Order Number 1349457

Comparison of Monte Carlo and analytic critical area calculation

Lee, Li-Chyn, M.S.

The University of Arizona, 1992

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

Comparison of Monte Carlo and Analytic Critical Area Calculation

by

Li-Chyn Lee

A Thesis Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements

For the Degree of

MASTER OF SCIENCE

WITH A MAJOR IN ELECTRICAL ENGINEERING

In the Graduate College

THE UNIVERSITY OF ARIZONA

1 9 9 2

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: Li-Chyn Lee

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

Harold G. Parks

Harold G. Parks
Associate Professor of
Electrical and Computer Engineering

7/21/92
Date

ACKNOWLEDGMENTS

Working on my thesis was a very interesting and satisfying experience that was made possible by the advice and support of a number of people. First of all, I would like to thank Dr. Harold G. Parks for his leadership, encouragement, and technical advice that made this work possible. I am also grateful to my thesis committee members, Dr. Scott E. Beck and Dr. Ronald D. Schrimpf, for their constructive suggestions.

Finally, I would like to thank my family, , friends, and colleagues for their moral support in bringing all my work together.

To my parents

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Since the profitability of VLSI industries is related to yield, the IC manufacturer finds it highly desirable to be able to predict the yield by computer-aided methods. A key part in the procedure to obtain yield by computer simulation is to find the critical area of a layout. This thesis is primarily devoted to the calculations of critical area. The so called critical area is that part of the layout where defects can cause a malfunction of the IC operation. There are two techniques to find the critical area. In the first technique, an analytic method is used to analyze the circuit geometry in order to find the critical area. In the second technique a Monte Carlo Method is used. A program using this Monte Carlo yield simulation (the main method used in this thesis) has been developed for determining critical area of the metal layer of a 4K random access memory. The analytic method is used in a supporting way. The thesis also proposes an easy method to process the vast amount of layout database. This method reduces the time consumed by Monte Carlo simulation.

CHAPTER 1

Introduction

Due to inherent fluctuations in the VLSI manufacturing process, the yield is always less than 100 percent. Since the profitability of VLSI industries is related to yield, the IC manufacturer finds it highly desirable to be able to predict the yield by computer-aided methods. This ability to predict yield allows manufacturers to maximize yield through improving design rules and control of the manufacturing process.

Defects in the fabrication process can be classified as local and global. The local defects cause local deformations of circuit topology. These deformations include a bridge between two patterns, which produce a short circuit, or a defect in a pattern which results in an open circuit or a pinhole between two layers which in turn causes a short between photolithographic levels. Gross defects cause global deformations on a chip which include layer misregistration and line width variation. Generally, gross defects affect parametrics resulting in variations in chip speed and power dissipation. Local defects, on the other hand, cause malfunction of a chip. In yield prediction, there is similarity in the procedures for finding yield based on short circuits, open circuits, and pinholes. In fact, there is a duality between open and short circuit models for critical area. In this thesis, yield prediction is only based on defects which cause short circuits for simplicity of presentation.

A key part of the procedure to obtain yield is to find the critical area of a layout. This thesis is primarily devoted to the calculation of critical area. The so called critical area is that part of the layout where defects can cause a malfunction of the

IC operation. There are two techniques to find the critical area. In the first technique, an analytic method is used to analyze the circuit geometry in order to find the critical area. In the second technique a Monte Carlo Method is used. In essence, Monte Carlo yield simulation places defects on a layout and then analyzes the influence caused by these defects. For each iteration of the Monte Carlo loop a defect is positioned on the layout, and the polygons surrounding this defect are checked in order to find out whether this defect causes a circuit failure or not. A program using this Monte Carlo yield simulation (the main method used in this thesis) has been developed for determining critical area of the metal layer of a layout. These two methods of finding the critical area of a layout will be discussed further in section 2.3.2. Two other terms used frequently in this thesis are the fault probability, $P(x)$, defined as the probability that a defect of size x will cause a fault (cause a chip to fail functionally) and the defect size distribution, $h(x)$, which is a normalized distribution function of defect size and has already been determined. The critical area of a circuit is obtained by integrating $P(x)$ and $h(x)$ with respect to defect size x, and hence how much $P(x)$ contributes to the critical area is greatly influenced by $h(x)$.

For VLSI yield simulation, one frequently has to deal with the database of a layout in order to calculate the yield, critical area, fault probability, etc. The database of a VLSI layout is tremendously large. A further complication is that most commercial layout software process data hierarchically, which means elementary cells contain basic patterns and data of a layout, and the database of the layout is nothing more than coordinates of elementary cells. This hierarchical layout structure saves a lot of disk space. However, it is time-consuming to smash this hierarchical structure in order to obtain pattern data for the whole chip rather than coordinates of elementary cells. The work presented in this thesis proposes an easy method to process the vast amount of data when calculating the fault probability for short circuit defects by using a Monte Carlo simulation. The usefulness of the method is demonstrated with

computer simulation using the metal layer of a chip. The critical area for short circuit defects for this layer is then calculated using this fault probability. Random defects were generated for the Monte Carlo simulation with a random number generator and placed on the layout to estimate the fault probability. The randomness of defect placement and number of defects for accurately estimating the fault probability and critical area calculations is investigated and discussed.

The outline of this thesis is as follows :

- **Chapter 1 Introduction**
- **Chapter 2 Background** - Section 1 introduces the yield model and the formula dealing with the layout database. Two methods of finding critical area are described in section 2, including the analytic method, where equations cited from previous works are introduced, and the Monte Carlo method, which is the one used in this thesis.
- **Chapter 3 Experiment** - The layout used for this thesis and the concept of smashing the hierarchical structure are discussed in section 1. Section 2 describes how to determine the proper number of defects to be placed on the layout. Section 3 explains the theorem of random number generation and a method to evaluate the randomness of the random numbers. In section 4, the procedures for proving the formula proposed in Chapter 2 and procedures for finding the critical area are briefly discussed.
- **Chapter 4 Database and Programs** - In this chapter, the database of LED (layout editor for VLSI layout design) and programs for determining fault-causing defects and defect placement, are described. The defect placement program puts defects on the layout in order to see in the LED the relation between defects and polygons.

- **Chapter 5 Result and Conclusion** - This chapter is divided into four sections. Section 1 contains the proof of formula proposed in Chapter 2. Section 2 is the further refinement which describes how compensation is made to minimize the error and how the fault probability is calculated with the truncated cell. Section 3 explains how to calculate the fault probability of the whole chip by use of partitioned fault probabilities of reduced areas. In section 4, the method of finding critical area using the data in section 3 and a linear regression is described.

CHAPTER 2

Background

2.1 Introduction

In this chapter we examine some previous work to establish the reader's background. These works include defect statistics, and the two main methods used to calculate the critical area in this thesis.

2.2 Theoretical Background

Stapper [1] uses a gamma distribution for defects which results in the well known "negative binomial yield model" :

$$Y = \left(\frac{1}{1 + \frac{A_c \overline{D}}{\alpha}} \right)^{\alpha}, \tag{2.1}$$

where A_c is the critical area and $\overline{D}$ is the average defect density. The parameter α relates to the variance of the gamma distribution. The critical area is defined as the area within which a defect occurs and causes a fault. In order to find the critical area A_c one must consider the defect density size distribution, D(x), and the average number of faults, $\overline{\lambda}$, caused by defects. The defect density size distribution, shown in Figure 2.1, has been determined by a number of people and has the units of defects per unit area [2], [4]. It is possible to relate D(x) to a probability distribution function $h(x)$ by

$$D(x) = \overline{D} h(x), \tag{2.2}$$

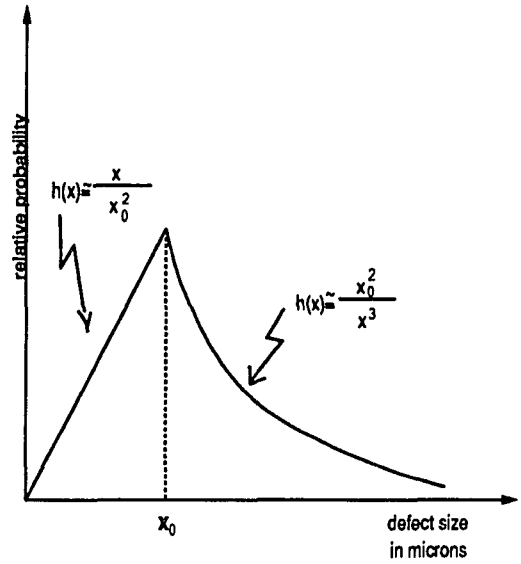


Figure 2.1: Defect size distribution. (x_0 is the resolution limit of the lithograph or measurement apparatus.)

where $h(x)$ is interpreted as the probability of a defect of size x occuring and $\overline{D}$ is given by :

$$\overline{D} = \int_0^\infty D(x)\,dx. \tag{2.3}$$

This, of course, implies that the normalization of $h(x)$ is

$$\int_0^\infty h(x)\,dx = 1. \tag{2.4}$$

The average number of faults ($\overline{\lambda}$) resulting from these defects can now be calculated by , [2] :

$$\overline{\lambda} = A_c\overline{D} \tag{2.5}$$

In order to evaluate the critical area, the fault probability, $P(x)$, defined as the probability that a defect of size x will cause a fault, must be considered. $P(x)$ is also called probability of failure [3] and relates to the critical area as a function of defect size :

$$P(x) = \frac{A_c(x)}{A} = \frac{\text{defects of size } x \text{ causing a fault}}{\text{total number defects of size } x} = \frac{\lambda(x)}{D(x)A}, \tag{2.6}$$

where A is the total chip area. Hence, the average number of circuit failures is given by,

$$\overline{\lambda} = \int_0^\infty \lambda(x)\,dx = \int_0^\infty P(x)D(x)A\,dx = A\overline{D}\int_0^\infty P(x)h(x)\,dx \tag{2.7}$$

Comparing eq. (2.5) and (2.7) the critical area is given by,

$$A_c = A\int_0^\infty P(x)h(x)\,dx. \tag{2.8}$$

As long as $P(x)$ is known, one can calculate A_c by eq. (2.8) for a given defect size distribution. Defining an average fault probability $\overline{P}$ as,

$$\overline{P} = \int_0^\infty P(x)h(x)\,dx \tag{2.9}$$

leads to

$$\overline{P} = \frac{A_c}{A}. \tag{2.10}$$

This thesis investigates a technique to calculate the fault probability by dealing with the data of elementary cells (basic patterns of the whole layout) instead of the database for the whole chip. Mathematically, this is expressed as :

$$P_{ther}(x) = \sum_{n=1}^{N_T} P_{cell_n}(x)Ratio(cell_n) \tag{2.11}$$

where :

$P_{ther}(x)$: The theoretical fault probability (defect size x) of the whole chip.

$P_{cell_n}(x)$: The fault probability (defect size x) of an elementary $cell_n$.

$Ratio(cell_n)$: The ratio of area of $cell_n$ to area of whole chip.

N_T : The total number of cells.

The accuracy of this technique will be discussed further in Chapter 5. Here we note that sensitivity of yield with respect to the fault probability can be found by differentiating eq. (2.10) and (2.1) with respect to A_c to obtain :

$$\frac{\partial Y}{Y} = -\overline{D}\left(\frac{1}{A_c} + \frac{\overline{D}}{\alpha}\right)^{-1}\left(\frac{\partial A_c}{A_c}\right) \tag{2.12}$$

$$\frac{\partial \overline{P}}{\overline{P}} = \frac{\partial A_c}{A_c} \tag{2.13}$$

Combining (2.13) and (2.12) gives :

$$\frac{\partial Y}{Y} = -\overline{D}\left(\frac{1}{A_c} + \frac{\overline{D}}{\alpha}\right)^{-1}\left(\frac{\partial \overline{P}}{\overline{P}}\right) \tag{2.14}$$

Since α is typically close to 1 and $1/A_c$ is far smaller than $\overline{D}/\alpha$, eq. (2.14) can be simplified to give :

$$\frac{\partial Y}{Y} \cong -\overline{D}\left(\frac{\overline{D}}{\alpha}\right)^{-1}\left(\frac{\partial \overline{P}}{\overline{P}}\right) \cong -\alpha\left(\frac{\partial \overline{P}}{\overline{P}}\right) \tag{2.15}$$

According to eq. (2.15), the deviation of the yield is approximately equal to the deviation of the fault probability. Therefore, one can focus on the fault probability instead of the yield when evaluating the accuracy of eq. (2.11).

2.3 Critical Area Calculation

Critical area calculation is an essential part of yield modeling. Once this area is obtained, yield can be easily calculated. In fact, finding critical area is the hardest part of yield modeling. There are essentially two methods used to find the critical area.

2.3.1 Analytic Method

Stapper [2], [3] derived the critical area of two conductive lines and a large number of long conductors by use of an analytical method. The diagram in Figure 2.2 shows two conductive lines, each of width w, and a space s between them. The length of these conductors is L. Stapper assumed that this length is much greater than the line width and spacing between lines so that end effects can be neglected. The locus of the center of defects which result in a failure is the so called critical area, and the critical area is therefore defined as the area in which the center of a defect must fall to

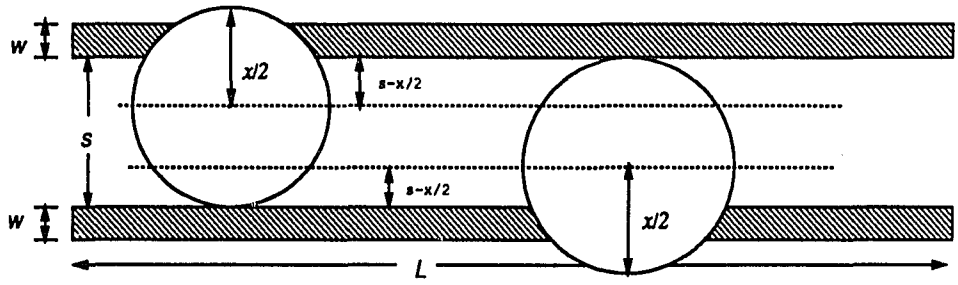


Figure 2.2: Critical area for defects size greater than s.

cause a failure or a fault. Defects of size $x < s$ are not large enough to cause a short circuit, however, defects of size $x \geq s$ can fall between lines to form a bridge and thus produce a short circuit. The critical area for defect sizes greater than s is illustrated in Figure 2.2. The critical area continues to grow as the defect size increases, and thus, the overall critical area as a function of defect size for these two conductors is given by

$$A_c = 0 \quad \text{for} \quad 0 \leq x \leq s \tag{2.16}$$

$$A_c = L(x - s) \quad \text{for} \quad s \leq x \leq \infty \tag{2.17}$$

This function is plotted in Figure 2.3.

Most VLSI circuits contains a lot of parallel conductive lines for interconnect, hence, it is desirable to investigate the critical area of N parallel conductive lines. Figure 2.4 shows N parallel conductors, each with width w and spacing s. All of these lines have the same length which is much greater than their width and the spacing between lines. The critical area for two conductors still holds for N conductive lines except that there are now $(N-1)$ spaces for N lines. Thus, Eq. (2.17) for defect size $x \geq s$ becomes $(N-1)L(x-s)$. These areas keep increasing until two adjacent critical areas meet. This happens in the middle of each line when the defect size $x = w + 2s$ as shown in Figure 2.5. For x greater than this value the critical area increases only in the region outside the parallel conductors. This is illustrated in

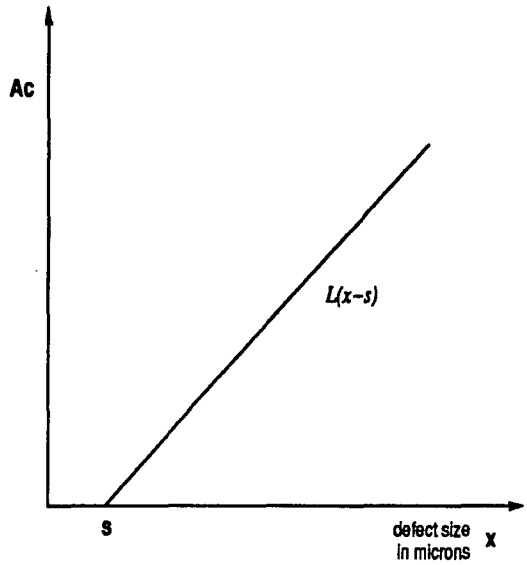


Figure 2.3: Critical area as function of defect size for two long conductive lines.

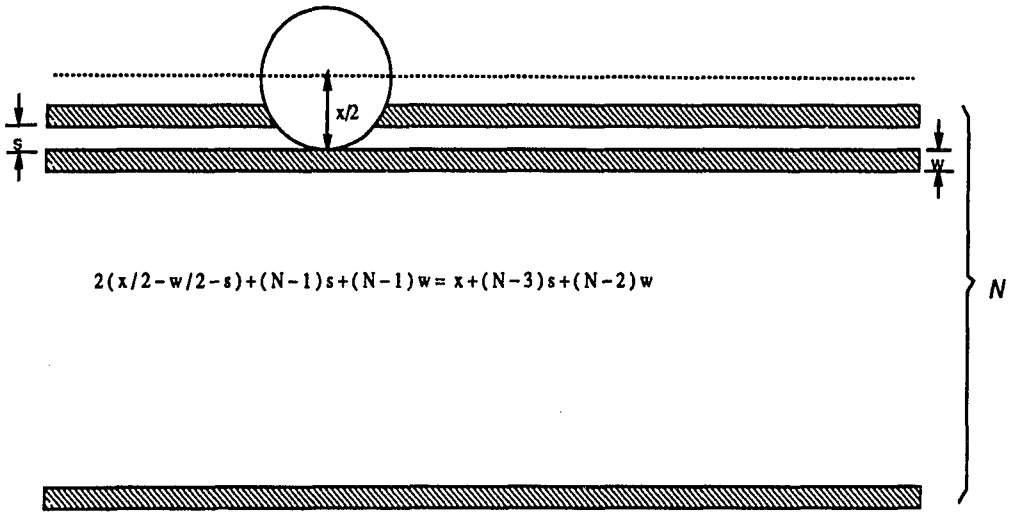


Figure 2.4: Critical area for defects size greater than $w + 2s$.

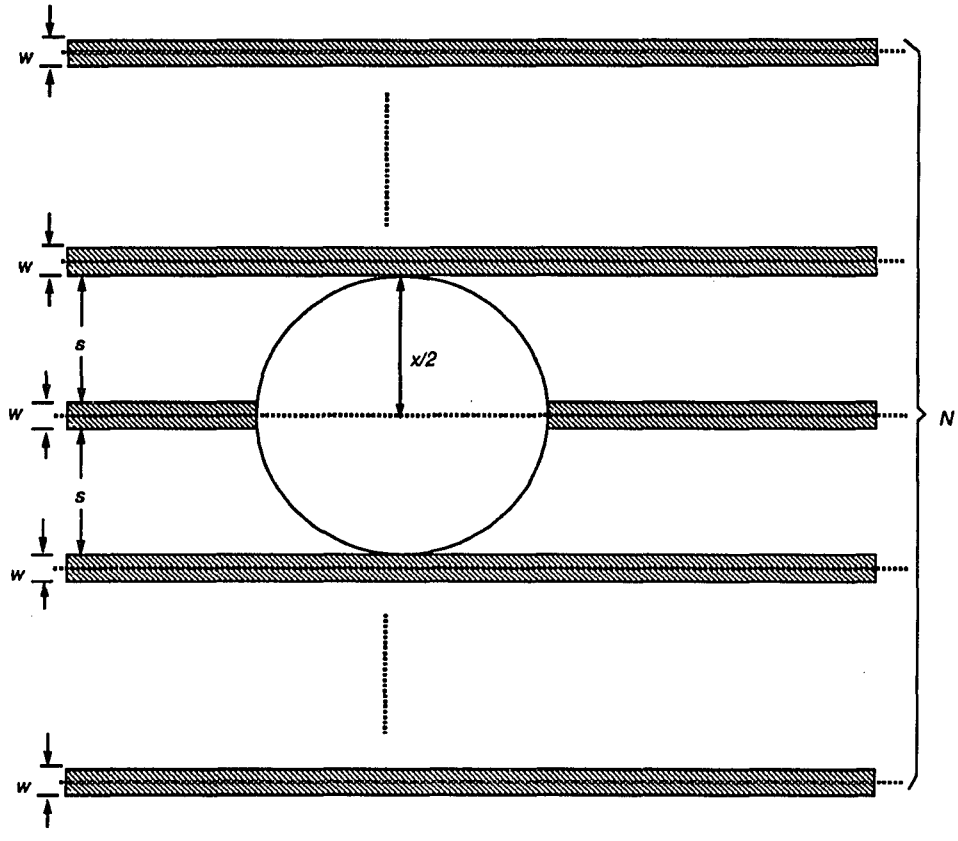


Figure 2.5: Critical area of defect size $x = w + 2s$.

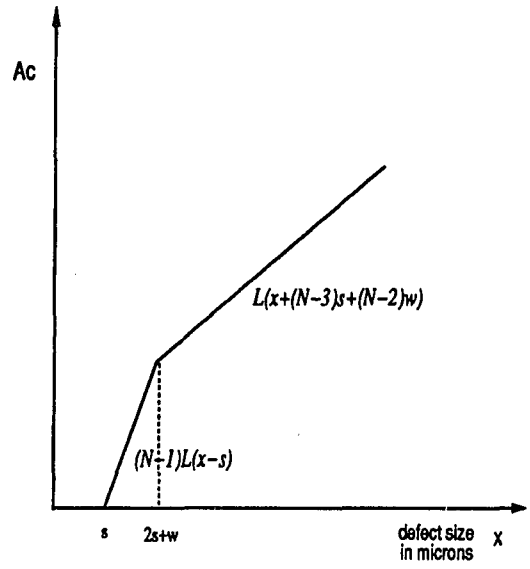


Figure 2.6: Critical area as function of defect size for N conductive lines.

Figure 2.4. The critical area then becomes the area inside N conductors plus the critical area outside the conductors :

$$\underbrace{Nw+(N-1)s}_{\text{area inside N conductors}} \quad + \quad \overbrace{2(\frac{x}{2}-s-w)}^{\text{area outside N conductors}} \quad = x+(N-3)s+(N-2)w$$

The overall critical area is therefore given by

$$A_c = 0 \quad \text{for} \quad 0 \leq x \leq s \tag{2.18}$$

$$A_c = (N-1)L(x-s) \quad \text{for} \quad s \leq x \leq 2s+w \tag{2.19}$$

$$A_c = L\,(x+(N-3)s+(N-2)w) \quad \text{for} \quad 2s+w \leq x \leq \infty \tag{2.20}$$

and is shown in Figure 2.6.

For a real VLSI layout, the situation of N conductors with equal length is seldom encountered, but people do design process monitors like this to optimize design rules, control the fabrication lines, and to determine defect statistics. Figure 2.7 shows part

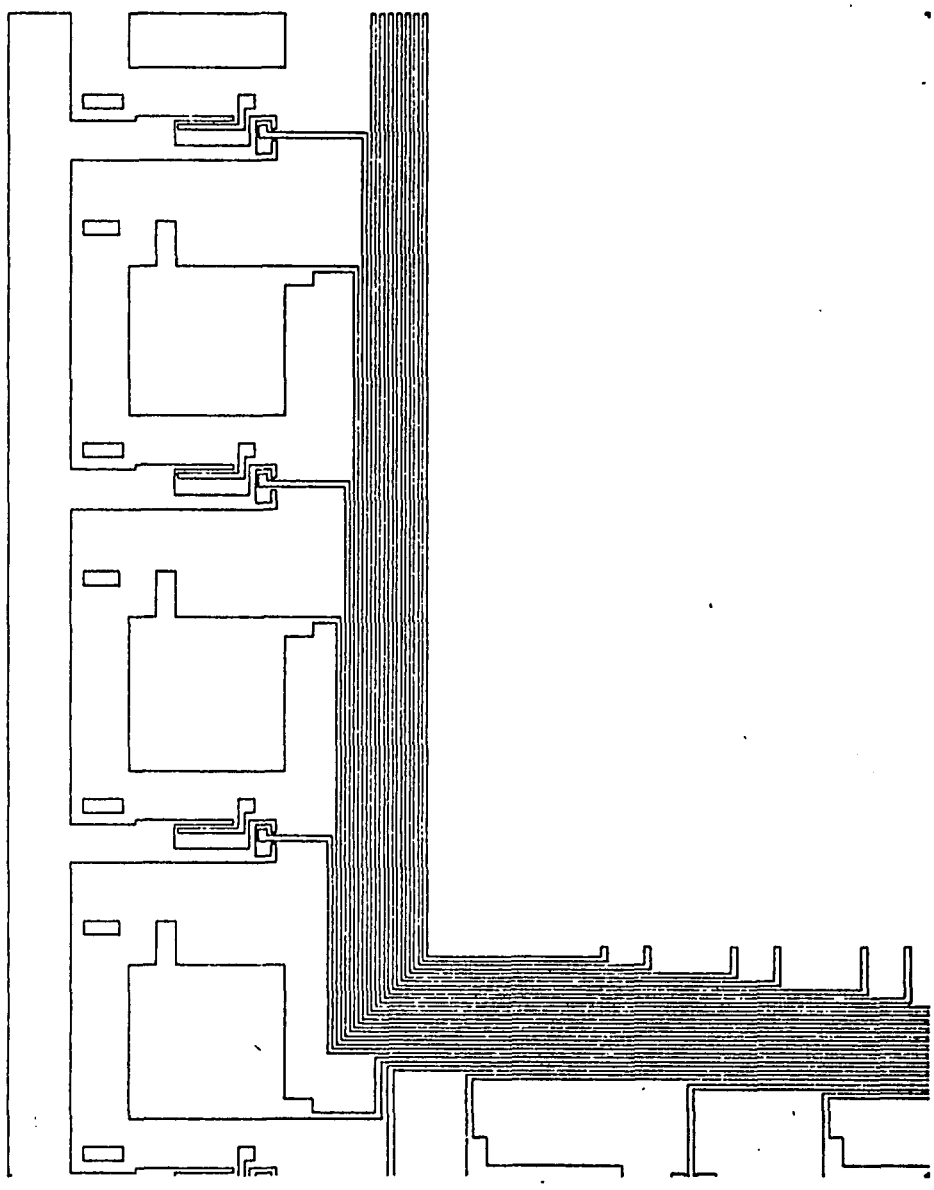


Figure 2.7: Patterns in pad area.

of the line patterns in the pad area of the GE 4k SRAM memory chip. It is obvious that applying the analytic method to this pattern is very tedious due to the varying length and number of parallel conductors, not to mention the patterns in other cells that are not as regular. Because of lack of generality and the layout-dependence, another method was adopted for this work as described in the next section.

2.3.2 Monte Carlo Method

The general scheme of the Monte Carlo method and its application to fault probability calculations are discussed in this section. Requirements on the sample size for this method are also discussed.

Assume that there is some unknown value, m, to be calculated. A random variable ξ is found with its expected value $\rho = m$. The variance of ξ is assumed to be σ^2. In order to estimate m, N random independent variables $\xi_1, \xi_2, \ldots, \xi_N$ with distribution identical to that of ξ are considered. If N is sufficiently large then the distribution of the sum $\rho_N = \xi_1 + \xi_2 + \ldots + \xi_N$ will be approximately normal and will have the mean $\mu_N = Nm$ and the variance $\sigma_N^2 = N\sigma^2$ [5]. The "three sigma rule" for a normal distribution states that the probability of obtaining a value of ξ deviating from the mean by less than 3σ in a single trial is 0.997, and ξ in this case can be expressed as

$$P\left(Nm - 3\sigma\sqrt{N} < \rho_N < Nm + 3\sigma\sqrt{N}\right) \approx 0.997 \tag{2.21}$$

A somewhat different form can be obtained by dividing the inequality in braces by N, and is given by,

$$P\left\{\left|\frac{1}{N}\sum_{i=1}^{N}\xi_i - m\right| < \frac{3\sigma}{\sqrt{N}}\right\} \approx 0.997 \tag{2.22}$$

This equation suggests that the value of m is well estimated by

$$\hat{m} = \frac{1}{N}\sum_{i=1}^{N}\xi_i \tag{2.23}$$

with a very small error.

The application of this method to the calculations of fault probability is stated as follows. Let the random variable, ξ, represent whether or not a defect causes a fault. If a defect causes a circuit failure ξ is set to 1, and the probability that ξ assumes this value is P. Otherwise, ξ is set to 0. Thus the probability distribution of ξ is

ξ	0	1
$f(\xi)$	$1-P$	P

where P is the fault probability. The expected value of ξ is

$$E(\xi) = \sum \xi f(\xi) = 0(1-P) + 1(P) = P. \tag{2.24}$$

Assume the variance of ξ is σ^2. This is exactly the same as required by the Monte Carlo method where N random defects representing N random independent variables, $\xi_1, \xi_2, \ldots, \xi_N$, are generated by a random number generator and placed on the layout. According to eq. (2.23) P can be estimated by

$$\hat{P} = \frac{1}{N}\sum_{i=1}^{N} \xi_i = \frac{\xi_1 + \xi_2 + \ldots + \xi_N}{N} = \frac{N_f}{N} \tag{2.25}$$

where N_f is the number of fault-causing defects.

Since each of the N trials constitutes a binomial trial with probability P, then

$$E(\hat{P}) = E(\frac{N_f}{N}) = \frac{E(N_f)}{N} = \frac{NP}{N} = P \tag{2.26}$$

that is, $\hat{P}$ is an unbiased estimator of P.

The sample variance is then given by,

$$S^2 = \frac{N\sum_i(\xi_i^2) - (\sum_i \xi_i)^2}{N(N-1)} = \frac{NN_f - N_f^2}{N(N-1)} = \frac{N_f(N-N_f)}{N(N-1)} \approx \hat{P}\hat{P}_s \tag{2.27}$$

where $\hat{P}_s$ is the estimator of the probability of success.

The next step is to determine the sample size for the Monte Carlo method [6]. Since the variance of ξ is unknown the sample standard deviation is used in the

$(1-\alpha)100\%$ confidence interval which is :

$$P\left(\left|P-\hat{P}\right| \leq t_{N-1}(\alpha/2)\frac{S}{\sqrt{N}}\right) = 1-\alpha \tag{2.28}$$

where S is the square root of eq. (2.27) and is expressed as :

$$S = \sqrt{\hat{P}\hat{P}_s} \tag{2.29}$$

From equations (2.28) and (2.29), the upper bound of the absolute error E_r between P and $\hat{P}$ can be expressed as

$$E_r = \left|P-\hat{P}\right| = t_{N-1}(\alpha/2)\sqrt{\frac{\hat{P}\hat{P}_s}{N}} \tag{2.30}$$

The relative error, e_r, is defined as

$$e_r = \frac{E_r}{\hat{P}} = \frac{\left|P-\hat{P}\right|}{\hat{P}} = t_{N-1}(\alpha/2)\sqrt{\frac{\hat{P}_s}{N\hat{P}}} \tag{2.31}$$

The sample size N is therefore obtained from eq. (2.30) and is given by :

$$N = \left(\frac{t_{N-1}(\alpha/2)}{e_r}\right)^2 \frac{\hat{P}_s}{\hat{P}} \tag{2.32}$$

where e_r is the relative error defined as $e_r = E_r/\hat{P}$.

This equation is very important for determining the appropriate number of independent random variables, $\xi_1, \ldots, \xi_N$, i.e., defects to be placed on the layout.

CHAPTER 3

Experiment

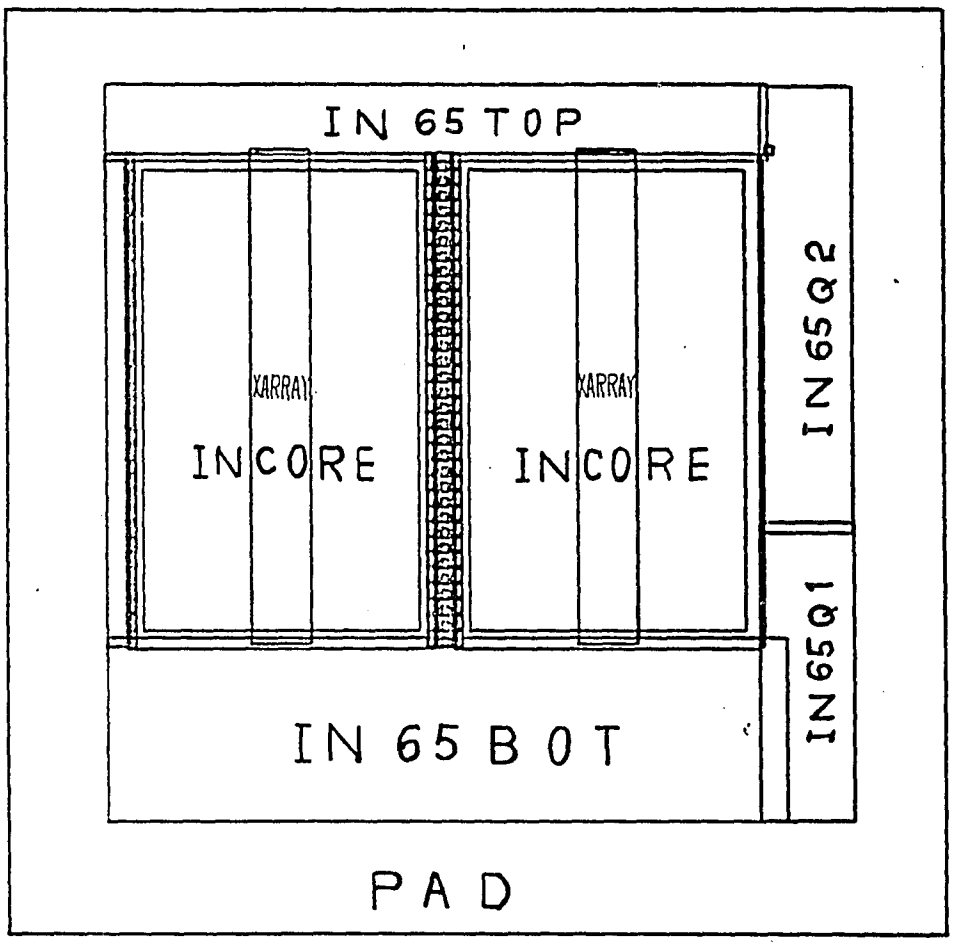
3.1 Introduction

This chapter outlines the complete framework for the Monte Carlo Approach, including the theorem of the generation of random numbers and a method of examining the randomness of these numbers. First, the layout of the 4k memory chip used in the experiments will be discussed. Next, the determination of defect number placed on the layout and procedures for proof of eq. (2.11) and its application will also be discussed.

3.2 About the Chip

The layout software used in this work was the SCALDstar Layout Editor (LED) from Valid Logic Systems Co., San Jose, California. Designs are divided into pieces called cells in the SCALD system. The SCALD database is hierarchical. Cells can contain other cells , and thus result in a "tree structured" design hierarchy. Figure 3.1 shows the hierarchical structure of the layout used in this thesis.

The layout used in this work is a 4096 bit static random access memory organized as 256 words $\times$ 16 bits. This circuit was designed as a process controll vehicle at the General Electric Company as part of their macrocell library. Figure 3.2 shows the bit map for this RAM, and Figure 3.3 is the layout of the metal layer for the whole chip. Elementary cells are identified in this figure such as $cell_1$, the memory unit. There are four equivalent $cell_1$ structures in this chip, each containing 1024 bits of memory.



① incore
② xarray
③ in65bot
④ in65q1
⑤ in65q2
⑥ in65top

Figure 3.1: Hierarchical structure of metal layer.

This is also the most regular and dense region of the chip. There are two $cell_2's$ which is the row decoder, and $cell_3$ contains the column decoder and I/O buffers. Finally, $cell_4$ and $cell_5$ are the address buffers, chip enable, and R/W enable.

In order to check whether or not a defect causes a failure, the database containing all pattern data of this chip is required. The hierarchical layout structure cripples this inspection. Therefore, smashing the hierarchical structure in order to obtain the desired data is necessary. Figure 3.4 shows the layout after part of the pad area has been smashed. The patterns of smashed cells can be seen from the screen of the LED, and the data of that part are also included in the database of the whole chip instead of the coordinates of the cells. The database of the whole chip before and after smashing are also shown below :

```
tech lee
boundary -260110 -271140 255930 250910
grid 25 8 20 25 8 20
≪ metal ≫
use "INCORE"
transform 1 0 -185200 0 1 -99190
box 0 0 152290 259200
use "INCORE"
transform 1 0 -5770 0 1 -99190
box 0 0 152290 259200
use "XARRAY"
transform 1 0 -125660 0 1 -99190
box 0 -7210 33210 271150
use "XARRAY"
transform 1 0 53770 0 1 -99190
box 0 -7210 33210 271150
use "IN65TOP"
transform 1 0 -203630 0 1 168650
box -1780 -3670 363130 39890
use "IN65Q1"
transform 0 -1 202490 1 0 -198710
box -8480 -3360 159760 48950
use "IN65PADFRAME"
transform 1 0 -259700 0 1 -271140
box -410 0 515630 522050
```

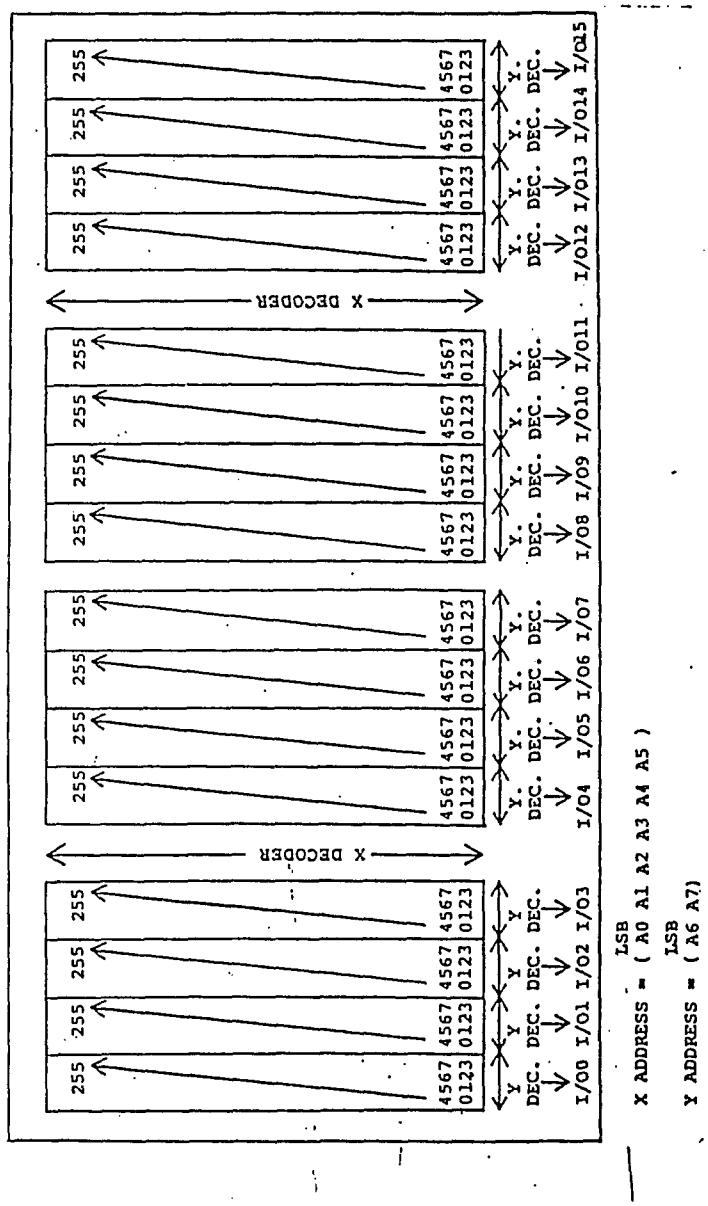


Figure 3.2: Bit map.

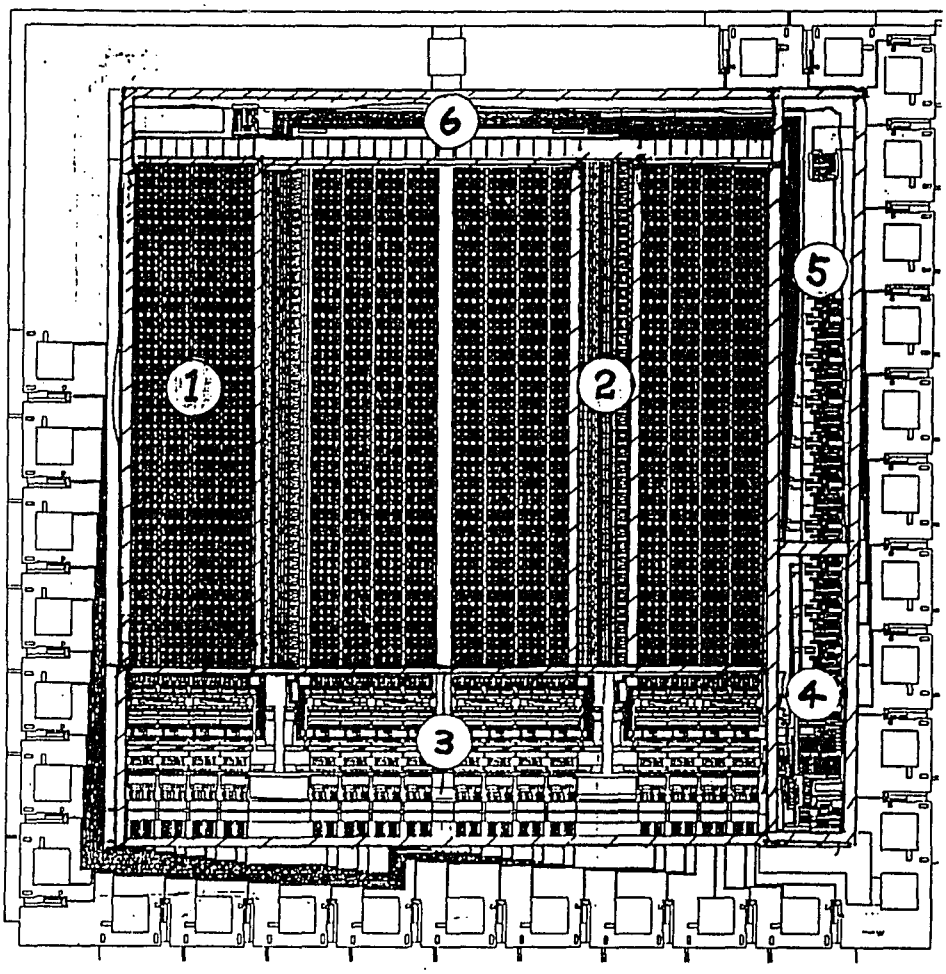


Figure 3.3: Metal layer and elementary cells.

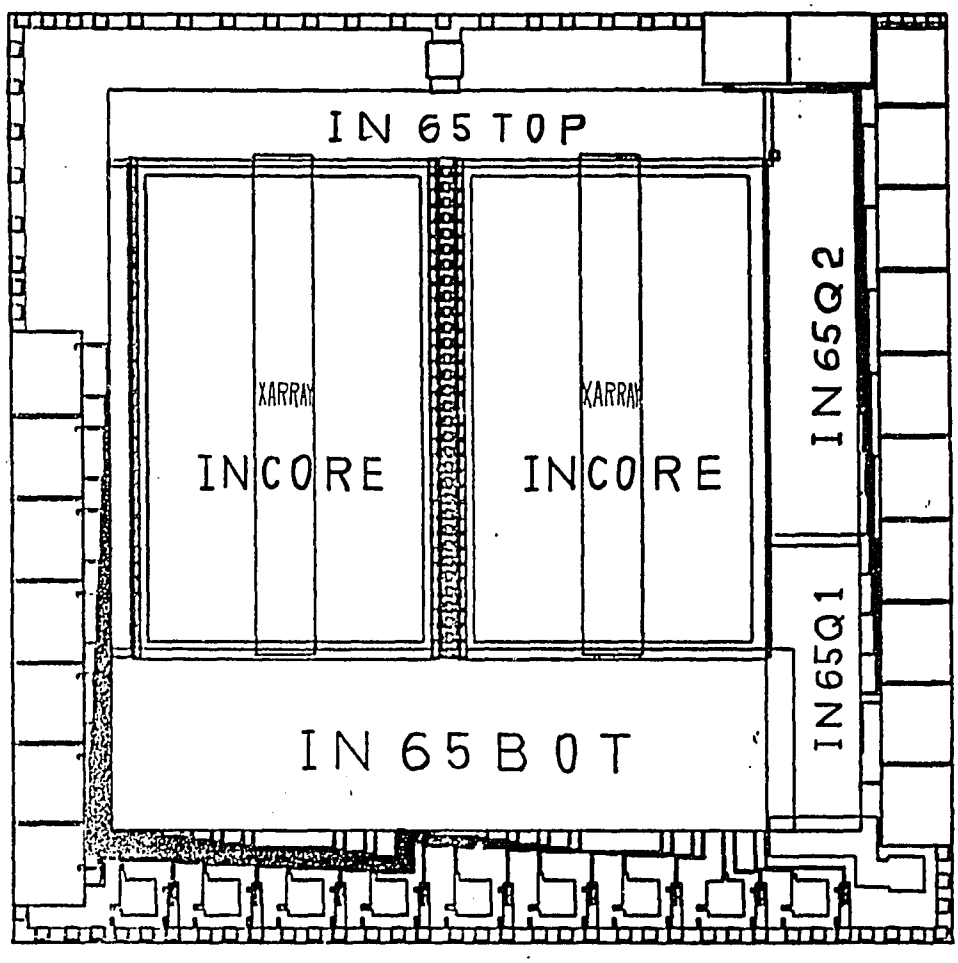


Figure 3.4: Layout patterns of whole chip after part of pad area was smashed.

```
use "IN65Q2"
transform 0 -1 201420 1 0 -43570
box -1560 -4130 250780 46790
use "INRING"
transform 1 0 -194220 0 1 -107830
box 900 -1190 169430 277280
use "INRING"
transform 1 0 -14790 0 1 -107830
box 900 -1190 169430 277280
box -20 0 340 300
⋮
≪ end ≫
tech lee
boundary -260110 -271140 255930 250910
grid 25 8 20 25 8 20
≪ metal ≫
region
rect -198530 -107830 -194520 168650
region
rect -205410 -107830 -198830 168650
region
rect 152730 -206680 153130 -206380
rect 152730 -206380 153730 -205980
region
rect 51930 -206680 52330 -206380
rect 51930 -206380 52730 -205980
region
rect 146930 -206680 147730 -206400
rect 147130 -206400 147730 -205500
region
rect 29330 -206680 30130 -206400
rect 28990 -206400 30130 -205480
use "INCORE"
transform 1 0 -5770 0 1 -99190
box 0 0 152290 259200
use "XARRAY"
transform 1 0 -125660 0 1 -99190
box 0 -7210 33210 271150
use "XARRAY"
transform 1 0 53770 0 1 -99190
box 0 -7210 33210 271150
use "IN65TOP"
transform 1 0 -203630 0 1 168650
```

```
box -1780 -3670 363130 39890
use "IN65Q1"
transform 0 -1 202490 1 0 -198710
box -8480 -3360 159760 48950
use "IN65PADFRAME"
transform 1 0 -259700 0 1 -271140
box -410 0 515630 522050
use "IN65Q2"
transform 0 -1 201420 1 0 -43570
box -1560 -4130 250780 46790
use "INRING"
transform 1 0 -194220 0 1 -107830
box 900 -1190 169430 277280
use "INRING"
transform 1 0 -14790 0 1 -107830
box 900 -1190 169430 277280
⋮
≪ end ≫
```

From this it can be seen that the cell coordinates have been replaced by the pattern data (region, rect, ...) after the cell is smashed. These data will be explained in more detail in the next Chapter.

3.3 Determination of the Proper Number of Defects

The accuracy of the fault probability calculation is greatly influenced by the number of defects placed on the chip. Two considerations were used to determine this number. First, it was observed that the fault probability, $P(x)$, oscillates for defect numbers between 0 and 400 for all of the elementary cells. In order to obtain a steady and accurate fault probability, the defect number placed on the layout has to be greater than four hundred according to this observation. The curves in Figureres 3.5 through 3.7 show these oscillations for some of the cells. Second, for the Monte Carlo method presented in Chapter 2, fault probability is estimated from N_f/N , where N is the total number of sample points generated from the random number generator, and N_f is the number of sample points which cause circuit failure.

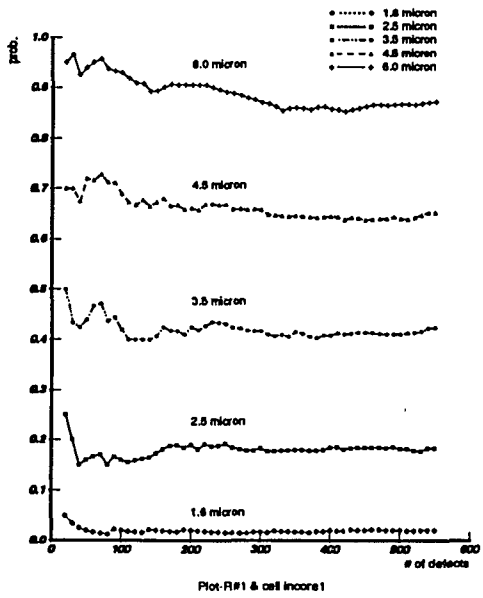


Figure 3.5: P(x) vs. # of defects for $cell_1$.

According to eq. (2.32) derived in section 2.3.2, the sample size, N, is :

$$N = \left(\frac{t_{N-1}(\alpha/2)}{e_r} \right)^2 \frac{\hat{P}_s}{\hat{P}} \tag{3.1}$$

$\hat{P}$ probability of failure.

$\hat{P}_s$ probability of success.

$t_{N-1}(\frac{\alpha}{2})$ the upper 100αth percentile of the student-t distribution with (N-1) degrees of freedom.

Thus, for worst case ($\hat{P}$ being equal to 0.01), the sample size has to be greater than 38,000 based on a confidence level $\alpha = 0.05$ and relative error $e_r = 0.1$.

The number of defects placed on a layout was therefore determined by choosing the larger number based on these two considerations. A defect number $N = 40,000$ is chosen for more accuracy.

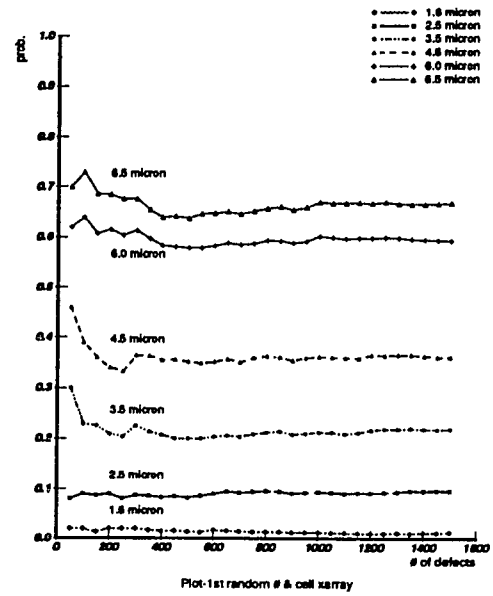


Figure 3.6: P(x) vs. # of defects for $cell_2$.

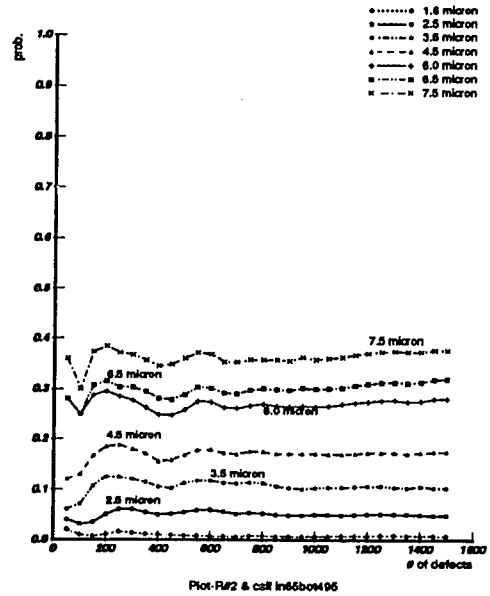


Figure 3.7: P(x) vs. # of defects for $cell_3$.

3.4 Random Number Generation

Since Monte Carlo simulation involves the use of random numbers when sampling from a particular distribution, the performance of this method depends very much on just how random these numbers are. In fact, these numbers are not really random, but only seem so, and are referred to as *pseudorandom* or *quasi-random.* Random numbers are said to be good only if they are uniformly distributed, statistically independent, and reproducible. The next two sections introduce the random number generation and a statistical test of these pseudorandom numbers.

3.4.1 Random Number Generator

The most commonly used present-day method for generating pseudorandom numbers is one that produces a nonrandom sequence of numbers according to some recursive formula, which is based on calculating the residues modulo of some integer m of a linear transformation. It is readily seen from this definition that each term of the sequence is available in advance, before the sequence is actually generated. Although these processes are completely deterministic, it will be shown in section 3.4.2 that the numbers generated by the sequence appear to be uniformly distributed. The method used in this paper is the congruential algorithm based on a fundamental congruence relationship, which is expressed as

$$X_{i+1} = (aX_i + c)(\text{mod } m), \quad i = 1, \ldots, n. \tag{3.2}$$

where a is the multiplier, c is the increment, and m is the modulus. The modulo notation (mod m) means

$$X_{i+1} = aX_i + c - mk_i, \tag{3.3}$$

where $k_i = [(aX_i + c)/m]$ denotes the largest positive integer in $(aX_i + c)/m$.

Given an initial starting value X_0 (also called the *seed*), eq. (3.3) yields a congruence relationship (modulo m) for any value i of the sequence $\{X_i\}$.

Generators that produce random numbers according to eq. (3.3) are called *mixed congruential generators.* The random numbers on the unit interval (0,1) can be obtained by

$$U_i = \frac{X_i}{m}. \tag{3.4}$$

Clearly, such a sequence will repeat itself in at most m steps, and will therefore, be periodic. From eq. (3.3) $X_i < m$ for all i. This means that the period of the generator cannot exceed m, that is, the sequence X_i contains at most m distinct numbers. The entire sequence reoccurs as soon as any number in the sequence is repeated. Then the generator gets into a loop, that is, there is a cycle of numbers that is repeated endlessly. It has been shown that all sequences having the form $X_{i+1} = f(X_i)$ get into a loop [7]. Therefore, the value of m has to be chosen as large as possible to ensure a sufficiently large sequence of distinct numbers in a cycle.

The random number generator used in this thesis is *nrand*48(), which is a built-in function in C language, and uses the same algorithm as described in eq. (3.2). The parameter $m = 2^{48}$, and the multiplier value a and the increment c are given by

$$a = 2736731631558 \tag{3.5}$$

$$c = 138. \tag{3.6}$$

3.4.2 The Randomness of the Random Number Generator

When using the random number generator to generate defects, it is assumed these defects are uniformly distributed over a fixed area. In fact, the random number generator is pseudorandom. A chi-square goodness-of-fit test [8]

$$\chi^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i} \tag{3.7}$$

was used to check the uniformity of the defect distribution, where χ^2 is approximated very closely by the χ^2 distribution with k-1 degrees of freedom. The symbols o_i and

e_i represent the observed and expected frequencies, respectively, for the ith interval. A good fit leads to the acceptance of the null hypothesis, H_0, which means the distribution being tested is uniform, whereas, a poor fit leads to its rejection, which means the distribution being tested is not uniform.

Briefly, the procedure is as follows. Forty thousand random numbers are generated. They are separated according to where they fall in the range of the uniform distribution (namely 0 to 1), and a chi-square test is calculated to measure the deviation of the actual observed frequencies from what one would expect to find if the distribution were truly uniform on the interval 0 to 1. More specifically, we partition the range (0 to 1) of the uniform distribution into $k = 100$ intervals (0.00, 0.01) , (0.01, 0.02), ..., (0.98, 0.99), (0.99, 1.00). If the distribution is truly uniform the expected frequencies for the ith (i = 1 to 100) interval is $e_i = (40,000 * 1/100) = 400$. We, then, separate these random numbers into those intervals and perform the χ^2 test. Two groups of random numbers are generated for comparison and are tested by eq. (3.7). For the first group of random numbers (R#1), $\chi^2_{1x} = 101.11 \text{and} \chi^2_{1y} = 102.96$. For the second group of random numbers (R#2), $\chi^2_{2x} = 95.86$ and $\chi^2_{2y} = 80.00$. For the level of significance (α) equal to 0.01 and 99 degree of freedom, the critical value χ^2_α is 134.64. Since χ^2_{1x}, χ^2_{1y}, χ^2_{2x}, and χ^2_{2y} are all smaller than χ^2_α, all four values fall in the acceptance region of H_0, and hence, we prove that these two groups of random numbers are uniformly distributed.

3.5 Procedure

The procedure for proving the applicability of eq. (2.11) and its use in calculating critical areas is discussed in this section.

3.5.1 Procedures of Proving Formula

The main advantage of equation (2.11) is that one can calculate the fault probability based on the database of elementary cells instead of on the whole database of a chip. The proof of this approach can be shown by comparing the fault probability calculated with eq. (2.11) with that obtained from an experiment based on the whole database of a layout. The former obtained from eq. (2.11) is represented by $P_{ther}(x)$, and the latter obtained from computer experiments is by $P_{exp}(x)$. The difference of finding these two values is that Monte Carlo simulation is used once in the calculations of $P_{exp}(x)$ based on placing defects over the whole database of the chip while it is used six times, once on each database of the six elementary cells for $P_{ther}(x)$. The seven defect diameters (1.6 μm, 2.5 μm, 3.5 μm, ..., 7.5 μm) were used in both $P_{exp}(x)$ and $P_{ther}(x)$ in order to examine the effect of defect size on the prediction. The percentage error as a function of defect size x between these two values was calculated in order to evaluate the accuracy of this formula, and the results are discussed in Chapter 5.

Because of the regularity of the patterns in $cell_1$, a further simplification of this cell is possible. A smaller array truncated from $cell_1$ is used to represent it in the prediction of $P_{ther}(x)$. The truncation, shown in Figure 3.8, is a 18×12 memory array, whereas, $cell_1$ is actually a 64×16 memory array. The result of this substitution for $cell_1$ in the prediction of $P_{exp}(x)$ is also discussed in Chapter 5.

A further consideration based on the correlation between elementary cells can improve the prediction of eq. (2.11) and will be discussed in section 5.3.

3.5.2 Procedures of Finding Critical Area

Figure 3.9 shows the main procedures to obtain critical area. After determining the appropriate number of defects (40,000) and proving the random numbers are

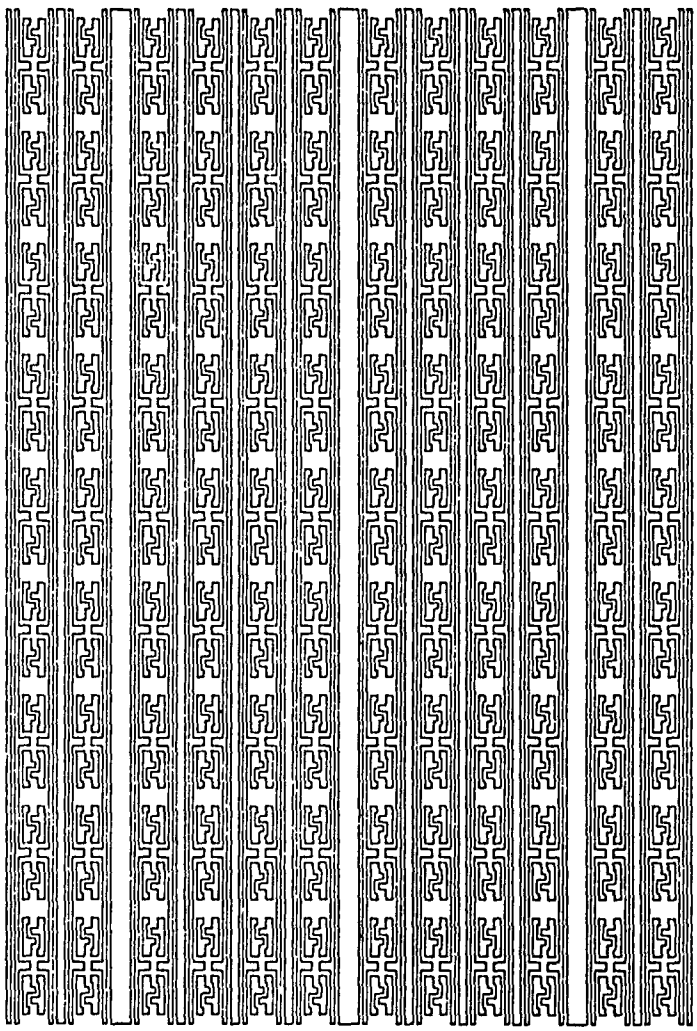


Figure 3.8: The truncation of cell incore.

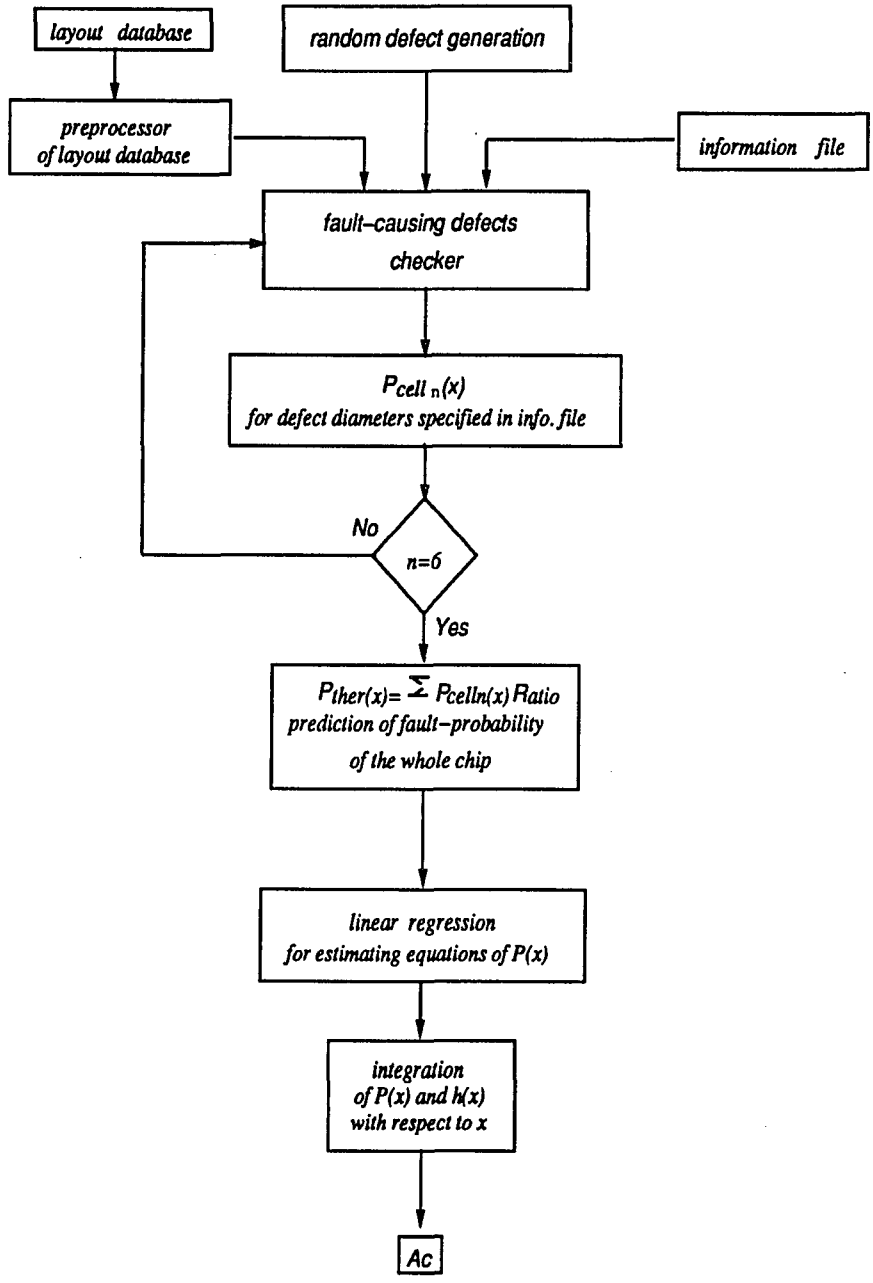


Figure 3.9: Procedures of finding critical area.

acceptable, random defects are placed on elementary cells, and $P_{cell_n}(\mathrm{x})$ according to eq. (2.6) was calculated for each of the elementary cells. A computer program, has been developed to check fault-causing defects for the random defects, placed on the layout. The defect diameters can be specified in the input file. The fault probability $P_{ther}(\mathrm{x})$ for these defect diameters are then obtained using eq. (2.11) based on P_{cell_n}. As can be seen from eq. (2.8), rewritten below, the equation of $P_{ther}(\mathrm{x})$ has to be evaluated in order to calculate A_c :

$$A_c = A \int_0^\infty P(x)h(x)\,dx. \tag{3.8}$$

Thirty eight defect diameters were chosen in order to find the curve of $P_{ther}(\mathrm{x})$ with respect to defect size x. The equation of $P_{ther}(\mathrm{x})$ was then estimated by linear regression based on these thirty eight P_{ther}. In order to obtain the best fit of the curvature, $P_{ther}(\mathrm{x})$ was divided into several sections. Different regression equations were used to estimate the curve in different sections. The critical area is then obtained by substituting $P_{ther}(\mathrm{x})$ and $h(x)$ into eq. (2.8). The division of $P_{ther}(\mathrm{x})$, the regression equations, and the calculation of the critical area will be discussed in section 5.5.

CHAPTER 4

Database and Program

4.1 Introduction

The major process in this thesis is to find the fault probability for the various cells, including the subcell of memory array, the whole cell of the chip, etc. A program, called fault-causing defect checker, has been developed to accomplish this task. In order to understand this program, the database of LED has to be exposed. Another program which places defects on the layout in order to examine the relationship between defects and polygons has also been developed. With this program the layout with defects can be seen in LED so that fault analysis can also be achieved visually.

4.2 Database of the LED

In Chapter 3, the databases before and after smashing of the hierarchical structure were shown. The difference between these two is that the database before the data is smashed contains the coordinates of subcells constructing the parent cell, while the database after it has been smashed contains the data of polygons in this parent cell. This hierarchical structure saves a lot of disk space to store a layout, but it makes the investigation of fault-causing defects difficult. This is the reason why the layout is smashed so as to obtain the data of all polygons in the layout.

Part of the layout database and the meaning of these numbers are shown in Figure 4.1.

```
tech lee
boundary  222925 64400 228275 67745
grid 25 8 20 25 8 20
<< metal >>
region
zoid 222925 64700 225020 64700 222925 64400 224720 64400      (x11 y11) (x12 y12) (x13 y13) (x14 y14)
zoid 224890 65000 225320 65000 224590 64700 225020 64700  →   (x21 y21) (x22 y22) (x23 y23) (x24 y24)
zoid 225190 65300 228275 65300 224890 65000 228275 65000      (x31 y31) (x32 y32) (x33 y33) (x34 y34)
region
rect 223650 64885 224150 65665          (a11 b11) (a12 b12)
rect 223650 65665 224700 66125  →       (a21 b21) (a22 b22)
rect 223650 66125 224040 67205          (a31 b31) (a32 b32)
rect 223650 67205 224200 67745          (a41 b41) (a42 b42)
<< end >>
```

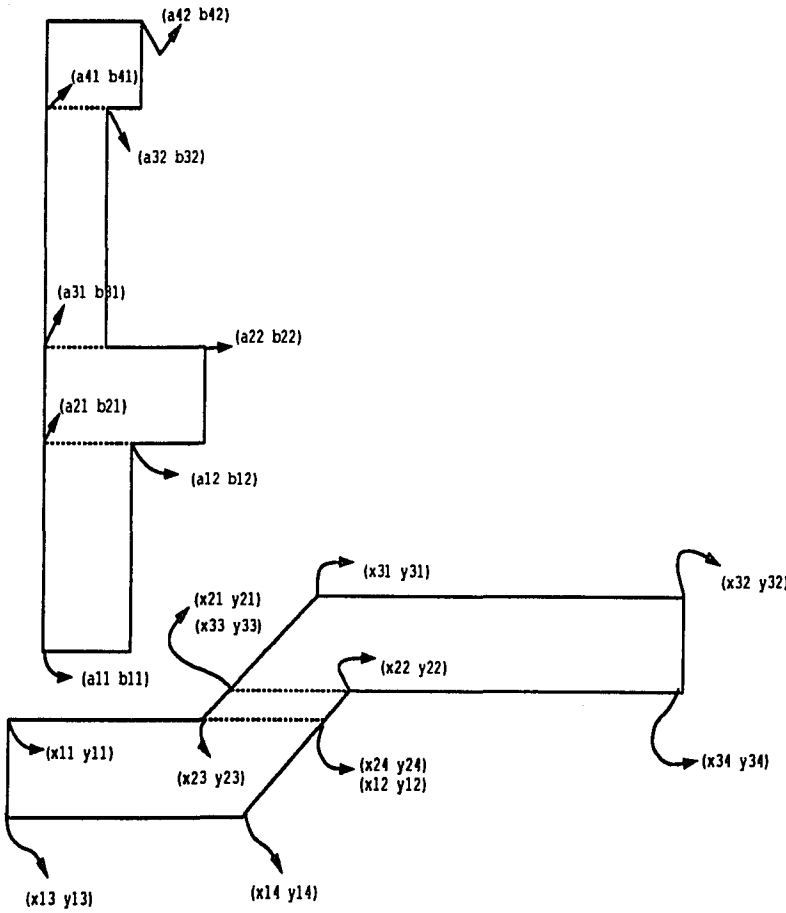


Figure 4.1: Database of layout.

There are two kinds of shapes in the metal layer, rectangle and trapezoid. The former is represented by *rect* in the database, and the latter is represented by *zoid.* Each pattern is a combination of these two shapes and is separated from other patterns by *region* in the database. There are two sample patterns in Figure 4.1, each one begins with *region* as stated above. The meaning of figures following *rect* and *zoid* are also shown. The four numbers following *rect* represent the coordinates of the lower left and upper right corner of a rectangle. The eight numbers following *zoid* represent the coordinates of the four trapezoid corners, in the order of upper left, upper right, lower left, and lower right. The units of these numbers are not micro meter which is used in LED. In fact, the unit of these figures is $\mu m \times 200$ because the minimum resolution of LED is 0.005 μm.

The first entry represents the design technology used in the layout, including CMOS, or some other technology, as defined by the user. The second entry represents the layout boundary which is the maximum and minimum coordinates of the database. The third entry represents how the grid is drawn in LED. The fourth entry of the database represents which layer the data belongs to.

4.3 Fault-Causing Defect Checker

Any defect is included as a fault-causing defect as long as it constructs a bridge between two polygons. The situation that two defects together cause a short circuit is not included here because fault probability is defined as the probability that a single defect of size x causes a circuit failure. A computer program, fault-causing defect checker, has been developed to accomplish this. The algorithm of this program is composed of the following steps :

1. The centers of random defects are generated with a random number generator.

2. The database is converted into parameters used by the program.

3. The center of one defect is read in.

4. The polygons surrounding this defect are found.

5. The overlap between defect and polygons coming from 4 is checked.

6. Determine if the polygons from 5 belong to the same pattern.

7. If two polygons belong to two different patterns a fault-causing defect is found.

8. Return to step 2 if any defect has not been checked.

9. Divide the number of fault-causing defects by the total number of defects to find P_{cell_n}.

This program is listed in the Appendix A.

4.4 Defect Placer

The yield prediction can be used to optimize design rules and evaluate process sensitivity. It is therefore necessary to show a layout with defects, used in the yield prediction, on it in LED. This program has been developed and listed in the Appendix B. Figure 4.2 shows a layout with 2.5 μm defects placed on it.

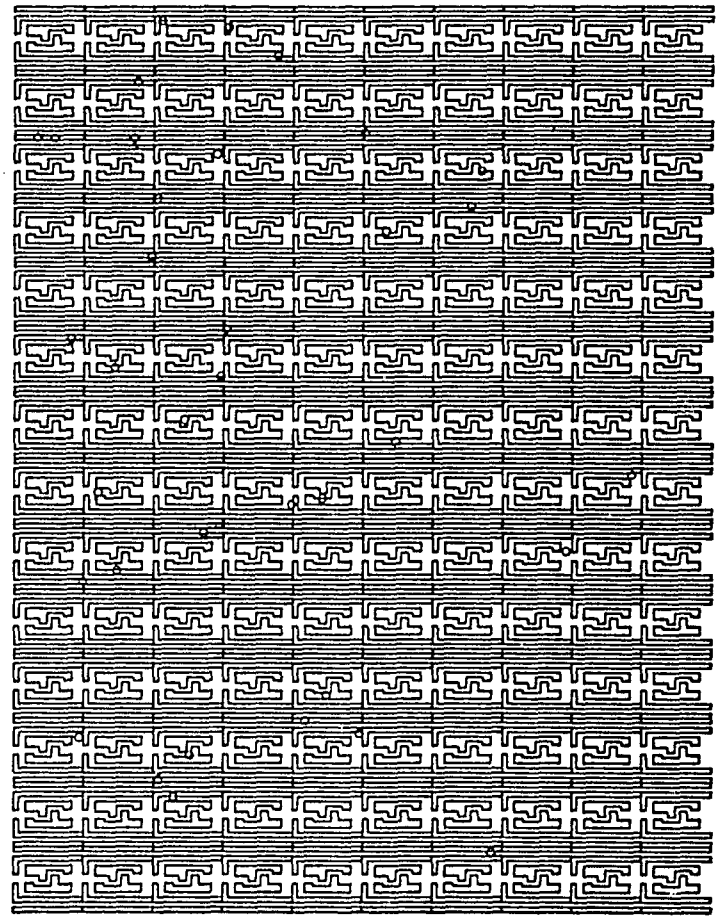


Figure 4.2: Metal layer and defects.

CHAPTER 5

Result and Conclusion

5.1 Introduction

The results of the experiments are presented in this chapter. A further approach to simplify the database used in the experiments is discussed, and a method to calculate the fault probability of the whole chip is also presented. The critical area is then evaluated using this fault probability.

5.2 Proof of Formula

Table 5.1 and Table 5.2 show the results of $P_{cell_n}(x)$ obtained from Monte Carlo simulation for R#1 and R#2. The defect diameters were chosen to be 1.6 μm, 1.8 μm, 2.0 μm, ...in order to investigate the influence of defect size on the prediction of eq. (2.11). There are seven defect sizes listed below, and the rest are listed in the Appendix C.

	$1.6\mu m$	$2.5\mu m$	$3.5\mu m$	$4.5\mu m$	$6.0\mu m$	$6.5\mu m$	$7.5\mu m$
P_{cell_1}	0.018425	0.19375	0.42345	0.65645	0.863525	0.901225	0.957575
P_{cell_2}	0.0077	0.091625	0.203575	0.3343	0.556525	0.622725	0.735175
P_{cell_3}	0.00395	0.0448	0.103375	0.17615	0.279975	0.3117	0.3738
P_{cell_4}	0.001425	0.0258	0.08435	0.14905	0.24655	0.277275	0.3393
P_{cell_5}	0.00365	0.046576	0.1177	0.1905	0.25925	0.283775	0.328725
P_{cell_6}	0.001125	0.0112	0.0631	0.128125	0.2197	0.249025	0.285525

Table 5.1 : Fault probability of elementary cells (R#1).

	1.6μm	2.5μm	3.5μm	4.5μm	6.0μm	6.5μm	7.5μm
P_{cell_1}	0.018075	0.19065	0.42745	0.663375	0.865725	0.9025	0.958525
P_{cell_2}	0.007875	0.092975	0.206125	0.33965	0.5623	0.628	0.740975
P_{cell_3}	0.00365	0.04325	0.102725	0.1747	0.275	0.308475	0.37025
P_{cell_4}	0.0016	0.02545	0.085325	0.150175	0.2491	0.2789	0.339575
P_{cell_5}	0.004325	0.046175	0.11425	0.187625	0.25715	0.282975	0.328325
P_{cell_6}	0.000875	0.010875	0.065675	0.1301	0.223475	0.2523	0.29025

Table 5.2 : Fault probability of elementary cells (R#2).

The six elementary cells listed in Tables 5.1-5.2 are the basic cells required in the calculations of eq. (2.11) as previously defined in Figure 3.3. Based on these six elementary cells, eq. (2.11) becomes :

$$\begin{aligned} P_{ther}(x) \;=\; & 4 \times P_{cell_1}(x) \times Ratio(cell_1) + 2 \times P_{cell_2}(x) \times Ratio(cell_2) \\ & + P_{cell_3}(x) \times Ratio(cell_3) + P_{cell_4}(x) \times Ratio(cell_4) \\ & + P_{cell_5}(x) \times Ratio(cell_5) + P_{cell_6}(x) \times Ratio(cell_6) \qquad (5.1) \end{aligned}$$

$$Ratio(cell_n) = \frac{area(cell_n)}{area(mainchip)}$$

where area(mainchip) is the total chip area. For the data in Tables 5.1-5.2 using equation 5.1, P_{ther}(x) was calculated for R#1 and R#2 as shown in Tables 5.3-5.4. P_{exp}(x) obtained by Monte Carlo simulation over the whole chip, are also shown in these two tables for comparison in eq. (5.1) above, the coefficient 4 and 2 in front of $P_{cell_1}(x)$ and $P_{cell_2}(x)$ indicate that there are 4 $cell_1's$ and 2 $cell_2's$ in the metal layer as explained in section 3.2 and shown in Figure 3.3.

	1.6μm	2.5μm	3.5μm	4.5μm	6.0μm	6.5μm	7.5μm
P_{exp}	0.01595	0.120875	0.257975	0.407	0.563175	0.5998	0.660625
P_{ther}	0.01030	0.111185	0.250238	0.397271	0.552424	0.588660	0.648142

Table 5.3 : P_{ther} & P_{exp} with the pad area excluded (R#1)

	1.6μm	2.5μm	3.5μm	4.5μm	6.0μm	6.5μm	7.5μm
P_{exp}	0.016475	0.12005	0.2611	0.409775	0.566475	0.6037	0.661775
P_{ther}	0.010135	0.109539	0.252158	0.400617	0.553228	0.589382	0.648790

Table 5.4 : P_{ther} & P_{exp} with the pad area excluded (R#2)

The *area(mainchip)* used in eq. (5.1) is the chip area excluding the pad area. This is because the pattern density in the pad area is low, and are not as susceptible to defects as those in other areas. The pad area is therefore omitted in P_{exp}(x) and P_{ther}(x). The impact of this approximation and a method to estimate the fault probability of the total chip area will be discussed later in section 5.4.

The evaluation of this prediction was examined using the percent error by which P_{ther}(x) deviated from P_{exp}(x) :

$$\text{Percent error } \% = \frac{P_{ther}(x) - P_{exp}(x)}{P_{exp}(x)} \times 100\% \tag{5.2}$$

	$1.6\mu m$	$2.5\mu m$	$3.5\mu m$	$4.5\mu m$	$6.0\mu m$	$6.5\mu m$	$7.5\mu m$
R#1 error %	-35.4261	-8.0166	-2.9991	-2.3904	-1.9090	-1.8573	-1.8896
R#2 error %	-38.4843	-8.7559	-3.4249	-2.2349	-2.3385	-2.3716	-1.9622

Table 5.5 : Percentage error for layout excluded pad area (R#1 & R#2)

A defect is included as a short circuit defect when it constructs a bridge between two patterns. The minimum feature of the metal layer is 1.5 μm, and therefore, fault-causing defects are very few when the defect diameter approaches this value. This property makes the errors in the diameter range, 1.6 μm to 2.5 μm, seem to be intolerable. The number of fault-causing defects is $\leq$ 600/40000 for 1.6 μm, $\leq$ 4700/40000 for 2.5 μm, and $\leq$ 17000/40000 for 3.5 μm. Thus, small deviation from these two numbers (600 and 4700) makes the error quite large. For example, a deviation of 200 makes the percent errors to be 33.33% for 600 at 1.6 μm, 4.25% for 4700 at 2.5 μm, and 1.18% for 17000 at 3.5 μm. Figures 5.1-5.2 are plots of fault probability versus defect size and show that because the fault probability is low these errors are negligible.

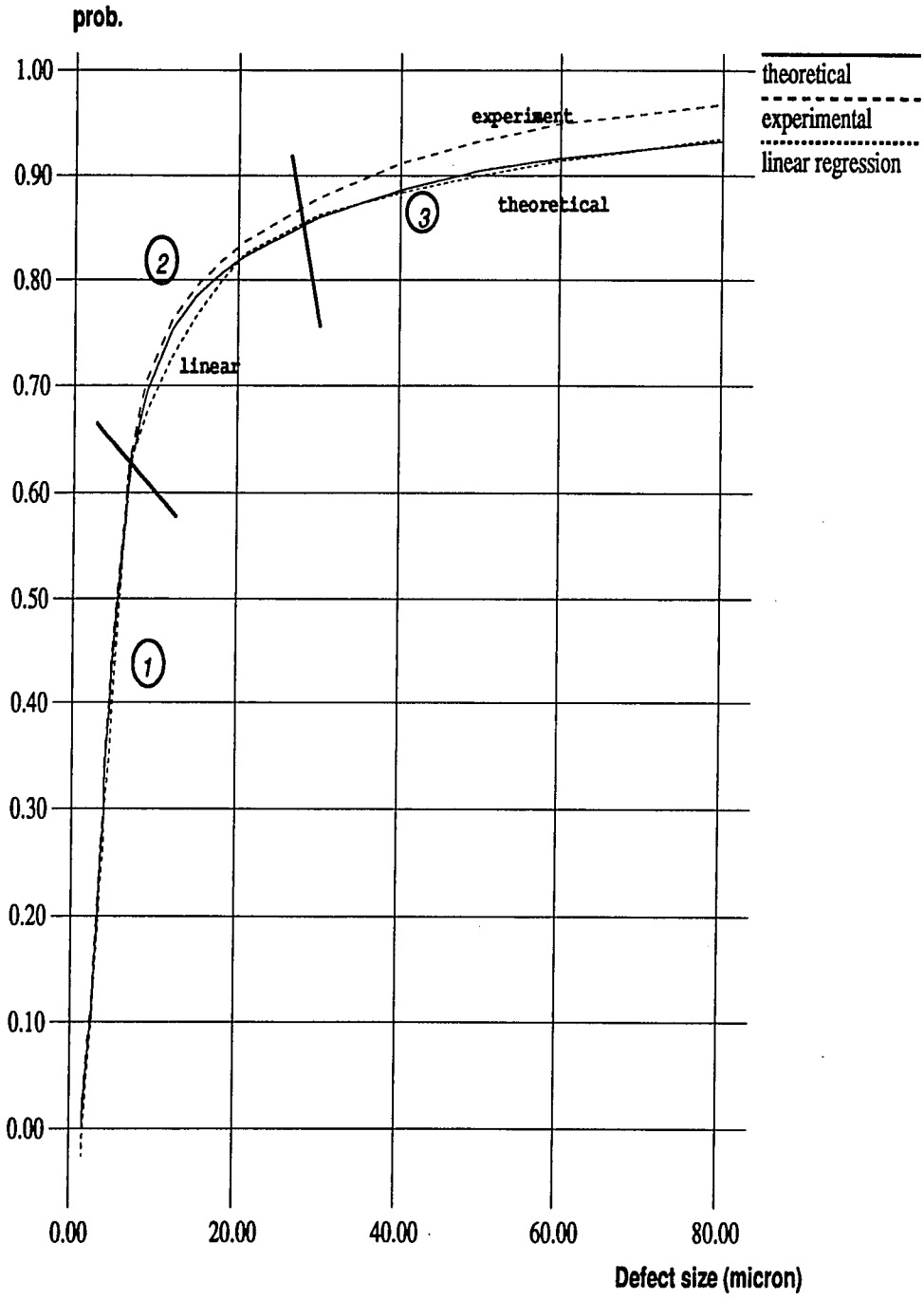


Figure 5.1: Fault probability vs. defect size (R#1).

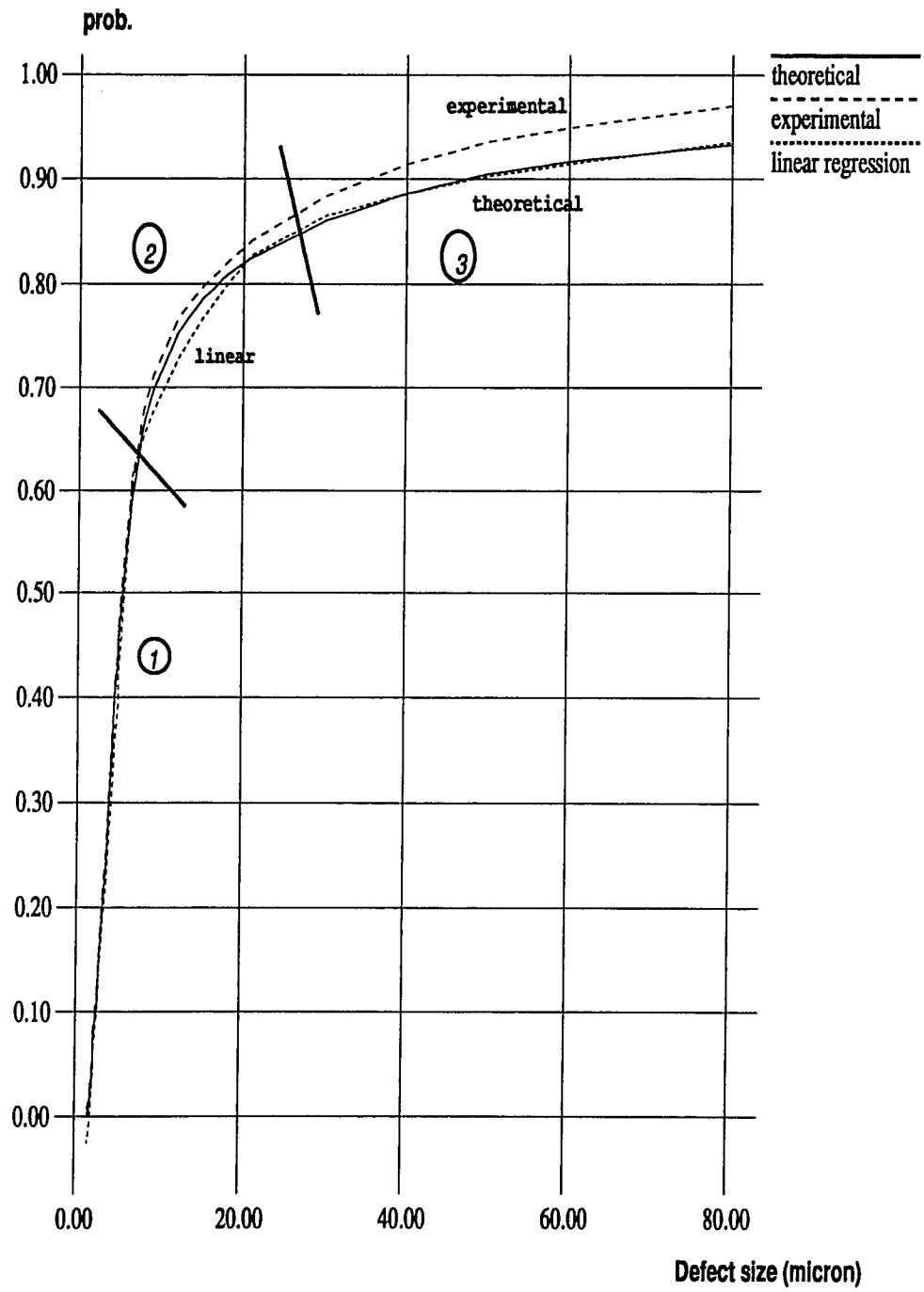


Figure 5.2: Fault probability vs. defect size (R#2).

5.3 Further Refinement

Although the percent error in 1.6 μm and 2.5 μm are negligible the error of other defect sizes still seems too large. A further improvement can be made by adding a compensation in eq. (2.11). When eq. (2.11) was originally formulated the correlation between cells was not considered. The correlation between cells represents the spacing between cells which could also cause faults by defects and can be regarded as two parallel conductors so that eq. (2.17) can be applied directly to this situation. There are four intervals between $cell_1$ and $cell_2$ and four intervals between $cell_1$ and peripheral polygons. The following equations were obtained using eq. (2.17) after measuring the space and length of these intervals.

$$P_1(x) = \frac{A_{c1}(x)}{area(mainchip)} = \frac{4594.61(x-1.5)}{area(mainchip)}$$
$$P_2(x) = \frac{A_{c2}(x)}{area(mainchip)} = \frac{5534.48(x-2.0)}{area(mainchip)}$$
$$P_3(x) = \frac{A_{c3}(x)}{area(mainchip)} = \frac{2764.74(x-2.75)}{area(mainchip)}$$

where A_{c1} and A_{c2} represent the critical area between $cell_1$ and peripheral polygons, A_{c3} represents the critical area between $cell_1$ and $cell_2$, and $P(x) = A_c(x)/A$ according to eq. (2.6). The sum of these three terms construct the compensation (offset(x)) of eq. (2.11). After including offset(x) to eq. (2.11), P_{ther}(x) and the percent error between P_{ther}(x) and P_{exp}(x) were calculated. These values are shown in Tables 5.6 and 5.7.

	1.6μm	2.5μm	3.5μm	4.5μm	6.0μm	6.5μm	7.5μm
P_{ther}	0.010408	0.112919	0.254846	0.404916	0.564624	0.602378	0.664896
% error	-34.748	-6.5822	-1.2131	-0.5121	0.2572	0.4298	0.6465

Table 5.6 : P_{ther} & percentage error after offset adding (R#1)

	1.6μm	2.5μm	3.5μm	4.5μm	6.0μm	6.5μm	7.5μm
P_{ther}	0.010243	0.111272	0.256765	0.408261	0.565427	0.6031	0.665544
% error	-37.8275	-7.3117	-1.6601	-0.3694	-0.1849	-0.09937	0.5696

Table 5.7 : P_{ther} & percentage error after offset adding (R#2)

The errors are less than one percent for most of the defect sizes after the compensation was added. This indicates the assumption of correlation between cells is accurate and effective. The compensation has, therefore, been added to eq. (2.11) for the prediction of fault probability. However, this term can not be used for all defect sizes because the critical area will overlap between cells and compensation after some defect size x_f. The value of x_f can be determined from the analytical method described in section 2.3.1. There is a transition in critical area when the defect size is equal to $2s + w$ for N conductive lines. This is because the critical areas begins to overlap with each other when the defect size is greater than $2s + w$. The same is true for the situation presented here. Thus, the compensating term is no longer valid for defect size greater than $2s + w$.

As mentioned in section 3.5.1, a 18×12 memory array truncated from $cell_1$ was substituted for $cell_1$ in the prediction of fault probability to see if the database can be further reduced. The fault probability of this truncated memory array is listed below in Table 5.8 and 5.9 (P_{cell_1} and the percent error between these two values are also included for reference).

	1.6μm	2.5μm	3.5μm	4.5μm	6.0μm	6.5μm	7.5μm
P_{cell_1}	0.018425	0.19375	0.42345	0.65645	0.863525	0.901225	0.957575
$P_{cell_{1_t}}$	0.018675	0.1981	0.425425	0.661275	0.865025	0.901225	0.957
% error	1.3568	2.245	0.4664	0.735	0.1737	0.0	0.06

Table 5.8 : Comparison of P_{cell_1} & $P_{cell_{1_t}}$ (R#1)

	$1.6\mu m$	$2.5\mu m$	$3.5\mu m$	$4.5\mu m$	$6.0\mu m$	$6.5\mu m$	$7.5\mu m$
P_{cell_1}	0.018075	0.19065	0.42745	0.663375	0.865725	0.9025	0.958525
$P_{cell_{1_t}}$	0.019175	0.195825	0.4248	0.6586	0.865275	0.903925	0.9599
% error	6.085	2.714	0.6199	0.7198	0.0519	0.1578	0.1434

Table 5.9 : Comparison of P_{cell_1} & $P_{cell_{1_t}}$ (R#2)

As shown in the above Tables, the percent error between $P_{cell_1}(x)$ and $P_{cell_{1_t}}(x)$ is quite small. A further inspection of the impact of this substitution on the P_{ther}(x) is necessary. P_{ther}(x) based on this replacement and the consequent percentage error by which P_{ther}(x) deviated from P_{exp}(x) are shown in Table 5.10-5.11.

	$1.6\mu m$	$2.5\mu m$	$3.5\mu m$	$4.5\mu m$	$6.0\mu m$	$6.5\mu m$	$7.5\mu m$
P_{ther}	0.010519	0.114860	0.255727	0.407068	0.565293	0.602378	0.664640
% error	-34.0484	-4.9765	-0.8715	0.0168	0.3761	0.4298	0.6077

Table 5.10 : P_{ther}(x) and % error (R#1)

	$1.6\mu m$	$2.5\mu m$	$3.5\mu m$	$4.5\mu m$	$6.0\mu m$	$6.5\mu m$	$7.5\mu m$
P_{ther}	0.010734	0.113581	0.255583	0.406131	0.565227	0.603736	0.666158
% error	-34.8487	-5.3885	-2.1129	-0.8892	-0.2204	0.0059	0.6623

Table 5.11 : P_{ther}(x) and % error (R#2)

From the observation of these two Tables, the replacement of P_{cell_1} by $P_{cell_{1_t}}$ did not influence P_{ther}(x) by very much, and , therefore, this is a good approach. The success of this substitution makes the prediction of P_{ther}(x) easier and faster, and the database required in the experiment is further reduced.

5.4 Fault Probability of Whole Chip

As mentioned in the previous section, all P_{ther}(x) and P_{exp}(x) obtained up to this point are based on the cell mainchip which is the whole chip less the pad area. The reason for doing this is that the size of polygons in the pad area is big and

consequently the susceptibility to defects is low. The most vulnerable polygons to defects in this region are those parallel lines for connection between pads and buffers. The critical area of these parallel lines can be estimated using the analytic method described in section 2.3.1, and consequently the fault probability of this region is obtained by $A_{c_{pad}}/area(pad)$. Equation (2.20) is used and the approximate number and length of parallel lines is estimated due to the geometric complexity of these lines. The lines can be separated into four groups :

1. L = 1036.20 μm, N = 12.
2. L = 784.13 μm, N = 11.
3. L = 118.12 μm, N = 9.
4. L = 800.12 μm ,N = 5.

The sum of the corresponding critical areas of these four groups are :

$$A_{c_{pad}}(x) = 26234.49 \times (x - 2.5) \quad \text{for} \quad 2.5 \leq x \leq 8\mu m \tag{5.3}$$

$$\begin{aligned} A_{c_{pad}}(x) = \; & 647.625 \times (x + 52.5) + 784.13 \times (x + 47) \\ & + 118.12 \times (x + 36) + 800.12 \times (x + 14) \\ & \text{for } x \geq 8\mu m \end{aligned} \tag{5.4}$$

The fault probability of the whole chip can therefore be expressed as :

$$\begin{aligned} P_{total}(x) = \; & P_{ther}(x)Ratio(mainchip) + P_{pad}(x)Ratio(pad) \\ = \; & P_{ther}(x)Ratio(mainchip) + \frac{A_{c_{pad}}(x)}{area(chip)} \end{aligned} \tag{5.5}$$

$$Ratio(cell_n) = \frac{area(cell_n)}{area(chip)}$$

where area(chip) is now the total area of the whole chip, P_{ther}(x) is defined in eq. (5.1), and the truncated $cell_1$ is employed. Equation (5.5) follows the same rules

as used before. The results of applying eq. (5.5) for the fault probability and the experiment value based on $area(chip)$ are shown below in Tables 5.12 and 5.13.

	$1.6\mu m$	$2.5\mu m$	$3.5\mu m$	$4.5\mu m$	$6.0\mu m$	$6.5\mu m$	$7.5\mu m$
$P_{total_{exp}}$	0.01215	0.075325	0.1675	0.2637	0.3682	0.39335	0.435025
P_{total}	0.006637	0.072471	0.165250	0.264638	0.370318	0.395666	0.438849

Table 5.12 : P_{total} & $P_{total_{exp}}$ with the pad area included (R#1)

	$1.6\mu m$	$2.5\mu m$	$3.5\mu m$	$4.5\mu m$	$6.0\mu m$	$6.5\mu m$	$7.5\mu m$
$P_{total_{exp}}$	0.01155	0.0764	0.168575	0.266675	0.3725	0.396575	0.438025
P_{total}	0.006772	0.071664	0.165160	0.264047	0.370276	0.396523	0.439807

Table 5.13 : P_{total} & $P_{total_{exp}}$ with the pad area included (R#2)

Table 5.14 shows the percent error between P_{total}(x) and $P_{total_{exp}}$(x) when the pad area is included.

	$1.6\mu m$	$2.5\mu m$	$3.5\mu m$	$4.5\mu m$	$6.0\mu m$	$6.5\mu m$	$7.5\mu m$
R#1 error %	-45.3730	-3.7888	-1.3431	0.3558	0.5753	0.5889	0.8791
R#2 error %	-41.364	-6.1983	-2.0260	-0.9855	-0.5969	-0.0131	0.4068

Table 5.14 : Percentage error for layout included pad area (R#1 & R#2)

Eq. (5.5) is the final formula used in the prediction of fault probability, and the calculation of critical area of the total chip is also based on the data in Table 5.12 and Table 5.13.

5.5 Result of Experiment

After the fault probabilities as a function of defect size were determined, the critical area was calculated by following the flow chart in Figure 3.9. The data for the fault probabilities as a function of defect size (listed in Tables 5.12-5.13 and in

Appendix C) were divided into three sections which were shown in Figures 5.1-5.2. Linear regression was then used to obtain equations for these three sections.

For defect sizes from 1.5 μm to 7.5 μm the direct linear regression was used as given by :

$$\mu_{Y|x} = \alpha + \beta x. \tag{5.6}$$

Here the regression coefficients α and β are parameters to be determined from the sample data. Denoting their estimates by a and b, respectively, $\mu_{Y|x}$ can be estimated by $\hat{y}$ from the sample regression line

$$\hat{y} = a + bx \tag{5.7}$$

where the estimates a and b represent the y intercept and slope, respectively. Given the sample $\{x_i, y_i; i = 1, 2, 3, \ldots, n.\}(x_i = defectsize, y_i = P_{total}(x))$, the least squares estimates a and b of the regression coefficients α and β are computed from the formulas

$$b = \frac{N\sum_{i=1}^{N} x_i y_i - \left(\sum_{i=1}^{N} x_i\right)\left(\sum_{i=1}^{N} y_i\right)}{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2} \tag{5.8}$$

$$a = \frac{\sum_{i=1}^{N} y_i - b\sum_{i=1}^{N} x_i}{N} \tag{5.9}$$

By substituting data from defect sizes 1.5 μm to 7.5 μm into the above equations, the coefficients of a and b for the first section of the line are given by,

$$a = -0.111508$$

$$b = 0.078619$$

The data in segments two and three can not be estimated directly by eq. (5.6) because the linear equation does not accurately describe the relationship between y_i and x_i. The scatter of the data in segments two and three suggests a logarithmic regression. Consider, for example, the equation :

$$\mu_{Y|x} = \alpha + \beta \ln(x) = \alpha + \beta x' \tag{5.10}$$

which may be rewritten as

$$\hat{y} = a + bx' \tag{5.11}$$

which is of exactly the same form as eq. (5.7). After taking the logarithm of every defect size in segments two and three, the coefficients of a and b can then be obtained from eq. (5.9), and are given by :

$$\begin{aligned} a_2 &= 0.221102 \\ b_2 &= 0.105385 \\ a_3 &= 0.351113 \\ b_3 &= 0.063606 \end{aligned}$$

The reason for separating the curve $x \geq 7.5\ \mu m$ into two sections is that a better fit can be achieved this way. The separation point is at $x \approx 30\mu m$. The complete equations of the P_{total}(x) are then given as follows :

$$\begin{array}{rccccccl} & 1.5 \leq x \leq 7.5 & & 7.5 \leq x \leq 30 & & 30 \leq x \leq 80 & \\ P(x) = & -0.111508 + 0.078619X & + & 0.221102 + 0.105385\ln(x) & + & 0.351113 + 0.063606\ln(x) & R\#1 \\ & \text{section 1} & & \text{section 2} & & \text{section 3} & \\ P(x) = & -0.112307 + 0.078753X & + & 0.223064 + 0.104719\ln(x) & + & 0.350090 + 0.063873\ln(x) & R\#2 \\ & \text{section 1} & & \text{section 2} & & \text{section 3} & \end{array}$$

The critical area is then given by substituting these equations and the defect size distribution into eq. (2.8), and the result of A_c/A is listed as follows :

$$\begin{aligned} \left.\frac{A_c}{A}\right|_{ther} &= 2.2362 \times 10^{-4} (R\#1) \\ \left.\frac{A_c}{A}\right|_{ther} &= 2.2268 \times 10^{-4} (R\#2) \end{aligned}$$

where $A = 4246046\ (\mu m)^2$. The experimental value of A_c/A is also listed bellow for comparison.

$$\begin{aligned} \left.\frac{A_c}{A}\right|_{exp} &= 2.3113 \times 10^{-4} (R\#1) \\ \left.\frac{A_c}{A}\right|_{exp} &= 2.3150 \times 10^{-4} (R\#2) \end{aligned}$$

Comparing $A_c/A|_{ther}$ and $A_c/A|_{exp}$ it is clear that the errors in fault probability prediction do not influence the A_c calculation very much.

5.6 Conclusion

Critical area evaluation is the key part of the procedure for accurately calculating yield of an integrated circuit. Stapper [3] has derived a simple formula to find the critical area of two conductive lines and an arbitrary large numbers of long conductors as described in section 2.3.1. This method for finding the critical area has been adopted in this thesis as a complementary method to achieve more accuracy in the fault probability prediction. The main method used in this thesis to find the critical area is the Monte Carlo simulation which requires a large sample size and complex procedures. In section 2.2, an equation is proposed for solving this problem. The equation deals with the subcircuits of a VLSI circuit. Monte Carlo simulations were used to find the fault probabilities of each significant subcell and these were combined according to eq. (2.11). The principles for choosing significant subcells are

1. choose dense and as regular patterns as possible for a subcell. $Cell_1, \cdots,$ $cell_6$ exemplified this rule. $Cell_7$ is excluded by this rule.

2. choose once if there are any identical subcells.

3. any subcell with a regular layout pattern can be replaced by part of it.

The combined fault probability is then compensated by considering the correlation between subcells as blocks of parallel lines making the fault probability more accurate. The fault probability of the pad area can be obtained by applying the analytical method to the parallel conductors, rather than using Monte Carlo simulation, because the most susceptible patterns to defects in the pad area are parallel lines which carry signals to and from the circuit.

Monte Carlo simulations are often time-consuming. In this thesis a method for obtaining the critical area by using a Monte Carlo Simulation and a method for partitioning the large chip database for accurate critical area evaluation has been demonstrated. This method, which estimates the fault probability of the whole chip by combining subcell fault probabilities, makes the Monte Carlo Simulation easier and faster. It has been shown that this method can predict the fault probability with reasonable accuracy. The critical area, calculated by use of this fault probability, as compared with a full Monte Carlo simulation, shows that the error is insignificant in fault probability. Therefore, the equation proposed in this thesis is valid and significantly reduces the time required for the Monte Carlo simulation of fault probability.

Appendix A
Fault-Causing Defect Checker

```
/* create defect INTERNALLY hostp include cirxl,ciryl and offset */
/* don't use more than one rectxxxxx   */
#include <stdio.h>
#include <math.h>
#define  deg    0.034906585
#define  rang   2147483647
  int      dnum,conn,insec,zoid;
  int      torects,totrz,totdata,totloop,totspv;
  int      cirxl,cirxr,ciryu,ciryd,totp,totz,totr;
  int      rradi,rcirx,rciry,posmarker[200];
  int      cirf,specvalu[6];
  float    radi,diam,prob;
  float    cirx,ciry,offsetx,offsety;
  float    cirxL,ciryL;
  FILE     *in1,*in3,*in4,*out1,*out2;
  char     outf[20],sentence[200];
  char     rects[20],caldata[20];
  float    cir1x[55000],cir2x[55000],cir1y[55000],cir2y[55000];
  struct xcor
  {   int   marker;
      int   item[4];
  };
  struct pcor
  {   char  xybig;
      int   marker;
      int   item[7];
  };
  struct zcor
  {   int   marker;
      int   item[6];
      int   a1;
      int   b1;
      int   c1;
      int   b2;
```

```
      int   c2;
  };
  struct xcor rectan[100000],*rec;
  struct pcor posrect[200],*pos;
  struct zcor trapezi[5000],*tra;

main()
{
  char    coma=',';
  int     i,j,k,l,m,n,o,p,r;
  int     amaxnum,maxnum,minnum,inter;
  int     xt1,xt2,xt,xtx,yt1,yt2,yt,ytx,xp,yp;
  int     zy1,zy2,zy,zyx,zx1,zx2,zx3,zx4,zxx,zxx1,zxx2,zx;
  int     dver,dhor,dhv,lef,rig,spv;

 out1=fopen("cir1","w");
 in4=fopen("hostp","r");
 fscanf(in4,"%d",&torects);
 fscanf(in4,"%d\n",&amaxnum);

 for (r=0;r<torects;r++)
 {
 tra=&trapezi[0];
 pos=&posrect[0];
 rec=&rectan[0];
 fscanf(in4,"%s\n",rects);
 printf("name of preprocess cell is ----> %s\n",rects);/*****/
 fscanf(in4,"%s\n",caldata);
 printf("name of caldata         is ----> %s\n",caldata);/*****/
 fscanf(in4,"%f %f\n",&cirxL,&ciryL);
 printf("The limit of x=%12f\nThe limit of y=%12f \n",cirxL,ciryL);
 fscanf(in4,"%f %f\n",&offsetx,&offsety);
 printf("offsetx=%12f\noffsety=%12f in %s\n",offsetx,offsety,rects);

 for (i=0;i<amaxnum;i++)
  defect(i);

 in1=fopen(rects,"r");
 fscanf(in1,"%d\n",&totrz);
 in3=fopen(caldata,"r");
 fscanf(in3,"%d\n",&totdata);
 fscanf(in3,"%d",&totspv);
```

```
for (i=0;i<totspv;i++)
{ fscanf(in3,"%d",&specvalu[i]);  }
j=0; k=0;
for (i=0;i<totrz;i++)
{ fscanf(in1,"%d",&zoid);
  switch(zoid)
  {
  case 0:
  { fscanf(in1,"%d %d %d %d %d\n",&(rec+j)->marker,
    &(rec+j)->item[0], &(rec+j)->item[1],&(rec+j)->item[2],
    &(rec+j)->item[3]);
    j+=1;  break; }
  case 1:
  { fscanf(in1,"%d %d %d %d %d %d %d",&(tra+k)->marker,
    &(tra+k)->item[0], &(tra+k)->item[1],&(tra+k)->item[2],
    &(tra+k)->item[3], &(tra+k)->item[4],&(tra+k)->item[5]);
    fscanf(in1,"%d %d %d %d %d",&(tra+k)->a1,&(tra+k)->b1,
           &(tra+k)->c1,&(tra+k)->b2,&(tra+k)->c2);
    k+=1;  break; }
  }
}
fclose(in1);

totr=j; totz=k;
for (p=0;p<totdata;p++)
{
 fscanf(in3,"%f\n",&diam);
 printf("diameter is              ---> %f\n",diam);/*****/
 fscanf(in3,"%s\n",sentence);
 printf("%s\n",sentence);/****/
 fscanf(in3,"%d\n",&minnum);
 printf("minimum # of defects is  ---> %d\n",minnum);/*****/
 fscanf(in3,"%d\n",&maxnum);
 printf("maximum # of defects is  ---> %d\n",maxnum);/***/
 fscanf(in3,"%d\n",&inter);
 printf("interval             is  ---> %d\n",inter);/****/
 fscanf(in3,"%f\n",&offsetx);
 printf("offset of x          is  ---> %f\n",offsetx);/***/
 fscanf(in3,"%f\n",&offsety);
 printf("offset of y          is  ---> %f\n",offsety);/***/
 fscanf(in3,"%d\n",&cirf);
 printf("infile name of defect center> %d\n",cirf);/*****/
```

```
fscanf(in3,"%s\n",outf);
printf("outfile name         is  ---> %s\n",outf);/****/

radi=diam/2.0; conn=0;
rradi=(int)(radi*200.0+0.5);

out2=fopen(outf,"w");
fprintf(out2,"     diameter      # of defects
             # of faults         probability\n\n");/****/
totloop=0;  spv=0;
for (n=inter;n<=maxnum+inter;n+=inter)
{ printf("%d\n",totloop);/****/
for (k=0;k<inter;k++)
{ if(cirf == 1 )
  { cirx= cir1x[totloop];  ciry= cir1y[totloop];  }
  if(cirf == 2 )
  { cirx= cir2x[totloop];  ciry= cir2y[totloop];  }
 cirx=cirx+offsetx;  ciry=ciry+offsety;
 rcirx=(int)(cirx*200.0+0.5); rciry=(int)(ciry*200.0+0.5);
 cirxl=rcirx-rradi; cirxr=rcirx+rradi;
 ciryu=rciry+rradi; ciryd=rciry-rradi;
 j=0; totloop++;
 for (i=0;i<totr;i++)
 { xt1=((rec+i)->item[0]>=cirxl) && ((rec+i)->item[0]<=cirxr);
   if (xt1==0) { xt2=((rec+i)->item[2]>=cirxl) &&
                 ((rec+i)->item[2]<=cirxr); }
   xt=(xt1||xt2);
   xtx=((rec+i)->item[0]<cirxl) && ((rec+i)->item[2]>cirxr);
   if ( (xt || xtx)==1 )
   { yt1=((rec+i)->item[1]>=ciryd) && ((rec+i)->item[1]<=ciryu);
     if (yt1==0) { yt2=((rec+i)->item[3]>=ciryd) &&
     ((rec+i)->item[3]<=ciryu); }
     yt=(yt1||yt2);
     ytx=((rec+i)->item[1]<ciryd) && ((rec+i)->item[3]>ciryu);
     if ( (yt || ytx)==1 )
     {  (pos+j)->marker=(rec+i)->marker;
        (pos+j)->item[0]=(rec+i)->item[0];
        (pos+j)->item[1]=(rec+i)->item[1];
        (pos+j)->item[2]=(rec+i)->item[2];
        (pos+j)->item[3]=(rec+i)->item[3];
        if (xtx==1) (pos+j)->xybig='x';
        if (ytx==1) (pos+j)->xybig='y';
```

```
        if (ytx==0 && xtx==0) (pos+j)->xybig='o';
        j+=1; }
  }
}
for (l=0;l<totz;l++)
{ zy1=((tra+l)->item[1]<=ciryu) && ((tra+l)->item[1]>=ciryd);
  if (zy1==0)
   { zy2=((tra+l)->item[4]<=ciryu) && ((tra+l)->item[4]>=ciryd); }
  zy=(zy1 || zy2);
  zyx=((tra+l)->item[4]<ciryd) && ((tra+l)->item[1]>ciryu);
  if ((zy || zyx)==1)
  { zx1=((tra+l)->item[0]<=cirxr) && ((tra+l)->item[0]>=cirxl);
    if(zx1==0) { zx2=((tra+l)->item[2]<=cirxr) &&
    ((tra+l)->item[2]>=cirxl);
     if(zx2==0) { zx3=((tra+l)->item[3]<=cirxr) &&
     ((tra+l)->item[3]>=cirxl);
      if(zx3==0) { zx4=((tra+l)->item[5]<=cirxr) &&
      ((tra+l)->item[5]>=cirxl); } } }
    zxx1=((tra+l)->item[0]<cirxl) && ((tra+l)->item[2]>cirxr);
    zxx2=((tra+l)->item[3]<cirxl) && ((tra+l)->item[5]>cirxr);
    zxx=(zxx1 || zxx2);
    zx=(zx1 || zx2 || zx3 || zx4);
    if ((zx || zxx)==1)
    { (pos+j)->marker=(tra+l)->marker;
      (pos+j)->item[0]=(tra+l)->item[1];
      (pos+j)->item[1]=(tra+l)->item[4];
      (pos+j)->item[2]=(tra+l)->a1;
      (pos+j)->item[3]=(tra+l)->b1;
      (pos+j)->item[4]=(tra+l)->c1;
      (pos+j)->item[5]=(tra+l)->b2;
      (pos+j)->item[6]=(tra+l)->c2;
      (pos+j)->xybig='z';
      j+=1;
    }
  }
}
totp=j;
if ( totp==1 || totp==0 ) { insec=0; j=0; dhv=0; goto jump1; }
if (totp==2)
 if ( pos->marker==(pos+1)->marker) { insec=0;
      j=0; dhv=0; goto jump1; }
insec=0; j=0;
```

```
   dhor=0; dver=0; dhv=0;
   for (i=0;i<totp;i++)
   { if ( (pos+i)->xybig=='x' )
     { dver=1; dhor=1; goto jump; }
     if ( (pos+i)->xybig=='y' )
     { dhor=1; dver=1; goto jump; }
     if ( (pos+i)->xybig=='o' )
      for (l=0;l<180;l++)
      { xp=(int)((cirx+radi*cos(deg*l))*200.0+0.5);
        yp=(int)((ciry+radi*sin(deg*l))*200.0+0.5);
        dhor=(xp<=(pos+i)->item[2]) && (xp>=(pos+i)->item[0]);
        dver=(yp<=(pos+i)->item[3]) && (yp>=(pos+i)->item[1]);
        if (dhor==1 && dver==1) goto jump;
      }
     if ( (pos+i)->xybig=='z' )
      for (l=0;l<180;l++)
      { xp=(int)((cirx+radi*cos(deg*l))*200.0+0.5);
        yp=(int)((ciry+radi*sin(deg*l))*200.0+0.5);
        dver=(yp<=(pos+i)->item[0]) && (yp>=(pos+i)->item[1]);
        if (dver==1)
        { lef=((pos+i)->item[2]*xp+(pos+i)->item[3]*yp
              +(pos+i)->item[4]);
          rig=((pos+i)->item[2]*xp+(pos+i)->item[5]*yp
              +(pos+i)->item[6]);
          dhor=(lef>=0) && (rig<=0);
        }
        else dhor=0;
        if (dhor==1 && dver==1) goto jump;
      }
 jump: dhv=(dver && dhor);
     if ( dhv==1 ) { posmarker[j]=(pos+i)->marker; j+=1; }
     if ( j>=2 )
     { insec=check_diff_rect(j);
       if ( insec>=2 ) { conn+=1; goto jump1; }
     }
   }
 jump1:  if ( totloop == specvalu[spv] )
         {  prob=((float)conn/(float)totloop);
            fprintf(out2,"       %3.1f            %5d%c
%5d           %f \n",diam,totloop,coma,conn,prob);/***/
            spv++;  }
    if (totloop >= maxnum)  goto jump2;
```

```
 }
jump2:  if (totloop >= minnum)
 { prob=((float)conn/(float)totloop);
   fprintf(out2,"       %3.1f            %5d%c
%5d            %f \n",diam,totloop,coma,conn,prob);/****/
 }
}
fprintf(out2,"input file name is     ---> %d\n",cirf);/***/
fprintf(out2,"output file name is    ---> %s\n",outf);/****/
fprintf(out2,"%s\n",sentence);/****/
fclose(out2);
}
fclose(in3);
}
fclose(in4);
fclose(out1);
}


check_diff_rect(comm1)
int comm1;
{
   int count1,a,p,t;

   t=0;
   for (p=0;p<comm1-1;p++)
     if (posmarker[p] != posmarker[p+1]) t+=1;
   return(t+1);
}

 defect(dco)
 int dco;
{
  float    cirx1,cirx2,ciry1,ciry2;
  long     rm,rn;
  static   unsigned short xy1[3]={34,22,56};
  static   unsigned short xx1[3]={2,53,7};
  static   unsigned short xy2[3]={29,46,87};
  static   unsigned short xx2[3]={15,27,64};

   rm = nrand48(xy1);
   cirx1 = (float)( cirxL * rm / rang );
```

```
    rn = nrand48(xx1);
    ciry1 = (float)( ciryL * rn / rang );
    cirx1=cirx1-offsetx; ciry1=ciry1-offsety;
    cir1x[dco]=cirx1; cir1y[dco]=ciry1;

    rm = nrand48(xy2);
    cirx2 = (float)( cirxL * rm / rang );
    rn = nrand48(xx2);
    ciry2 = (float)( ciryL * rn / rang );
    cirx2=cirx2-offsetx; ciry2=ciry2-offsety;
    cir2x[dco]=cirx2;  cir2y[dco]=ciry2;
}
```

Appendix B
Defect Placer

```
#include <stdio.h>
#include <math.h>
#include "sced.h"
#define  PI11  0.196349541

int scell();
int prompt_fun();
float radi;
float diam;
int   num;

/* cell definition                */
CELL_DEF UCellDef =
{
   2,2,scell,prompt_fun,NULL
};

/* strechable cell routine        */

int scell()
{
    int shape();
    int write_cell();

    add_trial_line(1, 1,1, 1,1);
    add_trial_line(2, 2,2, 2,2);
    add_write_func(write_cell);
}

/* write cell routine             */

int write_cell()
{
    Init();
```

```
    BeginCell(Cell);
    allow_layer("metal");
    allow_layer("n+");
    allow_layer("p+");
    allow_layer("p_glass");
    allow_layer("al1");
    allow_layer("al2");
    allow_layer("c2");
    allow_layer("fox_cut");
    allow_layer("gate_poly");
    allow_layer("wet_etch");

    INF(Cell);
    EndCell(Cell);
    Finalize();
}

INF(cell)
char  *cell;
{
    int         i,j,k,m;
    FILE        *fp,*out,*in;
    char        unix_name[100],str[100],str1[100];
    float       a,b,x[32],y[32],cirx,ciry;
    double      Dx[32],Dy[32];
    extern FILE *pi_fopen();

    FindUnixName(FindDirEntry(cell),unix_name);
    sprintf (str,"%s/../circen", unix_name);
    fp = pi_fopen(str, "r", 0);
    if (fp == NULL)
    {
fprintf (stderr, "could not open %s\n", str);
return (1);
    }

    fprintf(stderr,"started  ");
    for(j=0;j<num;j++)
    {
        fscanf(fp,"%f %f",&a,&b);
        for (i=0;i<32;i++)
        {
```

```
            Dx[i] = (int)(200.0*a + 200.0*radi * cos (PI11*i)+0.5);
            Dy[i] = (int)(200.0*b + 200.0*radi * sin (PI11*i)+0.5);
            x[i] = DB_CONV(Dx[i]); y[i] = DB_CONV(Dy[i]);
          }
          DefinePolygon("p_glass",x,y,32);
      }
      fprintf(stderr,"finished  %d \n",j);
      fclose (fp);
}

 char  radiu[50];
 char  numb[50];

prompt_fun()
{
   char  *sp;
   double d;
   double atof();

   get_query("input diameters=\n",radiu);
   store_query(1,radiu);
   sp=radiu;
   d=atof(sp);
   diam=(float)d;
   radi=diam/2.0;
   fprintf(stderr,"\nradius = %f \n",radi);/****/
   get_query("input # of defects=\n",numb);
   store_query(1,numb);
   sp=numb;
   d=atof(sp);
   num=(int)d;
   fprintf(stderr,"# of defects = %d\n",num);/****/
}
```

Appendix C
Fault Probability for various cells

	P_{cell_1}	P_{cell_2}	P_{cell_3}	P_{cell_4}	P_{cell_5}	P_{cell_6}
1.5 μm	0.000775	0.000375	0.00015	0.00005	0.0002	0.000025
1.6 μm	0.018425	0.0077	0.00395	0.001425	0.00365	0.001125
1.8 μm	0.05425	0.02495	0.011825	0.004925	0.012275	0.003025
2.0 μm	0.09095	0.04335	0.019575	0.0094	0.021	0.005175
2.2 μm	0.129275	0.0612	0.028775	0.014275	0.029825	0.007475
2.5 μm	0.19375	0.091625	0.0448	0.0258	0.046575	0.0112
2.8 μm	0.26165	0.122425	0.06145	0.0415	0.066675	0.02085
3.0 μm	0.306	0.144725	0.0731	0.053425	0.080575	0.032925
3.2 μm	0.353125	0.167475	0.084875	0.064475	0.09475	0.044875
3.5 μm	0.42345	0.203575	0.103375	0.08435	0.1177	0.0631
3.8 μm	o.4927	0.2396	0.12415	0.10305	0.138825	0.081775
4.0 μm	0.539925	0.2658	0.138525	0.115575	0.1544	0.09485
4.2 μm	0.5863	0.29255	0.1538	0.129125	0.168975	0.108075
4.5 μm	0.65645	0.3343	0.17615	0.14905	0.1905	0.128125
4.8 μm	0.705625	0.379675	0.19655	0.16885	0.203525	0.14615
5.0 μm	0.7362	0.41	0.2107	0.182175	0.211475	0.15815
5.2 μm	0.76455	0.439775	0.22475	0.1953	0.2205	0.17085
5.4 μm	0.78975	0.469175	0.238325	0.207525	0.230625	0.1834
5.6 μm	0.8158	0.499125	0.25285	0.220225	0.24045	0.19595
5.8 μm	0.840175	0.526625	0.266725	0.23335	0.24995	0.208175
6.0 μm	0.863525	0.556525	0.279975	0.24655	0.25925	0.2197
6.2 μm	0.88125	0.58295	0.29245	0.258825	0.269375	0.231925
6.5 μm	0.901225	0.622725	0.3117	0.277275	0.283775	0.249025
6.8 μm	0.923325	0.660825	0.33225	0.296925	0.297825	0.26715
7.0 μm	0.93535	0.683675	0.34445	0.30915	0.307325	0.27475
7.2 μm	0.9465	0.707	0.356175	0.3211	0.316025	0.27895
7.5 μm	0.957575	0.735175	0.3738	0.3393	0.328725	0.285525
8.0 μm	0.970475	0.76935	0.403325	0.3638	0.3464	0.2955
9.0 μm	0.982625	0.8345	0.45685	0.4007	0.3708	0.313475

Table C.1 : Fault probability of elementary cell (R#1)

	P_{cell_1}	P_{cell_2}	P_{cell_3}	P_{cell_4}	P_{cell_5}	P_{cell_6}
12.0 μm	0.998575	0.9502	0.579275	0.485725	0.4237	0.358625
15.0 μm	0.99945	0.9948	0.668175	0.546175	0.46515	0.393925
18.0 μm	0.999975	0.9995	0.733125	0.591725	0.498275	0.42565
21.0 μm	1.0	0.999875	0.781375	0.627925	0.527775	0.45415
30.0 μm	1.0	1.0	0.873475	0.707675	0.602825	0.5479
40.0 μm	1.0	1.0	0.9245	0.7704	0.664975	0.627125
50.0 μm	1.0	1.0	0.95065	0.820425	0.71945	0.700225
60.0 μm	1.0	1.0	0.964175	0.85875	0.764675	0.75055
80.0 μm	1.0	1.0	0.9775	0.905325	0.843075	0.8072

Table C.1 : continue

	P_{cell_1}	P_{cell_2}	P_{cell_3}	P_{cell_4}	P_{cell_5}	P_{cell_6}
1.5 μm	0.00085	0.000225	0.00025	0.000025	0.00025	0
1.6 μm	0.018075	0.007875	0.00365	0.0016	0.004325	0.000875
1.8 μm	0.055225	0.02485	0.01105	0.005225	0.011925	0.0028
2.0 μm	0.093	0.044125	0.01915	0.00955	0.0202	0.004675
2.2 μm	0.130525	0.0637	0.028325	0.01385	0.0294	0.0068
2.5 μm	0.19065	0.092975	0.04325	0.02545	0.046175	0.010875
2.8 μm	0.262125	0.1243	0.0609	0.042125	0.0652	0.020875
3.0 μm	0.3084	0.146125	0.0726	0.054125	0.079375	0.03335
3.2 μm	0.353875	0.17045	0.0841	0.0662	0.092475	0.04545
3.5 μm	0.42745	0.206125	0.102725	0.085325	0.11425	0.065675
3.8 μm	0.49845	0.2436	0122775	0.10415	0.135675	0.085225
4.0 μm	0.545475	0.26995	0.1365	0.11725	0.15055	0.0983
4.2 μm	0.59125	0.2973	0.1517	0.130325	0.165	0.1108
4.5 μm	0.663375	0.33965	0.1747	0.150175	0.187625	0.1301
4.8 μm	0.712025	0.386175	0.19335	0.170275	0.20135	0.149325
5.0 μm	0.74235	0.416575	0.20645	0.182925	0.2095	0.161675
5.2 μm	0.76845	0.445275	0.221144	0.196025	0.2186	0.174175
5.4 μm	0.79355	0.475375	0.234669	0.2089	0.22735	0.1878
5.6 μm	0.8187	0.504875	0.248175	0.223775	0.237	0.200125
5.8 μm	0.844225	0.534225	0.26185	0.236025	0.24715	0.212125
6.0 μm	0.865725	0.5623	0.275	0.2491	0.25715	0.223475
6.2 μm	0.883525	0.5888	0.28785	0.26105	0.267425	0.23555
6.5 μm	0.9025	0.628	0.308475	02789	0.282975	0.2523
6.8 μm	0.9232	0.6654	0.3289	0.29825	0.296975	0.27105
7.0 μm	0.9357	0.688925	0.3424	0.30985	0.30665	0.279625
7.2 μm	0.9474	0.710075	0.3539	0.3209	0.31505	0.284075
7.5 μm	0.958525	0.740975	0.37025	0.339575	0.328325	0.29025
8.0 μm	0.970975	0.7745	0.399575	0.365425	0.345825	0.29945
9.0 μm	0.98	0.839325	0.456275	0.401875	0.3715	0.31755
12.0 μm	0.99875	0.9504	0.58245	0.48605	0.4238	0.36025
15.0 μm	0.999325	0.994825	0.669775	0.545225	0.4635	0.394875
18.0 μm	0.99995	0.999675	0.734125	0.59155	0.495975	0.4248
21.0 μm	1.0	0.999875	0.78185	0.6282	0.52545	0.45125
30.0 μm	1.0	1.0	0.872925	0.708475	0.60215	0.5436
40.0 μm	1.0	1.0	0.9272	0.77345	0.664875	0.62405
50.0 μm	1.0	1.0	0.950425	0.824075	0.71655	0.700125
60.0 μm	1.0	1.0	0.96495	0.861	0.763175	0.749475
80.0 μm	1.0	1.0	0.9773	0.9075	0.8435	0.80805

Table C.2 : Fault probability of elementary cell (R#2)

	P_{cell_7}	$P_{cell_{1_t}}$	P_{ther}	P_{exp}	P_{ther}+offset
1.5 μm	0	0.000925	0.000436	0.00665	0.000436
1.6 μm	0	0.018675	0.010300	0.01595	0.010408
1.8 μm	0	0.0555	0.030729	0.0383	0.031053
2.0 μm	0.000075	0.094675	0.051717	0.060575	0.052258
2.2 μm	0.00015	0.132925	0.073721	0.0832	0.074739
2.5 μm	0.0004	0.1981	0.111185	0.120875	0.112919
2.8 μm	0.0036	0.2631	0.151289	0.160825	0.153771
3.0 μm	0.00555	0.308275	0.178494	0.188625	0.181584
3.2 μm	0.00755	0.3543	0.206980	0.217075	0.210676
3.5 μm	0.01035	0.425425	0.250238	0.257975	0.254846
3.8 μm	0.0142	0.49805	0.293342	0.304	0.298861
4.0 μm	0.01655	0.544525	0.323056	0.33425	0.329182
4.2 μm	0.018975	0.5909	0.352628	0.362875	0.359361
4.5 μm	0.02255	0.661275	0.397271	0.407	0.404916
4.8 μm	0.02635	0.70845	0.43172	0.4412	0.440275
5.0 μm	0.029025	0.73935	0.453777	0.463175	0.46294
5.2 μm	0.031325	0.766075	0.474890	0.48355	0.48466
5.4 μm	0.033575	0.793275	0.494482	0.504125	0.504859
5.6 μm	0.035625	0.8183	0.514714	0.523825	0.525699
5.8 μm	0.037975	0.8415	0.533774	0.543975	0.545366
6.0 μm	0.0406	0.865025	0.552424	0.563175	0.564624
6.2 μm	0.043325	0.879975	0.568110	0.578525	0.580917
6.5 μm	0.04685	0.901225	0.588660	0.5998	0.602378
6.8 μm	0.050175	0.9229	0.610377	0.6199	0.625006
7.0 μm	0.052325	0.935225	0.622655	0.6337	0.637891
7.2 μm	0.054725	0.94565	0.634136	0.6453	0.649979
7.5 μm	0.0577	0.957	0.648142	0.660625	0.664896
8.0 μm	0.06365	0.9685	0.667084	0.6785	0.685356
9.0 μm	0.0677	0.982375	0.695813	0.7072	0.695813
12.0 μm	0.0804	1.0	0.752913	0.762725	0.752913
15.0 μm	0.093075	1.0	0.785851	0.7948	0.785852
18.0 μm	0.1042	1.0	0.807715	0.8195	0.807715
21.0 μm	0.118125	1.0	0.824353	0.83865	0.824354
30.0 μm	0.171525	1.0	0.861191	0.87995	0.861191
40.0 μm	0.2391	1.0	0.886263	0.912325	0.886263
50.0 μm	0.30915	1.0	0.904334	0.93405	0.904334
60.0 μm	0.377375	1.0	0.916572	0.948625	0.916572
80.0 μm	0.4975	1.0	0.932125	0.9672	0.932125

Table C.3 : Various fault probability (R#1)

	P_{cell_7}	$P_{cell_{1_6}}$	P_{ther}	P_{exp}	P_{ther}+offset
1.5 μm	0	0.000725	0.000476	0.00595	0.000476
1.6 μm	0	0.019175	0.010135	0.016475	0.010243
1.8 μm	0	0.0558	0.030958	0.03825	0.031282
2.0 μm	0	0.091725	0.052528	0.0607	0.053069
2.2 μm	0.000025	0.1308	0.074336	0.0824	0.075354
2.5 μm	0.0004	0.195825	0.109539	0.12005	0.111272
2.8 μm	0.0037	0.26295	0.151503	0.15995	0.153985
3.0 μm	0.0.005725	0.3082	0.179585	0.1878	0.182675
3.2 μm	0.0.008125	0.353925	0.207423	0.21575	0.211119
3.5 μm	0.0115	0.4248	0.252158	0.2611	0.256765
3.8 μm	0.014675	0.493625	0.29614	0.3044	0.301659
4.0 μm	0.017075	0.54135	0.325617	0.334625	0.331743
4.2 μm	0.019775	0.58815	0.354876	0.3641	0.361609
4.5 μm	0.023075	0.6586	0.400617	0.409775	0.408261
4.8 μm	0.026625	0.70685	0.434745	0.444225	0.443299
5.0 μm	0.029125	0.736025	0.456484	0.46625	0.465646
5.2 μm	0.0317	0.765075	0.476606	0.486075	0.486376
5.4 μm	0.034125	0.790725	0.496235	0.50795	0.506613
5.6 μm	0.0366	0.816225	0.515874	0.5284	0.526859
5.8 μm	0.03915	0.840125	0.535584	0.54775	0.547176
6.0 μm	0.04175	0.865275	0.553228	0.566475	0.565427
6.2 μm	0.044225	0.8819	0.569019	0.582525	0.581825
6.5 μm	0.0477	0.903925	0.589382	0.6037	0.603100
6.8 μm	0.050875	0.9237	0.610409	0.6244	0.625038
7.0 μm	0.0534	0.9355	0.623311	0.636325	0.638547
7.2 μm	0.0557	947225	0.634717	0.647525	0.650561
7.5 μm	0.05865	0.9599	0.64879	0.661775	0.665544
8.0 μm	0.063775	0.970925	0.667408	0.680775	0.685681
9.0 μm	0.0691	0.982525	0.69547	0.711625	0.695470
12.0 μm	0.0828	1.0	0.753848	0.76775	0.753848
15.0 μm	0.094975	1.0	0.78605	0.797825	0.786050
18.0 μm	0.107475	1.0	0.807688	0.820425	0.807688
21.0 μm	0.12025	1.0	0.824056	0.8406	0.824056
30.0 μm	0.172775	1.0	0.860704	0.883475	0.860704
40.0 μm	0.2393	1.0	0.886723	0.913725	0.886723
50.0 μm	0.306825	1.0	0.90424	0.935575	0.904240
60.0 μm	0.37325	1.0	0.916645	0.949775	0.916646
80.0 μm	0.491225	1.0	0.932289	0.9689	0.932289

Table C.4 : Various fault probability (R#2)

	P_{total} (R#1)	$P_{total_{exp}}$ (R#1)	P_{total} (R#2)	$P_{total_{exp}}$ (R#2)
1.5 μm	0.000317	0.006225	0.000265	0.003754
1.6 μm	0.006637	0.012150	0.006772	0.01155
1.8 μm	0.019945	0.024600	0.019899	0.024134
2.0 μm	0.034021	0.037925	0.033125	0.038299
2.2 μm	0.048184	0.051400	0.047622	0.051991
2.5 μm	0.072471	0.075325	0.071664	0.076400
2.8 μm	0.098600	0.100925	0.098559	0.102091
3.0 μm	0.117161	0.119975	0.117152	0.120442
3.2 μm	0.135987	0.138550	0.135950	0.138857
3.5 μm	0.165250	0.167500	0.165160	0.168575
3.8 μm	0.195142	0.196575	0.194043	0.197130
4.0 μm	0.214841	0.215050	0.214001	0.216981
4.2 μm	0.234663	0.235375	0.233914	0.236358
4.5 μm	0.264638	0.263700	0.264047	0.266675
4.8 μm	0.287555	0.287275	0.287211	0.289252
5.0 μm	0.302726	0.302150	0.301767	0.303928
5.2 μm	0.316753	0.316400	0.316456	0.317217
5.4 μm	0.330841	0.329400	0.330159	0.331798
5.6 μm	0.344481	0.342475	0.343812	0.345481
5.8 μm	0.357339	0.355200	0.356954	0.358344
6.0 μm	0.370318	0.368200	0.370276	0.3725
6.2 μm	0.380597	0.379050	0.381072	0.381971
6.5 μm	0.395666	0.393350	0.396523	0.396575
6.8 μm	0.410994	0.407400	0.411274	0.410731
7.0 μm	0.419988	0.416175	0.420381	0.419035
7.2 μm	0.428190	0.424375	0.428747	0.426881
7.5 μm	0.438849	0.435025	0.439807	0.438025
8.0 μm	0.468933	0.465162	0.469679	0.466598
9.0 μm	0.454924	0.462179	0.455489	0.464971
12.0 μm	0.492470	0.498260	0.493011	0.501431
15.0 μm	0.514054	0.519546	0.514215	0.521454
18.0 μm	0.528749	0.536178	0.528739	0.536761
21.0 μm	0.540288	0.549308	0.540100	0.550539
30.0 μm	0.566673	0.578510	0.566366	0.580734
40.0 μm	0.585984	0.602429	0.586275	0.603312
50.0 μm	0.600879	0.619628	0.600819	0.620590
60.0 μm	0.612092	0.632316	0.612139	0.633042
80.0 μm	0.628890	0.651020	0.628993	0.652093

Table C.5 : Fault probability of the total chip

REFERENCES

[1] C. H. Stapper, "Defect Density Distribution for LSI Yield Calculations" IEEE Trans. on Electron Devices, vol. ED-20, pp. 655-657, July 1973.

[2] C. H. Stapper, "Modeling of Integrated Circuit Defect Sensitivities" IBM J. Develop. Vol. 27, NO. 6, pp. 549- 557, Nov. 1983.

[3] C. H. Stapper,"Modeling of Defects in Integrated Circuit Photolithographic Patterns" IBM J. DEVELOP. Vol. 28, NO. 4, pp. 461-473, July 1984.

[4] D. M. H. Walker, "Yield Simulation for Intergrated Circuits" Kluwer Academic Publishers, Boston, Dordrecht, Lancaster, 1987.

[5] I. M. Sobol "The Monte Carlo Method" Mir Publishers, English translation, 1975.

[6] I. Chen and A. J. Strojwas , "Realistic Yield Simulation for VLSIC Structural Failures" IEEE Trans. Computer-Aided Design vol. CAD-6, NO. 6, pp. 965-980, Nov. 1987.

[7] R. Y. Rubinstein "Simulation and The Monte Carlo Method" Wiley, New York, 1981.

[8] R. E. Walpole and R. H. Myers "Probability and Statistics for Engineers and Scientists" third edition, Macmillan Publishing Company, New York, 1978.

[9] B. T. Murphy, "Cost-Size Optima of Monolithic Integrated Circuit" Proceedings of The IEEE, pp. 1537-1545, Dec. 1964.