Modified Orthogonal Latin Square Codes for SEC-DAEC-DAAEC-TAEC

[1]Shruthymol T.D, [2]Reneesh C Zacharia
[1]MTech Student, [2]Assistant Professor
VLSI and Embedded System,
Mangalam College of Engineering Kottayam, India

***Abstract*— As integrated circuit technology scales into the deep-submicron regime, radiation-induced soft errors threat the reliability of on-chip memory applications. 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 affect more than one memory bit. Since SEC-DED codes cannot correct multiple errors, they are often combined with interleaving. Recently Single Error Correction- Double Adjacent Error Correction-Double Almost Adjacent Error Correction-Triple Adjacent Error Correction Codes (SEC-DAEC-DAAEC-TAEC) are presented to correct these errors. However, these conventional codes requires more area and delay.To obviate these drawback Modified Orthogonal Latin Square Codes for SEC-DAEC-DAAEC-TAEC Codes has been proposed.. The proposed codes have been implemented in Hardware Description Language and compared with some of the conventional SEC-DAEC-DAAEC-TAEC error correction code.**

***IndexTerms*— Double Adjacent Error correction, error correction codes, memory, Orthogonal Latin Squares (OLS), Single Error Correction Double Error Detection .**

I.INTRODUCTION

Transient errors due to radiation power supply noise ,etc, can cause bit-flips in a memory. To address these errors, Error Correcting Codes (ECC) schemes have emerged and Single Error Correction and Double Error Detection (SEC–DED) codes are widely employed in commercial on-chip memory applications . A soft error occur when a radiation event causes enough of a charge disturbance to reverse or flip the data state of a memory cell, register, latch or flip flop. The error is soft because the circuit/device itself is not permanently damaged by radiation if new data are written to the bit, the device will store it correctly.
The SEC– DED codes can correct single-bit errors and detect double-bits errors using redundant bits called parity check bits. 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. One option is to combine SEC-DED codes with interleaving . 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. A technique to design (SEC-DED Double Adjacent Error Correction (SEC-DED-DAEC) codes using Orthogonal Square Codes was then introduced. This can correct only single and double adjacent errors. In this brief a new class of SEC-DAEC-DAAEC-TAEC Codes is derived using Modified Orthogonal Latin Square Codes. These codes are one step majority logic decodable.(OS-MLD)and therefore, can be decoded with low latency. These codes are derived using OLS Codes.

II.CONVENTIONAL SEC-DAEC-DAAEC-TAEC CODES

Two optimization criteria have been used to select the codes:
1) the minimizing the total number of ones in the parity check matrix
2) the maximum number of ones inits rows. The first criterion is commonly used to minimize the decoder complexity while the second is used to optimize its speed.
The process used to design the codes is based on formulating the
problem as a Boolean satisfiability problem. An algorithm developed is used to solve it and to obtain a parity check matrix, which defines the code to be designed. After determining the values of n and k for the code to be designed (where k is the length of the original data word and n is the length of the final encoded word), the first step is the selection of error patterns to be corrected. In this case, the SEC feature is represented with error vectors (. . . 1 . . .), error vectors for DAEC shows the pattern (. . . 11 . . .), for DAAEC (. . . 101 . . .), and for TAEC (. . . 111 . . .),where the dots represents zero or more 0s (correct bits) and the 1s are the bits in error. The next step is to find the parity check matrix **H.** To find the matrix, a recursive

backtracking algorithm is used. It checks partial matrices and adds a new column only if the previous matrix satisfies the requirements. In this way, the algorithm startswith a partial_H matrix, with $n - k$ rows and only one column.New columns are added and the new partial matrices are checked recursively. Both the initial and the added columns must be nonzero,so there are $2^{n-k} - 1$ combinations for each column. The algorithm may choose better solutions according todifferent optimization criteria, but finding the best solution requires finishing the process. In this brief, two criteria have been used.

1) Smallest Hamming Weight of H*:* This solution commonly offers circuits with the lowest number of logic gates in a hardware implementation of the encoder and syndrome computation.

2) Smallest Hamming Weight of the Heaviest Row of H*:* The logic depth of each parity or syndrome bit generator depends on the Hamming weight of the associated row. The heaviest row is a factor for the maximum speed of the encoder and the decoder circuits.

For word length of *n*, a SEC-DAEC-TAEC code has to identify and correct $n + n - 1 + n - 2 = 3n - 3$ error patterns and a 3-bitburst ECC $n + n - 1 + n - 2 + n - 2 = 4n - 5$ error patterns. This means that the number of different syndromes has to be larger than $3n-3$ in the first case and larger than $4n-5$ in the second, which in turn means that the number of parity check bits $n-k$ should be such that $2n-k > 3n-3$ and $2n-k < 4n - 5$, respectively. It is important to note that these conditions are necessary but not sufficient: the lowest value of *n* meeting the above conditions minimizes the code redundancy, but it does not guarantee the existence of a code with the required features. If it does not exist, the alternatives are to increase the number of parity bits (i.e., increasing *n*, as *k* is commonly a value fixed by design) or to reduce the number of error patterns to be corrected.

III. OLS CODES

The OLS Codes were introduced decades ago to protect memories. On the end simultaneously have recently been propose to protect caches and inter connects. 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.

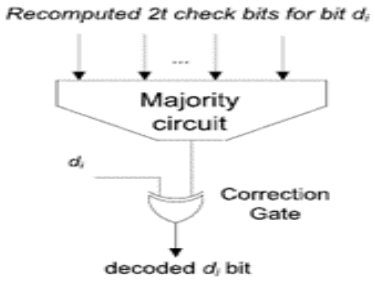


Fig.1 Illustration of OS-MLD decoding for OLS Codes [1]

The main advantage of OLS Codes is 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 (or syndrome bits) in which it participates. 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.

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.

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 bit 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. Another characteristic of OLS Codes is that they correct only errors on the data bits. This is not an issue as in memories, the goal is to recover the stored data correctly.

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 syndromes that is obtained by multiplying the stored word by the H matrix.

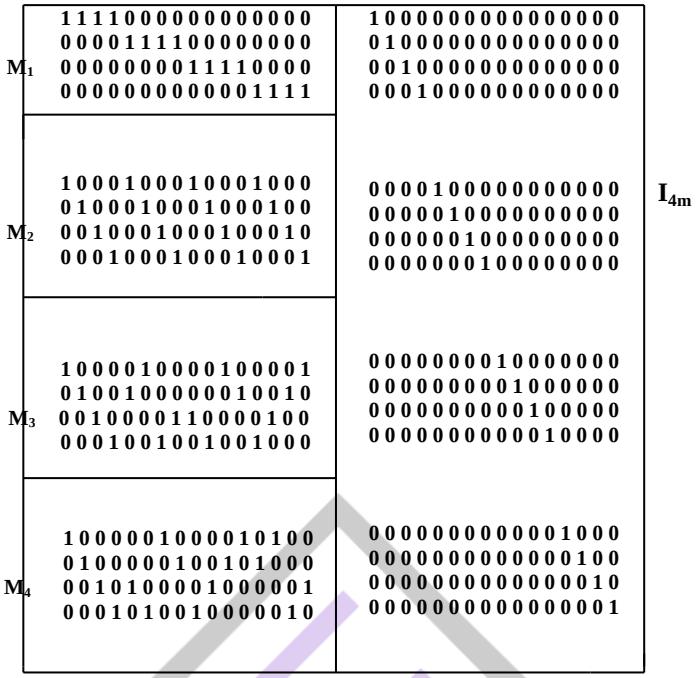


Fig .2 parity check matrix H for the OLS Codes with k= 16 and t = 2.

TABLE I
PARAMETERS OF SOME DEC-OLS CODES

k	n-k	m
16	16	4
64	32	8
256	64	16

The parity check matrix H for a DEC OLS Code with k= m2 is constructed as follows:

$$H = \begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \end{bmatrix} I_{4m} \qquad (1)$$

where I_{4m} is the identity matrix of size 4m and M_1, M_2, M_3, M_4 are the matrices with size $m \times m^2$ derived from OLS of size $m \times 3$ .The weight or number of ones ,of all the columns ,in the Mi matrices must be one. Therefore ,the first $k= m_2$ columns in H have a number of one's equal to 2t (four for DEC Codes) .In addition any pair of columns has at most a position with a one in common. This as discussed before enables the use of OS-MLD for decoding .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.The parameters of some DEC OLS Codes are summarized in Table 1.

IV. PROPOSED SEC-DED-DAEC-TAEC CODES USING ORTHOGONAL LATIN SQUARE CODES AND ANALYSIS

Proposed SEC-DED-DAEC-TAEC Code has the following properties:
All single bit errors can be corrected.
All double adjacent bit errors can be corrected.
All triple adjacent bit errors can be corrected.
Supposing that check bits are c0 – c6 and the 23-bit code word is $d_0 - d_{15}$, $c_0 - c_6$ the following equations are used to correct a single error, a double adjacent error ,a triple –adjacent or a double –almost adjacent error. The following error correction code analysis is based on [2]. Consider a code with k data bits and c check bits.

The code can represent 2^k binary values. The code word contains k+c bit positions that could have a single error,k+c-1 bit positions for double adjacent bit error,k+c-2 bit positions for triple adjacent bit error respectively. Including the set of correct values , there are (k+c)+(k+c-1)+(k+c-2) sets of binary values that must be represented in the bit code word.
Given 16 data bits ($d_0 - d_{15}$) the minimal number of check bits should be 7 according to

$$K+c-1 \leq 2^{c-2} \qquad (2)$$

Proposed error correcting codes requires an additional 7 check bits with SEC-DED-DAEC Codes.

$$d_0 \oplus d_3 \oplus d_4 \oplus d_5 \oplus d_8 \oplus d_{12} \oplus d_{13} \oplus C_0 = r_0$$

$$d_1 \oplus d_4 \oplus d_7 \oplus d_8 \oplus d_{11} \oplus d_{13} \oplus d_{14} \oplus C_1 = r_1$$

$$d_2 \oplus d_5 \oplus d_6 \oplus d_9 \oplus d_{10} \oplus d_{11} \oplus d_{14} \oplus C_2 = r_2$$

$$d_0 \oplus d_4 \oplus d_9 \oplus d_{12} \oplus d_{15} \oplus C_1 = r_3$$
$$d_1 \oplus d_5 \oplus d_8 \oplus d_{10} \oplus d_{11} \oplus d_{12} \oplus d_{14} \oplus C_4 = r_4 \qquad (3)$$

$$d_2 \oplus d_7 \oplus d_9 \oplus d_{10} \oplus d_{11} \oplus d_{12} \oplus d_{15} \oplus C_5 = r_5$$

$$d_3 \oplus d_6 \oplus d_9 \oplus d_{11} \oplus d_{12} \oplus d_{13} \oplus d_{15} \oplus C_6 = r_6$$

where $\oplus$ represents XOR and $c_0 - c_6$ are set to make $r_0 - r_6$ equal to zero under fault free conditions.
Where c (number of check bits) should be minimized to reduce the code length. Equation (3) can be transformed into a matrix form as follows.

$$\begin{bmatrix} 100111001000110010000000 \\ 010010011001011001000000 \\ 001001100111001000100000 \\ 100010000100100100100000 \\ 010000101111001000000010 \\ 000100100101110100000001 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \\ d_8 \\ d_9 \\ d_{10} \\ d_{11} \\ d_{12} \\ d_{13} \\ d_{14} \\ d_{15} \\ c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{bmatrix} = \begin{bmatrix} r_0 r_1 r_2 r_3 r_4 r_5 r_6 \end{bmatrix} \qquad (4)$$

Where the left most matrix is called check matrix that determines which data bits and check bit are XOR ed and the right most row vector $[r_0 r_6]$ is a syndrome that is a zero vector under fault free conditions and a non-zero vector under faulty conditions.
Table II provides syndromes $[r_0\text{-}r_6]$ that corresponds to single errors ,double almost adjacent errors accoording to (1.3) .In table II ,x represents a single error in bit position x.(x+1) represents a double adjacent error in bit positions x and (x+1),(x+2) represents a double almost adjacent error in bit positions x,(x+2) ,x+2 and x,x+1,x+2 represents a triple adjacent error in bit positions x,x+1,x+2.As seen from table II ,a single error in bit position x generates a syndrome that matches the x^{th} column in the check matrix. For example ,a single error in bit d0 generates a syndrome [1001000] that matches the first column in the check matrix. A single error in bit d1 generates a syndrome [0100100] that matches the second column in the check matrix. A double adjacent error in bit positions x and x+1 generates a syndrome that matches the XOR of the x^{th} and $(x+1)^{th}$ columns in the check matrix. For example a double adjacent error in bits d0 and d1 generates a syndrome [1101100] that matches the XOR of the first and the second columns in the check matrix. As seen from table II, a double almost adjacent error in bit positions x and x+2 generates a syndrome that matches the XOR of the x^{th} and $x+2^{th}$ columns in the check matrix .

TABLE II
SYNDROMES FOR DETERMINING SINGLE,DOUBLE AND TRIPLE ERRORS

Faulty bit	Syndrome for single, double and triple errors			
	single	Double		Triple
	x+1	x ,x+1	x,x+2	x,x+1,x+2
d_0	1001000	1101100	1011010	1111110
d_1	0100100	0110110	1100101	1110111
d_2	0010010	1010011	1111010	0111011
d_3	1000001	0101001	0010101	1111101
d_4	1101000	0111100	1111001	0101101
d_5	1010100	1000101	1110110	1100111
d_6	0010001	0110011	1110101	1010111
d_7	0100010	1000110	0111001	1011101
d_8	1100100	1111111	1110010	1101001
d_9	0011011	001101	0101100	0111010
d_{10}	0010110	0100001	1011001	1101110
d_{11}	0110111	1111000	1010110	0011001
d_{12}	1001111	0101110	1111011	0011010
d_{13}	1100001	1010101	1101010	1011110
d_{14}	0110100	0111111	1110100	1111111
d_{15}	0001011	1001011	0101011	1101011
c_0	1000000	1100000	1010000	1110000
c_1	0100000	0110000	0101000	0111000
c_2	0010000	0011000	0010100	0011100
c_3	0001000	0001100	0001010	0001110
c_4	0000100	0000110	0000101	0000111
c_5	0000010	0000011	-	-
c_6	0000001	-	-	-

Taking the parity check matrix in (1) 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 M1 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 (that is have a one in the same row in the H matrix) at most one parity check bit .This can be clearly observed in Fig 2.In addition those group 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 from the first group g_1.It can be observed that they do not share any parity check bits. Therefore when M_1 is removed they do not share any parity check bit. The same occurs for other group of bits 5-8(g_2),9-12(g_3) and 13 -16(g_4). For example ,the first four data bits share the first parity check bit in the M1 matrix and first group g_1.It can be observed that they do not share any other paritry 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(g2), 9-12(g3), and 13– 16 (g4) .In the reduced matrix ,each data bit participates in the three parity checks .Therefore ,if a majority vote is used to decode the bits, single and double errors can be corrected.

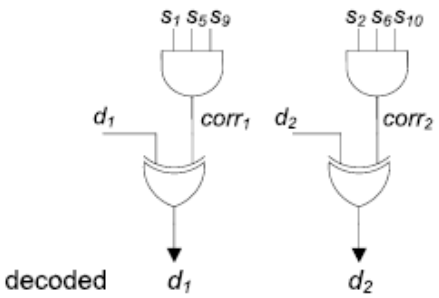


Fig.3 Illustration of the proposed decoder for data bits 1 and 2 [1]

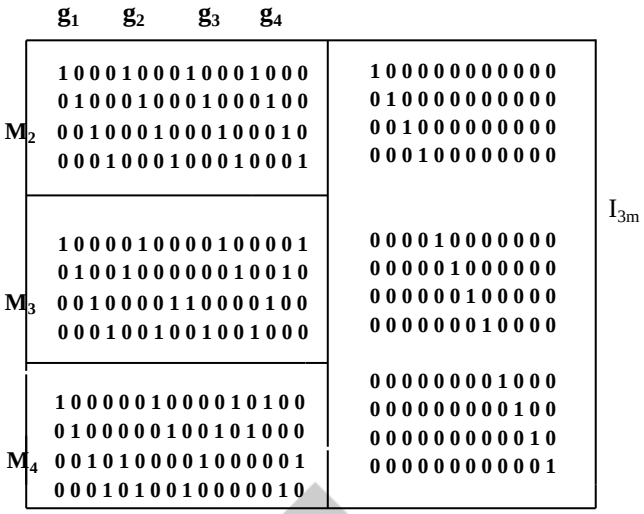
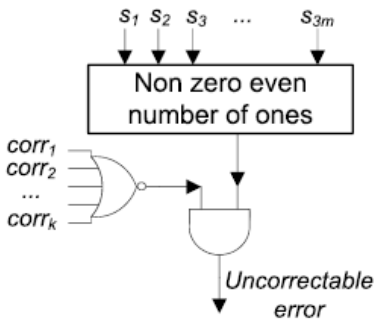


Fig.4 Reduced parity check matrix H after the removel of M_1

However ,double errors can also cause miscorrection 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 si values correspond to bits of the syndrome vector obtained by multiplying the word by H matrix. In this case , the code will obviously not miscorrect when there are two errors. 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. However, some double adjacent errors may affect bits on different groups. For example, an error on bits 8 and 9 affects it in g2 and another in g3.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 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 non adjacent 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 un detected. However this issue can also be solved by carefully placing the bits into the memory.

Fig.5 Reduced parity check matrix H after the removel of M_1 with the proposed bit placement.



.Fig.6.Detection of un correctable errors [1]

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 non adjacent errors that affect two bits,if any
5)Implement the circuit of Fig 6 to detect uncorrectable errors 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 .Obviously ,when the error affects a data bit and a parity bit,if the data bit is corrected error was correctable. This enables a simple method to detect the un correctable errors is shown in Fig 6.It is based on detecting a non zero 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.
The proposed scheme can be summarized as follows:
1)Reduce the H matrix of the DEC OLS code by eliminating M_1.
2)place the bits in the group 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 (three-way AND)to correct errors;

V. SIMULATION RESULTS

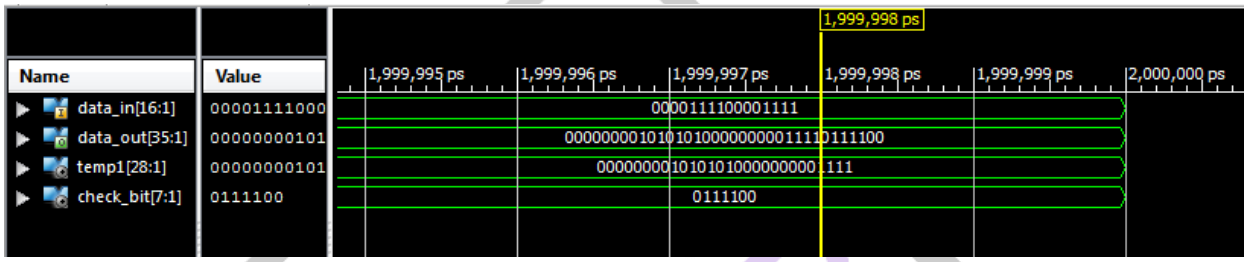


Fig .7 Simulation result of SEC-DAEC-DAAEC-TAEC Encoder

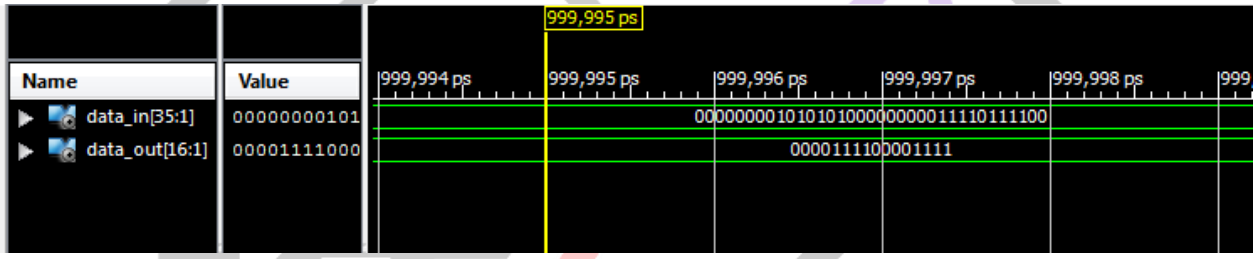


Fig .8 Simulation result of SEC-DAEC-DAAEC-TAEC Decoder

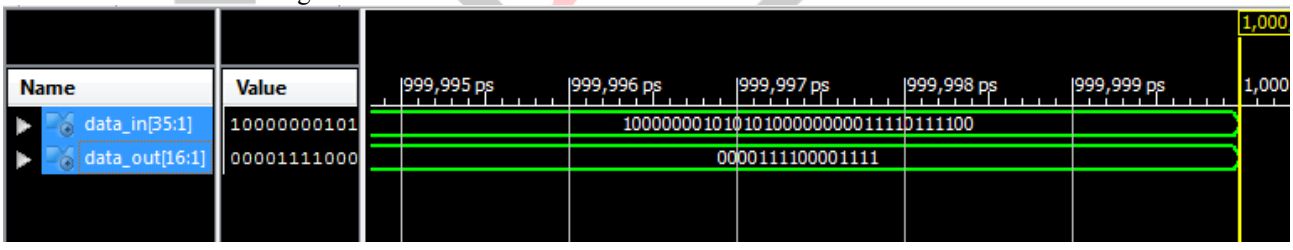


Fig .9 Simulation result of SEC-DAEC-DAAEC-TAEC Decoder (Single error Correction)

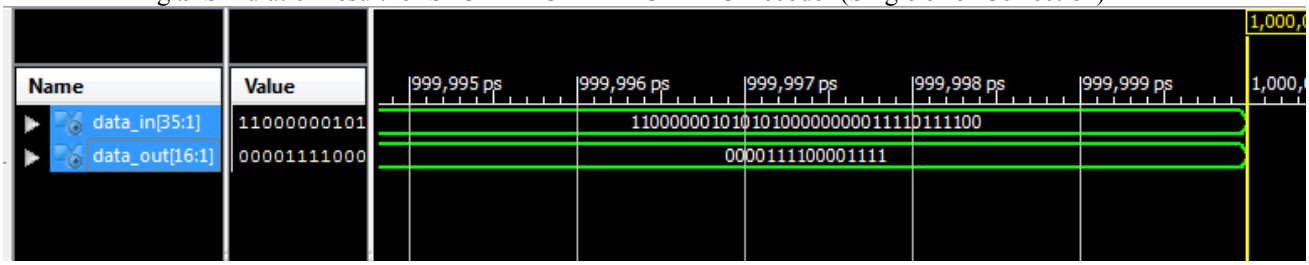


Fig .10 Simulation result of SEC-DAEC-DAAEC-TAEC Decoder (Doubleadjacent Error Correction)

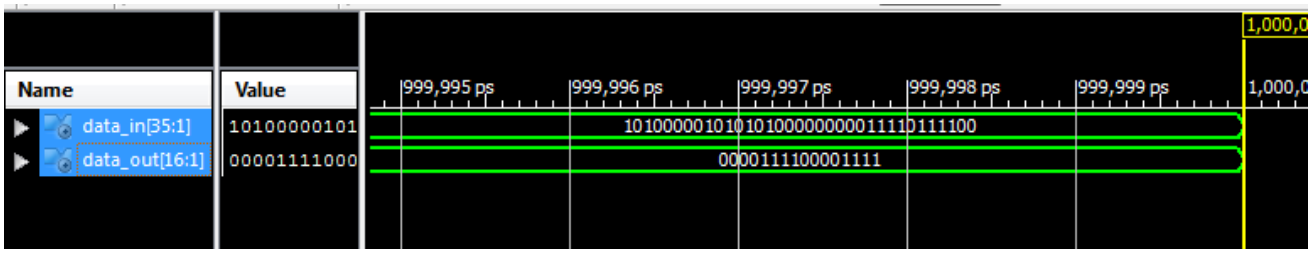


Fig .11 Simulation result of SEC-DAEC-DAAEC-TAEC Decoder (Double Almost Adjacent Error Correction)

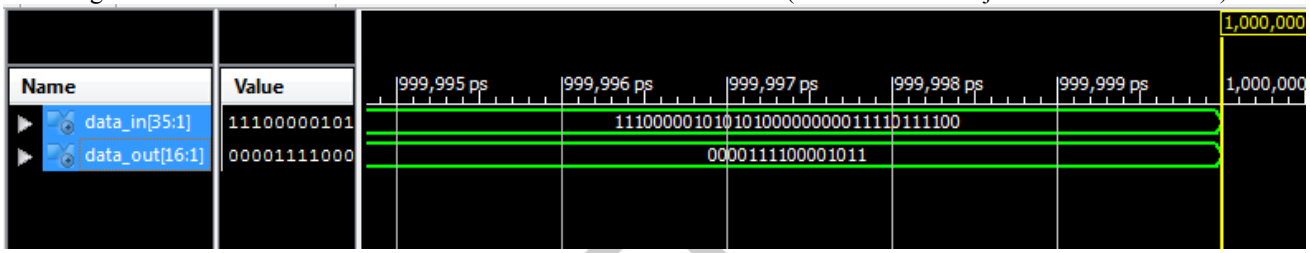


Fig.12 Simulation result of SEC-DAECD-DAAEC-TAEC Decoder (Triple Adjacent Error Correction)

TABLE III
Comparison Table For Proposed SEC-DED-DAEC-TAEC Codes

Proposed SEC-DAEC-DAAEC-TAEC	**Area (no:slices)**	**Delay(ns)**	**Power(mW)**
ENCODER	15	7.79	629
DECODER	217	16.46	127
Existing SEC-DAEC-DAAEC-TAEC	**Area (no:slices)**	**Delay(ns)**	**Power(mW)**
ENCODER	29	9.558	956
DECODER	294	16.965	163

VI .CONCLUSION

In this brief ,a new class of SEC-DED-DAEC-TAEC codes has been presented. The codes are derived from DECOLS Codes and can be decoded with low latency. The codes have been implemented in HDL .Simulation results confirm that the proposed codes using Orthogonal Square Codes can correct single errors, double-adjacent errors, triple-adjacent errors and double-almost-adjacent errors. The resulting implementations are compared with existing SEC-DED-DAEC-TAEC Codes and found that the proposed codes has significant reduction in area and delay.

REFERENCES

[1] Pedro Reverie ,Salvatore Pontarelli, Adrain Evans ,and Juan Antonio Maestro,″ A Class of SEC-DED-DAEC Codes derived from Orthogonal Latin Square Codes" *IEEE Transactions on VLSI Systems* ,vol 23,no.5,May 2015.

[2] X.She,N,Li and D.W Jenson,“ SEU tolerant memory using error correction code *“,IEEE Trans,Nucl.Sci,vol 59,no.1,pp.205210*,Feb 2012. .