Optimizing the Protection of Narrow Values in Memories Protected with Hamming Codes

Juan Antonio Maestro, *Member, IEEE,* Alfonso Sánchez-Macián, *Member, IEEE*, Pedro Reviriego, *Member, IEEE* and Sanghyeon Baeg, *Member, IEEE*

***Abstract*— Hamming codes are commonly used to protect memories against Single Event Upsets (SEUs). In many applications, it has been observed that a significant percentage of the values do not used all the bits in a word. This can be used to provide an improved level of protection. In this paper a method to increase the reliability of these narrow values in a memory that is protected with a Hamming code is proposed. The technique is also evaluated showing how the protection level of narrow values can be increased significantly. The technique does not require additional hardware as it is completely implemented in software.**

***Index Terms*— Hamming codes, single event upset, error correction codes, memory.**

I. Introduction

Errors caused by radiation particles are a major issue for memory reliability. In particular transient, errors such as Single Event Upsets (SEUs) can corrupt the data stored in a memory [1]. Error Correction Codes (ECCs) are commonly used in memories to avoid data corruption [2], in particular codes that can correct one error per memory word are commonly used. These are referred to as Single Error Correction (SEC) codes. For example Hamming codes [3] are commonly used due to their simplicity. A Hamming code ensures a minimum distance among coded words of three such that single errors can be corrected. However, when there are two errors, the error is detected but can be misinterpreted as a single error and therefore miscorrected. To avoid this issue codes with a minimum distance of four can be used, these codes are commonly referred to as Single Error Correction Double Error Detection (SEC-DED) codes [4].

In several studies it was shown that in many cases the values used in computing systems do not use all the bits available in a word [5][6][7][8]. This can occur for example when variables of a type shorter than the word are used in software applications. These values are referred to as narrow values (Figure 1). The bits that are not used in narrow values can be used to improve the protection against soft errors as suggested in [6]. This can be done by duplicating the narrow value of over the unused bits [7] or using more sophisticated schemes [8]. Then the added redundancy is used to detect the errors. Previous works considered the protection of narrow values in the register files of processors and to that end a bit that identified the narrow values was added to the register filter.

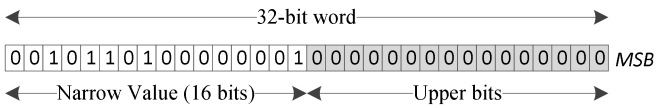

Figure 1: 16-bit narrow value in a 32-bit word

Narrow values will also be present in the main memory and therefore their protection could also be improved. The memory will typically be protected with an ECC and therefore the protection of narrow values has to be combined with the ECC.

In this paper the protection of narrow values in a memory that is protected with a Hamming code is considered. To that end, it is assumed that the protection is implemented in the software running on the processor. Narrow values are identified by the software at compilation time and the additional operations to protect narrow values are executed by the processor. The processor has only access to the data once it has been decoded in the memory read operation. To minimize the number of operations required for protection, the narrow value will not be modified (for example duplicated as proposed in [7]). The rest of the paper discusses the protection of narrow values under those assumptions. The results show that in addition to single error correction, double error detection can be provided for narrow values.

The rest of the paper is organized as follows, in section II the proposed technique is presented showing its performance for a standard Hamming code. Then in section III, an optimization is presented using a Hamming code with a modified shortening. This modification results in a better protection level for narrow values. Finally, the conclusions of this paper are presented in section IV.

II. Proposed Technique

As discussed in the introduction, Hamming codes have a minimum distance of three, which means that any two words differ at least in three bits. These codes are able to detect

Manuscript received September 25th, 2012. This work was supported by the Spanish Ministry of Science and Innovation under Grant AYA2009-13300-C03-01.

J.A. Maestro, A. Sánchez-Macian and P. Reviriego are with Universidad Antonio de Nebrija, C/ Pirineos, 55, E-28040, Madrid, Spain, (phone: +34 914521100; fax: +34 914521110; email: {jmaestro, asanchep, previrie}@nebrija.es).

S. Baeg is with the School of Electrical Engineering and Computer Science, Hanyang University, Ansan 425-791, Korea (e-mail: bau@hanyang.ac.kr).

double errors or, alternatively, correct single errors. In the latter case as mentioned before, double errors might be miscorrected. To avoid this problem, an extended Hamming code can be used which includes an additional parity bit, increasing the Hamming distance to four, so that double errors are always detected (not corrected). The drawback of this approach is the extra memory space required (one additional bit per word).

When working with narrow values, error detection can be enhanced by applying a second detection phase after the standard Hamming decoding. The process is shown in Figure 2.

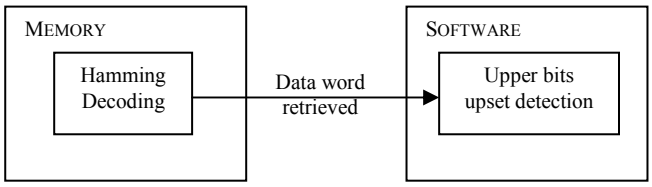

Figure 2: Proposed technique.

As stated in the introduction, narrow values are those which have zeroes in the upper bits. Consequently, when a one is found in those upper bits, an error had occurred in that word. So this second phase consists of checking the upper bits to identify those which are set to one. The upset of these bits can be caused by the element which originated the error, e.g. the impact of a particle, but it can also be the result of a Hamming miscorrection.

This additional validation is performed after the Hamming decoding has been completed and the data word is retrieved from the memory. It can be easily completed by adding a pair of software instructions. The implementation will depend on the software being used. As an example, it can be done by the execution of an AND bitwise operation between the decoded word and a check word with only the upper bits set to one. If the resulting value is different to zero, a double error was detected.

To identify which double errors are detected in each step, a classification is proposed next:

- *Undetected double errors*. Errors which produce a double bit upset and Hamming decoding miscorrects them into a different narrow value data word. The error code word produces a syndrome corresponding to a valid position different from an upper bit.
- *Syndrome out of range*. Double errors affecting two bits which produce a syndrome different from zero and not corresponding to a valid position during Hamming decoding. These errors are detected during Hamming decoding as no correction can be performed in the corresponding position.
- *Upper bit upset when decoding.* These errors produce changes in the lower bits, but an upper bit is miscorrected when decoding. They are detected in the second step of the technique as one of the upper bits is set to one.
- *Upper bit upset by a radiation induced error.* These errors modify one upper bit, and the decoding process produces a syndrome that corrects a different position from the lower bits. They are detected in the second step of the technique as one of the upper bits is set to one.
- *Double upper bit upset caused by a radiation error and decoding.* These errors combine the two previous scenarios. They affect at least one upper bit, and the miscorrection affects a different upper bit. They are detected in the second step of the technique as two of the upper bits are set to one.

To understand the process, an example is shown following with 32-bit words and different scenarios considering narrow values varying from 16 to 26 bits (i.e. 16 to 6 bits being always zero in narrow values).

In this case, the most common implementation of the Hamming code is assumed. This uses a parity Lexicographic check matrix H. The advantage of these matrices is that the resulting syndrome matches exactly the position of the bits to be corrected and no additional computation needs to be done. The H matrix for the Hamming code with 32 data bits and 6 parity bits is shown below:

$$\begin{pmatrix} 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0 \\ 0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0 \\ 0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0 \\ 1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0 \end{pmatrix} \quad (1)$$

All the two-error combinations are applied and error detection is simulated through the proposed technique. Results are included in Tables 1 and 2, and Figure 3.

TABLE I. DOUBLE ERROR DISTRIBUTION IN 32 BIT-WORDS WITH NARROW VALUES

Data bits in narrow values	Undetected	Syndrome Out of Range	Upper bit upset - decoding	Upper bit upset - error	Upper bit upset – decoding and error
16	168	175	63	183	114
17	189	175	64	177	98
18	213	175	63	168	84
19	240	175	60	156	72
20	270	175	55	141	62
21	303	175	48	123	54
22	339	175	39	102	48
23	378	175	28	78	44
24	420	175	15	51	42
25	465	175	0	21	42
26	465	175	6	27	30

Hamming detection for the different narrow value widths is constant as the matrices do not change. It detects a 25% of the possible double errors. Errors found due to the upper bit upset detection depend on the number of bits that must be zero to identify a word as narrow.

TABLE III. PERCENTAGE OF DETECTION PER STEP

Data Word Length	Data bits in narrow values	Undetected	Memory - Hamming	Software – Upper bit upset
32	16	24%	25%	51%
	17	27%	25%	48%
	18	30%	25%	45%
	19	34%	25%	41%
	20	38%	25%	37%
	21	43%	25%	32%
	22	48%	25%	27%
	23	54%	25%	21%
	24	60%	25%	15%
	25	66%	25%	9%
	26	66%	25%	9%

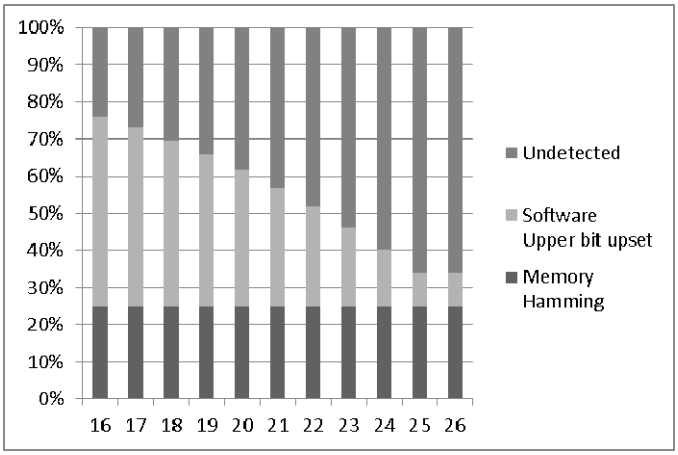

Figure 3: Double error detection for 32 bits and different narrow values.

As expected, the percentage of detection decreases when the narrow value bit length increases, ranging from 51% in narrow values with 16 data bits and 16 zero bits to 9% in narrow values defined with 6 zero upper bits.

The same simulation experiment was performed for 64 bits and four different narrow value bit lengths. Results are shown in Figure 4. The results show similar trends to those observed for 32 bit data words.

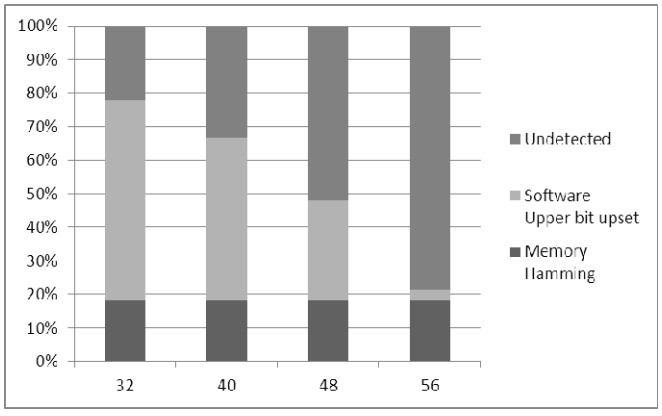

Figure 4: Double error detection for 64 bits and different narrow values.

III. OPTIMIZED IMPLEMENTATION

The approach presented in the previous section has a 24% of undetected double errors for a 32-bit data word in the best case (16-bit length for narrow values) when single error correction is performed.

It is still possible to enhance the Hamming code for narrow values obtaining a SEC-DED code (single error correction and double error detection) in these cases.

The optimization consists of performing a selective shortening of the code matrices. In the previous section, shortening is performed by truncating the lexicographic Hamming matrix to the required size. An alternative [9] is keeping only the columns with odd weight so the sum of two columns cannot result in a third one. Thus, double errors would always be detected.

In the previous example with 32-bit data words, the decoding matrix is shortened from the standard 63 columns Hamming matrix which include 32 columns with odd weight and 31 with even weight. The resulting matrix has 38 columns (data plus parity).

Unfortunately, 6 of the odd weight columns are used for parity purposes (they form the identity matrix), so only 26 columns are remaining for data positions. As the data words are composed by 32 bits, the shortened matrix has to include six columns with even weight.

To overcome this problem, these six columns are allocated to positions corresponding to zero bits in narrow values, i.e. the upper bits. The resulting matrix is shown below.

$$\begin{pmatrix} 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&0&0&0&0&1&1&1&1 \\ 0&0&1&1&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1 \\ 0&1&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \\ 1&0&1&0&1&1&0&0&1&1&0&1&0&0&1&0&1&1&0&1&0&0&1&1&0&0&1&0&1&1&0&0&1&1&0&1&0&0 \end{pmatrix} \quad (2)$$

The double errors are detected as follows:

- Errors affecting two bits corresponding to columns with odd weight will be detected as they will produce a value with an even number of ones. So one of the six upper bits is miscorrected (detected by software) or the syndrome does not match a column of the decoding matrix (detected by the decoding process).
- Errors affecting two bits corresponding to the six upper columns will be detected as those bits should be set to zero.
- Errors affecting one bit from the first 32 ones and another bit from the upper six bits will be detected as the upper bit should be set to zero (and will not be corrected by the decoding process).

If m is the number of data bits and k is the number of parity bits (in the example being discussed $m = 32$ and $k = 6$), it can be deduced that with this optimization SEC-DED is achieved for narrow values as long as they have at most m-k non-zero bits. Otherwise, the first case presented above would not be

true as some of the bits with even weight in the H matrix would be part of the narrow value.

Results of the simulation with this optimization for *m = 32* and *m = 64* bits are shown in Figures 5 and 6. It can be observed that the proposed optimization can detect all double errors when there are at least k zero bits (6 for m=32, 7 for m=64). This corresponds to narrow values of m-k bits (26 for m=32, 57 for m=64).

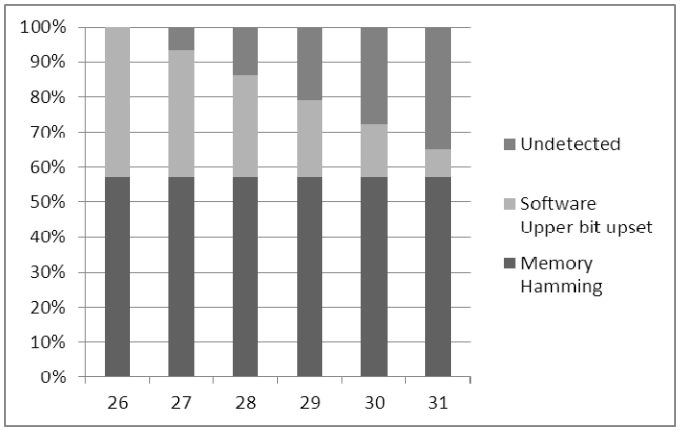

Figure 5: Optimization for m=32 and different narrow values.

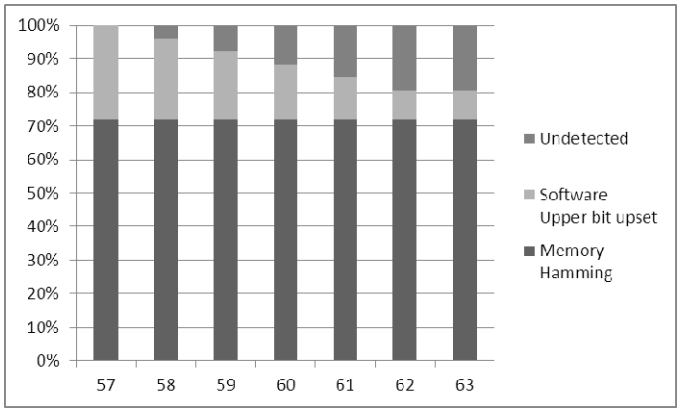

Figure 6: Optimization for m=64 and different narrow values.

From the previous figure it is also clear that the optimized approach provides a better solution for the non-narrow values as well. Errors on non-narrow values are detected only when the Hamming decoding produces a non valid syndrome (marked as Memory – Hamming in the figure). The number of double errors detected with this solution increases from 25% to 57% in the 32-bit data word and from 18% to 72% in the 64-bit data word, when performed at the same time as single error correction.

IV. Conclusions

In this paper a technique to increase the protection level of narrow values against soft errors in a memory protected with a Hamming code has been proposed. The technique has also been evaluated and the results show it can increase the number of double errors that are detected for narrow values. When the technique is combined with a modified shortening of the Hamming code, all double errors in narrow values are detected. These results show how the protection of narrow values in memories protected with error correction codes can be increased.

Future work will first consider increasing the protection of narrow values in memories protected with Single Error Correction Double Error Detection (SEC-DED) codes such as those described in [4]. The protection of narrow values when more advanced codes are used would also be an interesting area for future research.

References

[1] R.C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies", IEEE Trans. On Device and Materials Reliability, Vol. 5, No. 3, 2005, pp. 301-316.

[2] C.L. Chen and M.Y. Hsiao, "Error-correcting codes for semiconductor memory applications: a state-of-the-art review", IBM Journal of Research and Development 28(2), 1984, pp. 124-134.

[3] R.W. Hamming, "Error detecting and error correcting codes", Bell System Technical Journal, vol. 29, pp. 147 – 160, 1950.

[4] M.Y. Hsiao, "A Class of Optimal Minimum Odd-weight column SEC-DED Codes", IBM Journal of Research and Development, 14(4):395–401, 1970.

[5] D. Brooks and M. Martonosi, "Dynamically Exploiting Narrow Width Operands to Improve Processor Power and Performance", Proc. Int'l Symp. High-Performance Computer Architecture (HPCA), 1999.

[6] O. Ergin, O. Unsal, X. Vera and A. González, "Exploiting Narrow Values for Soft Error Tolerance", IEEE Computer Architecture Letters, vol. 5, 2006.

[7] J. Hu, S. Wang and S.G. Ziavras, "In-Register Duplication: Exploiting Narrow-Width Value for Improving Register File Reliability", Proc. Int'l Conf. Dependable Systems and Networks (DSN), 2006.

[8] O. Ergin, O. Unsal, X. Vera and A. Gonzalez, "Reducing Soft Errors through Operand Width Aware Policies", IEEE Transactions on Dependable and Secure Computing, vol. 6, issue, 3, July-Sept. 2009.

[9] S. Lin and D.J. Costello, "Error Control Coding, 2nd Ed.", Englewood Cliffs, NJ: Prentice-Hall, 2004.