International Journal of Scientific & Engineering Research, Volume 7, Issue 4, April-2016 294
ISSN 2229-5518

A Class of SEC-DED-DAEC Codes Derived From OLS Codes and Decoded With Low Latency

Magesh. E

Abstract—Radiation-induced soft errors are a major reliability concern for memories. To ensure that memory contents are not corrupted, single error correction double error detection (SEC-DED) codes are commonly used, however, in advanced technology nodes, soft errors frequently affect more than one memory bit. Since SEC-DED codes cannot correct multiple errors, they are often combined with interleaving. Interleaving, however, impacts memory design and performance and cannot always be used in small memories. This limitation has spurred interest in codes that can correct adjacent bit errors. In particular, several SEC-DED double adjacent error correction (SEC-DED-DAEC) codes have recently been proposed. Implementing DAEC has a cost as it impacts the decoder complexity and delay. Another issue is that most of the new SEC-DED-DAEC codes miscorrect some double nonadjacent bit errors. In this brief, a new class of SEC-DED-DAEC codes is derived from orthogonal latin squares codes. The new codes significantly reduce the decoding complexity and delay. In addition, the codes do not miscorrect any double nonadjacent bit errors. The main disadvantage of the new codes is that they require a larger number of parity check bits. Therefore, they can be useful when decoding delay or complexity is critical or when miscorrection of double nonadjacent bit errors is not acceptable. The proposed codes have been implemented in Hardware Description Language and compared with some of the existing SEC-DED-DAEC codes. The results confirm the reduction in decoder delay.

Index Terms— Double adjacent error correction (DAEC), error correction codes, memory, orthogonal latin squares (OLS), single error correction double error detection (SEC-DED).

1 INTRODUCTION

Traditionally, SEC-DED codes have been used. A SEC-DED code has a minimum Hamming distance of four and is able to correct single bit errors and detect double errors without miscorrection. This is important to avoid silent data corruption. SEC-DED codes are sufficient when errors affect only one bit, however, the percentage of soft errors affecting more than a single bit is increasing as technology scales. For memories implemented in 40 nm and below, multiple bit errors are a significant percentage of errors and thus SEC-DED codes alone are no longer sufficient to protect memories. Interleaving, places the bits that belong to the same logical word physically apart. As the errors caused by a radiation particle hit are physically close, this ensures that the errors affect at most one bit per logical word. Interleaving has an impact on the memory design. The routing is more complex and area and power consumption are increased. In addition, interleaving cannot always be used in small memories or register files nor can be practically applied to content addressable memories. Another alternative is to use error correction codes that can correct adjacent bits. In many cases, directly adjacent bits account for over 90% of the observed multiple bit errors. Several codes have been recently proposed to this end. For example, a code that can correct double and triple adjacent errors for words of 16 bit was presented in. In, a technique to design SEC-DED double adjacent error correction (SEC-DED-DAEC) codes was introduced.

The extension of SEC-DED-DAEC codes to also detect larger burst errors has also been recently considered in. One issue with those SEC-DED-DAEC codes is that they can miscorrect some double nonadjacent bit errors. The reduction of the miscorrection probability has been considered. In the algorithm tries to minimize the number of 4 cycles. In it was shown that miscorrection can be avoided for the most common error patterns and in some cases for all patterns at the cost of adding additional parity check bits. Another issue with SEC-DED-DAEC codes is that their decoding complexity and latency are larger than those of SEC-DED codes. This limits their use when speed is a critical factor. The main limitation for these codes is that they require a number of parity check bits equal to the number of data bits. The use of more advanced codes such as difference set and orthogonal latin squares (OLS) codes to correct adjacent errors has also been considered. Those codes are one-step majority logic decodable (OS-MLD) and therefore, can be decoded with low latency. They also support the correction of multiple nonadjacent bit errors, a

MAGESH. E is currently pursuing master degree program in Applied Electronics in Anna University, Chennai, India, PH-09489958255. E-mail: magesh.e001@mail.com

ISSN 2229-5518

protection level that may be excessive in some memory applications.

In this brief, a new class of SEC-DED-DAEC codes is presented. The proposed codes are derived from OLS codes. They require fewer parity check bits than double error correction (DEC) OLS codes and are simpler to decode. Compared with existing SEC-DED-DAEC codes, the new codes have two main advantages: first, there are no miscorrections for double nonadjacent errors and second, the decoding is much simpler and faster. The main drawback for the proposed codes is that they require more parity check bits than existing SEC-DED-DAEC codes. Therefore, the proposed code is critical or miscorrections cannot be tolerated.

2 OLS CODES

OLS codes were introduced decades ago to protect memories and have recently been proposed to protect caches and interconnect.

The block sizes for OLS codes are $k = m^2$ data bits and $2tm$ parity bits. Where t is the number of errors that the code can correct and m is an integer. For memories, the word sizes are typically a power of two and therefore m is commonly also power of two. The main advantages of OLS codes are that their decoding is simple and fast. This is because, as mentioned in the introduction, OLS codes can be decoded using OS-MLD. In OS-MLD, each bit is decoded by simply taking the majority value on the set of the recomputed parity check equations in which it participates. This is shown in Fig. 1 for a given data bit *di*. The idea behind OS-MLD is that when an error occurs in bit *di*, the recomputed parity checks in which it participates will take a value of one unless there are errors in other bits.

Therefore, a majority of ones in those recomputed checks is an indication that the bit is in error and therefore needs to be corrected. If the code is such that two bits share at most one parity check, then $t-1$ errors on other bits will not affect the majority of the $2t$ vote and therefore, the error will be corrected. Only a few codes have this property and can be decoded using OS-MLD. This is the case for difference set codes and for OLS codes, as mentioned in the introduction.

More formally, the construction of OLS codes is such that:

1) each data bit participates in exactly $2t$ parity check bits;
2) each other data bit participates in at most one of those parity check bits.

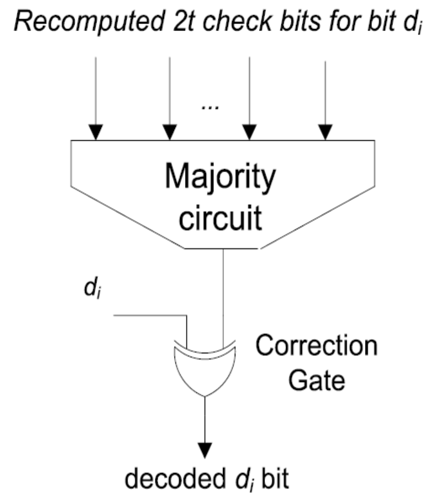

Fig.1. Illustration of OS – MLD decoding for OLS codes.

Therefore, for a number of errors t or smaller, when one error affects a given bit, the remaining $t - 1$ errors can, in the worst case affect $t - 1$ check bits on which that bit participates. Therefore, still a majority of $t + 1$ will trigger the correction on the erroneous bit. Conversely, when a given bit is correct, t errors on other bits will not cause miscorrection as a majority of $t + 1$ is needed. As shown in Fig. 1, the use of OS-MLD enables a simple and fast decoding that is attractive to protect memories when decoding latency is critical.

As mentioned in the introduction, the proposed codes are derived from DEC OLS codes. These are block linear codes that are defined by their parity generating G and parity check H matrixes. The parity check matrix is used to detect errors by computing the syndrome s that is obtained by multiplying the stored word by the H matrix. The parity check matrix H for a DEC OLS code with $k = m^2$ is constructed as follows:

$$H = \begin{bmatrix} M_1 & \\ M_2 & \\ M_3 & I_{4m} \\ M_4 & \end{bmatrix}$$

where I_{4m} is the identity matrix of size $4m$ and M_1, M_2, M_3, M_4 are matrices with size $m \times m^2$ derived from OLS of size $m \times m$. The weight or the number of ones, of all the columns, in the M_i matrices must be one. Therefore, the first $k = m^2$ columns in H have a number of ones equal to $2t$ (four for DEC codes). In addition, any pair of columns has at most a position with a one in common.

International Journal of Scientific & Engineering Research, Volume 7, Issue 4, April-2016 296
ISSN 2229-5518

$$
\begin{array}{c|cccccccccccccccc|cccccccccccccccc|c}
 & 1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0 & 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 & \\
M_1 & 0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0 & 0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 & \\
 & 0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0 & 0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0 & \\
 & 0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1 & 0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0 & \\
\hline
 & 1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0 & 0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0 & \\
M_2 & 0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0 & 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0 & \\
 & 0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0 & 0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0 & \\
 & 0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1 & 0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0 & I_{4m} \\
\hline
 & 1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1 & 0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0 & \\
M_3 & 0&1&0&0&1&0&0&0&0&0&0&1&0&0&1&0 & 0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0 & \\
 & 0&0&1&0&0&0&0&1&1&0&0&0&0&1&0&0 & 0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0 & \\
 & 0&0&0&1&0&0&1&0&0&1&0&0&1&0&0&0 & 0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0 & \\
\hline
 & 1&0&0&0&0&0&1&0&0&0&0&1&0&1&0&0 & 0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0 & \\
M_4 & 0&1&0&0&0&0&0&1&0&0&1&0&1&0&0&0 & 0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0 & \\
 & 0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&1 & 0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0 & \\
 & 0&0&0&1&0&1&0&0&1&0&0&0&0&0&1&0 & 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1 & \\
\end{array}
$$

Fig.2. Parity check matrix H for the OLS code with K = 16 and t = 2.

As an example, the H matrix for a code with $k = m^2 = 16$ data bits and $2tm = 16$ parity bits that can correct double errors is shown in Fig. 2.

3 SEC – DED – DAEC CODES

The proposed SEC – DED – DAEC codes are derived from DEC OLS codes. Taking the parity check matrix as a starting point, the first step is to remove the m parity check bits that correspond to one of the *Mi* matrices.

As an example, consider removing the M_1 matrix from the matrix in Fig. 2 as shown in Fig. 3. The data bits that participated in each of the removed parity check equations will not share any parity check in the reduced matrix. This is a direct consequence from the property of OLS codes that any two data bits share at most one parity check bit. This can be clearly observed in Fig. 2. In addition, those groups of m bits are marked as g_1, g_2, g_3, and g_4 in Fig. 3. For example, the first four data bits share the first parity check bit in the M_1 matrix and form the first group g_1. It can be observed that they do not share any other parity check bits. Therefore, when M_1 is removed they do not share any parity check bit. The same occurs for the other groups of bits 5–8 (g_2), 9–12 (g_3), and 13–16 (g_4). In the reduced matrix, each data bit participates in three parity checks. Therefore, if a majority vote is used to decode the bits; single and double errors can be corrected.

However, double errors can also cause miscorrections on other bits. Therefore, the modified matrix, when a majority vote is used, is only effective in correcting single errors. However, let us consider that instead of a majority vote, the logical AND of the three parity checks is used. In Fig. 4, this is shown for the first two data bits where the s_i values correspond to bits of the syndrome vector obtained by multi-plying the word by the H matrix. Single errors on data bits will also be corrected. Double errors affecting data bits will also be corrected as long as the data bits do not share any parity check bit. The two modifications can now be linked together by noting that errors that affect bits in one group of bits that share a parity check bit in M_1 will now be corrected. For example, an adjacent error in bits 1 and 2 will cause the recomputed parity checks 1, 2, 5, 6, 9, and 10 to give a value of one. The ones on parity checks 1, 5, and 9 will trigger a correction on bit 1 while the ones on parity checks 2, 6, and 10 will trigger a correction on bit 2. This is clearly observed in Fig. 4. In this case, the recomputed parity checks are denoted as *s*i to make clear that they are in fact bits from the syndrome. However, some double adjacent errors may affect bits on different groups.

For example, an error on bits 8 and 9 affects a bit in g_2 and another in g_3. These bits share parity check bit 7 and therefore, will not be corrected as that recomputed parity bit will take a value of zero in the syndrome as it has two bits in error. This effect can be avoided by carefully placing the bits in the memory. For example, the bits within each group can be reordered to ensure that the ones at the borders does not share any parity check bit with the adjacent bit on the other group. Another issue that can occur is that a double adjacent error affects two parity bits and the error is confused with a double nonadjacent error. For example, an error on parity check bits 4 and 5 produces the same syndrome as an error that affects data bit 16 and parity check bit 11. This can lead to silent data corruption leaving an error on data bit 16 undetected. However, this issue can also be solved by carefully placing the bits into the memory.

ISSN 2229-5518

$$
\begin{array}{c|cccccccccccccccc|cccccccccccc|c}
 & & g_1 & & & & g_2 & & & & g_3 & & & & g_4 & & & & & & & & & & & & & & & \\
\hline
 & 1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0 & 1&0&0&0&0&0&0&0&0&0&0&0 & \\
M_2 & 0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0 & 0&1&0&0&0&0&0&0&0&0&0&0 & \\
 & 0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0 & 0&0&1&0&0&0&0&0&0&0&0&0 & \\
 & 0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1 & 0&0&0&1&0&0&0&0&0&0&0&0 & \\
\hline
 & 1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1 & 0&0&0&0&1&0&0&0&0&0&0&0 & \\
M_3 & 0&1&0&0&1&0&0&0&0&0&0&1&0&0&1&0 & 0&0&0&0&0&1&0&0&0&0&0&0 & I_{3m} \\
 & 0&0&1&0&0&0&0&1&1&0&0&0&0&1&0&0 & 0&0&0&0&0&0&1&0&0&0&0&0 & \\
 & 0&0&0&1&0&0&1&0&0&1&0&0&1&0&0&0 & 0&0&0&0&0&0&0&1&0&0&0&0 & \\
\hline
 & 1&0&0&0&0&0&1&0&0&0&0&1&0&1&0&0 & 0&0&0&0&0&0&0&0&1&0&0&0 & \\
M_4 & 0&1&0&0&0&0&0&1&0&0&1&0&1&0&0&0 & 0&0&0&0&0&0&0&0&0&1&0&0 & \\
 & 0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&1 & 0&0&0&0&0&0&0&0&0&0&1&0 & \\
 & 0&0&0&1&0&1&0&0&1&0&0&0&0&0&1&0 & 0&0&0&0&0&0&0&0&0&0&0&1 & \\
\hline
\end{array}
$$

Fig.3. Reduced parity check matrix H after removal of M_1.

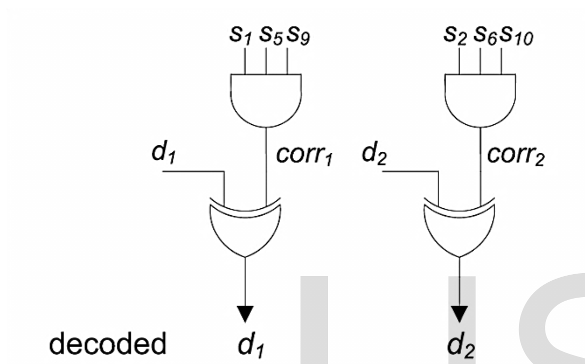

Fig.4. Illustration of the SEC – DED - DAEC decoder for data bits 1 and 2.

The proposed bit placement is as follows:

1) Ensure that the bits at the borders of the groups do not share any parity check bits and
2) Interleave the parity check bits with the data bits so that no double adjacent error affects two parity bits.

An example of this bit placement for the code with k = 16 is shown in Fig.5. The parity bits are marked in the figure and obviously, they can only be placed such that the adjacent columns do not participate in the parity bit. With this bit placement, all double adjacent errors affect at least a data bit and that data bit is corrected.

In addition, for nonadjacent errors that affect two bits, if any bit is corrected it means that the error is correctable. When the error affects two data bits, either they are both corrected or there is no correction. This enables a simple method to detect uncorrectable errors. The proposed scheme to detect the uncorrectable errors is shown in Fig. 6. It is based on detecting a nonzero even number of ones in the syndrome that can only be caused by a multiple bit error and checking if any correction has been made.

p_7 p_9 p_1 p_5 p_4 p_{11} p_{12} p_8 p_{10} p_6 p_2 p_3

$$
\begin{array}{cccccccccccccccccccccccccccc}
1&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0\\
0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&1&0&0&0&1&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&1&0&0&1&0\\
0&0&0&0&0&0&1&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&1\\
1&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1\\
0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&1&0\\
0&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&1&0&0&0&0&0&1&0&0&0\\
1&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0&1&0&0&0&1&0&0&0\\
0&0&0&0&1&0&0&0&1&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&1&0&0&0&1&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&1&0
\end{array}
$$

Fig.5. Reduced parity check matrix H after the removal of M1 with the proposed bit placement.

The proposed scheme can be summarized as follows:

1) Reduce H matrix of the DEC OLS code by eliminating M1;
2) Place the bits in the groups of m bits g1, g2,..., gm such that the bits at the borders of the groups do not share any parity check;
3) Interleave the parity bits with the data bits such that two adjacent bits never participate in the same parity bit;
4) Instead of majority voting, decode based on unanimity to correct errors;
5) Implement the circuit of Fig.6 to detect uncorrectable errors.

This scheme can correct all double adjacent errors that affect data bits and detect all non-correctable double errors. Therefore, the derived codes are SEC-DED-DAEC with no miscorrection. In addition, some double nonadjacent errors are also corrected and the fraction of these errors that can be corrected grows with the block size. The parameters of the derived codes for the block sizes that are commonly of interest for memory protection. It can be observed that the number of required parity check bits is significantly higher than traditional SEC-DED-DAEC codes but this is the price to pay for faster decoding and the absence of miscorrections. For example, for k = 64 a SEC-DED-DAEC with no miscorrection required 12 bit compared with the 24 of the proposed codes. In that case, the main benefit of the new codes is the simple and fast decoding.

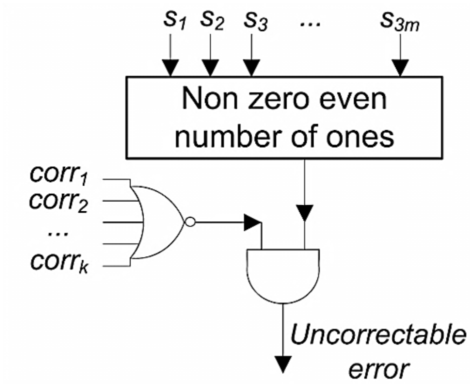

Fig.6. Detection of double uncorrectable errors in the proposed scheme.

4 EVALUATION

The proposed SEC-DED- DAEC extended codes have been implemented in MATLAB where their error correction capabilities were validated for single and double adjacent errors. As the number of combinations of single and double adjacent errors is small ($2n - 1$), these were tested exhaustively. For the nonadjacent double errors, 100 000 combinations were randomly generated and tested to ensure that the errors were corrected or detected as uncorrectable. The results confirm the theoretical analysis in that the codes are SEC-DED-DAEC with no miscorrection.

The encoders and decoders have also been implemented in Hardware Description Language (HDL). The synthesizer is configured to optimize the delay. Therefore, the results provide the lowest delay that can be achieved. The reported circuit area could be reduced at the expense of increasing the delay.

The area and delay results only for the encoders and decoders are presented in Tables 1 and 2. As expected, the decoders for the proposed codes are simpler and faster than those of existing SEC-DED-DAEC codes. In particular, the SEC-DED-DAEC codes for $k = 16$ and for $k = 64$ that avoid miscorrections for double nonadjacent errors that are separated up to a distance of five are used for comparison. The results show that the decoder area is less than one half of that required by the codes in and the delay is also greatly reduced (45% and 50% for $k = 16$ and $k = 64$, respectively). The reduction in the encoder delay is also significant: 12% and 24%, respectively. The results confirm that the proposed codes are significantly faster than existing SEC-DED-DAEC alternatives making them attractive for high-speed memories like caches. They also avoid miscorrections for double nonadjacent errors. The price to pay is that the number of parity check bits needed ($n - k$) is significantly larger than for existing SEC-DED-DAEC codes.

TABLE 1

AREA ESTIMATES (IN μM^2)

	Proposed codes			SEC-DED-DAEC		
k	*n-k*	Encoder	Decoder	*n-k*	Encoder	Decoder
16	12	158	457	7	190	1,098
64	24	831	1,976	9	805	4,369
256	48	3,687	6,927	-	-	-

TABLE 2

DELAY ESTIMATES (IN NANOSECONDS)

	Proposed codes			SEC-DED-DAEC		
k	*n-k*	Encoder	Decoder	*n-k*	Encoder	Decoder
16	12	0.22	0.25	7	0.25	0.47
64	24	0.25	0.34	9	0.33	0.69
256	48	0.28	0.45	-	-	-

5 CONCLUSION

In this brief, a new class of SEC-DED-DAEC codes has been presented. The codes are derived from DEC OLS codes and can be decoded with low latency. Another interesting feature is that the codes do not experience miscorrections when double nonadjacent error occurs. This is interesting to minimize silent data corruption. The codes can also correct some nonadjacent double errors. Compared with existing SEC-DED-DAEC codes, they require a larger number of parity check bits; therefore, they are attractive when low latency decoding is a required. The codes have been implemented in HDL and the resulting implementations compared with existing SEC-DED-DAEC codes to put the reductions in decoding latency in perspective.

The ideas used to derive the proposed SEC-DED-DAEC can also be used to derive burst error correction codes from OLS codes that can correct multiple errors. The key observation is that the structure of OLS codes is such that the data bits can be divided in groups of m bits that do not share any parity check. Therefore, any error affecting up to $2t - 1$ bits in one of these groups can be corrected. This can be exploited by carefully placing the data and parity

ISSN 2229-5518
check bits so that, in the best case, up to $2t - 1$ adjacent bit errors can be corrected.

References

[1] A. Neale and M. Sachdev, "A new SEC-DED error correction code subclass for adjacent MBU tolerance in embedded memory," IEEE Trans. Device Mater. Rel., vol. 13, no. 1, pp. 223–230, Mar. 2013.

[2] A. Dutta, "Low cost adjacent double error correcting code with complete elimination of miscorrection within a dispersion window for multiple bit upset tolerant memory," in Proc. IEEE/IFIP 20th Int. Conf. VLSI SoC, 2012, pp. 287–290.

[3] P. Reviriego, S. Pontarelli, J. A. Maestro and M. Ottavi, "Low- cost single error correction multiple adjacent error correction codes," Electron. Lett., vol. 48, no. 23, pp. 1470–1472, Nov. 2012.

[4] S. Baeg, S. Wen, and R. Wong, "Minimizing soft errors in TCAM devices: A probabilistic approach to determining scrubbing intervals," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 57, no. 4, pp. 814–822, Apr. 2010.

[5] X. She, N. Li, and D. W. Jensen, "SEU tolerant memory using error correction code," IEEE Trans. Nucl. Sci., vol. 59, no. 1, pp. 205–210, Feb. 2012.

[6] P. Reviriego, J. A. Maestro, S. Baeg, S. Wen, and R. Wong, "Protection of memories suffering MCUs through the selection of the optimal inter- leaving distance," IEEE Trans. Nucl. Sci., vol. 57, no. 4, pp. 2124–2128, Aug. 2010.

[7] E. Ibe, H. Taniguchi, Y. Yahagi, K. Shimbo, and T. Toba, "Impact of scaling on neutron-induced soft error in SRAMs from a 250 nm to a 22 nm design rule," IEEE Trans. Electron Devices, vol. 57, no. 7, pp. 1527–1538, Jul. 2010.

[8] R. C. Baumann, "Soft errors in advanced computer systems," IEEE Des.Test. Comput., vol. 22, no. 3, pp. 258–266, May/Jun. 2005.