Mathematical Yield Estimation for Two-Dimensional-Redundancy Memory Arrays

Mango C.-T. Chao, Ching-Yu Chin, and Chen-Wei Lin
Dept. of Electronics Engineering, National Chiao-Tung University, Hsinchu, Taiwan
{mango@faculty.nctu.edu.tw, alwaysrain.gr@gmail.com, eeer.ee97@nctu.edu.tw }

Abstract

Defect repair has become a necessary process to enhance the overall yield for memories since manufacturing a natural good memory is difficult in current memory technologies. This paper presents an yield-estimation scheme, which utilizes an induction-based approach to calculate the probability that all defects in a memory can be successfully repaired by a two-dimensional redundancy design. Unlike previous works, which rely on a time-consuming simulation to estimate the expected yield, our yield-estimation scheme only requires scalable mathematical computation and can achieve a high accuracy with limited time and space complexity. Also, the proposed estimation scheme can consider the impact of single defects, column defects, and row defects simultaneously. With the help of the proposed yield-estimation scheme, we can effectively identify the most profitable redundancy configuration for large memory designs within few seconds while it may take several hours or even days by using conventional simulation approach.

I. Introduction

Due to the continually increasing process uncertainty and memory-array size, manufacturing natural good memory arrays requires an extremely high-quality process control and is difficult to achieve for today's large memories [1] [2]. Defect repair, thus, becomes a necessary capability for memories to maintain its yield at a profitable level. This memory repair is implemented through programming pre-defined redundant elements (or so-called spare elements) to replace the defective elements in a memory array. The importance of memory repair on yield improvement not only holds for stand-alone memory chips, whose size may exceed multiple Giga-bits today, but also holds for SoC chips, on which more than half of the area is occupied by embedded memories today and this memory-area ratio may exceed 94% in 2014 according to the projection of ITRS [3]. Therefore, a significant amount of research effort has been put into the area of designing a proper redundancy design along with an effective redundancy-allocation algorithm for memory repair in the past [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14].

Figure 1 illustrates four types of redundancy structures used for memory repair, including (1) one dimensional repair [4] [5], (2) two dimensional repair [6] [7] [8] [9] [10], (3) block repair [11] [12], and (4) mixed segment repair [13] [14]. One dimensional repair uses only spare rows (or spare columns) to repair defective cells and hence cannot handle column defects (or row defects). Two dimensional repair can handle both row defects and column defects by using spare rows and spare columns simultaneously. However, if a large single defect contaminates multiple adjacent bit-cells, the two-dimensional repair needs to spend several spare rows and/or spare columns to repair that one particular defect, which may degrade its successful-repair probability or lead to a large amount of required redundancy. On the other hand, block repair can repair large single defects effectively by replacing a small sub-array (*block*) at a time, but cannot handle row defects and column defects effectively. Mixed segment repair combines the advantage of two dimensional repair and block repair by dividing spare rows and columns into multiple segments, which can be individually used to repair a single defect or combined as a whole spare row (or column) to repair a row defect (or column defect). However, this mixed segment repair requires a much more complicated address mapping hardware and redundancy-allocation algorithm, and hence may not be practical. For current large memory designs, the most commonly used redundancy structure is two dimensional repair, but its spare row (or spare column) is usually designed to cover multiple word-lines (or bit-lines) of bit-cells, such that a large single defect can also be repaired by using only one spare row or spare column.

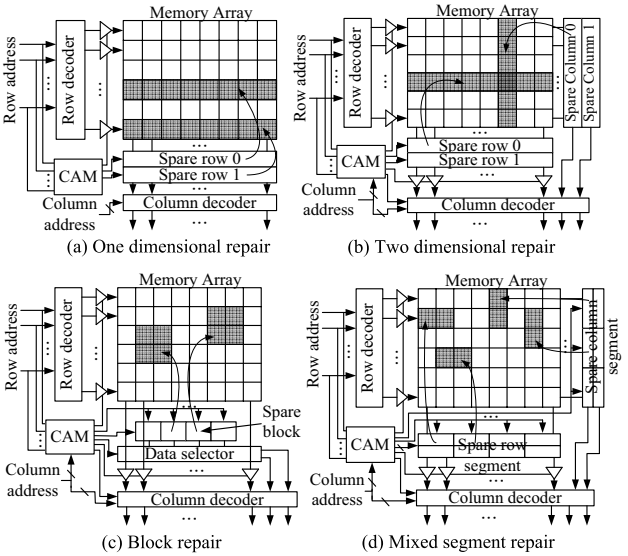

Fig. 1. **Redundancy structures for memory repair.**

For one dimensional repair or block repair, its optimal redundancy allocation can always be efficiently computed. However, for two dimensional repair, finding its optimal redundancy allocation is NP-complete and relies on effective heuristic algorithms. Fortunately, previous research work [15] has shown that the conventional *repair-most* algorithm can achieve almost the same successful-repair probability as the optimum one when the complete defect bitmap can be obtained after testing. [9] [15] [16] also proposed some effective redundancy-allocation algorithms, such as *local repair-most* and *essential spare pivoting*, to approximate its successful-repair probability to the optimum when only partial information from the defect bitmap can be used.

Once the redundancy structure and the redundancy-allocation algorithm are chosen, designers need to determine a proper amount of redundancy used for memory repair. Too less redundancy may result in a poor yield while too much redundancy may result in a waste of silicon area, which leads to unnecessary cost. In order to achieve the maximum profit, an effective yield estimation considering memory repair is required. For memories using one dimensional repair or block repair, a fast mathematical model has been developed for estimating the expected yield (i.e, the average

978-1-4244-8194-1/10/$26.00 ©2010 IEEE

successful repair probability) based on single-defect distributions in [17] [18]. However, for memories using two dimensional repair, the past research works [16] [19] [20] relied on simulation to obtained the expected yield, which randomly samples the locations of defects in multiple memory instances, performs a given redundancy-allocation algorithm to repair them, and then collects the percentage of successful-repair instances. Such a simulation may take long especially when multiple configurations of spare rows and spare columns are considered for large memory designs or when the successful-repair probability of multiple embedded-memory cores needs to be computed simultaneously for SoC designs. To the best of our knowledge, no effective mathematical model has been published in the past to estimate the memory yield based on the two dimensional repair.

In this paper, we propose an effective and efficient mathematical scheme to estimate the memory yield based on two dimensional repair. The proposed yield-estimation scheme considers the impact of single defects, row defects, and column defects at the same time. Also, the estimation scheme utilizes an induction-based approach to compute the successful-repair probability of a redundancy configuration with limited time and space complexity, instead of going through a time-consuming simulation used by previous yield-estimation schemes. Experimental results based on various memory sizes, redundancy configurations, and defect distributions demonstrate that the yield estimated by our proposed mathematical scheme closely matches the yield obtained by the conventional simulation approach. We will also show an example how the proposed yield-estimation scheme can help to identify the most profitable redundancy configuration within few seconds for a 512Mb memory design.

II. Mathematical Estimation to Successful-Repair Probability

A. Derivation Overview for Successful-Repair Probability

The yield of a memory array is determined by the probability that all the defective cells on this memory array can be successfully repaired by its redundancy design. The probability of the occurrence of defects, the size of the memory array, the amount of redundancy design, and the repair algorithm in use will all affect the value of this successful-repair probability. In our later analysis, we focus on the successful-repair probability based on the most popular redundancy structure, i.e., two dimensional repair (using both spare rows and spare columns as shown in Figure 1(b)). Also, we will use the probability distribution of defects as an input to our analysis, which can be monitored and collected from the past manufacturing data of the target process technology.

Following are the input parameters for the analysis of the successful-repair probability:

- R: the total number of rows.
- C: the total number of columns.
- M: the total number of spare rows.
- N: the total number of spare columns.
- $Dp(x, rd, cd)$: the probability that x single defects, rd row defects, and cd column defects occur.

Note that the x single defects are the defective words not covered by the rd defective rows and cd defective columns.

The yield of a memory array after repair (denoted as YR) is actually the expectation value of the successful-repair probability under the defect statistics of the process technology (i,e, $Dp(x, rd, cd)$), and can be expressed by Equation 1.

$$YR = \sum_{x=0} \sum_{rd=0}^{M} \sum_{cd=0}^{N} Dp(x, rd, cd) * DSR(x, rd, cd) \quad (1)$$

where $DSR(x, rd, cd)$ represents the probability that the x single defects, rd row defects, and cd column defects can be repaired by the M spare rows and N spare columns. In Equation 1, the index rd and cd runs from 0 to M and N for the summation, respectively, since we can never repair the case in which the number of row defects (or column defect) is larger than the number of spare rows (or spare column). The summation upper bound for the index x is not a fixed number. In our estimation, we stop increasing the index x when its resulting $Dp(x, rd, cd)$*$DSR(x, rd, cd)$ is negligible.

In this paper, we assume that the random variables of the number of row defects, column defects, and single defects are all independent and each random variable of the number of defects is modeled by a Poisson distribution, which is the most commonly used distribution for modeling the number of defects in a memory array [12] [20] [21] [22]. We use λ_{SD}, λ_{RD}, and λ_{CD} to denote the mean of the Poisson distributions of single defects, row defects, and column defects, respectively. Equation 2 shows the probability that exact x single defects occur (denoted as $f(x, \lambda_{SD})$) by using a Poisson distribution.

$$f(x, \lambda_{SD}) = \frac{(\lambda_{SD})^x e^{-\lambda_{SD}}}{x!} \quad (2)$$

As a result, the joint probability $Dp(x, rd, cd)$ can be computed by directly multiplying the individual probability density functions of having x single defects, rd row defect, and cd column defects, as shown in Equation 3. Note that the computation bottleneck of Equation 1 is on computing $DSR(x, rd, cd)$, not $Dp(x, rd, cd)$. Thus, our analysis can be easily applied to the cases which represent the defect distribution in other probabilistic distributions or a simple tabular format.

$$Dp(x, rd, cd) = f(x, \lambda_{SD}) \times f(rd, \lambda_{RD}) \times f(cd, \lambda_{CD}) \quad (3)$$

In the following subsections, we will explain the detailed computation of $DSR(x, rd, cd)$ in our estimation scheme.

B. Computation of $DSR(x, 0, 0)$

In this subsection, we derive the computation of the successful-repair probability considering only single defects (i.e, $DSR(x, 0, 0)$), meaning that no row defect or column defect exists on a memory array. Then we will show in Section II-F how the impact of row defects and column defects can be incorporated into the computation of $DSR(x, rd, cd)$. For simplification, we may use $DSR(x)$ to represent $DSR(x, 0, 0)$ in our later discussion.

In our estimation scheme, we use a four-dimension probability array, $S^{(i)}[m][n][z]$, to compute the value of $DSR(x)$. The array element $S^{(i)}[m][n][z]$ represents the probability that i single random defects can be repaired by using m spare rows, n spare columns, and z *spare units*. A spare unit here represents that a defect can be repaired by either a spare row or a spare column but which one of them is not determined yet. For example, the four defects shown in Figure 2 are repaired by one spare row and two spare units ($i = 4$, $m = 1$, $n = 0$, and $z = 2$), where each of d_3 and d_4 can be repaired by a spare row or spare column.

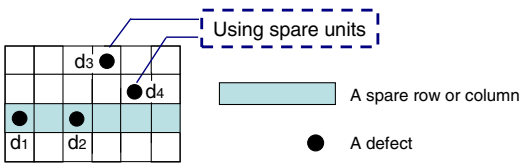

Fig. 2. **An example of using spare units**

In our estimation scheme, an induction-based approach is used to calculate the elements of S when the number of defects is $i+1$ ($S^{(i+1)}[m][n][z]$) based on the elements of S when the number of defects is i ($S^{(i)}[m][n][z]$). The computation of $S^{(i)}[m][n][z]$ starts from $i = 1$ and each iteration increases i by 1 until $i = x$.

The ranges of m, n, and z are $0 \leq m \leq M$, $0 \leq n \leq N$, and $0 \leq z \leq M+N$, respectively. The induction equation is shown as follows:

$$\begin{aligned} & S^{(i+1)}[m][n][z] \\ = \; & S^{(i)}[m][n][z] \times p_1(i,m,n,z) + \\ & S^{(i)}[m-1][n][z+1] \times p_2(i,m-1,n,z+1) + \\ & S^{(i)}[m][n-1][z+1] \times p_3(i,m,n-1,z+1) + \\ & S^{(i)}[m][n][z-1] \times p_4(i,m,n,z-1) \end{aligned} \quad (4)$$

The probability $S^{(i+1)}[m][n][z]$ is contributed from four probability events. The probabilities of these four events are denoted by p_1, p_2, p_3, and p_4, respectively. Each of p_1, p_2, p_3, and p_4 is a function of i, m, n, and z. The first event (associated with $p1$) occurs when the extra $(i+1)$th defect falls in a location covered by the m spare rows or the n spare columns repairing the original i defects. Hence no extra spare row, column, or unit needs to be used to repair this $(i+1)$ defect. Figure 3 shows an instance of this probability event.

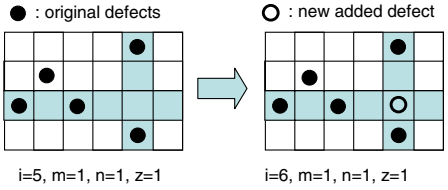

Fig. 3. **An instance of the probability event associated with p_1.**

The second event (associated with $p2$) occurs when the extra $(i+1)$th defect falls in a location where we need to fix the usage of a spare unit as a spare row to repair this defect. Hence, the number of used spare rows m is increased by 1 and the number of spare units z is decreased by 1. Figure 4 shows an instance of this probability event.

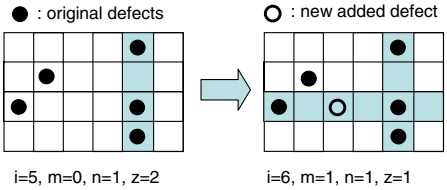

Fig. 4. **An instance of the probability event associated with p_2.**

Third, the extra $(i+1)$th defect may fall in a location where we need to fix the usage of a spare unit as a spare column to repair this defect. Hence, the number of used spare column n is increased by 1 and the number of spare units z is decreased by 1. Figure 5 shows an instance of this probability event.

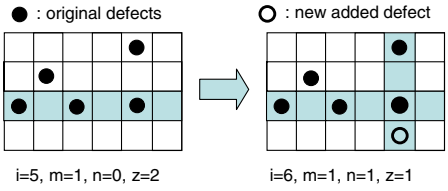

Fig. 5. **An instance of the probability event associated with p_3.**

Last, the extra $(i+1)$th defect may fall in a location where we can use another spare unit to repair this defect. Hence, the number of spare units z is increased by 1. Figure 6 shows an instance of this probability event.

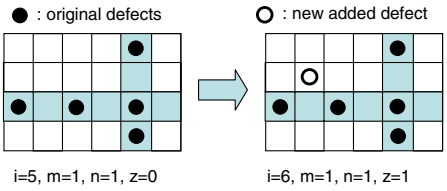

Fig. 6. **An instance of the probability event associated with p_4.**

Equation 5 shows the initial condition of $S^{(i)}[m][n][z]$ when $i=1$, where one single defect is always repaired by using a spare unit, which is the cheapest way of repairing a single defect. Thus, $S^{(1)}[0][0][1]$ is set to 1 and all other elements for $i=1$ are set to 0.

$$\begin{aligned} S^{(1)}[0][0][1] &= 1 \\ S^{(1)}[m][n][z] &= 0, \text{ otherwise} \end{aligned} \quad (5)$$

Then, $DSR(x)$ can be obtained by summing the probability of each $S^{(x)}[m][n][z]$, whose m and n are not larger than M and N, respectively, and whose z is not larger than $M+N-m-n$. The calculation of $DSR(x)$ is listed in Equation 6.

$$DSR(x) = \sum_{m=0}^{M} \sum_{n=0}^{N} \sum_{z=0}^{M+N-m-n} S^{(x)}[m][n][z] \quad (6)$$

C. Computation of p_1, p_2, p_3, and p_4

In this subsection, we will show the detailed computation of p_1, p_2, p_3, and p_4, which are all functions of m, n, z, and i. In the following illustrations of these four probabilities, we consider the case of adding one extra defect into the situation where the original i defects are already repaired by m spare rows, n spare columns, and z spare units. In other words, we are computing the probability $p_1(i,m,n,z)$, $p_2(i,m,n,z)$, $p_3(i,m,n,z)$, and $p_4(i,m,n,z)$.

An event of $p_1(i,m,n,z)$ occurs when the extra $(i+1)$th defect falls in the area covered by the original m spare rows and n spare columns, which is illustrated in Figure 7. Equation 7 lists the computation of $p_1(i,m,n,z)$, where the denominator $(R \times C - i)$ represents the total number of cell locations that the extra defect may fall. For the numerator of p_1, the term $(R \times n + C \times m - m \times n)$ represents the locations repaired by the m spare rows and n spare columns. Then we need to exclude the number of defects falling already inside the repaired area, i.e., the term $(i-z)$. Note that, after adding the extra defect for the event of $p_1(i,m,n,z)$, the $i+1$ defects can still be repaired by m spare rows, n spare columns, and z spare units.

$$p_1 = \frac{R \times n + C \times m - m \times n - (i-z)}{R \times C - i} \quad (7)$$

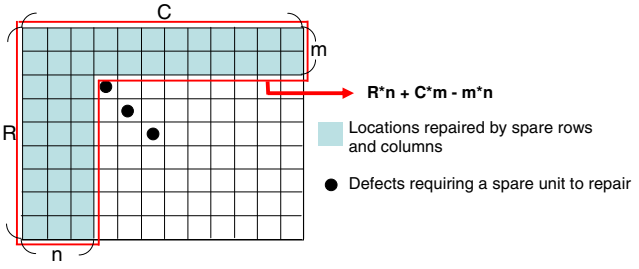

Fig. 7. **Illustration for the computation of $p_1(i,m,n,z)$.**

An event of $p_2(i,m,n,z)$ occurs when the extra $(i+1)$th defect falls in the area where we need to turn a spare unit into a spare row

to repair the extra defect. Figure 8 illustrates such area. Equation 8 lists the computation of $p_2(i, m, n, z)$. For the numerator of p_2, the term $(C-n-z)\times z$ represents the locations where the extra defect will for sure cause an event of p_2 (labeled as a diamond in Figure 8). The term $z\times(z-1)$ represents the locations where the extra defect will cause an event of p_2 with a 50% chance (labeled as a triangle in Figure 8) since we can turn a spare unit into either a spare row or spare column to repair the extra defect. Note that, after adding the extra defect for the event of $p_2(i, m, n, z)$, the $i+1$ defects are repaired by $m+1$ spare rows, n spare columns, and $z-1$ spare units.

$$p_2 = \frac{(C-n-z)\times z + 0.5\times z\times(z-1)}{R\times C - i} \quad (8)$$

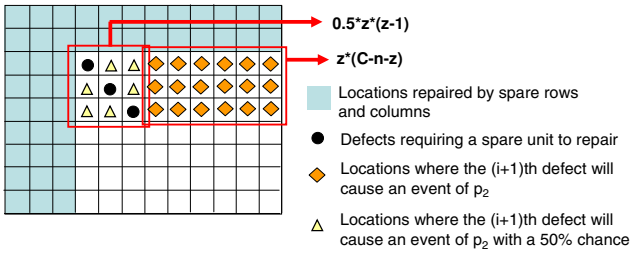

Fig. 8. **Illustration for the computation of $p_2(i, m, n, z)$.**

An event of $p_3(i, m, n, z)$ occurs when the extra $(i+1)$th defect falls in the area where we need to turn a spare unit into a spare column to repair the extra defect. Figure 9 illustrates such area. Equation 9 lists the computation of $p_3(i, m, n, z)$. In a similar manner as p_2, the term $(R-m-z)\times z$ and the term $z\times(z-1)$ in Equation 9 represent the locations where the extra defect will cause an event of p_3 for sure (labeled as a diamond in Figure 9) and with a 50% chance (labeled as a triangle in Figure 9), respectively. Note that, after adding the extra defect for the event of $p_3(i, m, n, z)$, the $i+1$ defects are repaired by m spare rows, $n+1$ spare columns, and $z-1$ spare units.

$$p_3 = \frac{(R-m-z)\times z + 0.5\times z\times(z-1)}{R\times C - i} \quad (9)$$

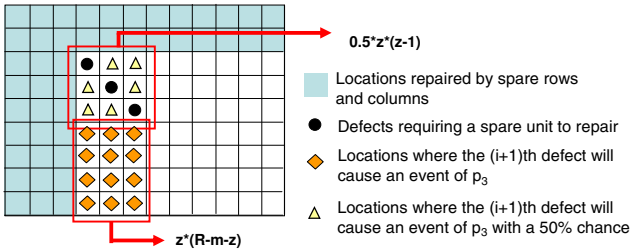

Fig. 9. **Illustration for the computation of $p_3(i, m, n, z)$.**

An event of $p_4(i, m, n, z)$ occurs when the extra $(i+1)$th defect falls in the cell locations where we need an extra spare unit to repair the defect. Those cells are labeled as a diamond in Figure 10. Equation 10 lists the computation of $p_4(i, m, n, z)$, in which the term $(R-m-z)\times(C-n-z)$ represents the number of those diamond cells in Figure 10. Note that, after adding the extra defect for the event of $p_4(i, m, n, z)$, the $i+1$ defects are repaired by m spare rows, n spare columns, and $z+1$ spare units.

$$p_4 = \frac{(R-m-z)\times(C-n-z)}{R\times C - i} \quad (10)$$

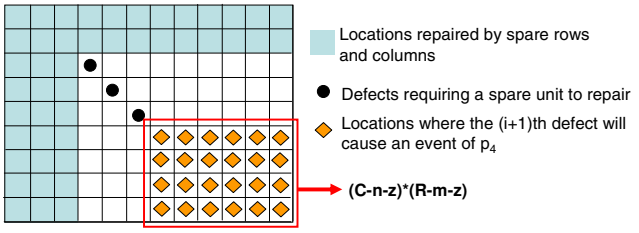

Fig. 10. **Illustration for the computation of $p_4(i, m, n, z)$.**

D. Optimality of Computed $DSR(x)$

The above estimation is actually a lower bound of the real $DSR(x)$ since we consider the successful-repaired combinations of using spare rows, spare columns, and spare units by adding one more defect to the defect map at a time. In reality, the repair analysis is applied after all x defects are added to the defect map. However, by using the concept of "spare units", our estimation scheme can cover a wide range of the successful-repaired combinations. Also, the applied repair algorithm in the repair analysis is only a heuristic and cannot consider all the combinations of using spare rows and spare columns as well. Thus, our estimation scheme can provide a close approximation to $DSR(x)$, which will be shown in our experiments in Section III-A.

E. Complexity of Computing $DSR(x)$

The complexity of computing $DSR(x)$ are determined by the size of the 4-dimensional array $S^{(x)}[m][n][z]$, which is $(M*N*(M+N)*x)$. To compute each array element of $S^{(x)}[m][n][z]$ (including the computation of p_1 to p_4 and Equation 6) requires 14 times of multiplication, 29 times of addition, and 4 times of division. Thus, the computation complexity and space complexity of computing $DSR(x)$ are both $O(M*N*(M+N)*x)$. Note that this complexity is proportional to the number of spare rows and spare columns instead of the size of the memory array RxC, and hence can be properly controlled. Also, the value of $S^{(x)}[m][n][z]$ drops quickly when x exceeds a certain point. We can stop the computation of $S^{(x)}[m][n][z]$ when its value becomes negligible, and hence the actual computation complexity can be further reduced.

F. Considering Row and Column Defects for $DSR(x, rd, cd)$

For single defects, one spare row or spare column may be able to repair multiple of them. However, for each row defect (or column defect), one individual spare row (or spare column) is required for repairing it. In other words, when rd row defects and cd column defects exist, the effective number of spare rows (or spare columns) left for repairing the random single defective words is $M-rd$ ($N-cd$), instead of the M (N) as shown in the previous subsections. Therefore, when computing $DSR(x, rd, cd)$, we only need to substitute the parameter M by $M-rd$ and N by $N-cd$ to consider the impact of row defects and column defects. The other computations remain the same.

III. Experiments and Discussion

A. Accuracy of $DSR(x)$

The most difficult and challenging task in our yield-estimation scheme is the analysis of $DSR(x)$. We first validate the accuracy of $DSR(x)$ by comparing its results with the results obtained from a random simulation using 1-million samples. In each sample, we randomly select x defects on an RxC memory array and then apply the repair-most algorithm to check whether these x defects can be repaired by M spare rows and N spare columns.

Table I reports the results of $DSR(x)$ for different numbers of single defects x when $R=C=100$ and $M=N=10$. Column 2 and Column 3 of Table I list the value of $DSR(x)$ obtained by our estimation scheme and the random simulation, respectively.

Column 4 lists the difference between the estimation scheme and the simulation. As the result shows, our estimation scheme can match the simulation result closely. The average difference between the estimated $DSR(x)$ and the simulated $DSR(x)$ is only 0.20%.

# of single defects (x)	$DSR(x)$ in %		
	estimation (a)	simulation (b)	$\|a-b\|$
21	98.88	98.88	0
22	94.15	93.79	0.36
23	83.79	83.15	0.64
24	68.34	67.91	0.43
25	50.67	50.49	0.18
26	34.15	34.18	0.03
27	21.00	20.82	0.18
28	11.86	11.78	0.08
29	6.20	6.16	0.04
30	3.01	3.08	0.07
Average	–	–	0.20

TABLE I
Accuracy of $DSR(x)$ given different numbers of single defects.

Similar experiments are conducted for Table II and Table III, which report the results of $DSR(x)$ for different numbers of spare columns (N) and different total numbers of columns (C), respectively. The other parameters are fixed as shown on top of the tables. The same trend can be observed as in Table I. The difference of the reported $DSR(x)$ between our estimation scheme and the random simulation is limited, only 0.19% for Table II and 0.21% for Table III in average. The above results demonstrate the accuracy of the estimated $DSR(x)$ over different combinations of parameters.

# of spare columns (N)	$DSR(x)$ in %		
	estimation (a)	simulation (b)	$\|a-b\|$
7	2.84	2.96	0.12
8	10.18	10.46	0.28
9	26.34	26.40	0.06
10	50.67	50.49	0.18
11	75.31	74.99	0.32
12	91.63	91.22	0.41
13	98.32	98.16	0.16
14	99.85	99.85	0
Average	–	–	0.19

TABLE II
Accuracy of $DSR(x)$ given different numbers of spare columns.

# of columns (C)	$DSR(x)$ in %		
	estimation (a)	simulation (b)	$\|a-b\|$
50	86.87	86.18	0.69
100	50.67	50.49	0.18
150	35.32	34.66	0.06
200	28.13	28.24	0.11
400	18.74	18.72	0.02
Average	–	–	0.21

TABLE III
Accuracy of $DSR(x)$ given different numbers of columns.

B. Accuracy of the Yield after Repair YR

In this subsection, we compare the yield after repair (denoted as YR in Equation 1) computed by our estimation scheme with that obtained from a 100K-sample random simulation based on the configuration of an industrial 512Mb non-volatile memory array. This memory array contains 8192 word-line (rows), each word-line contains 4096 words (column), and each word contains 16 bits. The redundancy design of this memory array includes 16 spare rows and 6 spare columns. Each spare row and spare column can repair 8 consecutive row addresses and 4 consecutive column addresses, respectively. To consider the impact of multiple addresses repaired by one spare row (or spare column), our estimation scheme needs to multiple the resolution of a row by 8 (or 4). As a result, the 8192x4096 array is effectively a 1024x1024 array during our estimation. In the random simulation, we sample the single defects, row defects, and column defects for a memory instance based on the Poisson distributions with λ_{SD}, λ_{RD}, and λ_{CD}, respectively. The repair-most algorithm will then be performed to check whether the defective instance can be successfully repaired. Note that the value of λ_{SD}, λ_{RD}, or λ_{CD} is randomly picked for illustration purpose and not related to any specific process technology.

Table IV reports the estimated and simulated yields based on different values of λ_{SD} for the single-defect distribution. The other parameters are fixed as listed on top of Table IV. As Table IV shows, the estimated yields match the simulated ones closely. The average difference is 0.20% and the maximum difference is 0.40% in Table IV. Also, one run of the 100K-sample random simulation takes slightly more than 1 hour in average, but our estimation scheme takes less than 0.2 second. This result further demonstrates the efficiency of our yield-estimation scheme in addition to its accuracy.

parameters: R=8192, C=4096, M=16, N=6, λ_{RD}=6, λ_{CD}=2			
# of single defect (λ_{SD})	YR in %		
	estimation (a)	simulation (b)	$\|a-b\|$
8	94.08	94.31	0.23
9	90.61	90.78	0.17
10	85.92	86.23	0.31
11	80.03	80.43	0.40
12	73.07	73.21	0.14
13	65.34	65.45	0.11
14	57.15	57.19	0.04
Average	–	–	0.20

TABLE IV
Accuracy of the estimated yield (YR) given different λ_{SD}.

In Table V and Table VI, we further validate the accuracy of the proposed yield-estimation scheme by varying the number of λ_{RD} for row defect distribution and the number of spare rows (M), respectively. The other parameters are fixed as shown on top of the tables. Similar to the result observed in Table IV, the estimated yields are all close to the simulated ones. The average difference between the estimated yield and the simulated one is only 0.10% and 0.16% in Table V and Table VI, respectively. Results shown in Table IV, Table V, and Table VI demonstrate that our yield-estimation scheme can accurately compute the yield after repair for a large industrial memory array.

parameters: R=8192, C=4096, M=16, N=6, λ_{SD}=12, λ_{CD}=2			
# of row defect (λ_{RD})	YR in %		
	estimation (a)	simulation (b)	$\|a-b\|$
2	94.38	94.35	0.03
3	90.92	90.87	0.05
4	86.19	86.17	0.02
5	80.19	80.27	0.08
6	73.07	73.21	0.14
7	65.13	65.38	0.20
8	56.73	56.93	0.20
Average	–	–	0.10

TABLE V
Accuracy of the estimated yield (YR) given different λ_{RD}.

C. Finding an Optimal Redundancy Configuration

In the following experiment, we attempt to identify the most profitable redundancy configuration (the number of spare rows and the number spare columns) by using our yield-estimation scheme for the same 512Mb memory array shown in Section III-B. Equation 11 lists the profit function used in this experiment, where the cost of a memory die is linear proportional to the memory-array size. In Equation 11, SP denotes the sale price of a good die. C_{ini} denotes the cost of manufacturing a given memory without any redundancy design. A_{rep} and A_{ini} denote the area with and without

parameters: R=8192, C=4096, N=6, λ_{SD}=12, λ_{RD}=6, λ_{CD}=2			
# of spare row (M)	YR in %		
	estimation (a)	simulation (b)	$\lvert a-b \rvert$
14	57.07	57.28	0.21
16	73.07	73.21	0.14
18	84.98	85.23	0.25
20	92.46	92.64	0.20
22	96.47	96.55	0.08
24	98.32	98.48	0.16
26	99.07	99.08	0.01
28	99.33	99.56	0.23
Average	–	–	0.16

TABLE VI
Accuracy of the estimated yield (YR) given different spare rows.

the redundancy design. Here, we assume that $SP = 3$ and $C_{ini} = 1$. Note that we chose the above profit function just for the illustration purpose. Our yield-estimation scheme can also help to determine the best redundancy configuration based on other profit functions.

$$\text{Profit} \quad = SP * YR - C_{ini} * \frac{A_{rep}}{A_{ini}} \tag{11}$$

Table VII lists the profit computed with the estimated yield YR for different numbers of spare rows (M) and spare columns (N). The other parameters are listed on top of Table VII. The largest computed profit is highlighted with a gray background, which is achieved by using 26 spare rows and 10 spare columns. Similarly, Table VIII lists the profit computed with the yield obtained by 100K-sample random simulation. As the result shows, its largest profit also falls on the same configuration (26 spare rows and 10 spare columns). The difference between the largest profits computed by our estimation scheme and the random simulation is 0.0009 (1.96370 minus 1.96280). Note that computing all 42 profits in Table VII with our estimation scheme only takes around 1 second, which is shorter than computing one profit for 42 individual times because the four-dimension array $S^{(i)}[m][n][z]$ used for a larger number of spare rows (or spare columns) can be reused for a smaller number of spare rows (or spare columns). However, computing the profits in Table VIII with 100K-sample random simulation takes around 45 hours. This result again demonstrates the advantage of having an effective and efficient yield-estimation scheme when determining the best redundancy configuration during design phase.

parameters: R=8192, C=4096, λ_{SD}=12, λ_{RD}=6, λ_{CD}=2							
	number of spare rows M						
N	16	18	20	22	24	26	28
4	0.64721	1.09615	1.42179	1.61993	1.72207	1.76630	1.78234
6	1.17052	1.52586	1.74830	1.86663	1.92017	1.94070	1.94654
8	1.53304	1.75817	1.87890	1.93394	1.95537	1.96150	1.96193
10	1.75846	1.87949	1.93451	1.95654	1.96237	1.96280	1.96142
12	1.87917	1.93450	1.95592	1.96235	1.96277	1.96139	1.95971
14	1.93449	1.95591	1.96233	1.96275	1.96137	1.95969	1.95771

TABLE VII
Profit computed by estimated yield for different numbers of spare rows and spare columns.

parameters: R=8192, C=4096, λ_{SD}=12, λ_{RD}=6, λ_{CD}=2							
	number of spare rows M						
N	16	18	20	22	24	26	28
4	0.66911	1.11025	1.43739	1.64482	1.76226	1.79810	1.81864
6	1.17472	1.52586	1.75370	1.86903	1.92497	1.94100	1.95344
8	1.54114	1.76447	1.88670	1.93604	1.95657	1.96270	1.96343
10	1.75756	1.88189	1.93451	1.95654	1.96297	1.96370	1.96232
12	1.87977	1.93450	1.95652	1.96325	1.96217	1.96229	1.96062
14	1.93509	1.95621	1.96263	1.96335	1.96227	1.96059	1.95861

TABLE VIII
Profit computed by simulated yield for different numbers of spare rows and spare columns.

IV. Conclusion

In this paper, we proposed a mathematical model to estimate the yield after repair based on the two-dimensional redundancy design. This yield-estimation scheme relied on an induction-based approach to compute the probability that the defects can be successfully repaired by the given spare row and spare columns. The time and space complexity of this induction-based approach is limited and scalable to large memory arrays. Also, this yield-estimation scheme consider the impact of single defects, row defects, and column defects simultaneously. A series of experiments based on a fair-size industrial memory design demonstrated that the proposed yield-estimation scheme can achieve almost the same accuracy but require much shorter runtime compared to the conventional simulation-based approach.

References

[1] Y Zorian, "Embedded memory test and repair: infrastructure IP for SOC yield," IEEE International Test Conference, pp. 340-349, 2002.

[2] Y. Hamamura, K. Nemoto, T. Kumazawa, H. Iwata, K. Okuyama, S. Kamohara, and A. Sugimoto, "Repair yield simulation with iterative critical area analysis for different types of failure," IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp. 305-313, 2002.

[3] ITRS, "International Technology Roadmap for Semiconductors," 2000. [Online]. Available: http://public.itrs.net/Links/2000Winter/Design.ppt

[4] A. Tanabe, T. Takeshima, H. Koike, Y. Aimoto, M. Takada, T. Ishijima, N. Kasai, H. Hada, K. Shibahara, T. Kunio, T. Tanigawa, T. Saeki, M. Sakao, H. Miyamoto, H. Nozue, S. Ohya, T. Murotani, K. Koyama, and T. Okuda, "A 30-ns 64-Mb DRAM with built-in self-test and self-repair function," IEEE Journal of Solid-State Circuits, Vol. 27 , Issue 11 , pp. 1525-1533, 1992

[5] Kim Ilyoung, Y. Zorian, G. Komoriya, H. Pham, F. P. Higgins, and J. L. Lewandowski, "Built in self repair for embedded high density SRAM," IEEE International Test Conference, pp. 1112-1119, 1998.

[6] M. F. Chang, W. K. Fuchs, and J. H. Patel, "Diagnosis and repair of memory with coupling faults," IEEE Transactions on Computers, Vol 38, Issue 4, pp. 493-500, 1989.

[7] D. K. Bhavsar, "An algorithm for row-column self-repair of RAMs and its implementation in the Alpha 21264," IEEE International Test Conference, pp. 311-318, 1999.

[8] H. Y. Lin, F. M. Yeh, I. Y. Chen, and S. Y. Kuo, "An efficient perfect algorithm for memory repair problems," IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp. 306-313, 2004.

[9] S. K. Lu, Y. C. Tsai, C. H. Hsu, K. H. Wang, and C. W. Wu, "Efficient built-in redundancy analysis for embedded memories with 2-D redundancy," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 14, Issue 1, pp. 34-42, 2006.

[10] R. F. Huang, J. F. Li, J. C. Yeh, and C. W. Wu, "A simulator for evaluating redundancy analysis algorithms of repairable embedded memories," IEEE International Workshop on Memory Technology, Design and Testing, pp. 68-73, 2002.

[11] S. K. Lu and C. H. Hsu, "Built-In self-repair for divided word line memory," IEEE International Symposium on Circuits and Systems, vol. 4, pp. 13-16, 2001.

[12] Nermine H. Ramadan, "Redundancy yield model for SRAMs," Intel Technology Journal Q4'97.

[13] N. Park and E. Lombardi, "Repair of memory arrays by cutting," International Workshop on Memory Technology, Design and Testing, pp. 124-130, 1998.

[14] J. F. Li, J. C. Yeh, R. F. Huang, and C. W. Wu, "A built-in self-repair design for RAMs with 2-D redundancy," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 13, Issue 6, pp. 742-745, 2005.

[15] C.-T. Huang; C.-F. Wu; J.-F. Li; C.-W. Wu, "Built-in redundancy analysis for memory yield improvement," IEEE Transactions on Reliability, vol. 52, issue 4, pp. 386-399, 2003.

[16] R.-F. Huang, C.-H. Chen, and C.-W. Wu, "Economic aspects of memory built-in self-repair," IEEE Design & Test of Computers, vol. 24, issue 2, pp. 164-172, 2007.

[17] J. Khare, D.B.I. Feltham, and W. Maly, "Accurate estimation of defect-related yield loss in reconfigurable VLSI circuits," IEEE Journal of Solid State Circuit, vol. 28, No. 2, Feb. 1993.

[18] N. H. Ramadan, "Redundancy Yield Model for SRAMs," Intel Technology Journal Q4'97.

[19] R.-F. Huang, J.-F.-Li, J.-C. Yeh, and C.-W. Wu, "Raisin: Redundancy Analysis Algorithm Simulation," IEEE Design & Test of Computers, vol. 24, issue 3, pp. 386-396, 2007.

[20] S. Shoukourian, V. Vardanian, and Y. Zorian, "SoC yield optimization via an embedded-memory test and repair infrastructure," IEEE Design & Test of Computers, vol. 21, issue 3, pp. 200-207, 2004.

[21] L. Youngs, and S. Paramanandam, "Mapping and repairing embedded-memory defects," IEEE Design & Test of Computers, Vol. 14, Issue 1, pp. 18-24, 1997.

[22] R. F. Huang, J. F. Li, J. C. Yeh, and C. W. Wu, "Raisin: Redundancy Analysis Algorithm Simulation," IEEE Design & Test of Computers, Vol. 24, Issue 4, pp. 386-369, 2007.