# Extending Quadruple Adjacent Error Detection and Correction to Seven Bit Adjacent Error Correction to Protect Memory from Soft Errors

**[1]R.Vinoth, [2]R.Sasireka, [3]L.Abisha, [4]M.Anupriya, [5]V.Harini**
[1]Associate Professor, [2]Assistant Professor, [3,4,5]UG students
[1,3,4,5]Department of Electronics and Communication Engineering
P.S.R Engineering college sivakasi, India
[2]Department of Biotechnology, Mepco Schlenk Engineering College, Sivakasi, India

**Abstract—** Soft errors induced by radiation are a major reliability concern for memories. For past years a memory reliability is a concern for soft errors. With changes in device feature includes size decreasing and memory density increasing, an event upset in memory may cause data errors that may generate adjacent bit upsets in a word. Error Correction Codes (ECC) are commonly used to protect the memory contents from soft error and to avoid data errors in a memory. Advanced technology scaling reduces the distance between the memory cells, which makes soft errors in more memory cells once a radiation particle hit. Therefore, the construction of ECCs with advanced error correction and low redundancy which causes an important problem, especially for adjacent or burst errors. The main objective of this project is to prevent the memory against soft error by Burst Error Correction and Error Detection techniques with low redundancy and design complexity. To attain this objective, a technique called extend 4-bit adjacent error correction with 7-bit adjacent error correction (7-AEC) is presented by using the proposed search algorithm.It provides the protection of SRAM from radiation effects and mitigating the MBUs that affect up to six adjacent bits.

**INDEX TERMS-**Burst error correction codes (ECC), Multiple bit upset (MBU), Memory, Soft errors, Quadruple adjacent error correction (QAEC).

## I. INTRODUCTION

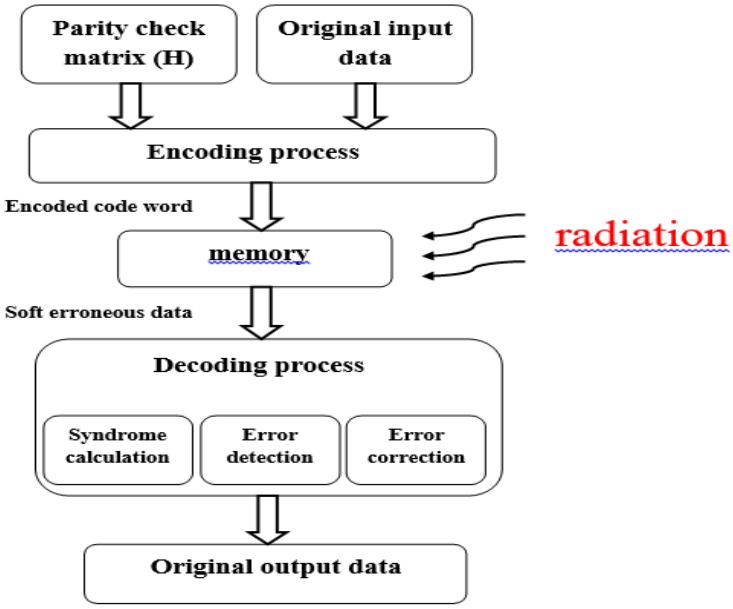
Error detection and correction codes have been used traditionally in memories to protect them from errors. Commonly the applications such as avions and space and military are mostly affected by the soft errors.Less energy is required to produce soft errors in memories. The high energy particle can directly affect several adjacent memory cells or the device (memory). In any system, memory is the crucial part and it occupies most percentage of the area compare than other components. Memories are easily suffered by environmental factors like radiation, temperature etc. This may affect the reliability of the electronic system.With the decrease of integrated circuit technology dimensions,the technology scaling reduces the distance between the memory cells - this makes more cells are affected (MBUs) by a particle hit. Back bone of system operation is data, which is stored in memories. So, we need to improve our memory performance by developing efficient error correction and detection codes. The various studies on error detection and correction methodologies based on number of bits affected was discussed when one memory faced any radiation hardening on it. They are, Single bit error or Upset (SEU), Multiple bit error or Upset (MBU), Burst error. Soft errors caused by the highly-charged particles with enumerous energy, protons, neutrons in space and alpha particles on ground or heavy ion strikes a storage element e.g., look-up-tables ,flip-flops etc., have become an important factor of memory reliability. The effect can produce an inversion in the stored value and this can modify the function of the design.Soft errors change the value in one or more memory cells and it will lead to data corruption and system failure. The error upset is a major concern in space applications, but decrease of technology scales MBUs become more and more common in recent years. ECCs which can correct data errors are widely used to prevent soft errors from causing data corruption in memories. In this project a 7 bit adjacent error correction (7-AEC) technique was proposed to prevent the memory against the bits of burst errors.

## II. OBJECTIVE OF THE PROJECT

The prime objective of our project are

- To prevent the memory against multiple bit upsets (MBUs) by Error Correction and Error Detection Techniques.
- To design ECC with advanced error correction and low redundancy.

## III. BLOCK DIAGRAM




## IV. BLOCK DIAGRAM EXPLANATION

H- Matrix and original data input is given as the major inputs. Encoding process is carried out by using this inputs. The encoded code word is stored in the memory. This stored data may be corrupted. Then the error occurred is called as soft error. Before use any data stored in the memory, these soft errored data is given to the input for decoding process.

Decoder complete with three basis steps,

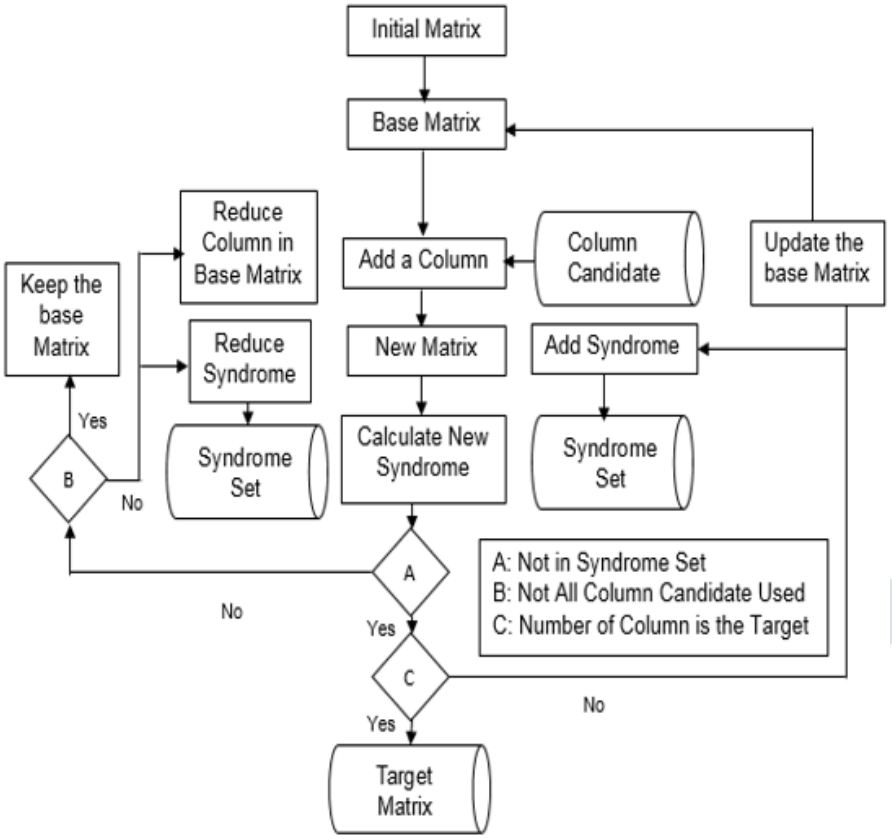
1) Syndrome calculation

2) Error detection

3) Error correction

After these steps, the error is detected and corrected through syndrome calculation if any error present. The decoded code word is get as the output from the decoder. Finally, the original input data is get back from memory by removing the extra or parity bits added to it.

## V. FLOW DIAGRAM




## VI. ERROR DETECTION AND CORRECTION CODES

An error detection and correcting code (EDCC) or forward error correction (FEC) code is a process of adding redundant data, or parity data, to a message, such that it can be recovered by a receiver even when at the presence of number of errors limits up to the capability of the code which is being used, either during the process of transmission, or on storage. So, the receiver need not to ask for retransmission of the data from the sender, a backchannel is not required in forward error correction.

## VII. ERROR DETECTION

If only needs to detect errors, one can compute the syndrome to see if it is all-zero. If the value of the syndrome is all-zero the assumptions can be made as that no errors occurred since the received vector is a code word. Suppose if the syndrome is not all zero, one knows that errors have occurred. The only time that the decoder will be incorrect, if the error pattern itself is a code word. Then, the syndrome value will all zero but errors have occurred. In the case of a binary symmetric channel, if it knows the number of code words of each Hamming weight, an expression for the probability of undetected error .

## VIII. ERROR CORRECTING CODES: (ECC)

Error correcting codes is a technique whereby more than the minimum number of binary digits are used to represent the messages. In binary error correcting codes, 2^k equally likely messages can be represented by k binary digits. If here indicated k digits are not coded, an error caused in one or more k binary digits will result in the wrong message being received. In binary error correcting codes, only certain binary sequences (called **code words**) are transmitted. The aim of the extra digits, called redundant or parity digits, is to detect and hopefully correct any errors that occurred in transmission. After transmission over a channel, we want to check that if the received binary sequence is in the of code words and if not, choose the code word most similar to what was received.

## IX. BINARY BLOCK LINEAR CODES

In previous works, codes for SEC-DAEC-DED, SEC-DAEC-TAEC, and 3-bit BEC have been proposed. All of them are binary linear block codes. The process used to design these codes is based on some rules for linear block codes construction. The presented code is also binary linear block codes and obey similar construction rules. Normally, the binary codes are described by the number of data-bits, *k*, redundancy bits, $(n - k)$, and the block size of the encoded-word, *n*. An $(n, k)$ code is defined by its generator matrix *G* or parity check matrix *H* in

$$G = [P_{k\times(n-k)} \cdot I_{k\times k}] \; H = [P^T \cdot I_{(n-k)}]$$

Where $I_{k\times k}$is the identity matrix, $P$ is the matrix with size $k \times (n - k)$, and $P^T$is the transpose of $P$. In the encoding process, the generator matrix $G$ is used to encode the data bits through the process in

$$v = u \cdot G$$

Where $u(u_0, u_1, \ldots, u_{k-1})$ are the data bits to be encoded, and $v(v_0, v_1, \ldots, v_{n-1})$ is the codeword. In the decoding process, the parity check matrix $H$ is used to decode the received codeword through the process in

$$S = r \cdot H^T$$

Where $r\ (r_0, r_1, \ldots, r_{n-1})$ is the received codeword, $S(s_0, s_1, \ldots, s_{n-k-1})$ is the syndrome, and a significant parameter for correcting errors. The errors injected into the received code can be described by using

$$r = v + e\ (e_0, e_1, \ldots, e_{n-1})$$

Where $e\ (e_0, e_1, \ldots, e_{n-1})$ is the error vector indicating that an error occurs in the $i_{\text{th}}$ bit when $e_i$= 1. When multiple bit errors occur in the received code word $r$ with the error vector $e = (e_0, e_1, \ldots, e_{n-1})$, the syndrome of the code considering the error vector in decoding process can be calculated by the method in

$$S = e \cdot H^T$$

This equation formulates the relationship between the syndrome and the corresponding error pattern. Considering the detailed structure of the parity matrix $H$, when one error occurs in the $i_{\text{th}}$ bit, the corresponding syndrome is equal to the $i_{\text{th}}$ column vector. When errors occur in the $i_{\text{th}}$ bit and the $j_{\text{th}}$ bit, the corresponding syndrome is equal to the *xor* result of the $i_{\text{th}}$ column vector and the $j_{\text{th}}$ column vector. Therefore, if one error can be corrected or detected, it obeys the following rules.

1) *Correctable Restriction:* The corresponding syndrome vector is unique in the set of the syndromes.
2) *Detectable Restriction:* The corresponding syndrome vector is nonzero.

**9.1 ALGORITHMS**

The error detection and correction methods are need special algorithms to solve the problems. There are many algorithms were proposed. The first algorithm is Hamming code.

**X. HAMMING CODE ALGORITHM**

This code identifies the error bit occurred in the data sequence and also correct it. Number of parity bits are used to locate at certain positions in the code word. These parity bits depends upon the number of data bits. This hamming code can be applied to any number of data bits. To detect and correct the errors present in the memory, this code uses redundancy bits which means difference between the actual data sequence and the transmitted bits. In order to reduce the error the redundancy bits are located at the certain calculated positions. This is called Hamming code algorithm. The distance between two redundancy bits called Hamming distance. The number of parity bits calculated by using data bits,

**2^p>=n+p+1**

here,

**n** represents the number of bits in the data

**p** represents the number of parity bits.

For example,

If we have 4 data bits (n=4), then the number of parity bits to be added can be found by using trial and error method. Let's p=2, then

2^p = 4

n+p+1 => 4+2+1 => 7

This is violates the actual expression. Let's take p=3, then

2^p = 8

n+p+1 => 4+3+1 => 8

From this expression we need 3 parity bits to transfer the 4 data bits. The required check bits can be calculated by,

**C i = 2 i – 1**

| Data bit | Check bit |
|---|---|
| 2 | 2 |
| 4 | 3 |
| 8 | 4 |
| 16 | 5 |
| 32 | 6 |
| 64 | 7 |
| 128 | 8 |
| 256 | 9 |

## XI. STEPS FOR HAMMING CODE:

**1**. Calculate the length of the data input method of Hamming Code that represents the sum of the duration of the input data and check bits long. The term of the output data from the method of Hamming Code is equal to the length of the data input method of Hamming Code.

**2**. Mark all the bit position of the check bits. The position other than the post of the check bit which is the post of the data bits.

**3**. Determine the calculation formula of each check bit. For n = 1 the number of check bits, do the following:

- Record all positions where n bits of the member position is 1, except for the bit position itself. Member position is the binary form of the bit position.
- The formula of check bits n equals the XOR operation of position - a position he noted.

**4**. Next calculate the value of the check bits for data input and data output.

**5**. If the value of check bit input is not equal to the value of the check bit output means there is an error.

**6**. Perform XOR operation to test the check bits output and input bit.

**7**. Convert the XOR operation results in the form of decimal number.

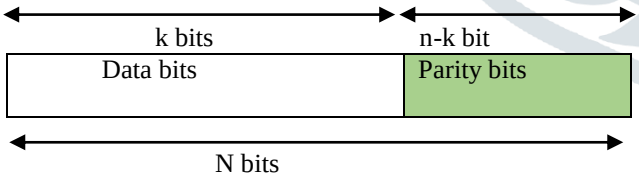
**8**. If the value of the results of the XOR operation is greater than the length of the data input or the value of the product of the XOR operation is similar to the position of the check bits, then it indicates there is more than one error.

This code is suitable for correcting only one bit error correction and detection. Then data input and output on the method of Hamming Code must be a result of the powers of 2n with *n* must be greater than one with a length of the data input and output of Hamming Code must be at least equal to 4 bits.

## XII. BINARY LINEAR BLOCK CODE

Linear block codes are more efficient encoding and decoding technique other hamming code. In the encoding procedure, k data bits are encoded to n code bits, each of 2^k data encoded into a unique n bit cord word. Each code word is a linear combination of row of G, G represents generator matrix (k×n).

## XIII. CODE WORD STRUCTURE




We have **n-k** parity bits, which collectively can represent **2^n-k** possibilities. For single bit error correction there are two cases needed to represent the parity bits.

Case 1: no error is occurred (1 possibility)

Case 2: exactly one of the cord word bit has an error (n possibility)

So we need,

$$n+1 = 2^{n-1}$$

$$n = 2^{n-1} - 1$$

Consider the simple linear parity code (k+1, k).

Parity bit is P = D1 + D2 + . . . Dk, so the code word is D1, D2 . . . DkP.

$$G = I_{k\times k} | 1^T$$

Here,

$I_{k\times k}$ is the identity matrix and $1^T$ (T-transpose) is the k bit column vector.

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}.$$

When k=3

Now let's consider the rectangular parity code, which has row(r)=2 and coloumn(c)=3. So, k=rc=6 and the number of parity bits are r+c=5, so this rectangular parity code is (11,6,3) linear code. If the data bits D1,D2,D3,D4,D5,D6 are organized with the first three in first row and second three data's are in second row then the parity equations are,

P1=D1 + D2 + D3
P2=D4 + D5 +D6
P3=D1 +D4
P4=D2 + D5
P5=D3+D6

G matrix will be,

$$G = \left(\begin{array}{cccccc|ccccc} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right).$$

G is a k×n, here (11×6) matrix and we can see the k×k indentity matrix followed by k×(n-k).

In the decoding procedure here we use the syndrome decoding, which uses syndrome bits. That's why linear block code is attractive. This is the efficient way to decode the linear block codes.

For decoding consider this matrix,

$$H = \left(\begin{array}{cccc|ccc} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right)$$

Cordword Is transmitted as cw and it is received as r. If c=1010001 but r=1110001, so the second bit was corrupted.
In our case k=4

$$\begin{aligned} H \cdot [1000000]^T &= [110]^T \\ H \cdot [0100000]^T &= [101]^T \\ H \cdot [0010000]^T &= [011]^T \\ H \cdot [0001000]^T &= [111]^T \end{aligned}$$

For completeness, syndromes for a single-bit error in one of the parity bits are, not surprisingly:

$$\begin{aligned} H \cdot [0000100]^T &= [100]^T \\ H \cdot [0000010]^T &= [010]^T \\ H \cdot [0000001]^T &= [001]^T \end{aligned}$$

The decoder implements the following steps to correct single-bit errors occur in data:
1. Compute c' = H · r^T
2. If c' is 0 then return k bits of r as the data bits.
3. If c' is not 0, then compare c' with the n pre-computed syndromes H . ei
ei = [00 . . . 1 . . . 0] is a 1× n matrix with in position i and 0 everywhere else.
4.If there is a match in the previous step for error vector el, then bit position l in the received word is in error. Flip that bit and then return the first k elements of r (note that we need to perform this check only for the first k error vectors because only one of those may need to be flipped, which is why it is sufficient to only store k single-error syndromes and not n errors). In this example, the syndrome for this H · [0100000]T = [101]T, which matches c' = H · rT. Hence, the decoder flips the second bit in the received word and returns the first k = 4 bits of r as the ML decoding. In this example, the returned estimate of message is [1010].
5. If there is no match, return the first k bits of r. Doing so is not necessarily ML decoding when multiple bit errors occur, but the bit error probability is small, then it is a very good approximation. It is unlikely that doing full-fledged ML decoding in this case which is worth the effort in terms of reduced bit error rate of a packet made up of many such coded blocks.

## XIV. CODE DESIGN TECHNIQUE

In this section, the code design technique for 3-bit BEC (Burst Error Correction) with QAEC (Quadruple Adjacent Error Correction) is discussed. The approach used is based on syndrome decoding and the analysis of the requirements in terms of parity check bits. The process used to design QAEC codes can be divided into error space satisfiability problem and unique syndrome satisfiability.

## XV. ERROR SPACE SATISFIABILITY

For a code with $d$ data bits and $k$ check bits, its input can represent $2^d$ binary values. If one error occurs, it has $d+k$ bit positions for a single error with output space of $(d+k) \cdot 2^d$ values. If adjacent errors occur, it has $d+k-1$ bit positions for double adjacent errors with output space of $(d+k-1) \cdot 2^d$ values, $(d+k-2)$ bit positions for triple adjacent errors with output space of $(d+k-2) \cdot 2^d$ values, . . . , $(d+k-(N-1))$ bit positions for $N$ adjacent errors with output space of $(d+k-(N-1)) \cdot 2^d$ values. If almost adjacent errors occur, it has $(d+k-2)$ bit positions for errors in 3-bit window with output space of $(d+k-2) \cdot 2^d$ values, $(d+k-3)$ bit positions for each type of errors in 4-bit window with output space of $(d+k-3) \cdot 2^d$ values.

To obtain the codes that can correct the errors with certain fault types, the sum of the output space value of error patterns should be less than or equal to the whole output space value $2^{d+k}$. For the presented code, to correct 3-bit burst and quadruple adjacent errors, the total condition of the error patterns is $(d+k-3)+(d+k-2)+(d+k-1)+(d+k)+(d+k-2)$, respectively, for quadruple adjacent errors, triple adjacent errors, double adjacent errors, single errors, and 3-bit almost adjacent errors. Based on the error space satisfiability principle, the relation between the space of the correct codeword and the space of the erroneous codeword can be derived from

$$2^d (5(d+k) - 8 + 1) \leq 2^{d+k}$$

Based on above equation, the minimum number of check bits used for 16, 32, and 64 data bits is shown in Table 1. Meanwhile, the available syndromes and the ones needed in the best case to correct 3-bit burst and quadruple adjacent error are also shown in Table 1. This table shows that the number of parity check bits needed is the same as for 3-bit BEC codes.

**Table 1: Minimum Check Bit and Syndrome Condition**

| Data Bits | Check Bits | Total Bits | Syndrome Needed | Syndrome Available | Syndromes Left |
|---|---|---|---|---|---|
| 16 | 7 | 23 | 107 | 120 | 13 |
| 32 | 8 | 40 | 192 | 247 | 55 |
| 64 | 9 | 73 | 357 | 502 | 145 |

The number of check parity bits used for the presented code should meet the requirement of the above equation. This restricts the check bit number and the minimum size of the dimension of $H$ matrix. The issue of the column vector selection for the unique syndrome of a correctable error is discussed in next section.

**UNIQUE SPACE SATISFIABILITY**

From the view of binary block linear codes, if a type of error patterns can be corrected, the syndrome of individual error patterns should be unique. For the presented code, the error patterns are *(. . . , 1, . . .)*for SEC, *(. . . , 11, . . .)* for DAEC, *(. . . , 111, . . .)* for TAEC, *(. . . , 1111, . . .)* for QAEC, and *(. . . , 101, . . .)* for 3-bit almost adjacent errors correction. Therefore, the unique syndrome satisfiability can be expressed by

$$S_{0i} \neq S_{0j}\ ,\ S_{1i} \neq S_{1j}\ ,\ S_{2i} \neq S_{2j} \quad (1)$$

$$S_{3i} \neq S_{3j}\ ,\ S_{4i} \neq S_{4j} \quad (2)$$

$$S_{0i} \neq S_{1j} \neq S_{2k} \neq S_{3l} \neq S_{4m} \quad (3)$$

Where $S_{0i}$ is the syndrome for single bit error, $S_{1i}$ is the syndrome for double adjacent bit errors, $S_{2i}$ is the syndrome for triple adjacent bit errors, $S_{3i}$is the syndrome for 3-bit almost adjacent bit errors, and $S_{4i}$is the syndrome for quadruple adjacent bit errors. The syndrome variables $S_{0i}$, $S_{1i}$, $S_{2i}S_{3i}$, and $S_{4i}$ are linear combinations of the $H$ matrix columns obeying the rules in

$$S_{0i} = h_i \quad (4)$$

$$S_{1i} = h_i \oplus h_{i-1} \quad (5)$$

$$S_{2i} = h_i \oplus h_{i-1} \oplus h_{i-2} \quad (6)$$

$$S_{3i} = h_i \oplus h_{i-2} \quad (7)$$

$$S_{4i} = h_i \oplus h_{i-1} \oplus h_{i-2} \oplus h_{i-3} \quad (8)$$

Where $i, j \in [1, n]$, $i \neq j$ . Equations (4)–(7) indicate the detail relation between the syndromes and the columns. It is also used to design the circuits of syndrome calculation block. The code design is a kind of Boolean satisfiability problem. Normally, the solution of this problem is based on the recursive back tracing algorithm, which is presented in this section. Here, from the view of the integrated circuits design, two criteria are considered to optimize the target codes.

1. *Smallest Hamming Weight of H:* This criteria commonly indicates that the solution can be completed by using the lowest number of the logic gates in the synthesis process of the encoder and decoder.
2. *Smallest Hamming Weight of the Heaviest Row of H:* Logic depth in the encoding and decoding process depends on the logic path with the largest delay. The smallest hamming weight in the heaviest row can decrease the delay of the encoder and decoder.

## XVI. SEARCHING ALGORITHMS

In this section, an algorithm is presented to solve the Boolean satisfiability problem based on the linear block code. A code optimization tool is based on algorithm which is developed to obtain the target H matrix with custom optimization restrictions. The introduction of the algorithm is divided into two subsections. In the first part, basic part of the algorithm is introduced to find the solutions meeting the requirement of the Boolean satisfiability. In second part, based on the basic part, the method with column weight restriction is designed to force the optimization process to use as few ones as possible, thus optimizing the total number of ones in the matrix and the number of ones in the heaviest row. This optimized version of the algorithm has been used to obtain all the codes presented in project.

## XVII. BASIC PART OF CODE DESIGN

In order to construct the expected codes, here the first step is to ensure the number of the check bits. From the aspect of low redundancy, the number of the check bits is set to the values shown in the above table. The number of the check bits is 7 for codes with 16 data bits, 8 for codes with 32 data bits, and 9 for codes with 64 data bits. The main idea of the algorithm is based on the recursive back tracing algorithm. At first, an identity matrix with block size (n − d) is constructed as the initial matrix, and the corresponding syndromes of the error patterns are added to the syndrome set. Then, a column vector selected from the 2n−d − 1 column candidates is added to the right side of the initial matrix. This process is defined as column added action. Meanwhile, the new syndromes which belong to the new added column are calculated. The column added action is successful when none of new syndromes is equal to the elements in a syndrome set and the corresponding new syndromes are added into the syndrome set. Then the base-matrix is updated to the previous matrix with the added column. Otherwise, the column-added action fails and then another new column from the candidates is selected. If all the column candidates are tried and the column-added action still fails, one column from the right side of previous matrix and the corresponding syndromes are reduced from the base-matrix and the syndrome set, respectively. Then, the algorithm continues the column added action until the matrix dimension reaches the expected value.

Normally, the recursive back tracing algorithm demands a large amount of computing resources and computing time. In order to accelerate the computing speed of the algorithm operation, firstly, the decimal operation is adopted instead of the matrix operation by conversing the column vectors into decimal numbers. Even the algorithm completing the execution of all conditions is not possible. Generally, if the code we expect exists, it is easy to obtain the first solution. With various optimization criteria, the algorithm can get better solutions. However, searching the best solution requires in the most cases the complete result of the whole searching process, unfeasible with today's computing resources. Therefore, it is more practical to use the best result obtained in a reasonable computation time.

## XVIII. OPTIMIZATION PART OF CODE DESIGN ALGORITHM

For both optimization criteria mentioned before, the column used to construct the matrix should have as low weight as possible. In this case, the weight of the used column is restricted so as to minimize the weight of the matrix and speed up the process of finding better solutions. With the presented column weight restriction method, the searching algorithm with more optimization criteria for the columns can find the solutions more effectively. The performance of the new finding process is remarkably effective for codes with small block size. When the number of data bits reaches 32 or 64, the computing time is still huge amount. Here, a function of recording the past procedure is also developed with the column weight restriction method to shorten the time cost of searching the target codes.

At the initialization step, all the column weight restrictions are set to $(n - d)$. Then, $A_0$ is set to 2, which means that the number of 1 in the corresponding column is 2. The searching algorithm with column weight restriction starts to find the solution. If the solution exists, record and update the status of the bit restricted and shift to the next bit column weight setting. Otherwise, releases the column weight restriction to $A_i = A_i + 1$. With the program constraints increasing, the target code is close to appear. Compared with the existing ECC, the more advanced performance are found by using this mentioned algorithm.

## XIX. RESULTS

Our proposed project is initially taken for the basis of adjacent bit error detection and correction for various bits for various word length such as 8-bit word length, 16 bit word length and 32 bit word length and 64 bit word length with the adjacent bit error correction up to 9 bits and N bit corrections also possible in any type of word length.

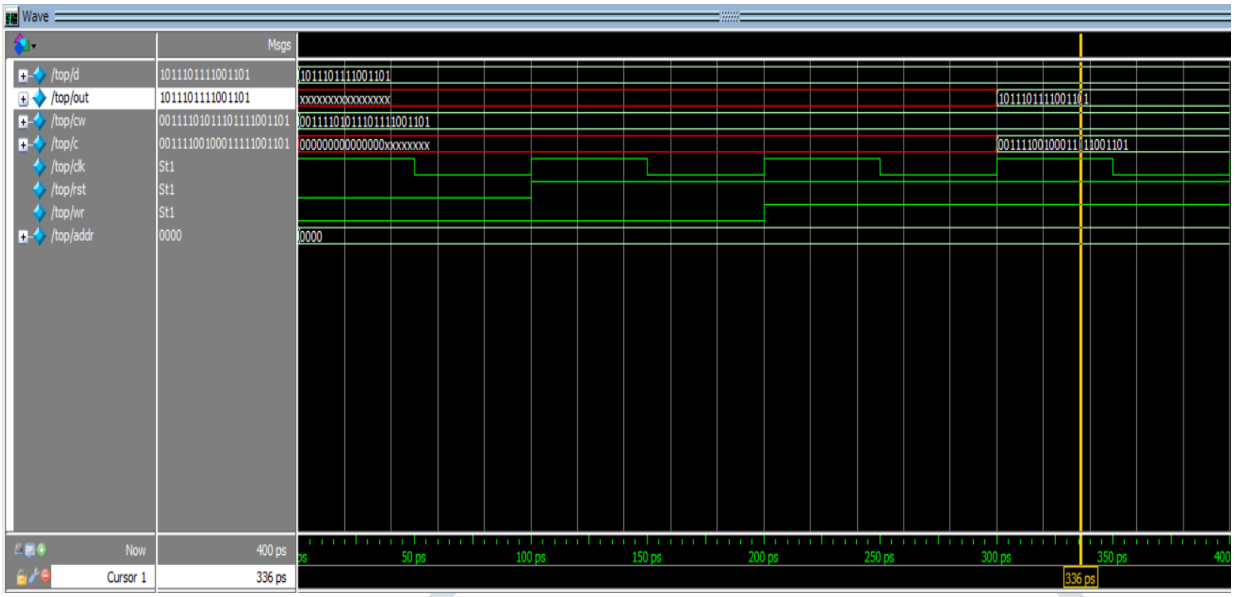
Even though the error correction is possible in 8 bit word length, where here is the introduction of the parity bits in addition to the data bits. The addition of parity bits are useful for the security of the data while they are in the way of transmission. The data transmission includes the encoding and decoding process. In the encoding process, the generated matrix which is used to encode the data bits. There arises the term code word. The code word is a product combination of data bits and the generated matrix.

In the decoding process, there is an introduction of the parity check matrix, which decodes the received code word. The syndrome is used for the error correction process, which is a product combination of the received code word and the transpose of the H matrix. The received code word is calculated with an identical formula that sum of the codeword and the error vector series. If in case of occurrence of error in multiple bits then the syndrome calculation formulation changes with the error vector in replacement of the received codeword.

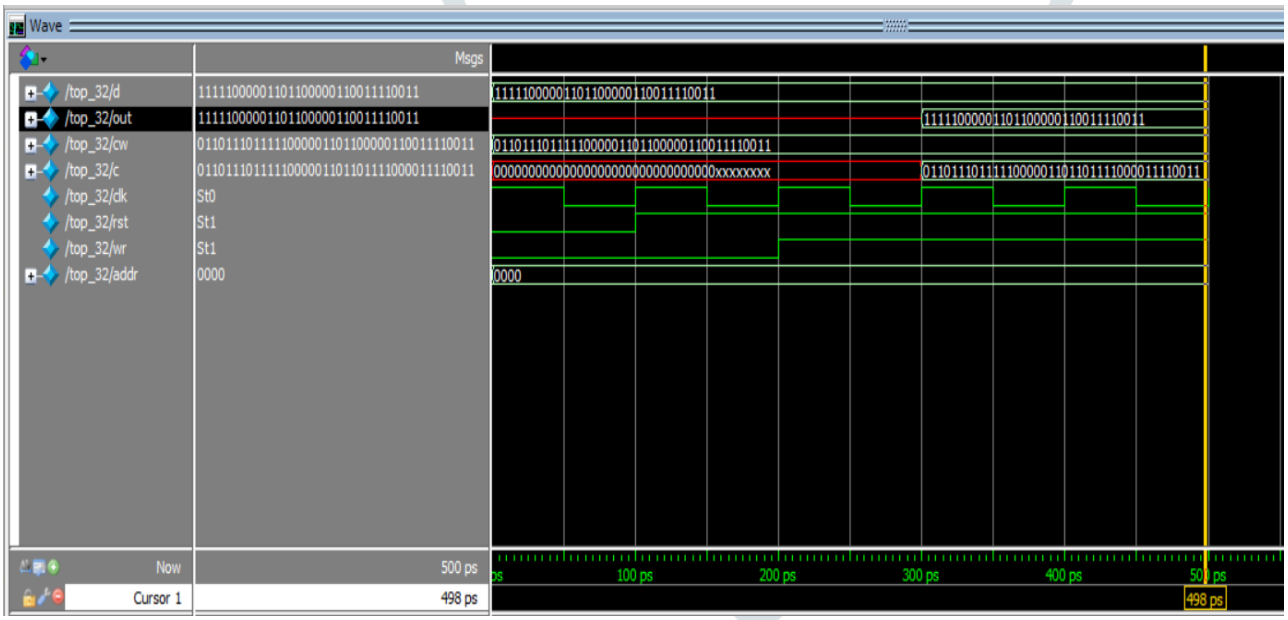
The adjacent error correction designs includes the error space satisfiability and unique syndrome satisfiability. The former principle states codes with error must satisfy the condition that sum of the output space value of error pattern must be less than or equal to whole output space value and the latter provides a condition that syndrome of individual error patterns should be unique. The total condition of the error pattern is checked accordingly.

In our proposed project, a six bit adjacent error correction is done for 8 bit word length, in which the 5 parity bits were included for the data transmission. The number of parity /check bits and the syndrome calculation is varying in consideration with the bits of the word length and the number of bits to be corrected. The adjacent bit error correction can be done for maximum of 9 bits correction as well N bit. Here our project shows upto the seven adjacent bit error correction for 64 bit data. In the same discussed above way, a six bit adjacent error correction is done for 16 bit word length, in which the 7 parity bits were included for the data transmission. Similar to the 8 and 16 bit word length, the error correction can be includes with the correction of six bit adjacent error correction in 32 bit word length, which have 8 parity bits and seven bit error correction for 64 bit data , which have 9 parity bits were included for the data transmission.

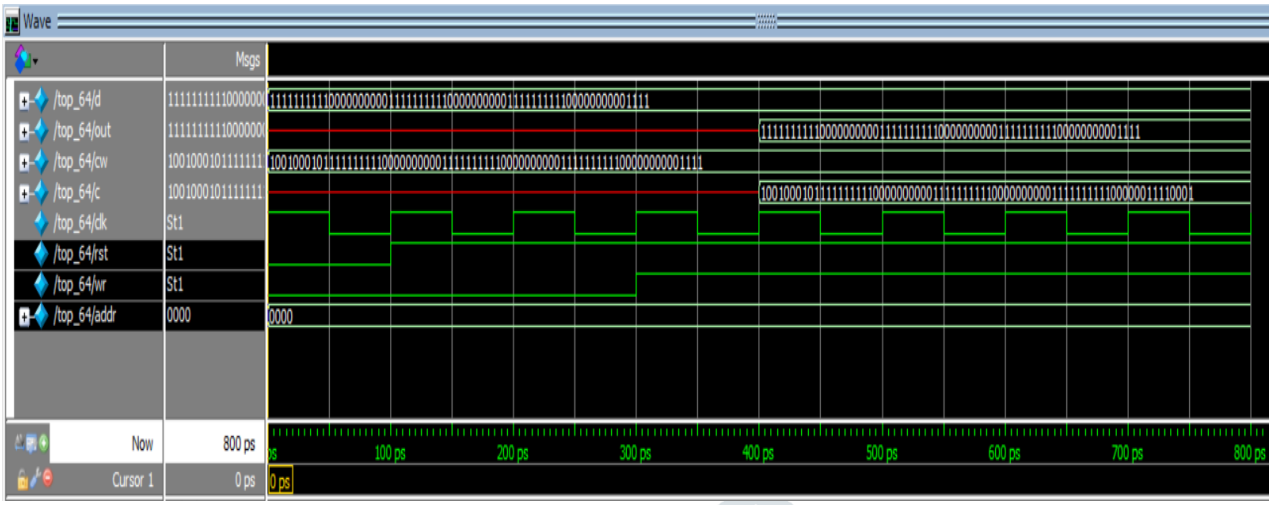
**TOP MODULE FOR 16 BIT DATA WORD FOR 6 BIT ERROR DETECTION AND ERROR CORRECTION:**




**TOP MODULE FOR 32 BIT DATA WORD WITH 6 BIT ERROR DETECTION AND ERROR CORRECTION:**



**TOP MODULE FOR 64 BIT DATA WORD WITH 7 BIT ERROR DETECTION AND ERROR CORRECTION:**




## XX. CONCLUSION

A '7' bit adjacent error correction (7-AEC) technique was proposed to prevent the memory against the bits of burst errors. And also, the H- Matrix was developed through Search Algorithm to compute the parity check bits for the corresponding stored data in memory. This technique do not require any additional parity check bits to detect and correct the adjacent n-bit burst error. The encoders and decoders for 8, 16, 32 and 64 bits are implemented using a 65-nm library. The results show that the proposed 7-AEC code has better encoding and decoding process in terms of error detection & correction, area, power and delay without increasing any parity bits of existing 3-bit BEC codes and quadruple adjacent error correction. This suggests that the proposed 7-AEC codes can be effectively used by designers to protect the SRAM memories from radiation effect and mitigate 7-adjacent bits.

## REFERENCES

[1]R. D. Schrimpf and D. M. Fleetwood," Radiation Effects and Soft Errors in Integrated Circuits and Electronic devices". Singapore: WorldScientific, 2004.

[2]K.Osado, Y.Saitoh,E.Ibe,K.Ishibashi,A cell tunnel leakage current ,SRAM for handling cosmic ray induced multi errors, IEEE 1952-1957(2003)

[3]B.Masmick,J.Wolf, "On linear unequal protection codes",published on IEEE information theory society,1967

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

[5]C.W.Slayman,"Cache and memory error detection, correction, and reduction techniques for terrestrial servers and workstations",published ,IEEE international conference,2005.

[6]Avjjit dutta, Nur.A.Touba," Multiple Bit Upset Tolerant Memory Using a Selective Cycle Avoidance Based SEC-DED-DAEC Code",IEEE VLSI 25th conference ,May 2007.

[7]Luis.J.Saiz-Adalid, P. Reviriego, P. Gil, S. Pontarelli, and J. A. Maestro,"MCU tolerance in SRAMs through low-redundancy triple adjacent error correction," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23,no. 10, pp. 2332–2336, Oct. 2015.

[8]R. Naseer and J. Draper, "Parallel double error correcting code design to mitigate multi-bit upsets in SRAMs," in Proc. IEEE 34th Eur. Solid-State Circuits, Sep. 2008, pp. 222–225.

[9]P.Reviriego ; S. Pontarelli ; J.A. Maestro ; M. Ottavi,"Low-cost single error correction multiple adjacent error correction codes",Journals and magazines,Vol 28.Issue 43,26 Nov,2012.

[10]Liyi Xiao, J. Li, J. Li, and J. Guo, "Hardened design based on advanced orthogonal Latin code against two adjacent multiple bit upsets (MBUs)in memories," in Proc. 16th Int. Symp. Quality Electron. Design, Mar. 2015, pp. 485–489.

[11]Saeed,Kwang-TingCheng,"Error-locality-aware linear coding to correct multi-bit upsets inSRAMs",published in IEEE International TestConference,2010

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

[13]MichealRitcher,KlausOberlaender,Micheal Goessel, "new linear SEC-DED codes with codes with reduced triple bit error miscorrection probability", 14th IEEE international presentations, 2008 selective bit placement, article in IEEE transcations on device and materials reliability,June 2012.

[14]AlfansoSanchezMacian,PedroReviriego,Juan .A.Maestro,"Enhanced detection of double and triple adjacent errors in hamming codes through Comput., vol. 22, no. 3, pp. 258–266, May/Jun. 2005.

[15]Klockmann,GeorgGeorgakos,Micheal Gossel, published 2017 in 2017 IEEE, On-line testing and robust system design (iotls)-a new 3-bit burst error correcting codes.

[16]Jiaqiang Li,Pedro Reviriego, Liyi Xiao, Costas Argyrides, and Jie Li,"Extending 3-bit burst error correction with quadruple adjacent error correction ,IEEE,transcations on Very Large Scale Interconnections(VLSI) SYSTEMS, VOL. 26, NO. 2,FEBRUARY,2018,pp221-228.