Memory Fault Diagnosis by Syndrome Compression

Jin-Fu Li and Cheng-Wen Wu
Laboratory for Reliable Computing
Department of Electrical Engineering
National Tsing Hua University
Hsinchu, Taiwan 30013

Abstract

In this paper we present a data compression technique that can be used to speed up the transmission of diagnosis data from the embedded RAM with built-in self-diagnosis (BISD) support. The proposed approach compresses the faulty-cell address and March syndrome to about 28% of the original size under the March-17N diagnostic test algorithm. The key component of the compressor is a novel syndrome-accumulation circuit, which can be realized by a content-addressable memory. Experimental results show that the area overhead is about 0.9% for a 1Mb SRAM with 164 faults. The proposed compression technique reduces the time for diagnostic test, as well as the tester storage capacity requirement.

1. Introduction

In system chips, memories are among the most widely used cores. Memory cores usually represent a significant portion of the chip area, and dominate the yield of the chip. Memory fault diagnosis thus is important so far as yield improvement is concerned: it is used in memory development to find design and/or process errors and inconsistency. Built-in self-diagnosis (BISD) is considered a feasible approach for embedded RAMs, but full diagnosis still requires off-chip analysis by software. Direct access of the RAM cores however is limited by the small number of test pins, so the diagnosis data usually has to be exported from the BISD circuit serially to the external tester, resulting in excessive time for diagnostic test and increased storage requirement for the tester. Compression can be used to solve this problem.

Compression has been used to solve a different but related problem recently, i.e., bitmap display of failed RAMs. Bitmap display on the tester has long been a challenging task, especially for a word-oriented RAM with address and data scrambling. In [1], a special display processor implemented on a commercial tester was presented. It can descramble and serialize the catch RAM data on the tester, allowing the fail-bit information to be displayed on the screen in correct order. As the RAM size grows, the transmission time from the catch RAM to the screen becomes intolerable. Recently, a lossless data compression algorithm has been implemented into the bitmap display processor [2]. The approach was a modified run-length coding algorithm, realized by an ad hoc fail-bit searching (scanning) circuit. It eliminates the bottleneck in the data flow by reducing the size of the bitmap image. So far, no related work has been reported for embedded RAM, wherein hardware overhead is a major concern.

On the other hand, to reduce the time for downloading test data from a workstation to tester, software based compression approaches have been reported [3, 4]. By performing Burrows-Wheeler transformation on the sequence of the test vectors and then applying the run-length coding algorithm, the test data can be compressed with better compression ratio than that of LZW method [5]. Also, to decrease the time for testing a system chip, several compression techniques were proposed in [6–8]. Jas *et al.* [6] proposed a hybrid approach to compress precomputed test vectors. A test set consisting of n test vectors (t_1, t_2, $\cdots$, t_n) was transformed to n difference vectors (t_1, $t_1 \oplus t_2$, $\cdots$, $t_{n-1} \oplus t_n$). The difference vectors were then compressed by the run-length coding algorithm. A statistical coding algorithm was proposed, guaranteeing that the compressed test data can be decoded by a pipeline decoder and the speed is as fast as the tester can transfer it [7]. In [8], a compression scheme based on Golomb codes was presented. First, the difference vectors are generated. Then, the encoding procedure selects the Golomb code parameter—the group size (m). The group size is determined by the longest run of 0s in the difference vectors. Each group consists of m members and a $\log m$-bit sequence (assume m is a power of 2, and the logarithm has a base of 2) uniquely identifies each member within the group. All these schemes are suitable only for data with known statistics.

We present a compression technique in this paper. It can be used to speed up the transmission of diagnosis data from the embedded RAM with BISD support to the external tester. The compression approach is based on a syndrome-accumulation algorithm to compress the faulty-cell address and March syndrome to about 28% of the original size us-

ing the March-17N test algorithm. The key component of the compressor is the syndrome-accumulation circuit that can be realized by a content-addressable memory (CAM). The area overhead is about 0.9% for a 1Mb SRAM with 164 faults. The proposed compression technique reduces the time for diagnostic test, as well as the tester storage capacity requirement.

2. Background

March tests are widely used to detect functional faults of RAMs [9]. Typical faults include stuck-at fault (SAF), transition fault (TF), state coupling fault (CFst), idempotent coupling fault (CFid), and inversion coupling fault (CFin). The March tests are usually used for production test since they have lower complexity. To aid the failure analysis and repair, however, more complicated diagnostic test algorithms [10–14] are required. They can provide more information, e.g., in addition to error location, fault type can be distinguished. A March-17N diagnostic test algorithm is shown as follows [15]:

$$\{\Updownarrow (w0); \Uparrow (r0, w1, r1); \Updownarrow (r1); \Uparrow (r1, w0, r0); \Updownarrow (r0); \Downarrow (r0, w1, r1); \Updownarrow (r1); \Downarrow (r1, w0, r0); \Updownarrow (r0)\}, \quad (1)$$

where $\Uparrow$, $\Downarrow$ and $\Updownarrow$ represent ascending, descending, and arbitrary (either ascending or descending) addressing order, respectively. Also, $w0$ (1) indicates a 0 (1) value is written into a cell, and $r0$ (1) indicates a Read operation of a cell with an expected 0 (1) value. By simulation [15, 16] or manual analysis, we can obtain its *fault dictionary* [15] as shown in Table 1. In the table, E_i=0 (1) means that the ith Read operation of the test algorithm has returned a correct (faulty) value. Also, e.g., CFst(L,0,1) means that when the value of the aggressor (coupling) cell is 0, with the address lower than the victim cell (indicated by an L), then the victim (coupled) cell is forced to 1; (H,↑,↕) means that when there is a rising transition in the aggressor cell, with the address higher than the victim cell (indicated by an H), then the content of the victim cell will be inverted; and so on.

If a fault is detected by a diagnosis test algorithm with k Read operations, then the *March syndrome* of the fault is defined as ($E_0E_1 \cdots E_{k-1}$). For example, the March syndrome for SAF(0) is (011100011100) (see Table 1). In other words, the March syndrome for a fault is the identification number of the fault. Note that a fault may have the same March syndrome under different March tests, and different faults may have the same March syndrome under the same March test. However, in Table 1, we see that each fault has a unique March syndrome, i.e., the March-17N test algorithm can distinguish all the listed faults. The *Hamming syndrome* is defined as the modulo-2 sum of the expected (fault-free) data output vector and the output vector from the memory under test, for word-oriented RAMs.

A BISD circuit with programmable diagnostic algorithms was proposed in [17]. Once a fault is detected, the BISD circuit serially exports the diagnosis data, including the faulty-cell address, March syndrome and/or Hamming syndrome. An error catch and analysis system [15] has also been developed to analyze the diagnosis data. The analysis engine converts the diagnosis data into the error bitmaps. After these bitmaps are processed, the system can provide the information about the fault location and fault type. The problem is that for a RAM with many errors, the time for a diagnostic test can be very long, and the storage capacity requirement for the tester can be very high. In the sequel, we present a compression approach to cope with the problem.

Table 1. The fault dictionary for the March-17N test algorithm.

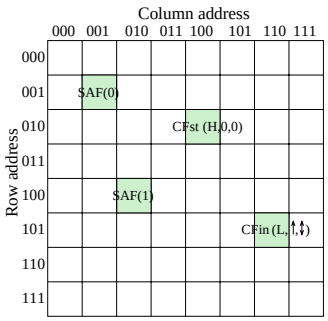
	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}	E_{11}
SAF(0)	0	1	1	1	0	0	0	1	1	1	0	0
SAF(1)	1	0	0	0	1	1	1	0	0	0	1	1
CFst(L,0,0)	0	0	0	1	0	0	0	1	1	1	0	0
CFst(H,0,0)	0	1	1	1	0	0	0	0	0	1	0	0
CFst(L,0,1)	1	0	0	0	1	1	1	0	0	0	0	1
CFst(H,0,1)	1	0	0	0	0	1	1	0	0	0	1	1
CFst(L,1,0)	0	1	1	1	0	0	0	0	1	1	0	0
CFst(H,1,0)	0	0	1	1	0	0	0	1	1	1	0	0
CFst(L,1,1)	1	0	0	0	0	0	0	0	0	0	1	1
CFst(H,1,1)	0	0	0	0	1	1	1	0	0	0	0	0
CFid(L,↑,1)	1	0	0	0	0	0	0	0	0	0	0	0
CFid(L,↑,0)	0	0	0	0	0	0	0	0	1	1	0	0
CFid(L,↓,1)	0	0	0	0	0	0	0	0	0	0	0	1
CFid(L,↓,0)	0	0	0	1	0	0	0	0	0	0	0	0
CFid(H,↑,1)	0	0	0	0	0	0	1	0	0	0	0	0
CFid(H,↑,0)	0	0	1	1	0	0	0	0	0	0	0	0
CFid(H,↓,1)	0	0	0	0	0	1	1	0	0	0	0	0
CFid(H,↓,0)	0	0	0	0	0	0	0	0	0	1	0	0
CFin(L,↑,↕)	1	0	0	0	0	0	0	0	1	1	0	0
CFin(L,↓,↕)	0	0	0	1	0	0	0	0	0	0	0	1
CFin(H,↑,↕)	0	0	1	1	0	0	1	0	0	0	0	0
CFin(H,↓,↕)	0	0	0	0	0	1	1	0	0	1	0	0

3. Compression by Syndrome Accumulation

3.1. Syndrome Accumulation Process

The original approach is as follows. Each time a fault is detected by the March test algorithm, the BISD circuit exports a $\log N$-bit address (for a bit-oriented RAM with N memory cells). A $\lceil \log k \rceil$-bit word that identifies the k-bit March syndrome is also exported. Apparently, there is redundancy since a fault usually is detected by many Read operations in the test algorithm. In Table 1, e.g., a CFst(L,0,1) is detected by five different Read operations. That is, the BISD circuit will export $5 \times (\log N + \lceil \log k \rceil)$ bits of diagnosis data for this fault, but only $\log N + k$ bits are actually required.

We now show an example to explain the notion of *compression by syndrome accumulation*. Consider the 64-bit



(a)

Fault	Diagnosis data
SAF(1)	D1=(100010, 100011100011)
CFin (L,↑,↕)	D2=(101110, 100000001100)
SAF(0)	D3=(001001, 011100011100)
CFst (H,0,0)	D4=(010100, 011100000100)

(b)

Figure 1. (a) A 64-bit RAM with four faulty cells. (b) Diagnosis data words of the faults.

RAM with four faulty cells as shown in Fig. 1(a). Assume the March-17N algorithm shown above is used to test the RAM. The *syndrome accumulation* process is as follows. The fault SAF(1) is the first to be detected by the test algorithm, and it is detected by the first Read operation of the algorithm (i.e., E_0, which is in the second element of the March-17N algorithm). The first *diagnosis data word* (containing the faulty-cell address—row and column addresses—and the *accumulated* March syndrome) thus is D1=(100010,100000000000). The fault CFin(L,↑,↕) can subsequently be detected, also by the first Read operation, and its address is compared with the faulty-cell address of D1. Because the address is different from the faulty-cell address of D1, the second diagnosis data word is D2=(101110,100000000000). In turn, the fault SAF(0) is detected by the second Read operation (i.e., E_1, which is also in the second element of the March-17N algorithm). Its address is compared with the existing faulty-cell addresses (i.e., those of D1 and D2). Since the address of the fault SAF(0) is different from the faulty-cell addresses of D1 and D2, the third diagnosis data word D3=(001001,010000000000) is generated. Similarly, the fault CFst(H,0,0) is detected by the second Read operation and its address is different from the faulty-cell addresses of D1, D2, and D3. Therefore, the fourth diagnosis data word is D4=(010100,010000000000). The third Read operation (i.e., E_2, which is in the third element of the March-17N algorithm) then detects the fault SAF(0) again. Its address is compared with the faulty-cell addresses of D1, D2, D3, and D4. Since the address of the fault SAF(0) is the same as the faulty-cell address of D3, no new diagnosis data word is generated, and the third bit of the March syndrome of D3 is set to 1, i.e., D3=(001001,011000000000). This syndrome accumulation process continues until the test algorithm is finished.

After the diagnostic test algorithm is finished, the final diagnosis data words (in this example, D1, D2, D3, and D4) are obtained, as shown in Fig. 1(b). The diagnosis data is compressed to $4 \times (6 + 12) = 72$ bits from the original $19 \times (6 + 4) = 190$ bits (since the four faults are detected by a total of 19 Read operations).

3.2. Syndrome Accumulation Circuit

The syndrome accumulation process can be realized by a syndrome accumulation circuit (SAC) as shown in Fig. 2. The key component of SAC is a CAM with typical functions—Write operation (WOP), Read operation (ROP), Mask Write operation (MWO), Mask Compare operation (MCO), and Erase operation (EOP) [18]. The WOP stores a word of data into the addressed cells, and sets the *valid bit* of the corresponding word. If the valid bit of a word is set to invalid by EOP, then the match signal (M_i) of this word will be always set to mismatch (M_i=0). Also, the addressing mechanism is different from the conventional address decoder. It mainly consists of an up/down shift register. The signal Hit is logically equivalent to M_0+M_1+$\cdots$+M_{K-1}, i.e., bit-wise OR of the match signals of all the columns. Also, the S and Hit signals determine the operation mode of the register, as shown in Table 2.

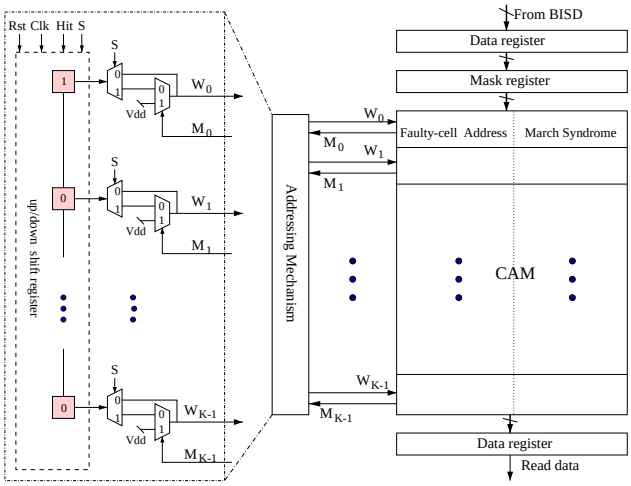


Figure 2. Syndrome accumulation circuit implementation using a CAM.

Table 2. Shift register operation modes.

S	Hit	Operation
1	0	shift down
1	1	Hold
0	0	shift up
0	1	shift up

Initially, EOP is performed for all words and the state of the CAM cell array is all 0. All match signals are low, and Hit=0. The shift register enters the reset state $(100\cdots0)$ (as shown in Fig. 2), which can be done by asserting the reset signal (Rst).

If S=1, then SAC performs the syndrome accumulation process as described above. The address of the first detected fault is stored in the first word (W_0) of the CAM by MWO since ($W_0W_1 \cdots W_{K-1}$)=(100$\cdots$0). This is done by controlling the word lines using the corresponding outputs from the shift register. If later the ith Read operation (E_i) detects this fault, then a 1 is written into the ith bit of the March syndrome of this word by MWO. Once a fault is detected subsequently, its address is compared with the stored faulty-cell addresses by MCO. If Hit=1 (i.e., M_0=1), then the March syndrome of W_0 is updated by MWO. This can be done easily using the auto-addressing mechanism of SAC—since M_0=1, W_0=1. If Hit=0, then the register is shifted down by one bit, and the state of the shift register is (010$\cdots$0). The faulty-cell address is thus stored in the second word (W_1). Also, the ith bit of the corresponding March syndrome is set to 1. We can continue this process until the diagnostic test algorithm is finished. At that time, the CAM stores the complete faulty-cell addresses and diagnosis data.

To export the diagnosis data words, we let S=0, then the shift register performs the shift-up operation. We thus can read the diagnosis data from the CAM word by word. Consequently, only $\log N + k$ bits of diagnosis data need to be exported for each faulty cell.

3.3. Compression Ratio and Hardware Overhead

We define the compression ratio as

$$R = \frac{\text{number of compressed bits}}{\text{number of original bits}}. \quad (2)$$

We assume that around 60% of the defects behave as stuck-at faults [19], and the remaining 40% of the faults are coupling faults. Let U_s and U_c denote the average numbers of Read operations that detect the stuck-at faults and coupling faults, respectively. From Table 1, e.g., the stuck-at faults (including SAF(0) and SAF(1)) are detected by 12 Read operations, so $U_s = 12/2 = 6$. The 20 coupling faults are detected by 56 Read operations, so $U_c = 56/20 = 2.8$. Therefore, the compression ratio can be estimated as

$$R = \frac{\log N + k}{(\log N + \lceil \log k \rceil) \times (U_s \times 0.6 + U_c \times 0.4)}. \quad (3)$$

We have performed experiments using random defects. Table 3 shows the R's for different RAMs under various diagnostic test algorithms that have been reported. The number of Read operations in the March Cd algorithm [13], Modified March C algorithm [12], March-26N algorithm [10], March-12N algorithm [11], and March-17N algorithm [15] are 8, 16, 12, 6, and 12, respectively. The results show that the R's are slightly affected by the size of RAMs. However, they are much more related to the number of Read operations of the diagnostic test algorithm.

To estimate the area overhead of SAC, we use the register-bit equivalent (rbe) model that is technology independent [20]. The areas of a static cell, dynamic cell, and CAM cell are 0.6 rbe, 0.3 rbe, and 1.2 rbe, respectively. Assume that the RAM size is N, and the number of CAM words is K (K is the largest number of faults that can be handled by SAC). Then the area overheads for SRAM and DRAM can be estimated as

$$A_s = \frac{(\log N + k) \times K \times 1.2}{N \times 0.6} \quad (4)$$

and

$$A_d = \frac{(\log N + k) \times K \times 1.2}{N \times 0.3}, \quad (5)$$

respectively. Note that the peripheral circuitry of the RAM and SAC are not estimated, since the cell array of the RAM and CAM dominates the area for large memories. Figure 3 depicts the relation between the A's and K for 1M-bit SRAM and DRAM, under the March Cd test. For example, $A_s \approx 0.9\%$ for a 1M SRAM with 164 faults. This covers most of the defective SRAMs, since the number of chips with less than 164 single-bit failures is about 99% as reported in [2]. Note that a large memory chip usually is divided into many blocks for minimizing the delay and power consumption. In that case, the SAC can be shared among blocks to reduce the overhead. Also, DRAM is much denser than SRAM, so for the same memory size, $A_d > A_s$.

Table 3. Compression ratios of different diagnostic test algorithms.

RAM size	256K	512K	1M
March Cd	36.84%	36.52%	36.23%
Modified March C	26.19%	25.79%	25.42%
March-26N	30.30%	29.95%	29.62%
March-12N	45.71%	45.45%	45.21%
March-17N	28.89%	28.55%	28.24%

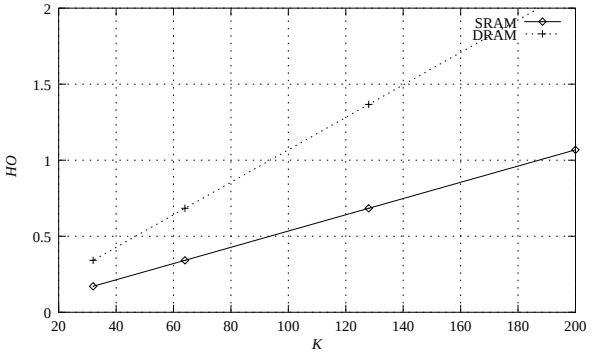


Figure 3. Relation between area overhead and K for 1M-bit SRAM and DRAM.

4. Conclusions

A data compression technique that can be used to speed up the transmission of diagnosis data from the embedded RAM with built-in self-diagnosis (BISD) support has been proposed. The experimental results show that the compression ratio of this scheme is dominated by the diagnostic test algorithm, especially the number of Read operations. The proposed method compresses the faulty-cell address and March syndrome to about 28% of the original size under the March-17N test algorithm. The key component of the compressor is a content-addressable memory. The hardware overhead is about 0.9% for a 1Mb SRAM with 164 faults. The limitation of this scheme is that the area overhead is high if the number of faults increase. This can be reduced if multiple RAM blocks share the same compressor.

References

[1] M. Rich, "A method of flexible catch RAM display for memory testing", in *Proc. Int. Test Conf. (ITC)*, 1986, p. 222.

[2] B. Brown, J. Donaldson, B. Gage, and A. Joffe, "Hardware compression speeds on bitmap fail display", in *Proc. Int. Test Conf. (ITC)*, 1997, pp. 89–93.

[3] T. Yamaguchi, M. Tilger, M. Ishida, and D. S. Ha, "An efficient method for compressing test data", in *Proc. Int. Test Conf. (ITC)*, Washington, DC, Nov. 1997, pp. 79–88.

[4] M. Ishida, D. S. Ha, and T. Yamaguchi, "COMPACT: a hybrid method for compressing test data", in *Proc. IEEE VLSI Test Symp. (VTS)*, 1998, pp. 62–69.

[5] T. Welch, "A technique for high-performance data compression", *IEEE Computer*, vol. 17, no. 6, pp. 8–19, 1984.

[6] A. Jas and N. A. Touba, "Test vector decompression via cyclical scan chains and its application to testing core-based designs", in *Proc. Int. Test Conf. (ITC)*, 1998, pp. 458–464.

[7] A. Jas, J. Ghosh-Dastidar, and N. A. Touba, "Scan vector compression/decompression using statistical coding", in *Proc. IEEE VLSI Test Symp. (VTS)*, 1999, pp. 114–120.

[8] A. Chandra and K. Chakrabarty, "Test data compression for system-on-chip using Golomb codes", in *Proc. IEEE VLSI Test Symp. (VTS)*, 2000, pp. 113–120.

[9] A. J. van de Goor, "Using march tests to test SRAMs", *IEEE Design & Test of Computers*, vol. 10, no. 1, pp. 8–14, Mar. 1993.

[10] M. F. Chang, W. K. Fuchs, and J. H. Patel, "Diagnosis and repair of memory with coupling faults", *IEEE Transactions on Computers*, vol. 38, no. 4, pp. 493–500, Apr. 1989.

[11] T. J. Bergfeld, D. Niggemeyer, and E. M. Rudnick, "Diagnostic testing of embedded memories using BIST", in *Proc. Design, Automation and Test in Europe (DATE)*, Paris, Mar. 2000, pp. 305–309.

[12] C. Hunter, "Integrated diagnostics for embedded memory built-in self test on powerPC devices", in *Proc. IEEE Int. Conf. Computer Design (ICCD)*, 1997, pp. 549–554.

[13] V. N. Yarmolik, Y. V. Klimets, A. J. van de Goor, and S. N. Demidenko, "RAM diagnostic tests", in *Proc. IEEE Int. Workshop on Memory Technology, Design and Testing (MTDT)*, 1996, pp. 100–102.

[14] L. Shen and B. F. Cockburn, "An optimal march test for locating faults in DRAMs", in *Proc. IEEE Int. Workshop on Memory Testing*, 1993, pp. 61–66.

[15] C.-F. Wu, C.-T. Huang, C.-W. Wang, K.-L. Cheng, and C.-W. Wu, "Error catch and analysis for semiconductor memories using March tests", in *Proc. IEEE Int. Conf. Computer-Aided Design (ICCAD)*, San Jose, Nov. 2000, pp. 468–471.

[16] C.-F. Wu, C.-T. Huang, and C.-W. Wu, "RAMSES: a fast memory fault simulator", in *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems (DFT)*, Albuquerque, Nov. 1999, pp. 165–173.

[17] C.-W. Wang, C.-F. Wu, J.-F. Li, C.-W. Wu, T. Teng, K. Chiu, and H.-P. Lin, "A built-in self-test and self-diagnosis scheme for embedded SRAM", in *Proc. Ninth IEEE Asian Test Symp. (ATS)*, Taipei, Dec. 2000, pp. 45–50.

[18] K.-J. Lin and C.-W. Wu, "Testing content-addressable memories using functional fault models and March-like algorithms", *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, vol. 19, no. 5, pp. 577–588, May 2000.

[19] R. Dekker, F. Beenker, and L. Thijssen, "A realistic fault model and test algorithm for static random access memories", *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, vol. 9, no. 6, pp. 567–572, June 1990.

[20] J. M. Mulder, N. T. Quach, and M. J. Flynn, "An area model for on-chip memories and its application", *IEEE Journal of Solid-State Circuits*, vol. 26, no. 2, pp. 98–106, Feb. 1991.