A DOUBLE ERROR CORRECTION CODES WITH LESS NUMBER OF PARITY BITS

[1]Reddi sekhar, [2]T.Rajasree, [3]V.Hemalatha,[4]R.Ashok kumar,[5]G.Mahesh Naidu

[1]Assistant professor,[2]Student ,[3]Student, [4]Student,[5]Student
[1]Electronics and Communication engineering,
[1]Aditya college of engineering, Madanapalle,India

ABSTRACT: Present we are designing Double Error Correction (DEC) codes for 32-bit data words that support fast decoding as they can be useful to protect memories. We are used previously Orthogonal Latin Square codes have been recently presented that achieve fast decoding but require a large number of parity check bits. In this present work, a DEC code derived from Difference Set codes is presented. The proposed code is able to reduce the number of parity check bits needed at the cost of a slightly more complex decoding. Therefore, it provides memory designers with an additional option that can be useful between memory size and speed.

Keywords: Error correction codes, memories, Difference Set codes, Orthogonal Latin Square codes.

I. INTRODUCTION

The System on Chip (SoCs) and FPGAs therefore their protection is critical to ensure system reliability. A single error correction code is effective in increasing system reliability. For memory systems in need of increased reliability, a single error correcting and double error detecting(SEC-DED)code can be incorporated. The ECCs used to protect SRAM memories have focused on providing single error correction and double error detection (SEC-DED).To correct multiple bit errors, more advanced ECCS are needed.

Error correction codes (ECCS) are commonly used to protect memories so that errors do not affect the data they store. Traditionally, single error correction-double error detection (SEC-DED) codes have been used ,but as technologies scales there is a need for more powerful error correction capabilities, For example Double Error correction (DEC) Codes. This poses a problem as parallel decoders for traditional DEC codes are much more complex and result in a significant increase in area, power and delay compare to the SEC-DED codes. This has motivated the use of alternative DEC codes that can be efficiently decoded using one step majority logic voting, such as Different Set (DS) or Orthogonal Latin Square (OLS) codes. The main limitations of these codes are that only one block length is supported in the case of DS codes while DEC OLS codes require significantly more parity check bits. Arranging the data into a matrix form at the logical level and then adding several simple ECCs in different directions can also provide a powerful error correction capability for memories. For example Horizontal Vertical Diagonal (HVD) codes and Matrix codes have been proposed. These codes can be decoded with limited complexity but unfortunately have a large number of parity check bits that increase the memory size.

One of the most common data memory widths is 32-bit. Unfortunately, there is no DEC DS or DLS code with that data block size. This has motivated the development of DLS based solutions that provide fast decoding but require a significant number of parity check bits. In this letter, a DEC solution for 32 data bit words based on DS codes combined with SEC-DED codes is proposed and evaluated. The results shows that the scheme can reduce the number of parity check bit at the expense of a moderate increase the decoder complexity

I. EXISTING SYSTEM

ORTHOGONAL LATIN SQUARES (OLS) CODES:

The Latin squares and their applications are well known. A Latin square of size m is an m * m matrix that has permutations of the digits 0, 1,… and m-1 in both its rows and columns. There can be more than one Latin square for each value of m. In that case, two latin squares are said to be orthogonal if every ordered pair of elements appears only once when they are superimposed. Orthogonal Latin Squares (OLS) codes are derived from Orthogonal Latin squares. These codes have k=m2 data bits and 2tm check bits where 't' is the number of errors that the codes can correct. OLS codes can be decoded using OS-MLD. OS-MLD is a simple procedure in which each bit is decoded by simply taking the majority value of the set of the recomputed parity check equations, in which it participates.

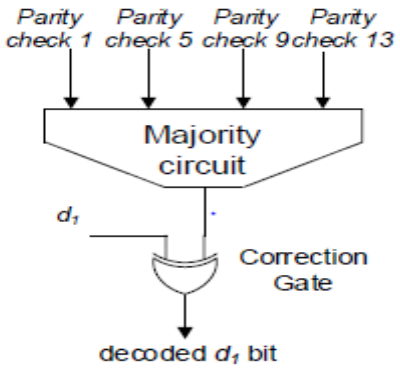


Figure **1:**OS-MLD decoding for the first bit of the DEC OLS code

The selection of the error correction capability for a given word size. As mentioned in the introduction, OLS codes can be decoded using One Step Majority Logic Decoding (OS-MLD) as each data bit participates in exactly 2t check bits and each other bit participates in at most one of those check bits. This enables a simple correction when the number of bits in error isn't or less. The 2t check bits are recomputed and a majority vote is taken, if a value of one is obtained, that bit is error and must be corrected.

Encoder:

The OLS encoder circuit uses XOR logic gates for which the output is only when either X is equal to 1 or Y is equal to 1, but not when both X and Y are equal to 1. The encoder circuit has number of bit input and number of bits bit output. The basic OLS encoder circuit uses 4 input EX-OR gate.

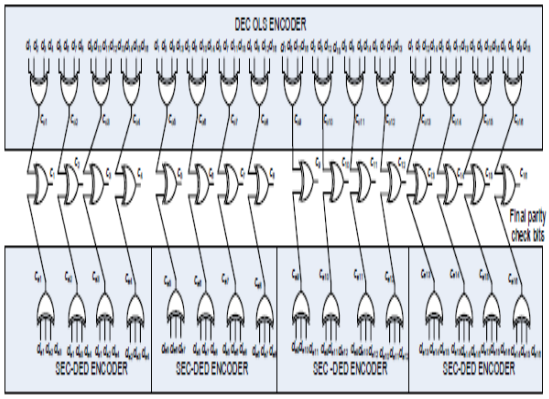


Figure 2 : encoder block

The OLS code is then selected so that it satisfies Condition 2 where the number of bits that the OLS code can correct is equal to the maximum number of erasures in any line plus the number of transient errors that are to be tolerated. Let cfullbe the number of check bits for the selected OLS code.

The configuration bit is a '1' if that check bit is to be used,and is a '0' if that check bit is to be disabled. The configuration bits can either be stored with fuses if they are to be one-time programmed at manufacture time, or they can be stored in flip-flops if they are to be programmed by software each time the chip.

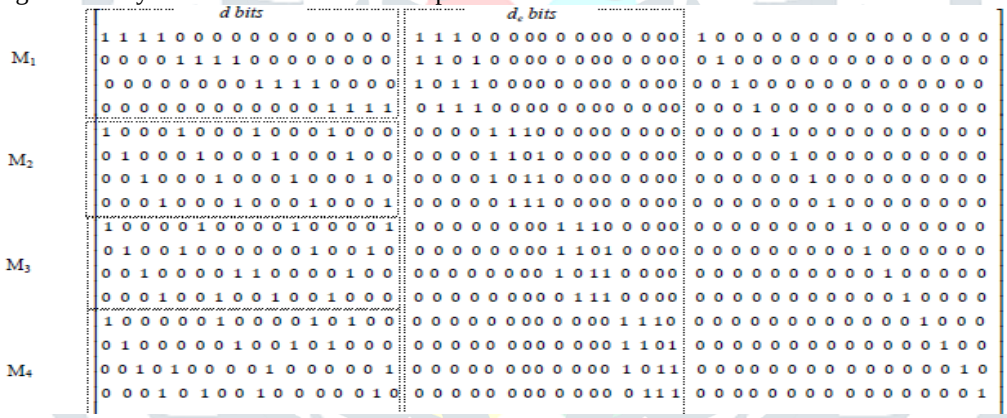


Figure 3: Parity check matrix *H* for the single sub-block SEC-DED extended OLS code

Decoder:

The first one is vote to correct another one is correct by voting. The EX-OR logic circuit used for decoder circuit and input to parity check bit added. The EX-OR gate output is majority circuit input. The OLS code can be decoded using OS-MLD. This makes the decoding process simple and fast. In OS-MLD decoded value of each bit is obtained by taking the majority value of the recomputed parity check equations (syndrome bits) in which that bit participates.

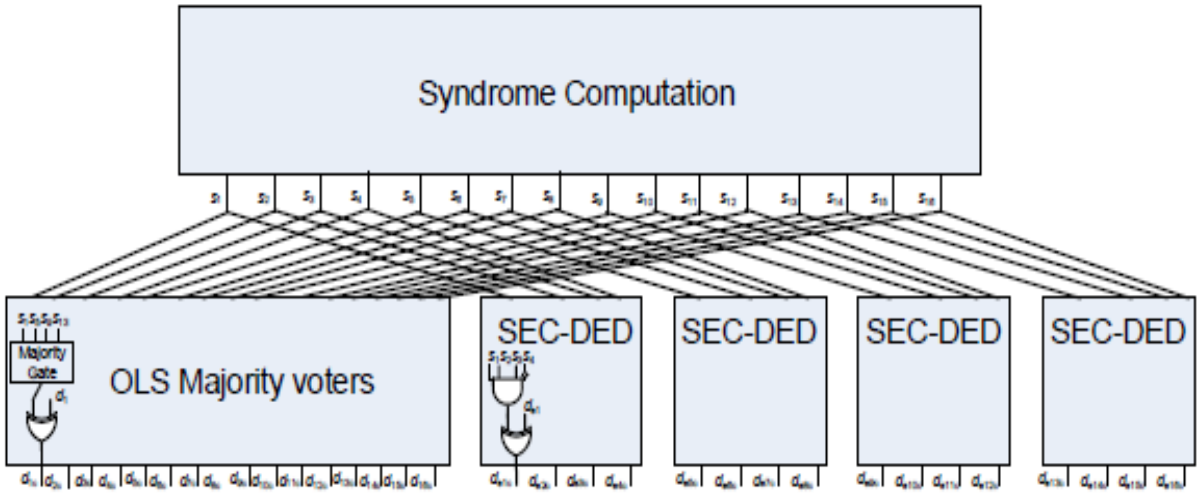


Figure 4:Decoder for the proposed DEC OLS code single sub-block extended with SEC-DED.

The majority voter is replaced with a threshold voter that gives a '1' output if the number of inputs that are equal to '1' is greater than or equal to the threshold *T*. The value of *T* is configured so that it is equal to ⌈ (number of nonmasked inputs)/2 ⌉ . if there are 7 inputs to the voter, but 2 of them are not part of the reduced code, then they are masked off, and the threshold of the voter is set to 3. As long as no more than 2 out of the 5 non-masked voter inputs are correct, the output of the voter will be correct, so two errors can be tolerated. The control lines are generated by control logic that decodes the configuration bits.

Syndrome computation :

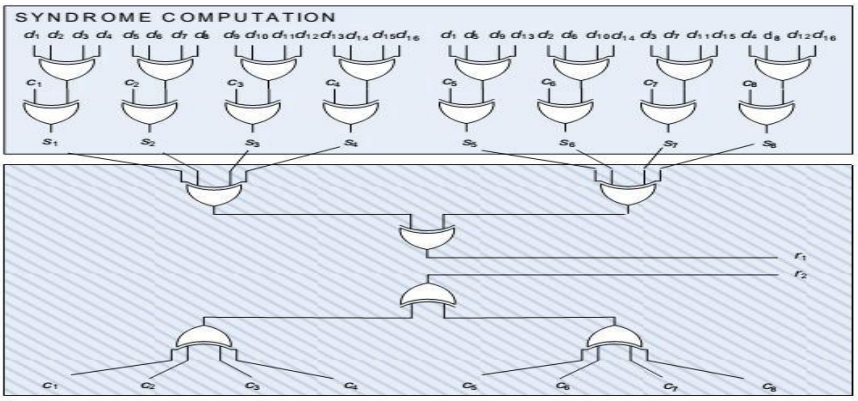


Figure 5: Syndrome Computation

An code has message symbols and coded symbols, where code symbol belongs to code has an error can be corrected by finding out the error location and error value, where an erasure is defined as an error with a known error location code is specified as with m-bit codes on m-bit exist for all n and k for which $0 < n < k \leq 2m-1$. For this code is targeted for syndrome computation block over the data Therefore the, n bit is the total no. of code after encoding called codeword.

II. PROPOSED DECCODE

As mentioned in the introduction, the proposed solution is a combination of DS and SEC-DED codes. The only DEC DS code that exists is a (21,11) code that is One Step Majority Logic Decodable (OS-MLD) and can correct double errors and detect triple errors. The proposed scheme is based on this code that is firstly optimized by removing one parity bit to obtain a (20,11) DEC code that is also OS-MLD but that cannot detect triple errors. To explain the proposed code, we will first describe the encoding process whose block diagram is shown in Figure 6. It can be seen that the 32-bit data word is divided into three blocks, the first consists of bits 1 to 11, the second of bits 12 to 22 and finally the third block includes bits 23 to 32. Therefore, the first two blocks have 11 bits and the last one only 10 bits. The *xor* of the blocks is used as an input to a (20,11) DS encoder to obtain 9 parity check bits, *s*1*d*, *s*2*d*, ..., *s*9*d*. The first and second data blocks are fed into two SEC-DED encoders to obtain other two groups of 5 parity check bits: *s*1*a*, *s*2*a*, ...,*s*5*a* and *s*1*b*, *s*2*b*, ..., *s*5*b*. As a result of the encoding, we obtain 19 parity check bits. This compares favorably with previous proposals based on OLS codes that require 21 to 23 parity check bits.

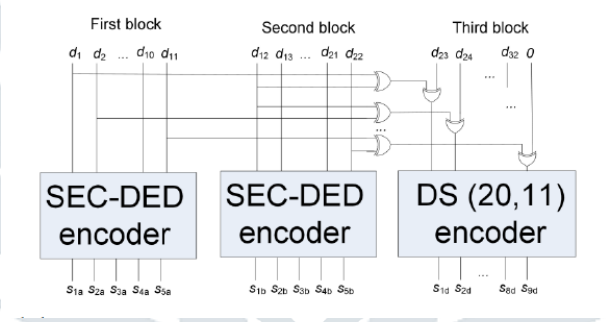


Fig.6. Encoder for the proposed scheme.

Let us now discuss how the proposed code can implement double error correction on the data bits. The block diagram and correction logic of the decoder is illustrated in Figure 7. The first step is to compute the syndromes and correction signals for each of the codes: e1a, e2a, ...,e11a, e1b, e2b, ..., e11b and e1d, e2d, ..., e11d. For the SEC-DED codes, in addition to the correction signals, a Double Error Detected (DED) signal is generated. In a typical decoder, each of those correction signals would be xored with the corresponding data bit to implement the correction. In our case, the first two groups of correction signals could be used to implement SEC on the first two data blocks. However, the code should implement DEC instead of SEC. In the case of the DS code, the inputs are not data bits but rather the xorof three data bits. This means that a correction signal can correspond to an error in any of the three bits used to generate the xorused as input forwhich the correction signal is activated. In summary, the correction signals from each of the codes cannot be directly used to implement DEC and some additional logic is needed as described in the following.

For the bits in the first and second blocks, the correction of the data bits can be implemented as follows:

- If the DED is not activated, then use the correction signal from the SEC-DED code to perform correction. In this case, there can be at most one bit in error in the block and therefore the SEC correction signal can be used.
- If the DED is activated, then use the correction signal from the DS code to perform correction. In this case, there cannot be errors on the other blocks as we assume a maximum of two errors. Therefore, the DS correction signals for the two bits will be activated and can be used for correction.

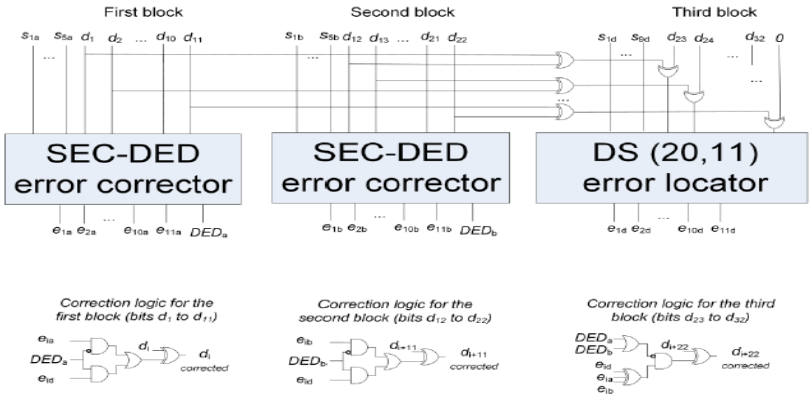


Fig.7. Decoder for the proposed scheme.

For the bits in the third block, the correction of the data bits is slightly more complex. This is because the input to the DS code is the xor of the bits from the three blocks. This means that the correction signals of the DS code may be activated by errors on any of the blocks. For example, an error on bit d1 would activate the correction signal e1d. Additionally a double error may compensate so that it is not seen by the DS decoder. For example, an error on bits d1 and d12 would produce a correct value in the first input to the DS decoder. The logic used for correction is shown in Figure 7 and works as follows:

- If any of two DED signals from the first blocks is activated, no correction is made. This is because the third block has no errors in this case.
- In the rest of the cases, the correction is done when the *xor* of the two SEC correction signals that correspond to the bits used to generate the DS input bit and the DS correction signal for that bit is one. When there are only errors in the third block this is equivalent to using the DS decoder. On the other hand, if there are errors on the first or second block, the use of the SEC correction signals ensures that no miscorrection is done on the third block. For example, an error on bit *d*1 would activate the correction signals *e*1*a* and *e*1*d* so that the *xor* is zero and bit *d*23 is not corrected. On the other hand, a double error on bits *d*1 and *d*23 would activate only *e*1*a* so that the *xor* is one and bit *d*23 is corrected.

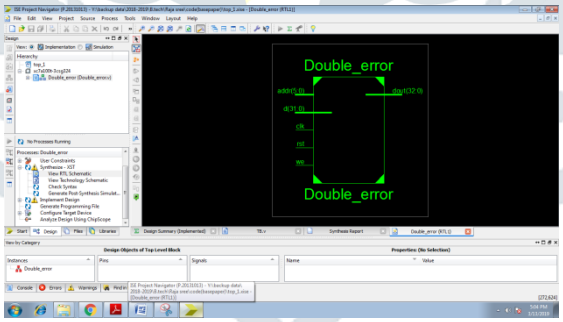
Finally, errors can also affect the stored parity check bits. However, single or double errors on parity bits cannot cause miscorrections. The same applies to an error on a parity bit and a data bit and in addition, in that case, the data bit is always corrected.

This case by case analysis shows that the proposed scheme is able to ensure that the 32 data bits are correctly recovered when the memory suffers a single or a double error. In the next section, the code is evaluated in terms of its error correction capabilities and also its implementation complexity.

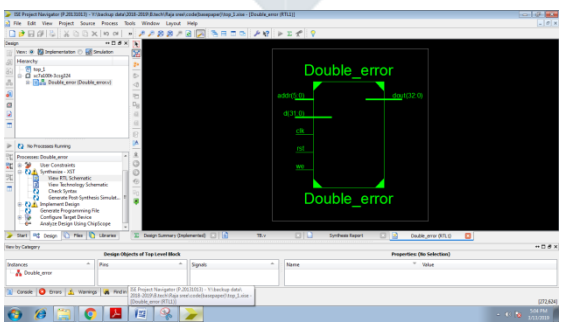
I. Results And Discussion

The proposed encoder and decoder have been implemented in HDL and evaluated in terms of their error correction capabilities and their implementation cost. In the first case, all possible combinations of single and double bit errors have been injected on coded words to check that the decoder is able to correctly recover the 32 data bits. This validates the case analysis presented in the previous section and confirms that the code provides DEC protection.

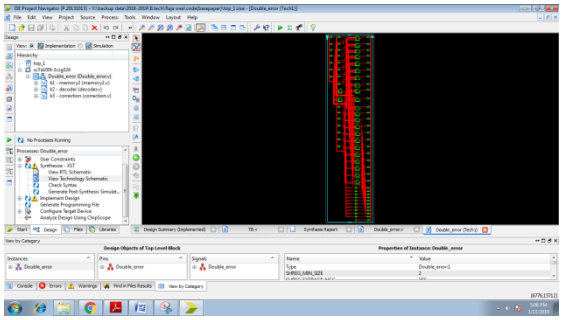
Block Diagram:



RTL schematic:



Technology Schematic:



Comparison Table

Parameters	OLS code	Proposed
Parity bits	21	19
Area(Luts)	28	24
Delay(ns)	1.253	1.638
Power(mW)	10.67	8.20

Conclusion

In this letter a Double Error Correction (DEC) code to protect 32 bit data word memories has been proposed. The code is based on the use of a DEC Difference Set (DS) code combined with two Single Error Correction-Double Error Detection (SEC-DED) codes. The proposed solution reduces the number of parity check bits compared to existing schemes based on Orthogonal Latin Square (OLS) codes while providing a decoding delay that is lower than that of a traditional existing systems. Therefore it provides memory designers with another option that can be useful to trade-off delay and memory size. Finally, the scheme proposed in this paper can be used to combine other OS-MLD DEC codes with SEC-DED codes to protect different word lengths. This is however constrained by the availability of OS-MLD DEC codes. The study of this generalization is left for future work.

References

[1] C. L. Chen and M. Y. Hsiao, Error-correctingcodes for semiconductor memory applications: A state of the art review, IBM J. Res. Develop., vol. 28, No. 2, pp. 124-134, 1984.

[2] V. Gherman, S. Evain, N. Seymour, and Y. Bonhomme, Generalized parity check matrices for SEC-DED codes with fixed parity, in Proc. IEEE On Line Testing Symp., July 2011.

[3] R. Naseer and J. Draper, DEC ECC design to improve memory reliability in sub 100 nm technologies, Proc. IEEE ICECS, pp. 586-589, 2008.

[4] S. Lin and D. J. Costello, Error Control Coding, 2nd ed. Englewood Cliffs, NJ, USA: Prentice Hall, 2004.

[5] M. Kishani, H.R. Zarandi, H. Pedram, et al, HVD: horizontal-vertical-diagonal error detecting and correcting code to protect against with soft error, Design Automation for Embedded Systems, vol. 15, No (3-4), 2011, pp.289-310.

[6] M.S. Rahman, M.S. Sadi, S. Ahammed, Soft error tolerance using Horizontal-Vertical-Double-Bit Diagonal parity method, International Conference on Electrical Engineering and Information Communication Technology. IEEE, 2015, pp.1-6.

[7] C. Argyrides, D.K. Pradhan, T. Kocak, Matrix Codes for Reliable and Cost Efficient Memory Chips, IEEE Transactions on Very Large Scale Integration Systems, vol. 19, No. 3, 2011, pp.420-428.

[8] S. Liu, Y. Xiao, G.Mao, Extend orthogonal Latin square codes for 32 bit data protection in memory applications, Microelectronics Reliability, Elsevier, vol. 63, August 2016, pp. 278-283.

[9] S. Liu, P. Reviriego, A. Sanchez Macian, J.A. Maestro, L. Xiao, Comments on “Extend orthogonal Latin square codes for 32-bit data protection in memory applications, Microelectronics Reliability, (2016)”, Microelectronics Reliability, Elsevier, vol. 69, February 2017, pp. 126-129.

[10] M. Demirci, P. Reviriego, J.A. Maestro, Optimized Parallel Decoding of Difference Set Codes for High Speed Memories, Microelectronics Reliability, vol. 54, No 11, November 2014, pp. 2645-2648.