

YIELD ANALYSIS ON EMBEDDED MEMORY
WITH REDUNDANCY

By

NIN JING

Bachelor of Medicine

Nanjing University

Nanjing, P.R.China

1995

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December 2002

YIELD ANALYSIS ON EMBEDDED MEMORY

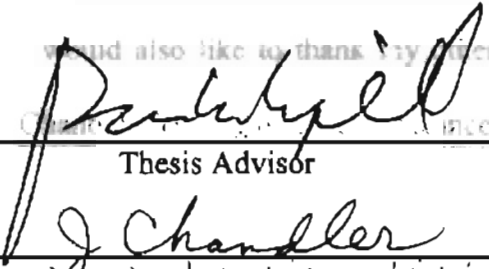
WITH REDUNDANCY

express my sincere appreciation to my advisor, Dr. N. Park for his intellectual support, constructive criticisms, inspiration and support throughout my thesis work. His offering suggestions and patient guidance lead to the completion of my thesis with

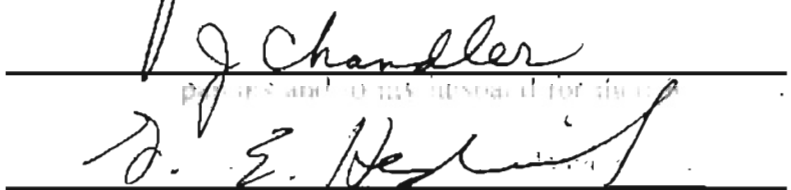
Thesis Approved: Park is a dedicated scholar and has advised me

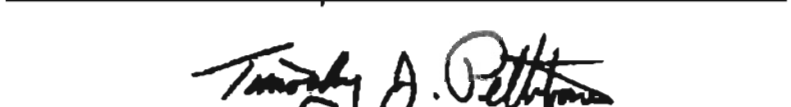
future graduate
of the Graduate
College

would also like to thank my other committee members
for their assistance and helpful comments



Thesis Advisor







Dean of the Graduate College

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to my advisor, Dr. N. Park for his intelligent supervision, constructive criticisms, inspiration and support throughout my thesis work. His enlightening suggestions and patient guidance lead to the completion of my thesis within half a year. Dr. Park is a dedicated scholar and has advised me in my future career endeavors. I would also like to thank my other committee members, Dr. G.E. Hedrick and Dr. J.P. Chandler. Their guidance, assistance and helpful comments are invaluable

I am grateful to my parents and to my husband for their love, understanding and encouragement. I would also like to acknowledge my daughter, Catherine. This thesis is dedicated to my family.

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to my advisor, Dr. N. Park for his intelligent supervision, constructive criticisms, inspiration and support throughout my thesis work. His enlightening suggestions and patient guidance lead to the completion of my thesis within half a year. Dr. Park is a dedicated scholar and has advised me in my future career endeavors. I would also like to thank my other committee members, Dr. G.E. Hedrick and Dr. J.P. Chandler. Their guidance, assistance and helpful comments are invaluable

I am grateful to my parents and to my husband for their love, understanding and encouragement. I would also like to acknowledge my daughter, Catherine. This thesis is dedicated to my family.

TABLE OF CONTENTS

Chapter		Page ⁱ
I.	INTRODUCTION	1
II.	PRELIMINARIES AND REVIEW	5
	2.1 Fault Distribution Models	5
	2.1.1 Random Fault Distribution	5
	2.1.2 Clustered Fault Model	5
	2.2 Notations Used in This Work	6
	2.3 Assumptions Made in This Research	7
	2.3.1 Assumption for Random Fault Model	8
	2.3.2 Assumption For Clustering Fault Model	8
	2.4 Generalized Architectural Models	9
III.	YIELD ANALYSIS	13
	3.1 Yield Analysis Under Random Fault Distribution Model	13
	3.1.1 Yield Analysis in Single-region Memory Array	13
	3.1.2 Yield Analysis in Multi-region Memory Array	14
	3.1.3 Yield Analysis in Multi-region Memory with Hierarchical Active Redundancy	16
	3.2 Yield Analysis Under Fault Clustering Model	18
IV.	PARAMETRIC SIMULATION	20
	4.1 Simulation with The Faults Under Random Distribution	20
	4.2 Simulation with The Clustered Fault Model	26
V.	DISCUSSION	31
	5.1 Overhead for Multi-region Memory Architectural Model	31
	5.2 Yield Comparison between Random Fault Distribution and Fault Clustering	32
VI.	CONCLUSION	34
	REFERENCES	36
	GLOSSARY	40

APPENDIX A: THE BOOLEAN EXPRESSION ON THE ROW/COLUMN DELETION IN THE SINGLE-REGION MEMORY	41
APPENDIX B: CODE FOR YIELD ANALYSIS IN SINGLE-REGION MEMORY ARRAY UNDER RANDOM FAULT DISTRIBUTION	45
LIST OF FIGURES	
APPENDIX C: CODE FOR YIELD ANALYSIS IN MULTI-REGION MEMORY ARRAY UNDER RANDOM FAULT DISTRIBUTION	47
APPENDIX D: CODE FOR YIELD ANALYSIS IN MULTI-REGION MEMORY ARRAY WITH HIERARCHICAL REPAIR UNDER RANDOM FAULT DISTRIBUTION	51

LIST OF FIGURES

Figure	Page
1. Three Different Memory Architecture Models	12
2. A Multi-region Memory Model With $n = 2$ And $m = 4$	15
3. A Multi-region Memory Model With $n > 2$ And $m > 2$	17
4. Relationship Between Size Of Memory Array And Yield Under Random Fault Distribution	21
5. Relationship Between Number of Spares And Yield Under Random Fault Distribution	22
6. Yield In Different Repair Under Random Fault Distribution	23
7. Yield Analysis in SR, MR and MRH ($n=4$)	24
8. Relationship Between Memory Size and Yield Under Fault Clustered Distribution With Different Redundancy	28
9. Relationship Between Number of Redundancy And Yield Under Fault Clustered Distribution With Memory Size of 64	28
10. Yield In Different Repair Under Fault Clustered Distribution	29
11. Yield In SR, MR and MRH ($n=4$) Under Fault Clustered Distribution	29

LIST OF TABLES

Table	Page
1. Simulation Parameters (I)	21
2. Yield Analysis Results Of Failure Rate 0.01 Under Fault Random Distribution	22
3. The Yield Analysis Results of 64 x 64 Single –region Memory Array Under $p = 0.01$	29
4. The Yield Analysis Results In Three Different Reconfigurable Memory Models Under Fault Random Distribution With $p = 0.2%$, $n = 2$ And $s = 3$	24
5. Yield Analysis Result In SR, MR And MRH Under Fault Random Distribution When $p = 0.2%$, $n = 4$ And $s = 5$	25
6. Simulation Parameters (II)	27

CHAPTER I replacing it. In order to salvage a
INTRODUCTION and columns to fault rows and

Advances in very-large scale integration (VLSI) and wafer-scale integration (WSI) technology have made possible the manufacturing of large density memory chips. With integrated-circuit densities of RAMs (Random Access Memory) increasing, the probability of defects in them also increases. For viable yield, these chips must be designed to be defect tolerant.

The general architecture under consideration is a RAM subdivided into modules that are linked together by an interconnection structure. The individual modules are memory cell arrays with independent control units. In a partitioned RAM system, defect tolerance can be achieved in two ways [2]. The first approach is to add local redundancy in the form of spare rows/columns to the modules of the RAM. A second option is to have global redundancy in the form of spare modules to replace defective ones. Purely local redundancy in the spare row/column approach is good for array cell and purely global redundancy in the form of spare modules requires the use of a spare module for even a single -cell failure in any module. This approach leads to an excessive number of spares for any reasonable yield, thus, making the area overhead high and redundancy utilization low.

Therefore, redundancy in the form of additional (spare) rows and columns has been extensively used to reconfigure faulty memory arrays and enhance the yield [4]. In this application, spare rows and columns are added to a memory chip containing a programmable address decoder. Faulty rows and columns from the primary portion of the array are then replaced by spare rows and columns by programming the address decoder

to map addresses meant for the faulty row or column replacing it. In order to salvage a faulty array however, an assignment of spare rows and columns to faulty rows and columns must be found which eliminates all faulty cells. This is known as the array reconfiguration problem. [5]

As early as 1986, Kuo and Fuchs [4] showed that the problem of reconfiguring redundant RAM (RRAM, i.e. a random access memory with spare rows/columns) is NP-complete for functions with finite set size. This is equivalent to state that the detection and diagnosis problem can be solved correctly with 100% confidence (no uncertainty region) only when the fully exhaustive search for finding the repair-solution has been completed. Thus heuristic algorithms are likely to be the major viable approach to solve this problem. Several key heuristic algorithms have been developed to guide the process of searching for a solution. They can be classified broadly into three categories, namely, must-repair analysis, irreparability tests, and row/column selection heuristics. Given a defective RRAM, a Must-Repair algorithm [14] is used to find the set of must-repair rows and columns and remove them. If there are not enough spares to cover the must-repair rows and columns, the array is irreparable, determined by irreparability tests. Irreparability tests can be performed in two ways according to the specific conditions: fault count after Must-Repair [15] or Early Abort test [4]. A Row/Column selection algorithm [2] iteratively replaces the row or column that contains the most number of faults with a spare row or spare column, until all faults are eliminated or it runs out of spares. Based on the above heuristics, more algorithms are developed to reconfigure defective RRAMs faster and more effectively. A fast greedy heuristic algorithm [16] and the f^* -test [8] were presented to test for repairability with an improved bound for

detecting irreparability. Shen and Lombardi [17] proposed a new efficient and heuristic approach to detect reparability/irreparability for memory chips with redundancy. The main benefit of this approach is its practicality with respect to fast execution time and the improved ability to diagnose a VLSI redundant memory before the generation of the repair-solution. Most recently, a linear search algorithm was proposed for repair analysis with 4 spare rows/4 spare columns [18]. Unfortunately, all existing repair algorithms are either those which take polynomial time but may not find a solution even if one exists or are exhaustive tree-search algorithms with a worst case exponential time complexity to find a solution if one exists. Hence, An algorithm-free structure function is desired to solve the problem of memory reconfiguration.

In this thesis, an algorithm-free structure function was developed to measure extensively the yield of a fault-tolerant embedded memory system. Consequently, the relationship has been investigated among the size of the memory module, the number of spare row/column and the yield under certain distribution models at which memory faults may occur. The more accurate prediction of the reliability, the conditional probability that the system performs correctly under specific system failure rate would make manufacturing of more dependable embedded memory system in terms of reliability and system yield while maintaining minimal amount of redundancy come true.

The remainder of this thesis is divided into the following chapters. In Chapter II, preliminaries related to this research work and generalized architectural models of reconfigurable embedded memory with redundancy will be introduced and discussed. Yield analysis based on the proposed architectural models and different fault distribution models will be described in Chapter III. Parametric simulations and their results will be

shown in Chapter IV. The overhead of the architectural models and the results obtained from the parametric simulations will be discussed in Chapter V. Finally, conclusion and future work will be given in Chapter VI.

CHAPTER II

PRELIMINARIES AND REVIEW

2.1 Fault Distribution Models

Two major probabilistic models have been adopted for fault distribution. One is random fault distribution and the other is clustered fault model.

2.1.1 Random Fault Distribution

The simplest model of defect spatial distribution would be to assume that defects are randomly distributed. Shi and Fuchs have shown a probabilistic model in which faults occur independently [21]. In this model, each single fault is assumed to be limited to a single element and each fault appears with equal probability $p(n)$ and faults are statistically independent of each other. Because of its simplicity, this model has been most used to analyze the memory configuration [22]. Random fault distribution, however, is less accurate when the average defect densities were determined while the intensity of that distribution varies from wafer to wafer. Therefore, such a model has long been known to poorly predict the yield of chips of proposed [23].

2.1.2 Clustered Fault Model

Defects in VLSI circuits tend to occur in clusters due to defects that span multiple circuit elements [24], [25]. This physical phenomenon is referred to as defect clustering. There is a distinction between physical defects and circuit faults. A defect is any imperfection on the wafer, but only the fraction of defects that actually affect the circuit

operation are called faults. In this research, modeling the distribution of faults is concentrated on rather than that of defects in order to estimate yield.

The accuracy of the estimated yield depends on the statistical model selected to describe the spatial distribution of manufacturing faults. The spatial fault distribution and, in particular, the manufacturing fault clustering depends on the dimension of the chips. Fault clusters in integrated circuits can be roughly categorized into four classes [26]:

- the first class includes clusters much larger than the chip size (large size clustering);
- the second class of clusters deals with fault clusters that are smaller than the chip area (small size clustering);
- the third class of clusters deals with fault clusters with dimension similar to that of the chip area (medium size clustering);
- the fourth class of fault clusters deals with clusters that vary in dimension.

Different models have been proposed for these classes of fault clusters. Among these models, the negative binomial distribution has been introduced because it is the best to fit the experimental data in the case of large-size clustering as well as small-size fault clustering [27]. Unified yield method [26] and the center-satellite model [28] have been used in the presence of medium size clustering. These models, however, are based on parameters and assumptions that are difficult to determine and use [28].

2.2 Notations used in this work

- A_N an $N \times N$ memory array with N rows, N columns and N^2 elements
- N number of rows or columns in a single-region memory array, excluding

spares

- s : number of spare rows or columns
- $a_{i,j}$ an element of A_N
- p fault probability within an memory array
- K the total number of possible faulty cells in a memory array
- r_i a row in A_N
- c_j a column in A_N
- m number of memory modules (blocks) in a multi-region memory array
- n number of rows or columns in a memory module (block) in a multi-region memory system, excluding spares
- λ average number of faults per chip
- α the clustering parameter for the considered area
- Y manufacturing yield

2.3 Assumptions made in this research

- Faulty cells occurring in a memory array can be locally tolerated by field-reconfiguration of the given spare row/column redundancy.
- Only stuck-at faults are considered. In a memory array, there could be two types of faults, a stuck-at fault and a coupling fault. A stuck-at fault is said to occur at cell i if the logic value of the cells is always at 0 or 1, while a coupling fault $i \rightarrow j$ is said to occur if a read or write operation at address i forces the contents of cell j to a certain value, 0 or 1. In the coupling fault, cell i is called the coupling cell and cell j is the coupled cell. Hence, with regard to these two different types of

fault cells, the requirement for repairing is different also. To repair stuck-at faults, the row or column where they are located may be replaced with a spare row or spare column, respectively. As to coupling faults, either the coupled cell or the coupling cell may be replaced. If the coupling cell is replaced, the repair must ensure that the coupling cell is not accessed, even internally to the chip. In most DRAMs (Dynamic Random Access Memory), this can be done by disabling the row (word line), and not by disabling the column (bit line). Thus a coupling fault is repaired by replacing the coupled cell with a spare row or column, or by replacing the coupling cell with a spare row [6]. In recent years, the problem of reconfiguring RRAMs with stuck-at faults has attracted a great deal of research [6] [7] [8] [9].

- The number of rows and columns in the memory array (block) are equal.
- The number of spare rows and columns in the memory array or block are equal.
- No solution cost is considered regarding to memory reconfiguration.
- Number of spare rows and spare columns grows linearly with N .

2.3.1 Assumptions for Random Fault Model

- The faults are under a probabilistic model in which all faults are independently random distributed, as described in [6].
- Each fault appears with equal probability of p and p is a constant with $0 < p < 1$.

2.3.2 Assumptions for Clustered Fault Model

- The number of faults is distributed according to the negative binomial distribution.
- Faults are uniformly distributed on the chip area.
- Only Fault Prone (denoted by FP) chip is considered on a wafer.
- Single fault clustering is considered.
- Although there are several fault types originating from manufacturing defects, only single cell fault is considered for the simplicity.

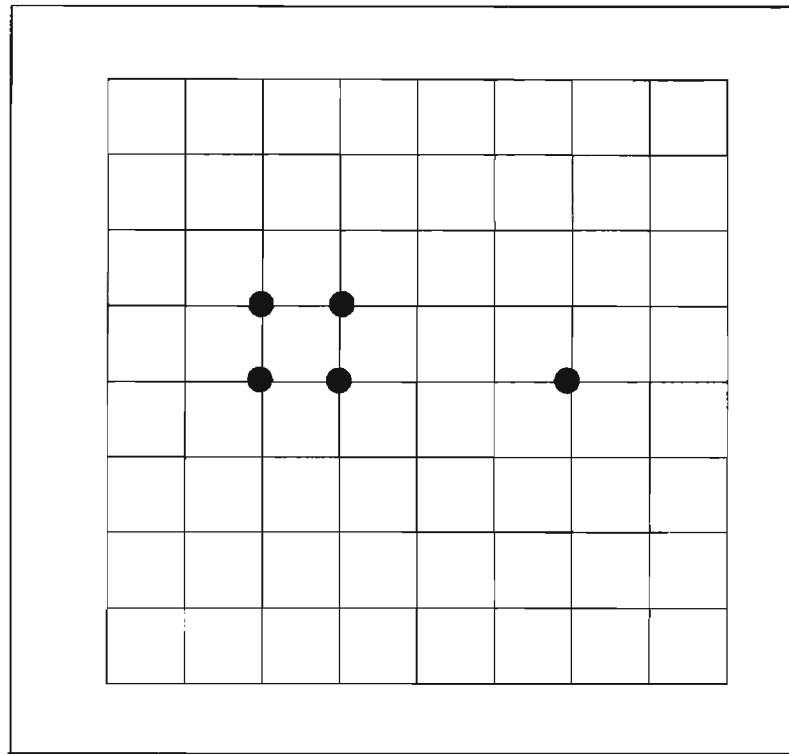
2.4 Generalized architectural models

In this thesis, generalized architectural models for reconfigurable memory with redundancy based on the idea of memory arrays cutting as stated in [19] are given in Figure 1. Therefore, 3 architectural models are considered in this paper. Characteristics of these models can be summarized as follows.

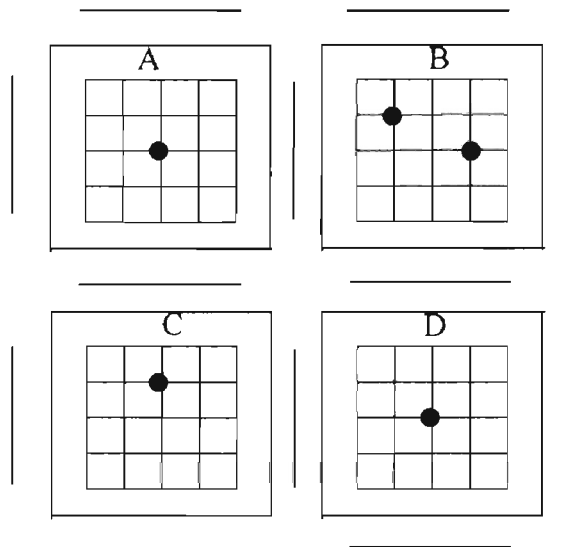
- Figure 1(a), which is called single-region memory array model, contains $N \times N$ cells. There are 5 defective elements which can be located at any of the N^2 memory cells, namely random distribution. Each fault occurs independently of other fault(s). This allows the possibility of overlapping defects, complicating the analysis somewhat but making the model more realistic. The system can be reconfigured by line replacement, which is called row/column deletion. In this case, one row and two columns replacement can reconfigure the faulty memory, i.e. the memory is repairable/fault-tolerant. There are other ways, e.g. two rows replacement could be used but this is not considered in this research as stated in the assumption.

- In a multi-region memory model, shown in Figure 1(b), a memory array is partitioned onto several modules (blocks) with different repair probabilities. Each module may have several spare elements (redundant rows and/or columns) to repair defective cells encountered in the block. Some spare elements may be common for the neighboring blocks, which are called shared spares. Others which are specific to only one block are called local spares.
- Multi-region memory with hierarchical redundancy as shown in Figure 1(c), local spares are like those in the multi-region memory system. However, shared spares are split into 2 portions, each of which can repair half of one row/column. It is expected that when a spare line is divided into segments and then used to repair faulty cells on the different row (or columns) in the original memory array, it will be highly beneficial in the repair process of RRAMs.

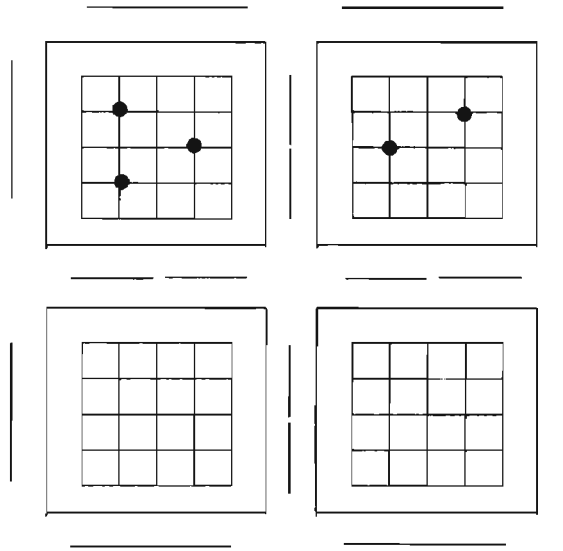
For faulty memory arrays containing spare rows and columns, some of the most challenging problems are the achievement of acceptable yield and the minimization of redundancy area [8] [19]. When the failure rate is too low compared to the number of spares, then the array can be trivially repaired, but the spares are wasted. However, when the failure rate is too high compared to the number of spares, then the array is almost never repairable. Effective use of redundancy requires the modeling of the trade-offs between yield enhancement and the level of redundancy. Thus, appropriate balancing of the acceptable yield and minimal redundancy area is desirable for high-reliable, low-cost manufacturing of memory arrays. Multi-region memory model with redundancy and multi-region memory with hierarchical active redundancy as shown in Figure 1(b) and 1(c) can be obtained from single-region memory model with redundancy by cutting spare



a



b



c

Figure 1. Three different memory models studied in this paper. Figure 1(a) is single-region memory array with redundancy; 1(b) is multi-region memory with redundancy; 1(c) is multi-region memory with active active hierarchical redundancy.

lines very accurately into segments with the advances in laser and integration technologies. It will be meaningful to compare the yield among these three models to determine the optimal memory design of trade-offs between yield enhancement and the level of redundancy.

CHAPTER III

YIELD ANALYSIS

3.1 Yield analysis under random fault distribution model

3.1.1 Yield Analysis in Single-region Memory Array

In $N \times N$ single-region memory array, it uses row and column sparing as the means of providing s spare rows and s spare columns of memory integrated circuits (ICs). If a memory IC fails, then the row/column containing that IC is eliminated from the system and replaced with a spare row/column. Therefore, the process of reconfiguration requires that the fault be detected, located and successfully removed from the system so that the system can still be operable. As long as all the faulty cells are located within s rows and columns, the memory system will be able to tolerate these failures.

Lemma 1: in a given $N \times N$ memory array, the reliability of the system is $Y = 1$ when $K \leq s$.

Proof: K faulty cells at most can be distributed within K rows/columns when $K < N$ or N rows/columns when $K \geq N$. When $K \leq s$, all the fault containing lines can be replaced by the spares, therefore, Y is 1.

Lemma 2: In $N \times N$ memory array, when $s \geq N$, $Y = 1$ without regarding p .

Proof: It is obvious that when there are more spares than the size of memory array, all the rows/columns can be replaced by the spares to recover the system operable.

When the number of failures is more than s , only those with positions within s rows and s columns can be reparable. The amount of the combinations of all K failures within in 1 row/column in $N \times N$ array is

$$T_1 = \binom{N}{1} \cdot \binom{N}{K} \quad (1)$$

Also, the number of the combinations of all K failures located within 1 row/columnn

$$C_1 = \binom{N}{K} \quad (2)$$

The number of the combinations of all K failures within 2 rows/columns in N x N array is

$$T_2 = \binom{N}{2} \cdot \left(\binom{N \cdot s}{K} - \binom{2}{1} C_1 \right) \quad (3)$$

The number of the combinations of all K failures within s rows/columns in N x N array is

$$T_s = \binom{N}{s} \cdot \left(\binom{N \cdot s}{K} - \sum_{i=1}^{s-1} \binom{s}{i} C_i \right) \quad (4)$$

The probability of repairable combinations of K failures is

$$Y = \left(\sum_{i=1}^s T_i / (N \cdot N) \right)^2 \quad (5)$$

3.1.2 Yield Analysis in Multi-region Memory Array

In $(n \times m) \times (n \times m)$ multi-region memory array, the memory array is partitioned into m modules of same size. Each module is a single-region memory system, which can uses the segments of spare row/column to support the system fault-tolerant. Considering numerous combinations of potential failures spatial location, multi-region system should be beneficial in the repair process.

Lemma 3: In the multi-region system with n equal to 2 and $s \geq m + 1$, the reparability $Y=1$ no matter the size of m and p .

Proof: As shown in Figure 2, there are 2 rows and 2 columns in each module. Around each module, there are 4 either local or shared spares. Hence, the number of spares given in each module is more than the size of the module, based on the lemma 2. $Y=1$.

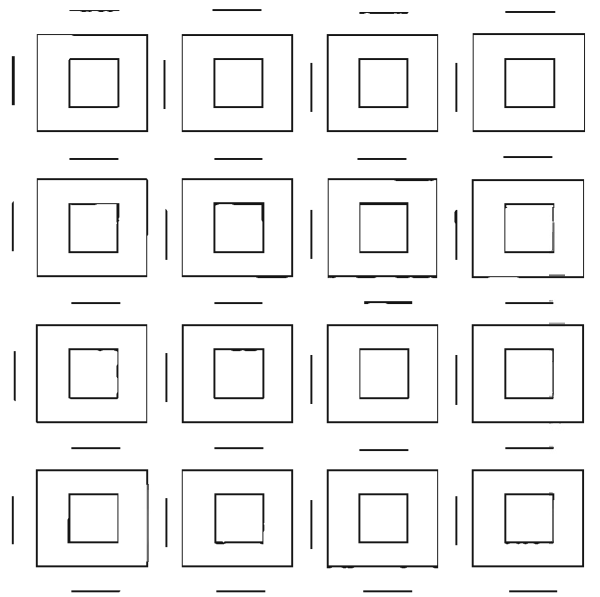


Figure 2: A multi-region memory model with $n = 2$ and $m = 4$. There are totally 5 spare rows and 5 spare columns.

When $n > 2$, $m=2$, there are totally 4 modules. Each module has 2 local spares and 2 shares spares, as shown in Figure 1(b). Yield analysis is based on the comprehensive calculation. The procedure is as following:

- (1). List all the spatial position combinations of K faulty cells. For instance, 6 faulty cells, it could be that 1 cell in module A, 2 cells in module B, 2 cells in module C and

1 cell in module D. It also could be 1 cell in module A, 3 cell in module B, 2 cell in module C and none in module D. The reliability is different under different combination.

(2). Analyze the yield of every combination.

$$Y_i = Y_A \cdot Y_B \cdot Y_C \cdot Y_D$$

(3). Calculate the yield of the system

$$Y = \sum_{i=1}^t Y_i / t$$

While t is total number of the combinations given the K faults and m modules.

When $n > 2$ and $m > 2$, there are 3 groups of modules which share the commonality. As shown in Figure 3, group 1 has 2 local spares and 2 shared ones. Group 2 has only 1 local spare and 3 shared one. While in Group 3, each module has none local but 4 shared spares. Therefore, the process of yield analysis is:

- (1). Given K failures, list all the combinations that K failures could located in the 3 groups.
- (2). In each group, analyze the yield just as above mentioned when $n > 2$ and $m = 2$;
- (3). Calculate the Y for the whole system.

3.1.3 Yield Analysis in Multi-region Memory with Hierarchical Active Redundancy

In $(n \times m) \times (n \times m)$ multi-region memory with hierarchical active redundancy, the local spares are same as those in multi-region memory array. However, the shared spares are cut into 2 segments. When a line containing less than $n/2$ faulty cells, it will be replaced by the segment of shared spares first. Consequently, for the modules with local

spares, the repair is at the two levels depended on the number of failures on a line. The first level is spare segments and the second is spare line.

The yield analysis is similar to that in multi-region memory system except adjustment of s for different groups of modules based on the hierarchy levels.

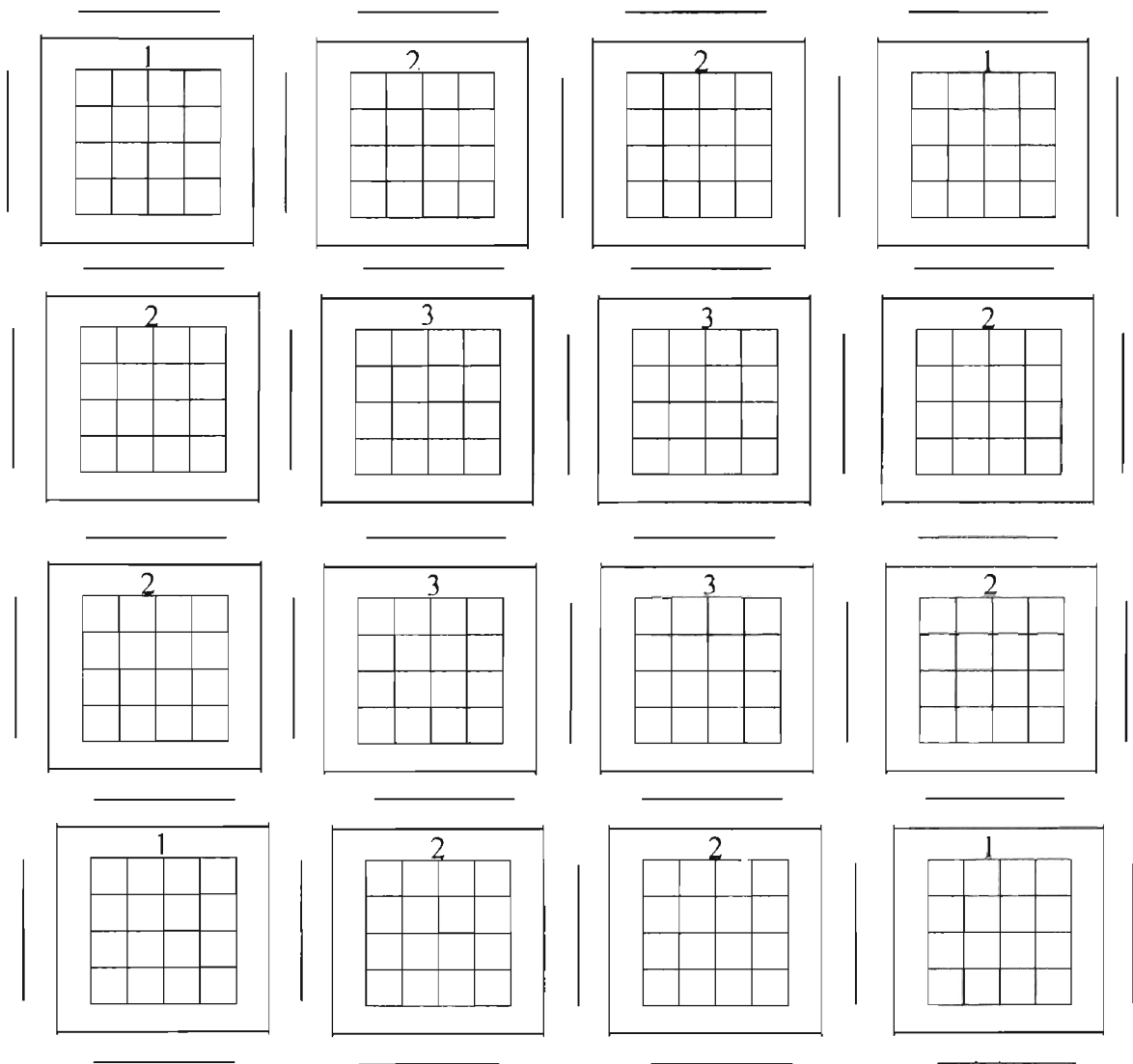


Figure 3. Multi-region memory model with $n > 2$ and $m > 2$. There are totally 5 spare rows and 5 spare columns in the system.

3.2 Yield analysis under clustered fault model

The negative binomial distribution was developed during the 1980's for estimating the yield of memory chips with redundant word and bit lines, i.e. redundant rows and columns [29]. It is a compounding distribution derived from the Poisson and Gamma distributions. There are two parameters in negative binomial distribution. They are the mean λ and a clustering parameter α . As α increases, the negative binomial distribution approaches the Poisson distribution. As α decreases, the negative binomial distribution approaches a distribution yielding low values with near certainty and high value rarely (the distribution grows an abrupt, long, thin tail).

The object for negative binomial distribution is the estimation of the probabilities of all the fixable combinations of different failures. The resulting probabilities are added because the fixable combinations are mutually exclusive events [30].

Similar formula is used in this work as stated in [31]. According to the notation presented in Chapter II-2.1, denoting the chip's yield can be written as follows:

$$Y = \sum_{i=1}^k \text{prob}(FP_i) = \sum_w \text{prob}(FP_i) \text{prob}(FP_i \text{ is fixable}) \quad (6)$$

FP_i is the i th fault pattern which can be fixed in a RRAM. k is the total number of fixed fault patterns which could occur. $p(K)$ is the probability that K faults are found on the chip at the end of the manufacturing process, which can be expressed by the generalized negative binomial statistics [32]:

$$p(K) = \frac{\Gamma(K + \alpha)}{K! \Gamma(\alpha)} \frac{(\lambda / \alpha)^K}{(1 + \lambda / \alpha)^{K + \alpha}} \quad (7)$$

Using equation (7) we now turn to a further expression derived from equation (6), that is

$$Y = \sum_{K=0}^{\infty} p(K) * prob(FP \text{ is fixable}) \quad (8)$$

The formula (8) consists of two terms. The latter one can be obtained from Chapter III – 3.1 determined by different memory architectures because the faults are under random distribution within a memory chip as stated in the assumptions.

The truncation of the infinite sum in formula (8) can affect error ϵ level. With certain value of λ and α we using in formula (8), the bigger K we truncate the sum at will give rise to a smaller error ϵ . Therefore, it is worthwhile to select an approximation level ϵ and derivation of the truncation level K to compute the yield.

Lemma 4: when the number of fault is 0, that is $K = 0$, all the fault patterns can be fixed. (Proof is obvious since no fault occurs in a chip, the yield is 1).

Consequently, the formula (8) can be rewritten as follows:

$$Y = p(0) + \sum_{K=1}^{\infty} p(K) * prob(FP \text{ is fixable}) \quad (9)$$

and

$$p(0) = \frac{1}{(\lambda / \alpha)^{\alpha}} \quad (10)$$

CHAPTER IV

PARAMETRIC SIMULATION

4.1 Simulation with the faults under random distribution

The effect of the different memory model, memory size and redundancy amount on the reliability of the fault-tolerant memory system is studied through numerical experiments in this section. Simulation was performed using a Sun Microsystems Enterprise 3000 in the Oklahoma State University Computer Science Department. Parameters used in this simulation are summarized in Table 1. Parametric simulation results are given in Figures 4 to 7 and specific values are shown in Tables 2 to 5 where 3 different memory reconfiguration models are compared with regards to the reconfiguration efficiency. For example, under the failure rate 0.2% and 5 spare rows and columns, when the memory size is 4K, the reliability in single-region memory system is $1.01E-7$, in multi-region memory system which is partitioned into 16 modules, it is 0.0093 and in multi-region memory with hierarchical active redundancy it is increased into 0.127435. The reconfiguration efficiency in multi-region memory with hierarchical active redundancy is very significant. The following observations can be obtained from the results:

- I. In all the three reconfiguration memory models, under failure rate of 1% and specific amount of the spares, system yield decreases with memory size increasing.
- II. The increment of the spares can significantly enhance the system yield in all the three reconfiguration memory models. The trade-off, however, is the overhead of demanding more system resource.

III. Compared to single-region memory reconfiguration model, the efficiency is significantly increases in multi-region memory model. This effect is

Table 1
SIMULATION PARAMETERS (I)

Parameter	Meaning	Value
n	Row/column size of memory module	2-64
m	Module number	2-32
S	Number of spare rows/columns	variable
P	Probability of fault rate	$\leq 1\%$
n^2	Module size	4-4K
$(n \cdot m)^2$	Memory array size	4-4K

especially obvious with the memory size increasing. As the observation I, memory size has a dramatic effect on the system reliability given the failure rate and the amount of spares. While in multi-region memory model, this effect is becoming less dramatic because of shared spares between neighboring modules and more flexible reconfiguration strategy with segments of the spares instead of one line to replace the fault line in the memory module.

IV. Multi-region memory with hierarchical active redundancy shows the most beneficial effect on sustaining the higher level of the system yield. It inherits the shared spares between neighboring modules in multi-region memory with redundancy. Moreover, the replacement becomes flexible

Relationship between Size of Memory Array and Yield Under Fault Random Distribution ($p=0.01$)

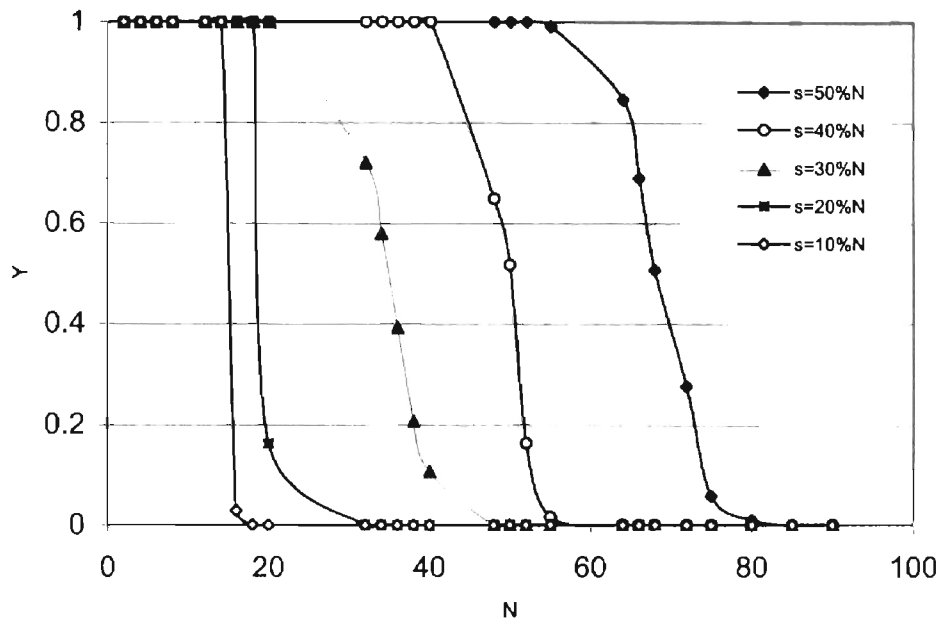


Figure 4: Yield analysis results of a single-region memory array with different size and different number of the spare rows and columns under random distribution of faults. (The failure rate $p = 0.01$)

Size of memory (N)	Yield (Y)				
	s = 10%N	s = 20%N	s = 30%N	s = 40%N	s = 50%N
2	1	1	1	1	1
4	1	1	1	1	1
8	1	1	1	1	1
16	0.0289	1	1	1	1
32	5.99E-12	8.96E-4	7.21E-2	1	1
64	2.02E-63	2.32E-32	1.76E-12	8.32E-4	8.46E-1

Table 2. Yield analysis results of failure rate 0.01 under fault random distribution.

Relationship between the number of spares and yield under random fault distribution (N=64, p=0.01)

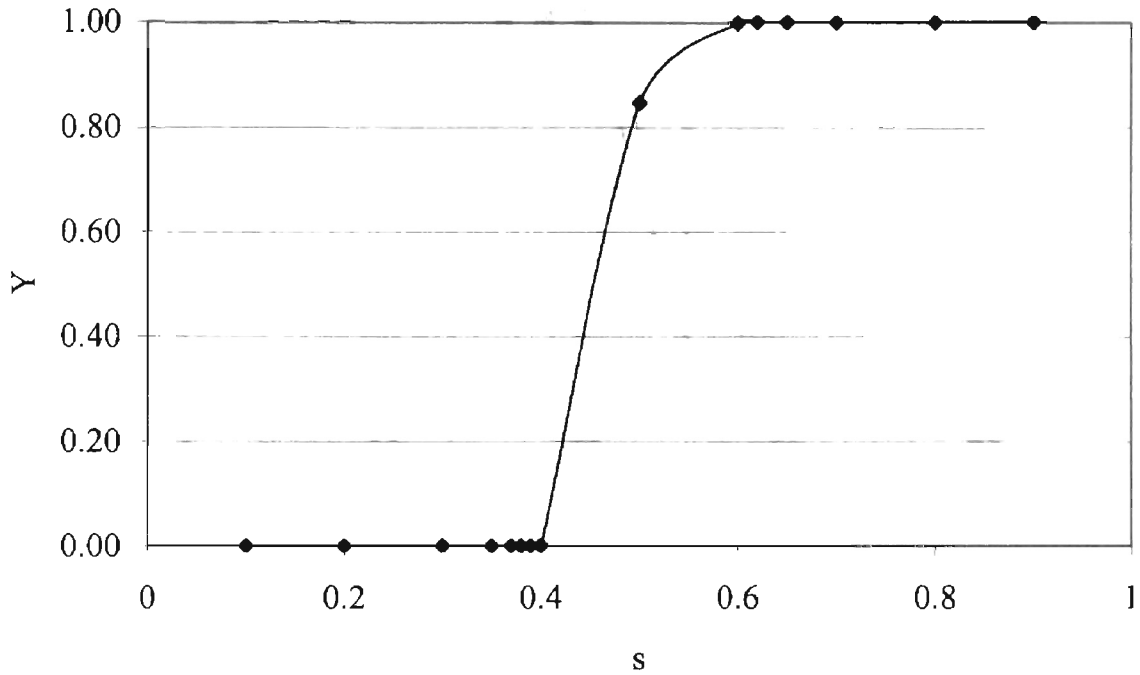


Figure 5. The yield analysis results of 64 x 64 single-region memory array under random fault distribution with $p = 0.01$. The values on X axis refer to the ratio between the number of spares (s) to the size of the memory array (N).

Ratio of S over N	Yield (Y)
10%	2.02E-63
20%	2.32E-32
30%	1.76E-12
40%	8.32E-4
50%	8.46E-1
60%	9.98E-1
70%	1
80%	1
90%	1
100%	1

Table 3. The yield analysis results of 64 x 64 single-region memory array under $p = 0.01$.

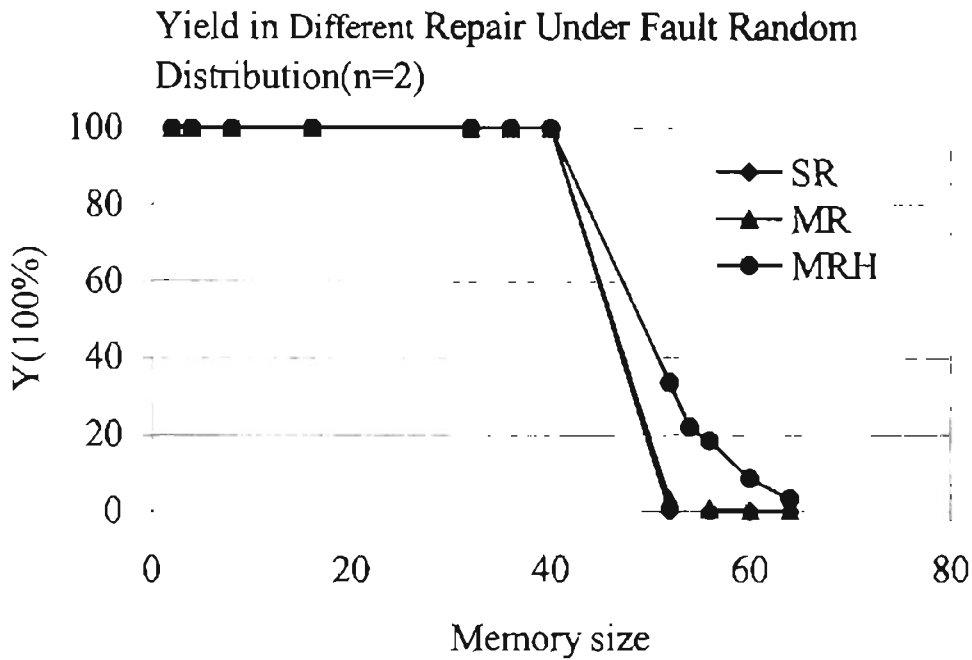


Figure 6. Yield analysis in single-region, multi-region and multi-region with hierarchical active redundancy memory models under fault random distribution. In the latter two models, there are 4 modules partitioned from the memory array. The failure rate $p = 0.2\%$ and $s = 3$.

Memory size	Yield (Y)		
	Single Region	Multi Region	MR Hierarchy
2	1	1	1
4	1	1	1
8	1	1	1
16	1	1	1
32	1	1	1
52	3.17E-7	0.0323	0.3326
56	6.83E-10	0.00638	0.1837
60	1.058E-12	7.26E-4	0.08583
64	1.2474E-15	1.54548E-5	0.03322

Table 4. The yield analysis results in three different reconfigurable memory models under fault random distribution with $p = 0.2\%$, $n = 2$ and $s = 3$.

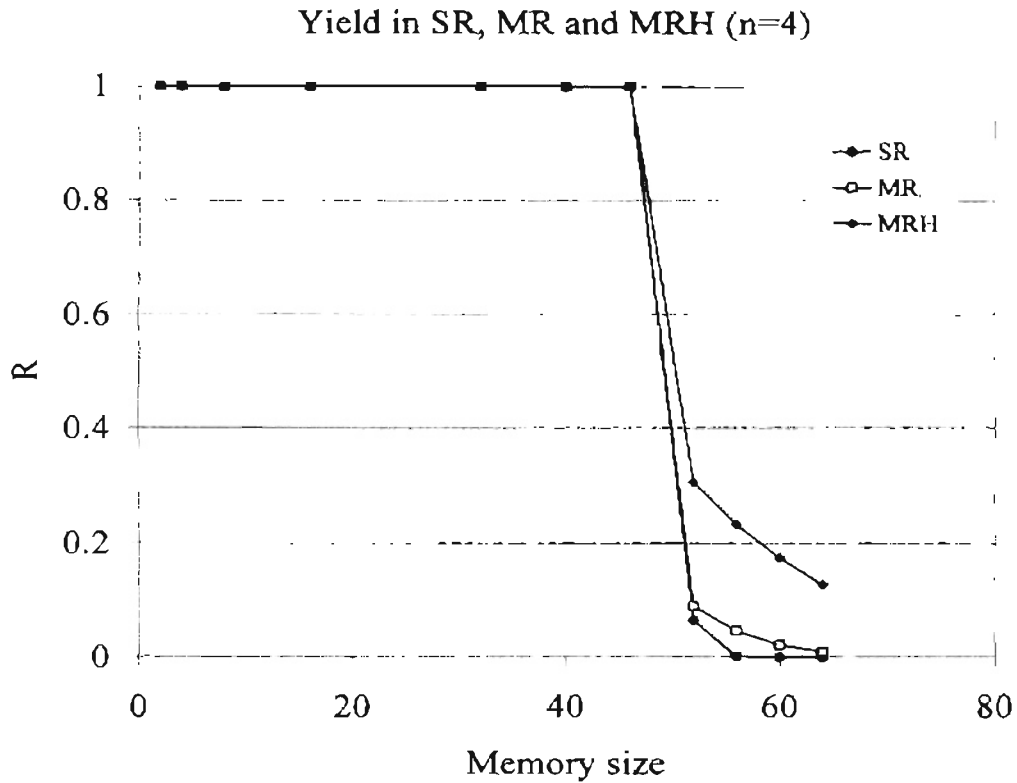


Figure 7. Yield analysis in single-region, multi-region and multi-region with hierarchical active redundancy memory models under fault random distribution. In the latter two models, there are 16 modules partitioned from the memory array. The failure rate $p = 0.2\%$ and $s = 5$.

Memory size	Yield (Y)		
	Single Region	Multi Region	MR Hierarchy
2	1	1	1
4	1	1	1
8	1	1	1
16	1	1	1
32	1	1	1
52	0.06475	0.090124	0.3070
56	0.001415	0.0462	0.2329
60	1.589E-5	0.02168	0.1744
64	1.01E-7	0.0093	0.127435

Table 5. Yield analysis result in SR, MR and MRH under fault random distribution when $p = 0.2\%$, $n = 4$ and $s = 5$.

even within a memory module. The latter improvement makes the reconfiguration model of multi-region memory with hierarchical active redundancy outstanding regarding to the reconfiguration efficiency. The advantage from this model is more significant with the memory size increasing.

Intelligent exploitation of the proposed measurements estimation technique makes possible the designing and manufacturing of balanced onboard memory system satisfying reliability while maintaining minimal amount of redundancy. Since the limitation of the computer resource, even bigger memory array, for instance, more than 4K, are not analyzed in this research. However, the trend derived from this paper shows multi-region memory reconfiguration model with hierarchical active model is definitely an optimal design to support high yield without exhausting much system resource.

4.2 Simulation with the clustered fault model

Under the fault clustering distribution with the number of faults modeled with negative binomial statistics in a wafer and faults randomly distributed in the chips, the relationships among the different memory architectural models, memory size, number of redundancy and the yield of the fault-tolerant memory system are also studied by the simulation. Parameters used in the simulation are summarized in Table 6. Among these parameters, α (the frequency distribution of the number of faults per chip) and γ (the average number of faults for the product chip) are critical for the binomial negative statistics. A value of $\alpha = 2$ has been found to be a good compromise over a long period of time for modeling the yields [30]. Since only small-size clustering is considered in this

simulation, λ is set to 2. The values for the Gamma function in binomial negative distribution are obtained from [36].

Table 6
SIMULATION PARAMETERS (II)

Parameter	Meaning	Value
n	Row/column size of memory module	2-64
m	Module number	2-32
S	Number of spare rows/columns	variable
n^2	Module size	4-4K
$(n \cdot m)^2$	Memory array size	4-4K
α	Frequency distribution of the number of faults	2
λ	Average number of faults for the product chip	2
K^*	Truncation level	10

Parametric simulation results are given in Figures 8 to 11. The relationship between the yield and the number of redundant row/column in a 64 x 64 single-region memory array under fault clustered distribution model, which was shown in Figure 9, fits the Figure 1 in [33]. The following observations were derived from results in these studies.

- I. Similar to the random distribution model of failures, the system yield decreased with memory size increasing. However, the effect of memory

Relationship between memory size and yield under fault clustered distribution with different redundancy

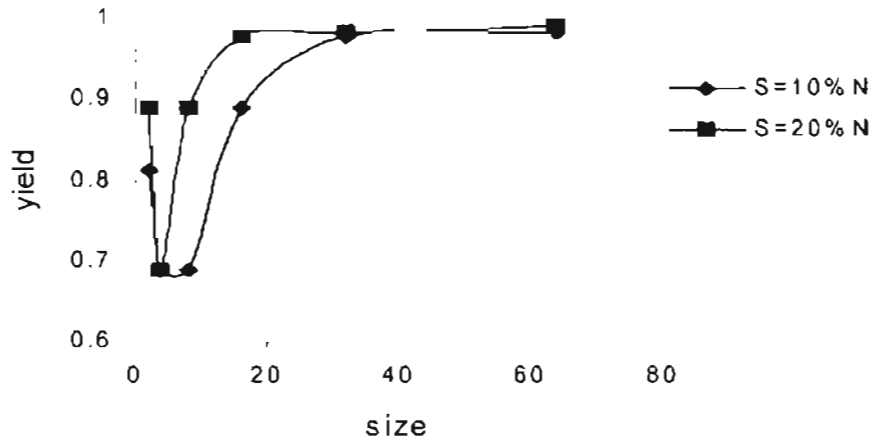


Figure 8. Yield analysis results of a single-region memory array with different size and different number of redundancy under fault clustered distribution model.

Relationship between number of redundancy and yield under fault clustered distribution with memory size of 64

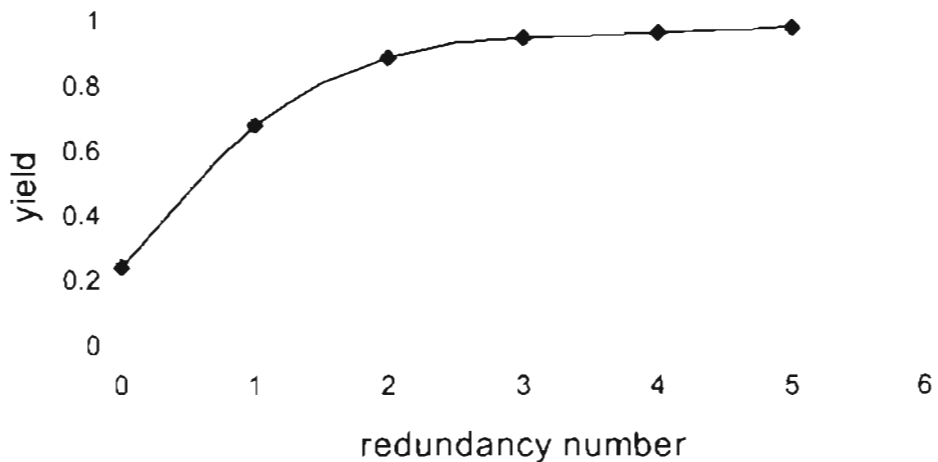


Figure 9. The yield analysis results of 64 x 64 single-region memory array under fault clustered distribution model.

Yield in different repair under fault clustered distribution

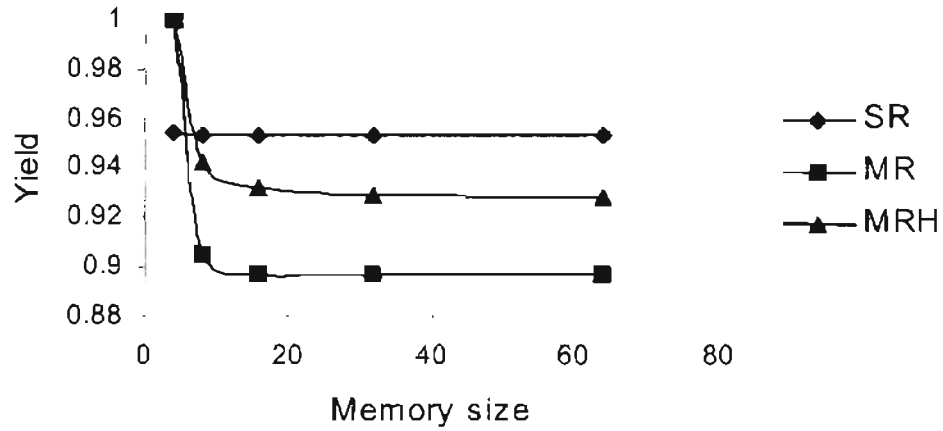


Figure 10. Yield analysis in single-region (SR), multi-region (MR) and multi-region with hierarchical active redundancy (MRH) memory models under fault clustered distribution. In the latter two models, there are 4 modules partitioned from the memory array.

Yield in SR, MR and MRH (n=4) under fault clustered distribution

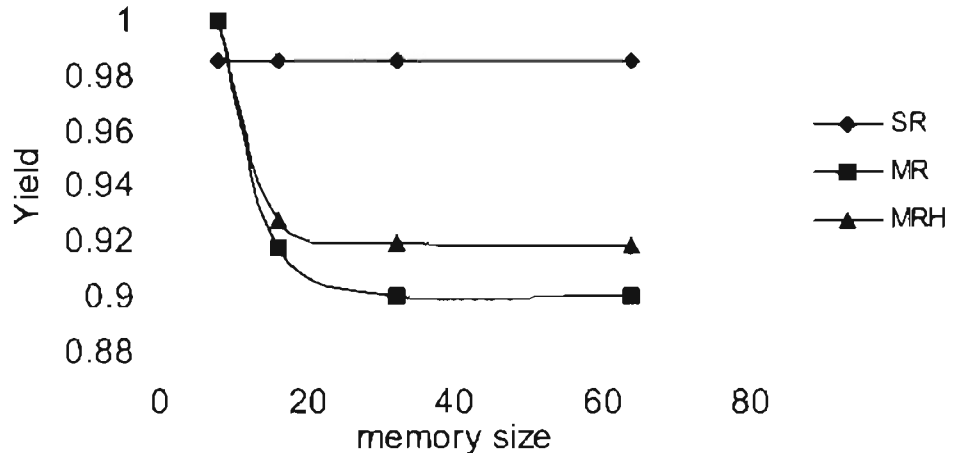


Figure 11. Yield analysis in single-region (SR), multi-region (MR) and multi-region with hierarchical active redundancy memory models under fault clustered distribution. In the latter two models, there are 16 modules partitioned from the memory array.

size on system yield is less significant than that in fault random distribution model.

- II. The increment of the spares can significantly improve the system yield in all the three reconfiguration memory models.
- III. Contrary to the result derived in the fault random distribution model, the efficiency of reconfiguring single-region memory array is the highest among all the three memory systems under fault clustering distribution. Multi-region memory array with active hierarchical redundancy still can perform better than multi-region memory array, but cannot compete with single-region memory system when the number of faults is under binomial negative distribution and only within a memory chip, those faults are distributed under unified distribution.

Among all the factors, memory size, partition of the memory array, hierarchical spares and the number of redundancy, the number of redundancy plays the most important role in improving the system yield under the fault clustered distribution model.

CHAPTER V

DISCUSSION

5.1 Overhead for multi-region memory architectural model

In a single region RRAM with s redundant rows/columns, address comparators are required to realize the row/column deletion in presence of defective word/bit addresses. These addresses are programmed in the address comparators and compared with the input address. Thus at most s defective normal bitlines/wordlines can be repaired.

When a memory array is divided into subarrays (modules), two approaches can be used for row/column replacement. The first approach is denoted as simultaneous replacement. In this approach, the number of address comparators should be equal to the number of spare row/column in a subarray. Each address comparator compares only the intra-subarray address signals and the output is commonly supplied to all the subarrays. The inter-subarray address signals in turn to select one of the spare rows/columns. In this approach, to replace one defective normal line, all the other normal lines with the same intra-subarray address are also replaced even if they are not defective. This causes two problems associated with this approach. First, the usage efficiency of spare line is lower, and the number of spare lines should be larger, which results in chip-area increase. Second, the probability of unsuccessful repair due to defects in the spare lines that replaced normal lines is higher, which results in yield degradation.

The second approach is an individual replacement. Each spare line in every subarray has its own address comparator. The number of address comparators is therefore the value of multiplication of the number of spare row/column in a subarray and

the number of subarrays in the multi-region memory system. Each address comparator compares both intra- and inter- subarray address signals. This approach, however, has the disadvantage of lower usage efficiency of address comparators, resulting in an increase in the area of address comparators.

5.2 Yield comparison between random fault distribution and fault clustering

To predict the manufacturing yield, it is required for the knowledge of the average number of faults, the location fault distribution (that is probabilistic quantity) and the spare allocation problem. As is known, in presence of redundancies for a real VLSI system, a spare distribution is a sequence of different kinds of spare. It is optimal when covers the higher number of faults with the lower number of spares, which is called optimal spare allocation problem and in general, it is a NP-complete problem. In literature, different fault distribution models have been taken with the trade-off of the level of accuracy and ease of use.

The two fault distribution models studied in this work have been widely used and considered as one of the statistics that best fits experimental data. The random fault statistics provides the probability of chip acceptability given the presence of a set of manufacturing faults. The negative binomial statistics fault model, however, takes into account the clustering phenomena. It is shown in this research in both fault distribution models, the number of redundancy can significantly improve the system yield. But as we know, the introduction of too many spares definitely increases the system overhead. In real world, even spares cannot be fault-free completely, which in turn, could not sustain a high stable yield level with paying the cost of high overhead.

Second common between random fault distribution model and fault clustered distribution model is that memory size exerts negative effect on the system yield. Under same fault occurring rate, the larger the memory size, the less the yield can be enhanced in presence of spares. However, the size of RAM quadruples almost every two or three years. It is impractical to restrict the size of memory to try to achieve higher yield.

In this research, different memory architectural models have been introduced in order to search a new strategy to improve the yield. Although with the overhead mentioned in Chapter 5.1, it is obvious that under fault random distribution model, multi-region with hierarchical active redundancy is outstanding to sustain a higher level of yield compared to the common single-region memory system in presence of limited number of spares. The flexibility derived from both shared spares between the neighboring modules and replacement of segments of the spares, however, becomes less important in presence of clustered faults. Yet it is clear that multi-region with hierarchical active redundancy is a good strategy to improve the system yield with less overhead as long as the faults distribution is under random statistics.

CHAPTER VI

CONCLUSION

This thesis work presented yield measurement and estimation technique for fault-tolerant embedded memory system under random and clustered fault distribution. By exploring architecture-driven methods rather than repair algorithm-driven methods, yields of three different reconfigurable memory architectures, i.e. single-region memory architecture, multi-region memory architecture and multi-region memory with hierarchical redundancy architecture are compared to investigate the optimal design and manufacturing of reliable embedded memory system. According to the simulation results given, the multi-region reconfigurable memory with hierarchical redundancy demonstrated the best yield under random fault distribution among the three architectures considered. Multi-region memory also shows the beneficial effect on sustaining the higher level of the system yield compared to the single-region memory.

Under clustered fault distribution, the multi-region memory with hierarchical redundancy does not perform as satisfying as that under random fault distribution. Single-region memory architecture displays the highest yield among all the three architectures. Multi-region memory system with hierarchical redundancy is still shown to be more reliable than multi-region memory system. However, the flexibility provided by both hierarchical redundancy and multi-region memory does not contribute significantly to the improvement of the system yield under clustered fault distribution.

Based on the results obtained from this thesis, the work can be extended to the larger size memory to investigate more practical yield analysis among these three memory architecture models under different fault distributions in the future.

BIBLIOGRAPHY

1. Fuchs, W.K. and Chang M., "Diagnosis and Repair of Large Memories: A Critical Review and Recent Results," *Proc. Intl. Workshop on Defect and Fault Tolerance in VLSI Systems, Plenum Press, New York*, pp.24-37, February 1987.
2. Ganapathy, K.N., Singh, A.D. and Pradhan, D.K., "Yield optimization in large RAM's with hierarchical redundancy", *Solid-State Circuits, IEEE Journal of* , Volume: 26 Issue: 9, Page(s): 1259 –1264, Sept. 1991.
3. Shen, Y. N. and Lombardi, F., "Repairability / unrepairability Detection Technique for Yield Enhancement of VLSI Memories with Redundancy", *Computers and Digital Techniques, IEE Proceedings E [see also Computers and DigitalTechniques, IEE Proceedings-]* , Volume: 137, Issue: 2 , Page(s): 133 – 136, March 1990.
4. Kuo, S.Y. and Fuchs, W.K., "Efficient spare allocation in reconfigurable arrays", *Proc. ACM/IEEE DAC*, 1986.
5. Blough, D.M. and Pelc, A, " A clustered failure model for the memory array reconfiguration problem", *Computers, IEEE Transactions on*, Volume: 42 Issue: 5 , Page(s): 518 –528, May 1993.
6. Low, C.P. and Leong, H.W. "Probabilistic analysis of memory reconfiguration in the presence of coupling faults", *Defect and Fault Tolerance in VLSI Systems, 1992. Proceedings., 1992 IEEE International Workshop on* , Page(s): 157 – 166,1992.

7. Che, W. and Koren, I., "Fault spectrum analysis for fast spare allocation in reconfigurable arrays ", *Defect and Fault Tolerance in VLSI Systems, 1992. Proceedings., 1992 IEEE International Workshop on* , Page(s): 60 -69 1992.
8. Low, C.P. and Leong, H.W. , "A new class of efficient algorithms for reconfiguration of memory arrays", *Computers, IEEE Transactions on* , Volume: 45 Issue: 5 , Page(s): 614 –618, May 1996.
9. Choi, M., Park, N., Meyer, F.J., Lombardi, F. and Piuri, V., "Reliability measurement of fault-tolerant onboard memory system under fault clustering", *Instrumentation and Measurement Technology Conference, 2002. IMTC/2002. Proceedings of the 19th IEEE*, Volume: 2, Page(s): 1161 –1166, 2002.
10. Lombardi, F., Park, N., Al-Hashimi, M. and Pu, H.H., "Modeling the dependability of N-modular redundancy on demand under malicious agreement" , *Dependable Computing, 2001. Proceedings. 2001 Pacific Rim International Symposium on*, Page(s): 68 –75, 2001.
11. Choi, M. and Park, N. "Dynamic yield analysis and enhancement of FPGA reconfigurable memory system", *Instrumentation and Measurement Technology Conference, 2001. IMTC 2001. Proceedings of the 18th IEEE*, Volume: 1, Page(s): 386 -391 vol.1, 2001.
12. Park, N. and Lombardi, E. "Repair of memory arrays by cutting", *Memory Technology, Design and Testing, 1998. Proceedings. International Workshop on* , Page(s): 124 –130, 1998.

13. Shen, Y.N., Park, N. and Lombardi, F. , “Spare cutting approaches for repairing memories” , *Computer Design: VLSI in Computers and Processors, 1996. ICCD '96. Proceedings., 1996 IEEE International Conference on* , Page(s): 106 – 111,1996.
14. Evans, R.C., “Testing repairable RAMs and mostly good memories”, *Proc. IEEE International Test Conference*, Page(s): 49-55, 1981.
15. Tarr, M., Boudreau, D. and Murphy, R. “Defect Analysis System Speeds Test and Repair of Redundant Memories,” *Electronics*, Page(s): 175-179, January, 1984.
16. Leong, H.W. and Low, C.P., “New results on fault covering in RRAMs”, *Circuits and Systems, 1991., IEEE International Symposium on* , Page(s): 2148 -2151 vol.4, 1991.
17. Shen, Y. N. and Lombardi, F., “Repairability/unrepairability detection technique for yield enhancement of VLSI memories with redundancy” , *Computers and Digital Techniques, IEE Proceedings E [see also Computers and DigitalTechniques, IEE Proceedings-]* , Volume: 137 Issue: 2 , Page(s): 133 – 136, March 1990.
18. Kwon, H., Moon, J., Byun, J., Park, S. and Chung, J., “Linear search algorithm for repair analysis with 4 spare row/4 spare column”, *ASICs, 2000. AP-ASIC 2000. Proceedings of the Second IEEE Asia Pacific Conference on*, Page(s): 269 –272,2000.

19. Park, N. and Lombardi, E., "Repair of memory arrays by cutting", *Memory Technology, Design and Testing, 1998. Proceedings., International Workshop on*, Page(s): 124 –130, 1998.
20. Shi, W. and Fuchs, W.K., "Probabilistic analysis and algorithms for reconfiguration of memory arrays", *Computer-Aided Design of Integrated Circuits and System, IEEE Transactions on*, Volume: 11 Issue: 9, Page(s): 1153 – 1160, Sept. 1999.
21. Murphy, B.T. , "Cost-size optima for monolithic integrated circuits", *Proc. IEEE*, Volume: 52, Page(s): 1537-1545, Dec. 1996.
22. Blough, D.M., "Performance evaluation of a reconfiguration-algorithm for memory arrays containing clustered faults", *IEEE Transactions on Reliability*, Volume: 45, Page(s): 274-284, June 1996.
23. Koren, I., Koren, Z. and Stapper, C. H. , "A unified negative binomial distribution for yield analysis of defect tolerance circuits", *IEEE Transactions on Computers*, Volume: 42, NO 6, June 1993.
24. Stapper, C. H., "Large-area fault clustering in VLSI circuits: a review", *IBM Journal of Research and Development*, Volume: 33, NO 2, March 1989.
25. Meyer, F. J. and Pradhan, D. K. , "Modeling defect spatial distribution", *IEEE Transactions on Computers*, Volume: 38, NO. 4, April 1989.
26. Stapper, C.H., McLaren, A.N. and Dreckmann, M. , "Yield model for productive optimization of VLSI memory chips with redundancy and partially good product", *IBM Journal of Research and Development*, Volume. 24, Page(s): 398-409, May 1980.

27. Stapper, C.H., "Improved yield models for fault-tolerant memory chips", *IEEE Transactions on Computers*, Volume: 42 Issue: 7 , Page(s): 872 –881, July 1993.
28. Ciciani, B. and Iazeolla, G., "A Markov chain-based yield formula for VLSI fault-tolerant chips", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Volume: 10 Issue: 2 , Page(s): 252 –259, Feb. 1991.
29. Stapper, C.H., Armstrong, F. M. and Saji, K., "Integrated circuit yield statistics", *Proc. IEEE*, Volume: 71, Page(s): 453-469, Apr. 1983.
30. Battaglini, G. and Ciciani, B., "An improved analytical yield evaluation method for redundant RAM's", International Workshop on Memory Technology, Design and Testing, 1998. Proceedings. , Page(s): 117 –123, 1998.
31. Battaglini, G. and Ciciani, B., "Realistic yield-evaluation of fault-tolerant programmable logic arrays", *IEEE Transactions on Reliability*, Volume: 47 Issue: 3 Part: 1, Page(s): 212 –224, Sept. 1998.
32. Stapper, C.H., "Small-area fault clusters and fault-tolerance in VLSI circuits", *IBM Journal of Research and Development*, Volume: 33, Page(s): 174-177, Mar. 1989.
33. Abramowitz, M. and Stegun, I.A. , " Handbook of mathematical functions, with formulas, graphs, and mathematical tables", 1965.

ABBRECIATIONS AND ACRONYMS

VLSI	Very-large scale integration
WSI	Wafer-scale integration
RAM	Random Access Memory
RRAM	Redundant RAM (, i.e. a random access memory with spare rows/columns
DRAM	Dynamic Random Access Memory
FP	Fault Prone
SR	Single –region memory array model with redundancy
MR	Multi-region memory model with redundancy
MRH	Multi-region memory with hierarchical redundancy
IC	Memory integrated circuits

APPENDIX A

THE BOOLEAN EXPRESSION ON THE ROW/COLUMN DELETION IN THE SINGLE-REGION MEMORY

In a single-region memory array which size is $N \times N$ with S spare rows and S spare columns. Suppose there are K faulty cells ($K = N^2 \cdot p$, p is the fault rate here) in the memory array.

Assumption:

- Once all the defective cells have been replaced, even there are more spare row or column existing, no replacement occurs again
- There is no difference among all the spare rows or all the spare columns, i.e. if one defective cell can be replaced by one spare row (column), it can be replaced equally by all the other spare rows (columns).
- The possibility of overlapping defects is allowed.

1. Condition: $\min(K \leq N, K \leq S)$

- All K faulty cells are at the same row

$$\max(r_1, \min(c_1, \dots, c_K))$$

- All K faulty cells are at the same column

$$\max(c_1, \min(r_1, \dots, r_K))$$

- None of K faulty cells are at the same line ($0 < k < K$, k is a random integer)

$$\max(\min(r_1, \dots, r_k, c_1, \dots, c_{K-k}), \min(r_1, \dots, r_K), \min(c_1, \dots, c_K), \min(c_1, \dots, c_k, r_1, \dots, r_{K-k}))$$

- Among all K faulty cells, they can be divided into x groups so that each group of cells are at the same row (there are k_1 groups with only one node inside). Also they can be divided into y groups so that each group of cells are at the same column (there are k_2 groups with only one node inside)

$$\max(\min(r_1, \dots, r_x), \min(c_1, \dots, c_y), \min(r_1, \dots, r_{x-k_1}, c_1, \dots, c_{k_1}), \min(c_1, \dots, c_{y-k_2}, r_1, \dots, r_{k_2}))$$

2. Condition: $\min(K \leq N, K > S)$

- All K faulty cells are at the same row
 r_1
- All K faulty cells are at the same column
 c_1
- None of K faulty cells are at the same line
 - i. The amount of faulty cells are more than the addition of spare rows and columns, i.e. $K > 2S$
 $\phi(x) = 0$ (the system cannot work by replacement)
 - ii. $S < K \leq 2S$
 $\max(\min(r_1, \dots, r_S, c_1, \dots, c_{K-S}), \min(c_1, \dots, c_S, r_1, \dots, r_{K-S}))$
- Among all K faulty cells, they can be divided into x groups so that each group of cells are at the same row (there are k_1 groups with only one node inside). Also they can be divided into y groups so that each group of cells are at the same column (there are k_2 groups with only one node inside)
 - i. $x \leq S$ and $y \leq S$,
 $\max(\min(r_1, \dots, r_x), \min(c_1, \dots, c_y), \min(r_1, \dots, r_{x-k_1}, c_1, \dots, c_{k_1}), \min(c_1, \dots, c_{y-k_2}, r_1, \dots, r_{k_2}))$
 - ii. $x \leq S$ and $y > S$
 $\max(\min(r_1, \dots, r_x), \min(r_1, \dots, r_{x-k_1}, c_1, \dots, c_{k_1}))$
 - iii. $x > S$ and $y > S$ and $(x-k_1) \leq S$ and $(y-k_2) \leq S$ and $k_1 \leq S$ and $k_2 \leq S$
 $\max(\min(r_1, \dots, r_{x-k_1}, c_1, \dots, c_{k_1}), \min(c_1, \dots, c_{y-k_2}, r_1, \dots, r_{k_2}))$
 - iv. $x > S$ and $y > S$ and $(x-k_1) \leq S$ and $(y-k_2) > S$ and $k_1 \leq S$ and $k_2 \leq S$
 $\min(r_1, \dots, r_{x-k_1}, c_1, \dots, c_{k_1})$

v. $x > S$ and $y > S$ and $(x - k_1) \leq S$ and $(y - k_2) \leq S$ and $k_1 \leq S$ and $k_2 > S$

$$\min(r_1, \dots, r_{x-k_1}, c_1, \dots, c_{k_1})$$

vi. $x > S$ and $y > S$ and $(x - k_1) > S$ and $(y - k_2) \leq S$ and $k_1 \leq S$ and $k_2 \leq S$

$$\min(c_1, \dots, c_{y-k_2}, r_1, \dots, r_{k_2})$$

vii. $x > S$ and $y > S$ and $(x - k_1) \leq S$ and $(y - k_2) \leq S$ and $k_1 > S$ and $k_2 \leq S$

$$\min(c_1, \dots, c_{y-k_2}, r_1, \dots, r_{k_2})$$

viii. $x > S$ and $y > S$ and NOT(($x - k_1) \leq S$ and $k_1 \leq S$)) and NOT(($y - k_2) \leq S$ and $k_2 \leq S$))

$$\phi(x) = 0$$

3. Condition: $\min(K > N, K \leq S)$

(All K faulty cells cannot be at the same row or the same column)

- None of K faulty cells are at the same line

$$\max(\min(r_1, \dots, r_k, c_1, \dots, c_{K-k}), \min(r_1, \dots, r_k), \min(c_1, \dots, c_k), \min(c_1, \dots, c_k, r_1, \dots, r_{K-k}))$$

- Among all K faulty cells, they can be divided into x groups so that each group of cells are at the same row (there are k_1 groups with only one node inside). Also they can be divided into y groups so that each group of cells are at the same column (there are k_2 groups with only one node inside)

$$\max(\min(r_1, \dots, r_x), \min(c_1, \dots, c_y), \min(r_1, \dots, r_{x-k_1}, c_1, \dots, c_{k_1}), \min(c_1, \dots, c_{y-k_2}, r_1, \dots, r_{k_2}))$$

4. Condition: $\min(K > N, K > S)$

(All K faulty cells cannot be at the same row or the same column)

- None of K faulty cells are at the same line

- The amount of faulty cells are more than the addition of spare rows and columns, i.e. $K > 2S$

$\phi(x) = 0$ (the system cannot work by replacement)

ii. $S < K \leq 2S$

$$\max(\min(r_1, \dots, r_S, c_1, \dots, c_{K-S}), \min(c_1, \dots, c_S, r_1, \dots, r_{K-S}))$$

- Among all K faulty cells, they can be divided into x groups so that each group of cells are at the same row (there are k_1 groups with only one node inside). Also they can be divided into y groups so that each group of cells are at the same column (there are k_2 groups with only one node inside)

i. $x \leq S$ and $y \leq S$,

$$\max(\min(r_1, \dots, r_x), \min(c_1, \dots, c_y), \min(r_1, \dots, r_{x-k_1}, c_1, \dots, c_{k_1}), \min(c_1, \dots, c_{y-k_2}, r_1, \dots, r_{k_2}))$$

ii. $x \leq S$ and $y > S$

$$\max(\min(r_1, \dots, r_x), \min(r_1, \dots, r_{x-k_1}, c_1, \dots, c_{k_1}))$$

iii. $x > S$ and $y > S$ and $(x-k_1) \leq S$ and $(y-k_2) \leq S$ and $k_1 \leq S$ and $k_2 \leq S$

$$\max(\min(r_1, \dots, r_{x-k_1}, c_1, \dots, c_{k_1}), \min(c_1, \dots, c_{y-k_2}, r_1, \dots, r_{k_2}))$$

iv. $x > S$ and $y > S$ and $(x-k_1) \leq S$ and $(y-k_2) > S$ and $k_1 \leq S$ and $k_2 \leq S$

$$\min(r_1, \dots, r_{x-k_1}, c_1, \dots, c_{k_1})$$

v. $x > S$ and $y > S$ and $(x-k_1) \leq S$ and $(y-k_2) \leq S$ and $k_1 < S$ and $k_2 > S$

$$\min(r_1, \dots, r_{x-k_1}, c_1, \dots, c_{k_1})$$

vi. $x > S$ and $y > S$ and $(x-k_1) > S$ and $(y-k_2) \leq S$ and $k_1 \leq S$ and $k_2 \leq S$

$$\min(c_1, \dots, c_{y-k_2}, r_1, \dots, r_{k_2})$$

vii. $x > S$ and $y > S$ and $(x-k_1) \leq S$ and $(y-k_2) \leq S$ and $k_1 > S$ and $k_2 \leq S$

$$\min(c_1, \dots, c_{y-k_2}, r_1, \dots, r_{k_2})$$

viii. $x > S$ and $y > S$ and NOT(($x-k_1) \leq S$ and $k_1 \leq S$) and NOT(($y-k_2) \leq S$ and $k_2 \leq S$))

$$\phi(x)=0$$

APPENDIX B

CODE FOR YIELD ANALYSIS UNDER SINGLE-REGION MEMORY ARRAY

```
import java.io.*;
import java.util.*;
import java.util.StringTokenizer;

public class SR
{
    public static void main (String[] args)
    {
        //initialize N, K~N*N, S~N, p=0.01
        int N=32;
        int S;
        int K;
        if((N/10)>=1)
            S=(int)(N*0.1)+1;
        else
            S=1;

        if((N*N/100)>=1)
            K=(int)(N*N*0.01)+1;
        else
            K=1;
        double p=0.01;

        //initialize P1, P2, P3
        double P1=Math.pow((1-p), N*N);
        double P2=0;
        double P3=0;
        double P21;
        double P22=0;

        //choose K cells with repetition allowed;
        //these K cells will fit<=S rows , <=S columns
        int round=1;
        double[] ar=new double[S+1];
        double sum=Combine(N, round)*Combine(N,K);
```

```

arr[l]=Combine(N,K);

while(round<S)
{
    round++;
    double m=Combine(N,round);
    double n=Combine(N*round,K);
    double sum l;
    double u=0;
    for(int i=round-1;i>0;i--)
        u=Combine(round,i)*arr[i]+u;
    sum l=m*(n-u);
    arr[round]=sum l/m;
    System.out.println("sum l is "+sum l+" round "+round+" u "+u +"n
"+n);
    sum=sum+sum l;
}
System.out.println("sum is "+sum+" total is "+Combine(N*N,K));
double s=sum/Combine(N*N,K);
P2=s*s;
double R=0;

if(K<=S)
    R--1;
else if(K>S )
    R P2;
System.out.println("R is "+R);
} //end main

public static double Combine(int m, int n)
{
    if(m<n)
        return 0;
    int x=m;
    int y=n;
    double z=(double)x/y;
    for(int i 0;i<n-1;i++)
    {
        x=x-1;
        y=y-1;
        z=z*(double)(x)/(double)(y);
    }
    return z;
} //end function of combination
} //end

```

APPENDIX C

CODE FOR YIELD ANALYSIS IN MULTI-REGION MEMORY ARRAY UNDER RANDOM FAULT DISTRIBUTION

```
import java.io.*;
import java.util.*;
import java.util.StringTokenizer;

//when n=2 multiregion repair
public class MR1
{
    public static void main (String[] args)
    {
        //in a matrix of n*m size, n is fixed at 2
        //S is 3
        int n=2;
        int m=26;
        double p=0.0022;
        int K=(int)((double)n*m*n*m*p)+1;

        //get all the possible pattern
        String[] s=rePattern(patternList(K,4),4);
        int count =0;
        double P=0;
        for(int i=0;i<s.length;i++)
        {
            double prob=1;
            if(s[i].compareTo("")==0)
                break;
            else
            {
                count++;
                String str=s[i];
                System.out.println(s[i]);
                for(int j=0;j<s[i].length();j++)
                {
                    if(str.substring(j,j+1).compareTo("0")!=0)
                    {
```

```

        char ch1=str.charAt(j);
        int k=(int)ch1-48;
        prob=prob*calSingle(m,k,2,0.75,0.75);
    }
    }
    P=P+prob;
}
}
System.out.println("R is "+P/count+ " "+count);
}

```

```

public static String[] patternList(int K, int S)
{
    double size=Math.pow(S,K);

    String[] str=new String[(int)size];
    for(int i=0;i<(int)size;i++)
        str[i]="";

    if(K == 1)
    {
        for(int i=0;i<S;i++)
            str[i]=Integer.toString(i);
    }//end if
    else
    {
        int count=0;
        String[] st=patternList(K-1,S);
        for(int i=0;i<st.length;i++)
        {
            for(int j=0;j<S;j++)
            {
                str[count]=st[i].concat(Integer.toString(j));
                count++;
            }//end for
        }//end for
    }//end else

    return str;
}//end function

public static String[] rePattern(String[] str, int S)
{
    int length=str.length;
    String[] s=new String[length];

```



```

        if(k<=f1*s || k<=f2*s)
            return 1;

    int round=1;
    double[] arr=new double[s+1];
    double sum=Combine(n, round)*Combine(n,k);
    arr[1]=Combine(n,k);

    while(round<s)
    {
        round++;
        double m=Combine(n,round);
        double t=Combine(n*round,k);
        double sum1;
        double u=0;
        for(int i=round-1;i>0;i--)
            u=Combine(round,i)*arr[i]+u;

        sum1=m*(t-u);
        arr[round]=sum1/m;

        sum=sum+sum1;
    }

    double s1=sum/Combine(n*n,k);
    double P=s1*s1*f1*f2;
    return P;
} //end function

public static double Combine(int m, int n)
{
    if(m<n)
        return 0;
    int x=m;
    int y=n;
    double z=(double)x/y;

    for(int i=0;i<n-1;i++)
    {
        x=x-1;
        y=y-1;
        z=z*(double)(x)/(double)(y);
    }
    return z;
} //end function of combination
}

```

APPENDIX D

CODE FOR YIELD ANALYSIS IN MULTI-REGION WITH HIERARCHICAL REPAIR UNDER RANDOM FAULT DISTRIBUTION

```
import java.io.*;
import java.util.*;
import java.util.StringTokenizer;

//when n =4, multiregion repair, hierarchy
public class MR2h
{

    public static void main (String[] args)
    {
        //the size of n*m, n=4, S=5
        int n=4;
        int m=12;
        double p=0.0022;
        int K=(int)((double)n*m*n*m*p)+1;

        //for multiregion, first divide into 4 groups
        //get all the possible patterns

        String[] s=rePattern(patternList(K,4),4);
        double[] P1=new double[100000];
        double[] P2=new double[100000];
        double[] P3=new double[100000];
        double[] P4=new double[100000];
        int num1=0;
        int num2=0;
        int num3=0;
        int num4=0;
        double P=0;
        int count=0;

        //analyze each pattern
        for(int i =0;i<s.length;i++)
        {
```

```

if(s[i].compareTo("")==0)
    break;

else if(s[i].compareTo("")!=0)
{
    String st=s[i];
    System.out.print(st+"\t");
    for(int j=0;j<4;j++)
    {
        if(st.substring(j,j+1).compareTo("0")!=0)
        {
            int k=Integer.parseInt(st.substring(j,j+1));
            String[] string=rePattern(patternList(k,4),4);

            System.out.println("k is "+k);
            //first group
            if(j==0)
            {
                for(int l=0;l<string.length;l++)
                {
                    if(string[l].compareTo("")!=0)
                    {
                        double prob=1;
                        for(int m1=0;m1<4;m1++)
                        {
                            if(string[l].substring(m1,m1+1).compareTo("0")!=0)
                            {
                                char ch1=string[l].charAt(m1);
                                int k1=(int)ch1-48;
                                double prob1=calSingle(n,k1,3,0.75,0.75);
                                prob=prob*prob1;
                            }
                        }
                        P1[num1] prob;
                        num1++;
                    }
                }
            }
            //end if
        }
    }
    //end for
}
//end if in the first group
//second group
else if(j==1)
{

```



```

        }//end if
    }//end for
}//end if in the second group
else
{
    for(int l=0;l<string.length;l++)
    {
        if(string[l].compareTo("")!=0)
        {
            double prob=l;
            for(int m l=0;m l<4;m l++)
            {
                if(string[l].substring(m l,m l+1).compareTo("0")!=0)
                {
                    char ch1=string[l].charAt(m l);
                    int k1=(int)ch1-48;
                    double prob1=calSingle(n,k1,3,0.5,0.5);
                    prob=prob*prob1;
                }
            }
        }
        P3[num3]=prob;
        num3++;
    } //end for
} //end else in the third group

//calculate the whole probability
for(int u=0;u<=num1;u++)
{
    for(int v=0;v<=num2;v++)
    {
        for(int w=0;w<=num3;w++)
        {
            for(int x=0;x<=num4;x++)
            {
                P=P+P1[u]*P2[v]*P3[w]*P4[x];
                count++;
            }
        }
    }
} //end for
} //end
} //end
num1=0;
num2=0;
num3=0;
num4=0;

```

```

        }//end if
    }//end for
} //end if
} //end for
System.out.println("R is "+P/count+ " "+count);
} //end main

public static String[] patternList(int K, int S)
{
    double size=Math.pow(S,K);
    String[] str=new String[(int)size];
    for(int i=0;i<(int)size;i++)
        str[i]="";

    if(K > 1)
    {
        for(int i=0;i<S;i++)
            str[i]=Integer.toString(i);
    } //end if
    else
    {
        int count=0;
        String[] st=patternList(K-1,S);
        for(int i=0;i<st.length;i++)
        {
            for(int j=0;j<S;j++)
            {
                str[count]=st[i].concat(Integer.toString(j));
                count++;
            } //end for
        } //end for
    } //end else

    return str;
} //end function

public static String[] rePattern(String[] str, int S)
{
    int length=str.length;
    String[] s=new String[length];

    for(int i=0;i<length;i++)
        s[i]="";

    int num = 0;

```

```

for(int i=0;i<length;i++)
{
    if(str[i].compareTo("")!=0)
    {
        int[] count=new int[S];
        for(int k=0;k<S;k++)
            count[k]=0;

        for(int j=0;j<str[i].length();j++)
        {
            char ch1=str[i].charAt(j);
            int y=(int)ch1-48;
            count[y]++;

        }//end for
        String string="";
        for(int l=0;l<S;l++)
            string=string.concat(Integer.toString(count[l]));

        int mark=0;
        for(int m=0;m<num;m++)
        {
            if(s[m].compareTo(string)!=0)
            {
                mark=1;
                break;
            }
        }//end for
        if(mark==0)
        {
            s[num]=string;
            num++;
        }
    }//end if
} //end for
return s;
} //end function

```

```

public static double calSingle(int n, int k,int s, double f1, double f2)
{
    if(k<=f1*s || k<=f2*s)
        return 1;

    int round=1;
    double[] arr=new double[s+1];
    double sum=Combine(n, round)*Combine(n,k);

```



```

arr[1]=Combine(n,k);

while(round<s)
{
    round++;
    double m=Combine(n,round);
    double t=Combine(n*round,k);
    double sum1;
    double u=0;
    for(int i=round-1;i>0;i--)
        u=Combine(round,i)*arr[i]+u;

    sum1=m*(t-u);
    arr[round]=sum1/m;
    sum=sum+sum1;
}
double s1=sum/Combine(n*n,k);
double P=s1*s1*f1*f2;
return P;
} //end function

public static double Combine(int m, int n)
{
    if(m<n)
        return 0;

    int x=m;
    int y=n;
    double z=(double)x/y;

    for(int i=0;i<n-1;i++)
    {
        x=x-1;
        y=y-1;
        z=z*(double)(x)/(double)(y);
    }
    return z;
} //end function of combination
} //end class

```

VITA 2)

Nin Jing

Candidate for the Degree of

Master of Science

Thesis: YIELD ANALYSIS ON EMBEDDED MEMORY WITH REDUNDANCY

Major Field: Computer Science

Biographical:

Education: Graduated from No.1 Middle School, Nanjing, Jiangsu, P.R. China in 1986; received Bachelor of Medicine degree in Department of Clinical Medicine from Nanjing University, Nanjing, Jiangsu, P.R. China in July 1995; received Master of Science degree in Department of Physiology and Biophysics from Indiana University Purdue University Indianapolis, Indianapolis, IN, in September 2000; Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in December 2002.

Experience: Clinical Residence in Xiakuan Hosptial, Nanjing, Jiangsu, P.R. China, from August 1997 to May 1998.