DEC ECC design to improve memory reliability in Sub-100nm technologies

Conference Paper · October 2008

DOI: 10.1109/ICECS.2008.4674921 · Source: IEEE Xplore

Riaz Naseer
University of Southern California

DEC ECC Design to Improve Memory Reliability in Sub-100nm Technologies

Riaz Naseer and Jeff Draper
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
{naseer, draper}@isi.edu

***Abstract*—Exacerbated SRAM reliability issues, due to soft errors and increased process variations in sub-100nm technologies, limit the efficacy of conventionally used error correcting codes (ECC). The double error correcting (DEC) BCH codes have not found favorable application in SRAMs due to non-alignment of their block sizes to typical memory word widths and particularly due to the large multi-cycle latency of traditional iterative decoding algorithms. This work presents DEC code design that is aligned to typical memory word widths and a parallel decoding implementation approach that operates on complete memory words in a single cycle. The practicality of this approach is demonstrated through ASIC implementations, in which it incurs only 1.4ns and 2.2ns decoding latencies for 16- and 64-bit words, respectively, using 90nm ASIC technology. A comparative analysis between conventionally used ECC and DEC ECC for reliability gains and costs incurred has also been performed.**

I. INTRODUCTION

SRAM reliability faces serious challenges from radiation-induced soft errors (transient faults induced by ionizing radiation) and process-variation-induced defects in sub-100nm technologies [1][2]. SRAM cells are designed with minimum geometry devices to increase density and performance, resulting in reduced critical charge to upset cells and more pronounced effects from process variations. Therefore, it has become conventional to protect memories with the application of error correcting codes (ECC) such as single-error-correcting (SEC) Hamming code [3], single-error-correcting-double-error-detecting (SEC-DED) extended-Hamming, or SEC-DED Hsiao codes [4][5]. With multi-bit upsets (MBU) becoming a major contributor to soft errors [6], conventional SEC or SEC-DED codes may not be sufficient to meet reliability goals [1]. To mitigate these effects, the use of more powerful ECC and/or memory scrubbing with conventional ECC are being suggested [9].

BCH (Bose-Chaudhuri-Hocquenghem) codes are a class of powerful random error-correcting cyclic codes [10]. Although BCH codes have been used in communication systems, they are typically not applied in high-speed memory applications due to their relatively large redundancy requirements and decoding complexity [5]. For example, Table I shows the amount of redundancy required by SEC-DED and double-error-correcting (DEC) BCH codes for typical data widths. Commonly employed iterative BCH decoding schemes such as Berlekamp-Massey, Euclidian and Minimum Weight Decoding algorithms in communication systems require a multi-cycle decoding latency [10]. Given the dependence of microprocessor performance on memory latency and bandwidth, this multi-cycle decoding latency is not tolerable for memory systems. Another impediment is the block size of primitive BCH codes, which does not align with typical memory word sizes that are usually a power of 2 [10].

In this work, we present a design of DEC BCH codes which are aligned to prevailing memory word sizes such as 16, 32 and 64 bits. Exploiting the available silicon resources in scaled technologies, a new parallel implementation approach is presented for these DEC BCH codes. As most applications access data in blocks and not bit-by-bit, reliability gains of applying the SEC and DEC ECC are quantified on a codeword basis through a rigorous probability analysis. This analysis shows that a codeword protected by DEC exhibits orders of magnitude reliability improvement compared to SEC protected codewords.

Using the parallel implementation approach, synthesis results show that it is practical to implement single-cycle DEC decoders for typical memory word sizes. In particular, DEC decoders implemented with IBM 90nm ASIC technology incur latencies of 1.4ns and 2.2ns for block sizes of 16 and 64 bits, respectively; this is a remarkable improvement compared to the multi-cycle decoding techniques mentioned earlier. As most memories are now being protected by SEC-DED codes [5], we compare the cost of our proposed DEC implementation with the cost of SEC-DED codes. For this purpose, we implemented Hsiao codes using the same technology, as Hsiao codes offer better SEC-DED implementation compared to extended-Hamming codes [4]. The implementation results presented in section V show that DEC decoders incur 55% to 69% latency penalty compared to

TABLE I. AREA PENALTY FOR ECC CODES

	SEC-DED		BCH DEC	
Data bits	**Check bits**	**% area penalty**	**Check bits**	**% area penalty**
16	6	37.5	10	62.5
32	7	21.87	12	37.5
64	8	12.5	14	21.87
128	9	7.03	16	12.5

978-1-4244-2182-4/08/$25.00 ©2008 IEEE.

SEC-DED codes. This penalty is certainly tolerable given the reliability benefits of the DEC codes. Furthermore, low-power techniques that employ low data retention voltage (DRV) for standby SRAMs in portable applications can benefit from these DEC implementations. Particularly, a large number of errors occurring due to extremely low DRV with intra-chip V_t variations [14] can be effectively corrected by DEC ECC.

Section II describes the DEC code design for typical memory word sizes and section III quantifies reliability gains. In section IV, we discuss the parallel implementation approach for DEC BCH codes. Section V presents our synthesis results for these codes implemented using IBM 90nm ASIC technology, and section VI concludes the paper.

II. DEC BCH Code Design for Memory Word Sizes

A binary (n, k) linear block code is a k-dimensional subspace of a binary n-dimensional vector space. Thus, an n-bit codeword contains k-bits of data and r (= n – k) check bits. An r x n parity check matrix H, or alternatively k x n generator matrix G, is used to describe the code [10]. Due to the cyclic property of BCH codes, a systematic generator matrix for primitive BCH codes of the form $G_{k,n} = [I_{k,k} \mid P_{k,r}]$ can be generated by combining two sub-matrices. $I_{k,k}$ is an identity matrix of dimension k, and $P_{k,r}$ is a parity sub-matrix consisting of the coefficients of k parity polynomials of degree r. The k parity polynomials can be obtained from a polynomial division involving the generator polynomial g(x) of the BCH code as in (1).

$$P_i = remainder\left(X^{n-k+i} / g(x)\right) \bmod 2 \qquad i = 0,1,\ldots k-1 \qquad (1)$$

For example, a systematic generator matrix G for a primitive DEC BCH (31, 21), shown in Fig. 1, is computed using this method. Notice that the described code derivation procedure is based on the cyclic property of BCH codes and is different from the normally used procedure in [10] and other texts. As evident from the code parameters, the information word size of 21 bits is not a common memory word size. We adopt a code shortening procedure to align it to a memory word size, in this case 16 bits. Starting with row 17 of the matrix G, the column containing the entry "1" in the identity sub-matrix section is identified and this column and respective row containing the 1 are deleted from G. In this way, a reduced generator matrix G for DEC (26, 16) is obtained, also shown in Fig. 1.

III. Reliability Gain Analysis

Before applying any ECC technique at the system level, it is imperative to quantify its reliability benefits. For simplicity of the analysis, we focus only on error correction and ignore the error detection capability of the ECC schemes considered. Abstracting all parameters causing errors to a probability ε of a bit being upset, the probability of success for that bit is 1 – ε. For a k-bit data word to be error free, all of its bits should be error free. Therefore, subtracting the probability of success from the total probability, the codeword failure probability for a word without ECC protection is given by (2):

$$P_{unprotected-CW} = 1 - (1-\varepsilon)^k \qquad (2)$$

A single-bit error correcting code results in codeword success in two cases: 1) no error has occurred and 2) any one

G (31, 26) = [
1000000000000000000001001011011
0100000000000000000001101110110
0010000000000000000000110111011
0001000000000000000001010000110
0000100000000000000000101000011
0000010000000000000001011111010
0000001000000000000000101111101
0000000100000000000001011100101
0000000010000000000001100101001
0000000001000000000001111001111
0000000000100000000001110111100
0000000000010000000000111011110
0000000000001000000000011101111
0000000000000100000001000101100
0000000000000010000000100010110
0000000000000001000000010001011
0000000000000000100001000011110
0000000000000000010000100001111
0000000000000000001001011011100
0000000000000000000100101101110
0000000000000000000010010110111
]

G (26, 16) = [
10000000000000001001011011
01000000000000001101110110
00100000000000000110111011
00010000000000001010000110
00001000000000000101000011
00000100000000001011111010
00000010000000000101111101
00000001000000001011100101
00000000100000001100101001
00000000010000001111001111
00000000001000001110111100
00000000000100000111011110
00000000000010000011101111
00000000000001001000101100
00000000000000100100010110
00000000000000010010001011
]

Figure 1. Systematic generator matrix of DEC (31, 21) & (26, 16)

bit out of n bits has encountered an error while the remaining 'n-1' bits are error free. The number of ways an occurrence of a single-bit error can happen in the second case involves combinatorial mathematics. Therefore, using binomial law, the codeword failure probability for the SEC code is bounded by (3):

$$P_{SEC-CW} = 1 - \{(1-\varepsilon)^n + \binom{n}{1}\varepsilon(1-\varepsilon)^{n-1}\} \qquad (3)$$

Following the similar argument as for SEC and the fact that DEC code results in codeword success with zero, single and double bit errors, the codeword failure probability for a word protected by DEC is bounded by (4):

$$P_{DEC-CW} = 1 - \{(1-\varepsilon)^n + \binom{n}{1}\varepsilon(1-\varepsilon)^{n-1} + \binom{n}{2}\varepsilon^2(1-\varepsilon)^{n-2}\} \qquad (4)$$

Fig. 2 shows the codeword failure probabilities for unprotected, SEC, and DEC protected words versus the raw bit error rate (BER) for 16- and 32-bit words. In the log-log graph, the slopes of the curves are linear, quadratic and cubic for unprotected, SEC and DEC case, respectively. This implies that the asymptotic reliability gain provided by DEC codes is quadratic compared to linear by the SEC. The

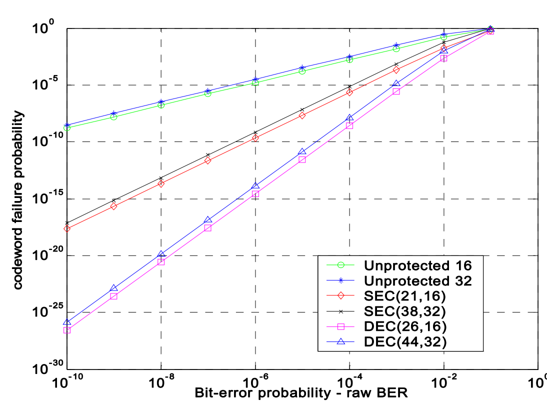

Figure 2. Codeword Failure Probabilities versus Bit Error Rate

absolute value of the gain provided by these codes depends on the prevailing raw BER. For example, for a raw BER of 1e-5 (errors/bit-day), which is common for heavy-ion induced soft errors in today's technologies [9], the DEC code provides ~4 orders of magnitude higher gain compared to SEC. Looking at the curves for different block sizes, it is observed that though there is an offset between the curves, the respective slopes stay the same, indicating that the relative asymptotic gain provided by each ECC scheme remains the same irrespective of the block size. This implies that DEC codes offer much better reliability gains compared to SEC for any block size, as expected.

IV. Implementation Approach

A. BCH Code

We have adopted a pure combinational logic approach to implement DEC BCH codes. This approach is constructed on a standard array based syndrome decoding procedure. In this procedure, a set of syndromes is pre-computed corresponding to correctable error patterns and stored in a ROM-based lookup table (LUT). The resources to store and access this LUT increase exponentially as the block size of the code increases, (which is generally the case in communication systems), thereby inhibiting the usefulness of this procedure. With relatively smaller block sizes for typical memory words, we adopted and modified this procedure to decode DEC BCH codes. Instead of using a ROM-based LUT, error correction bits are set according to a Boolean function mapping of syndrome patterns. This allows Boolean function implementation using a standard cell ASIC design methodology. In the following, we describe the encoder and decoder circuit implementation in details.

1) BCH Encoder

The encoding process converts a data word (row vector $\underline{b}$) into a codeword (row vector $\underline{c}$) by multiplying it with the generator matrix using modulo-2 arithmetic, i.e., $\underline{c} = \underline{b} * G$. With systematic generator matrix G, data bits are passed as-is in the encoding process and only the check bits need to be computed. The computation of check bits is accomplished through XOR trees as shown in Fig. 3 for DEC (26, 16). The inputs to each XOR tree are data bits chosen according to non-zero entries in respective columns of the parity sub-matrix as was shown in Fig. 1. The generated codeword is then stored in memory along with the appended check bits.

2) BCH Decoder

For decoding purposes, a parity check matrix H, of the form: $H_{r,k} = [P^t_{r,k} \mid I_{r,r}]$, is required where P^t is the transpose of the parity sub-matrix in systematic G, and I is the identity matrix. The input to the decoder is the read codeword vector $\underline{v}$ which may contain errors in data or check bit locations. A syndrome $\underline{s}$ is computed, using modulo-2 arithmetic, by multiplying H with $\underline{v}^t$ (transpose of the read codeword $\underline{v}$), i.e., $\underline{s} = H * \underline{v}^t$. A non-zero syndrome implies the presence of errors, in which case corresponding bits in the error location vector $\underline{e}$ are set by the error pattern decoder. The error location vector $\underline{e}$ is added with the received codeword $\underline{v}$ to get the corrected data. The error pattern decoder and error corrector circuits can be optimized to correct errors only in data bit locations.

A block diagram of the decoder is shown in Fig. 4 containing the three main parts: 1) Syndrome Generator, 2) Error Pattern Decoder and 3) Error Corrector. The circuit for the syndrome generator is similar to the encoder circuit. Essentially, it re-computes the check bits and compares those with the received check bits. In case of no error, a zero syndrome is generated and can be used to alert the error flag. Alternatively, a non-zero syndrome is generated if the computed check bits are not matched to the received check bits. An error pattern decoder circuit is implemented using combinational logic that maps the syndromes for correctable error patterns. This mapping is pre-computed by multiplying all correctable error patterns with the parity check matrix H. For binary vectors, an erroneous bit is corrected merely by complementing it; therefore, the error corrector circuit is simply a stack of XOR gates.

B. Hsiao Code

Hsiao codes offer an optimal implementation for SEC-DED codes compared to Hamming codes [4]. The implementation architectures for Hsiao codes are well understood [4], and therefore not repeated here.

V. Results and Discussion

The major area overhead for any ECC scheme is due to the redundancy required for its operation and is a function of the error detection and correction capability of the code and block size. With increasing block size, the required redundancy overhead decreases for all linear block codes. It can be observed from Table I given in Section I, that DEC BCH codes incur approximately 70% more overhead for check bits as compared to SEC-DED. Therefore, here we restrict our discussion to the implementation of encoder and decoder circuits only.

For analyzing latency and area trade-offs, we have implemented DEC BCH codes for typical memory word sizes of 16, 32 and 64 bits, along with their SEC-DED Hsiao counterparts. Synopsys Design Compiler (DC) has been used for synthesizing all encoder and decoder circuits targeted to an

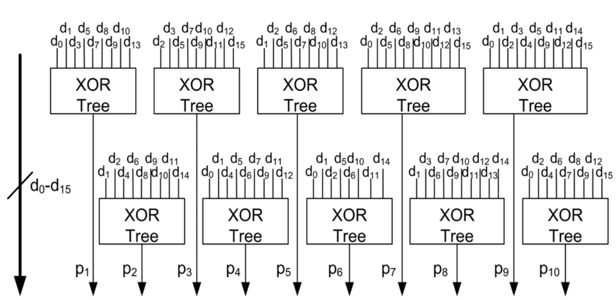

Figure 3. Encoder Circuit for DEC (26, 16)

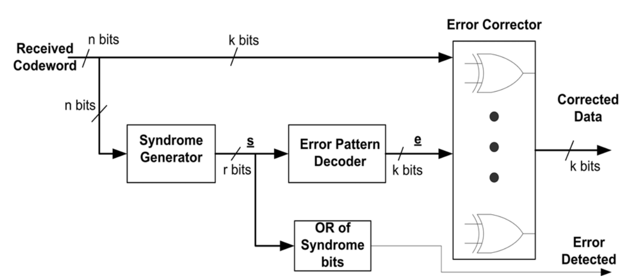

Figure 4. Block Diagrag of BCH Decoder

IBM 90nm standard cell ASIC library. Table II shows the latency and area results for post-synthesis encoder circuits while Table III shows the latency and area results for decoder circuits.

A major inference from the synthesis results is that the decoding latency for the DEC codes is reasonably small, and it is much better compared to multi-cycle LFSR based decoders used in communication systems. Therefore, this parallel implementation of the DEC codes makes it feasible to utilize DEC ECC for memory applications. Another observation is that the encoding latencies both for Hsiao and DEC codes are approximately identical. Since the syndrome generator circuit is similar to the encoder circuit (as described in the previous section), error detection can be accomplished with quite similar latency for both SEC-DED and DEC codes. The ECC implementation architecture can remarkably benefit from this observation. In particular, as most memory accesses would be error-free, data can be passed for processing to the next stage without the full decoder delay. In erroneous cases, when a non-zero syndrome is detected, only then is the full decoder latency needed to correct the errors. For these cases only, the next processing stage can be stalled for a cycle or two depending on the speed of the processing stage, incurring a minimal overall performance penalty.

As most memory organizations already employ SEC-DED ECC, a comparative analysis can be made from Table II and III. The percentage latency penalty for DEC ECC varies between 55% and 69% for different block sizes. Notice that the full decoder latency is incurred only for the erroneous cases. Although the area occupied by DEC decoders is comparatively larger than that of SEC-DED decoders (3.5X and 13X for 16 and 64-bit, respectively), this does not become a dominant factor with available silicon resources in current technologies. In fact, the full ECC encoder and decoder circuits can be built within the same chip as memory. As derived in section III, the reliability gain provided by the DEC BCH code is four orders of magnitude superior to Hsiao codes. Therefore applications requiring high reliability, especially against soft errors or low-power standby SRAMs with extremely low DRV, can use DEC codes with relatively reasonable performance overheads.

VI. Conclusion

A DEC BCH code design that is aligned to typical memory word sizes has been presented. A parallel approach for implementing these DEC BCH codes for memory applications is described. This approach enables a simple single-cycle implementation of DEC decoders rather than iterative multi-cycle decoding used in communication systems. ECC encoder and decoder circuits have been synthesized using standard cell IBM 90nm technology for typical memory word sizes. Though the DEC decoders incur more error-correcting latency, compared to optimized SEC-DED codes (55% for 16-bit and 69% for 64-bit word), a careful ECC implementation architecture can minimize the effects of this penalty on overall system performance. In particular, data can be forwarded to following stages after approximately equal error detection latency for both SEC-DED and DEC codes, incurring the full error-correcting decoder latency only in erroneous cases. The DEC code offers four orders of magnitude better reliability compared to conventional SEC-DED codes for typical soft error rates. Therefore, it is especially helpful for memories severely affected by increased error rates in scaled technologies, either due to soft errors or errors occurring due to process variations. Furthermore, low-power techniques employing extremely low data retention voltages during standby modes can benefit from these DEC codes.

TABLE II. Encoder Latency and Area Results

Data Bits	SEC-DED Hsiao Code		DEC BCH Code	
	Latency (ns)	Area (μm^2)	Latency (ns)	Area (μm^2)
16	0.4	290	0.5	495
32	0.5	604	0.6	1250
64	0.7	1167	0.7	2335

TABLE III. Decoder Latency and Area Results

Data Bits	SEC-DED Hsiao Code		DEC BCH Code	
	Latency (ns)	Area (μm^2)	Latency (ns)	Area (μm^2)
16	0.9	935	1.4	4288
32	1.1	1376	1.8	11734
64	1.3	2681	2.2	37279

References

[1] R.C. Baumann, "The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction", IEEE IEDM 2002, p. 329-332.

[2] A. Agarwal, B.C. Paul, S. Mukhopadhyay, K. Roy,"Process variation in embedded memories: failure analysis and variation aware architecture", IEEE Journal of Solid-State Circuits Vol 40, Sep. 2005 pp.1804 - 1814

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

[4] M.Y. Hsiao, "A Class of Optimal Minimum Odd-weight-column SEC-DED Codes", IBM Journal of R & D Vol. 14, July 1970, pp. 395-401

[5] C.W. Slayman, "Cache and memory error detection, correction, and reduction techniques for terrestrial servers and workstations", IEEE Trans. Device & Materials Reliability Vol. 5, Sept. 2005 pp. 397 – 404

[6] D. Radaelli, H. Puchner, S. Wong, S. Daniel, "Investigation of multi-bit upsets in a 150 nm technology SRAM device", IEEE Trans. On Nucl. Sci. Vol. 52, Issue 6, pp. 2433- 2437, Dec 2005

[7] F.X. Ruckerbauer, G. Georgakos, "Soft Error Rates in 65nm SRAMs- Analysis of new Phenomena", 13th IEEE Intern'l On-Line Testing Symposium, IOLTS 07. July 2007 pp.203 – 204

[8] D. Giot, P. Roche, G. Gasiot, R. Harboe-Sorensen, "Multiple-Bit Upset Analysis in 90 nm SRAMs: Heavy Ions Testing and 3D Simulations", IEEE Trans. Nucl. Sci., Vol. 54, Aug. 2007 pp.904 – 911

[9] M. Bajura et al, "Models and Algorithmic Limits for an ECC-based Approach to Harden Sub-100-nm SRAMs", IEEE Trans. Nucl. Sci. Vol. 54, no. 4, pp. 935-945 Aug 2007

[10] S. Lin, D.J. Costello, "Error Control Coding: Fundamental and Applications", Prentice-Hall, 1983

[11] Shyue-Win Wei, Che-Ho Wei, "High-speed hardware decoder for double-error-correcting binary BCH codes", Communications, Speech and Vision, IEE Proceedings Jun 1989 Vol. 136, Issue 3, pp. 227- 231

[12] E.-H. Lu1 and T. Chang, "New decoder for double-error-correcting binary BCH codes", IEE Communications Proc. June 1996, Vol. 143, Issue 3, p. 129-132

[13] W.M. El-Medany, C.G. Harrison, P.G. Farrel, and C.J. Hardy, "VHDL Implementation of a BCH Minimum Weight Decoder for Double Error", Radio Science Conf., Proc. of the 18th 2001 Vol. 2, pp. 361-368

[14] A. Kumar, H. Qin, P. Ishwar, J. Rabaey, and K. Ramchandran, "Fundamental Bounds on Power Reduction during Data-Retention in Standby SRAM", IEEE ISCAS 2007, pp. 1867-1870.