ASEAN Engineering Journal

Full Paper

MODIFIED MATRIX CODES FOR SHIELDING MEMORIES AGAINST ADJACENT ERRORS

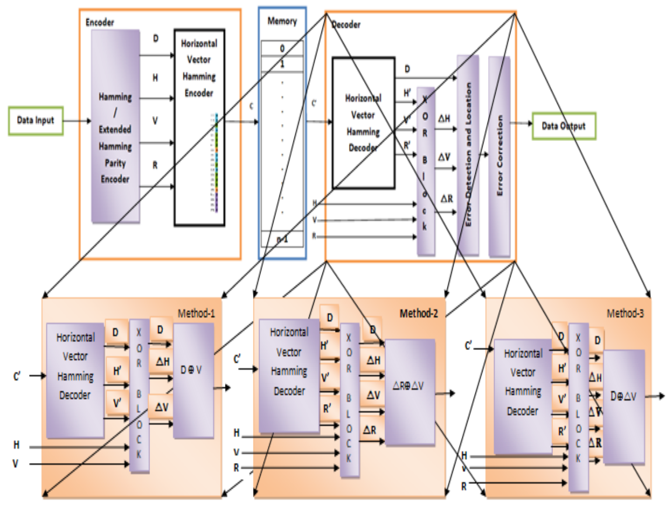
Article history
Received
13 June 2023
Received in revised form
28 September2023
Accepted
05 November 2023
Published online
31 May 2024

Neelima K[a]*, C. Subhas[b]

[a]Department of ECE, School of Engineering, Mohan Babu University erstwhile Sree Vidyanikethan Engineering College, Tirupati, India
[b]Chadalawada Ramanamma Engineering College, Tirupati, India

*Corresponding author
neelima.k@vidyanikethan.edu

Graphical abstract


Encoder
Hamming / Extended Hamming Parity Encoder
Horizontal Vector Hamming Encoder
Data Input
Memory
Decoder
Horizontal Vector Hamming Decoder
Error Detection and Location
Error Correction
Data Output
Method-1
Method-2
Method-3
XOR BLOCK

Abstract

Soft errors are caused in memories due to radiation effects as the technology scales down. This paper concentrates on correcting adjacent errors in memories using indirect decoding mechanisms. Several matrix codes were also used to correct a maximum of two adjacent and random errors. In this paper, matrix representation of two rows for half the data bits matrix representations is used and each row is encoded with extended hamming code parity bits. Three ways of decoding are proposed. Among them, the method-1 uses only extended hamming bits of rows and vertical parity bits for decoding which is capable of correcting odd number of adjacent bits in half of data bits. The method-2 uses all parity bits and is capable of correcting 1 bit less than half the data bits. The method-3 uses all parity bits and with a change in decoding mechanism allows correction of half erroneous data bits. The method-3 based decoder proves to be more reliable either in lower half or upper half of Data, enabling it to be used in image processing applications. But method-3 compromises with decrease in code rate, increase in bit overhead, area and power delay product by atleast 26.38%, 10.76%, 9.6% and 5%.

Keywords: Bit Overhead, Code Rate, Code Efficiency, Hamming Code, Matrix Codes.

1.0 INTRODUCTION

The soft errors gain prominence as the compact structures cause high radiation effects inducing random and adjacent errors. In many modern applications, there is a necessity of high reliability which encourages error detection and correction codes. The applications for which they can be used are military applications, medical applications, etc. The introduction of telemedicine to society needs higher reliability of data stored and communicated. In embedded systems customized for these applications store a lot of information by the way of encoded data [1,2,3].

The memories are the devices in embedded systems that are majorly prone to errors. Also the operating system is also stored in read only memory and the data used momentarily can be stored in random access memory [4,5]. The data that is to be stored is first encoded by parity bits and the information is stored as code word along with parity bits using write operation in encoder.

Depending upon whether ROM or RAM, the number of write operations are used. For ROM, the data is written once and read many times. But in RAM, many write and many read operations are used [6,7]. So in decoder, the code word is read and the parity bits are verified for error detection. The change or bit flips in parity predict the possibility of erroneous data. Then by using the specified algorithm, the error correction mechanism is performed to obtain error free data [8,9].

The model utilized for memory representation for error detection and correction is as shown in figure 1 where H indicates Horizontal and V indicates Vertical parity bits.

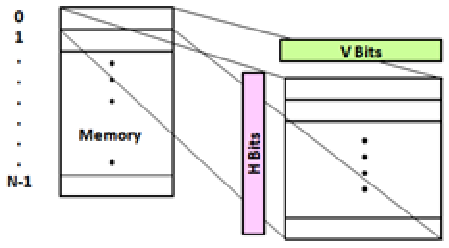
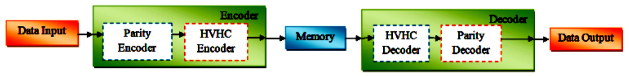


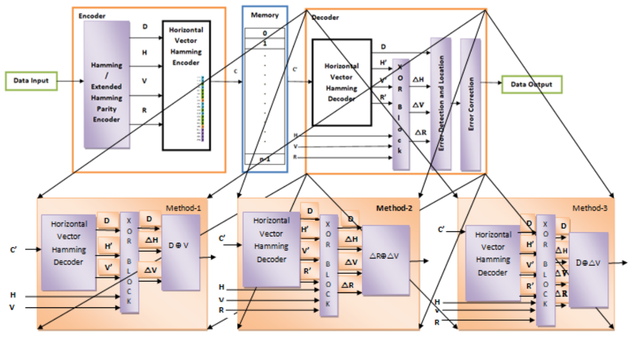
Figure 1 Matrix representation and Parity Generation Process

The basic EDAC codes [10] utilize the various blocks as shown in figure 2(a). The encoder uses two step process i.e., the HVHC (Horizontal Vector Hamming Code) Encoding and the discussed Encoders to generate code word.

This code word is written into the memory [11,12]. Using read operation, the data is decoded first by HVHC Decoder and later by Parity Decoder which verify the parity bits for correcting detected errors. The entire process of EDAC is based on Hamming codes for all the existing or proposed Parity codes in detail is shown in figure 2 (b).



(a) EDAC Process outline


Encoder
Data Input
Hamming / Extended Hamming Parity Encoder
Horizontal Vector Hamming Encoder
Memory
Decoder
Horizontal Vector Hamming Decoder
Error Detection and Location
Error Correction
Data Output
Method-1
Method-2
Method-3

(b) EDAC Process in detail

Figure 2 EDAC Process

Majorly the changes are incorporated in parity encoder and decoder blocks for EDAC Codes. The additional bits required depends on type of method adopted in encoder on the basis of Hamming Distance which aid in the development of code word that is written into memory [13,14].

Several error detection and correction codes have been proposed in literature which have gained importance in recent years. Among them the hamming and extended hamming codes are used as basis [15,16,17].

Upon these, matrix code and its modified versions show good reliability and have achieved correction capability of quad errors. Also Hsiao code, Golay Code, etc are also used for SEC-DED, SEC-DEC-TED and SEC-DEC-TEC [18,19,20]. This paper proposes modifications for decoding mechanism so as to obtain higher order reliability. Here one basic way of encoding is proposed and three versions of decoding are proposed.

2.0 METHODOLOGY

The basic mechanism of encoding that is used is as depicted in figure 1. Consider the 8-bit data being arranged as shown in figure 3. Here V[3...0] are the vertical encoded parity bits, R[5...0] are hamming bits and H[1...0] are the extended hamming code parity bits [21,22].

D_7	D_6	D_5	D_4	H_1	R_5	R_4	R_3
D_3	D_2	D_1	D_0	H_0	R_2	R_1	R_0
V_3	V_2	V_1	V_0				

Figure 3 Encoding Mechanism

The vertical parity bits are calculated as $V_i = D_i \oplus D_{i+1}$ ...(1)

The hamming parity bits are calculated as

$R_0 = D_0 \oplus D_1 \oplus D_3$ ...(2)
$R_1 = D_0 \oplus D_2 \oplus D_3$ ...(3)
$R_2 = D_1 \oplus D_2 \oplus D_3$ ...(4)
$R_3 = D_4 \oplus D_5 \oplus D_7$ ...(5)
$R_4 = D_4 \oplus D_6 \oplus D_7$ ...(6)
$R_5 = D_5 \oplus D_6 \oplus D_7$ ...(7)

The extended hamming parity bits are calculated as

$H_0 = D_0 \oplus D_1 \oplus D_2 \oplus D_3$ ...(8)
$H_1 = D_4 \oplus D_5 \oplus D_6 \oplus D_7$ ...(9)

The code word consists of 20 bits and the positions are represented as shown below

20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
V3	V2	V1	V0	H1	D7	D6	D5	R5	D4	R4	R3	H0	D3	D2	D1	R2	D0	R1	R0

The above code word is stored in memory with the specified location by write operation.

The decoder is designed by using the parity bits for error evaluation in data and correction to ensure reliability. The decoding mechanisms are developed in three ways represented by method-1, method-2 and method-3.

In method - 1, we use only the extended parity bits and vertical bits for decoding and the hamming bits remain as hidden bits and are not used. Consider the data to be "00000000" as the data to be encoded [23]. Then by using the encoding equations the corresponding parity bits are also all zeros as shown below. Then the code word is "00000000000000000000"

If the same code word is read from memory, then there is no error in data. Suppose a single bit error exists say in $D_0 = 1$, then the code word changes as "00010000000010000111". As the error is reflected in parity bits H_0 and V_0, the decoding mechanism will be as $H_0 = 1$ and V[3...0] = "0001", $D_{out}[3...0] = D_{read}[3...0] \oplus V[3...0]$ = "0000" and then $D_{out}[7...4] = D_{read}[7...4]$ then D_{out} = "00000000", which is error free data. Suppose two adjacent bits are in error say in D[1...0] = 1, then the code word changes as "00110000000000011110". As the error is reflected in parity bits V_0 and V_1, the decoding mechanism will be as H = "00" but V[3...0] ="0011", the error may exist in either upper half or lower half of data read from memory then $D_{out}[3...0]$ =

$D_{read}[3...0] \oplus V[3...0]$ = "0000" and then $D_{out}[7...4] = D_{read}[7...4] \oplus V[3...0]$ = "0011" then D_{out} = "00110000", which has induced two erroneous bits in upper half of data but corrected lower half of data.

Suppose three adjacent bits are in error say in D[2...0] = 1, then the code word changes as "01110000000010110100". As the error is reflected in parity bits H_0, V_0, V_1 and V_2, the decoding mechanism will be as H_0 = 1 and V[3...0] ="0111", then $D_{out}[3...0] = D_{read}[3...0] \oplus V[3...0]$ = "0000" and then $D_{out}[7...4] = D_{read}[7...4]$ = "0000" then D_{out} = "00000000", which is error free data.

Suppose four adjacent bits are in error say in D[3...0] = 1, then the code word changes as "11110000000001111111". As the error is reflected in parity bits V_0, V_1, V_2 and V_3, the decoding mechanism will be as H = "00" but V[3...0] ="1111", the error may exist in either upper half or lower half of data read from memory then $D_{out}[3...0] = D_{read}[3...0] \oplus V[3...0]$ = "0000" and then $D_{out}[7...4] = D_{read}[7...4] \oplus V[3...0]$ = "1111" then D_{out} = "11110000", which has induced four erroneous bits in upper half of data but corrected lower half of data [24].

From five adjacent bits onwards being in error, D_{out} = "11111111", which is completely erroneous data and induces 8 errors. The complete process is shown in table 1. From Table 1, it is clear that this method can correct either single bit error or three adjacent errors in 8-bit erroneous data read from memory.

Table 1 Decoding of data read from memory for various adjacent erroneous bits using Method-1.

D_{read}	H	V	Number of Errors Induced	D_{out}	Number of Errors Corrected
00000000	00	0000	0	00000000	0
00000001	01	0001	1	00000000	1
00000011	00	0011	2	00110000	0
00000111	01	0111	3	00000000	3
00001111	00	1111	4	11110000	0
00011111	10	1110	5	11111111	0
00111111	00	1100	6	11110011	0
01111111	10	1000	7	11111111	0
11111111	00	0000	8	11111111	0

In method - 2 tries to improve the number of bits corrected by modifying the way the erroneous bits are decoded [25,26]. The hamming bits are also used for decoding in addition to extended hamming bits and vertical parity bits. The modification here lies in using the new condition of $D_{correctedi} = \triangle r \oplus \triangle V$. if the number of errors estimated is found to be even. This enables correction of 3 - erroneous bits. Let R', H' and V' be the hamming, extended hamming and vertical parity bits calculated from data read from memory respectively. Then $\triangle R = R - R'$, $\triangle H = H - H'$ and $\triangle V = V - V'$. The data can be found from R bits as shown below

$D_0 = R_0 \oplus R_1$ ...(10)
$D_1 = R_0 \oplus R_2$ ...(11)
$D_2 = R_1 \oplus R_2$ ...(12)
$D_3 = R_0 \oplus R_1 \oplus R_2$ ...(13)
$D_4 = R_3 \oplus R_4$ ...(14)
$D_5 = R_3 \oplus R_5$ ...(15)
$D_6 = R_4 \oplus R_5$ ...(16)
$D_7 = R_3 \oplus R_4 \oplus R_5$ ...(17)

Suppose a single bit error exists say in D0 = 1, then the code word changes as "00010000000010000111". As the error is reflected in parity bits H_0', R_0', R_1' and V_0', the decoding mechanism will be as $\triangle H_0$ = 1 and $\triangle V[3...0]$ = "0001", $D_{out}[3...0] = D_{read}[3...0] \oplus \triangle V[3...0]$ = "0000" and then $D_{out}[7...4] = D_{read}[7...4]$ then D_{out} = "00000000", which is error free data.

Suppose two adjacent bits are in error say in D[1...0] = 1, then the code word changes as "00110000000000011110". As the error is reflected in parity bits R_1', R_2', V_0' and V_1', the decoding mechanism will be as $\triangle H$ = "00" but $\triangle V[3...0]$ ="0011", the error may exist in either upper half or lower half of data read from memory then verify $\triangle R$ = 000110. As only LSB bits are changed then $D_{out}[0] = D_{out}[1] = D_{out}[2] = D_{out}[3] = \triangle R[0] \oplus \triangle R[1] \oplus \triangle R[2] \oplus \triangle V[3] \oplus \triangle V[2] \oplus \triangle V[1] \oplus \triangle V[0]$ = 0 and then $D_{out}[7...4] = D_{read}[7...4]$ = "0000" then D_{out} = "00000000", which is error free data.

Suppose three adjacent bits are in error say in D[2...0] = 1, then the code word changes as "01110000000010110100". As the error is reflected in parity bits H_0', V_0', V_1' and V_2', the decoding mechanism will be as $\triangle H_0$ = 1 and $\triangle V[3...0]$ ="0111", then $D_{out}[3...0] = D_{read}[3...0] \oplus \triangle V[3...0]$ = "0000" and then $D_{out}[7...4] = D_{read}[7...4]$ = "0000" then D_{out} = "00000000", which is error free data.

Suppose four adjacent bits are in error say in D[3...0] = 1, then the code word changes as "11110000000001111111". As the error is reflected in parity bits R_0', R_1', R_2', V_0', V_1', V_2' and V_3', the decoding mechanism will be as $\triangle H$ = "00" but $\triangle V[3...0]$ ="1111", the error may exist in either upper half or lower half of data read from memory then verify $\triangle R$ = 000111. As only LSB bits are changed then $D_{out}[0] = D_{out}[1] = D_{out}[2] = D_{out}[3] = \triangle R[0] \oplus \triangle R[1] \oplus \triangle R[2] \oplus \triangle V[3] \oplus \triangle V[2] \oplus \triangle V[1] \oplus \triangle V[0]$ = 1 and then $D_{out}[7...4] = D_{read}[7...4]$ = "0000" then D_{out} = "00001111", which induces 4 errors in data.

From five adjacent bits onwards being in error, the D_{out} can't correct errors and instead this method induces errors. The complete process is shown in table 2. From Table 2, it is clear that this method can correct up to three adjacent errors in 8-bit erroneous data read from memory.

Table 2 Decoding of data read from memory for various adjacent erroneous bits using Method-2.

D_{read}	H	R	V	Number of Errors Induced	D_{out}	Number of Errors Corrected
00000000	00	000000	0000	0	00000000	0
00000001	01	000011	0001	1	00000000	1
00000011	00	000110	0011	2	00000000	2
00000111	01	000000	0111	3	00000000	3
00001111	00	000111	1111	4	00001111	0
00011111	10	011111	1110	5	11111111	0
00111111	00	110111	1100	6	00001111	0
01111111	10	000111	1000	7	11110000	0
11111111	00	111111	0000	8	11111111	0

The method - 3 tries to improve the number of bits corrected by modifying the way the erroneous bits are decoded. This also uses hamming bits for decoding in addition to extended hamming bits and vertical parity bits. The modification here lies in using the new condition of verifying $\triangle R$ and then $D_{corrected} = D_{read} \oplus \triangle V$ if the number of errors estimated is found to be even. This enables correction of 4 - erroneous bits.

Let R', H' and V' be the hamming, extended hamming and vertical parity bits calculated from data read from memory respectively. Then △R = R – R', △H = H – H' and △V = V – V'.

Suppose a single bit error exists say in D_0 = 1, then the code word changes as "00010000000010000111". As the error is reflected in parity bits H_0', R_0', R_1' and V_0', the decoding mechanism will be as △H0 = 1 and △V[3...0] = "0001", D_{out}[3...0] = D_{read}[3...0] ⊕ △V[3...0] = "0000" and then D_{out}[7...4] = D_{read}[7...4] then D_{out} = "00000000", which is error free data.

Suppose two adjacent bits are in error say in D[1...0] = 1, then the code word changes as "00110000000000011110". As the error is reflected in parity bits R_1', R_2', V_0' and V_1', the decoding mechanism will be as △H = "00" but △V[3...0] ="0011", the error may exist in either upper half or lower half of data read from memory then verify △R = 000110. As only LSB bits are changed then D_{out}[3...0] = D_{read}[3...0] ⊕ △V[3...0] = "0000" and then D_{out}[7...4] = D_{read}[7...4] = "0000" then D_{out} = "00000000", which is error free data.

Suppose three adjacent bits are in error say in D[2...0] = 1, then the code word changes as "01110000000010110100". As the error is reflected in parity bits H_0', V_0', V_1' and V_2', the decoding mechanism will be as △H_0 = 1 and △V[3...0] ="0111", then D_{out}[3...0] = D_{read}[3...0] ⊕ △V[3...0] = "0000" and then D_{out}[7...4] = D_{read}[7...4] = "0000" then D_{out} = "00000000", which is error free data.

Suppose four adjacent bits are in error say in D[3...0] = 1, then the code word changes as "11110000000001111111". As the error is reflected in parity bits R_0', R_1', R_2', V_0', V_1', V_2' and V_3', the decoding mechanism will be as △H = "00" but △V[3...0] ="1111", the error may exist in either upper half or lower half of data read from memory then verify △R = 000111. As only LSB bits are changed then D_{out}[3...0] = D_{read}[3...0] ⊕ △V[3...0] = "0000" and then D_{out}[7...4] = D_{read}[7...4] = "0000" then D_{out} = "00000000", which is error free data.

From five adjacent bits onwards being in error, the D_{out} can't correct errors and instead this method induces errors. The complete process is shown in table 3. From Table 3, it is clear that this method can correct up to four adjacent errors in 8-bit erroneous data read from memory.

Table 3 Decoding of data read from memory for various adjacent erroneous bits using Method-3.

D_{read}	H	R	V	Number of Errors Induced	D_{out}	Number of Errors Corrected
00000000	00	000000	0000	0	00000000	0
00000001	01	000011	0001	1	00000000	1
00000011	00	000110	0011	2	00000000	2
00000111	01	000000	0111	3	00000000	3
00001111	00	000111	1111	4	00000000	4
00011111	10	011111	1110	5	11110001	0
00111111	00	110111	1100	6	11110011	0
01111111	10	000111	1000	7	11110111	0
11111111	00	111111	0000	8	11111111	0

3.0 RESULTS AND DISCUSSION

The designs are modeled in Verilog HDL and are verified in Xilinx ISE 14.5 Tool for 28nm Zynq FPGA with part number XC7Z100-2FFG1156.

The assessment of these methods is done for 8, 16, 32 and 64 – Bit Data. The comparison is as shown in table 4.

From table 4(a), method-3 of decoding is capable of correcting a maximum of 4 adjacent errors i.e., either in upper half or lower half of data read from memory with a code rate of 40%.

Table 4(a) Comparison Table for Decoding Methods for 8-Bit Data

Parameter/ Methods	Method-1	Method-2	Method-3
# Data Bits, k	8	8	8
# Parity Bits, r	6	12	12
# Code Word, n=k+r	14	20	20
Bit Overhead, r/k	0.75	1.5	1.5
Code Rate, k/n	0.57	0.4	0.4
Code Efficiency, r/n	0.43	0.6	0.6
Correction Capability	Only odd number up to 3 Adjacent Errors	Up to 3 Adjacent Errors	Up to 4 Adjacent Errors

From table 4(b), method-3 of decoding is capable of correcting a maximum of 8 adjacent errors i.e., either in upper half or lower half of data read from memory with a code rate of 47%.

Table 4(b) Comparison Table for Decoding Methods for 16-Bit Data

Parameter/ Methods	Method-1	Method-2	Method-3
# Data Bits, k	16	16	16
# Parity Bits, r	10	18	18
# Code Word, n=k+r	26	34	34
Bit Overhead, r/k	0.625	1.125	1.125
Code Rate, k/n	0.615	0.47	0.47
Code Efficiency, r/n	0.385	0.53	0.53
Correction Capability	Only odd number up to 7 Adjacent Errors	Up to 7 Adjacent Errors	Up to 8 Adjacent Errors

From table 4(c), method-3 of decoding is capable of correcting a maximum of 16 adjacent errors i.e., either in upper half or lower half of data read from memory with a code rate of 53%.

Table 4(c) Comparison Table for Decoding Methods for 32-Bit Data

Parameter/ Methods	Method-1	Method-2	Method-3
# Data Bits, k	32	32	32
# Parity Bits, r	18	28	28
# Code Word, n=k+r	50	60	60
Bit Overhead, r/k	0.56	0.875	0.875
Code Rate, k/n	0.64	0.53	0.53
Code Efficiency, r/n	0.36	0.47	0.47
Correction Capability	Only odd number up to 15 Adjacent Errors	Up to 15 Adjacent Errors	Up to 16 Adjacent Errors

From table 4(d), method-3 of decoding is capable of correcting a maximum of 32 adjacent errors i.e., either in upper half or lower half of data read from memory with a code rate of 58%.

Table 4(d) Comparison Table for Decoding Methods for 64-Bit Data

Parameter/ Methods	Method-1	Method-2	Method-3
# Data Bits, k	64	64	64
# Parity Bits, r	34	46	46
# Code Word, n=k+r	98	110	110
Bit Overhead, r/k	0.53	0.72	0.72
Code Rate, k/n	0.65	0.58	0.58
Code Efficiency, r/n	0.35	0.42	0.42
Correction Capability	Only odd number up to 31 Adjacent Errors	Up to 31 Adjacent Errors	Up to 32 Adjacent Errors

The bit overhead is less for decoding using method-1 and the other two methods share the same overhead by 50%, 44.44%, 36% and 26.38% for 8, 16, 32 and 64 bit data respectively. Even though the code rate is optimal, method-3 proves to be a better choice as the number of bits that can be corrected is N/2 in an N-bit Data. The code rate improves as the number of data bits are increased and has improved from 40% to 58% i.e., an improvement of 31%.

The simulation Results are shown in figures 4, 5 and 6 for the three methods of decoding respectively. The results show that the method-1 corrects only odd number of N/2 adjacent errors. The method-2 corrects up to N/2 -1 adjacent errors and the method-3 corrects up to N/2 Adjacent Errors in an N-Bit Data.

From figure 4, if the data is “00000000”, then only 1 or 3 bits are corrected, i.e., for “00000001” and “00000111”, the correct output is obtained by using method-1 of decoding.

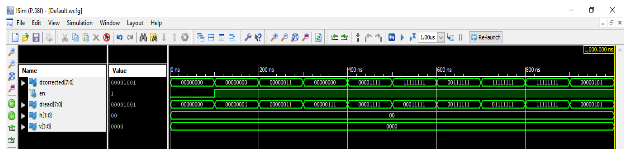


Figure 4 Simulation Result of Method – 1

From figure 5, if the data is “00000000”, then only 1, 2 or 3 bits are corrected, i.e., for “00000001”, “00000011” and “00000111”, the correct output is obtained by using method-2 of decoding.

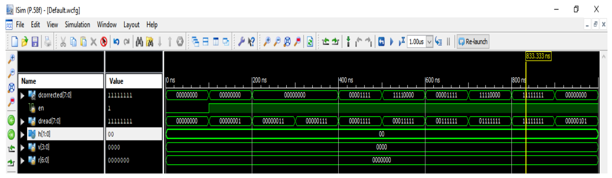


Figure 5 Simulation Result of Method – 2

From figure 6, if the data is “00000000”, then only 1, 2, 3 or 4 bits are corrected, i.e., for “00000001”, “00000011”, “00000111” and “00001111”, the correct output is obtained by using method-3 of decoding.

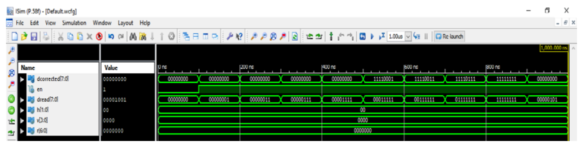


Figure 6 Simulation Result of Method – 3

Further the complexity of developed Encoders and Decoders are evaluated in terms of area and power-delay product as shown in figures 7 and 8 respectively.

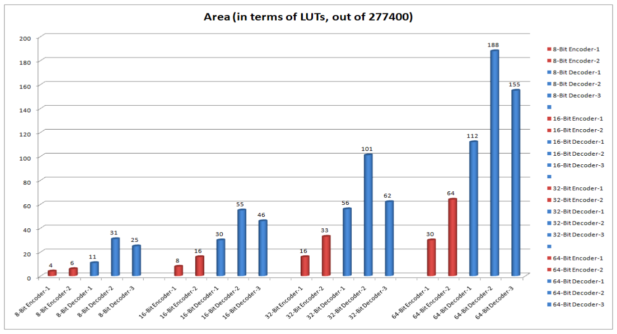


Figure 7 The simulation result of area occupied in terms of LUTs for the developed encoders and decoders

In figure 7, the red colour bars represent the encoders without hamming bits and with hamming bits. Also the blue colour bars represent the decoders using method-1 corresponding to encoder-1, method-2 and method-3 corresponding to encoder-2 respectively. It shows that both in encoder and decoder, the area is increased to increase the reliability in correcting the adjacent errors.

The decoder-3 shows optimum results in terms of area for various data sizes by atleast 16.36% reduction in area occupied when compared to decoder-2 and increase in area occupied by a minimum of 9.6% to a maximum of 56%. So area remains a compromise for higher order reliability.

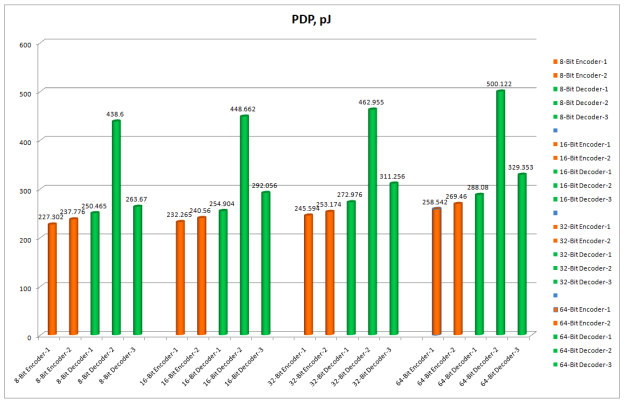


Figure 8 The simulation result of power delay product for the developed encoders and decoders

In figure 8, the orange color bars represent the encoders without hamming bits and with hamming bits. Also the green color bars represent the decoders using method-1 corresponding to encoder-1, method-2 and method-3 corresponding to encoder-2 respectively. It shows that both in encoder and decoder, the power delay product has increased slightly to increase the reliability in correcting the adjacent errors.

The decoder-3 shows optimum results in terms of power delay product for various data sizes by atleast 32.77% to 39.88% reduction when compared to decoder-2 and increase by a minimum of 5% to a maximum of 12.72%. So power delay product also increases slightly to achieve higher order reliability.

4.0 CONCLUSION

As the technology scales down, the soft errors are caused in memories due to radiation effects. This paper concentrates on correcting adjacent errors in memories using indirect decoding mechanisms. The hamming, extended hamming and vertical parity bits are used in correcting errors. Three decoding mechanisms are proposed. The designs are modelled in Verilog HDL and are verified in Xilinx ISE 14.5 Tool for 28nm Zynq FPGA with part number XC7Z100-2FFG1156. The assessment of these methods is done for 8, 16, 32 and 64 – Bit Data. The method-3 based decoder proves to be more reliable which is capable of correcting N/2 adjacent errors either in lower half or upper half of N-Bit Data enabling it to be used in image processing applications. But method-3 compromises with decrease in code rate, increase in bit overhead, area and power delay product by atleast 26.38%, 10.76%, 9.6% and 5%.

Acknowledgement

This research is supported by JNTUA Ananthapuramu.

References

[1] A. S.-Macián, P. Reviriego, J. A. Maestro. 2014. Hamming SEC-DAED and Extended Hamming SEC-DED-TAED Codes Through Selective Shortening and Bit Placement. *IEEE Transactions on Device and Materials Reliability*, 14(1): 574-576, March. DOI: 10.1109/TDMR.2012.2204753.

[2] S. Tambatkar, S. N. Menon, V. Sudarshan, M. Vinodhini, N. S. Murty. 2017. Error detection and correction in semiconductor memories using 3D parity check code with hamming code. *2017 International Conference on Communication and Signal Processing (ICCSP)*, 0974-0978. DOI: 10.1109/ICCSP.2017.8286516.

[3] K. Neelima, C. Subhas. 2020. Efficient Adjacent 3D Parity Error Detection and Correction Codes for Embedded Memories. *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 1-5. DOI: 10.1109/CONECCT50063.2020.9198452.

[4] S. Sharma, P. Vijayakumar. 2012. An HVD based error detection and correction of soft errors in semiconductor memories used for space applications. *2012 International Conference on Devices, Circuits and Systems (ICDCS)*, 563-567. DOI: 10.1109/ICDCSyst.2012.6188771.

[5] V. Badole, A. Udawat. 2014. Implementation of multidirectional parity check code using hamming code for error detection and correction. *International Journal of Research in Advent Technology*, 2: 1-6. DOI: https://1library.net/document/zx25v8oq-implementation-multidirectional-parity-check-using-hamming-detection-correction.html.

[6] M. S. Rahman, M. S. Sadi, S. Ahammed, J. Jurjens. 2015. Soft error tolerance using Horizontal-Vertical-Double-Bit Diagonal parity method. *2015 International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*, 1-6. DOI: 10.1109/ICEEICT.2015.7307411.

[7] L. J. Saiz-Adalid, P. Gil, J. C. Ruiz, J. Gracia-Morán, D. Gil-Tomás, J.-C. Baraza-Calvo. 2016. Ultrafast Error Correction Codes for Double Error Detection/Correction. *2016 12th European Dependable Computing Conference (EDCC)*, 108-119. DOI: 10.1109/EDCC.2016.28.

[8] K. Neelima, C. Subhas. 2019. Multiple Adjacent Bit Error Detection and Correction Codes for Reliable Memories: A Review. *Conference Advances in Communications, Signal Processing, and VLSI. Lecture Notes in Electrical Engineering*, 357-371. DOI: 10.1007/978-981-33-4058-9_32.

[9] J. Athira, B. Yamuna. 2018. FPGA Implementation of an Area Efficient Matrix Code with Encoder Reuse Method. *2018 International Conference on Communication and Signal Processing (ICCSP)*, 0254-0257. DOI: 10.1109/ICCSP.2018.8524371.

[10] A. J. Olazábal, J. P. Guerra. 2019. Multiple Cell Upsets Inside Aircrafts. New Fault-Tolerant Architecture. *IEEE Transactions on Aerospace and Electronic Systems*, 55(1): 332-342. DOI: 10.1109/TAES.2018.2852198.

[11] J. Samanta, J. Bhaumik, S. Barman. 2019. Compact and power efficient SEC-DED codec for computer memory. *Microsystem Technologies*, 27: 359-368, Feb. DOI:10.1007/s00542-019-04366-7.

[12] K. N. Dang, X. T. Tran. 2019. An Adaptive and High Coding Rate Soft Error Correction Method in Network-on-Chips. *VNU Journal Of Science: Computer Science And Communication Engineering*, 35(1): 32–45. DOI:10.25073/2588-1086/vnucsce.218.

[13] A. Radonjic, V. Vujicic. 2019. Integer codes correcting sparse byte errors. *Cryptography and Communications*, 11(5): 1069-1077, Sep. DOI: 10.1007/s12095-019-0350-9.

[14] A. Subbiah, T. Ogunfunmi. 2019. A Flexible Hybrid BCH Decoder for Modern NAND Flash Memories Using General Purpose Graphical Processing Units (GPGPUs). *Micromachines*, 10(6): 1-15. May. DOI: 10.3390/mi10060365.

[15] S. Lin, K. A. Ghaffar, J. Li, K. Liu. 2020. A Scheme for Collective Encoding and Iterative Soft-Decision Decoding of Cyclic Codes of Prime Lengths: Applications to Reed–Solomon, BCH, and Quadratic Residue Codes. *IEEE Transactions on Information Theory*, 66(9): 5358-5378. DOI: 10.1109/TIT.2020.2978383.

[16] U. S. Himaja, M. Vinodhini, N. S. Murty. 2018. Multi-Bit Low Redundancy Error control with Parity Sharing for NoC Interconnects. *2018 3rd International Conference on Communication and Electronics Systems (ICCES)*, 61-65. DOI: 10.1109/CESYS.2018.8724118.

[17] A. Das, N. A. Touba. 2018. Low complexity burst error correcting codes to correct MBUs in SRAMs. *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, 219-224. DOI: 10.1145/3194554.3194570.

[18] S. Divya Lakshmi, K. Neelima, C. Subhas. 2021. Proficient Matrix Codes for NOC Applications. 2021 International Symposium on Devices, Circuits and Systems (ISDCS), 1-4. DOI: 10.1109/ISDCS52006.2021.9397914.

[19] J. Guo, L. Xiao, Z. Mao, Q. Zhao. 2014. Enhanced Memory Reliability Against Multiple Cell Upsets Using Decimal Matrix Code. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 22(1): 127-135, Jan. DOI: 10.1109/TVLSI.2013.2238565.

[20] Neelima, K., Subhas, C. 2022. Half Diagonal Matrix Codes for Reliable Embedded Memories. International Journal of Health Sciences, 6(S2): 11664 – 11677. DOI: https://doi.org/10.53730/ijhs.v6nS2.8117

[21] Ms. Gunduru Swathi Lakshmi, Ms. Neelima K, Dr. C. Subhas. 2019. Error Detection and Correction Methods for Memories used in System-on-Chip Designs. *International Journal of Engineering and Advanced Technology (IJEAT),* ISSN: 2249 – 8958, 8(2S2): 60-66. DOI: B10140182S219/19©BEIESP.

[22] Nimisha N, P Rajkumar , S Rajkumar. 2020. Error Correction Codes using Burst and Random Errors for Multiple Cell Upsets in Space Application. *International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering,* 8(5): 30-45. May. DOI: 10.17148/IJIREEICE.2020.8507.

[23] Neelima Koppala, Chennapalli Subhas. 2022. Low Overhead Optimal Parity Codes. *Telkomnika (Telecommunication Computing Electronics and Control)*, 20(3): 501-509, ISSN: 1693-6930. DOI: 10.12928/TELKOMNIKA.v20i3.23301.

[24] S. Manoj, C. Babu. 2016. Improved error detection and correction for memory reliability against multiple cell upsets using DMC & PMC. 2016 IEEE Annual India Conference (INDICON), 1-6. DOI: 10.1109/INDICON.2016.7839094.

[25] Gobinda Prasad Acharya, Muddapu Asha Rani, Ganjikunta Ganesh

Kumar, Lavanya Poluboina. 2022. Adaptation of March -SS Algorithm to word-oriented memory built-in self-test and repair. Indonesian Journal of Electrical Engineering and Computer Science, 26(1): 96-104. DOI: 10.11591/ijeecs.v26.i1.pp96-104.

[26] Neelima K, C. Subhas, 2023. Modified Proficient Adjacent Error Correcting Codes. e-Prime - *Advances in Electrical Engineering, Electronics and Energ*y, 5: 100277, ISSN 2772-6711, https://doi.org/10.1016/j.prime.2023.100277.