A (16,8) ERROR CORRECTING CODE (T=2) FOR CRITICAL MEMORY APPLICATIONS

M.S. Hodgart and H.A.B. Tiggeler

Surrey Space Centre,

University of Surrey,

Guildford, Surrey, GU2 5XH, UK.

Tel: +(44) 01483 259278   Fax:+(44) 01483 259503

Email: S.Hodgart@ee.surrey.ac.uk , H.Tiggeler@ee.surrey.ac.uk.

http://www.ee.surrey.ac.uk/EE/CSER/UOSAT/

Abstract

High density SRAMs generate errors in their stored data because of natural radiation. This is a particular problem for computing on-board a satellite , where the single-error correction of the usual Hamming code can be inadequate. The two-bit error correcting code described here is a more powerful and efficient alternative.

1. Introduction

For the secure transaction of data between a central processing unit (CPU) and its local random access memory (RAM) the traditional means of error detection and correction (EDAC) is a Hamming code. In the theory of error control the designation (n, k) denotes a block code that takes a k-bit data word and maps it to an n-bit code word. A typical Hamming code is (12,8) shortened from full (15,11). As is well known this code allows $t = 1$ correction of one error bit per stored word. For computers on board a satellite, and using the latest high density byte-wide RAMs, there is however a definite risk of *two* error bits occurring within one byte of stored data; either from the impact of a particularly energetic single event upset (SEU), or from a second SEU creating a second error, and before the computer has had time to 'wash' the first error [1].

The $t = 2$ bit-correcting code described here deals with this problem. This code will be useful for any system or situation where the Hamming code is inadequate, because of the increased susceptibility to error of the computer memory.

For an EDAC that controls the errors in *blocks* of data - i.e. a RAM disc operation - a low-redundancy modified Reed-Solomon code is ideal. For example, ref. [2] describes a (520, 512) RS code that corrects up to $t = 2$ *bytes* of error in a block of 520 bytes, using only 8 bytes of parity. In this application - coding for a RAM-disc - there is time available for actual computation: which takes place in parallel to the writing or reading of the block to or from the memory.

But no computation is allowed for an EDAC that is coding individual words. In this case the encoding must 'flow forward' from a k-bit data word directly to a n-bit code word. Similarly the decoding must 'flow back' from the n-bit code word directly to the k-bit data word. Any necessary error correction must also be implemented in this flow process and all the EDAC must be implemented in combinatorial logic.

The power of a block code is measured by d_{min} - the minimum distance that exists between pairs of code words. The code is usually *systematic* - the k bits of data are unchanged - and the code word consists of an additional $n - k$ generalised parity bits that create the distance property of the code. With its $t = 1$ capability the Hamming code has a $d_{min} = 3$. The extended Hamming code has a $d_{min} = 4$. But a $t = 2$ correcting code must have distance $d_{min} = 5$. The code should also be linear (the EXOR sum of code words also being a code word). Ideally the code should have a structure $(n, 8)$ where $n \leq 16$, making the most efficient use of byte-size RAM.

The short block codes, with a greater distance than Hamming, that are offered in most texts, belong to the BCH family of codes. But no BCH code can meet the above requirement.

2. The new code

Now it is a mathematical fact - easily established by computer search - that the subset of all possible 16-bit words having a distance d_{min} = 5 has exactly 2^8 = 256 members. It follows that a (16, 8) block code must exist with the required distance property. It remains only to find such a code and a mathematical rule for making it linear and systematic.

We draw attention to a class of codes that have long been known but do not seem to have been exploited to any significant extent. These are the *quasi- cyclic* codes. They would seem to be the most powerful available short block codes having the half-rate structure $(n, n/2)$.

A quasi-cyclic (16, 8) code with the requisite d_{min} is described in considerable detail in [3]. A draw-back is that the code derived in this early text is not systematic. Some analysis by us (not given here) was however able to determine the required mapping to a linear and systematic (16, 8) code which maintains d_{min} = 5. This is the code presented here.

2.1 Encoding

Following standard theory an 8-bit data vector **m** is stored unaltered in RAM. Simultaneously the encoder propagates a parallel 8-bit parity vector **p** into parallel RAM, derived from this vector **m** and the parity matrix **P** according to the relation

$$\mathbf{p} = \mathbf{m}\,\mathbf{P}$$

where all the arithmetic is modulo-2 (fig.1)

The parity-generating matrix that defines this linear and systematic (16,8) code is

$$\mathbf{P} = \begin{vmatrix} 0&1&0&0&1&1&0&1 \\ 1&0&1&0&0&1&1&0 \\ 0&1&0&1&0&0&1&1 \\ 1&0&1&0&1&0&0&1 \\ 1&1&0&1&0&1&0&0 \\ 0&1&1&0&1&0&1&0 \\ 0&0&1&1&0&1&0&1 \\ 1&0&0&1&1&0&1&0 \end{vmatrix}$$

A 16- bit code vector may be specified as **u** = [**p m**]. The quasi-cyclic nature of the code is seen from **P**, in that every row (or column) is a cyclic shift of the neighbouring row or column

2.2 Decoding

The advantage of Hamming is the relatively simple flow-through logic for decoding. Such does not seem to be the case for the quasi-cyclic code (we do not know of any simple theory). Consequently the design and implementation of flow-through decoder for the quasi-cyclic code becomes practicable only with the latest technology (see below).

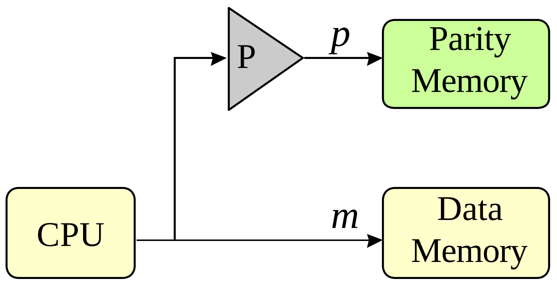

Figure 1 Encoder Schematic

Following standard theory, a decoder must read the entire vector $\mathbf{u} \longrightarrow \mathbf{u}' = [\mathbf{p}' \;\; \mathbf{m}']$ where $\mathbf{p} \longrightarrow \mathbf{p}'$ and $\mathbf{m} \longrightarrow \mathbf{m}'$ are the read vectors, and may be in error from the original **p** and **m** respectively. A syndrome **s** is then derived by re-encoding **m'** and EXOR-ing with **p'** as in

$$\mathbf{s} = \mathbf{m}'\,\mathbf{P} \oplus \mathbf{p}'$$

From standard theory this syndrome is the product of whatever is the error vector **e** and the parity check matrix **H**, as in

$\mathbf{s} = \mathbf{e}.\mathbf{H}^T$

where

$$\mathbf{H} = \begin{vmatrix} 1&0&0&0&0&0&0&0&0&1&0&1&1&0&0&1 \\ 0&1&0&0&0&0&0&0&1&0&1&0&1&1&0&0 \\ 0&0&1&0&0&0&0&0&0&1&0&1&0&1&1&0 \\ 0&0&0&1&0&0&0&0&0&0&1&0&1&0&1&1 \\ 0&0&0&0&1&0&0&0&1&0&0&1&0&1&0&1 \\ 0&0&0&0&0&1&0&0&1&1&0&0&1&0&1&0 \\ 0&0&0&0&0&0&1&0&0&1&1&0&0&1&0&1 \\ 0&0&0&0&0&0&0&1&1&0&1&1&0&0&1&0 \end{vmatrix}$$

is readily derived from the specifying **P** matrix.

A maximum likelihood function can be defined

$$\hat{\mathbf{e}} = \hat{\mathbf{e}}(\mathbf{s})$$

where an error vector $\hat{\mathbf{e}}$ with the least number of error bits (either one or two) is deemed responsible for the observed syndrome **s** . A look-up table (LUT) is needed to implement this function. The entries of **s** in this table are the single columns in the **H** matrix, corresponding to single bit errors in actual **m'** (8 right-most columns); and the EXOR sums of pairs of columns, corresponding to double bit errors in **u'**, or a single bit error in **m'** with a single bit error in **u'** (i.e. a double-bit error distributed across the two bytes).

A correctable syndrome matches one of these possibilities in the table, and generates an appropriate vector $\hat{\mathbf{e}}$. There are $8 + 8{\times}7/2 + 8{\times}8 = 100$ correctable syndromes and therefore error vectors, assuming there is no interest in correcting the parity part **p'**. Since the EDAC consists entirely of logical gates (including the LUT) the vectors **p'** and **m'** propagate directly to the desired **s**, and then through more logic to emerge as an 8-bit vector $\hat{\mathbf{e}}$. When there is no error then $\mathbf{s} = \mathbf{0}$ and the vector $\hat{\mathbf{e}} = \mathbf{0}$ (all-zero). If and when an error exists and it is correctable then the decoding is completed (see fig. 2) with

$$\hat{\mathbf{m}} = \hat{\mathbf{e}}(\mathbf{s}) \oplus \mathbf{m}'$$

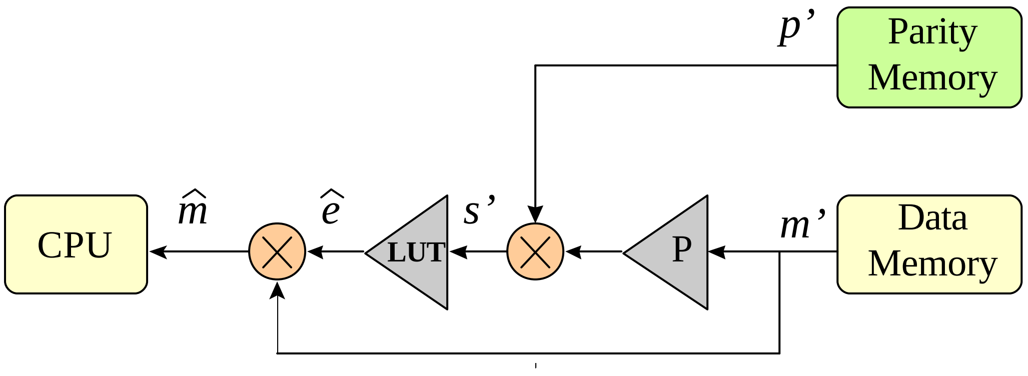

Figure 2 Decoder Schematic

3. Physical implementation

The flow-through EDAC sits between CPU and memory. An advantage of this structure is the simple control logic. Only a Read and Write strobe are required from the CPU to direct the data flow with the EDAC controller. The EDAC itself can be implemented in two different programmable technologies, namely the Complex Programmable Logic Device CPLD or Field Programmable Gate Array FPGA. For its wide fan-in gate capabilities the CPLD normally has the advantage . But a major consideration is power consumption and radiation tolerance in a satellite application, which normally favours the FPGA. We find however that one proprietary FPGA excels above others; and in the latest family of products maintains the low power consumption of an FPGA while providing the speed of a CPLD**.**

A practical codec was developed in VHDL and synthesised using a proprietary tool, based on the principles described in this paper. The look-up table is generated by a simple C program and then converted to VHDL by a ROM generator [5]. In a typical implementation the total gate count is less than 2000 when the codec is optimised for minimum delay using proprietary software. A meaningful measure of the performance is a comparison between the delay time through a codec based on the Hamming code and delay time through a codec based on the quasi-cyclic code, using the same technology. On this basis - for the latest Actel SX family - a typical delay in encoding is not significantly greater (10 ns to 12 ns); while the delay in decoding increases from 18 ns to 26 ns. For satellite operation this is not a significant increase in overhead, compared to a relatively long access time - typically 100 ns - of the usual low-power memory employed

4. Summary

A novel (but derived from an old) code has been described that improves on the famous Hamming code, for a computing application. This code is needed when the environment is likely, or could just possibly, create errors that are uncorrectable by Hamming. The code confers greatly enhanced data security for general-purpose computing .

Implementation of the EDAC is transparent to the computer: there are no interrupts and no additional computation. The overall system cost is a 100% increase in stored data, which is no greater than the Hamming code in a comparable context (a 12,8 code still needs two bytes for storage) . The increase in delay time, from implementing this new code in a typical application, is small.

5. Conclusion

The paper has identified an application-specific low-complexity codec: the design (i) allows a natural (power of 2) block size of data; (ii) has DEC and EED capability; (ii) achieves a low complexity by implementing a non-standard method of decoding. The complexity of the completed code is sufficiently low that a single low-cost 8000 gates Actel A54SX08 FPGA is all that is needed for its implementation.

References

[1] C.I. UNDERWOOD, R. ECOFFET, "Observations of Single-Event Upset and Multiple-Bit Upset in Non-Hardened High-Density SRAMs in the TOPEX/Poseidon Orbit", *IEEE/NSREC Conference*, Snowbird, Utah, USA, July, 1993.

[2] HODGART M.S, and TIGGELER H.A.B, "Fast Low Complexity Reed Solomon Codec for Space and Avionics Ramdisk Applications" DASIA98, Athens, Greece, 25-28 May 1998.

[3] S.B.Wicker, V.J.Bhargava "Reed Solomon Codes and their applications" 1994 IEEE press.

[4] PETERSEN W.W and E.J.WELDON "Error-correcting Codes" MIT Press. 2nd edition 1972 pp 256- 261.

[5] I. Kleyner, R. Katz, H. Tiggeler "System-On-Chip Data Processing and Data Handling Spaceflight Electronics" MAPLD99, Laurel, Maryland, USA, 28-30 September 1999.

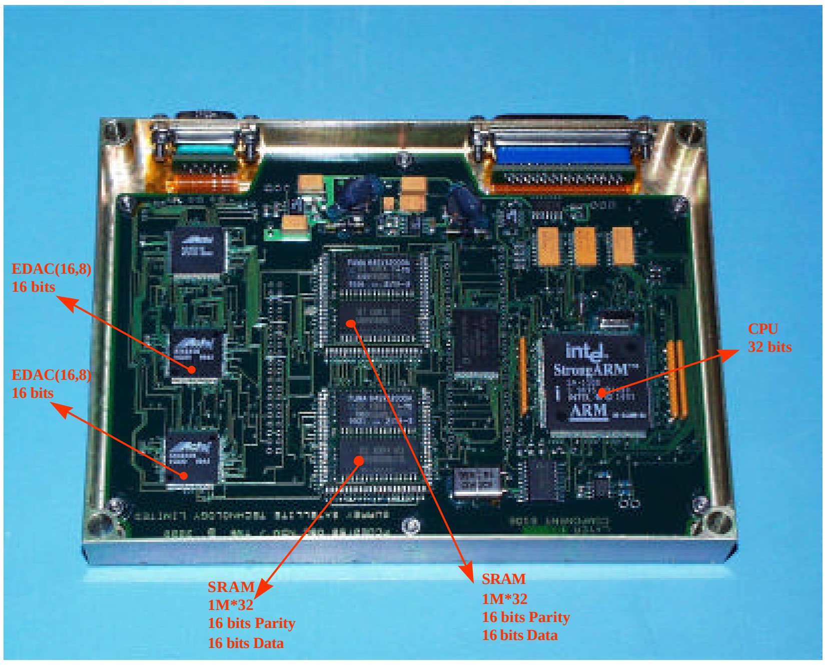

Figure SNAP Nano-Satellite On-Board Computer.