A Single Error Correction Double Burst Error Detection Code

Lance Bodnar
Viasat Inc.
6155 El Camino Real
Carlsbad, CA 92009-1699 USA

Gregory Chapelle
Nokia Mobile Phones
12278 Scripps Summit Dr.
San Diego, CA 92131 USA

Abstract -**Particle radiation induced Single Event Upset (SEU), if undetected in computer memory, can have potentially catastrophic effects. An improved error indicating system capable of correcting single errors and detecting multiple adjacent bit burst errors is discussed. This system uses the minimum number of redundant bits possible, and in some cases the number of bits is equivalent to simple parity checking.**

I. INTRODUCTION

Instructions stored in RAM are prone to errors induced by high energy particles that disrupt individual bits or several bits in sequence [1] . The most likely error events are the inversion of a single bit or two neighboring bits in an instruction word. This is due to the physical layout of the memory device. The error induced by a high energy particle is termed a single even upset (SEU) in the device. This effect dominates at high altitude, but still has some effect at ground level, because particles still manage to reach the lower atmosphere.

For high flying aircraft, an SEU can cause a system reset or, worse, can go undetected in flight critical systems, when the stored data are corrupted. To counteract this danger, error correction codes are often employed to detect and correct these errors as they occur in processor memory. However, the number of redundant bits needed to implement modern FECs can be relatively large, driving the number of memory elements needed to support it, and thus increase the actual physical memory space susceptible to SEU.

Our patented error correction code [3] solves these two competing objectives, by not needing a large number of bits for FEC, and by keeping the number memory parts small. It does this by concentrating on single bit and adjacent double bit errors that are by far the most common types of SEU errors. This reduces the constraint on the number of redundant bits needed to implement the error correction and detection code. In many instances this new code may be realized with just the available parity bit locations in the physical memory. A significant savings in memory parts, while obtaining the needed data protection.

II. BACKGROUND

Single bit Error Correction and Double bit Error Detection (SEC-DED) methods find their roots in the early days of computers [2]. A classic SEC-DED that is able to correct every possible bit error and detect all double bit errors requires a minimum distance of 3 between every code word. This classic code is of length $n = 2^m - 1$, where m = the number of redundant bits for the error correction code. As the number of redundant bits added to yield the added data protection, the code word lengths become large quickly.

The number of bits required to support the code minimum distance can be estimated from the Gilbert-Varshamov bound. The Gilert-Varshamov bound states that there exists a linear code C over a field of q elements, having length n, at most m parity checks, and a minimum distance of at least d provided the following bound is met

$$\sum_{i=0}^{d-1}(q-1)^i \binom{n-1}{i} < q^m . \qquad (1)$$

Consider a system with a processor word size of 48-bits wide, stored in three 18-bit wide RAM chips, giving a total of 54-bits of storage per RAM address. The available bits to provide error correction would be $54 - 48 = 6 = m$, but the Gilbert-Varshamov bound isn't met. To guarantee a minimum distance of 3 between all words the system requires $m = 7$ or greater. Because an error correction code isn't obviously available, most system designers resort to using the redundant bits m just as simple parity bits. Parity bits only offer error detection with no possibility of error correction even for single bit errors.

Our error correction code [3] eases this dilemma, by reducing the constraint on a minimum distance of 3 between *all* code words, and only requires this minimum distance between adjacent words in the codebook. Furthermore, we can ease the constraint more by noting that sequential errors will not occur in adjacent bits across a physical boundary between two RAM banks. That is, we disallow detection of adjacent errors in bits 36:35 and 18:17 in our example. This is permissible, since a high energy particle has a very low probability of striking two RAM banks simultaneously, just based on the physical geometry of a circuit board layout. This relaxing of bit error criteria is not strictly necessary, as subsequent work [4] has demonstrated that codes can be designed that don't have this additional criteria. However, for illustrative purposes on how the physical constraints can be

0-7803-8104-1/03/$17.00 ©2003 IEEE

used to relax coding requirements, we will leave this criteria in for the discussion of this paper's results.

III. CODE CONSTRUCTION

The code is created by constructing its parity check matrix **H**. The parity check matrix has dimension m rows by k columns, where from the previous example $m=6$ and $k=54$. A row vector **c** of length k is a valid codeword if and only if

$$\mathbf{cH}^T = \mathbf{0}_{1 \times m}, \quad (2)$$

where $\mathbf{0}_{1 \times m}$ is a 1 by m row vector of zeros. We can generate a codeword **c** from a $k-m$ (48 bit) information vector **i** using a $k-m$ row by k column generator matrix **G** that satisfies

$$\mathbf{GH}^T = \mathbf{0}_{(k-m) \times k}$$

$$\mathbf{c} = \mathbf{iG}. \quad (3)$$

It is well known that if the code is *systematic*, that is, all the information bits appear unaltered and in the same order in the codeword, then **G** can easily be constructed from **H**. The requirement for **G** and the relationship between **G** and **H** when they describe a binary systematic code is

$$\mathbf{G} = [\, \mathbf{P}^T \, \mathbf{I}_k \,]$$

$$\mathbf{H} = [\, \mathbf{I}_{n-k} \, \mathbf{P} \,], \quad (4)$$

where **P** is an (n–k) by k matrix and in our case n is 54, k is 48, and n–k = m = 6.

If some event causes errors to occur in a stored codeword the result is $\mathbf{r} = \mathbf{c}+\mathbf{e}$, where the i^{th} bit of **e** is 1 if an error has occurred in that bit and a 0 if no error occurred. All sums are performed mod 2, which is equivalent to a bitwise XOR. If one or more errors occur then **e** is nonzero and,

$$\mathbf{rH}^T = (\mathbf{c}+\mathbf{e})\mathbf{H}^T = \mathbf{cH}^T + \mathbf{eH}^T = \mathbf{0}_{1xm} + \mathbf{s} = \mathbf{s}, \quad (5)$$

where the nonzero 1 by m row vector **s** is called the *syndrome* for the error pattern **e**. Note that **s** depends only on **e** and not the codeword **c**. Also note that if the error pattern **e** happens to be a valid codeword, then $\mathbf{s} = \mathbf{0}_{1xm}$ and the error is undetected. If only one error occurs in a codeword, then only one bit position of **e** will contain a 1. If the i^{th} bit of **e** is 1, then it's easy to see that $\mathbf{eH}^T = \mathbf{s}$ will simply be the i^{th} column of **H**. If **e** is a weight 2 error pattern, say bits i and j are 1, then $\mathbf{eH}^T = \mathbf{s}$ will be the sum of columns i and j of **H**. Each column of **H** can be viewed as a m bit number, or an element from $GF(2^m)$. So there are 2^m-1 possible nonzero m bit numbers (63 in our example) for the n (54) columns of **H**. The syndrome of all zeros (000000) is excluded since it indicates no error in the codeword. Using our example, the construction for the parity check matrix **H** would be as follows:

1. Start with a 6 by 63 matrix whose columns are all possible nonzero 6 bit numbers.

2. Delete 9 columns. These will be the syndromes for 2 bit burst errors. The resulting matrix **H** is now a parity check matrix for a (54, 48) linear code.

3. Each of the remaining 54 columns is unique, meaning that each single bit error pattern will have a unique syndrome and it will be possible to correct any single error by computing the syndrome of the received word. In addition, if the 54 columns are arranged so that any two adjacent columns sum to one of the 9 columns (syndromes) deleted in step 2, then each 2 bit adjacent error pattern can be detected since it has a different syndrome than any single error pattern.

4. To create the systematic code described earlier, we specify that the first 6 columns of **H** form a 6 by 6 identity matrix $\mathbf{I}_6$. This implies that 5 of the columns deleted in step 2 must be (110000), (011000), (001100), (000110), and (000011).

One possible SEC-DBED code generated from this construction method is shown in Fig. 1. The syndromes for burst errors of two are also shown in Fig. 1. The syndromes for single bit errors will contain a one in the i^{th} bit corresponding to the corrected code word as the i^{th} column of **H**. This allows the detection of two bit burst errors and correction of any single bit errors.

This method of code generation could be automated to search out a set of possible candidate codes. The method also generalizes to different size information vectors and number of additional bits needed to have a realizable code. This is discussed further in [4].

IV. IMPLEMENTATION AND PERFORMANCE

The implementation of this code is system specific, but the general architecture is a processor reading from memory with additional hardware or software checking for nonzero

COLUMN	53			50				45					40				36
HEX	20	10	08	04	02	01	31	29	25	23	13	0B	07	37	2F	0E	3E	26
BINARY	1	0	0	0	0	0	1	1	1	1	0	0	0	1	1	0	1	1
	0	1	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	0
	0	0	1	0	0	0	0	1	0	0	0	1	0	0	1	1	1	0
	0	0	0	1	0	0	0	0	1	0	0	0	1	1	1	1	1	1
	0	0	0	0	1	0	0	0	0	1	1	1	1	1	1	1	1	1
	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0

COLUMN	35					30					25					20		18
HEX	0A	05	09	3B	38	19	1F	1C	3D	0D	2C	2A	1A	28	24	16	15	34
BINARY	0	0	0	1	1	0	0	0	1	0	1	1	0	1	1	0	0	1
	0	0	0	1	1	1	1	1	1	0	0	0	1	0	0	1	1	1
	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
	0	1	0	0	0	0	1	1	1	1	1	0	0	0	1	1	1	1
	1	0	0	1	0	0	1	0	0	0	0	1	1	0	0	1	0	0
	0	1	1	1	0	1	1	0	1	1	0	0	0	0	0	0	1	0

COLUMN	17		15					10					5					0
HEX	14	12	11	22	3A	36	35	39	3F	3C	1D	2E	1E	2D	2B	1B	17	27
BINARY	0	0	0	1	1	1	1	1	1	1	0	1	0	1	1	0	0	1
	1	1	1	0	1	1	1	1	1	1	1	0	1	0	0	1	1	0
	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1	1	0	0
	1	0	0	0	0	1	1	0	1	1	1	1	1	1	0	0	1	1
	0	1	0	1	1	1	0	0	1	0	0	1	1	0	1	1	1	1
	0	0	1	0	0	0	1	1	1	0	1	0	0	1	1	1	1	1

WEIGHT 2 BURST ERROR SYNDROMES: 30 18 0C 06 03 21 33 32 0F

Fig. 1. Parity check matrix H for the SEC-DBED code.

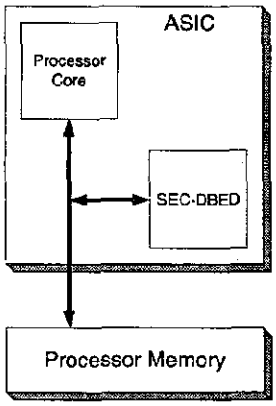

Fig. 2. Architecture implementation.

syndromes. The generation of the syndrome can be handled by a simple FGPA or ASIC monitoring the processor access of data from the memory, to correct or detect the word errors. When the processor writes to memory a code word is generated to include in the memory, thereby enabling the SEC-DBED capability.

With ASICs commonly able accommodate DSP cores accessing external memory, the implementation of the syndrome checking can be handled by the ASIC itself with an internal SEC-DBED function block, as shown in Fig. 2.

Consider an example of how the code of Fig. 1 carries out the function of correcting single bit errors and detecting double bit errors. Given a 48-bit word, i, from the processor memory, the code word generated is obtained from Eqn. (3). If i=1101 1011 0111 1000 1010 0101 1111 0000 0010 0100 0011 1100, listed from MSB to LSB, then the code word would be equal to j+i where j is the additional six bits available and is 000100.

This code word can be listed in hexadecimal as 04DB78A5F0243C.

If an error event changes the 24th bit, then the received vector would be equal to **r**=04DB78A5F0244C, listed from MSB to LSB in hexadecimal. Generating the syndrome from Eqn. (5) produces s=101010 = 2A hexadecimal and indicates the bit position by examining Fig. 1 generator matrix. The bit position where syndrome 2A occurs in the parity check matrix **H** is in the 24th bit and now this error can be corrected in the 24th bit.

If an error event changes the 30th and 31st bits, then the received code words with the double bit error would be equal to **r**= 00 0100 1101 1011 0111 1000 0110 0101 1111 0000 0010 0100 0011 1100 binary = 04DB7865F0243C hex, listed from MSB to LSB. Generating the syndrome from Eqn. (5) produces s=10 0001 = 21 hex and this syndrome is one of the weight 2 burst error syndromes shown at the bottom of Fig. 1. Since the weight 2 burst error syndromes are common for all possible adjacent bit errors, the double bit error can't be corrected. However, the known error syndrome for the 2 burst error allows the error to be detected. Corrective action by the processor can then occur and will depend upon the system requirements. Corrective action could be anything from reseting the processor, reloading or rereading the memory, or simply logging the accumulated number of errors that have occurred during the system operation.

We have seen how relying on information theoretic bounds without understanding the underlying physical mechanism for errors could lead to the conclusion that a code does not exist. By relaxing some of the coding constraints to match the physical implementation of the code, a realizable code can be obtained that performs the required error correction and detection.

V. CONCLUSION

In this paper we have shown the design approach for a family of single bit error correction and double bit burst detection codes that utilize a small number of additional bits to realize this operation. We have demonstrated how it can realistically be used in a system to guarantee the robustness and correctness of data storage in a critical system environment.

The system sturdiness is significantly improved by using the available bits to allow additional error information to be obtained. This extra error information comes only at the cost of some additional processing, and the same number of bits are used that otherwise might only have provided simple error detection through parity detection.

REFERENCES

[1] Y. Tosaka, S. Satoh, T. Itakura, H. Ehara, T. Ueda, G. Woffinden, S. Wender, "Measurement and Analysis of Neutron-Induced Soft Errors in Sub-Half-Micron CMOS Circuits", *IEEE Transactions on Electron Devices*, Vol. 45, No. 7, July 1998, pp. 1453-1458.

[2] S. Lin and D. Costello, *Error Control Coding: Fundamentals and Applications*, Prentice-Hall Inc., 1983.

[3] US Patent 6519734, "Single bit error correction, double burst error detection technique", February 11, 2003, L. Bodnar and G. Chapelle.

[4] US Patent 6536009, "Technique for generating single-bit error-correcting, two-bit burst error detecting codes", March 18, 2003, L. Bodnar.