An Efficient Test and Repair Flow for Yield Enhancement of One-Time-Programming NROM-Based ROMs

Tsu-Lin Li†, Nonmember, Masaki Hashizume ‡, Member, and Shyue-Kung Lu †, Nonmember

SUMMARY NROM is one of the emerging non-volatile-memory technologies, which is promising for replacing current floating-gate-based non-volatile memory such as flash memory. In order to raise the fabrication yield and enhance its reliability, a novel test and repair flow is proposed in this paper. Instead of the conventional fault replacement techniques, a novel fault masking technique is also exploited by considering the logical effects of physical defects when the customer’s code is to be programmed. In order to maximize the possibilities of fault masking, a novel data inversion technique is proposed. The corresponding BIST architectures are also presented. According to experimental results, the repair rate and fabrication yield can be improved significantly. Moreover, the incurred hardware overhead is almost negligible.

key words: NROM, data inversion, fault masking, yield

1. Introduction

The NROM technology is one of the novel non-volatile-memory technologies [1]–[7]. It provides simpler fabrication process with low cost and high data density. With the above advantages, the NROM technology is a promising solution and will become popular in the non-volatile-memory market. However, with the ever-increasing VLSI technologies, the miniaturization of transistors makes embedded memories smaller and denser. It then inevitably affects the reliability and yield of the manufactured products [8]. Therefore, we should seek for efficient test and repair solutions for embedded memories.

For random access memories (RAMs), the conventional test and repair solutions usually incorporate built-in self-test (BIST) as well as built-in redundancy analysis (BIRA) modules for test and repair purposes. Redundant elements including spare rows and/or spare columns are usually used to replace faulty memory cells [9]–[12]. That is, these techniques are based on the fault replacement method. For NROM-based ROMs, spares are also used to replace faulty cells during the wafer test phase.

The main drawback of the conventional fault replacement flow is that faulty memory dies should be discarded directly when the incorporated spares are exhaustively used and there are still faulty bits. According to the statistics from the memory foundry, the single defective bits occupy more than 50% of all the fault types. Moreover, it has 96.4% of probability that the number of failed addresses in a memory row is one [13]. According to these facts, besides the traditional fault replacement techniques, we also use the fault masking techniques to further increase the fabrication yield. Different from the RAMs, the codes to be written into NROMs are known. Therefore, if the corresponding bits in the user’s code have the same values as the faulty bits, the fault effects will then be masked directly. Therefore, the discarded dies in the original wafer test phase then still has the possibility to be revived in the package test phase when the programmed codes are considered.

In order to further enhance the fabrication yield, a novel data inversion technique is exploited to maximize the possibilities of fault masking. A control column is added into the NROM array for controlling of the data inversion mechanism. The proposed techniques can be easily integrated into the conventional BIST architectures. A simulator is also developed and implemented for evaluating repair rates and hardware overhead. Experimental results show that the repair rates of the proposed test and repair flow can be significantly improved. Furthermore, the incurred hardware overhead is almost negligible.

The rest of this paper is organized as follows. The preliminaries of one-time-programming NROM-based ROMS are given in Sect. 2. The conventional and the proposed novel test and repair flows are described in Sect. 3. The data inversion technique and the corresponding BIST architecture are introduced in Sect. 4 and Sect. 5, respectively. Experimental results are shown in Sect. 6. Finally, some conclusions are given in Sect. 7.

2. Preliminaries of One-Time-Programming NROM-Based ROMs

The cross section view of a basic NROM cell is shown in Fig. 1 (a). It is basically an n-channel MOSFET with an oxide-nitride-oxide (ONO) gate dielectric stack for replacing the conventional gate dielectric. Bit1 and Bit2 denote the stored binary values depending on the trapped electrons at the corresponding end of the nitride layer above the channel and adjacent to the n+junctions. The top view of the NROM cell is shown in Fig. 1 (b), where the bit lines (BL) are diffusions and the word lines (WL) are polysilicon. Each NROM cell includes two bit lines (BL1, BL2) and a word line.

The wordline WL is set at 3.3 V, BL1 at 0 V, and BL2 at 1.5 V when reading the value of Bit1. If more electrons
are trapped in $Bit_1$, the sensed current from $BL_2$ to $BL_1$ becomes smaller. That is, the threshold voltage of $Bit_1$ becomes a larger value. If the sensed current is smaller than the reference current, it denotes a binary bit 0 (Fig. 2 (a)). Alternately, if the sensed current is larger than the reference current, it denotes that the read data is logic 1 as shown in Fig. 2 (b).

When reading the stored data in $Bit_2$, we apply $3.3 \text{ V}$ at $WL$, $1.5 \text{ V}$ at $BL_1$, and $0 \text{ V}$ at $BL_2$. According to the value of the sensed current, the same criterion can be applied to determine the stored binary value of $Bit_2$. The read-0 and read-1 operations for $Bit_2$ are shown in Fig. 2 (c) and Fig. 2 (d), respectively. When an NROM cell is manufactured, there are no trapped electrons in $Bit_1$ and $Bit_2$. Therefore, the initial values of both stored bits are logic 1.

3. Proposed Test and Repair Flow

Figure 3 shows the conventional test and repair flow. It mainly consists of two phases—wafer test and package test. The wafer test phase is used to make sure if each bit cell in a NROM-based ROM can function correctly. When a NROM-based ROM passes the wafer test, it is considered as a blank ROM. After the customer’s code is programmed, it then enters the package test phase. The objective of this phase is to make sure that the programmed code can be read out without errors. The margin-read-1 (0) test is used to test a memory cell to see if it can store the quality value of 1 (0). The detail steps of the conventional test flow can be found in [13].

As shown in Fig. 3, the repair analysis is performed after the margin-read-1 test. The defect map generated after the margin-read-1 test is used as the inputs for the repair analysis. According to this flow, if faulty memory dies cannot be repaired in the repair analysis step, they will be discarded directly. Therefore, most yield loss occurs in the wafer test phase rather than the package test phase. To cure this dilemma, we will use the observation that memory bits failing the margin-read-1 test are still able to store the value 0 [13], [14]. Therefore, the fault replacement analysis is moved on to the package test phase as shown in Fig. 4. Moreover, the extra fault masking analysis and the repair rate estimation steps are also included in this flow.

Since the package process incurs costs, we cannot package all dies after the wafer test phase. Alternatively, an estimation step is required to estimate the probability of the failed dies in the wafer test phase that can be repaired successfully by using the fault masking and fault replacement steps in the package test phase. A probabilistic model is adopted in this step. The fault masking step basically checks two status—direct masking and complementary masking,
for each pair of faulty physical word and its corresponding logical word. If a memory pair has the status of direct masking, the fault effects can be masked after programming the logical word. On the contrary, if a memory pair has the status of complementary masking, the fault effects can be masked after programming the complemented logical word. Unlike the conventional test and repair flow shown in Fig. 3, the fault replacement analysis in Fig. 4 only performs redundancy analysis algorithms for failed memory rows/columns that cannot be repaired in the fault masking analysis step.

4. Data Inversion Technique

As we know, most of faulty memory words contain only single defective bits. That is, they fail the margin-read-1 test (cannot store value 1). Fortunately, a NROM cell which cannot correctly store the bit value 1 still has very high probability to correctly store binary value 0[13],[14]. If the data word to be written into a faulty memory word contains a single defect bit and the fault effect cannot be masked, then we can complement the data word and then program it into the faulty memory word. Therefore, the fault effect then can be masked.

For an $M \times N$ memory array, an $M$-bit control column ($CC$) should be added into the memory array to implement the data inversion technique. The control column stores a control bit ($CB_i$, $0 \leq i \leq M - 1$) for each memory row. When $CB_i = 0$, it indicates that the data bit to be programmed into the faulty cell is a binary value 0. Therefore, the data word can be written into the memory row $R_i$ directly and the fault effect can be masked by the written data bits. Alternately, if $CB_i = 1$, it indicates that the data bit to be written into the faulty cell in the memory row $R_i$ is logic 1. Since the faulty memory cell will fail the margin-read-1 test, therefore, the fault effect can not be masked. To mask the fault effect, the data bits should be complemented first before programmed into the faulty word.

An $8 \times 8$ example is shown in Fig. 5 where $R_i$ and $C_j$ denote row $i$ and column $j$, $0 \leq i,j \leq 7$. The memory cells with the cross marks denote that they fail the margin-read-1 test. For example, $cello_0$ can only store logic value 0 and cannot store correct logic value 1. Since $CB_0 = 1$, it indicates that the data bit to be programmed into $cello_0$ is 1. Therefore, we should complement the data bit first before programming into $cello_0$. For $cell_1$, the corresponding $CB_1$ is 0. It indicates that the data bit to be written into this cell is logic 0 and the fault effect then can be masked. After fault masking for all memory rows containing single defective bits, the faulty row $R_6$ and the faulty column $C_5$ should be replaced during the fault replacement step as shown in Fig. 4. In this example, the spare row $SR_1$ and the spare column $SC_1$ are used to repair $R_6$ and $C_5$, respectively. When reading a memory word from the NROM array, we should first check its corresponding control bit. If its value is 1, the data word read from the memory should be complemented to get the original data word. If the value is 0, the data word read from the memory can be used directly.

5. Built-In Self-Test Architecture

After passing the package test phase, the customer’s code has been programmed into the NROM array. The customer has no chance to reprogram the code since we concern about the one-time-programmable NROM-based ROMs. Therefore, we should program the control column before executing the package test.

The BIST architecture of an NROM with the proposed data inversion technique incorporated is shown in Fig. 6. In addition to the BIST controller, the NROM memory array is incorporated with a spare group, a MISR (multiple input shift register), and an output complementer. The spare group consists of the spare rows, spare columns, and an additional control column. In the BIST mode, the BIST controller supervises BIST operations and the read out data are then sent to the MISR for data compression.

The original seed stored in the MISR is used to get a pre-calculated signature. After finishing the BIST operations, the compressed signature in the MISR is shifted out to the BIST controller through the shift_out signal. Thereafter, the compressed signature is compared with the golden signature stored in the BIST controller. If there are faults contained in the NROM array, the BistFail signal will be pulled up to high. Otherwise, the BistFail signal will be pulled down to low. After the Finish signal is pulled high,
the final result is shifted out from the SignatureOut port. The ShiftEnd signal will be pulled high after the complete signature is shifted out. The final signature is then sent to the BIST controller to be compared with the golden signature and determine the test result of the NROM array.

After comparing with the gold signature and the result shows that the NROM array is faulty, the BIST operations will be resumed again. However, each time when faulty cells are detected, the faulty syndrome will be send out through the SignatureOut port. Therefore, the locations of faulty cells can be identified easily.

In the normal read mode, the stored data word and its corresponding control bit stored in the control column are read out simultaneously. The value of the control bit is used to control the output complementer, which determine whether the read out data should be complemented or not. If the value of the control bit is logic 1, the read out data will be complemented (through the inverters). Alternatively, the read out data can be used directly. The incorporated spare rows/columns are used to replace faulty rows/columns after the wafer test. This can be done by programming fuses incorporated into the NROM array.

6. Experimental Results

A) Hardware Overhead

The hardware overhead required for implementing the proposed data inversion technique is defined as the ratio between the transistor count of the extra components and the transistor count of the whole memory array. The hardware overhead (HO) can be expressed as:

\[
HO = \frac{(0.5 \times M \times (SC + 1) + 0.5 \times N \times SR)}{0.5 \times M \times N}. \tag{1}
\]

The coefficient 0.5 is the transistor count for an NROM bit cell since each NROM cell can store two binary bits. The term \((SC + 1)\) denotes there are \(SC\) spare columns and one extra control column. The notation \(SR\) denotes the number of spare rows. Table 1 shows the hardware overhead for implementing the data inversion technique with different amounts of spare rows/columns. The first column denotes the memory sizes. From this table we can see that the hardware overhead is almost negligible. For example, for a 2048 x 2048 x 16 NROM array with \(SR = 3\) and \(SC = 3\), the hardware overhead is only 0.342%.

B) Repair Rate

Repair rate is defined as the probability of successful masking and replacement of defective cells. For the implemented simulator, the repair-most algorithm is used for redundancy analysis. We assume that the number \((d)\) of injected defects is based on Poisson distribution as

\[
P(d) = \frac{e^{-\lambda} \lambda^d}{d!},
\]

where \(P(d)\) and \(\lambda\) denotes the probability of the number of defects \(d\) and the average number of defects, respectively. The types of the injected faults include a faulty cell, a faulty row, and a faulty column. During simulation, the probabilities of these fault types can be modified. Moreover, the user code is randomly generated. The injected faults must be first detected by the BIST session. Table 2 shows the repair rates for a 1024 x 128 x 16-bit NROM array with \(SR = 2\) and \(SC = 1\).

In this table, we assume that the percentages of faulty cell, faulty row, and faulty column are set to 96%, 2%, and 2%, respectively. This setting is based on the statistics of foundry data presented in [14].

| Method      | \(\lambda\) |
|-------------|--------------|
|             | 2 | 3 | 4 | 5 | 6 | 7 |
| Conventional| 86.4 | 63.8 | 41.0 | 29.2 | 16.6 | 5.8 |
| [13]        | 96.2 | 84.4 | 72.8 | 59.6 | 41.9 | 34.4 |
| Estimation  | 99.6 | 98.4 | 96.4 | 93.4 | 86.5 | 81.9 |
| DI          | 99.4 | 98.8 | 96.6 | 94.0 | 87.0 | 81.8 |
| Difference  | -0.2 | 0.4 | 0.2 | 0.6 | 0.5 | -0.1 |

7. Conclusions

The data inversion technique is proposed for yield improvement of NROM-based ROMs. Instead of the traditional fault replacement techniques, the fault masking techniques are also exploited. Based on this technique, the novel test and repair flow is also proposed. The proposed flow can be easily incorporated into the ROM BIST architectures. A simulator is implemented to evaluate the hardware overhead and
repair rate. According to experimental results, the repair rates can be improved significantly. Moreover, the incurred hardware overhead is almost negligible.

References

[1] C.-Y. Lu, T.-C. Lu, and R. Liu, “Non-volatile memory technology—today and tomorrow,” Proc. Int. Symp. on the Physical and Failure Analysis of Integrated Circuits Conf. (IPFA), pp.18–23, 2006.
[2] K. Kahng and S.-M. Sze, “A floating gate and its application to memory devices,” IEEE Trans. Electron Device, vol.14, no.9, p.629, 1967.
[3] D.-F. Bentchkowsky, “The metal-nitride-oxide-silicon (MNOS) transistor—characteristics and applications,” Proc. IEEE, vol.58, no.8, pp.1207–1219, Aug. 1970.
[4] I. Fujiwara, H. Aozasa, A. Nakamura, Y. Komatsu, and Y. Hayashi, “0.13 μm MNOS single transistor memory cell with separated source lines,” Proc. IEDM Tech. Digest. Int., pp.995–998, Dec. 1998.
[5] K.-T. Chang, W.-M. Chen, C. Swift, J.-M. Higman, W.-M. Pulson, and K.-M. Chang, “A new SONOS memory using source-side injection for programming,” IEEE Electron Device Lett., vol.19, no.7, pp.253–255, 1998.
[6] M.-H. White, D.-A. Adams, and J. Bu, “On the go with SONOS,” IEEE Circuits and Devices Magazine, vol.16, no.4, pp.22–31, July 2000.
[7] http://www.mxic.com.tw/
[8] A. Allan, D. Edenfeld, W.-H. Joyner, A.B. Kahng, M. Rodgers, and Y. Zorian, “2001 technology roadmap for semiconductors,” Computers, vol.35, no.1, pp.42–53, Jan. 2002.
[9] Y. Zorian and S. Shoukourian, “Embedded-memory test and repair: infrastructure IP for SoC yield,” IEEE Des. Test Comput., vol.20, no.3, pp.58–66, May 2003.
[10] N. Mukherjee, A. Pogiel, J. Rajski, and J. Tyszer, “Fault diagnosis for embedded read-only memories,” Proc. Int. Test Conf. (ITC), pp.1–10, 2009.
[11] Z.-Y. Wang, Y.-M. Tsai, and S.-K. Lu, “Built-in self-repair techniques for heterogeneous memory cores,” Proc. Pacific Rim Int. Symp. in Dependable Computing., pp.69–74, 2009.
[12] J.-F. Li, J.-C. Yeh, R.-F. Huang, C.-W. Wu, P.-Y. Tsai, A. Hsu, and E. Chow, “A built-in self-repair scheme for semiconductor memories with 2-D redundancy,” Proc. Int’l Test Conf. (ITC), Charlotte, pp.393–402, Sept. 2003.
[13] M.C.-T Chao, C.-Y. Chin, Y.-T. Tsou, and C.-M. Chang, “A novel test flow for one-time-programming applications of NROM technology,” Proc. IEEE Int’l Test Conf. (ITC), pp.1–9, 2009.
[14] M.C.-T Chao, C.-Y. Chin, Y.-T. Tsou, and C.-M. Chang, “A novel test flow for one-time-programming applications of NROM technology,” IEEE Trans. VLSI Systems, vol.19, no.12, pp.2170–2183, Dec. 2011.