Exploiting Unused Spare Columns to Improve Memory ECC

Rudrajit Datta and Nur A. Touba

Computer Engineering Research Center
Department of Electrical and Computer Engineering
University of Texas, Austin, TX 78712

Abstract

Spare columns are often included in memories for the purpose of allowing for repair in the presence of defective cells or bit lines. In many cases, the repair process will not use all spare columns. This paper proposes an extremely low cost method to exploit these unused spare columns to improve the reliability of the memory by enhancing its existing error correcting code (ECC). Memories are generally protected with single-error-correcting, double-error-detecting (SEC-DED) codes using the minimum number of check bits. In the proposed method, unused spare columns are exploited to store additional check bits which can be used to reduce the miscorrection probability for triple errors in SEC-DED codes or non-adjacent double errors in single adjacent error correcting codes (SEC-DAEC) codes.

1. Introduction

Memories are very dense structures that are especially susceptible to defects. In most cases, memories take up a very large percentage of a chip's area. In order to improve yield, spare rows and columns are often included in a memory to allow for repairing the memory [Kim 98], [Zorian 03]. Defective cells, bit lines, or word lines, can be repaired by replacing the defective rows or columns with the spares. While the worst-case most defective memories on the tail end of the statistical curve may use all of the spare resources, most memories will have unused spare resources after the repair process. This paper proposes a methodology to exploit these unused resources, when available, to improve the reliability of the memory by enhancing its existing error coding.

Transient errors due to radiation, power supply noise, etc., can cause bit-flips in a memory. To protect the data integrity of the memory, an error correcting code (ECC) is generally employed. The most common error correcting code that is used is single-error-correcting, double-error-detecting (SEC-DED) codes [Hamming 50], [Hsiao 70]. These codes can correct single bit errors in any word of the memory and can detect double bit errors. These codes require storing additional check bits in the memory. For a memory with 32 bit data words, 7 check bits are required. So the memory would need 39 columns for each logical word plus any additional spare columns that are included for repair. In some cases, check bits are used along with spare rows and columns to get combined fault-tolerance. In [Stapper 92], interleaved words with redundant word lines and bit lines are used in addition to the check bits on each word.

Some previous work has been done to enhance the reliability of memories without increasing the size of the memory. One limitation of SEC-DED codes is that if a triple-bit error occurs, it may not be detected, but rather it may be miscorrected as if it were a single bit error [Hsiao 70]. The probability of miscorrection for triple bit errors for conventional SEC-DED codes for 32 bit data words is around 60% or more. A code with r check bits can protect up to $2^{r-1}-r$ data bits. Most memories will not have exactly that number of data bits, and hence *shortened codes* must be used. For shortened codes, there is a degree of freedom in selecting the set of columns in the parity-check matrix (H-matrix). In [Richter 08], a search procedure was described for selecting the columns in an H-matrix for a shortened code that minimizes the miscorrection probability for triple bit errors. For example, they found an SEC-DED code for 32 bit data words which has a triple error miscorrection probability of 47%. This increases the reliability of the memory at no additional cost other than a few extra XOR gates in the checker.

Another limitation of SEC-DED codes is that they can only detect, but not correct, double-bit errors. Studies have shown that 1-5% of single event upsets (SEUs) can cause multiple-bit errors (MBUs) [Satoh 00], [Makihara 00], [Kawakami 04]. Most MBUs will affect nearby cells. In [Dutta 07], it was shown that by carefully selecting and ordering the columns in the H-matrix for an SEC-DED code, it is possible to correct all adjacent double-bit errors in addition to correcting all single bit errors thereby creating an SEC-DAEC code. Since the most likely double-bit errors will be adjacent, this is very useful. The limitation of SEC-DAEC codes is that they may not detect all non-adjacent double-bit errors. The SEC-DAEC code reported in [Dutta 07] for 32 bit data words has a 51% double-bit miscorrection probability. In

1093-0167/09 $25.00 © 2009 IEEE
DOI 10.1109/VTS.2009.52

[Richter 08], a better code was found which has a 37% double-error miscorrection probability.

In this paper, we propose an extremely low-cost scheme that exploits unused spare columns to store additional check bits which can be used to significantly reduce the miscorrection probability for triple-errors in SEC-DED codes or non-adjacent double-errors in SEC-DAEC codes. The proposed scheme does not require that any additional columns be added to the memory itself, but rather it simply exploits unused spare columns left over after memory repair. Implementing the proposed scheme requires only adding a few additional XOR gates to the check bit generation logic and providing a simple mechanism to disregard the additional check bits corresponding to the spare columns that become unavailable due to their being used for repair.

The paper is organized as follows. Sec. 2 provides an overview of linear block codes and properties of SEC-DED and SEC-DAEC codes. Sec. 3 describes the proposed scheme for using extra check bits to reduce miscorrection. Sec. 4 describes the hardware implementation for the proposed scheme. Experimental results are shown in Sec. 5, and Sec. 6 is a conclusion.

2. Linear Block Codes

Conventional SEC-DED codes [Hamming 50], [Hsiao 70] are systematic linear block codes [Peterson 72], [Pradhan 96]. In a *(n,k)* linear block code, *k* data bits are encoded by *n*-bit codewords. The number of check bits is $r=(n-k)$. The $(r \times n)$ parity-check matrix (*H*-matrix) completely defines the code. *C* is a codeword of the code if and only if:

$$H \cdot C^T = 0$$

Where C^T is the transpose of the codeword *C*. Let each element in the error vector *E* be a 1 if the corresponding bit is in error and a 0 if the bit is error-free, then an erroneous message can be represented as $V_{error} = V \oplus E$. The syndrome, *S*, is defined as:

$$S = H \cdot V_{error} = H \cdot (V \oplus E) = H \cdot V \oplus H \cdot E = H \cdot E$$

If there is no error (i.e., $E=0$), then the syndrome is all zero (i.e., $S=0$). If the syndrome is non-zero, then an error is detected. Let h_i represent the *i*-th column of the *H*-matrix. If the *i*-th bit has a single error, then the error vector, *E*, is all zero with only the *i*-th bit being a one. The syndrome, which is equal to the product of H and E, will be equal to h_i. For an SEC Hamming code, each column vector in the *H*-matrix is non-zero and distinct [Hamming 50]. This ensures that the syndrome for any single bit error will result in a unique syndrome. By decoding the syndrome, it is possible to determine which bit the error is in and flip the value of that bit to correct the error.

For a double-bit error, the syndrome is equal to the XOR of two columns of the H-matrix. If the XOR of any two columns is equal to the syndrome for any single bit error (i.e., equal to any column in the H-matrix), then the double-bit error syndrome would alias with the single-bit error syndrome. The correction logic would miscorrect the double-bit error thereby missing the error. To avoid this, it was shown in [Hsiao 70], that if every column of the *H*-matrix has an odd number of 1's and is distinct, then the code will be SEC-DED. The reason is that the XOR of any two columns with an odd number of 1's will produce a syndrome with an even number of 1's and hence is guaranteed to be different from any single column. This means that the syndromes for double-bit errors will always be different from the syndromes for single-bit errors, so the code will always detect double-bit errors and not miscorrect them. Hsiao codes are also called *odd-weight column codes*. Note that many double-bit errors have the same syndrome, so it is generally not possible to correct double-bit errors since their syndromes cannot be distinguished.

For triple-bit errors, the syndrome is formed from three columns being XORed together. If the syndrome matches one of columns of the *H*-matrix, then it will be miscorrected as a single-bit error. The number of possible triple-bit errors is C_3^n, and the fraction of those that match columns of the H-matrix is the miscorrection probability for triple-errors. For most conventional SEC-DED codes, it is in excess of 50%.

In [Dutta 07], the *H*-matrix is constructed using odd-weight columns where the columns are carefully ordered so that adjacent columns when XORed together give a syndrome that is not equal to the syndrome for any single-bit error or the syndrome for any other adjacent double-bit error. The number of possible adjacent double-bit errors is equal to *n-1* and the number of single-bit errors is *n*, so the combined set of *2n-1* syndromes must all be distinct from each other. This permits correction of both single-bit errors and adjacent double-bit errors (i.e., SEC-DAEC). However, non-adjacent double-bit errors may match one of the *(n-1)* syndromes of the adjacent double-bit errors and hence may result in miscorrection.

3. Proposed Scheme

The proposed scheme involves exploiting unused spare columns in the memory to store additional check bits. These additional check bits add extra rows to the *H*-matrix and increases the dimension of the syndrome. This makes it easier to distinguish syndromes thereby reducing the chance of miscorrection as well as reducing the chance of a multi-bit error's syndrome aliasing with the error-free all-zero syndrome and not being detected at all.

Note that if all the spare columns are used for repair, then for some chips, it may not be possible to store any additional check bits. Thus, the *H*-matrix that is selected should be such that if no additional check bits are available, it still retains the SEC-DED property. The easiest way to ensure this is to start with an SEC-DED code, and then incrementally add the extra rows to it.

The rows are added one at a time in a greedy fashion so that if only one spare is available after repair, then the maximum benefit for that one row is achieved. Consider the example in Fig. 1 which is a (7,3) SEC-DED Hsiao code. It has $C_3^7 = 35$ possible 3-bit errors, and 28 of those will result in miscorrection. In Fig. 2, an extra check bit is added to the *H*-matrix from Fig. 1. This results in an additional row (the bottom-most one) and an additional column (the right-most one). The last 5 columns in Fig. 2 correspond to check bits and hence form an identity matrix. The left three columns correspond to message bits. The bottom-most bit in the first three columns may be set to any value so as to minimize the miscorrection probability. In Fig. 2, the bottom-most bit in the second column is set to 1 and the others to 0. Now only 12 of the 35 possible triple-bit errors will result in miscorrection.

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 1. Example of (7,3) SEC-DED Hsiao Code

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 2. Adding One Row to Example in Fig. 1

Starting from an SEC-DED code, the proposed scheme adds rows one at a time. The columns corresponding to check bits form an identity matrix, so the degree of freedom is in selecting the 1's and 0's in the row for the columns corresponding to message bits. There are few different strategies that can be used. If the number of message bits is less than say 30, it is possible to do an exhaustive search. Each possible combination of 1's and 0's for the row can be tried and the miscorrection probability computed. The one that minimizes the miscorrection probability is then selected. As the number of message bits gets larger, however, then an exhaustive search is no longer possible.

For larger codes, an alternative to an exhaustive search would be to do a random search and simply keep the best code found. The number of triple-errors is equal to C_3^n which is manageable for *n* up to hundreds. It is feasible to enumerate all the triple-errors and compute the exact miscorrection probability for each candidate row. From our experiments, this gave quite good solutions. When comparing the results for an exhaustive search with those of a random search, there was not a significant difference in the results as can be seen in the experimental data in Sec. 5.

The procedure is the same for SEC-DAEC codes. In this case, the goal is to minimize the number of non-adjacent double-bit errors that miscorrect. This is even faster to evaluate since there are fewer possibilities.

When searching the codes, other criteria can be optimized as well such as total number of XOR gates or logic depth of the syndrome generator.

Each row is added one at a time up to the maximum number of spare columns available in the memory. In the best-case, if no spare columns are used for repair, then all the extra rows will be active for error detection and correction. In the worst-case, when all spare columns are used for repair, then none of the extra rows will be active, and hence only the original SEC-DED code that was used as the starting point will remain.

4. Implementing Proposed Scheme

The proposed scheme can be implemented with very little modification to a normal memory that uses spare columns and is protected with an SEC-DED code. Figure 3 shows an example of the scheme assuming a single spare column. The additional logic that is added to support the scheme is the following:

1. An extra XOR tree in the check bit generator and syndrome generator to support one additional check bit.
2. An extra 2-input AND gate to disable the extra syndrome bit when determining error detection if the spare is used for repair.
3. An extra 2-input OR gate in the correction logic for each data bit to disregard the extra syndrome bit if the spare is used for repair. This is shown in Fig. 4.

Other than what is listed above, the rest of the circuitry is already present in a conventional memory with a spare column and SEC-DED ECC.

If the spare is used for repair, then the MUXes at the input and output of the memory will shift the bits so that the defective column is bypassed. The control signal for the MUX on the far right will be a '1' if the spare is used for repair or if the spare column itself has a defect. If this control signal is a '0', then the spare is available for storing the extra check bit.

So if the spare is not used for repair, then the extra check bit generated by the check bit generator is stored in the spare column, otherwise, it is simply ignored. At the output of the memory, the extra syndrome bit that is generated is ignored if the spare is used for repair in which case error detection and correction are performed just as if that extra syndrome bit didn't exist. However, if the spare is not used for repair, then the extra syndrome bit is used to help increase the chance of detecting a multi-bit error as well as reduce the probability of miscorrection.

If multiple spare columns are used, then there are multiple control signals indicating whether each spare is used for repair or available for storing check bits. The extra control logic that was added to use one spare column would simply be replicated for each additional spare column.

5. Experimental Results

Experiments were performed for common data word sizes to quantify the benefits of the proposed scheme. Table 1 shows the results for minimizing the triple-error miscorrection probability for SEC-DED codes. Results are compared with the best codes from [Hsiao 70] and [Richter 08]. For each code, the number of 2-input XORs that is required is shown along with the raw number of triple-bit errors that are miscorrected and the probability of miscorrection. For the proposed method, results are shown for the cases where one, two, and three spare columns are available after repair. As can be seen, the miscorrection probability is reduced dramatically at the cost of only a small number of additional XOR gates. For 2 and 3 spare columns, the code can detect nearly all triple-errors.

Table 2 shows the results for SEC-DAEC codes. Here the goal is to minimize the number of non-adjacent double-bit errors that are miscorrected. Again, as can be seen, the miscorrection probability drops significantly.

Table 3 shows the comparison between random and exhaustive searches for different size of codes. As can be seen, there is very little difference in the results for the two approaches. Note that for 16 data bits, the solution found by random search had fewer XOR gates than the one found by the exhaustive search. This is because the primary criteria is minimizing the miscorrection probability. In this case, the exhaustive search did find a solution with a lower miscorrection probability.

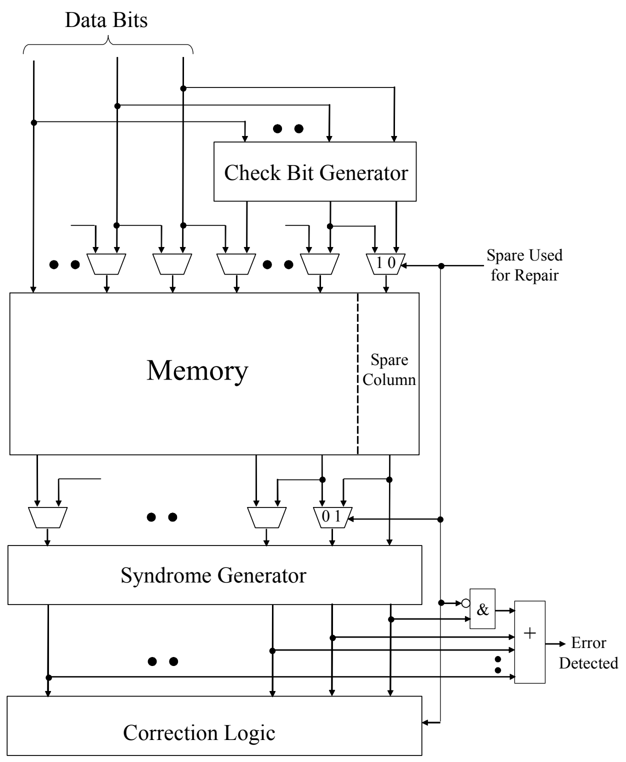

Figure 3. Block Diagram of Proposed Scheme for One Spare Column

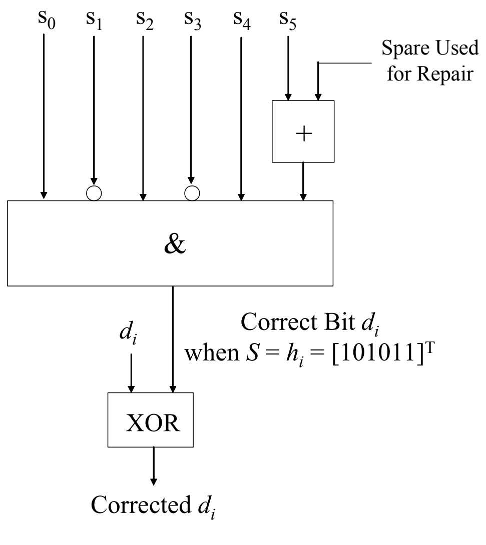
s_0 s_1 s_2 s_3 s_4 s_5
Spare Used for Repair
+
&
d_i
Correct Bit d_i when $S = h_i = [101011]^T$
XOR
Corrected d_i

Figure 4. Example of Bit-Slice of Correction Logic for Proposed Scheme

Table 1. Comparison of Triple-Error Miscorrection Probability for SEC-DED codes

Data Bits	[Hsiao 70]		[Richter 08]		Proposed Method					
					1 Spare Column		2 Spare Columns		3 Spare Columns	
	XORs	Miscorrected	XORs	Miscorrected	XORs	Miscorrected	XORs	Miscorrected	XORs	Miscorrected
16	48	1,000 (64.9%)	-	-	58	448 (25.3%)	70	176 (8.7%)	76	52 (2.3%)
32	96	5,452 (59.66%)	115	4, 284 (46.88%)	118	2,548 (25.8%)	129	1,200 (11.3%)	138	588 (5.1%)
64	181	33,568 (56.28%)	250	26,616 (44.63%)	265	16,176 (26.0%)	308	9,084 (14.1%)	351	7,392 (11.0%)

Table 2. Comparison of Non-Adjacent Double-Error Miscorrection Probability for SEC-DAEC codes

Data Bits	[Dutta 07]		[Richter 08]		Proposed Method					
					1 Spare Column		2 Spare Columns		3 Spare Columns	
	XORs	Miscorrected	XORs	Miscorrected	XORs	Miscorrected	XORs	Miscorrected	XORs	Miscorrected
16	48	118 (56.2%)	-	-	55	68 (29.4%)	62	33 (13.0%)	67	24 (8.7%)
32	96	379 (53.4%)	115	274 (39.0%)	117	203 (27.4%)	130	108 (13.8%)	140	72 (8.8%)
64	224	1316 (53.0%)	250	864 (34.8%)	263	688 (26.9%)	306	469 (17.8%)	353	395 (14.6%)

Table 3. Comparison of Random and Exhaustive Searches when Adding One Spare Row

Data Bits	Triple-Error Miscorrection Probability				Non-Adjacent Double-Error Miscorrection Probability			
	Random Search		Exhaustive Search		Random Search		Exhaustive Search	
	XORs	Miscorrected	XORs	Miscorrected	XORs	Miscorrected	XORs	Miscorrected
16	56	453 (25.5%)	58	448 (25.3%)	56	72 (31.2%)	55	68 (29.4%)
18	63	352 (13.5%)	63	352 (13.5%)	65	51 (17.0%)	65	51 (17.0%)
20	76	496 (15.1%)	76	496 (15.1%)	73	65 (17.2%)	73	65 (17.2%)

6. Conclusions

In this paper, a scheme for exploiting unused spare columns after repair is described for improving memory reliability. It is shown that very little additional hardware beyond what is already present for a memory with spare columns and SEC-DED ECC is required to use this scheme. The experimental results show that the miscorrection probability can be significantly reduced.

Note that if a memory has both spare rows and spare columns, then the spare rows could be used first thereby increasing the number of spare columns that remain for providing additional check bits.

Acknowledgements

This research was supported in part by the National Science Foundation under Grant No. CCR-0426608.

References

[Dutta 07] Dutta, A., and N.A. Touba, "Multiple-Bit Upset Tolerant Memory Using a Selective Cycle Avoidance Based SEC-DED-DEAC Code," *Prof. of VLSI Test Symopsium*, pp. 349-354, 2007.

[Hamming 50] Hamming, R.W., "Error Correcting and Error Detecting Codes", *Bell Sys. Tech. Journal*, Vol. 29, pp. 147-160, Apr. 1950.

[Hsiao 70] Hsiao, M. Y., "A Class of Optimal Minimum Odd-weight-column SEC-DED codes", *IBM Journal of Research and Development*, Vol. 14, pp. 395-401, 1970.

[Kawakami 04] Kawakami, Y., et al., "Investigation of Soft Error Rate Including Multi-Bit Upsets in Advanced SRAM Using Neutron Irradiation Test and 3D Mixed-mode Device Simulation", *Proc. of IEEE Int'l Electronic Device Meeting*, pp. 945-948, Dec. 2004.

[Kim 98] Kim, I., Y. Zorian, G. Komoriya, H. Pham, F.P. Higgins, and J.L. Lewandowski, "Built In Self Repair for Embedded High Density SRAM," *Proc. of International Test Conference*, pp. 1112-1119, 1998.

[Makihara 00] Makihara, A., et al., "Analysis of Single-Ion Multiple-Bit Upset in High-Density DRAMS", *IEEE Trans. on Nuclear Science*, Vol. 47, No. 6, Dec. 2000.

[Peterson 72] Peterson, W.W., and E.J. Weldon, *Error Correcting Codes*, MIT Press, Cambridge, MA, 1972

[Pradhan 96] Pradhan, D.K., *Fault-Tolerant Computer System Design*, Prentice Hall, Upper Saddle River, NJ, 1996.

[Richter 08] Richter, M., K. Oberlaender, and M. Goessel, "New Linear SEC-DED Codes with Reduced Triple Error Miscorrection Probability", *Proc. of International On-Line Testing Symposium*, pp. 37-42, 2008.

[Satoh 00] Satoh, S., Y. Tosaka, S.A. Wender, "Geometric Effect of Multiple-bit Soft Errors Induced by Cosmic-ray Neutrons on DRAMs", *Proc. of IEEE Int'l Electronic Device Meeting*, pp. 310-312, Jun. 2000.

[Stapper 92] Stapper, Charles H., Hsing-san Lee, "Synergistic Fault-Tolerance for Memory Chips", *Proc of IEEE Transactions on Computers*, Vol. 41, No. 9, pp 1078-1087, Sep. 1992.

[Zorian 03] Zorian, Y., and S. Skoukourian, "Embedded-Memory Test and Repair: Infrastructure IP for SOC Yield," *IEEE Design & Test of Computers*, Vol. 20, Issue 3, pp. 58-66, May 2003.