Chapter 2
Redundancy

2.1 Introduction

For designing redundancy circuit, the estimation of the advantages and disadvantages is indispensable. The introduction of redundancy in a memory chip results in yield improvement and fabrication-cost reduction. However, it also causes the following penalties. First, spare memory cells to replace faulty cells, programmable devices to memorize faulty addresses, and control circuitry to increase chip size. Second, the time required for the judgment whether the input address is faulty or not is added to the access time. Third, special process steps to fabricate the programmable devices and test time to store faulty addresses into the devices are required. Therefore, the design of redundancy circuit requires a trade-off between yield improvement and these penalties. The estimation of yield improvement requires a fault-distribution model. There are two representative models, Poisson distribution model and negative-binomial model, which are often used for the yield analysis of memory LSIs. The "replacement" of normal memory elements by spare elements requires checking whether the accessed address includes faulty elements, and if yes, inhibiting the faulty element from being activated and activating a spare element instead. These procedures should be realized with as small penalty as possible. One of the major issues for the replacement is memory-array division. Memory arrays are often divided into subarrays for the sake of access-time reduction, power reduction, and signal/noise ratio enhancement. There are two choices for memories with array division: (1) a faulty element in a subarray is replaced only by a spare element in the same subarray (intrasubarray replacement) and (2) a faulty element in a subarray may be replaced by a spare element in another subarray (intersubarray replacement). The former has smaller access penalty, while the latter realizes higher replacement efficiency. It is also possible that a subarray is replaced by a spare subarray. The devices for memorizing faulty addresses and test for finding out an effective replacement are also important issues for redundancy.

The fault distribution models are presented in Sect. 2.2. The yield improvement analysis using the models is described in Sect. 2.3. Section 2.4 describes the circuit techniques for realizing the replacement. The intrasubarray replacement, intersubarray replacement, and subarray replacement are described in Sects. 2.5, 2.6,

M. Horiguchi and K. Itoh, *Repair for Nanoscale Memories*, Integrated Circuits and Systems 1, 19
DOI 10.1007/978-1-4419-7958-2_2,

and 2.7, respectively. The programmable devices for storing faulty addresses are described in Sect. 2.8. Finally, testing techniques for redundancy are explained in Sect. 2.9.

2.2 Models of Fault Distribution

2.2.1 Poisson Distribution Model

Let us consider a memory chip with N "elements" (Fig. 2.1). Here, an element may be a memory cell, a row of memory cells, a column of memory cells, a subarray, and so on. If faults are randomly distributed in the chip, the probability of an element being faulty, p, is independent of the probability of other elements being faulty or nonfaulty. Therefore, the probability that k elements are faulty and $(N - K)$ elements are not faulty is expressed as the product of their probabilities, $p^K(1 - p)^{N-K}$. Since the number of cases of selecting K faulty elements out of N elements is expressed by

$$\binom{N}{K} = {}_N C_K = \frac{N!}{(N-K)!K!} = \frac{N(N-1)\cdots(N-K+1)}{K!}, \tag{2.1}$$

the probability of existing K faulty elements in the chip is expressed as

$$P(K) = \binom{N}{K} p^K (1-p)^{N-K}. \tag{2.2}$$

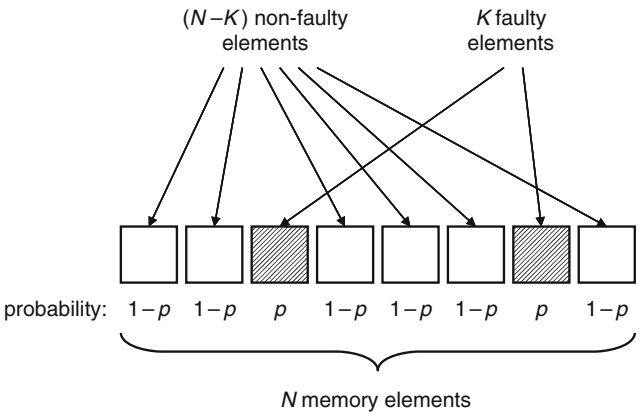


Fig. 2.1 Probability of existing K faulty elements out of N elements when faulty probability of an element is p

This is called binomial distribution and the coefficient $\binom{N}{K}$ is called binomial coefficient. Usually, N is very large and p is very small. When $N \to \infty$, keeping $\lambda = Np$ constant, (2.2) becomes

$$\begin{aligned} P(K) &= \frac{N(N-1)\cdots(N-K+1)}{K!} \cdot p^K \cdot (1-p)^{N-K} \\ &= 1 \cdot \left(1 - \frac{1}{N}\right) \cdots \left(1 - \frac{K+1}{N}\right) \cdot \frac{\lambda^K}{K!} \cdot \left(1 - \frac{\lambda}{N}\right)^N \left(1 - \frac{\lambda}{N}\right)^{-K} \\ &\to \frac{\lambda^K}{K!}\left(1 - \frac{\lambda}{N}\right)^N = \frac{\lambda^K \exp(-\lambda)}{K!} \quad (N \to \infty). \end{aligned} \tag{2.3}$$

This is called *Poisson distribution*. Figure 2.2 shows examples of the distribution. The probability $P(K)$ monotonously decreases with K for $\lambda < 1$, and has a peak around $K \sim \lambda$ for $\lambda > 1$. Poisson distribution is characterized by only one parameter λ. The average $\bar{K}$ and standard deviation $\sigma(K)$ of the number of faulty elements are expressed as

$$\bar{K} = \sum_{K=0}^{\infty} KP(K) = \exp(-\lambda) \sum_{K=1}^{\infty} \frac{\lambda^K}{(K-1)!} = \lambda, \tag{2.4}$$

$$\begin{aligned} \sigma(K) &= \sqrt{\sum_{K=0}^{\infty} K^2 P(K) - \bar{K}^2} = \sqrt{\exp(-\lambda) \sum_{K=1}^{\infty} \frac{K\lambda^K}{(K-1)!} - \lambda^2} \\ &= \sqrt{\exp(-\lambda)\left\{\sum_{K=2}^{\infty} \frac{\lambda^K}{(K-2)!} + \sum_{K=1}^{\infty} \frac{\lambda^K}{(K-1)!}\right\} - \lambda^2} \\ &= \sqrt{\exp(-\lambda)(\lambda^2 \exp\lambda + \lambda \exp\lambda) - \lambda^2} = \sqrt{\lambda}. \end{aligned} \tag{2.5}$$

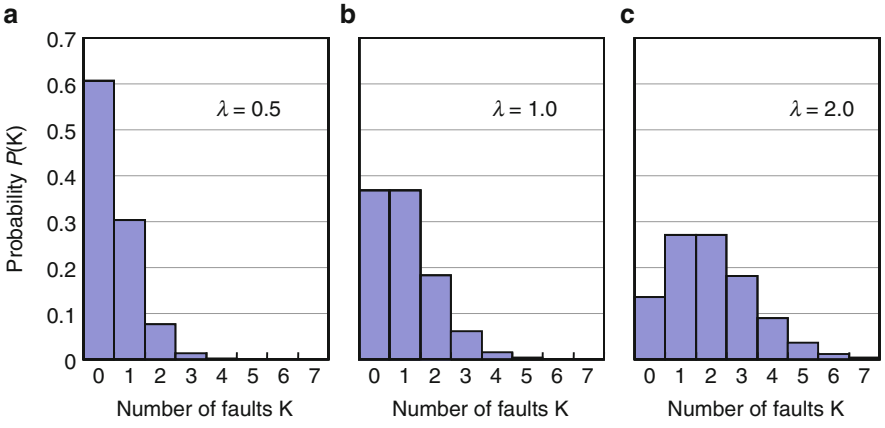


Fig. 2.2 Examples of Poisson distribution: (**a**) $\lambda = 0.5$, (**b**) $\lambda = 1.0$, and (**c**) $\lambda = 2.0$

Thus, the parameter λ is equal to the average number of faults and is expressed as:

$$\lambda = AD, \tag{2.6}$$

where A is the chip area and D is the fault density. The probability of a chip having no faulty elements (raw yield, i.e., yield without redundancy) is expressed as

$$P(0) = \exp(-\lambda) = \exp(-AD). \tag{2.7}$$

Poisson distribution model is often used for yield analysis because of its mathematical simplicity [1–5]. It is useful for rough yield estimation or the comparison of redundancy techniques. More precise yield estimation, however, requires a model that takes "fault clustering" into account described below.

2.2.2 Negative-Binomial Distribution Model

It has been reported that actual faults are not randomly distributed but clustered and that the number of faulty elements does not match the Poisson distribution model [6, 7]. In this case, the parameter λ is no longer constant but does distribute. Compound Poisson distribution model

$$P(K) = \int_0^\infty \frac{\lambda^K \exp(-\lambda)}{K!} \cdot f(\lambda) \mathrm{d}\lambda \tag{2.8}$$

was proposed [6] as a distribution model for nonconstant λ. The first factor in the integral is Poisson distribution and the second factor $f(\lambda)$ is a function called "compounder" representing the distribution of λ. The average $\bar{K}$ and standard deviation $\sigma(K)$ of the number of faulty elements are given by the following equations:

$$\begin{aligned} \bar{K} &= \sum_{K=0}^{\infty} KP(K) = \int_0^\infty \left\{ \exp(-\lambda) f(\lambda) \sum_{K=1}^{\infty} \frac{\lambda^K}{(K-1)!} \right\} \mathrm{d}\lambda \\ &= \int_0^\infty \exp(-\lambda) f(\lambda) \cdot \lambda \exp(\lambda) \mathrm{d}\lambda = \int_0^\infty f(\lambda) \cdot \lambda \, \mathrm{d}\lambda = \bar{\lambda}, \end{aligned} \tag{2.9}$$

$$\sigma(K) = \sqrt{\sum_{K=0}^{\infty} K^2 P(K) - \bar{K}^2} = \sqrt{\int_0^\infty \left\{ \exp(-\lambda) f(\lambda) \sum_{K=1}^{\infty} \frac{\lambda^K \cdot K}{(K-1)!} \right\} \mathrm{d}\lambda - \bar{\lambda}^2}$$

$$= \sqrt{\int_0^\infty \left[\exp(-\lambda) f(\lambda) \left\{ \sum_{K=2}^{\infty} \frac{\lambda^K}{(K-2)!} + \sum_{K=1}^{\infty} \frac{\lambda^K}{(K-1)!} \right\} \right] \mathrm{d}\lambda - \bar{\lambda}^2}$$

$$
\begin{aligned}
&= \sqrt{\int_0^\infty \{\exp(-\lambda) f(\lambda)(\lambda^2 \exp\lambda + \lambda \exp\lambda)\} \mathrm{d}\lambda - \bar{\lambda}^2} \\
&= \sqrt{\int_0^\infty f(\lambda) \cdot \lambda \, \mathrm{d}\lambda + \int_0^\infty f(\lambda) \cdot \lambda^2 \, \mathrm{d}\lambda - \bar{\lambda}^2} \\
&= \sqrt{\bar{\lambda} + \{\sigma(\lambda)\}^2}.
\end{aligned} \tag{2.10}
$$

The candidates for $f(\lambda)$ include uniform distribution and triangular distribution. However, Gamma distribution

$$
f(\lambda) = \frac{\lambda^{\alpha-1} \exp(-\lambda/\beta)}{\Gamma(\alpha)\beta^\alpha} \tag{2.11}
$$

has been shown the most suitable for actual fault distribution[1] [6, 8]. The meanings of the parameters α and β are as follows: α corresponds to fault clustering (smaller α means stronger clustering), and the product $\alpha\beta$ is equal to the average of λ, λ_0. The standard deviation of λ is equal to $\beta\sqrt{\alpha}$. Figure 2.3 shows examples of (2.11) for various parameters maintaining $\lambda_0 = \alpha\beta = 1.0$. When $\alpha \to \infty$, the distribution becomes the delta function, corresponding to no λ distribution.

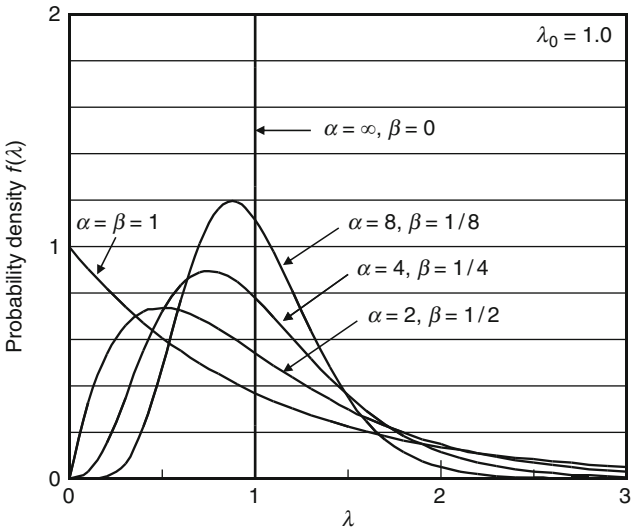


Fig. 2.3 Probability density function of gamma distribution as a compounder (average of $\lambda = 1.0$)

[1] $\Gamma(\alpha)$ is gamma function defined as $\Gamma(\alpha) = \int_0^\infty t^{\alpha-1} \exp(-t)\mathrm{d}t$. $\Gamma(\alpha) = (\alpha - 1)!$ for integer α.

Substituting (2.11) and $\beta = \lambda_0/\alpha$ into (2.8) results in

$$\begin{aligned} P(K) &= \int_0^\infty \frac{\lambda^K \exp(-\lambda)}{K!} \cdot \frac{\lambda^{\alpha-1} \exp(-\lambda/\beta)}{\Gamma(\alpha)\beta^\alpha} \mathrm{d}\lambda \\ &= \frac{1}{K!\Gamma(\alpha)\beta^\alpha} \int_0^\infty \lambda^{K+\alpha-1} \exp\left\{-\left(1+\frac{1}{\beta}\right)\lambda\right\} \mathrm{d}\lambda \\ &= \frac{1}{K!\Gamma(\alpha)\beta^\alpha} \left(\frac{\beta}{\beta+1}\right)^{K+\alpha} \int_0^\infty t^{k+\alpha-1} \exp(-t) \mathrm{d}t \\ &= \frac{\Gamma(K+\alpha)\beta^K}{K!\Gamma(\alpha)(\beta+1)^{K+\alpha}} \\ &= \frac{\alpha(\alpha+1)\cdots(\alpha+K-1)(\lambda_0/\alpha)^K}{K!(1+\lambda_0/\alpha)^{K+\alpha}}. \end{aligned} \tag{2.12}$$

This is called *negative binomial distribution* [9]. The average and standard deviation of the number of faulty elements are calculated from (2.9) and (2.10):

$$\bar{K} = \bar{\lambda} = \lambda_0, \tag{2.13}$$

$$\sigma(K) = \sqrt{\bar{\lambda} + \{\sigma(\lambda)\}^2} = \sqrt{\lambda_0 + \beta^2\alpha} = \sqrt{\lambda_0(1+\lambda_0/\alpha)}. \tag{2.14}$$

Comparing (2.14) with (2.5), we can find that the standard deviation of the negative-binomial distribution is larger than that of Poisson distribution by a factor of $\sqrt{1+\lambda_0/\alpha}$. The raw yield is expressed as

$$P(0) = \frac{1}{(1+\lambda_0/\alpha)^\alpha} = \frac{1}{(1+AD/\alpha)^\alpha}. \tag{2.15}$$

When $\alpha \to \infty$, (2.15) becomes identical to (2.7). Figures 2.4 and 2.5 show examples of the distribution with $\alpha = 4.0$ (weaker fault clustering) and $\alpha = 1.0$ (stronger fault clustering), respectively. Compared with Fig. 2.2 (corresponding to the case $\alpha = \infty$), the probability for $K = 0$ and that for large K increase and the probability for medium K decreases as α decreases. Equations (2.7) and (2.15) are plotted in Fig. 2.6. The raw yield using the Poisson distribution model is expressed by the straight line ($\alpha = \infty$) in semilog scale. The raw yield using the negative-binomial distribution model is expressed by a concave-up line and is greater than that using the Poisson model.

The negative-binomial distribution model is often used for yield estimation of memory LSIs [10–12] because it gives good agreement with actual fault distribution. In order to use this model, however, we must determine two parameters λ_0 (average number of faults) and α (fault clustering factor) from experimental data. In addition, it should be noted that the parameter α may depend on the kind of the memory element.

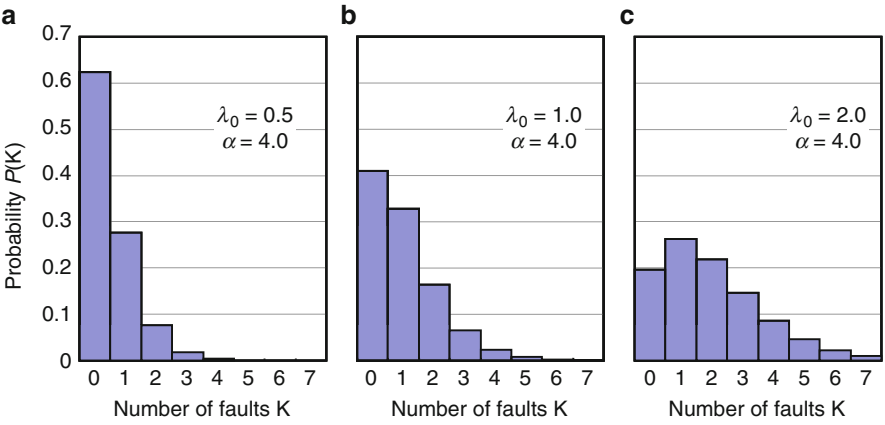


Fig. 2.4 Examples of negative binomial distribution with $\alpha = 4.0$: (**a**) $\lambda_0 = 0.5$, (**b**) $\lambda_0 = 1.0$, and (**c**) $\lambda_0 = 2.0$

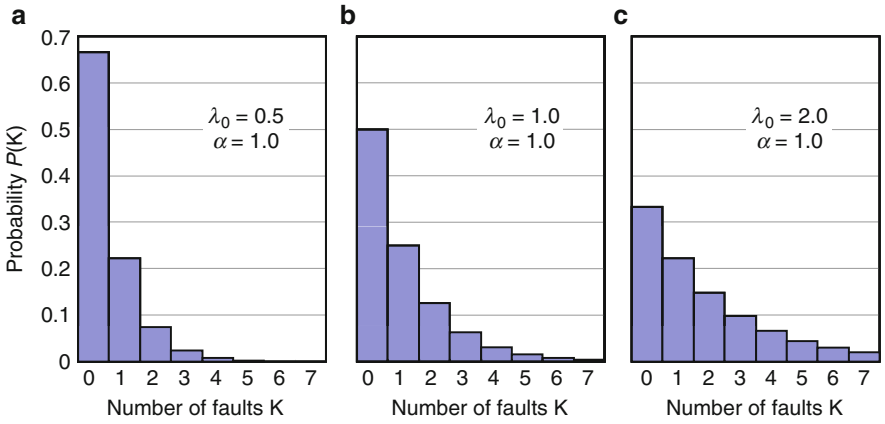


Fig. 2.5 Examples of negative binomial distribution with $\alpha = 1.0$: (**a**) $\lambda_0 = 0.5$, (**b**) $\lambda_0 = 1.0$, and (**c**) $\lambda_0 = 2.0$

2.3 Yield Improvement Through Redundancy

In this section, yield improvement through redundancy is analyzed using the models described above. We assume the followings for simplicity:

1. Faults on spare elements are neglected.
2. Fatal faults are neglected. A fatal fault is defined as a fault that makes the entire chip no good. For example, a defect on peripheral circuit of a memory LSI may cause a fatal fault.

Without redundancy, the yield Y_0 is equal to $P(0)$ as shown in Fig. 2.6 because only chips without faulty elements are accepted. If R spare elements

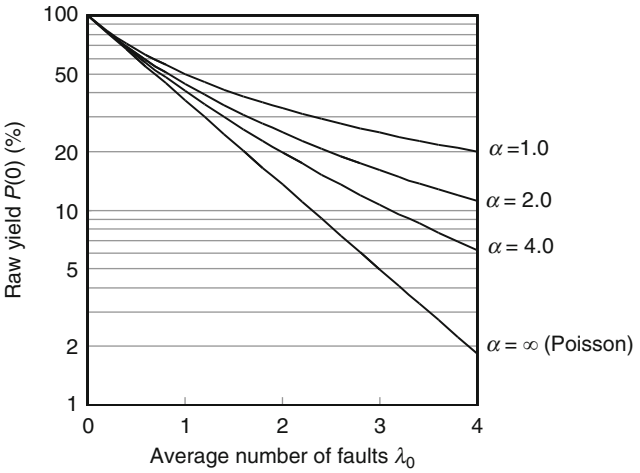
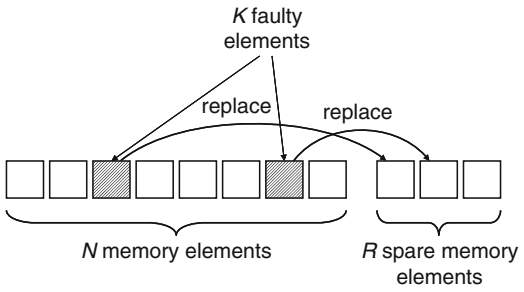


Fig. 2.6 Comparison of raw yield using Poisson and negative-binomial distribution models

Fig. 2.7 Principle of redundancy



are added in the chip, chips with K faulty elements ($K \leq R$) become acceptable by replacing the faulty elements with spares as shown in Fig. 2.7. Therefore, the yield becomes

$$Y = \sum_{K=0}^{R} P(K). \tag{2.16}$$

Figures 2.8 and 2.9 show the calculated yield using the Poisson distribution and the negative-binomial distribution models, respectively. The raw yield Y_0 ($R = 0$) is lower but the yield improvement is larger with Poisson distribution model than with negative-binomial model. This is apparent in Figs. 2.10 and 2.11, where the relationships between yields with and without redundancy are plotted. Thus, it should be noted that using Poisson distribution model tends to underestimate Y_0 and overestimate the yield improvement.

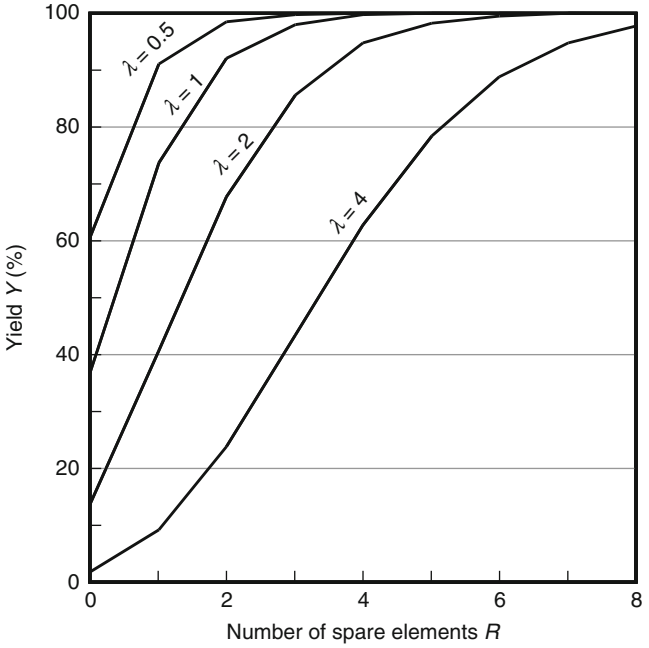


Fig. 2.8 Yield improvement through redundancy using Poisson distribution model

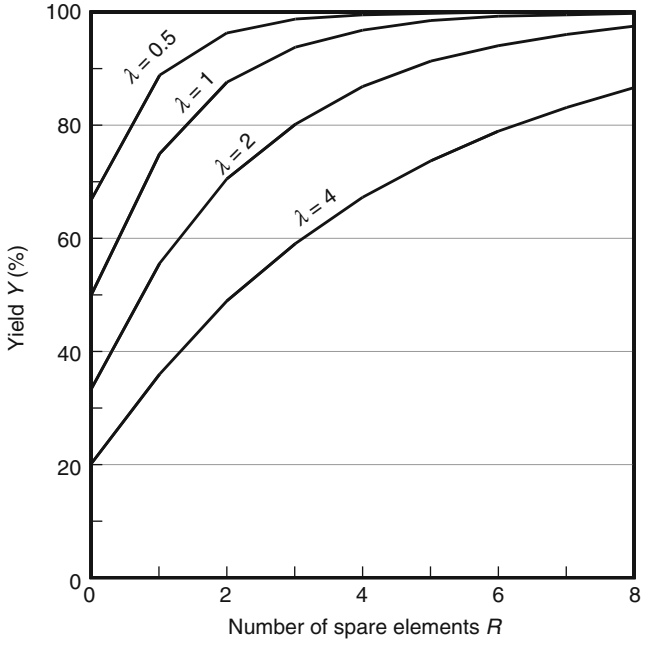


Fig. 2.9 Yield improvement through redundancy using negative-binomial distribution model ($\alpha = 1.0$)

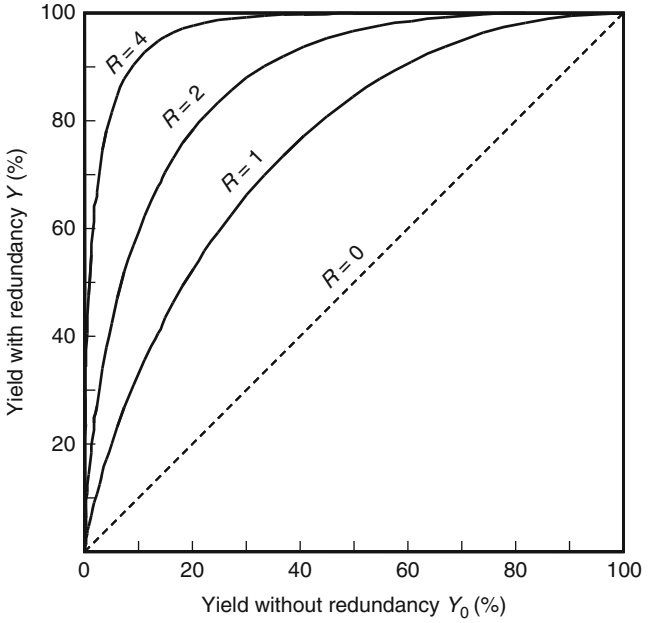


Fig. 2.10 Yield improvement through redundancy using Poisson distribution model: number of spare elements as a parameter

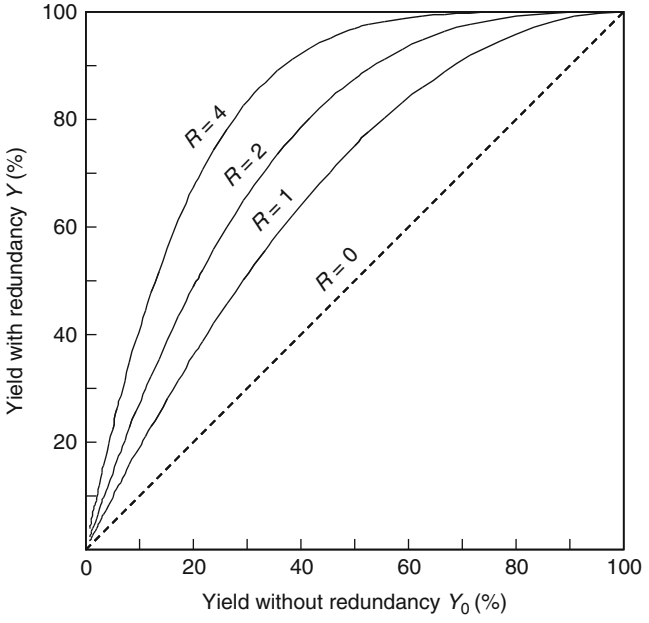


Fig. 2.11 Yield improvement through redundancy using negative-binomial distribution model ($\alpha = 1.0$): number of spare elements as a parameter

2.4 Replacement Schemes

2.4.1 *Principle of Replacement*

Since redundancy techniques repair memory LSIs by replacing faulty normal elements by spare elements and by hiding the faulty elements from users, the following steps are required.

1. Replacement information (which normal element is replaced by which spare element) is stored in on-chip programmable devices in advance.
2. When accessed, it is judged whether the demanded address is faulty ("hit") or not ("miss") using the information stored above.
3. In case of miss, the normal element is activated and no spare elements are activated.
4. In case of hit (a) the normal element is inhibited from being activated and (b) the spare element that replaces the normal element is instead activated.

Step 1 requires programmable devices. In addition, they must be nonvolatile to retain the programmed replacement information during power-off. The devices used for this purpose include fuses, antifuses, and nonvolatile memory cells as is described in Sect. 2.8. There are two schemes for storing and reading the replacement information in the storage (steps 1 and 2): decoder programming and address comparison. The former utilizes spare decoders, which is the same as normal decoders except that the address is programmable (Fig. 2.12). The latter utilizes comparators for selecting spare elements (Fig. 2.13). In addition, there are two schemes for disabling the faulty normal element (step 4a): direct disabling and

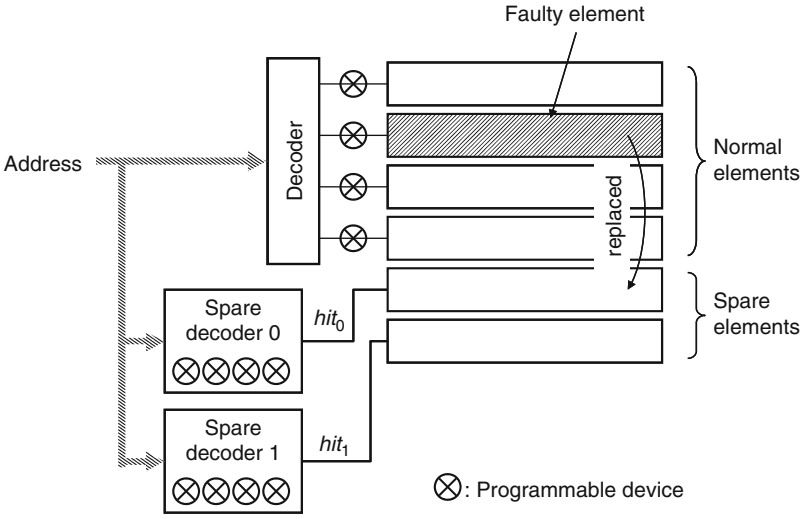


Fig. 2.12 Replacement using decoder programming and direct disabling schemes

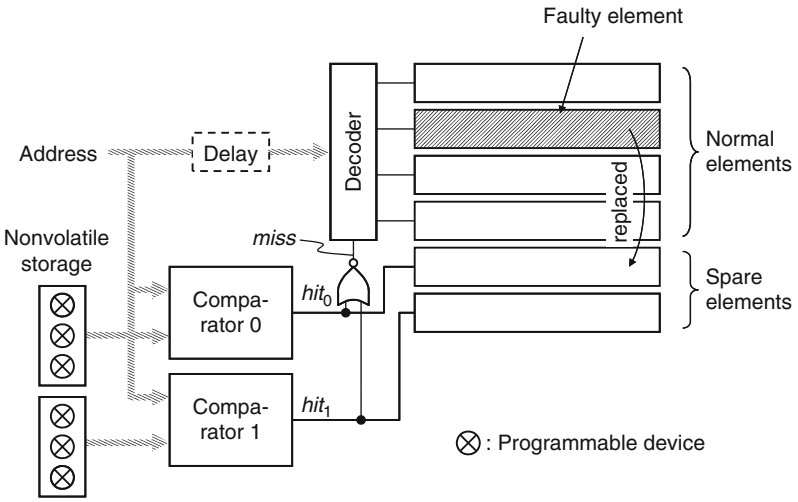


Fig. 2.13 Replacement using address comparison and indirect disabling schemes

indirect disabling. The former cuts off the signal path to the faulty element by blowing a fusible link (Fig. 2.12). The latter utilizes the spare-selection signals hit_i (Fig. 2.13). The activation of normal element is inhibited if one of the hit_i is asserted, and is allowed if none of the hit_i is asserted. Therefore, there are four (2×2) possible combinations. Figure 2.12 shows a replacing scheme using decoder programming and direct disabling schemes, while Fig. 2.13 shows a scheme using address comparison and indirect disabling schemes. On the other hand, there is a quite different replacing scheme, shifting scheme, as shown in Fig. 2.14. The switches inserted between the decoder and the memory array are controlled by the programmable storage, in which the address of a faulty element is programmed. In case of no faulty elements, each output of the decoder is connected with the corresponding normal element and the spare element is not connected. If there is a faulty normal element, the connections are shifted so that the outputs of the decoder are connected with the spare element and the normal elements except for the faulty one.

2.4.2 Circuit Implementations

Next, circuit implementations of the replacement schemes are described. Figure 2.15 shows a spare row decoder using the decoder programming scheme [13]. The circuit has two fuses per an address bit, one for true signal a_i and the other for complement signal $\bar{a}_i$. One of the two fuses is blown by laser according to the faulty address. The node N is precharged to high level by signal PC before the input of address signals. If the input address coincides with the programmed address the node N remains high.

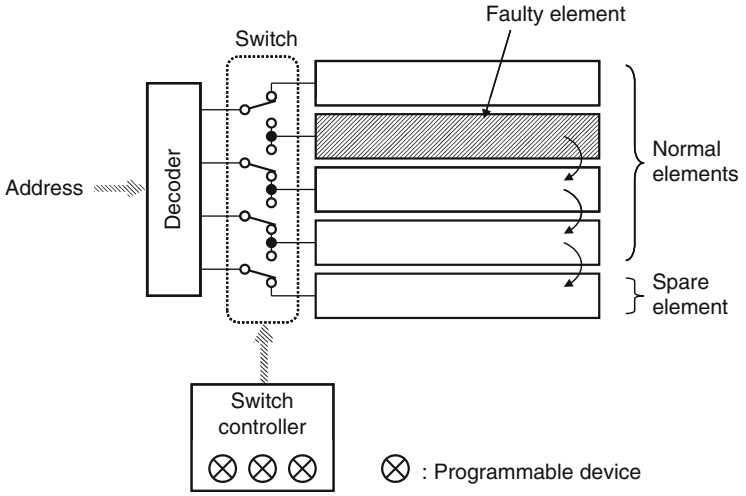


Fig. 2.14 Replacement using shifting scheme

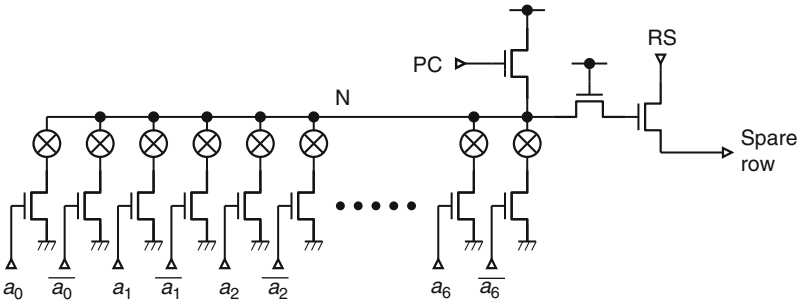


Fig. 2.15 Spare row decoder using decoder programming scheme. Reproduced from [13] with permission; © 2010 IEEE

The spare row line is activated when row selection signal RS goes high. If the input address does not coincide with the programmed address, node N goes low and the spare line is not activated. If no faulty address, all the fuses are unblown. Since node N always goes low irrespective of the input address, the spare row line is never activated. Figure 2.16 shows a normal row decoder using the direct disabling scheme [13]. The output of the NOR gate is ANDed with the predecoder output signals RS_0–RS_3 to relax the circuit pitch. A normal row is disconnected to the output of the decoder by blowing the corresponding fuse by laser. Thus, this circuit requires as many fuses as the rows and the pitch of fuses is equal to row pitch. Figure 2.17 shows a row redundancy circuit using the address comparison and indirect disabling schemes [14]. Each address comparator unit stores a faulty address and compares it with the input address to generate the spare-selection signal hit_i. If hit_i is asserted, the corresponding spare row is activated at the timing

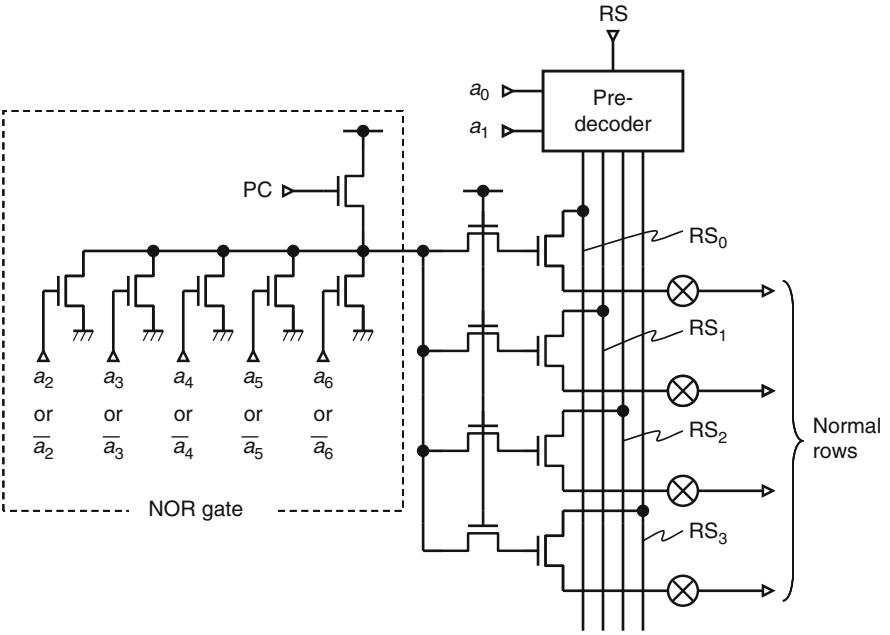


Fig. 2.16 Normal row decoder using direct disabling scheme [13]

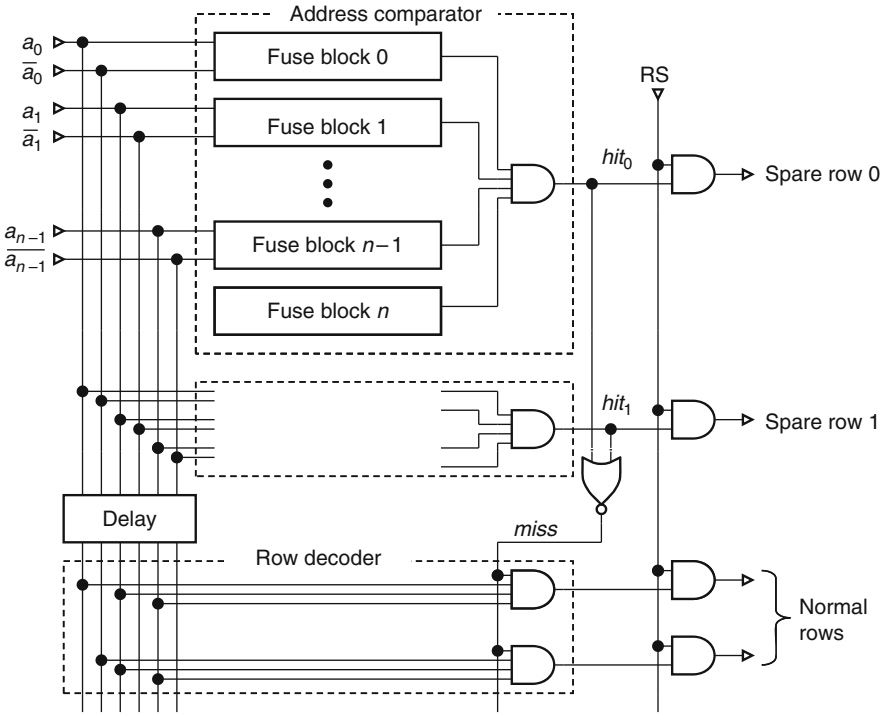


Fig. 2.17 Row replacement using address comparison and indirect disabling schemes [14]

of RS and the row decoder is disabled through the NOR gate. Note that a delay circuit is inserted at the input of the row decoder. Without this delay, the arrival of the address signals at the row decoder precedes the generation of hit_i, and a normal row may be wrongly activated. Therefore, this circuit has an access time penalty of the delay time. The detail of the fuse block included in the address comparator is shown in Fig. 2.18 [14, 15]. This circuit utilizes an electrically blown fuse. The blowing MOS transistor M_0 is controlled by programming signal $\bar{P}$ and address signal $\bar{a}_i$. If the fuse is blown, the signal $\bar{f}_i$ is pulled down to low and f_i is high. Since M_1 is on and M_2 is off, the output signal c_i is equal to a_i. On the contrary, if the fuse is unblown, c_i is equal to $\bar{a}_i$. The programming and normal operations of the fuse block are summarized in Table 2.1. Note that $c_i = 1$ if the address during programming is equal to the address during normal operation and otherwise $c_i = 0$. All the outputs of the fuse blocks are ANDed to generate the hit signal and to activate the corresponding spare row line. An extra fuse block is added to indicate whether the address comparator is valid or not. Without blowing the fuse in this extra block, the comparator is invalid and the corresponding spare row line is never activated.

The number of fuses required for each scheme is as follows. The decoder programming scheme require $2nR$, while the address comparison requires $(n + 1)$ R (including the fuses for extra blocks), where $n = \log_2 N$ is the number of address bits, N is the number of normal lines, and R is the number of spare lines. The direct disabling scheme requires 2^n because the number of normal lines is $N = 2^n$ while the indirect disabling scheme requires none. Therefore, the combination of address

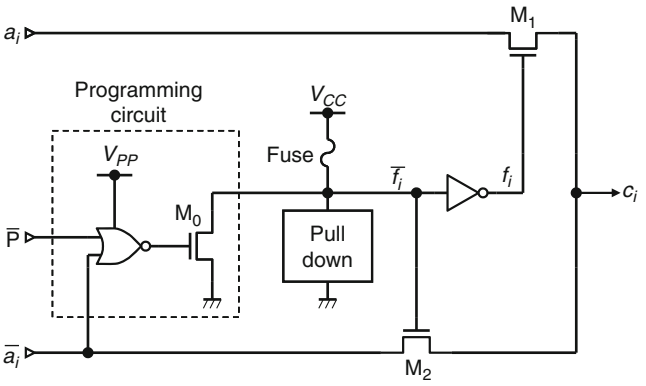


Fig. 2.18 Detail of fuse block [14, 15]

Table 2.1 Truth table of fuse block

Programming ($\bar{P} = 0$)					Normal operation		
a_i	$\bar{a}_i$	Fuse	f_i	$\bar{f}_i$	a_i	$\bar{a}_i$	c_i
0	1	Unblown	0	1	0	1	1
					1	0	0
1	0	Blown	1	0	0	1	0
					1	0	1

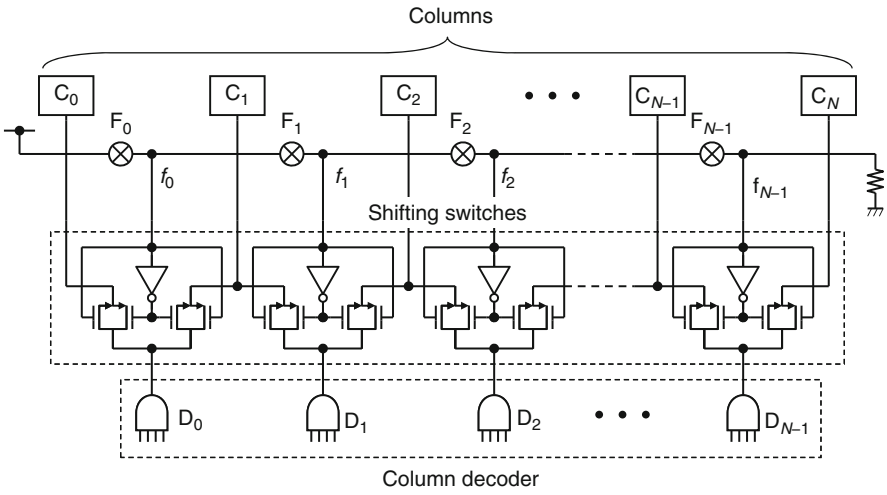


Fig. 2.19 Column replacement using shifting scheme. Reproduced from [16] with permission; © 2010 IEEE

comparison and indirect disabling schemes requires the fewest fuses. Since the size of fuses is not so scaled down according to design rules, the combination is the best choice for high-density memory LSIs with a large number of spare lines, despite the access penalty.

Figure 2.19 shows the shifting scheme applied to column redundancy of a high-speed SRAM [16]. Shifting switches are inserted between the outputs of the column decoder, D_0–D_{N-1}, and the columns, C_0–C_N, including a spare column C_N. The connections are changed by blowing one of the fuses, F_0–F_{N-1}. If no fuses are blown, all the control signals f_0–f_{N-1} are at high level. Column C_0 is activated by D_0, C_1 is activated by D_1, and so on. The spare column C_N is never activated. If, for example, column C_1 is found to be faulty, fuse F_1 is blown. Since f_0 is at high level and f_1–f_{N-1} are at low level, C_0 is activated by D_0, C_2 is activated by D_1, and C_N is activated by D_{N-1}. Thus, the activation of the faulty column C_1 is inhibited. This scheme requires N fuses. However, it features no access-time degradation because all the signal-path lengths, including the spare one are uniform.

The shifting scheme is suitable for the row redundancy of a content addressable memory (CAM) because a CAM has a priority encoder (PE). If the search data matches the data stored in two or more rows, the PE outputs the address of the row having the highest priority. The order of priority is usually ascending or descending order of row address. If a faulty row is replaced by a spare row with the ordinary replacing scheme, the order of address is not maintained. With the shifting scheme, however, the order of row address is maintained even after shifting [17, 18]. Figure 2.20 shows a CAM with row redundancy using the shifting scheme. The information of faulty address stored in the fuses is decoded and is transferred to the register FR in advance and is used for switching both wordlines and matchlines.

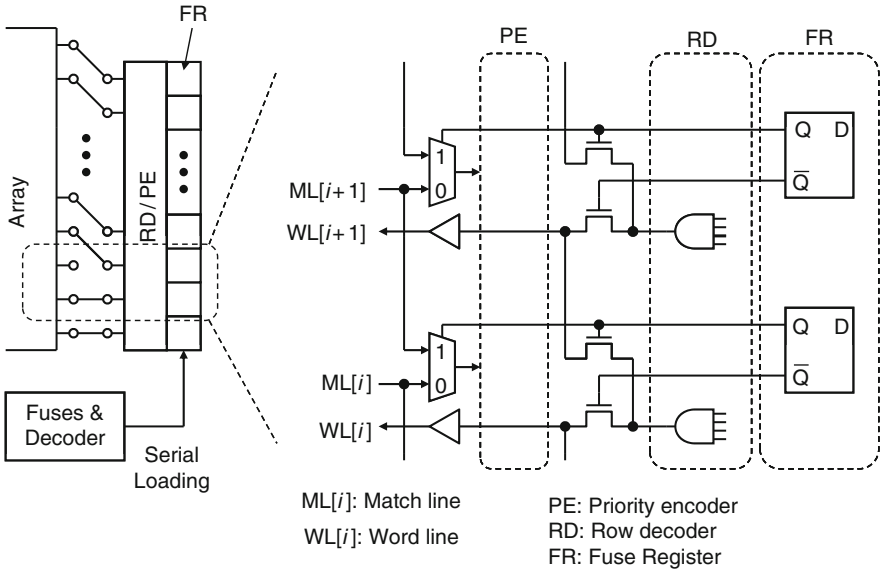


Fig. 2.20 Row replacement of a CAM using shifting scheme. Reproduced from [17] with permission; © 2010 IEEE

The shifting scheme is also suitable for wide I/O memories. Figure 2.21 shows the shifting scheme applied to a DRAM macro with 128 I/Os [19]. The macro has 128 normal blocks and two spare blocks on both sides. The redundancy scheme applied to this macro features that the connections of switches are variable according to column address (the connections of the switches in Figs. 2.19 and 2.20 are fixed after blowing fuses). A column of each block is selected by multiplexer MUX controlled by column selection lines CSL. When column 0 is selected (Fig. 2.21a), only normal blocks are connected to the I/O lines because column 0 of each block is not faulty. However, when another column is selected (Fig. 2.21b, c), the connections are changed so that the faulty columns are not connected to the I/O lines. The detail of the shifting switch is shown in Fig. 2.22. Faulty block addresses of each column are programmed in the shift point register SPR by blowing fuses. The faulty block addresses of the selected column, FB_0 and FB_1, are read out and compared with each block address generated by a wired logic circuit. The connection of each switch is determined by the comparison results as shown in the table.

Although the circuit in Fig. 2.19 requires N fuses, the circuit in Fig. 2.20 requires only $\log_2 N + 1$ because of decoding. The circuit in Fig. 2.21 requires $2(\log_2 N + 1)$ because there are two spare columns. The number of programmable elements required for replacement schemes is summarized in Table 2.2.

A disadvantage of the shifting scheme is that the number of spares is limited. Since R sets of spares require $(R + 1)$-terminal shifting switches, a large R causes the complexity of switches and the control circuits. Therefore, $R = 1$ or 2 is practical.

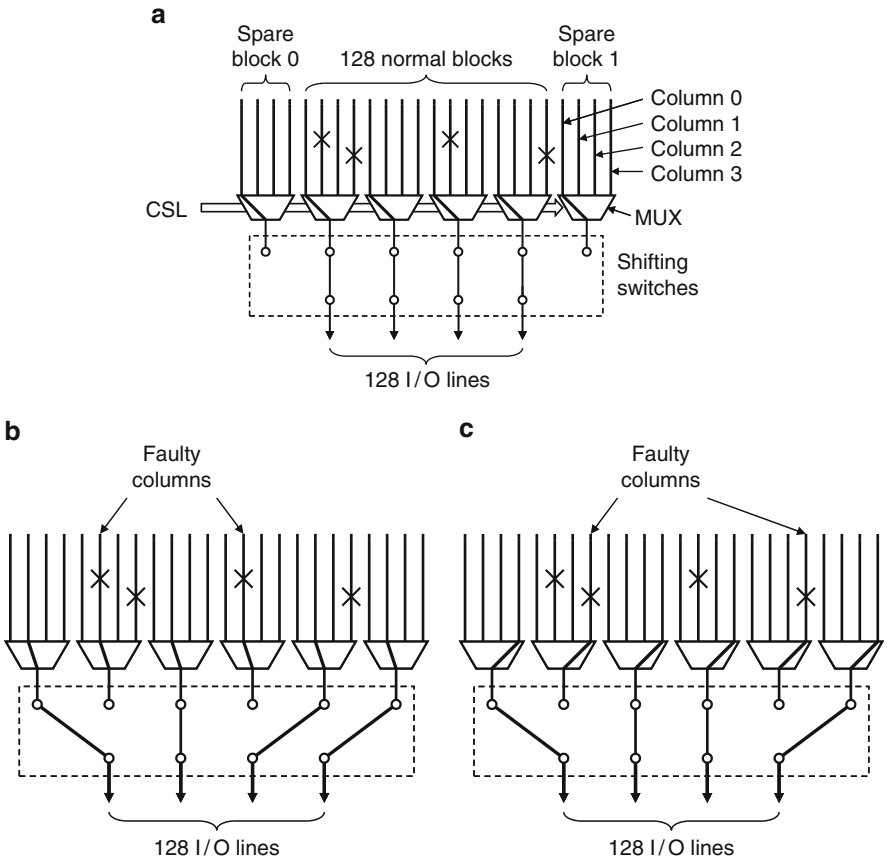


Fig. 2.21 Flexible DQ replacement of an embedded RAM using shifting scheme: (**a**) nonfaulty case, (**b**) two faulty columns are replaced, and (**c**) when another address is selected, other two faulty columns are replaced. Reproduced from [19] with permission; © 2010 IEEE

2.5 Intrasubarray Replacement

One of the problems for redundancy with the increase in memory capacity is memory-array division. Figure 2.23 shows the trend of memory-array division of DRAMs [20]. The number of subarrays doubled each generation before 64 Mbit. This is mainly due to the dataline (bitline) division shown in Fig. 2.24. The dataline D is divided into D_0–D_3 to reduce the parasitic capacitance for signal/noise ratio enhancement and charging/discharging current reduction [21–24]. The number of divisions even quadrupled each generation after the introduction of hierarchical wordline structure as shown in Figs. 2.25 and 2.26 [25–27]. Here, a wordline is divided into a plurality of sub wordlines (SWLs), each of which is activated by the AND of a main wordline (MWL) and a block selection line BS.

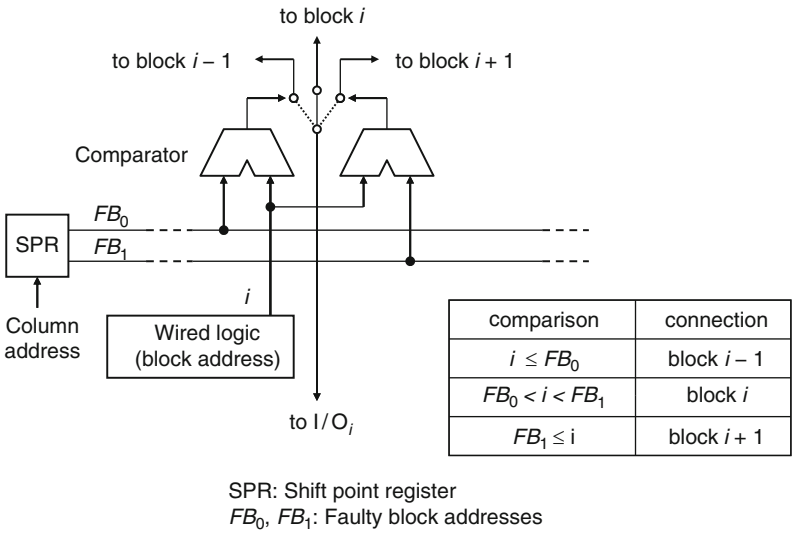


Fig. 2.22 Detail of shifting switch and its control circuit [19]

Table 2.2 Number of programmable elements required for replacement

Replacement scheme	Number of programmable elements	Figure
Decoder programming + direct disabling	$2R \log_2 N + N$	2.15, 2.16
Decoder programming + indirect disabling	$2R \log_2 N$	–
Address comparison + direct disabling	$R(\log_2 N + 1) + N$	–
Address comparison + indirect disabling	$R(\log_2 N + 1)$	2.17
Shifting (without decoding)	N $(R = 1)$	2.19
Shifting (with decoding)	$R(\log_2 N + 1)$	2.20, 2.21

These memory array divisions degrade yield because the boundaries between subarrays work as barriers to the faulty-element replacement and reduce the replacement flexibility.

There are two approaches to cope with this problem. One is to enhance the replacement flexibility under the restriction of intrasubarray replacement, that is, a faulty element is replaced only by a spare element in the same subarray. The other is to allow intersubarray replacement, that is, a faulty element can be replaced by a spare element in another subarray. The former is explained in this section and the latter is described in the next section.

Here, let us define two terms used in the following discussion. *Replacement unit* is defined as a set of memory cells replaced simultaneously (by programming one set of programmable devices). The replacement unit is assumed as a row/column so far. However, adjacent two rows/columns are often replaced simultaneously in actual memory design. This is realized by neglecting the least significant bit of row/column address. In this case the replacement unit is two rows/columns. In addition, in the case of the simultaneous replacement described below, all the rows/columns in all the subarrays are replaced simultaneously. In this case the

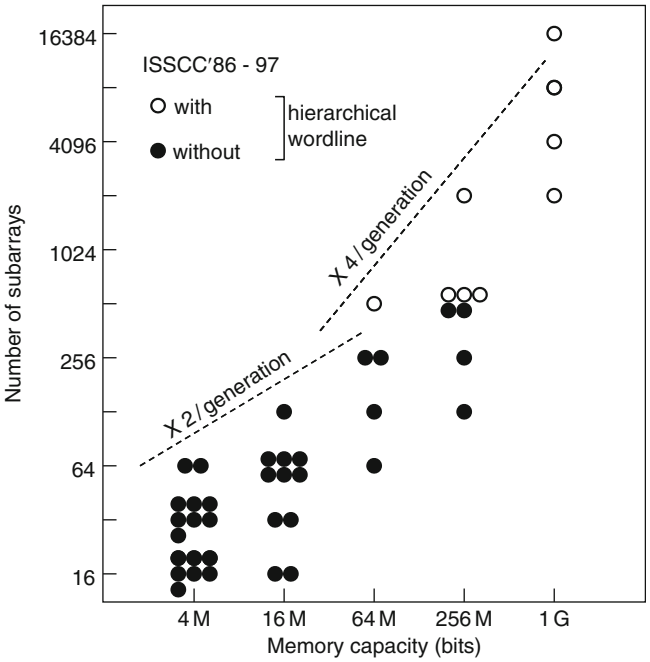


Fig. 2.23 Trend of DRAM memory-array division. Reproduced from [20] with permission; © 2010 IEEE

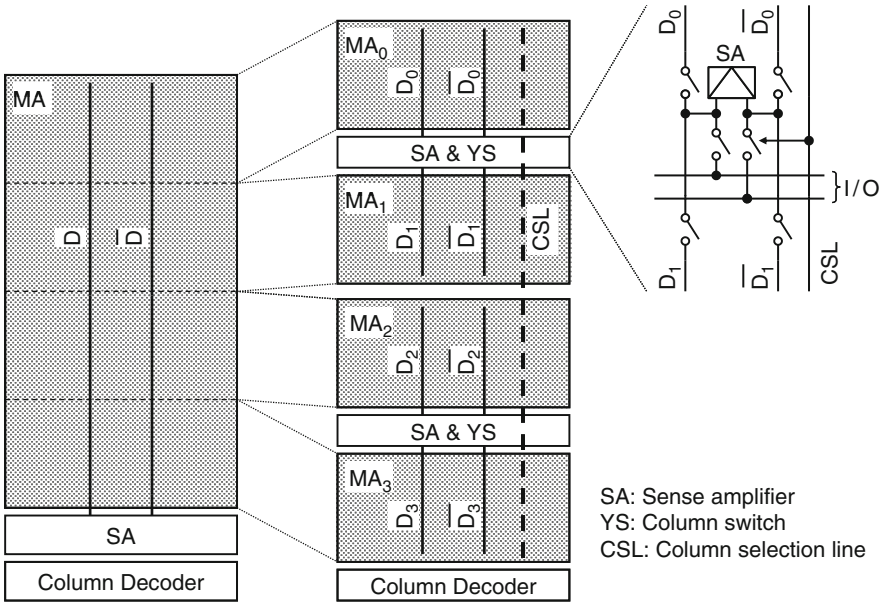


Fig. 2.24 Memory-array division using multidivided dataline (bitline) structure [21, 23, 24]

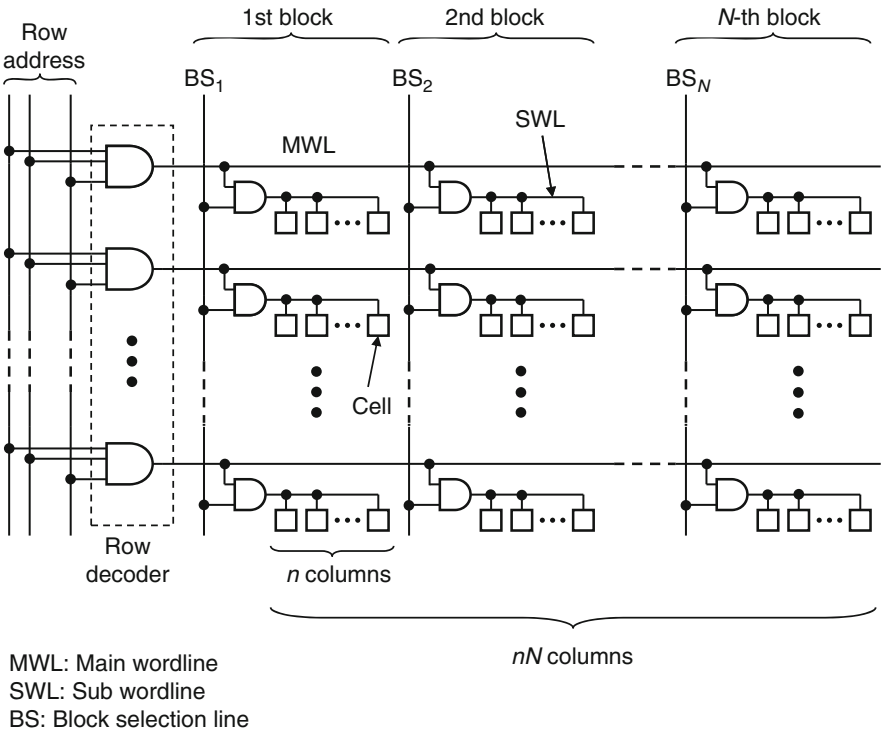


Fig. 2.25 Memory-array division using hierarchical wordline structure for high-speed SRAMs [25]

replacement unit is M rows/columns, where M is the number of subarrays. *Replacement region* is defined as an area in which any normal row/column can be replaced by any spare row/column. The replacement region is a subarray in the case of intrasubarray replacement, while it is plural subarrays in the case of intersubarray replacement.

2.5.1 Simultaneous and Individual Replacement

Figure 2.27 shows the row redundancy technique applied to a memory without array division. The memory has L (here, $L = 4$) spare wordlines SW_0–SW_3 and as many address comparators AC_0–AC_3. Faulty word addresses are programmed in the address comparators and are compared with the input address. Thus, at most L faulty normal wordlines can be repaired. In this example, faulty normal wordlines W_0–W_3 are replaced by spare wordlines SW_0–SW_3, respectively, as shown by the arrows in the figure. Now let us consider dividing the memory array into subarrays. Two approaches within the restriction of intrasubarray replacement are shown in Figs. 2.28 and 2.29. Here the memory array MA in Fig. 2.27 is divided into four

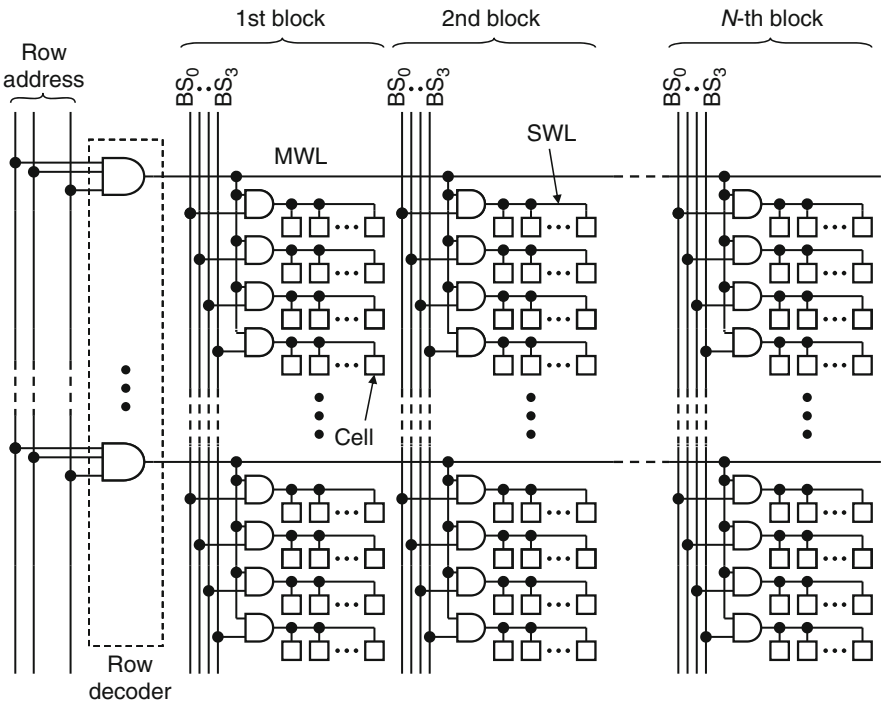


Fig. 2.26 Memory-array division using hierarchical wordline structure for high-density DRAMs [26, 27]

subarrays MA_0–MA_3, only one of which is activated. Among m row address bits, the upper two bits a_{m-2} and a_{m-1} (intersubarray address signals) are used for selecting a subarray and the lower $(n-2)$ bits a_0–a_{m-3} (intrasubarray address signals) are used for selecting a wordline in the subarray.

In the simultaneous replacement (Fig. 2.28), the number of address comparators is equal to L, the number of spare wordlines in each subarray. The total number of spare wordlines is therefore LM, where $M(=4)$ is the number of subarrays. Each address comparator compares only the intrasubarray address signals, and the output is commonly supplied to every subarray. The intersubarray address signals, in turn, select one of the four spare wordlines. As many faulty wordlines can be repaired as are shown in Fig. 2.27, if L is the same. In this approach, four normal lines are replaced simultaneously by spare lines as shown by the arrows in Fig. 2.28. That is, to replace one faulty line, three other normal lines with the same intrasubarray address are also replaced even if they are not faulty. The replacement unit is therefore four wordlines. This causes the following problems. First, the usage efficiency of spare lines is lower (25% in this case) and the number of spare lines should be larger, resulting in chip-area increase. Second, the probability of unsuccessful repair due to faults on the spare lines that replaced normal lines is higher, resulting in yield degradation.

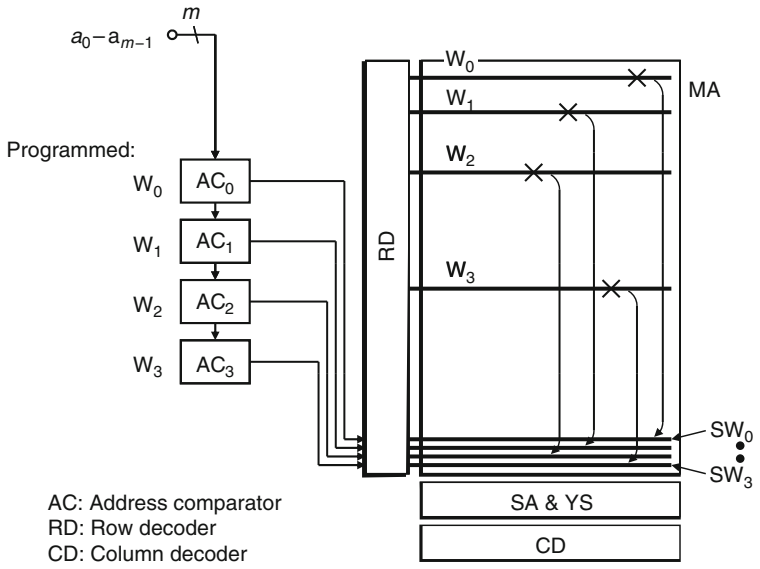


Fig. 2.27 Row redundancy applied to a memory without array division

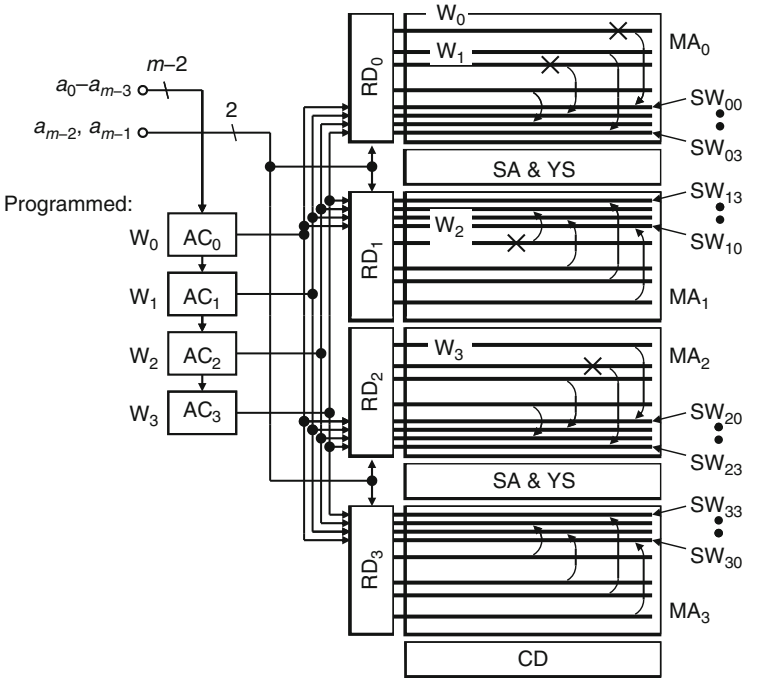


Fig. 2.28 Simultaneous intrasubarray replacement applied to a memory with array division

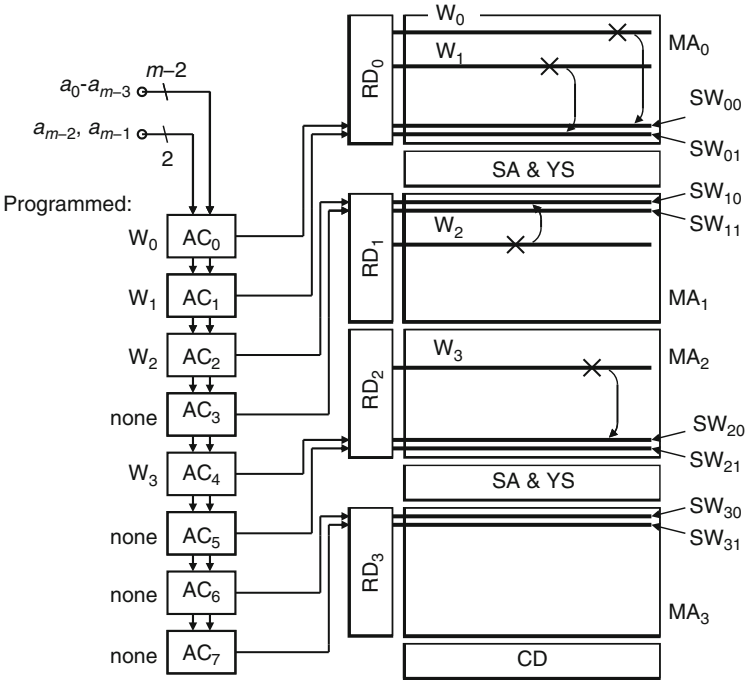


Fig. 2.29 Individual intrasubarray replacement applied to a memory with array division

In the individual replacement (Fig. 2.29), every spare line in every subarray has its own address comparator and the replacement unit is one wordline. The number of address comparators is therefore LM. Each address comparator compares both intra- and intersubarray address bits. This approach has the following advantages over the simultaneous replacement. First, a smaller L is statistically required (here, $L = 2$) to repair as many faults, if faults are randomly distributed. Second, since only one normal line is replaced at a time by a spare line, the probability of a fault on the spare line is lower. This approach, however, has the disadvantage of lower usage efficiency of address comparators (50% in this case). The number of address comparators should be larger, resulting in chip-area increase.

Since there are only four subarrays in Figs. 2.28 and 2.29, the problems described above are not so serious. However, they can be critical in the design of ultrahigh-density memories in nanometer era because of the aforementioned trend of the number of subarrays to increase.

2.5.2 Flexible Replacement

Figure 2.30 shows the flexible intrasubarray replacement scheme [5, 20] proposed to overcome the problem described above. The spare lines and address comparators

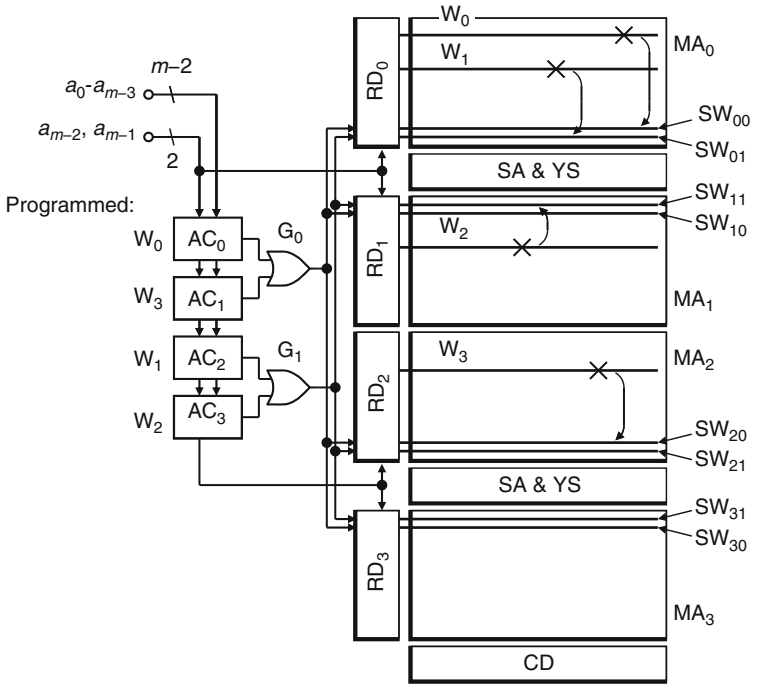


Fig. 2.30 Flexible intrasubarray replacement applied to a memory with array division [5, 20]

are not directly connected but through the OR gates G_0 and G_1. Since each address comparator compares both intra- and intersubarray address bits, the replacement unit is one wordline. Each logical-OR of the outputs of the two address comparators is commonly applied to all the subarrays. The intersubarray address bits select one of the four spare wordlines. In this example, the addresses of faulty normal wordlines W_0, W_1, W_2, and W_3 are programmed in address comparator AC_0, AC_2, AC_3, and AC_1, respectively. Wordlines W_0, W_1, W_2, and W_3 are thereby replaced by spare wordlines SW_{00}, SW_{01}, SW_{11}, and SW_{20}, respectively. This technique features a flexible relationship between spare lines and address comparators. In Figs. 2.28 and 2.29, this relationship is fixed so that a spare line can be activated only by a particular address comparator. For example, spare wordline SW_{20} in Fig. 2.28 is activated only by AC_0, and SW_{20} in Fig. 2.29 is activated only by AC_4. However, in Fig. 2.30, a spare line can be activated by one of several address comparators. For example, spare wordline SW_{20} can be activated by either AC_0 or AC_1 through OR gate G_0. In addition, an address comparator can activate one of several spare lines. For example, SC_0 can activate SW_{00}, SW_{10}, SW_{20}, or SW_{30}. This flexible relationship provides the following advantages. First, both spare-line usage efficiency and address comparator usage efficiency are good, while the former is poor and the latter is good in the simultaneous replacement (Fig. 2.28), and the former is good and the latter is poor in the individual replacement (Fig. 2.29). This

enables smaller chip-area penalty due to redundancy. Second, the probability of unsuccessful repair is low, while it is high with the simultaneous replacement. This is because only one normal line is replaced at a time by a spare line. Third, more flexible selection of the number of spare lines in a subarray L and the number of address comparators R enable a more efficient redundancy circuit. Generally, the following relationship stands between L and R:

$$L \leqslant R \leqslant \frac{ML}{M_0}, \tag{2.17}$$

where M is the number of physical subarrays, and M_0 is the number of subarrays in which faulty normal lines are simultaneously replaced by spare lines. Therefore, M/M_0 is the number of logically independent subarrays. The left-hand inequality sign indicates that the number of spare lines in a subarray in excess of the number of address comparators is useless. The right-hand inequality sign indicates that the number of address comparators in excess of the number of logically independent spare lines is useless (ML/M_0 is the number of logically independent spare lines in a whole memory). The relationship between L and R is fixed: $M_0 = M$ and $R = L$ in the simultaneous replacement, and $M_0 = 1$ and $R = ML$ in the individual replacement. In the flexible replacement, however, L and R can be chosen independently as long as the relationship (2.17) is satisfied. The characteristics of the intrasubarray replacement techniques are summarized in Table 2.3, which also include those of intersubarray replacement described in the next section.

The flexible replacement can also be applied to column redundancy as shown in Fig. 2.31. The memory array is divided into four subarrays similar to Figs. 2.28–2.30. Each column selection line CSL transmits the output of the column decoder to column switches (see Fig. 2.24 in detail). There is a spare column composed of a spare column selection line (SCSL) and spare datalines SD_0–SD_3.

Table 2.3 Characteristics of intrasubarray and intersubarray replacement techniques

	Replacement unit	Replacement region	No. of address comparators R	Usage of spare lines	Usage of address comparators	Figure
Intrasubarray						
Simultaneous replacement	M lines	Subarray	L	Poor	Good	2.28
Individual replacement	One line	Subarray	ML	Fair	Poor	2.29
Flexible replacement	One line	Subarray	$L \leq R \leq ML$	Fair	Good	2.30, 2.31
Intersubarray						
Distributed spare lines	One line	Chip	L	Good	Good	2.39
Concentrated spare lines	One line	Chip	L'	Very good	Very good	2.40

M is the number of subarrays, L is the number of spare lines in a subarray, and L' is the number of spare line in the spare subarray

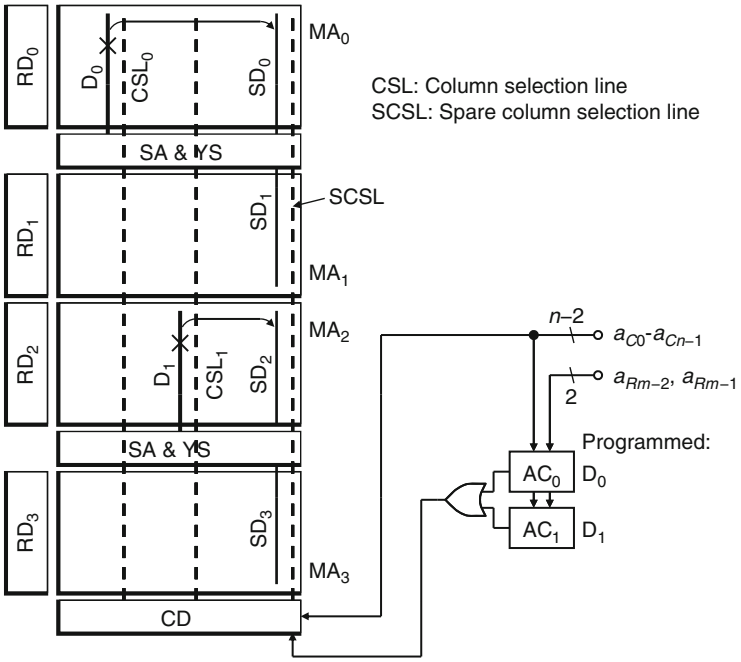


Fig. 2.31 Flexible intrasubarray replacement applied to column redundancy

The line SCSL is activated by the OR of two address comparators AC_0 and AC_1. Note that not only column address bits $a_{C_0} - a_{C_{n-1}}$ but also the upper two of row address bits, $a_{R_{m-2}}$ and $a_{R_{m-1}}$ (intersubarray address signals) are programmed in each comparator. In this example, the addresses of faulty normal datalines D_0 and D_1 are programmed in AC_0 and AC_1, respectively. Thus, the two datalines are respectively replaced by spare datalines SD_0 and SD_2 although there is only one spare column. Further improving the efficiency of column redundancy by using the results of row-address comparison is proposed in [28].

Let us calculate the repairable probability of the flexible intrasubarray replacement as a function of the number of faults. Note that the probability depends not only on the total number of faults but also on their distribution. Random fault distribution is assumed, and faults on spare lines and fatal faults are neglected here for simplicity. The repairable probability in the case of K faults being distributed in M subarrays is denoted as $Y(K, M)$. In the case of $M = 1$ (no array division), it is obvious that

$$Y(K,1) = 1(k \leqslant L \text{ and } k \leqslant R), \quad 0(k > L \text{ or } k > R). \tag{2.18}$$

Let us focus on a particular subarray M_0. If k faults are located in M_0, the remaining $(K - k)$ faults are in the other $(M - 1)$ subarrays M_1–M_{M-1} as shown in Fig. 2.32. The probability that k faults among K faults are located to M_0 is given by the binomial distribution:

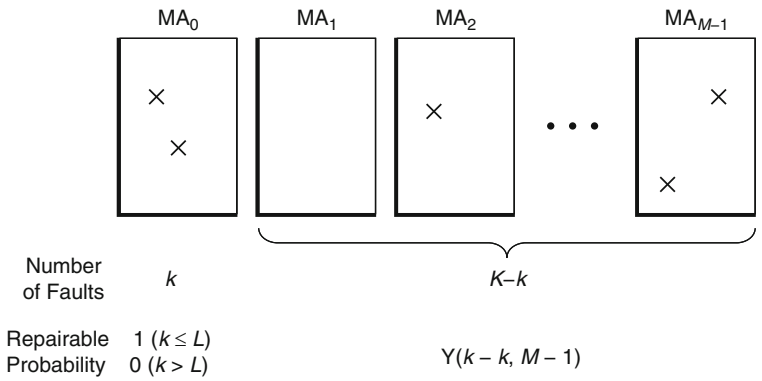


Fig. 2.32 Calculation of repairable probability of intrasubarray replacement

$$P(k) = \binom{K}{k}\left(\frac{1}{M}\right)^{k}\left(1-\frac{1}{M}\right)^{K-k}. \tag{2.19}$$

Since the repairable probability of M_1–M_{M-1} is given by $Y(K-k, M-1)$, the repairable probability of the entire memory is expressed by the following recurrence formula:

$$\begin{aligned} Y(K,M) &= \sum_{k=0}^{\max(K,L)} P(k)Y(K-k,M-1) \\ &= \sum_{k=0}^{\max(K,L)} \binom{K}{k}\left(\frac{1}{M}\right)^{k}\left(1-\frac{1}{M}\right)^{K-k} Y(K-k,M-1). \end{aligned} \tag{2.20}$$

In the case of $K > R$, however, $Y(K, M) = 0$. The calculated repairable probability is shown by the broken lines in Fig. 2.33. It gradually decreases with the increase in the number of faults K.

The calculated yield using a more practical model is shown in Fig. 2.34 [5], comparing the simultaneous replacement and flexible replacement. The yield improvement factors through both the replacement techniques are almost the same in a 4-Mbit DRAM. The advantage of the flexible replacement becomes apparent in 64-Mbit and 1-Gbit DRAMs, especially for a large defect density, that is, in the early stages of production.

A disadvantage of the flexible replacement is the problem of a "global" defect, that is, a defect causing two or more faults. Let us consider the multidivided dataline structure (Fig. 2.24). A defect on a dataline is "local" and produces no effect on the other subarrays. However, if a defect exists on a sense-amplifier, two datalines connected to the amplifier fail simultaneously. A defect on a column selection line causes all the data lines connected to the line to fail simultaneously. Thus these types of defects are global. In the case of the hierarchical wordline structure (Figs. 2.25 and 2.26),

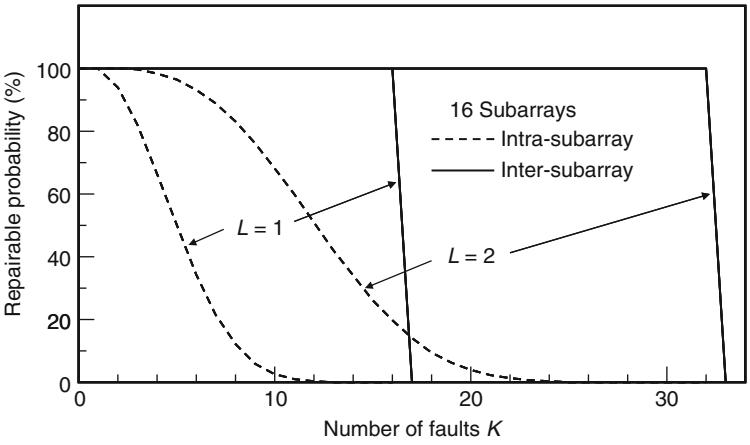


Fig. 2.33 Comparison between intra and intersubarray replacement. Reproduced from [20] with permission; © 2010 IEEE

a defect on a MWL is global. Since only one faulty normal line is replaced by a spare line in the flexible replacement, more than one address comparator is needed to repair the faults caused by a global defect. This may result in a deficiency of address comparators. Variable replacement unit using a ternary address comparator [5] can solve this problem. It features that not only ZERO and ONE, but also a "don't-care" value can be programmed. The don't-care value is assumed to coincide with both ZERO and ONE. Programming don't care in an address comparator specifies that the address bit is not compared with the input address. Using the don't care makes replacement unit (a group of memory cells replaced at a time) variable according to defect mode. For example, a sense-amplifier defect can be repaired by programming a don't care at the intersubarray address bit that specifies an upper/lower subarray of the sense amplifier ($a_{R_{m-2}}$ in Fig. 2.31), and by programming ZERO or ONE at the other address bits. This is because the intrasubarray address is common to the two faulty datalines. Table 2.4 shows the numbers of address comparators required to repair the various defects. A dataline or subwordline (local) defect requires an address comparator whether the replacement unit is fixed or variable. A global defect, however, requires two or more address comparators in the fixed replacement unit, while it requires only one in the variable replacement unit. Thus, the variable replacement unit reduces the number of address comparators required to repair the same number of defects and to achieve the same yield. A circuit diagram of the ternary address comparator is shown in Fig. 2.35. This circuit has two fuses, F_0 and F_1. Table 2.5 shows the programming method. ZERO or ONE is programmed by blowing one of the fuses as in Table 2.1. Don't care is programmed by remaining both fuses unblown, and the output c_i is always "1." Note the bottom two rows of the table. When both fuses are blown, the output c_i is always "0." This can be used to invalidate the address comparator when the corresponding spare line is found to be faulty. In this sense this circuit is "quaternary" rather than ternary.

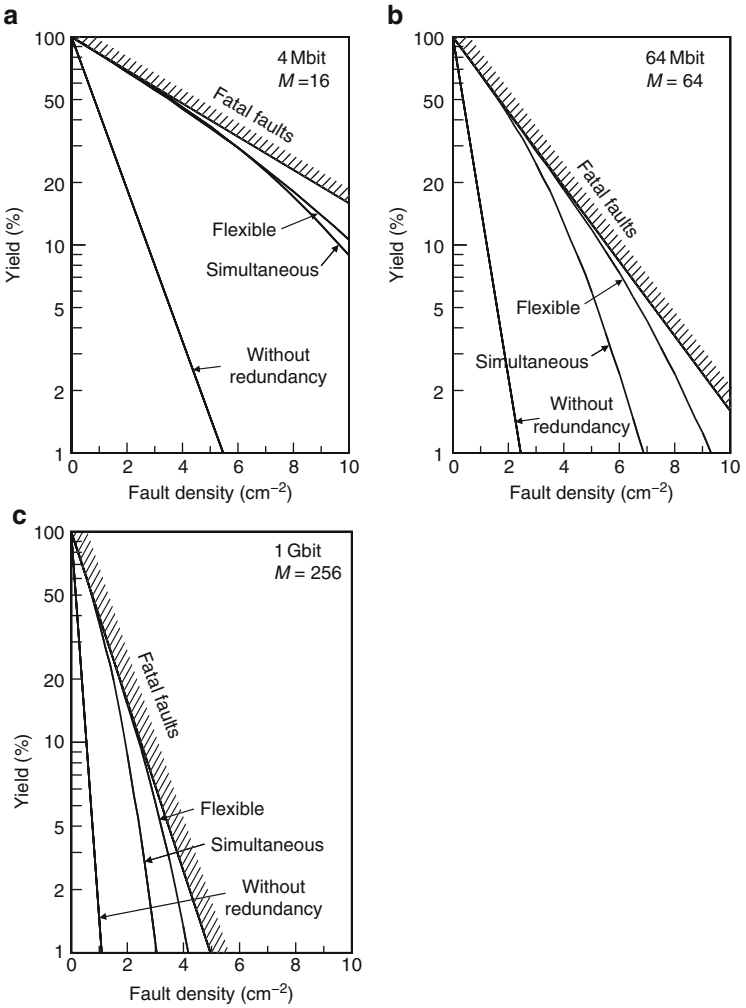


Fig. 2.34 Calculated yield using intrasubarray replacement redundancy techniques, simultaneous replacement ($L = R = 8$) and flexible replacement ($L = 4, R = 16$): (**a**) 4-Mbit DRAM (number of subarrays $M = 16$), (**b**) 64-Mbit DRAM ($M = 64$), and (**c**) 1-Gbit DRAM ($M = 256$). Reproduced from [5] with permission; © 2010 IEEE

Table 2.4 Number of address comparators required for repairing defects [5, 20]

Defect mode	No. of address comparators	
	Binary (fixed replacement unit)	Ternary (variable replacement unit)
Dataline	1	1
Sense amplifier	2	1
CSL	*M*	1
Sub wordline	1	1
Main wordline	*N*	1

M number of subarrays connected to a CSL, *N* number of sub wordlines connected to a main wordline

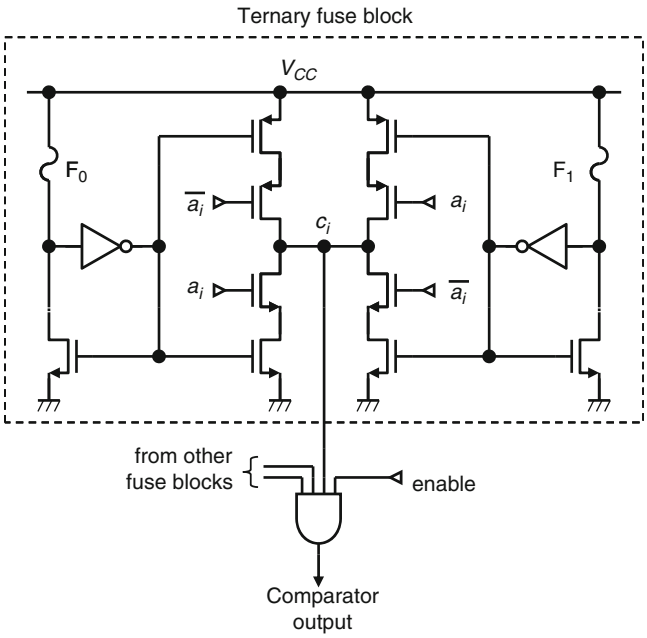


Fig. 2.35 Ternary address comparator for variable replacement unit. Reproduced from [5] with permission; © 2010 IEEE

Table 2.5 Truth table of ternary fuse block

Stored address	F_0	F_1	a_i	a_i	c_i
Zero	Blown	Unblown	0	1	1
			1	0	0
One	Unblown	Blown	0	1	0
			1	0	1
Don't care	Unblown	Unblown	0	1	1
			1	0	1
Invalid	Blown	Blown	0	1	0
			1	0	0

2.5.3 Variations of Intrasubarray Replacement

It is possible to apply the intersubarray replacement described in the next section to memories with array divisions for further enhancing the replacement efficiency. In some cases, however, the intersubarray replacement can never be applied.

One of the cases is bank division. Some memories, such as synchronous DRAMs (SDRAMs) have a plurality of memory banks. Since different banks may be simultaneously active, interbank replacement (replacing a faulty normal line in a bank by a spare line in another bank) is not allowed. Figure 2.36 shows a double-data-rate (DDR) SDRAM with flexible intrabank replacement redundancy and variable replacement unit [29]. Address comparators for row redundancy RAC_0–RAC_3 are shared by four banks. Each comparator can be used for any

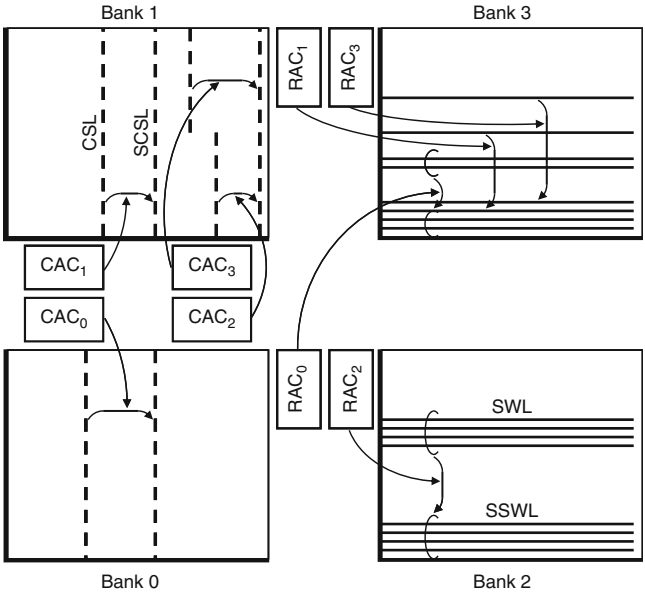


Fig. 2.36 Flexible intrasubarray replacement applied to a memory with bank division. Reproduced from [29] with permission; © 2010 IEEE

bank. In this case, RAC_0, RAC_1, and RAC_3 are used for replacing faulty wordlines in bank 3 by spare wordlines, and RAC_2 is used for bank 2. Similarly, address comparators for column redundancy CAC_0–CAC_3 are shared by two banks. Here, CAC_0 is used for bank 0 and CAC_1–CAC_3 are used for bank 1. In addition, the replacement unit is variable for both row and column redundancy. One, two, four, or eight SWLs can be replaced by spare wordlines at a time. A column-selection line (CSL) or a half of CSL can be replaced by a spare CSL.

Another case in which the intersubarray replacement is not allowed is multibit prefetch. Prefetching two or more bits simultaneously and outputting them serially are often used for increasing data-transfer rate. Figure 2.37 shows a DDR-SDRAM using 2-bit prefetch. Two subarrays store data of even and odd column addresses, respectively. The subarrays are simultaneously activated and two sets of data from them are serially outputted at the rising and falling edges of a clock signal. If the input column address is even (i.e., the lowest address bit $a_{C_0} = 0$), the activated column addresses in both subarrays are the same. However, if the input address is odd $(a_{C_0} = 1)$, they are different; two bits of column address a_{C_1} and a_{C_2} in MA(even) equal to that in MA (odd) plus one. In this case, the two bits of input address are incremented by one and are supplied to MA(even). A straightforward approach is providing two sets of address comparators (including fuses), one for even and the other for odd. However, this doubles the circuit area. An approach to reduce the area penalty, even-odd sharing

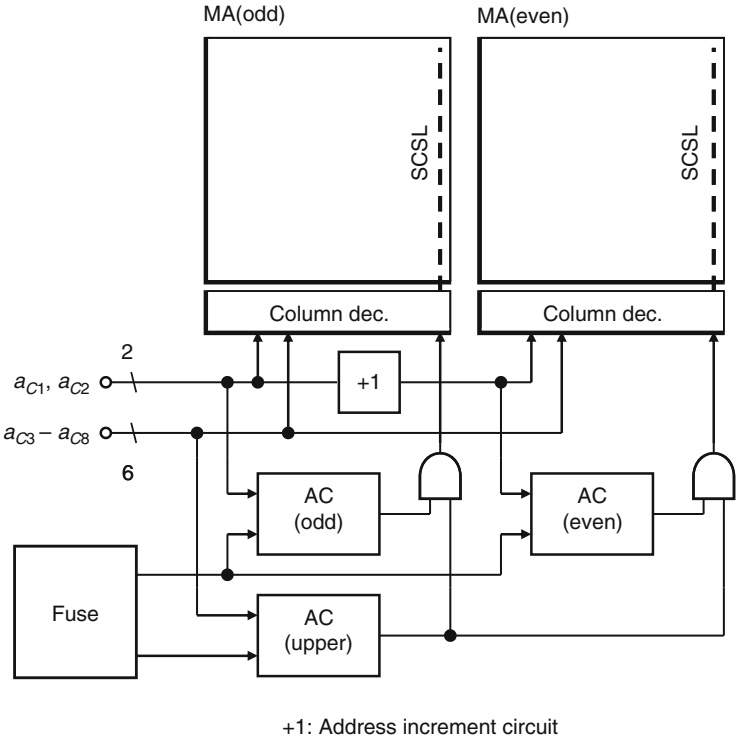


Fig. 2.37 Intrasubarray replacement applied to a memory using 2-bit prefetch [30]

scheme, is shown in Fig. 2.37 [30]. It features partial circuit sharing between the two subarrays. The lower-address (a_{C_1} and a_{C_2}) comparators for MA(even) and MA(odd) are separate, and the upper-address comparators and fuse sets are common. A spare column is activated by logical AND of the outputs of lower- and upper-address comparators. This scheme therefore halves the number of fuses required. In other words, if this approach has the same number of fuses as the straightforward approach, the number of spare columns can be doubled in order to improve the chip yield.

The access-time penalty due to redundancy is the delay time required for the address comparison as described in the previous section. Figure 2.38 shows a technique to eliminate this delay time for a high-speed SRAM [31]. In this technique, a faulty wordline in a subarray is replaced by a spare wordline in the adjacent subarray. The two subarrays are simultaneously activated and one of the data from them is selected according to the result of address comparison. This technique is difficult to be applied to row redundancy of general-purpose DRAMs because the dataline charging/discharging current is doubled. However, it is effective for DRAM column redundancy [32]. It is also reported that this technique was applied to the row redundancy of an ultrahigh-speed DRAM [33], in which power dissipation is not a major concern. Another disadvantage of this technique is that the number of spare lines is limited to one. In order to provide two spare lines, three

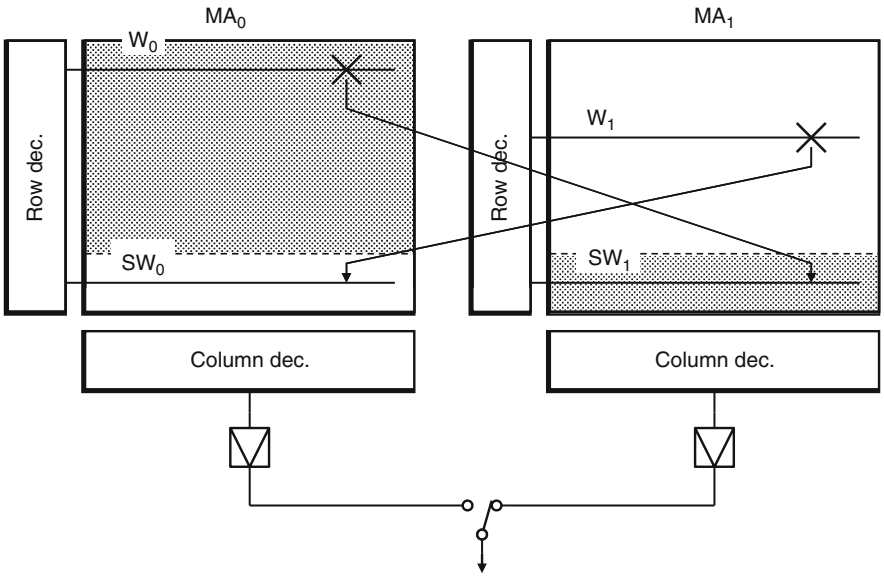


Fig. 2.38 No access-penalty intrasubarray replacement redundancy technique (simultaneous activation of normal and spare lines) [31, 33]

subarrays would have to be simultaneously activated. Note that this technique is not the intersubarray replacement which is described in the next section. This will be clear if the hatched areas in Fig. 2.38 are assumed to be a subarray and the nonhatched areas are assumed to be another subarray.

2.6 Intersubarray Replacement

With the further increase in memory-array division, the probability of clustered faults in a particular subarray becomes no more negligible. In the intrasubarray replacement, the number of spare lines in a subarray, L, must be larger or equal to the maximum number of faulty lines in each subarray to repair clustered faults. This causes the increase in L and chip-area penalty. To solve this problem, intersubarray replacement redundancy techniques [34–36] have been proposed, which permit a faulty normal line to be replaced by a spare line in any subarray. The replacement region is therefore an entire memory chip. They are classified into two categories.

In the distributed-spare-line approach [34] shown in Fig. 2.39, each subarray has its own spare lines like the intrasubarray replacement. Each spare line, however, can replace any faulty normal line not only in the same subarray but also in another subarray. Therefore at most LM faults clustered in a subarray can be repaired, where M is the number of subarrays. In this example, four clustered faulty normal wordlines W_0–W_3 are replaced by the spare wordlines in subarrays MA_0, MA_1, and MA_2. It is

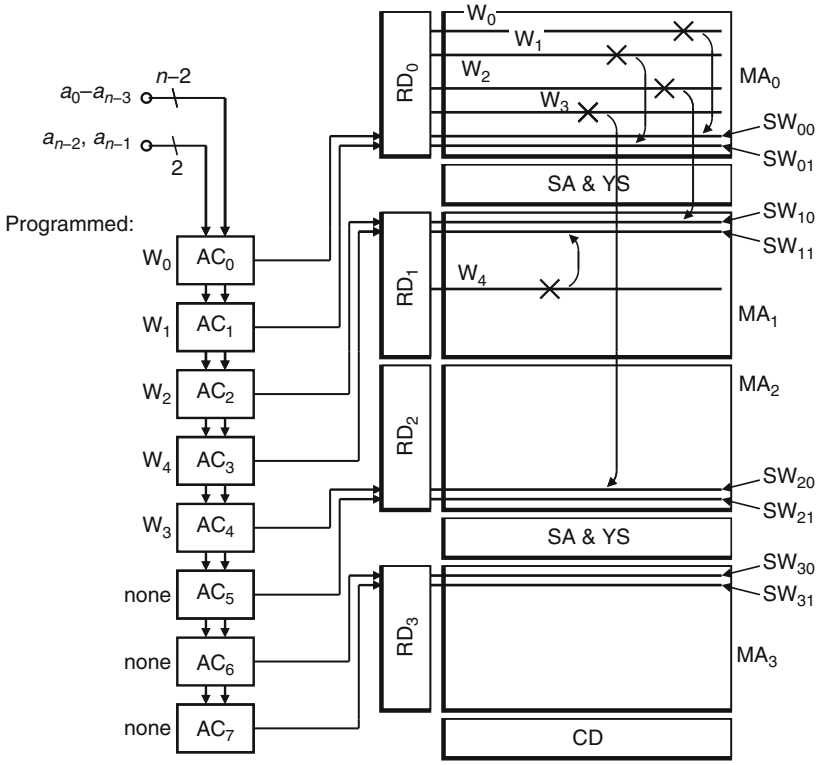


Fig. 2.39 Intersubarray replacement with distributed spare lines [34]

sufficient for successful repair that the number L is the average number of faulty lines in a subarray and is smaller than that of intrasubarray replacement. The number of address comparators R is equal to LM in this case. The number, however, can be reduced through the similar technique as the flexible replacement shown in Fig. 2.30.

In the concentrated-spare-line approach [35, 36] shown in Fig. 2.40, each subarray has no spare lines. There is a spare subarray MA_S, instead, composed of L' (here, $L' = 5$) spare lines. Each spare line can replace a faulty normal line in any subarray. Therefore, at most L' faults clustered in a subarray can be repaired. The number of address comparators R is equal to L'. This approach has an advantage of more flexible selection of $L'(= R)$ and better usage of address comparators compared to the distributed-spare-line approach. This is because the size of the spare subarray need not be the same as that of a normal subarray. The problem of this approach is that additional circuits (a decoder, a sense amplifier, etc.) for MA_S are needed. A solution of this problem using the hierarchical bitline architecture is proposed in [35].

Figure 2.33 compares the repairable probability using intra- and intersubarray replacement redundancy techniques [34, 36]. The probability for the former is calculated using (2.20). In the intrasubarray replacement, the repairable probability of a memory composed of M subarrays decreases with the increase in the number of

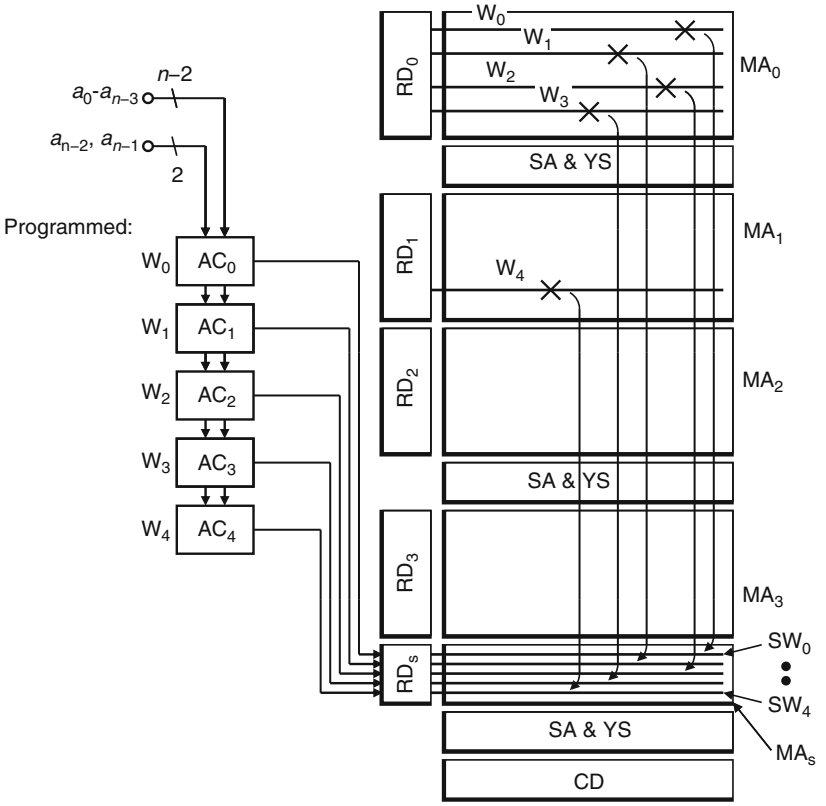


Fig. 2.40 Intersubarray replacement with concentrated spare lines [35, 36]

faults, K, because the probability of excessive ($>L$) faults in a particular subarray increases. On the other hand, the repairable probability is constantly 100% as long as $K \leq LM$ in the intersubarray replacement. When the number of subarrays is $M = 16$, the expectation of repairable faults with intersubarray replacement is about three times that with intrasubarray replacement.

The access-time penalty of the intersubarray replacement is usually larger than that of intrasubarray replacement. This is because not only an activated-line but also an activated subarray may be changed according to the result of address comparison.

2.7 Subarray Replacement

One of the problems with the increase in memory capacity is defects causing DC-characteristics faults, which mean the violation of DC specification of memories. In particular, excessive-standby-current (I_{SB}) is the most popular DC-characteristics

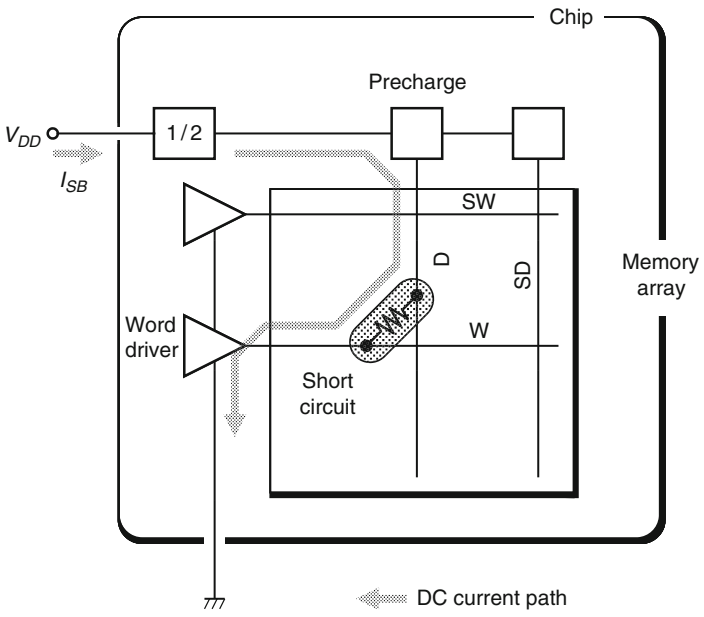


Fig. 2.41 Model of standby-current fault of a DRAM [37]

fault. An example of a defect that causes an I_{SB} fault of a DRAM is shown in Fig. 2.41 [37]. A short circuit between a wordline (electrically connected to the ground during standby state) and a dataline (connected to the dataline precharge voltage, $V_{DD}/2$) creates an illegal DC current path from $V_{DD}/2$ to ground. Replacing the wordline and dataline by a spare wordline and a spare dataline, respectively, inhibits the faulty lines from being accessed, but the current path still remains. Thus, the fault is not repaired by line replacement.

Several techniques have been reported to repair such faults. They include limiting the illegal current through the short circuit to a small value by a current limiter [36], cutting off the current path by a power switch controlled by a fuse [38]. Figure 2.42 shows another technique [37] where the replacement unit is a subarray. A subarray, including an I_{SB} fault, is replaced by an on-chip spare subarray. Each subarray has power switches for bitline precharge voltage $V_{DD}/2$ and memory-cell plate voltage V_{PL}, logic gates for timing signals, and a fuse to control them. The power switches of the faulty subarray are turned off and those of the spare subarray are turned on. Thus the I_{SB} fault is repaired by cutting of the DC current. The logic gates of the faulty subarray are also turned off to eliminate unnecessary power dissipation in the subarray. It is reported that this technique combined with the line replacement doubles the yield of a 256-Mbit DRAM [36]. One advantage of this technique is that the wordlines in an unused spare subarray can be used as spare wordlines of the concentrated-spare-line intersubarray replacement redundancy described in Sect. 2.6 [39].

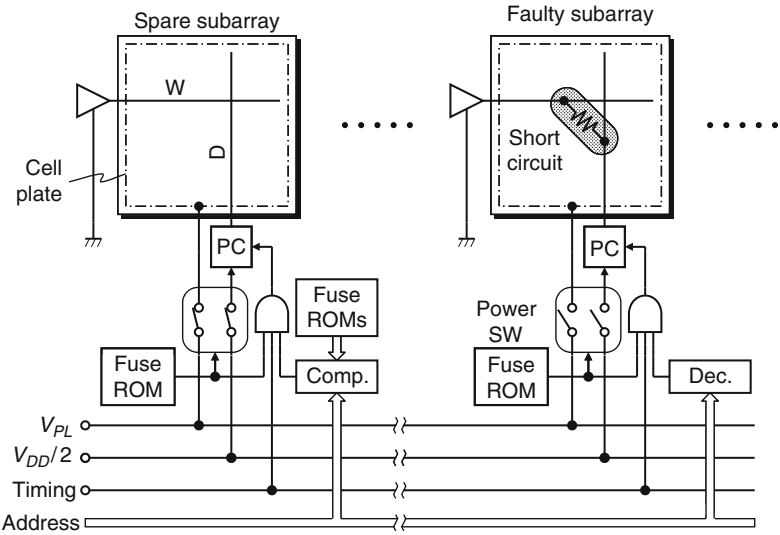


Fig. 2.42 Subarray-replacement redundancy technique [37]

Since this redundancy technique requires spare subarrays, it is not suitable for a small-capacity memory with a small number of subarrays. However, the number of subarrays rapidly increases with every generation as shown in Fig. 2.23. Therefore, the area penalty is allowable for DRAMs of 256 Mbit and beyond. It is interesting that the memory-array division, which was the barrier to the line-replacement redundancy, in turn, supports the subarray-replacement redundancy.

2.8 Devices for Storing Addresses

Redundancy techniques require devices for storing faulty addresses. These devices must be programmable as well as nonvolatile to retain the replacement information during power-off. The devices include fuses, antifuses, and nonvolatile memory cells. A fuse is initially in a conductive (low-resistance) state and reaches a nonconductive (high-resistance) state by programming. On the contrary, an antifuse is initially in a high-resistance state and reaches a low-resistance state. On the other hand, a nonvolatile memory cell exploits the change of its threshold voltage.

2.8.1 Fuses

The materials used for fuses include polysilicon, silicides ($MoSi_2$, $TaSi_2$), and metals (Cu, etc.). A fuse is blown by either electric current or laser and each has advantages and disadvantages.

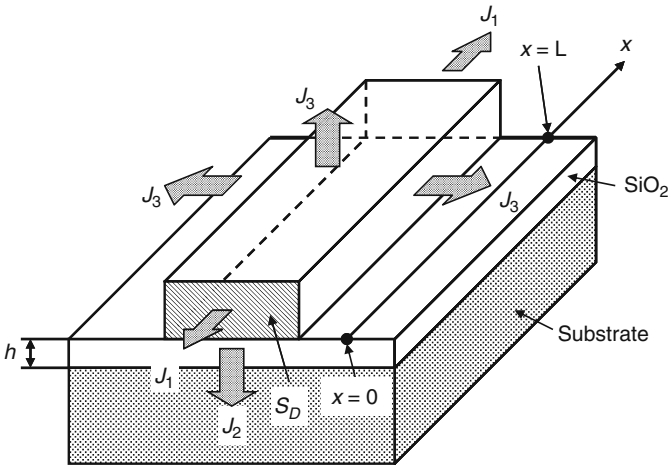


Fig. 2.43 Model of an electrical fuse and a coordinate used for one-dimensional thermal equation. Reproduced from [42] with permission; © 2010 JSAP

Electrical blowing requires no special equipment. In addition, faulty addresses can be programmed even after packaging (postpackaging programming) [40, 41]. However, it requires an on-chip programming circuit shown in Fig. 2.18, including a large transistor M_0 for driving a large blowing current. A special high-voltage V_{PP} is supplied from an extra pad probed at wafer test [15]. This voltage enhances the conductance of M_0 to supply a sufficient current. Whether the fuse is blown or not is determined by the address signal a_i according to the left half of Table 2.1. After programming, the V_{PP} pad is grounded by an on-chip pull-down circuit (not shown) to prevent inadvertent programming. A model of the electrical blowing of a polysilicon fuse is shown in Fig. 2.43 [42]. The Joule heat generated by the current and the fuse resistance diffuses along the fuse to the metal-polysilicon contacts at $x = 0$ and $x = L$ (J_1), to the substrate through SiO_2 (J_2), and to the surrounding glass layer (J_3). The temperature increase at the contacts and the substrate is assumed to be zero because the thermal conductivities of metal and Si are far larger than those of SiO_2 and glass. The fuse is heated up to the melting point (1,420°C) and is finally blown off. The temperature increase $T(x)$ at position x after a sufficiently long time (i.e., thermal equilibrium) is expressed as:

$$T(x) = T_0\left(1 - \frac{\sinh\frac{x}{L_0} + \sinh\frac{L-x}{L_0}}{\sinh\frac{L}{L_0}}\right), \tag{2.21}$$

where L is the fuse length, L_0 is the characteristic length determined by the thermal conductances of the polysilicon, SiO_2, and glass layer, and T_0 is the equilibrium temperature without J_1 [42]. The calculated temperature-increase distribution along the fuse is shown in Fig. 2.44. The fuse length should be designed to be at least several times of L_0 (here, $L_0 = 3.55$ μm).

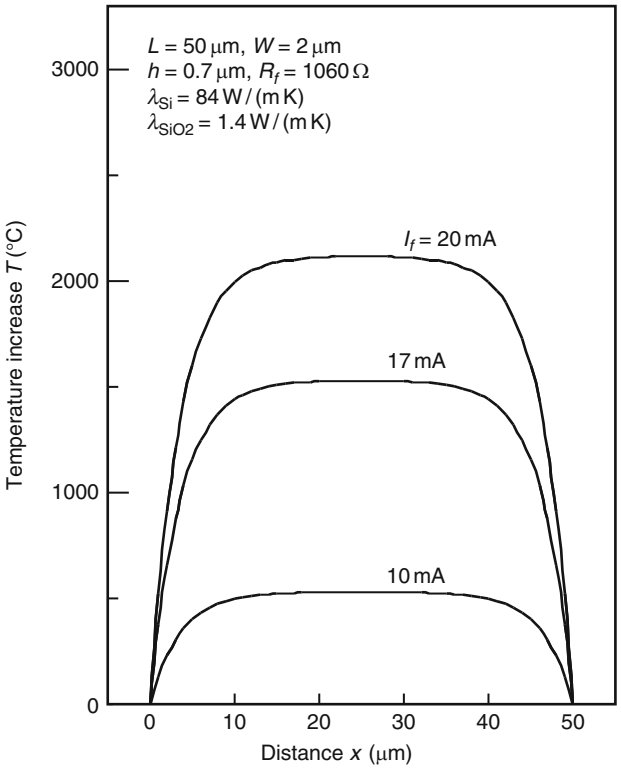


Fig. 2.44 Temperature distribution along an electrical fuse. Reproduced from [42] with permission; © 2010 JSAP

Blowing a fuse by laser requires a laser equipment. However, the area penalty is smaller because the on-chip programming circuit is not required. The area for fuses is determined by the diameter of laser spot and its alignment accuracy. It is therefore suitable for high-density memories with a large number of spare lines. However, the postpackaging programming is impossible.

Since the fuse and the surrounding layers are broken by blowing, contamination from the wreck is a major concern for both electrical blowing and laser blowing. Device structures for preventing the contamination are reported in [42, 43].

2.8.2 Antifuses

Various kind of antifuses have been proposed, including electrically destroying a p–n junction of a polysilicon diode [44], diffusing impurities into an intrinsic polysilicon by laser pulses [45], and electrically breaking down a dielectric [46]. Figure 2.45 shows the cross section of the antifuse for DRAM redundancy. It utilizes an oxide–nitride–oxide (ONO) film as a dielectric. This structure is compatible with the

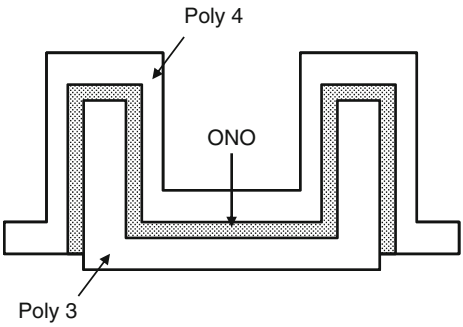


Fig. 2.45 Cross section of an antifuse using oxide–nitride–oxide (ONO) film for DRAM capacitor [46]

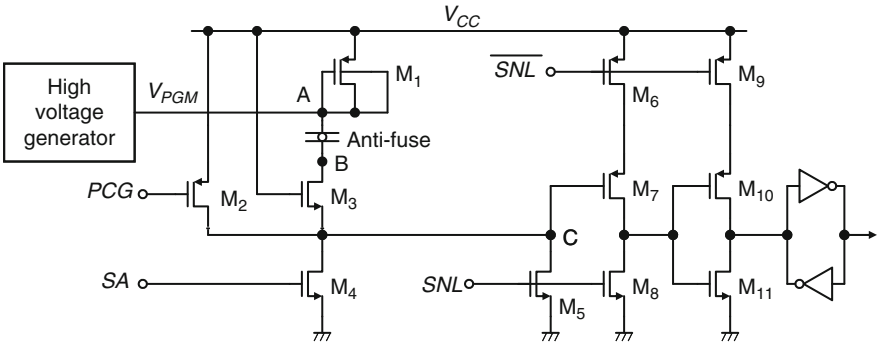


Fig. 2.46 Antifuse programming and sensing circuit. Reproduced from [46] with permission; © 2010 IEEE

memory cell of the DRAM and no additional process is required. Initially, the ONO film has an extremely high resistance (>100 MΩ). By applying a high-programming voltage (≥8 V), the film destructively breaks down and produces a short circuit between the polysilicon layers with a significantly lower resistance (<10 kΩ). The programming and sensing circuit of the antifuse is shown in Fig. 2.46. The programming procedure is as follows. First, signal PCG is low and node B is precharged to $V_{CC} - V_t$ (V_t is the threshold voltage of M_3) through M_2 and M_3 to prevent unselected antifuses from being programmed. The programming voltage V_{PGM} is generated by an on-chip high-voltage generator and is applied to all antifuses. Next, signal SA of selected antifuses is set high to discharge node B through M_3 and M_4 and to apply the full V_{PGM}, while the voltage across the unselected antifuses is limited. Note that M_3 protects M_5 and M_7 from the high voltage. In order to sense the state of the antifuse, signals SNL and $\overline{SNL}$ are set high and low, respectively. The voltage of node C is $V_{CC} - V_t$ if the antifuse is programmed, or zero if not. This analog voltage is sensed and latched by the subsequent circuit.

This technique features that the postpackaging programming is possible [46, 47] like the electrical fuses. Although the programming circuit is required, its area

penalty is smaller than that of electrically blowing fuses. This is because the high-voltage generator is common to all the antifuses, while as many numbers of programming transistors as fuses are required. Therefore, this technique is promising for high-density memories with a large number of spares.

2.8.3 *Nonvolatile Memory Cells*

One of the programmable devices used for the redundancy of nonvolatile memories is the nonvolatile memory cell itself. Figure 2.47 shows an address comparator (corresponding to the address comparator in Fig. 2.17) used for row redundancy of an EEPROM [48]. The address programming circuit corresponds to the fuse block $0 - n-1$, while the spare-row enabling circuit corresponds to the fuse block n. Transistors M_1 and M_4 are the floating gate transistors with tunnel oxide areas at nodes A and B, respectively. These devices are erased ($V_t > 6$ V) in advance. M_4 is erased by setting $\bar{a}_i$ and S at low and high level, respectively, and taking the E (erase) line to the programming voltage (20 V). M_1 is erased by setting S at low level. Since the gate of M_2 is at high level, the output signal *hit* is low and the spare row is never enabled. To enable the spare row and to program a faulty address, the address is set at a_i and $\bar{a}_i$, while E is held at 0 V and S is pulled up to the programming voltage. Thus, if $a_i = 0$ ($\bar{a}_i = 1$), transistor M_4 is programmed into the "on" state ($V_t < -2$ V) turning M_6 on. If $a_i = 1$ ($\bar{a}_i = 0$), M_4 is not programmed and is left in the high V_t state turning M_5 on. Concurrently, M_1 is also programmed into a low V_t state and M_2 is turned off. If the input address coincides to the programmed address, all the

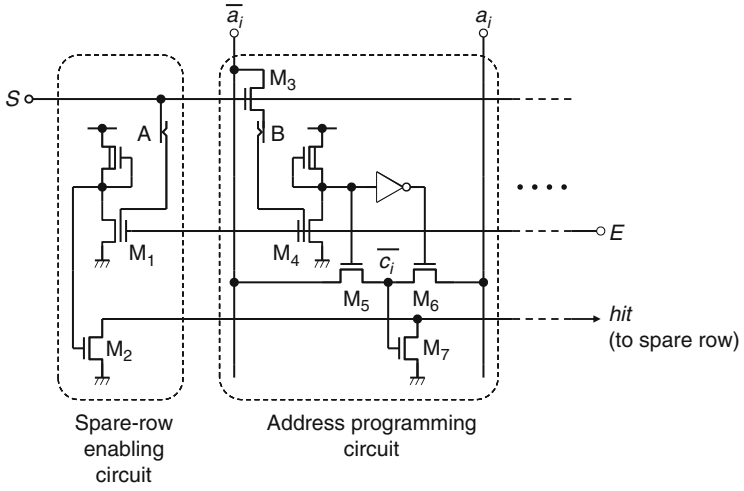


Fig. 2.47 Address comparator using EEPROM cells as programmable elements. Reproduced from [48] with permission; © 2010 IEEE

Table 2.6 Truth table of the address comparator using EEPROM cells

Erasing (E = high)			Programming (E = low)			Normal operation		
a_i	$\bar{a}_i$	M4	a_i	$\bar{a}_i$	M4	a_i	$\bar{a}_i$	$\bar{c}_i$
1	0	Erased (high V_t)	0	1	Programmed (low V_t)	0	1	0
						1	0	1
			1	0	Not programmed (high V_t)	0	1	1
						1	0	0

pull-down transistors (M_2, M_7) are turned off and the output signal *hit* is pulled up to a high level by a pull-up circuit (not shown), thus enabling the spare row. The operations of the circuit are summarized in Table 2.6. This technique features that postpackaging programming is possible like the antifuses. A nonvolatile memory cell fabricated with standard CMOS process and its application to memory redundancy are also reported [49, 50].

2.9 Testing for Redundancy

Testing for memories with redundancy includes the following three categories:

1. Normal-element test: testing normal elements and determining which normal elements should be replaced by which spare elements. This test is essential for redundancy techniques.
2. Spare-element test: checking whether each spare element that replaces a faulty normal element is faulty or not in advance. This test is optional but is effective for reducing the probability of unsuccessful repair caused by faulty spare elements.
3. Postrepair diagnostics: checking after packaging whether a memory LSI is a perfect chip or a repaired chip (signature), and if the latter, which addresses are repaired (roll-call). This test enables chip diagnostics by the manufacturer.

The normal-element test is usually performed by a memory tester. A memory tester has a fail-bit memory (FBM) for recording the pass/fail of each bit of the memory under test. After scanning an entire memory, a replacing solution is determined by analyzing the failure pattern in the FBM. Faulty rows must be replaced by spare rows and faulty columns must be replaced by spare columns. On the other hand, faulty bits can be replaced by either spare rows or spare columns. Therefore, a decision algorithm for finding a replacement solution is needed. The algorithm should meet the following requirements. First, the probability of overlooking (being unable to find a replacement solution though it does exist) should be minimized to maximize yield. Second, the decision time should be minimized in order not to occupy an expensive memory tester for a long time. In particular, a nonrepairable chip should be aborted as early as possible. Some repair algorithms to make the decisions are proposed to find an optimal replacement solution [51–53]. Figure 2.48 shows an algorithm using row/column fault counters. After testing an

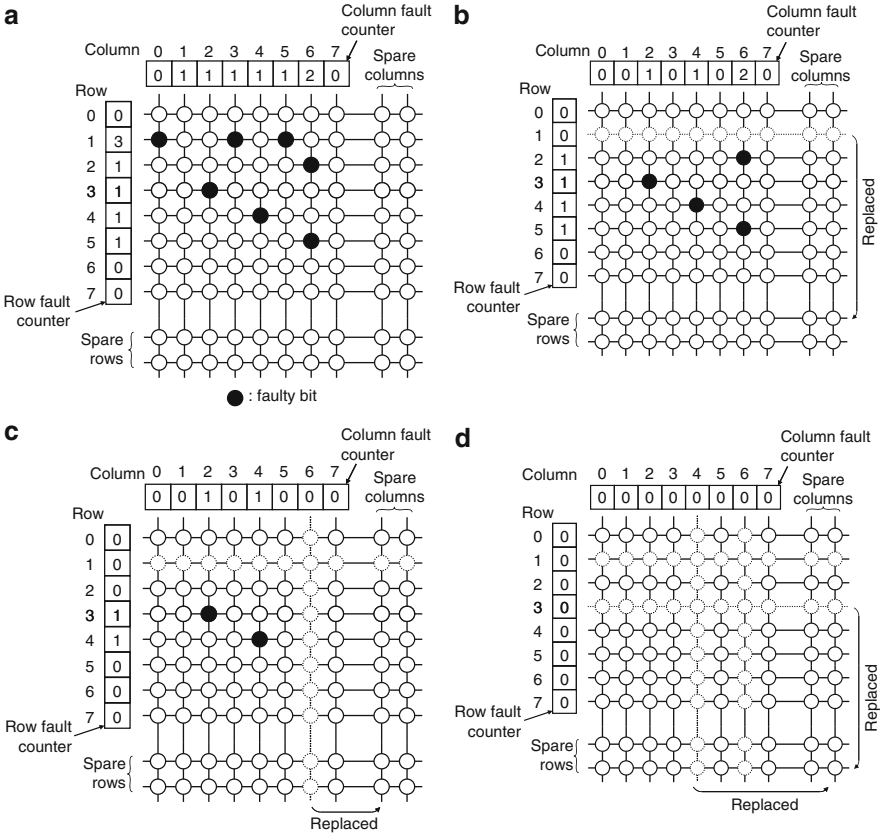


Fig. 2.48 Replacement algorithm: (**a**) just after testing an entire memory, (**b**) row one is replaced by a spare row, (**c**) column six is replaced by a spare column, and (**d**) row three and column four are replaced by spares

entire memory, the number of faulty bits on each row/column is recorded in each row/column fault counter (Fig. 2.48a). The algorithm consists of three steps.

Step 1: If the total number of faulty bits exceeds $N_X R_Y + N_Y R_X - R_X R_Y$, the chip cannot be repaired and is aborted, where N_X, N_Y, R_X, and R_Y are the number of normal rows, normal columns, spare rows, and spare columns, respectively. Since the example of Fig. 2.48a does not meet this condition, we go to the next step.

Step 2: A row having more faulty bits than the number of remaining spare columns must be replaced by a spare row, and a column having more faulty bits than the number of remaining spare rows must be replaced by a spare column. This step is repeated until no rows/columns meet the condition. In this example, row 1 has three faults and replaced by a spare row. The number of remaining spare rows becomes 1 and the contents of the counters are revised (Fig. 2.48b).

Step 3: A row or a column with maximum number of faulty bits is replaced by a spare row or a spare column. This step is repeated until all faulty bits are repaired (success) or spares are exhausted (failure). In this example, column 6 is first replaced by a spare column because it has two faults (Fig. 2.48c). Next row 3 and column 4 (or row 4 and column 2) are replaced by spares (Fig. 2.48d).

This algorithm is sufficiently good for usual faulty-bit distribution patterns though it cannot find a solution for some patterns [52]. Build-in self-test (BIST) and build-in self-repair (BISR) techniques utilizing on-chip circuitry instead of testers as well as repair algorithms suitable for them are also reported for test-cost reduction [54, 55].

Figure 2.49 shows a circuit for the spare-element test [56]. Before blowing fuses, each signal $\overline{hit_i}$ (the output of each address comparator) is high. The test circuit is enabled by setting the test pad at high level. During the test mode, any spare row can be selected by the row address signals a_0–a_2 and the memory cells connected to the selected spare row can be written and read. Thus, this circuit allows testing the spare rows without fuse programming. After the test, fuses are blown if necessary. During normal operation (test pad is low), each spare row is controlled by the signal $\overline{hit_i}$.

Figure 2.50a shows a DRAM with the postrepair diagnostics [57], which is realized by adding the extra circuitry within the dashed line. An additional fuse is blown only in the case of a repaired chip to enable the comparator and the roll-call decoder. The operating waveforms of row-redundancy diagnostics are shown in Fig. 2.50b. They are the same as those of "$\overline{\text{RAS}}$-only refresh" except that a super high-level voltage (8–10 V) is applied to the data input terminal D_{IN}. A row address A_R is specified at the falling edge of $\overline{\text{RAS}}$. The result of the test is available at the data output terminal D_{OUT}. If the device under test is a repaired chip, D_{OUT} goes high or low according to A_R being one of the repaired addresses or not. On the other hand, D_{OUT} remains high impedance in the case of a perfect chip. Column redundancy diagnostics are performed with waveforms like "early write" operation.

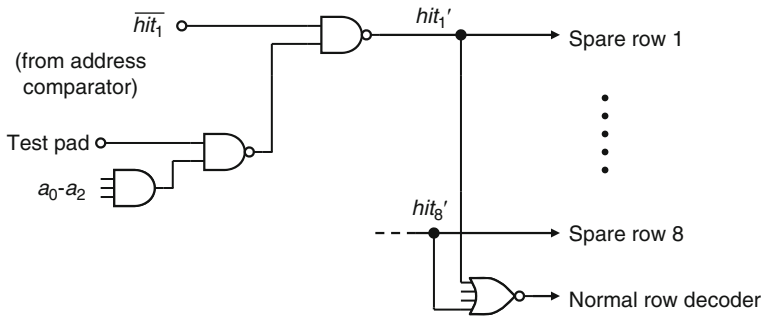


Fig. 2.49 Circuit for testing spare elements. Reproduced from [56] with permission; © 2010 IEEE

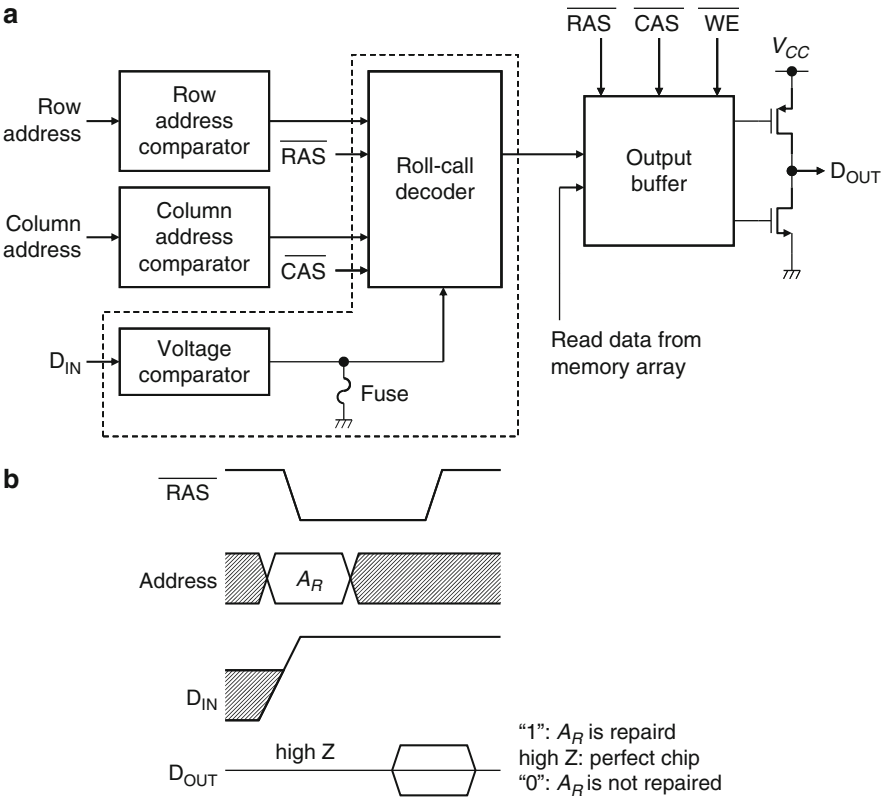


Fig. 2.50 Redundancy diagnostics circuit: (**a**) circuit diagram and (**b**) operating waveforms of row-redundancy diagnostics. Reproduced from [57] with permission; © 2010 IEEE

References

1. S. E. Schuster, "Multiple word/bit line redundancy for semiconductor memories," IEEE J. Solid-State Circuits, vol. SC-13, pp. 698–703, Oct. 1978.
2. T. Mano, M. Wada, N. Ieda and M. Tanimoto, "A redundancy circuit for a fault-tolerant 256K MOS RAM," IEEE J. Solid-State Circuits, vol. SC-17, pp. 726–731, Aug. 1982.
3. S. Fujii, K. Natori, T. Furuyama, S. Saito, H. Toda, T. Tanaka and O. Ozawa, "A low-power sub 100 ns 256K bit dynamic RAM," IEEE J. Solid-State Circuits, vol. SC-18, pp. 441–446, Oct. 1983.
4. Y. Nishimura, M. Hamada, H. Hidaka, H. Ozaki and K. Fujishima, "A redundancy test-time reduction technique in 1-Mbit DRAM with a multibit test mode," IEEE J. Solid-State Circuits, vol. 24, pp. 43–49, Feb. 1989.
5. M. Horiguchi, J. Etoh, M. Aoki, K. Itoh and T. Matsumoto, "A flexible redundancy technique for high-density DRAMs," IEEE J. Solid-State Circuits, vol. 26, pp. 12–17, Jan. 1991.
6. C. H. Stapper, Jr., "On a composite model to the IC yield problem," IEEE J. Solid-State Circuits, vol. SC-10, pp. 537–539, Dec. 1975.

7. C. H. Stapper, A. N. McLaren and M. Dreckmann, "Yield model for productivity optimization of VLSI memory chips with redundancy and partially good product," IBM J. Res. Dev., vol. 24, pp. 398–409, May 1980.
8. T. Okabe, M. Nagata and S. Shimada, "Analysis on yield of integrated circuits and a new representation for the yield," Trans. IEE J., vol, 92-C, pp. 399–406, Dec. 1972 (in Japanese).
9. C. H. Stapper, "Yield model for fault clusters within integrated circuits," IBM J. Res. Dev., vol. 28, pp. 636–640, Sep. 1984.
10. S. Kikuda, H. Miyamoto, S. Mori, M. Niiro and M. Yamada, "Optimized redundancy selection based on failure-related yield model for 64-Mb DRAM and beyond," IEEE J. Solid-State Circuits, vol. 26, pp. 1550–1555, Nov. 1991.
11. T. Yamagata, H. Sato, K. Fujita, Y. Nishimura and K. Anami, "A distributed globally replaceable redundancy scheme for sub-half-micron ULSI memories and beyond," IEEE J. Solid-State Circuits, vol. 31, pp. 195–201, Feb. 1996.
12. K. Imamiya, J. Miyamoto, N. Ohtsuka, N. Tomita and Y. Iyama, "Statistical memory yield analysis and redundancy design considering fabrication line improvement," IEICE Trans. Electron., vol. E76-C, pp. 1626–1631, Nov. 1993.
13. R. P. Cenker, D. G. Clemons, W. R. Huber, J. B. Petrizzi, F. J. Procyk and G. M. Trout, "A fault-tolerant 64K dynamic random-access memory," IEEE Trans. Electron Devices, vol. ED-26, pp. 853–860, June 1979.
14. E. A. Reese, D. W. Spaderna, S. T. Flannagan and F. Tsang, "A 4K $\times$ 8 dynamic RAM with self-refresh," IEEE J. Solid-State Circuits, vol. SC-16, pp. 479–487, Oct. 1981.
15. K. Kokkonen, P. O. Sharp, R. Albers, J. P. Dishaw, F. Louie and R. J. Smith, "Redundancy techniques for fast static RAMs," in ISSCC Dig. Tech. Papers, Feb. 1981, pp. 80–81.
16. A. Ohba, S. Ohbayashi, T. Shiomi, S. Takano, K. Anami, H. Honda, Y. Ishigaki, M. Hatanaka, S. Nagao and S. Kayano, "A 7-ns 1-Mb BiCMOS ECL SRAM with shift redundancy," IEEE J. Solid-State Circuits, vol. 26, pp. 507–512, Apr. 1991.
17. H. Noda, K. Inoue, M. Kuroiwa, A. Amo, A. Hachisuka, H. J. Mattausch, T. Koide, S. Soeda, K. Dosaka and K. Arimoto, "A 143MHz 1.1W 4.5Mb dynamic TCAM with hierarchical searching and shift redundancy architecture," ISSCC Dig. Tech. Papers, Feb. 2004, pp. 208–209.
18. A. Roth, D. Foss, R. McKenzie and D. Perry, "Advanced ternary CAM circuits on 0.13 μm logic process technology," in Proc. CICC, Oct. 2004, pp. 465–468.
19. T. Namekawa, S. Miyano, R. Fukuda, R. Haga, O. Wada, H. Banba, S. Takeda, K. Suda, K. Mimoto, S. Yamaguchi, T. Ohkubo, H. Takato and K. Numata, "Dynamically shift-switched dataline redundancy suitable for DRAM macro with wide data bus," IEEE J. Solid-State Circuits, vol. 35, pp. 705–712, May 2000.
20. M. Horiguchi, "Redundancy techniques for high-density DRAMs," in Proc. Int. Conf. on Innovative Systems Silicon, Oct. 1997, pp. 22–29.
21. R. Hori, K. Itoh, J. Etoh, S. Asai, N. Hashimoto, K. Yagi and H. Sunami, "An experimental 1 Mbit DRAM based on high S/N design," IEEE J. Solid-State Circuits, vol. SC-19, pp. 634–640, Oct. 1984.
22. K. Itoh, "Trends in megabit DRAM circuit design," IEEE J. Solid-State Circuits, vol. 25, pp.778–789, June 1990.
23. K. Itoh, *VLSI Memory Design*, Baifukan, Tokyo, 1994 (in Japanese), Chapter 2.
24. K. Itoh, *VLSI Memory Chip Design*, Springer, NY, 2001, Chapter 3.
25. M. Yoshimoto, K. Anami, H. Shinohara, T. Yoshihara, H. Takagi, S. Nagao, S. Kayano and T. Nakano, "A divided word-line structure in the static RAM and its application to a 64k full CMOS RAM," IEEE J. Solid-State Circuits, vol. SC-18, pp. 479–485, Oct. 1983.
26. K. Noda, T. Saeki, A. Tsujimoto, T. Murotani and K. Koyama, "A boosted dual word-line decoding scheme for 256Mb DRAMs," in Symp. VLSI Circuits Dig. Tech. Papers, June 1992, pp. 112–113.
27. D. Galbi, K. Althoff, R. Parent, O. Kiehl, R. Houghton, F. Bonner, M. Killian, A. Wilson, K. Lau, M. Clinton, D. Chapman and H. Fischer, "A 33-ns 64-Mbit DRAM with master-wordline architecture," in Proc. ESSCIRC, Sep. 1992, pp. 131–134.

28. K. Furutani, T. Hamamoto, T. Miki, M. Nakano, T. Kono, S. Kikuda, Y. Konishi and T. Yoshihara, "Highly flexible row and column redundancy and cycle time adaptive read data path for double data rate synchronous memories," IEICE Trans. Electron., vol. E88-C, pp. 255–263, Feb. 2005.
29. Y. Takai, M. Fujita, K. Nagata, S. Isa, S. Nakazawa, A. Hirobe, H. Ohkubo, M. Sakao, S. Horiba, T. Fukase, Y. Takaishi, M. Matsuo, M. Komuro, T. Uchida, T. Sakoh, K. Saino, S. Uchiyama, Y. Takada, J. Sekine, N. Nakanishi, T. Oikawa, M. Igeta, H. Tanabe, H. Miyamoto, T. Hashimoto, H. Yamaguchi, K. Koyama, Y. Kobayashi and T. Okuda, "A 250-Mb/s/pin, 1-Gb double-data-rate SDRAM with a bidirectional delay and an interbank shared redundancy scheme," IEEE J. Solid-State Circuits, vol. 35, pp. 149–162, Feb. 2000.
30. H. Yahata, Y. Okuda, H. Miyashita, H. Chigasaki, B. Taruishi, T. Akiba, Y. Kawase, T. Tachibana, S. Ueda, S. Aoyama, A. Tsukimori, K. Shibata, M. Horiguchi, Y. Saiki and Y. Nakagome, "A 256-Mb double-data-rate SDRAM with a 10-mW analog DLL circuit," in Symp. VLSI Circuits Dig. Tech. Papers, June 2000, pp. 74–75.
31. K. Sasaki, K. Ishibashi, T. Yamanaka, N. Hashimoto, T. Nishida, K. Shimohigashi, S. Hanamura and S. Honjo, "A 9-ns 1-Mbit CMOS SRAM," IEEE J. Solid-State Circuits, vol. 24, pp. 1219–1225, Oct. 1989.
32. H. Yamauchi, T. Suzuki, A. Sawada, T. Iwata, T. Tsuji, M. Agata, T. Taniguchi, Y. Odake, K. Sawada, T. Ohnishi, M. Fukumoto, T. Fujita and M. Inoue, "A circuit technology for high-speed battery-operated 16-Mb CMOS DRAM's," IEEE J. Solid-State Circuits, vol. 28, pp. 1084–1091, Nov. 1993.
33. Y. Yokoyama, N. Itoh, M. Katayama, M. Hasegawa, K. Takashima, H. Akasaki, M. Kaneda, T. Ueda, Y. Tanaka, E. Yamasaki, M. Todokoro, K. Toriyama, H. Miki, M. Yagyu, T. Kobayashi, S. Miyaoka and N. Tamba, "A 1.8-V embedded 18-Mb DRAM macro with a 9-ns RAS access time and memory-cell area efficiency of 33%," IEEE J. Solid-State Circuits, vol. 36, pp. 503–509, Mar. 2001.
34. K. Ishibashi, K. Komiyaji, S. Morita, T. Aoto, S. Ikeda, K. Asayama, A. Koike, T. Yamanaka, N. Hashimoto, H. Iida, F. Kojima, K. Motohashi and K. Sasaki, "A 12.5-ns 16-Mb CMOS SRAM with common-centroid-geometry-layout sense amplifiers," IEEE J. Solid-State Circuits, vol. 29, pp. 411–418, Apr. 1994.
35. M. Asakura, T. Oishi, S. Tomishima, H. Hidaka, K. Arimoto and K. Fujishima, "A hierarchical bit-line architecture with flexible redundancy and block compare test for 256Mb DRAM," in Symp. VLSI Circuits Dig. Tech. Papers, May 1993, pp. 93–94.
36. T. Kirihata, Y. Watanabe, H. Wong, J. K. DeBrosse, M. Yoshida, D. Katoh, S. Fujii, M. R. Wordeman, P. Poechmueller, S. A. Parke and Y. Asao, "Fault-tolerant designs for 256 Mb DRAM," IEEE J. Solid-State Circuits, vol. 31, pp. 558–566, Apr. 1996.
37. G. Kitsukawa, M. Horiguchi, Y. Kawajiri, T. Kawahara, T. Akiba, Y. Kawase, T. Tachibana, T. Sakai, M. Aoki, S. Shukuri, K. Sagara, R. Nagai, Y. Ohji, N. Hasegawa, N. Yokoyama, T. Kisu, H. Yamashita, T. Kure and T. Nishida, "256-Mb DRAM circuit technologies for file applications," IEEE J. Solid-State Circuits, vol. 28, pp. 1105–1113, Nov. 1993.
38. K. Furutani, T. Ooishi, M. Asakura, H. Hidaka, H. Ozaki and M. Yamada, "A board level parallel test circuit and a short circuit failure repair circuit for high-density, low-power DRAMs," IEICE Trans. Electron., vol. E80-C, pp. 582–589, Apr. 1997.
39. M. Asakura, T. Ohishi, M. Tsukude, S. Tomishima, H. Hidaka, K. Arimoto, K. Fujishima, T. Eimori, Y. Ohno, T. Nishimura, M. Yasunaga, T. Kondoh, S. Satoh, T. Yoshihara and K. Demizu, "A 34ns 256Mb DRAM with boosted sense-ground scheme," in ISSCC Dig. Tech. Papers, Feb. 1994, pp. 140–141.
40. K. Lim, S. Kang, J. Choi, J. Joo, Y. Lee, J. Lee S. Cho and B. Ryu, "Bit line coupling scheme and electrical fuse circuit for repairable operation of high density DRAM," Symp. VLSI Circuits Dig. Tech. Papers, June 2001, pp. 33–34.
41. K. H. Kyung, C. W. Kim, J. Y. Lee, J. H. Kook, S. M. Seo, D. Y. Kim, J. H. Kim, J. Sunwoo, H. C. Lee, C. S. Kim, B. H. Jeong, Y. S. Sohn, S. P. Hong, J. H. Lee, J. H. Yoo and S. I. Cho, "A 800Mb/s/pin 2Gb DDR2 SDRAM with an 80nm triple metal technology," ISSCC Dig. Tech. Papers, Feb. 2005, pp. 468–469.

42. K. Shimohigashi, M. Ishihara and S. Shimizu, "Redundancy techniques for dynamic RAMs," Jpn. J. Appl. Phys., vol. 22, pp. 63–67, Apr. 1983.
43. S. Ohbayashi, M. Yabuuchi, K. Kono, Y. Oda, S. Imaoka, K. Usui, T. Yonezu, T. Iwamoto, K. Nii, Y. Tsukamoto, M. Arakawa, T. Uchida, M. Okada, A. Ishii, T. Yoshihara, H. Makino, K. Ishibashi and H. Shinohara, "A 65nm embedded SRAM with wafer level burn-in mode, leak-bit redundancy and Cu e-trim fuse for known good die," IEEE J. Solid-State Circuits, vol. 43, pp. 96–108, Jan. 2008.
44. T. Mano, K. Takeya, T. Watanabe, N. Ieda, K. Kiuchi, E. Arai, T. Ogawa and K. Hirata, "A fault-tolerant 256K RAM fabricated with molybdenum-polysilicon technology," IEEE J. Solid-State Circuits, vol. SC-15, pp. 865–872, Oct. 1980.
45. O. Minato, T. Masuhara, T. Sasaki, Y. Sakai, T. Hayashida, K. Nagasawa, K. Nishimura and T. Yasui, "A Hi-CMOSII 8K $\times$ 8 bit static RAM," IEEE J. Solid-State Circuits, vol. SC-17, pp. 793–798, Oct. 1982.
46. J.-K. Wee, W. Yang, E.-K. Ryou, J.-S. Choi, S.-H. Ahn, J.-Y. Chung and S.-C. Kim, "An antifuse EPROM circuitry scheme for field-programmable repair in DRAM," IEEE J. Solid-State Circuits, vol. 35, pp. 1408–1414, Oct. 2000.
47. J.-K. Wee, K.-S. Min, J.-T. Park, S.-P. Lee, Y.-H. Kim, T.-H. Yang, J.-D. Joo and J.-Y. Chung, "A post-package bit-repair scheme using static latches with bipolar-voltage programming antifuse circuit for high-density DRAMs," IEEE J. Solid-State Circuits, vol. 37, pp. 251–254, Feb. 2002.
48. E. M. Lucero, N. Challa and J. Fields Jr., "A 16 kbit smart 5 V-only EEPROM with redundancy," IEEE J. Solid-State Circuits, vol. SC-18, pp. 539–544, Oct. 1983.
49. S. Shukuri, K. Yanagisawa and K. Ishibashi, "CMOS process compatible ie-flash (inverse gate electrode flash) technology for system-on-a chip," IEICE Trans. Electron., vol. E84-C, pp. 734–739, June 2001.
50. M. Yamaoka, K. Yanagisawa, S. Shukuri, K. Norisue and K. Ishibashi, "A system LSI memory redundancy technique using an ie-flash (inverse-gate-electrode flash) programming circuit," IEEE J. Solid-State Circuits, vol. 37, pp. 599–604, May 2002.
51. M. Tarr, D. Boudreau and R. Murphy, "Defect analysis system speeds test and repair of redundant memories," Electronics, vol. 57, pp. 175–179, Jan. 1984.
52. J. R. Day, "A fault-driven, comprehensive redundancy algorithm," IEEE Design Test Comput., vol. 2, pp. 35–44, June 1985.
53. W. K. Huang, Y.-N. Shen and F. Lombardi, "New approaches for the repairs of memories with redundancy by row/column deletion for yield enhancement," IEEE Trans. Comput. Aided Des., vol. 9, pp. 323–328, Mar. 1990.
54. T. Kawagoe, J. Ohtani, M. Niiro, T. Ooishi, M. Hamada and H. Hidaka, "A built-in self-repair analyzer (CRESTA) for embedded DRAMs," in Proc. ITC, Oct. 2000, pp. 567–574.
55. C.-T. Huang, C.-F. Wu, J.-F. Li and C.-W. Wu, "Built-in redundancy analysis for memory yield improvement," IEEE Trans. Reliab., vol. 52, pp. 386–399, Dec. 2003.
56. N. Ohtsuka, S. Tanaka, J. Miyamoto, S. Saito, S. Atsumi, K. Imamiya, K. Yoshikawa, N. Matsukawa, S. Mori, N. Arai, T. Shinagawa, Y. Kaneko, J. Matsunaga and T. Iizuka, "A 4-Mbit CMOS EPROM," IEEE J. Solid-State Circuits, vol. SC-22, pp. 669–675, Oct. 1987.
57. D. Kantz, J. R. Goetz, R. Bender, M. Baehring, J. Wawersig, W. Meyer and W. Mueller, "A 256K DRAM with descrambled redundancy test capability," IEEE J. Solid-State Circuits, vol. SC-19, pp. 596–602, Oct. 1984.

http://www.springer.com/978-1-4419-7957-5

Nanoscale Memory Repair
Horiguchi, M.; Itoh, K.
2011, X, 218 p., Hardcover
ISBN: 978-1-4419-7957-5