yield of group G wafers found using Equation 4.2. This can be explained in terms of the statistical error in the calculation of Y_o values. We argue in Appendix B that the effect of statistical error in the calculation of Y_o cancels out in Equation 4.3 whereas it becomes magnified in Equation 4.2. Given that both Equations 4.2 and 4.3 are alternative ways of computing average defect limited yield, we treat Table 4.5 as our final result.

It is observed from Table 4.5 that the match between group L and group G defect limited yields is highly accurate. Given the unsupervised clustering nature of the detection and measurement of zero yield regions, we can justifiably claim that the extracted defect limited yields for the two groups has matched much better than one would expect at the onset of such an analysis. Also, we notice that a better match results for product A possibly because its sample size (the number of product A wafers) is larger than the sample size of product B. This is supported by dividing the 606 wafers of product A into two groups, A1 and A2, of 303 each and then carrying out the analysis. Table 4.6 shows the defect limited yield values for the three categories, A1, A2, and B.

Product	No. of Wafers	Group L	Group G	Mismatch
A1	303	0.670	0.686	2.38%
A2	303	0.777	0.758	2.45%
B	284	0.828	0.823	0.60%

Table 4.6: Average Y_r, defect limited yield, values for the L and G group wafers

The results of applying the spatial randomness test and the method to calculate Y_o to products A and B are summarized below:

1. The distribution of failing chips in regions which contain both good and bad chips exhibit minor local clustering.

2. The average yield of wafers with zero yield regions is lower than the average yield of wafers with only random mixing of good and bad chips.

3. The average defect limited yield for wafers with zero yield regions is equal to the average defect limited yield of wafers with no zero yield regions.

These results validate the model that a wafer consists of one or both of the two types of regions; zero yield regions which contain almost all bad chips and non-zero yield regions which consist of both good and bad chips mixed in a random fashion. The failing chips in non-zero yield regions may be independently and identically distributed or may exhibit local clustering.

Chapter 5

Conclusions

5.1 Summary

The distribution of good and bad chips on a semiconductor wafer contains information important for yield management in semiconductor manufacturing. This distribution typically results in two types of regions, one that contains both good and bad chips mixed in a random fashion called a non-zero yield region and the other that contains almost all bad chips called a zero yield region. We have developed two statistical methods in this work to investigate the distribution of good and bad chips on a given wafer. The first method is a spatial randomness test that detects the presence of zero yield regions on a wafer by finding the presence and nature of any spatial dependencies in the distribution of bad chips on the wafer. This method uses a set-theoretic statistical image analysis tool called aura framework. The aura framework is based on ideas similar to Gibbs/Markov Random Fields. The second method is a novel statistical method which finds those bad chips on a wafer that belong to zero yield regions. This method is based on hypothesis testing on nearest neighbors of bad chips. The application of the two statistical methods developed in this work to wafer maps of commercial IC products has shown that zero and non-zero yield regions indeed exist on wafers. In conclusion, we show that the methods developed have both yield analysis and yield modeling value.

5.2 Wafer Yield Analysis

It is argued in chapter 2 that there are two broad classes of deformation mechanisms that can make an IC nonfunctional. One of these, called local random deformation mechanisms, causes non-zero yield regions and the other, called gross random deformation mechanisms, causes zero yield regions. The local deformation mechanisms are considered

inherent to the manufacturing environment. That is, although these can be controlled and suppressed to improve the manufacturing yield, they cannot be totally eliminated. The gross deformation mechanisms, on the other hand, can be eliminated almost completely as experience with a process increases and the fab operations become better understood. The first method that finds the presence of zero yield regions on a given wafer equivalently finds whether the wafer encountered gross yield loss. The second method serves to divide the bad chips between zero and non-zero yield regions. This is equivalent to knowing the fraction of yield loss due to each of the two classes of deformation mechanisms. The knowledge of the dominant class of yield loss mechanisms can be crucial for yield management and process engineering to determine the subsequent steps in investigation of yield loss. The detection of the precise causes of yield loss usually involves processing and analysis of a wide variety of special test wafers and test structures on product wafers. Prior knowledge of the dominant type of deformation mechanisms will facilitate the choice of test wafers and structures thus saving both processing time and resources invested in diagnostic. Even if preset yield targets are being met, the application of the two methods developed can be useful. A proactive opportunity for yield improvement exists if non-negligible gross yield loss is present in wafers with acceptable overall yield levels. Figures 5.1 and 5.2 each show a set of yield per wafer, gross yield per wafer, and defect limited yield per wafer histograms for the two products analyzed in chapter 4. It can be seen from these plots that defect limited yields are higher than overall yields. That is, if gross yield loss is eliminated the yield distribution will shift to its right. Most mature fabs strive to increase their desired yield levels by eliminating gross deformation mechanisms.

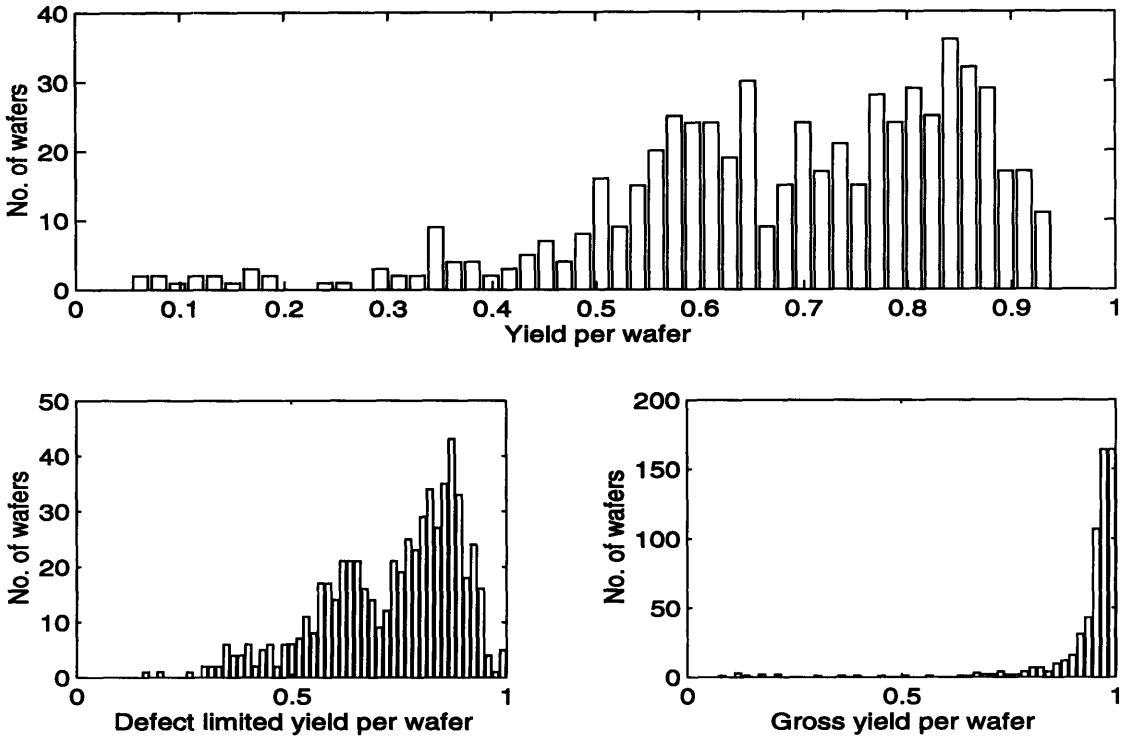


Figure 5.1: Yield histogram plots for product A.

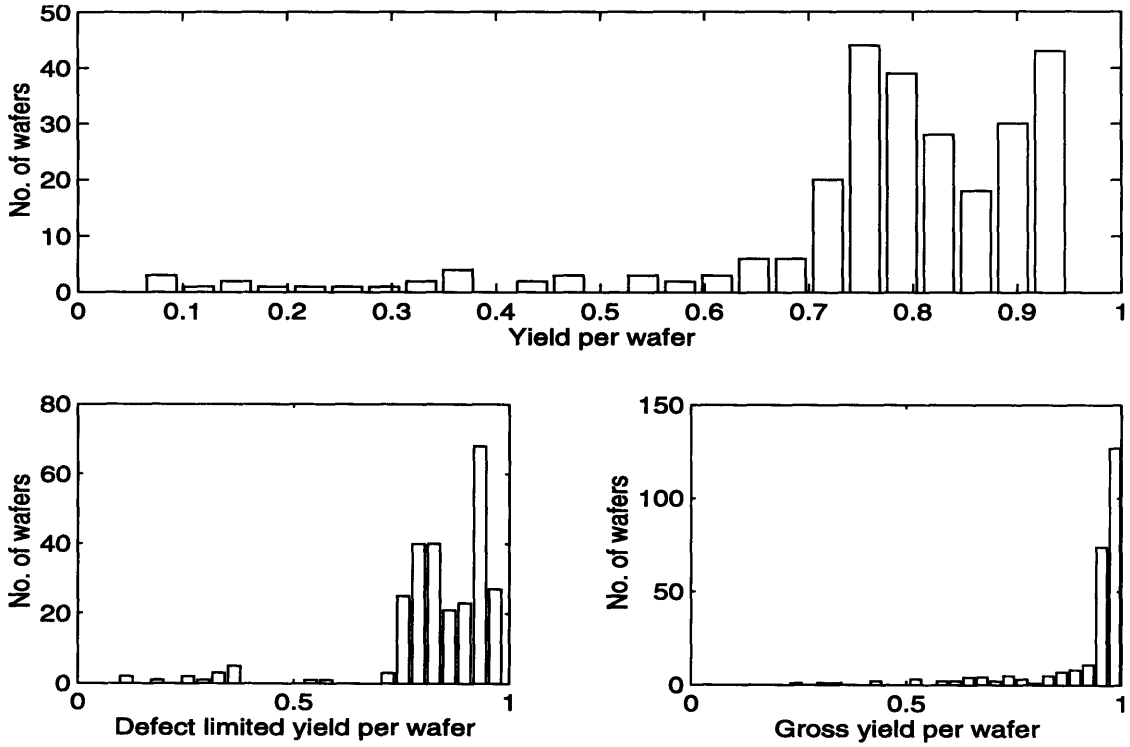


Figure 5.2: Yield histogram plots for product B.

Further advantages of this analysis can be listed as follows:

1. The cost of implementing the analysis scheme is fairly low since both the methods developed are straightforward to implement in software.

2. The computational requirements of the methods developed are trivial compared to some other methods of two dimensional analysis.

3. Minimum acquisition effort is required in collecting the chip functionality data since it is usually available as a by product of the manufacturing process.

5.3 Wafer Yield Modeling

We show in chapter 4 that the model that wafers consist of zero and non-zero yield regions holds fairly well. The first method developed is used to divide wafers of a particular IC product into two categories. All wafers with no zero yield regions are classified together and called group L and all wafers with one or more zero yield regions are classified together and called group G. The average yield of group L wafers is found to be higher than the average yield of group G wafers. This result makes sense if individual wafers in both L and G groups are considered to encounter defect limited yield loss from same local deformation mechanisms. The group G wafers, in addition, also encounter gross yield loss, i. e. yield loss caused by gross deformation mechanisms. Hence, average yield of group G wafers is found, as expected, to be lower than the average yield of group L wafers. Furthermore, we found the average yield of non-zero yield regions of group G wafers using the second method since it separates the zero and non-zero yield regions. The average yield of non-zero yield regions of group G wafers should be close to average yield of group L wafers if we have a sufficiently large number of wafers in each group and if the model of spatial distribution assumed in this work is correct. Table 4.5 shows that the average yield of non-zero yield regions, i. e. the average defect limited yield, of group G

wafers matches very closely with the average yield of group L wafers for the two commercial IC products used. Thus the model assumed for the distribution of good and bad chips on wafers is explicitly verified.

The verification of the proposed spatial distribution of good and bad chips on wafers can be used to boost up the low confidence, as done in [2], that has been placed in the practice of yield modeling and prediction. The original Murphy's model [8] has been repeatedly modified [1, 2, 4] to take into account of the correct nature of statistical distribution of faults on wafers. Hence, it is the statistical distribution of faults on wafers that tends to be the uncertain element in yield models. Since each bad chip represents the location of one or more faults, we have investigated the general nature of the distribution of faults on wafers.

5.3.1 Components of Yield Modeling

Most widely used yield models [1, 2, 4, 6] assume that wafers typically contain both zero and non-zero yield regions. The faults in non-zero yield regions are distributed in some random fashion. This random distribution is either such that the probability of getting faults in each chip is independent and identical or faults shows limited tendency to exist in the vicinity of each other called local clustering of faults. The distribution of faults is therefore appropriately modeled using either the Poisson, for independent and identical distribution of faults, or the negative generalized binomial distributions, for local clustering of faults. The extent of local clustering depends on the technology used and the chip size. Yield models for non-zero yield regions are commonly called defect limited yield models. The presence of zero yield regions is modeled by the scaling factor Y_o which is the fraction of the wafer occupied by non-zero yield regions. The yield predicted by a defect limited yield model is multiplied by Y_o to provide the overall wafer yield. We list additional components of correct yield modeling as follows:

1. Parameters of the yield model used should be carefully chosen. For example, an input to defect limited yield models is defect density. Carefully measured defect density should be used instead of using numbers based on past experience or curve fitting using a simple yield model. Also, a careful estimate for Y_o should be made and the yield predicted by defect limited yield models scaled appropriately.

2. A choice between Poisson and generalized negative binomial model should be made depending on the level of local clustering. See section 2.5 for a more detailed discussion.

3. Critical area, see section 2.6, instead of die area should be used as input to the defect limited yield model to avoid underpredictions by yield models as sometimes reported in literature.

4. When measured yield is compared with the predicted yield to access the success of yield models, Y_o should be calculated first, using the second method developed, to extract the underlying defect limited yield. This extracted defect limited yield should be compared with the yield predicted by the defect limited yield model used. The gross yield variation can be severe and may reduce credibility in the yield modeling if the Y_o value is not accounted for. If measured defect limited yield does not match the predicted defect limited yield, effort should be made to investigate the cause for discrepancy which may be due to the value of the defect density used, choice between Poisson and generalized negative binomial models, or the calculation of the critical area.

Appendix A

Testing the Spatial Randomness Test and the Method to Calculate Y_0

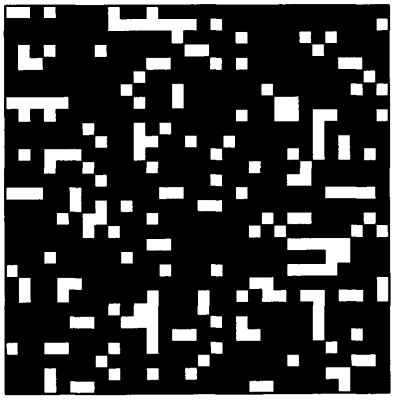
This appendix shows results for applying the spatial randomness test and the method to calculate Y_o to several 30x30 binary lattices in which a '1' is represented by a white pixel and a '0' is represented by a black pixel. Each set of results contains the original lattice, which is a simulated pseudorandom lattice with or without a cluster of zeros added, and a lattice which marks out only those 0's that the method to calculate Y_o determined to belong to a cluster of 0's. Such a lattice is labeled as a noise filtered lattice.

Since 1's represent good chips and 0's represent bad chips on wafers, yield is calculated as the ratio of the total number of 1's on the lattice to the total number of sites on the lattice. The spatial randomness test was applied twice to each lattice using the first order nearest, denoted by N, and the first order nonnearest, denoted by NN, neighborhood structures. The spatial dependency found in the lattice based on the first order nearest neighbors is denoted by SDN and the spatial dependency found based on the first order nonnearest neighbors is denoted by SDNN. Trace value refers to the value of the trace of the normalized aura matrix, which is the statistics of interest in the spatial randomness test, for the given lattice calculated using various neighborhood structures. The trace values calculated using 8-nearest or 16-nonnearest neighbors, 2-nearest or nonnearest neighbors in the horizontal direction, 2-nearest or nonnearest neighbors in the vertical direction, 2-nearest or nonearest neighbors in the left diagonal direction, and 2-nearest or nonnearest neighbors in the right diagonal direction are represented by C, H, V, LD, and RD respectively.

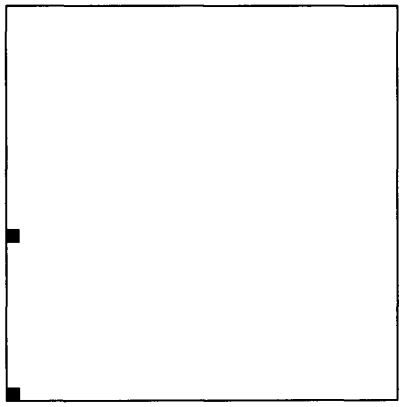
Both the actual and calculated Y_o values for each wafer are given. The actual value of Y_o is known because clusters of 0's of known sizes were added to the lattices with pseudo-

random distribution of 1's and 0's. The actual value of Y_o is compared with the value of Y_o calculated by the method to calculate Y_o and the discrepency is shown as a percentage error.

A simulated pseudorandom lattice

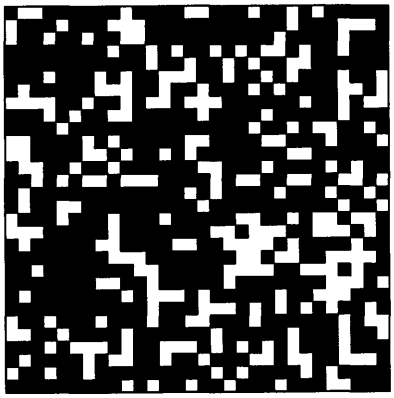


The noise filtered lattice

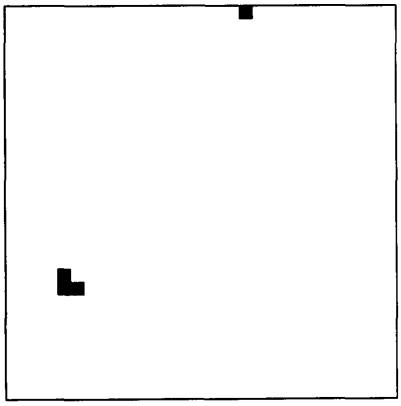


Yield = 0.218; SDNN = Random; SDN = Random
NN: C=1.002 H=1.091 V=1.013 LD=1.015 RD=1.005
N: C=1.007 H=1.095 V=0.986 LD=0.974 RD=0.975
Actual Yo = 1.000; Calculated Yo = 0.998; % error in calculation of Yo = 0.2%

A simulated pseudorandom lattice

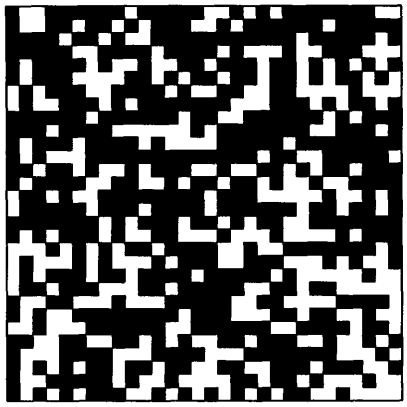


The noise filtered lattice

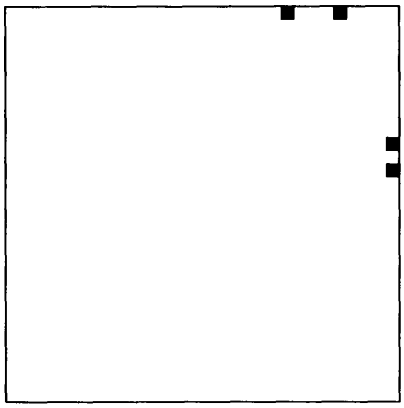


Yield = 0.318; SDNN = Random; SDN = Random
NN: C=1.007 H=1.003 V=1.020 LD=0.997 RD=0.989
N: C=1.013 H=1.000 V=1.013 LD=1.026 RD=1.013
Actual Yo = 1.000; Calculated Yo = 0.996; % error in calculation of Yo = 0.4%

A simulated pseudorandom lattice

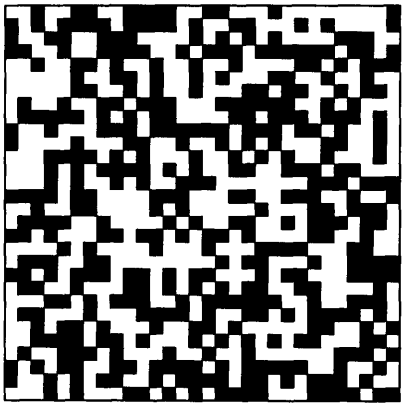


The noise filtered lattice

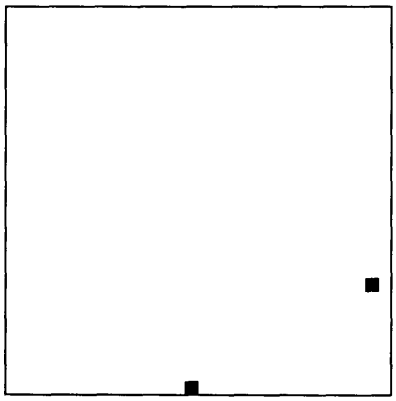


Yield = 0.392; SDNN = Random; SDN = Random
NN: C=1.010 H=1.049 V=1.039 LD=0.954 RD=0.965
N: C=1.028 H=0.996 V=1.014 LD=1.049 RD=1.051
Actual Yo = 1.000; Calculated Yo = 0.996; % error in calculation of Yo = 0.4%

A simulated pseudorandom lattice

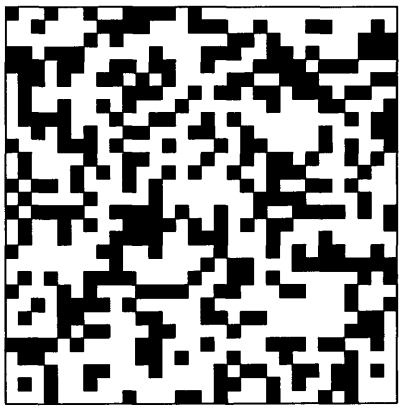


The noise filtered lattice

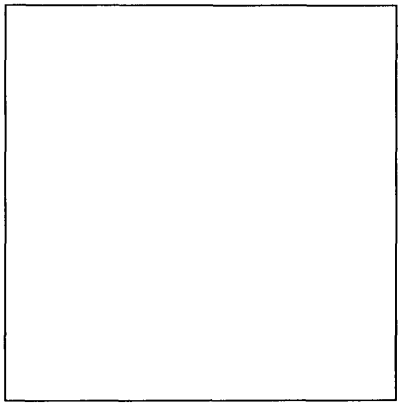


Yield = 0.522; SDNN = Random; SDN = Random
NN: C=0.999 H=0.993 V=1.009 LD=0.957 RD=1.015
N: C=0.977 H=0.939 V=1.003 LD=1.008 RD=0.959
Actual Yo = 1.000; Calculated Yo = 0.998; % error in calculation of Yo = 0.2%

A simulated pseudorandom lattice

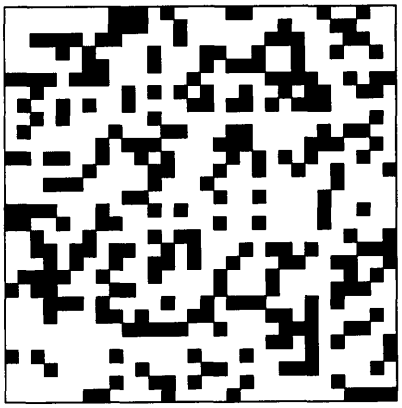


The noise filtered lattice

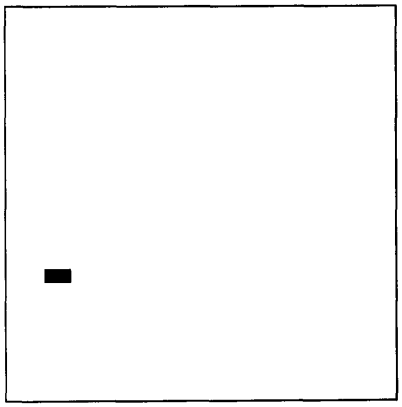


Yield = 0.621; SDNN = Random; SDN = Random
NN: C=1.004 H=1.014 V=1.017 LD=0.999 RD=0.957
N: C=1.000 H=1.013 V=0.983 LD=0.977 RD=1.028
Actual Yo = 1.000; Calculated Yo = 1.000; % error in calculation of Yo = 0.0%

A simulated pseudorandom lattice

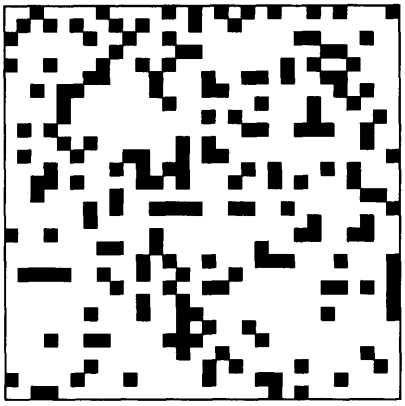


The noise filtered lattice

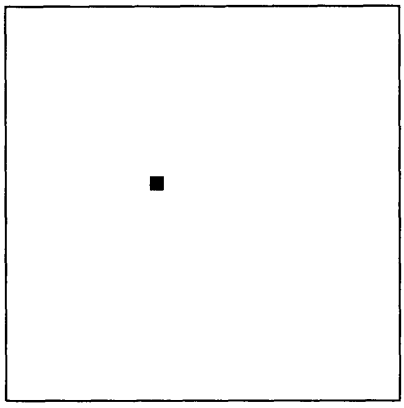


Yield = 0.696; SDNN = Random; SDN = Random
NN: C=0.991 H=1.001 V=1.000 LD=1.003 RD=1.002
N: C=1.017 H=1.061 V=0.979 LD=0.973 RD=1.056
Actual Yo = 1.000; Calculated Yo = 0.998; % error in calculation of Yo = 0.2%

A simulated pseudorandom lattice

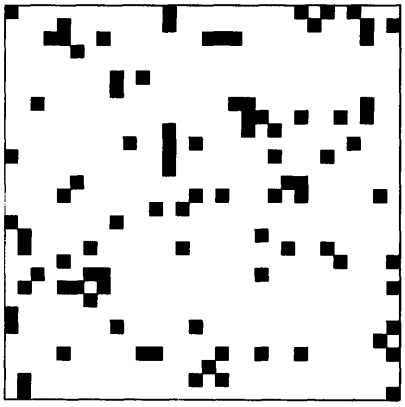


The noise filtered lattice

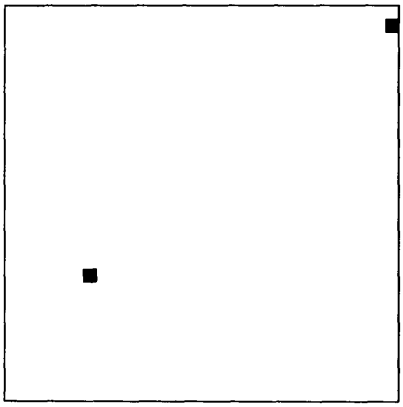


Yield = 0.783; SDNN = Random; SDN = Random
NN: C=1.010 H=1.003 V=1.069 LD=1.003 RD=1.071
N: C=0.976 H=0.969 V=0.985 LD=0.979 RD=0.972
Actual Yo = 1.000; Calculated Yo = 0.999; % error in calculation of Yo = 0.1%

A simulated pseudorandom lattice

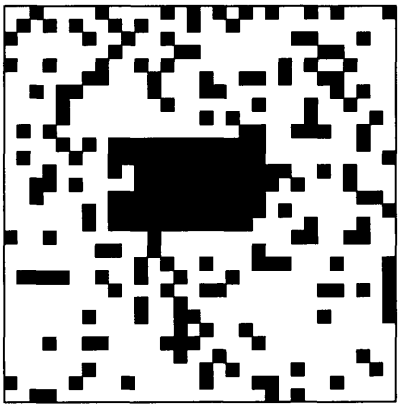


The noise filtered lattice

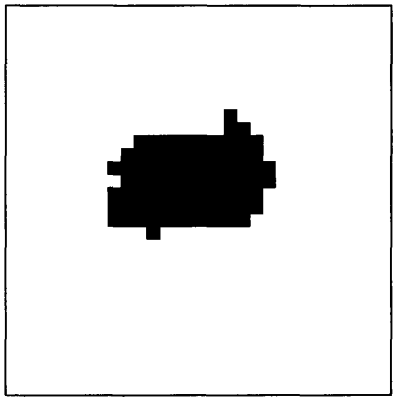


Yield = 0.896; SDNN = Random; SDN = Random
NN: C=1.007 H=1.088 V=1.049 LD=0.983 RD=0.965
N: C=1.042 H=0.993 V=1.073 LD=1.052 RD=1.051
Actual Yo = 1.000; Calculated Yo = 0.998; % error in calculation of Yo = 0.2%

A cluster added lattice

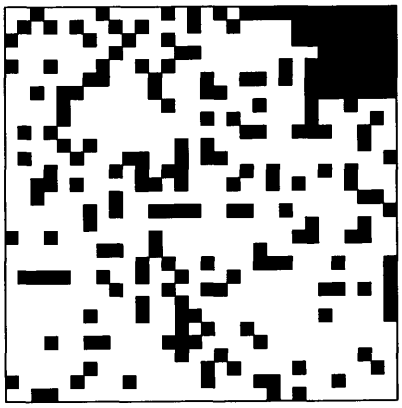


The noise filtered lattice

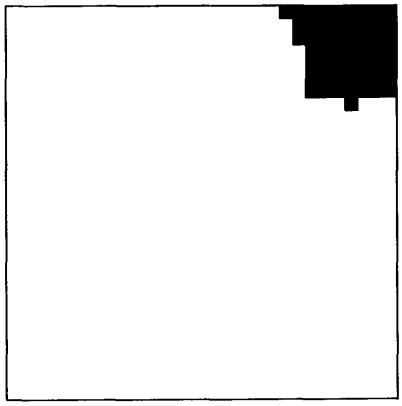


Yield = 0.720; SDNN = Clustered; SDN = Clustered
NN: C=1.153 H=1.199 V=1.183 LD=1.128 RD=1.163
N: C=1.175 H=1.188 V=1.173 LD=1.163 RD=1.174
Actual Yo = 0.909; Calculated Yo = 0.903; % error in calculation of Yo = 0.6%

A cluster added lattice

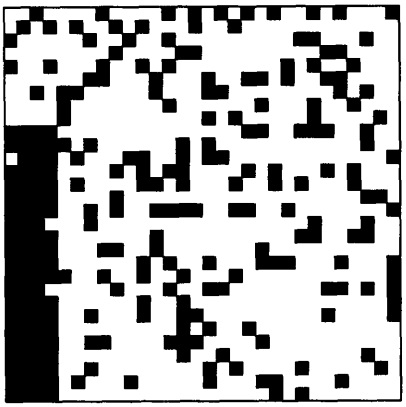


The noise filtered lattice

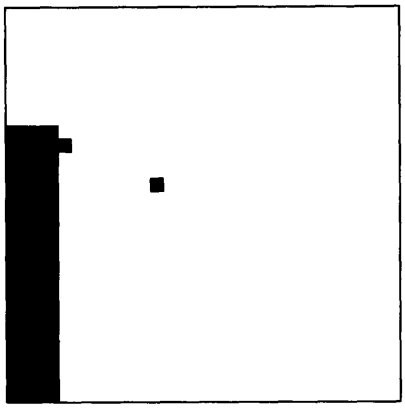


Yield = 0.740; SDNN = Clustered; SDN = Clustered
NN: C=1.137 H=1.161 V=1.195 LD=1.123 RD=1.167
N: C=1.130 H=1.145 V=1.150 LD=1.109 RD=1.117
Actual Yo = 0.941; Calculated Yo = 0.939; % error in calculation of Yo = 0.2%

A cluster added lattice

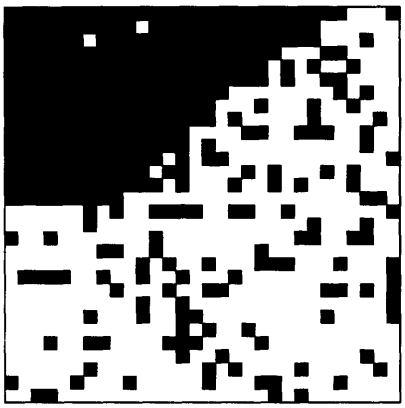


The noise filtered lattice

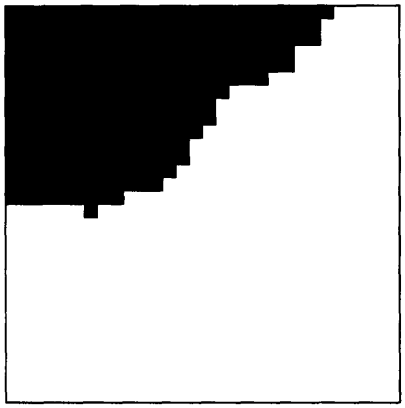


Yield = 0.711; SDNN = Clustered; SDN = Clustered
NN: C=1.172 H=1.144 V=1.270 LD=1.140 RD=1.208
N: C=1.178 H=1.160 V=1.210 LD=1.179 RD=1.162
Actual Yo = 0.908; Calculated Yo = 0.907; % error in calculation of Yo = 0.1%

A cluster added lattice

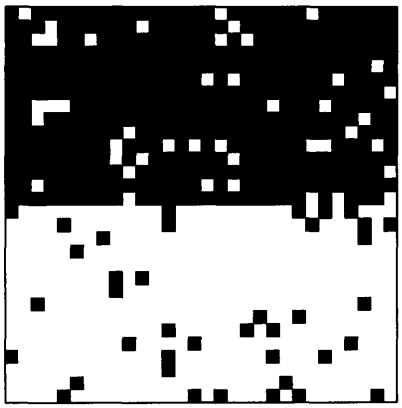


The noise filtered lattice

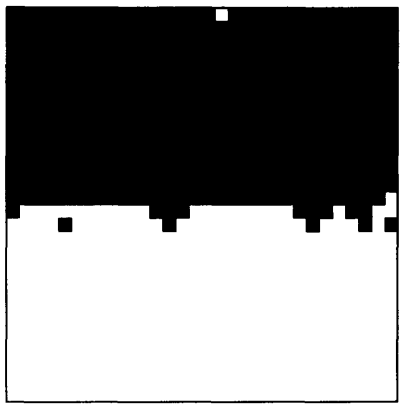


Yield = 0.573; SDNN = Clustered; SDN = Clustered
NN: C=1.453 H=1.465 V=1.471 LD=1.417 RD=1.484
N: C=1.463 H=1.499 V=1.470 LD=1.443 RD=1.440
Actual Yo = 0.714; Calculated Yo = 0.708; % error in calculation of Yo = 0.8%

A lattice with uneven halves

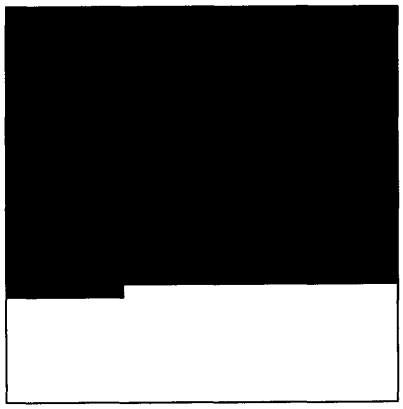


The noise filtered lattice

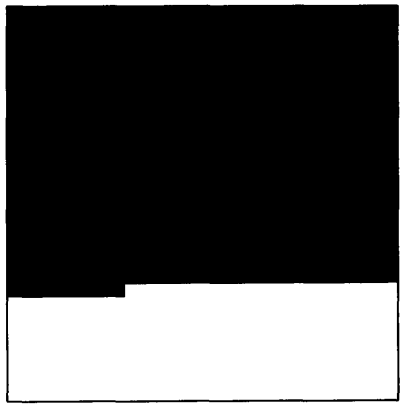


Yield = 0.506; SDNN = Clustered; SDN = Clustered
NN: C=1.619 H=1.706 V=1.608 LD=1.585 RD=1.580
N: C=1.645 H=1.651 V=1.640 LD=1.638 RD=1.653
Actual Yo = 0.534; Calculated Yo = 0.520; % error in calculation of Yo = 2.6%

A noiseless lattice

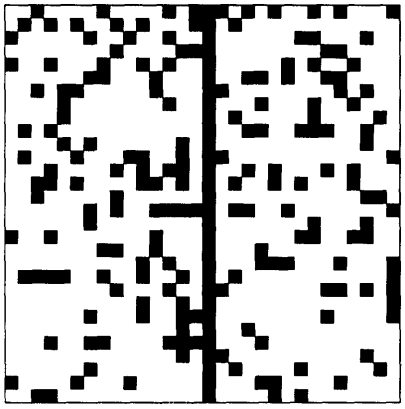


The noise filtered lattice

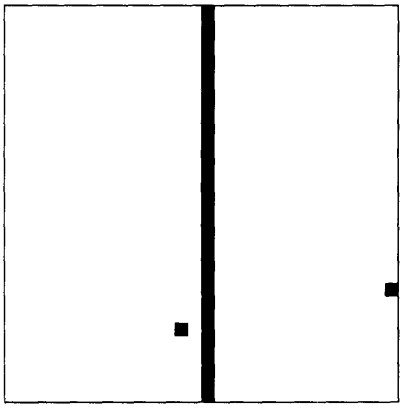


Yield = 0.290; SDNN = Clustered; SDN = Clustered
NN: C=1.878 H=1.995 V=1.838 LD=1.833 RD=1.844
N: C=1.939 H=1.997 V=1.919 LD=1.916 RD=1.922
Actual Yo = 0.290; Calculated Yo = 0.290; % error in calculation of Yo = 0.0%

A lattice with a vertical stripe

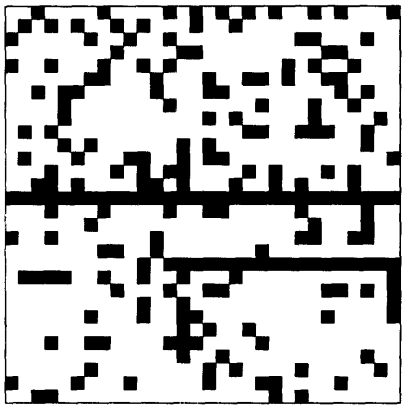


The noise filtered lattice

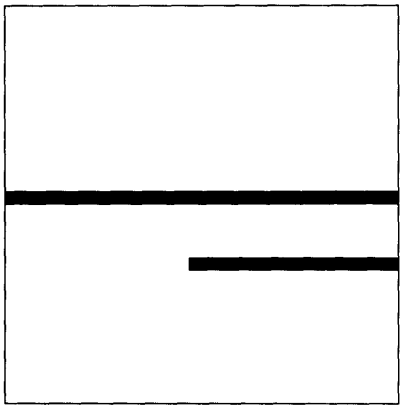


Yield = 0.761; SDNN = Random (error); SDN = Vertical
NN: C=1.033 H=1.026 V=1.171 LD=1.022 RD=1.088
N: C=1.002 H=0.965 V=1.096 LD=0.975 RD=0.971
Actual Yo = 0.964; Calculated Yo = 0.967; % error in calculation of Yo = 0.3%

A lattice with horizontal stripes



The noise filtered lattice



Yield = 0.746; SDNN = Horizontal; SDN = Horizontal
NN: C=1.032 H=1.157 V=1.073 LD=1.025 RD=1.048
N: C=1.026 H=1.121 V=0.993 LD=0.993 RD=0.998
Actual Yo = 0.949; Calculated Yo = 0.947; % error in calculation of Yo = 0.2%

Appendix B

Calculation of Average Defect Limited Yield

The wafer yield, Y, is given by the product of gross yield, Y_o, and defect limited yield, Y_r.

$$Y = Y_o \times Y_r, \tag{2.1}$$

It follows from the discussion in chapter 2 of mechanisms that cause gross and defect limited yield losses that Y_o and Y_r are independent, but not identical, random variables both distributed between 0 and 1. Hence, for a group of N wafers, both Equations 2.2 and 2.3, given below, are valid ways of calculating the average defect limited yield of the group of wafers. We use different symbols for the average defect limited yield calculated using the two ways to facilitate the following discussion.

$$\tilde{Y}_r = \frac{1}{N}\sum_{i=1}^{N}\frac{Y^i}{Y_o^i} \tag{2.2}$$

$$\overline{Y}_r = \frac{\frac{1}{N}\sum_{i=1}^{N} Y^i}{\frac{1}{N}\sum_{i=1}^{N} Y_o^i} \tag{2.3}$$

The task here is to investigate which of the two Equations, 2.2 or 2.3, produces a better estimate for the average defect limited yield. The wafer yield for each member of the group is known correctly since it is simply calculated as the ratio of the number of good chips to the total number of chips on the wafer. The Y_o value for a given wafer is calculated using the second statistical method developed in chapter 3. This method provides only an estimate of the actual value of Y_o of the wafer. We determined an empirical error upper bound of 3% on the calculation of Y_o. Let us represent the calculated Y_o as a sum of the actual Y_o and an error term ε.

$$Y_o^{calculated} = Y_o + \varepsilon \tag{2.4}$$

We make no assumptions about the properties of ε except for the empirically verified property that ε randomly takes both positive and negative values.

Now we can express the extracted defect limited wafer yield in terms of its true value and the error term ε.

$$Y_r^{calculated} = \frac{Y}{Y_o + \varepsilon} = \frac{Y}{Y_o}\left(1 - \frac{\varepsilon}{Y_o} + \left(\frac{\varepsilon}{Y_o}\right)^2 - \left(\frac{\varepsilon}{Y_o}\right)^3 + \ldots\right) \tag{2.5}$$

Since we assume ε to have an upper bound of 3% of Y_o, the higher order terms in Equation 2.4 cannot be ignored. Since Equation 2.2 first calculates Y_r values for individual wafers, $\tilde{Y}_r$ is average of Y_r terms each of which contains error of the form illustrated in Equation 2.5. Equation 2.3, on the other hand, first calculates average value of Y_o. This is a good estimate of the actual average Y_o since most of the error cancels out.

$$\sum_{i=1}^{N}\left(Y_o^i + \varepsilon^i\right) = \sum_{i=1}^{N} Y_o^i + \sum_{i=1}^{N} \varepsilon^i \sim \sum_{i=1}^{N} Y_o^i \tag{2.6}$$

Hence, we expect Equation 2.3 to produce a better estimate of the average defect limited yield of the group of wafers.

Appendix C

Implementation of the Spatial Randomness Test and the Method to Calculate Y_0 in C Programming Language

In this appendix we provide C code that implements the spatial randomness test and the method to calculate Y_0 developed in chapter 3. C code for three programs is listed at the end of this appendix. The program labeled Program 1 takes as input a given number of binary (0, 1) wafer maps, all of the same product, contained in an input file. This program divides the wafer maps in the input file into two groups, G and L. Group L wafers are deemed to possess no zero yield regions while group G wafers are deemed to possess one or more zero yield regions. The output, printed on the computer screen, consists of the number of wafer maps in each of the two groups, and average yield, $\overline{Y}$, average gross yield, $\overline{Y}_0$, and average defect limited yield, $\overline{Y}_r$, of the two groups of wafers. Also, an output file, named "ryo", is generated that contains, for each wafer map in the input file, the original wafer map, a noise filtered wafer map, yield of the wafer map, trace values calculated using the first order non-nearest neighborhood structures, the spatial dependency of the wafer map, and the Y_0 value calculated for the wafer map. The noise filtered wafer map shows any clusters of bad chips, zero yield regions, present on the wafer and trace is the statistic defined in section 3.4.2. An additional file, named "trace", is generated that is used in the determination of the parameters of the spatial randomness test. The other two programs, labeled Program 2 and Program 3, are used only for finding the parameters of the spatial randomness test.

The input file contains wafer maps in plain text with a good chip represented by a "1" and a bad chip represented by a "0". The input file format is shown in Figure C.1. The first line in the input file is the number of wafer maps in the file. The next several lines provide information on the template of the wafer maps, as explained in Figure C.1. These are fol-

lowed by wafer maps each of which is preceeded by a tag or name for the wafer map. The leftmost column of the wafer maps must leave exactly two blank spaces on the left. It is important to follow the specified input file format precisely in order to run the programs successfully. The name of the input file will be requested by Program 1 on execution after being compiled.

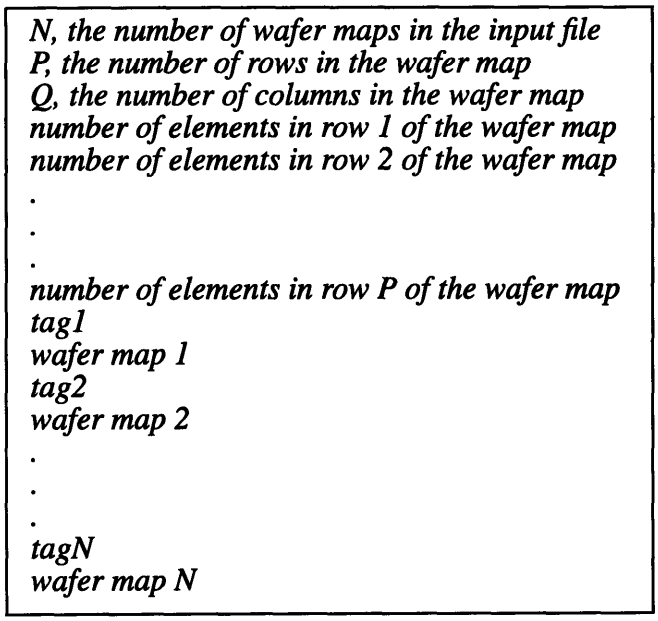


Figure C.1: Format of the input file for Program 1.

Program 1, which applies the spatial randomness test and the method to calculate Y_0 on the wafer maps contained in the input file, needs to be modified before being used for wafer maps that have a template different from the one for which the program is already being used. This modification consists of changing the bounds on the randomness band which constitute parameters of the spatial randomness test. The training procedure, given below, determines the required parameters and assumes that an input file of wafer maps in the format specified in Figure C.1 is available at the onset.

The Training Procedure

Step 1: Make a directory by the name of the product to be analyzed. Copy the three programs 1, 2, and 3 in this directory.

Step 2: Compile Program 2 and run its executable. Type the input file name in response to the program's request. An output file called "rws" will be created. The file "rws" contains 90 simulated pseudorandom binary wafers of the template specified in the input file.

Step 3: Compile Program 1 and run its executable. Type the file name "rws" in response to the program's request for an input file name. At this stage, ignore all the output written to the computer screen and the output file "ryo" created. Only the output file "trace" is required in the training procedure.

Step 4: Compile Program 3 and run its executable. Type the file name "trace" in response to the program's request for an input file name. This program will write its output to the computer screen. Consider the two lines from the output which read as follows:

"The maximum trace values are as follows
e1 e2 e3 e4 e5"

where e1, e2, e3, e4, e5 are five numbers.

Step 5: This step involves modifying Program 1. Find the comment "Needs to be changed for specific templates" in the beginning of the second page of Program 1. Replace the four variables, maxc[0], maxc[1], maxs[0], and maxs[1] as follows:

```
maxc[0] = e1 + 0.020;
maxc[1] = e1 + 0.050;
maxs[0] = (e2 + e3 + e4 + e5)/4 + 0.030;
maxs[1] = (e2 + e3 + e4 + e5)/4 + 0.070;
```

Step 6: Save Program 1. The training procedure completes here. Program 1 after compilation now can be used for all wafer maps that have the same template as the template used in the training procedure.

Program 1

```
/* The Spatial Randomness Test and the Method to Calculate Yo implemented in C */

/* This program reads in a number of wafer maps, finds whether or not a spatial
dependency is present in each wafer map and calculate its Yo accordingly. Results
for individual wafer maps are written in the output file "ryo." */

/* This program corresponds to any given template symmetric about its horizontal
axis. The information on template shape is read from the input file and stored in
the array a[P]. The entry a[i] of the array stores the number of elements in row
i of the template. The general looping is given by
   for(i=2; i<(P+2); i++)
   for(j=((Q+4-a[i])/2); j<((Q+4+a[i])/2); j++) */

/* A copy of this program can be modified to yield a program for a template of
any given shape and size. The only change needed in the program for a new tem-
plate is in the bounds on the randomness band. These bounds should correspond to
the size and shape of the new template. */

#include <stdio.h>
#include <math.h>

#define M 20

main()
{
  int i, j, k, N, r, c, n, q, nc, l, js, jf;
  int P, Q; /* wafer has P rows and Q columns */
  char wc;
  char fn[20];
  int a[100], b[100]; /* arrays which stores information on the template shape */
  char wm[100][100], wmr[100][100], wms[100][100], wmo[100][100]; /* contain
wafer maps*/
  int wmc[100][100]; /* stores 1s and 0s in the wafer map as integers */
  double m[2][2], y, tr[5], trn[5], maxc[2], maxs[2], yl, yg, yol, yog, p, pu, e,
u, pn, yn, yo, yrl, yrg, ee;
  int ones, zeros, zn, zr, sites;
  FILE *fp1, *fp2, *fp3;

  printf("\nPlease enter the input file name\n");
  gets(fn);
  puts(fn);

  e = 2.5;
  u = 0.05;
  /* e and u are parameters of the method to calculate Yo. They are supposed to
be determined by training on a known data set for a given template. However, we
determined that these parameters are quite insensitive to template variations.
For simplicity, we hardwire these values. */

  /* The randomness band for very low or very high yield wafers is larger than
the randomness band for middle yield wafers. This is because clusters of 0s in
middle yield wafers are more difficult to detect.
     maxc[0] = upper bound on the randomness band when using 16-nonnearest neigh-
bors and when 0.25 < wafer yield < 0.75.
     maxc[1] = upper bound on the randomness band when using 16-nonnearest neigh-
bors and when wafer yield < 0.25 or wafer yield > 0.75.
     maxs[0] = upper bound on the randomness band when using 2-nonnearest neigh-
```

```
bors and when 0.25 < wafer yield < 0.75.
     maxs[1] = upper bound on the randomness band when using 2-nonnearest neigh-
bors and when wafer yield < 0.25 or wafer yield > 0.75.
     Note that we only need upper bound on the randomness band since we are look-
ing only for clustering */
  maxc[0] = 1.074; /* Needs to be changed for specific templates */
  maxs[0] = 1.17925; /* Needs to be changed for specific templates */
  maxc[1] = 1.104; /* Needs to be changed for specific templates */
  maxs[1] = 1.21925; /* Needs to be changed for specific templates */

  nc = 0; /* initialize variables */
  yl = yg = 0;
  yol = yog = 0;
  yrl = yrg = 0;

  fp1=fopen(fn, "r"); /* check for errors in file opening */
  if (fp1 == NULL)
    {
      printf("\nCould not open file %s\n", fn);
      exit(1);
    }
  /* "ryo", the output file, contains results for individual wafers */
  fp2=fopen("ryo", "w"); /* check for errors in file opening */
  if (fp2 == NULL)
    {
      printf("\nCould not open file ryo\n");
      exit(2);
    }

  /* the file "trace" contains trace values for the neighborhood structures */
  fp3=fopen("trace", "w"); /* check for errors in file opening */
  if (fp3 == NULL)
    {
      printf("\nCould not open file trace\n");
      exit(3);
    }

  fscanf(fp1, "%d\n", &N); /* read in the number of wafer maps */
  fscanf(fp1, "%d\n", &P); /* read in the number of rows in the wafer */
  fscanf(fp1, "%d\n", &Q); /* read in the number of columns in the wafer */
  fprintf(fp3, "%d\n", N);

  for(i=0; i<100; i++)
    a[i] = 0; /* initialize array a */
  for(i=2; i<(P+2); i++)
    fscanf(fp1, "%d\n", &a[i]);
  for(i=0; i<(P+4); i++)
    b[i] = a[i]; /* copy array a into array b */

  for(q=0; q<N; q++) /* Start looping for N wafers in the input file */
    {
      if (fgets(fn, 20, fp1) == NULL)
        printf("\nError in reading the wafer tag\n");
      if (puts(fn) == '\0')
        printf("\nSomething wrong in reading the wafer tag\n");
      for(i=2; i<(P+2); i++)
        fgets(wmr[i], 100, fp1); /* read the wafer map in 2D array wmr */
      for(i=2; i<(P+2); i++)
        {
```

```
      strncpy(wm[i],wmr[i],100);
      strncpy(wmo[i],wmr[i],100); /* store the original wafer map in wmo to write
to ryo */
      jf = (Q+4+b[i])/2;
      wmo[i][jf] = '\0';
    }
      /* The wafer map is reflected about its edges by two sites to enable the
neighborhood operations at the edges.  The wafer map in wmr is copied to wm and
reflection about edges of wm starts here */
      strncpy(wm[0],wm[4],100); /* top row second order reflection */
      strncpy(wm[1],wm[3],100); /* top row first order reflection */
      strncpy(wm[P+3],wm[P-1],100); /* bottom row second order reflection */
      strncpy(wm[P+2],wm[P],100); /* bottom row first order reflection */
    for(i=2; i<(P+2); i++) /* vertical reflections */
    {
      js = (Q+4-a[i])/2;
      jf = (Q+4+a[i])/2;
      wm[i][js-2] = wm[i][js+2];
      wm[i][js-1] = wm[i][js+1];
      wm[i][jf+1] = wm[i][jf-3];
      wm[i][jf] = wm[i][jf-2];
    }
      for(i=0; i<(P+4); i++) /* reflections in template gaps */
        {
        if(a[i+1]>a[i]) /* upward horizontal */
        {
          if(i>1)
            {
            for(j=((Q+4-a[i+1])/2); j<((Q+4-a[i])/2); j++) /* left side */
            {
              wm[i-1][j] = wm[i+3][j];
              wm[i][j] = wm[i+2][j];
            }
            for(j=((Q+4+a[i])/2); j<((Q+4+a[i+1])/2); j++) /* right side */
            {
              wm[i-1][j] = wm[i+3][j];
              wm[i][j] = wm[i+2][j];
            }
        }
        }
        if(a[i-1]>a[i]) /* downward horizontal */
        {
          if(i<(P+2))
            {
            for(j=((Q+4-a[i-1])/2); j<((Q+4-a[i])/2); j++) /* left side */
            {
              wm[i+1][j] = wm[i-3][j];
              wm[i][j] = wm[i-2][j];
            }
            for(j=((Q+4+a[i])/2); j<((Q+4+a[i-1])/2); j++) /* right side */
            {
              wm[i+1][j] = wm[i-3][j];
              wm[i][j] = wm[i-2][j];
            }
        }
        }
    }
      for(i=0; i<(P+4); i++)
        {
```

```
    if(a[i+1]>a[i])
    {
      js = (Q+4-a[i+1])/2; /* upward left corners */
      wm[i][js-1] = wm[i+2][js+1];
      wm[i-1][js-2] = wm[i+3][js+2];
      wm[i-1][js-1] = wm[i+3][js+1];
      wm[i][js-2] = wm[i+2][js+2];
      jf = (Q+4+a[i+1])/2; /* upward right corners */
      wm[i][jf] = wm[i+2][jf-2];
      wm[i-1][jf+1] = wm[i+3][jf-3];
      wm[i-1][jf] = wm[i+3][jf-2];
      wm[i][jf+1] = wm[i+2][jf-3];
    }
    if(a[i-1]>a[i])
    {
      js = (Q+4-a[i-1])/2; /* upward left corners */
      wm[i][js-1] = wm[i-2][js+1];
      wm[i+1][js-2] = wm[i-3][js+2];
      wm[i+1][js-1] = wm[i-3][js+1];
      wm[i][js-2] = wm[i-2][js+2];
      jf = (Q+4+a[i-1])/2; /* upward right corners */
      wm[i][jf] = wm[i-2][jf-2];
      wm[i+1][jf+1] = wm[i-3][jf-3];
      wm[i+1][jf] = wm[i-3][jf-2];
      wm[i][jf+1] = wm[i-2][jf-3];
    }
}
  for(i=0; i<(P+4); i++)
    {
    if(a[i+1]>a[i])
    {
      b[i-1] = a[i+1];
      b[i] = a[i+1];
    }
}
  for(i=(P+3); i>-1; i--)
    {
    if(a[i-1]>a[i])
    {
      b[i+1] = a[i-1];
      b[i] = a[i-1];
    }
    }
  for(i=0; i<(P+4); i++)
    {
    jf = (Q+4+b[i])/2 + 2;
    wm[i][jf] = '\0';
    js = (Q+4-b[i])/2 - 2;
    for(j=0; j<js; j++)
    wm[i][j] = ' ';
    }
  /* Reflection about edges of wm finishes here */
  if (fputs(fn, fp2) == '\0') /* write the wafer tag to the output file */
printf("\n Error in writing the wafer tag to the output file\n");

  /* Wafer yield calculation */
  ones = 0;
  zeros = 0;
  for(i=0; i<(P+4); i++)
```

```
    for(j=0; j<(Q+4); j++)
      {
        if(wm[i][j]=='1')
          {
        wmc[i][j] = 1;
        wms[i][j] = '1';
        if(i>=2 && i<(P+2) && j>=((Q+4-a[i])/2) && j<((Q+4+a[i])/2))
          ones = ones + 1;
          }
        else if(wm[i][j]=='0')
          {
        wmc[i][j] = 0;
        wms[i][j] = 'r';
        if(i>=2 && i<(P+2) && j>=((Q+4-a[i])/2) && j<((Q+4+a[i])/2))
          zeros = zeros + 1;
          }
      }
      sites = ones+zeros;
      y=ones;
      y = y/(ones+zeros);


      /* The statistical methods will result in large errors if the wafer con-
sists of almost only good or almost only bad chips. Hence, we apply the method to
wafers with yields between 5% and 95%. Wafers with less than 5% yield are
regarded as consisting of only zero-yield region and wafers with yield greater
than 95% are regarded as consisting of only non-zero yield region. */

      if (y > 0.05 && y < 0.95)
      /* The aura framework based spatial randomness test begins here - applica-
tion of Thesis Method 1. */
     {
      /* Experimentation with different neighborhood structures has shown that a
first order non-nearest neighborhood structure is simple and effective enough to
be used. Nearest neighbors are not used to avoid detecting any small clustering
of chips due to the physical tendency of local defects to somewhat cluster
together */

      /* m is the 2x2 aura measure matrix */

      /* A site value is stored in int r and a neighbor value is stored in int c.
All sites are visited and the appropriate aura measures, determined by the values
of the site and its neighbor, are calculated. */

      /* Calculation of trace for 16 non-nearest neighbors begins here */
      m[0][0] = m[0][1] = m[1][0] = m[1][1] = 0;
      for(i=2; i<(P+2); i++)
        for(j=((Q+4-a[i])/2); j<((Q+4+a[i])/2); j++)
        {
        r=wmc[i][j];
        c=wmc[i-2][j-2];
        m[r][c]=m[r][c]+1;
        c=wmc[i-2][j-1];
        m[r][c]=m[r][c]+1;
        c=wmc[i-2][j];
        m[r][c]=m[r][c]+1;
        c=wmc[i-2][j+1];
        m[r][c]=m[r][c]+1;
        c=wmc[i-2][j+2];
```

```
  m[r][c]=m[r][c]+1;
  c=wmc[i-1][j-2];
  m[r][c]=m[r][c]+1;
  c=wmc[i-1][j+2];
  m[r][c]=m[r][c]+1;
  c=wmc[i][j-2];
  m[r][c]=m[r][c]+1;
  c=wmc[i][j+2];
  m[r][c]=m[r][c]+1;
  c=wmc[i+1][j-2];
  m[r][c]=m[r][c]+1;
  c=wmc[i+1][j+2];
  m[r][c]=m[r][c]+1;
  c=wmc[i+2][j-2];
  m[r][c]=m[r][c]+1;
  c=wmc[i+2][j-1];
  m[r][c]=m[r][c]+1;
  c=wmc[i+2][j];
  m[r][c]=m[r][c]+1;
  c=wmc[i+2][j+1];
  m[r][c]=m[r][c]+1;
  c=wmc[i+2][j+2];
  m[r][c]=m[r][c]+1;
  }
m[0][0] = m[0][0]/(16*zeros); /* normalization of the aura matrix */
m[0][1] = m[0][1]/(16*zeros);
m[1][0] = m[1][0]/(16*ones);
m[1][1] = m[1][1]/(16*ones);
tr[0] = m[0][0] + m[1][1]; /* trace = sum of diagonal elements */
/* Calculation of trace for 16 non-nearest neighbors finishes here */

/* Calculation of trace for 2 non-nearest H neighbors begins here */
m[0][0] = m[0][1] = m[1][0] = m[1][1] = 0;
for(i=2; i<(P+2); i++)
  for(j=((Q+4-a[i])/2); j<((Q+4+a[i])/2); j++)
  {
  r=wmc[i][j];
  c=wmc[i][j-2];
  m[r][c]=m[r][c]+1;
  c=wmc[i][j+2];
  m[r][c]=m[r][c]+1;
  }
m[0][0] = m[0][0]/(2*zeros); /* normalization of the aura matrix */
m[0][1] = m[0][1]/(2*zeros);
m[1][0] = m[1][0]/(2*ones);
m[1][1] = m[1][1]/(2*ones);
tr[1] = m[0][0] + m[1][1]; /* trace = sum of diagonal elements */
/* Calculation of trace for 2 non-nearest H neighbors finishes here */

/* Calculation of trace for 2 non-nearest V neighbors begins here */
m[0][0] = m[0][1] = m[1][0] = m[1][1] = 0;
for(i=2; i<(P+2); i++)
  for(j=((Q+4-a[i])/2); j<((Q+4+a[i])/2); j++)
  {
  r=wmc[i][j];
  c=wmc[i-2][j];
  m[r][c]=m[r][c]+1;
  c=wmc[i+2][j];
  m[r][c]=m[r][c]+1;
```

```
        }
      m[0][0] = m[0][0]/(2*zeros); /* normalization of the aura matrix */
      m[0][1] = m[0][1]/(2*zeros);
      m[1][0] = m[1][0]/(2*ones);
      m[1][1] = m[1][1]/(2*ones);
      tr[2] = m[0][0] + m[1][1]; /* trace = sum of diagonal elements */
      /* Calculation of trace for 2 non-nearest V neighbors finishes here */

      /* Calculation of trace for 2 non-nearest LD neighbors begins here */
      m[0][0] = m[0][1] = m[1][0] = m[1][1] = 0;
      for(i=2; i<(P+2); i++)
        for(j=((Q+4-a[i])/2); j<((Q+4+a[i])/2); j++)
        {
        r=wmc[i][j];
        c=wmc[i-2][j-2];
        m[r][c]=m[r][c]+1;
        c=wmc[i+2][j+2];
        m[r][c]=m[r][c]+1;
        }
      m[0][0] = m[0][0]/(2*zeros); /* normalization of the aura matrix */
      m[0][1] = m[0][1]/(2*zeros);
      m[1][0] = m[1][0]/(2*ones);
      m[1][1] = m[1][1]/(2*ones);
      tr[3] = m[0][0] + m[1][1]; /* trace = sum of diagonal elements */
      /* Calculation of trace for 2 non-nearest LD neighbors finishes here */

      /* Calculation of trace for 2 non-nearest RD neighbors begins here */
      m[0][0] = m[0][1] = m[1][0] = m[1][1] = 0;
      for(i=2; i<(P+2); i++)
        for(j=((Q+4-a[i])/2); j<((Q+4+a[i])/2); j++)
        {
        r=wmc[i][j];
        c=wmc[i-2][j+2];
        m[r][c]=m[r][c]+1;
        c=wmc[i+2][j-2];
        m[r][c]=m[r][c]+1;
        }
      m[0][0] = m[0][0]/(2*zeros); /* normalization of the aura matrix */
      m[0][1] = m[0][1]/(2*zeros);
      m[1][0] = m[1][0]/(2*ones);
      m[1][1] = m[1][1]/(2*ones);
      tr[4] = m[0][0] + m[1][1]; /* trace = sum of diagonal elements */
      /* Calculation of trace for 2 non-nearest RD neighbors finishes here */

      /* writing trace values to output file "trace" */
      if (fputs(fn, fp3) == '\0')
        printf("\n Error in writing wafer tag to file trace\n");
     fprintf(fp3, "%.3lf %.3lf %.3lf %.3lf %.3lf\n", tr[0], tr[1], tr[2], tr[3],
tr[4]);

      if(y<0.25 || y>0.75) /* choosing the appropriate randomness band depending
on yield */
        l = 1;
      else
        l = 0;

      /* The trace values corresponding to all of the five neighborhoods are
stored in the array tr. The wafer is classified as possessing one of the five
dependencies, Clustered, Horizontal, Vertical, Left Diagonal, and Right Diagonal
```

```
or no dependency, Random. If the trace for the 16 non-nearest neighbors exceed
its threshold or if trace values for any two direction subneighborhoods exceed
their thresholds simultaneously, the wafer is assigned Clustered. If trace for
only one direction subneighborhood exceeds its threshold, the corresponding
dependency is chosen. If no trace exceeds its threshold, the wafer is assigned
Random. */

      if(tr[0] > maxc[l]) /* Clustered */
        {
          wc = 'c';
          nc = nc +1;
          yg = yg + y;
        }
      else if((tr[1]>maxs[l] && tr[2]>maxs[l]) || (tr[1]>maxs[l] &&
tr[3]>maxs[l]) || (tr[1]>maxs[l] && tr[4]>maxs[l]) || (tr[2]>maxs[l] &&
tr[3]>maxs[l]) || (tr[2]>maxs[l] && tr[4]>maxs[l]) || (tr[3]>maxs[l] &&
tr[4]>maxs[l]))
        {
          wc = 'c'; /* Clustered */
          nc = nc +1;
          yg = yg + y;
        }
      else if(tr[1] > maxs[l])
        {
          wc = 'h'; /* Horizontal */
          nc = nc +1;
          yg = yg + y;
        }
      else if(tr[2] > maxs[l])
        {
          wc = 'v'; /* Vertical */
          nc = nc +1;
          yg = yg + y;
        }
      else if(tr[3] > maxs[l])
        {
          wc = 'l'; /* Left diagonal */
          nc = nc +1;
          yg = yg + y;
        }
      else if(tr[4] > maxs[l])
        {
          wc = 'r'; /* Right diagonal */
          nc = nc +1;
          yg = yg + y;
        }
      else
        {
          wc = 'R'; /* no spatial dependency - Random */
          yl = yl + y;
        }

      /* At this stage the wafer has already been classified as either random or
possessing one or more zero yield regions. The calculation of Yo begins here -
application of Thesis Method 2 */

      p=y; /* step 1. set p equal to wafer yield */
      pu=0; /* pu is the p value in previous step */
```

```
      /* The neighborhood structure for doing hypothesis testing on neighbors of
each bad chip is determined by the outcome of the spatial randomness test. A sec-
ond order (24) nearest neighborhood structure is used if the wafer has no partic-
ular directional dependency. Otherwise, a second order nearest structure in the
same direction as indicated by the spatial randomness test is used */
      if (wc == 'c' || wc == 'R') /* use 24 nearest neighbors */
        for(k=0; (k<M && (p-pu)>u); k++) /* iterate further only if convergence
did not take place or if no. of iterations is less than M */
        {
        pu = p; /* record the value of new p */
        ee = e*sqrt(p*(1-p)/24);
        if ((p-ee) > 0) /* set the threshold */
          pn = p-ee;
        else if(wc == 'c')
          pn = p-ee/10; /* the threshold is lower for low yields */
        else
          pn = p/10;
        zn = zr = 0; /* initiate the number of n and r zeros */
        for(i=2; i<(P+2); i++)
          for(j=((Q+4-a[i])/2); j<((Q+4+a[i])/2); j++)
            {
            yn = 0; /* initiate neighborhood yield */
            if(wms[i][j]!='1') /* visit only zero sites */
            {
            /* yn is incremented if the neighbor value of the site is non-zero */
              if (wms[i][j-1] == '1')
                yn = yn+1;
              if (wms[i][j+1] == '1')
                yn = yn+1;
              if (wms[i-1][j] == '1')
                yn = yn+1;
              if (wms[i+1][j] == '1')
                yn = yn+1;
              if (wms[i-1][j-1] == '1')
                yn = yn+1;
              if (wms[i-1][j+1] == '1')
                yn = yn+1;
              if (wms[i+1][j-1] == '1')
                yn = yn+1;
              if (wms[i+1][j+1] == '1')
                yn = yn+1;
              if (wms[i-2][j-2] == '1')
                yn = yn+1;
              if (wms[i-2][j-1] == '1')
                yn = yn+1;
              if (wms[i-2][j] == '1')
                yn = yn+1;
              if (wms[i-2][j+1] == '1')
                yn = yn+1;
              if (wms[i-2][j+2] == '1')
                yn = yn+1;
              if (wms[i-1][j-2] == '1')
                yn = yn+1;
              if (wms[i-1][j+2] == '1')
                yn = yn+1;
              if (wms[i][j-2] == '1')
                yn = yn+1;
              if (wms[i][j+2] == '1')
                yn = yn+1;
```

```
        if (wms[i+1][j-2] == '1')
          yn = yn+1;
        if (wms[i+1][j+2] == '1')
          yn = yn+1;
        if (wms[i+2][j-2] == '1')
          yn = yn+1;
        if (wms[i+2][j-1] == '1')
          yn = yn+1;
        if (wms[i+2][j] == '1')
          yn = yn+1;
        if (wms[i+2][j+1] == '1')
          yn = yn+1;
        if (wms[i+2][j+2] == '1')
          yn = yn+1;
        yn = yn / 24; /* calculate neighborhood yield from the sum */
        if (yn<pn) /* compare the neighborhood yield with threshold */
          {
           wms[i][j] = 'n'; /* 0 classified to zero yield (n) region */
           zn = zn + 1;
          }
        else
          {
           wms[i][j] = 'r'; /* 0 classified to non-zero yield (r) region*/
           zr = zr + 1;
          }
      }
      }
  if ((zn+zr)!=zeros) /* a check */
    printf("Error, there is a problem in zero class assignments\n");
  else
    {
      p = ones;
      p = p / (sites - zn); /* update p */
    }
    }

else if (wc == 'h')
  /* Use a 4-nearest horizontal neighborhood structure */
  for(k=0; (k<M && (p-pu)>u); k++)
  {
  pu = p; /* record the value of new p */
  ee = e*sqrt(p*(1-p)/4);
  if ((p-ee) > 0) /* set the threshold */
    pn = p-ee;
  else
    pn = p-ee/10;

  zn = zr = 0;
  for(i=2; i<(P+2); i++)
    for(j=((Q+4-a[i])/2); j<((Q+4+a[i])/2); j++)
      {
      yn = 0;
      if(wms[i][j]!='1')
      {
        if (wms[i][j-1] == '1')
          yn = yn+1;
        if (wms[i][j+1] == '1')
          yn = yn+1;
        if (wms[i][j-2] == '1')
```

```
          yn = yn+1;
        if (wms[i][j+2] == '1')
          yn = yn+1;
        yn = yn / 4;
        if (yn<pn)
          {
            wms[i][j] = 'n';
            zn = zn + 1;
          }
        else
          {
            wms[i][j] = 'r';
            zr = zr + 1;
          }
      }
      }
  if ((zn+zr)!=zeros) /* a check */
    printf("Error there is a problem in zero class assignments\n");
  else
    {
      p = ones;
      p = p / (sites - zn); /* update p */
    }
    }

else if (wc == 'v')
  /* Use a 4-nearest vertical neighborhood structure */
  for(k=0; (k<M && (p-pu)>u); k++)
  {
  pu = p; /* record the value of new p */
  ee = e*sqrt(p*(1-p)/4);
  if ((p-ee) > 0) /* set the threshold */
    pn = p-ee;
  else
    pn = p-ee/10;
  zn = zr = 0;
  for(i=2; i<(P+2); i++)
    for(j=((Q+4-a[i])/2); j<((Q+4+a[i])/2); j++)
      {
      yn = 0;
      if(wms[i][j]!='1')
      {
        if (wms[i-2][j] == '1')
          yn = yn+1;
        if (wms[i-1][j] == '1')
          yn = yn+1;
        if (wms[i+1][j] == '1')
          yn = yn+1;
        if (wms[i+2][j] == '1')
          yn = yn+1;
        yn = yn / 4;
        if (yn<pn)
          {
            wms[i][j] = 'n';
            zn = zn + 1;
          }
        else
          {
            wms[i][j] = 'r';
```

```
            zr = zr + 1;
          }
        }
        }
      if ((zn+zr)!=zeros) /* a check */
        printf("Error there is a problem in zero class assignments\n");
      else
    {
      p = ones;
      p = p / (sites - zn); /* update p */
    }
      }

else if (wc == 'l')
  /* Use a 4-nearest left diagonal neighborhood structure */
  for(k=0; (k<M && (p-pu)>u); k++)
    {
  pu = p; /* record the value of new p */
  ee = e*sqrt(p*(1-p)/4);
  if ((p-ee) > 0) /* set the threshold */
    pn = p-ee;
  else
    pn = p-ee/10;
  zn = zr = 0;
  for(i=2; i<(P+2); i++)
    for(j=((Q+4-a[i])/2); j<((Q+4+a[i])/2); j++)
      {
      yn = 0;
      if(wms[i][j]!='1')
      {
        if (wms[i-2][j-2] == '1')
          yn = yn+1;
        if (wms[i-1][j-1] == '1')
          yn = yn+1;
        if (wms[i+1][j+1] == '1')
          yn = yn+1;
        if (wms[i+2][j+2] == '1')
          yn = yn+1;
        yn = yn / 4;
        if (yn<pn)
          {
            wms[i][j] = 'n';
            zn = zn + 1;
          }
        else
          {
            wms[i][j] = 'r';
            zr = zr + 1;
          }
      }
      }
  if ((zn+zr)!=zeros) /* a check */
    printf("Error there is a problem in zero class assignments\n");
  else
    {
      p = ones;
      p = p / (sites - zn); /* update p */
    }
    }
```

```
else if (wc == 'r')
  /* Use a 4-nearest right diagonal neighborhood structure */
  for(k=0; (k<M && (p-pu)>u); k++)
    {
  pu = p; /* record the value of new p */
  ee = e*sqrt(p*(1-p)/4);
  if ((p-ee) > 0) /* set the threshold */
    pn = p-ee;
  else
    pn = p-ee/10;
  zn = zr = 0;
  for(i=2; i<(P+2); i++)
    for(j=((Q+4-a[i])/2); j<((Q+4+a[i])/2); j++)
      {
      yn = 0;
      if(wms[i][j]!='1')
      {
        if (wms[i-2][j+2] == '1')
          yn = yn+1;
        if (wms[i-1][j+1] == '1')
          yn = yn+1;
        if (wms[i+1][j-1] == '1')
          yn = yn+1;
        if (wms[i+2][j-2] == '1')
          yn = yn+1;
        yn = yn / 4;
        if (yn<pn)
          {
            wms[i][j] = 'n';
            zn = zn + 1;
          }
        else
          {
            wms[i][j] = 'r';
            zr = zr + 1;
          }
      }
      }
  if ((zn+zr)!=zeros) /* a check */
    printf("Error there is a problem in zero class assignments\n");
  else
    {
      p = ones;
      p = p / (sites - zn); /* update p */
    }
    }
yo = zn;
yo = 1 - yo/sites; /* calculate Yo */
if (wc == 'R') /* update the total yield for the L group */
  {
    yol = yol + yo;
    yrl = yrl + y/yo;
  }
else /* update the total yield for the G group */
  {
    yog = yog + yo;
    yrg = yrg + y/yo;
  }
```

```
      /* Mark out the cluster of 0s by printing only those 0s that belong to zero
yield (n) regions and those 1s that are inside zero yield regions as 0s - all
other sites are printed as 1s */
      zn = zr = 0;
      for(i=2; i<(P+2); i++)
        for(j=((Q+4-a[i])/2); j<((Q+4+a[i])/2); j++)
          {
        if(wms[i][j]=='n') /* n 0s are marked 0s */
          {
            wmr[i][j]='0';
            zn = zn + 1;
          }
        else if (wms[i][j] == '1') /* finding 1s surrounded by n 0s */
          {
            c=0;
            if (wms[i-1][j-1] == 'n')
              c=c+1;
            if (wms[i-1][j] == 'n')
              c=c+1;
            if (wms[i-1][j+1] == 'n')
              c=c+1;
            if (wms[i][j-1] == 'n')
              c=c+1;
            if (wms[i][j+1] == 'n')
              c=c+1;
            if (wms[i+1][j-1] == 'n')
              c=c+1;
            if (wms[i+1][j] == 'n')
              c=c+1;
            if (wms[i+1][j+1] == 'n')
              c=c+1;
            if (c>4)
              { /* the 1 is inside a zero yield region */
            wmr[i][j]='0';
            zn = zn + 1;
              }
            else /* the 1 is not inside a zero yield region */
              wmr[i][j]='1';
          }
        else
          wmr[i][j]='1';
        }

      /* printing results for the wafer map to the output file, "ryo" */
      fprintf(fp2,"\n  The Original Lattice\t\t\t");
      fprintf(fp2,"  The Noise Filtered Lattice\n");
      for(i=2; i<(P+2); i++)
        {
          if (fputs(wmo[i], fp2) == '\0')
            printf("\n Error in printing a row of the wafer map\n");
          fprintf(fp2,"\t");
          if (fputs(wmr[i], fp2) == '\0')
            printf("\n Error in printing the noise-filtered wafer map\n");
        }

      fprintf(fp2,"\n\nOverall Lattice Yield = %.3lf\n\n", y);
      fprintf(fp2,"Trace values based on Non-Nearest Neighbors\n");
      fprintf(fp2,"C=%.3lf H=%.3lf V=%.3lf LD=%.3lf RD=%.3lf\n\n", tr[0], tr[1],
```

```
tr[2], tr[3], tr[4]);
      fprintf(fp2,"Spatial Dependency based on Non-Nearest Neighbors = %c\n",
wc);
      fprintf(fp2,"Calculated Yo \t= %.3lf\n", yo);

      } /* N looping finishes here */

      else if(y<0.05) /* the non-yielding wafer is assigned to G group */
    {
      fprintf(fp2,"Wafer yield close to 0\n");
      fprintf(fp2,"Entire wafer considered zero yield - no noise filtering
required\n");
      fprintf(fp2,"^L\n");
      yg = yg + y;
      yog = yog + y;
      yrg= yrg + y;
      nc = nc + 1;
    }
    else /* the high yielding wafer is assigned to L group */
    {
      fprintf(fp2,"Wafer yield close to 1\n");
      fprintf(fp2,"Entire wafer considered non-zero yield - no noise filtering
required\n");
      fprintf(fp2,"^L\n");
      yl = yl + y;
      yol = yol + 1;
      yrl = yrl + y;
    }
    }

  /* The Overall Results */
  printf("\nThe program divided the %d wafers in the input file into two groups,
G and L. Group L wafers are deemed to possess no zero yield regions while group G
wafers are deemed to possess one or more zero yield regions.\n", N);
  printf("N(L) = %d, N(G) = %d\n", N-nc, nc);
  printf("Average Y(L) = %.3lf, Average Y(G) = %.3lf\n", yl/(N-nc), yg/nc);
  printf("Average Yo(L) = %.3lf, Average Yo(G) = %.3lf\n", yol/(N-nc), yog/nc);
  printf("Average Yr(L) = %.3lf, Average Yr(G) = %.3lf\n", yl/(N-nc), yg/yog);
  fclose(fp1);
  fclose(fp2);
  fclose(fp3);
}
```

Program 2

```
/* Program to generate pseudorandom binary (0, 1) wafers */
 /* The output file contains 90 psuedorandom wafers. These consist of nine
groups, 1<= i(integer) <=9, such that each location of a wafer in ith group takes
the value 1 with an independent and identical probability = i/10; it takes value
0 with probability 1-i/10. Hence expected yield of a wafer in the ith group is i/
10. */

#include <stdio.h>
#include <math.h>
#define MAXSIZE1 100
#define MAXSIZE2 20

main()
{
  int i, j, k, N, r, c, n, q, js, jf, count;
  int P, Q; /* wafer has P rows and Q columns */
  int a[100]; /* wafer has a[i] chips in row i */
  char wm[100][100]; /* contains the wafer map */
  char fn[20];
  double p;
  FILE *fp1, *fp2;

  double srandom();

  printf("\nPlease enter the input file name \n");
  gets(fn);

  fp1=fopen(fn, "r"); /* check for errors in file opening */
  if (fp1 == NULL)
    {
      printf("\nCould not open file %s\n", fn);
      exit(1);
    }
  /* "rws", the output file, contains the simulated pseudorandom wafers */
  fp2=fopen("rws", "w"); /* check for errors in file opening */
  if (fp2 == NULL)
    {
      printf("\nCould not open file rws\n");
      exit(2);
    }

  fscanf(fp1, "%d\n", &N); /* read in the number of wafermaps in input file*/
  fscanf(fp1, "%d\n", &P); /* read in the number of rows in the wafer */
  fscanf(fp1, "%d\n", &Q); /* read in the number of columns in the wafer */

  for(i=0; i<100; i++)
    a[i] = 0; /* initialize array a */
  for(i=2; i<(P+2); i++)
    fscanf(fp1, "%d\n", &a[i]);

  fprintf(fp2, "90\n"); /* write the number of wafermaps */
  fprintf(fp2, "%d\n", P); /* write the number of rows in the wafer */
  fprintf(fp2, "%d\n", Q); /* write the number of columns in the wafer */
  for(i=2; i<(P+2); i++)
    fprintf(fp2, "%d\n", a[i]);
  count = 0;
  for(k=1; k<10; k++) /* generate 9 groups of pseudorandom wafers */
```

```
    {
      p = k;
      p = p/10;
      for(r=0; r<10; r++) /* each group has ten wafers */
    {
      count = count + 1;
      srandom(k*10+r);

      for(i=2; i<(P+2); i++)
        {
          for(j=0; j<((Q+4+a[i])/2); j++)
            wm[i][j] = ' ';
          jf = (Q+4+a[i])/2;
          wm[i][jf] = '\n';
        }

      for(i=2; i<(P+2); i++)
        for(j=((Q+4-a[i])/2); j<((Q+4+a[i])/2); j++)
          {
        if((random()/(pow(2,31)-1)) <= p)
          wm[i][j] = '1';
        else
          wm[i][j] = '0';
          }
      fputc('r', fp2);
      fprintf(fp2,"%d\n", count);
      for(i=2; i<(P+2); i++)
        for(j=0; j<=((Q+4+a[i])/2); j++)
          if (fputc(wm[i][j], fp2) == '\0')
            printf("\n Error1\n");
    }
    }

  fclose(fp1);
  fclose(fp2);
}
```

Program 3

```
/* This program finds bounds on the randomness band for the five neighborhood
structures */
#include <stdio.h>
#include <math.h>

main()
{
  int i, j, k, N;
  char fn[20];
  double read[5], min[5], max[5], y;
  FILE *fp1;

  printf("\nPlease enter the input file name\n");
  gets(fn);
  fp1=fopen(fn, "r"); /* check for errors in file opening */
  if (fp1 == NULL)
    {
      printf("\nCould not open file %s\n", fn);
      exit(1);
    }
  fscanf(fp1,"%d\n",&N);

  if (fgets(fn, 20, fp1) == NULL)
    printf("\nError in reading the wafer tag\n");
  for(i=0;i<5;i++)
    {
      fscanf(fp1,"%lf\n",&read[i]);
      max[i] = read[i];
      min[i] = read[i];
    }
  for(j=0; j<(N-1); j++)
    {
      if (fgets(fn, 20, fp1) == NULL)
        printf("\nError in reading the wafer tag\n");
      for(i=0;i<5;i++)
    {
      fscanf(fp1,"%lf\n",&read[i]);
      if(read[i] < min[i])
        min[i] = read[i];
      if(read[i] > max[i])
        max[i] = read[i];
    }
    }

  printf("\nThe minimum trace values are as follows\n");
  for(i=0;i<5;i++)
    {
      printf("%.3lf ",min[i]);
    }
  printf("\nThe maximum trace values are as follows\n");
  for(i=0;i<5;i++)
    {
      printf("%.3lf ",max[i]);
    }
  printf("\n");
  fclose(fp1);
}
```

References

[1] Stapper C. H., Armstrong F. M., Saji K., Integrated Circuit Yield Statistics, *Proceedings of the IEEE*, vol. 71, no. 4, pp 453-469, April 1983.

[2] Stapper C. H., Fact and Fiction in Yield Modeling, *Microelectronics Journal*, vol. 20, nos. 1-2, pp 129-151, 1989.

[3] Stapper C. H., LSI Yield Modeling and Process Monitoring, *IBM Journal of Research and Development*, pp 228-233, May 1976.

[4] Cunningham J. A., The Use and Evaluation of Yield Models in Integrated Circuit Manufacturing, *IEEE Transactions on Semiconductor Manufacturing*, vol. 3, no. 2, pp 60-71, May 1990.

[5] Jaeger R. C., *Introduction to Microelectronic Fabrication*, Modular Series on Solid State Devices, Addison-Wesley Publishing Company.

[6] Maly W., Strojwas A. J., Director S. W., VLSI Yield Prediction and Estimation: A Unified Framework, *IEEE Transactions on Computer-Aided Design*, vol. CAD-5, no. 1, pp 114-130, January 1986.

[7] Pineda de Gyvez J., *Integrated Circuit Defect-Sensitivity: Theory and Computational Models*, The Kluwer International Series in Engineering and Computer Science, Kluwer Academic Publishers, 1993.

[8] Murphy B. T., Cost-Size Optima of Monolithic Integrated Circuits, *Proceedings of the IEEE*, vol. 52, pp. 1537-1545, December 1964.

[9] Ferris-Prabhu A. V., *Introduction to Semiconductor Device Yield Modeling*, Artech House, 1992.

[10] Ferris-Prabhu A. V., Modeling the Critical Area in Yield Forecasts, *IEEE Journal of Solid State Circuits,* vol. SC-20, no.4, pp 874-878, August 1985.

[11] Ferris-Prabhu A. V., Defect Size Variations and their Effects on the Critical Area, *IEEE Journal of Solid State Circuits*, vol. SC-20, no.4, pp 878-880, August 1985.

[12] Ham W. E., Yield-Area Analysis: Part I-A Diagnostic Tool for Fundamental Integrated-Circuit Process Problems. *RCA Review*, vol. 39, pp 231-249, June 1978.

[13] Paz O., Lawson Jr. T. R., Modification of Poisson Statistics: Modeling Defects Induced by Diffusion, *IEEE Journal of Solid State Circuits*, vol. SC-12, no. 5, pp 540-546, 1977.

[14] Flack V. F., Introducing Dependency into IC Yield Models, *Solid State Electronics*, vol. 28, no. 6, pp 555-559, 1985.

[15] Filliben J. J., Kafader K., Shier D. R., Testing for Homogeneity of Two-Dimensional Surfaces, *Mathematical Modeling*, vol. 4, pp 167-189, 1983.

[16] Jain A. K., Dubes R. C., *Algorithms for Clustering Data*, Prentice Hall Advanced Reference Series, 1988.

[17] Lim J. S., *Two-dimensional Signal and Image Processing*, Prentice Hall, 1989.

[18] Drake A. W., *Fundamentals of Applied Probability Theory*, McGraw Hill Publishing Company, 1967.

[19] Therrien C. W., *Decision Estimation and Classification: An Introduction to Pattern Recognition and Related Topics*, John Wiley & Sons, 1989.

[20] Bartlett M. S., *The Statistical Analysis of Spatial Patterns*, John Wiley and Sons Inc, New York, 1975.

[21] Besag J. E., Nearest-Neighbor Systems and the Auto-Logistic Model for Binary Data. *Journal of the Royal Statistical Society*, vol. 34, pp 75-83, 1972.

[22] Besag J. E., Spatial Interaction and the Statistical Analysis of Lattice Systems. *Journal of the Royal Statistical Society*, vol. 36, pp 192-236, 1974.

[23] Longtin M. D., *Sequential Screening in Semiconductor Manufacturing: Exploiting Spatial Dependence*, June 1992, SM Thesis, MIT.

[24] Elfadel I. M., Picard R. W., Gibbs Random Fields, Cooccurrences, and Texture Modeling, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-16, no. 1, pp 24-37, January 1994.

[25] Picard R. W., *Texture Modeling: Temperature Effects on Markov/Gibbs Random Fields*, June 1991, DSC Thesis, MIT.

[26] Elfadel I. M., Picard R. W., Auras Part I: Theory and Relation to Morphology and Cooccurrence. *Vision and Modeling Group TR #149*, MIT Media Laboratory, Feb. 1991.

[27] Elfadel I. M., Picard R. W., Miscibility Matrices Explain the Behavior of Greyscale Textures Generated by Gibbs Random Fields. *In Proc. SPIE Conference on Intelligent Robots and Computer Vision,* Boston, MA, Nov. 1990.

[28] Davis L. S., John S. A., Aggarwal J. K., Texture Analysis Using Generalized Co-occurrence Matrices, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 3, pp 251-259, July 1979.

[29] Geman S., Geman D., Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 6, pp 721-741, November 1984.

[30] Derin H., Elliot H., Cristi R., Geman D., Bayes Smoothing Algorithms for Segmentation of Binary Images Modeled by Markov Random Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 6, pp 707-720, November 1984.

[31] Bouman C., Liu B., Multiple Resolution Segmentation of Texture Images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-13, no. 2, pp 99-113, Febuary 1991.