

Low overhead optimal parity codes

Neelima Koppala¹, Chennapalli Subhas²

¹Research Scholar, Department of ECE, JNTUA, Ananthapuramu, India

²Professor and Head of ECE, Vice-Principal, JNTUA College of Engineering, Kalikiri, India

Article Info

Article history:

Received Jun 12, 2021

Revised Mar 21, 2022

Accepted Mar 29, 2022

Keywords:

Adjacent error

Bit overhead

Code rate

Matrix code

Parity bits

ABSTRACT

The error detecting and correcting codes are used in critical applications like in intensive care units, defense applications, and require highly reliable data. This brief focuses on codes to detect and correct adjacent errors within a single clock cycle by using modulo-2 addition of data bits for parity generation, syndrome calculation, error location identification and correction by improving code rate and minimizing bit overhead. The optimal parity codes devised can correct odd number of adjacent errors upto $(N/2)-1$ data bits when compared with the existing codes with less delay. Four optimal codes are proposed using existing decimal matrix codes properties. The proposed codes prove better in terms of area, delay, power, bit overhead, code rate, code efficiency, and with good reliability. The components are developed in veriloghardware description language and verified for zynq 7000 series field programmable gate array in xilinx integrated synthesis environment 14.5 Tool. Among the codes devised, optimal code-4 proves to be a better code with 65.3% code rate and 53.12% bit overhead. Also when compared with other codes, it uses 33.3% less area and 1.89% less power delay product for encoder and 32.2% less area and 0.36% power delay product for decoder respectively.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Neelima Koppala

Research Scholar, Department of ECE, JNTUA

Ananthapuramu, India

Email: neelumtech17@gmail.com

1. INTRODUCTION

In the applications like intensive care units where continuous evaluation and back up of data extracted from monitoring devices is required for patients, fast and reliable error detecting and correcting codes have attracted research [1]-[4]. Even in military applications, the backup data is usually necessary for missile launch and guidance [5]-[8]. This attracted a great attention for reliable codes capable of operating at high frequencies [9]-[12].

The recent developments and advanced research at various levels of abstraction induced hope of using very small area specifically allocated in each embedded memory [13]-[15] with built-in self test capability [16]-[19]. But the promising future of electronics at nanoscale allows the devices get closer where the devices undergo radiation effects giving rise to multiple event upsets that induce soft errors in memories [20]-[23]. For embedded memories, the correction capability must be as high as possible within a single clock cycle to ensure at-speed testing [24]-[28]. This paper addresses the correction of maximum number of erroneous bits as fast as possible by using minimum number of redundant bits and maximum code rate.

The basic error detection and correction codes utilize the various blocks as shown in Figure 1. The encoder uses two step processes i.e., the horizontal vector hamming code encoding and the discussed encoders to generate code word. This code word is written into the memory [29], [30]. Using read operation,

the data is decoded first by horizontal vector hamming code decoder and later by parity decoder which verify the parity bits for correcting detected errors. The entire process of error detection and correction is based on Hamming codes for all the existing or proposed parity codes [29], [30].

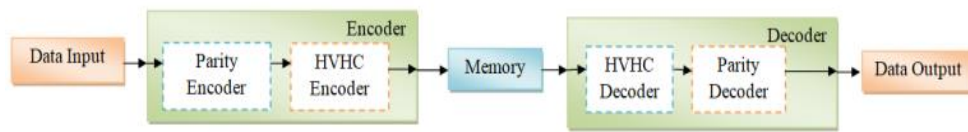


Figure 1. Error detection and correction process

Majorly the changes are incorporated in parity encoder and decoder blocks for error detection and correction 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. To obtain efficient error detection and correction codes, the decoder block is optimized with respect to area, delay, and power, this paper proposes four new codes, as matrix codes are quite faster than other codes and are easy to incorporate for memory. In this paper, the process represented in Figure 1 is used and the codes either existing or proposed are used in the place of parity encoder and decoder blocks only. The paper is structured as the accessible codes in section 2 which prove to correct a maximum of 32 adjacent bits but use large number of parity bits with less code rate. To overcome this, the proposed error detection and correction codes i.e., the proposed solution as four optimal parity codes in section 3 are discussed. The section 4 deals with simulation results and comparison in terms of assessed parameters followed by conclusion.

2. METHOD

The method adopted is represented in two sections 2.1 as accessible codes which describe the mechanism of encoding and decoding adopted for proposed codes. Also the proposed codes are described in section 2.2 which are optimised for less number of parity bits. Further the designs are considered for evaluating technological parameters such as area, power and delay, nontechnological parameters such as code rate and bit overhead.

2.1. Accessible error detection and correction codes

In prose, several error detection and correction codes are projected, among them the easiest way of correcting errors is done by treating data to be written into memory as matrix and developing the redundant bits. Usually the data along with parity bits is written into memory and later extracted from memory by read operation [29]. The [2] uses the H, V, and D-Bits i.e., the horizontal, vertical and diagonal redundant bits respectively as in 3D parity codes [8]. Figure 2 represents 64-bit data in the form of matrix with 8 rows × 8 columns giving rise to 8-H, 8-V and 15-D Bits. These parity bits are developed by modulo-2 additions of data bits read from memory. In the decoder, if an error exists, then the combination of all H, V, and D Bits indicate the location where the data bit is corrected by flipping it. Hence usually decoder requires a complex circuit when compared to encoder. The D bits are used for location of error exactly in the memory. The deviation in H bits decides the existence of errors and the V bits indicate the maximum number of adjacent bits that are correctable for the given data by using ultrafast decoding scheme [6], [7] for a correction capability of maximum of 8 adjacent bits out of 64-data bits. Say if m₂₉ is erroneous, then it is reflected in h₄, v₆, and d₆ bits which deviate from the actual encoded parity bits, by using syndrome bits the data are corrected.

	V ₁	V ₂	V ₃	V ₄	V ₅	V ₆	V ₇	V ₈	
h ₁	m ₀	m ₁	m ₂	m ₃	m ₄	m ₅	m ₆	m ₇	
h ₂	m ₈	m ₉	m ₁₀	m ₁₁	m ₁₂	m ₁₃	m ₁₄	m ₁₅	d ₁
h ₃	m ₁₆	m ₁₇	m ₁₈	m ₁₉	m ₂₀	m ₂₁	m ₂₂	m ₂₃	d ₂
h ₄	m ₂₄	m ₂₅	m ₂₆	m ₂₇	m ₂₈	m ₂₉	m ₃₀	m ₃₁	d ₃
h ₅	m ₃₂	m ₃₃	m ₃₄	m ₃₅	m ₃₆	m ₃₇	m ₃₈	m ₃₉	d ₄
h ₆	m ₄₀	m ₄₁	m ₄₂	m ₄₃	m ₄₄	m ₄₅	m ₄₆	m ₄₇	d ₅
h ₇	m ₄₈	m ₄₉	m ₅₀	m ₅₁	m ₅₂	m ₅₃	m ₅₄	m ₅₅	d ₆
h ₈	m ₅₆	m ₅₇	m ₅₈	m ₅₉	m ₆₀	m ₆₁	m ₆₂	m ₆₃	d ₇
		d ₁₅	d ₁₄	d ₁₃	d ₁₂	d ₁₁	d ₁₀	d ₉	d ₈

Figure 2. 3D parity check code (8×8 matrix arrangement)

Neelima and Subhas [3], the existing 8×8 matrix representation is changed to 4×16 matrix representation by using matrix reordering technique to enable the consideration of 16 bits at a time for processing of data. The 4×16 matrix uses 39 redundant bits i.e., 4 H bits, 16 V bits, and 19 D bits. From Figure 3, say if m29 is erroneous, then it is reflected in h2, v14 and d4 parity bits which after comparison with the encoded parity bits corrected by using syndrome bits, for a correction capability of maximum of 16 adjacent bits for 64-data bits.

	V ₁	V ₂	V ₃	V ₄	V ₅	V ₆	V ₇	V ₈	V ₉	V ₁₀	V ₁₁	V ₁₂	V ₁₃	V ₁₄	V ₁₅	V ₁₆	
h ₁	m ₀	m ₁	m ₂	m ₃	m ₄	m ₅	m ₆	m ₇	m ₈	m ₉	m ₁₀	m ₁₁	m ₁₂	m ₁₃	m ₁₄	m ₁₅	
h ₂	m ₁₆	m ₁₇	m ₁₈	m ₁₉	m ₂₀	m ₂₁	m ₂₂	m ₂₃	m ₂₄	m ₂₅	m ₂₆	m ₂₇	m ₂₈	m ₂₉	m ₃₀	m ₃₁	d ₁
h ₃	m ₃₂	m ₃₃	m ₃₄	m ₃₅	m ₃₆	m ₃₇	m ₃₈	m ₃₉	m ₄₀	m ₄₁	m ₄₂	m ₄₃	m ₄₄	m ₄₅	m ₄₆	m ₄₇	d ₂
h ₄	m ₄₈	m ₄₉	m ₅₀	m ₅₁	m ₅₂	m ₅₃	m ₅₄	m ₅₅	m ₅₆	m ₅₇	m ₅₈	m ₅₉	m ₆₀	m ₆₁	m ₆₂	m ₆₃	d ₃
	d ₁₉	d ₁₈	d ₁₇	d ₁₆	d ₁₅	d ₁₄	d ₁₃	d ₁₂	d ₁₁	d ₁₀	d ₉	d ₈	d ₇	d ₆	d ₅	d ₄	

Figure 3. 3D parity check code (4×16 matrix arrangement)

	V ₁	V ₂	V ₃	V ₄	V ₅	V ₆	V ₇	V ₈	V ₉	V ₁₀	V ₁₁	V ₁₂	V ₁₃	V ₁₄	V ₁₅	V ₁₆	V ₁₇	V ₁₈	V ₁₉	V ₂₀	V ₂₁	V ₂₂	V ₂₃	V ₂₄	V ₂₅	V ₂₆	V ₂₇	V ₂₈	V ₂₉	V ₃₀	V ₃₁	V ₃₂	
h ₁	m ₀	m ₁	m ₂	m ₃	m ₄	m ₅	m ₆	m ₇	m ₈	m ₉	m ₁₀	m ₁₁	m ₁₂	m ₁₃	m ₁₄	m ₁₅	m ₁₆	m ₁₇	m ₁₈	m ₁₉	m ₂₀	m ₂₁	m ₂₂	m ₂₃	m ₂₄	m ₂₅	m ₂₆	m ₂₇	m ₂₈	m ₂₉	m ₃₀	m ₃₁	
h ₂	m ₃₂	m ₃₃	m ₃₄	m ₃₅	m ₃₆	m ₃₇	m ₃₈	m ₃₉	m ₄₀	m ₄₁	m ₄₂	m ₄₃	m ₄₄	m ₄₅	m ₄₆	m ₄₇	m ₄₈	m ₄₉	m ₅₀	m ₅₁	m ₅₂	m ₅₃	m ₅₄	m ₅₅	m ₅₆	m ₅₇	m ₅₈	m ₅₉	m ₆₀	m ₆₁	m ₆₂	m ₆₃	d ₁
	d ₃₃	d ₃₂	d ₃₁	d ₃₀	d ₂₉	d ₂₈	d ₂₇	d ₂₆	d ₂₅	d ₂₄	d ₂₃	d ₂₂	d ₂₁	d ₂₀	d ₁₉	d ₁₈	d ₁₇	d ₁₆	d ₁₅	d ₁₄	d ₁₃	d ₁₂	d ₁₁	d ₁₀	d ₉	d ₈	d ₇	d ₆	d ₅	d ₄	d ₃	d ₂	

Figure 4. 3D parity check code (2×32 matrix arrangement)

In Figure 4, the 2×32 matrix uses 67 parity bits i.e., 2 H bits, 32 V bits and 33D bits for 3D parity encoder. Say if m29 is erroneous, then it is reflected in h1, v30, and d3 bits which after comparison with the original redundant bits, is flipped for correct data output, for a correction capability of maximum of 32 adjacent errors for 64-bit data. In [4], a similar concept with H, V, forward and backward diagonal parity bits which increases the number of parity bits and requires a complex decoder for identification of candidate bit for a correction capability of maximum of 3 adjacent bits for 32-data bits. In [5], multidirectional parity code, similar to 3D codes uses syndrome for correction, thus reducing the decoder complexity comparatively for a correction capability of maximum of 7 adjacent bits for 32-data bits. In [6], horizontal vertical double diagonal parity code requires few refinement steps to be added to identify the candidate bits but still adds complexity in decoding process due to additional refinement steps, for a correction capability of maximum of 3 adjacent bits for 32-data bits.

2.2. Optimal error detection and correction codes

The matrix codes can be optimized by using minimal number of parity bits for a 2×32 matrix representation. This algorithm uses the same process of encoding and decoding as that of decimal matrix code. The algorithms use vector adjustment, $S=\{S,32'b0\}$ for most significant bit data correction and $S=\{32'b0,S\}$ for least significant bit data correction, for a correction capability of maximum of 31 adjacent bits out of 64 data bits.

<p>Inputs: Enable, datain</p> <p>Outputs: H, V</p> <p>If Enable = 1 then</p> $H[8:0] = \text{datain}[7:0] + \text{datain}[23:16]$ $H[17:9] = \text{datain}[15:8] + \text{datain}[31:24]$ $H[26:18] = \text{datain}[39:32] + \text{datain}[55:48]$ $H[35:27] = \text{datain}[47:41] + \text{datain}[63:56]$ $V_i = \text{datain}_i \oplus \text{datain}_{i+32} \quad \text{where } i = 0,1, \dots,31$ <p>Else</p> $H = 0 \text{ and } V = 0$	<p>Inputs: Enable, H, V, dataread</p> <p>Output: dataout</p> <p>If Enable = 1 then</p> $HD[8:0] = \text{dataread}[7:0] + \text{dataread}[23:16]$ $HD[17:9] = \text{dataread}[15:8] + \text{dataread}[31:24]$ $HD[26:18] = \text{dataread}[39:32] + \text{dataread}[55:48]$ $HD[35:27] = \text{dataread}[47:41] + \text{dataread}[63:56]$ $VD_i = \text{dataread}_i \oplus \text{dataread}_{i+32} \quad \text{where } i = 0,1, \dots,31$ <p>Syndrome Calculation</p> $Hdiff = HD \oplus H$ $S = VD \oplus V$ <p>Error Location and Correction</p> <p>If Hdiff = 0 and S = 0 then</p> $\text{dataout} = \text{dataread}$ <p>Else</p> $\text{dataout} = \text{dataread} \oplus S$ <p>Else</p> $\text{dataout} = 0$
---	---

(a)

(b)

Figure 5. Optimal parity code-1: (a) encoder and (b) decoder

The optimal parity code-1 is developed based on the decimal matrix based code except that the redundant memory details are not used for error correction as shown in Figure 5, where Figure 5(a) is encoder and Figure 5(b) is decoder. The total numbers of redundant bits are $36+32=68$ bits for 64-bits of data. In encoder, the H-bits are calculated by addition and vertical bits are calculated by xoring the input data bits, which are given as inputs to decoder. The encoder reuse technique is used to calculate the decoder horizontal decoded and vertical decoded-bits respectively. Then they are xored with the actual encoded horizontal and vertical bits for syndrome. If difference in horizontal bits and syndrome are zero, then the output data bits are same as that of data read from the memory else the erroneous bits are corrected by modulo-2 operations on data read from the memory and syndrome. This method can correct odd number of adjacent errors i.e., upto $(N/2)-1$ in N-bit data read from memory. So here as S has only 32 bits only either half of data bits can be corrected and the other half of the bits remain unchanged. For example, consider the $data_{in} = 64'b0$ then outputs H and V will be $36'b0$ and $32'b0$ respectively. In decoder say:

The $data_{read} = 64'b11111111111111111111111111111111111100000000000000000000000000$

i.e., 31 bits of error in most significant bit resulting in a large variation of value read from memory. Then,

Horizontal decoded bit = $36'b0000000000000001000000000000000000$

and

Vertical decoded bit = $32'b111111111111111111111111111110$.

As $H = 36'b0$ and $V = 32'b0$, then

$H_{diff} = 36'b00000000000000010000000000000000$

and

$S = \{32'b111111111111111111111111111110, 32'b0\}$

then $data_{out} = data_{read} \oplus S = 64'b0$.

Similarly, the optimal parity code-2 as shown in Figure 6, where Figure 6(a) is encoder and Figure 6(b) is decoder. Figure 6 is based on modification of optimal parity code-1 except that the change is in the number of H-bits and its calculation by 16-bit addition. Here 34 horizontal bits and 32 vertical bits are required i.e., 66 parity bits are required for 64-bits of data. For example, consider the $data_{in} = 64'b0$ then outputs H and V will be $34'b0$ and $32'b0$ respectively. In decoder say:

The $data_{read} = 64'b11111111111111111111111111111111111100000000000000000000000000$

i.e., 31 bits of error in most-significant bit (MSB) resulting in a large variation of value read from memory. Then,

Horizontal decoded bit = $34'b0000000000000001000000000000000000$

and

Vertical decoded bit = $32'b111111111111111111111111111110$.

As $H = 34'b0$ and $V = 32'b0$, then

$H_{diff} = 34'b00000000000000010000000000000000$

and

$S = \{32'b111111111111111111111111111110, 32'b0\}$

then $data_{out} = data_{read} \oplus S = 64'b0$.

The optimal parity code-3 as shown in Figure 7, where Figure 7(a) is encoder and Figure 7(b) is decoder. Figure 7 uses the hamming code parity for horizontal bits and modulo-2 addition for vertical bits which reduces the complexity of adder. As per the hamming parity scheme, for each horizontal representation of matrix 32-bits require 6-bits of parity bits to obtain hamming code as:

– $H(0) = \text{xor}(0, 1, 3, 4, 6, 8, 10, 11, 13, 15, 17, 19, 21, 23, 25, 26, 28, 30)$,

required for parity i.e., 34 bits for 64-data bits. Hence, the OPC-4 improves the code rate. For example, consider the datain = 64'b0 then outputs H and V will be 2'b0 and 32'b0 respectively. In decoder say:

The dataread = 64'b11111111111111111111111111111111100000000000000000000000000000000000

i.e., 31 bits of error in MSB resulting in a large variation of value read from memory. Then,

Horizontal decoded bit = 2'b10

and

Vertical decoded bit = 32'b1111111111111111111111111111111110.

As H = 2'b0 and V = 32'b0, then

Hdiff = 2'b10

and

S = {32'b 11111111111111111111111111111110, 32'b0}

Then dataout = dataread ⊕ S = 64'b0.

3. RESULTS AND DISCUSSION

The verilog hardware description language is used to model codes, zynq field programmable gate array (28 nm) with part number XC7Z100-1FFG1156 in Xilinx Integrated Synthesis Environment 14.5 tool is used to verify them. The Figure 9 emphasizes simulation result of proposed optimal parity codes with the signals represented as dcorrected, en, dread, h and v in the order of visualization with sizes of 64, 1, 64, 2, 32 bits respectively. Here the original data is considered to be 64'b0, if the dread is 64'h00000007fffff i.e., 31 adjacent bits in least significant bit of dread, then all bits are corrected giving rise to correction capability of 31 adjacent bits but if more number of adjacent bits are found to be erroneous, then these codes fail to correct. Hence the correction capability is 51.56% erroneous bits in 64-bit data read from memory by using the proposed optimal parity codes.

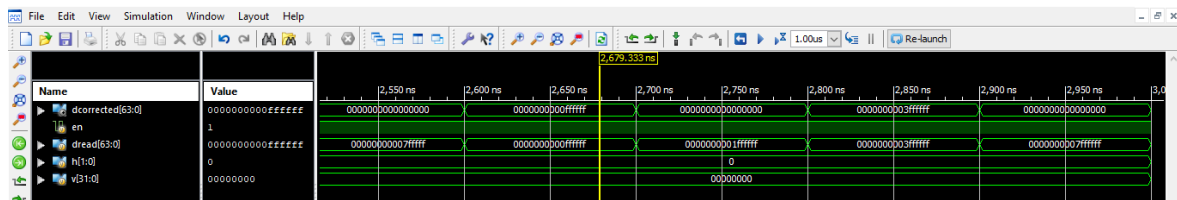


Figure 9. Integrated simulator result

As the focus is to improve the code rate for better efficiency of error detection and correction codes with fewer bits overhead. The same is proved from Figure 10, i.e., if maximum code rate is required, then horizontal vertical double diagonal code has 70.3% code rate and 42.1% of minimum bit overhead that compromises reliability. But if reliability is a concern, then among the codes capable of correcting (N/2)-1 adjacent bits, optimal code-4 is best suited as it has 65.306% code rate and only 53.125% of bit overhead.

The area is evaluated in terms of number of slice look-up tables occupied by the codes out of available 554800 slice look-up tables in zynq field programmable gate array with part number XC7Z100-1FFG1156 for their corresponding encoder and decoder. As per the requirement, the area overhead and delay should be kept at minimum for a good code. The same is proved for the field programmable gate array based code design in the Figure 11 and Figure 12.

The Figure 11 shows the comparison of existing and proposed codes in terms of slice look-up tables area. From Figure 11, among all the codes, optimal parity code-4 occupies minimum area of 30 slice look-up tables while the other existing and proposed codes occupy a minimum of 38 to a maximum of 98 slice look-up tables i.e., the area utilization is reduced by atleast 21.05% to 69.38% for encoder. Similarly, the optimal parity codes 3 and 4 occupy minimum area of 80 slice look-up tables while the other existing and

proposed codes occupy a minimum of 118 to a maximum of 175 slice look-up tables i.e., the area utilization is reduced by atleast 32.2% to 54.28% for decoder.

From Figure 12, among all the codes, optimal parity code-4 utilises power delay product of 435.74 pWs while the other existing and proposed codes use a minimum of 403.44 pWs to a maximum of 671.4 pWs i.e., the power delay product is reduced by atleast -8% to 35.09% for encoder. Similarly, the optimal parity code-4 utilises power delay product of 736.28 pWs while the other existing and proposed codes use a minimum of 763.28 pWs to a maximum of 1115.86 pWs i.e., the power delay product is reduced by atleast 3.53% to 34.01% for decoder. Hence the optimal code-4 proves to be best among the codes mentioned for error detection and correction especially when large number of bits get damaged and need repair from memory or equivalent device within very less time. These are well suited for image or video processing applications.

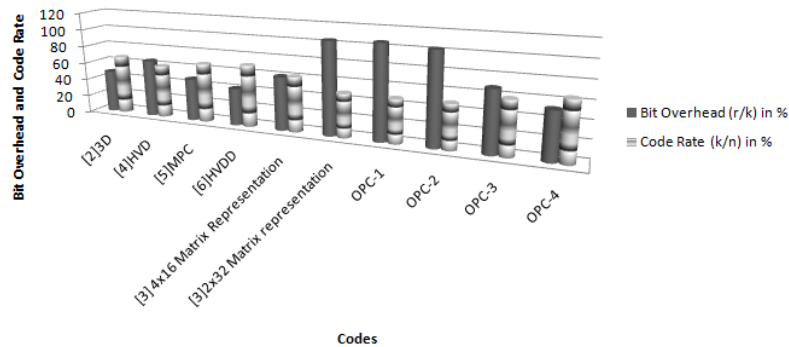


Figure 10. Codes comparison for code rate and bit overhead

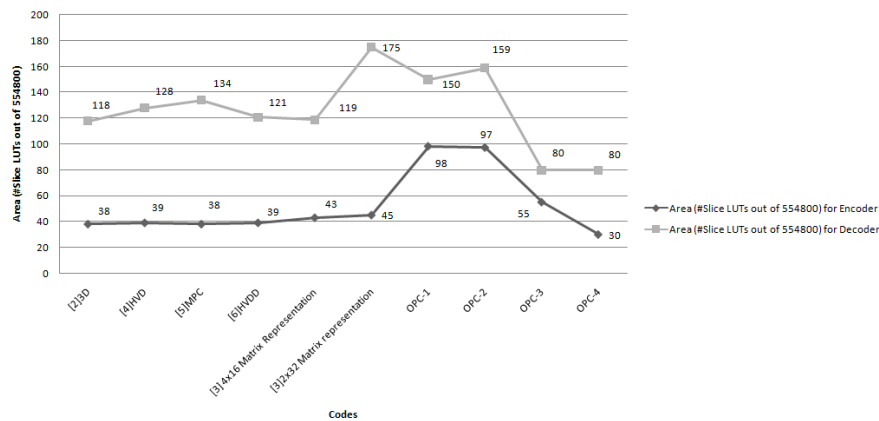


Figure 11. Codes comparison for area

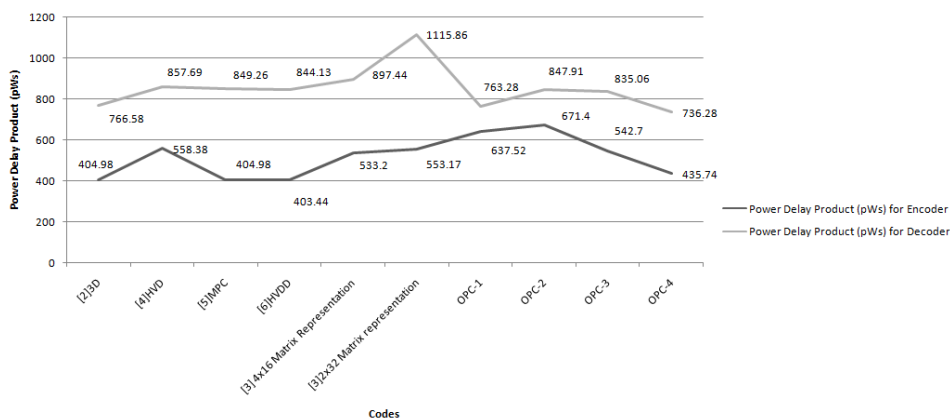


Figure 12. Codes comparison for power delay product

4. CONCLUSION

The embedded memories used in highly sensitive applications are usually error prone devices due to congestion at nanometer technologies exposed to radiation effects. During read operation operating at rated clock frequency multiple bit upsets happen and they exhibit as soft errors, then the error detection and correction codes are used to overcome them. The proposed codes are aimed to increase the code rate by using fewer bits overhead. These are capable of correcting at most $(N/2)-1$ adjacent errors in N bit data. The verilog hardware description language coded designs are verified in Xilinx tool for 28nm zynq field programmable gate array. From results obtained, among all the codes mentioned, the optimal code-4 proves to be efficient with 65.3% code rate and only 53.125% bit overhead, atleast 21.05% and 32.2% less area utilization for encoder and decoder respectively. Also it uses atleast 3.53% and 34.01% less power delay product for its encoder and decoder respectively. Further these codes can incorporate low power techniques for faster operation and optimized area.

ACKNOWLEDGEMENTS

I thank my guide Dr. C. Subhas and the faculty of Electronics and Communication Engineering, Jawaharlal Nehru Technological University Ananthapur, Ananthapuramu for their enormous support.




REFERENCES

- [1] A. S.-Macián, P. Reviriego, and J. A. Maestro, "Hamming SEC-DAED and Extended Hamming SEC-DED-TAED Codes Through Selective Shortening and Bit Placement," in *IEEE Transactions on Device and Materials Reliability*, vol. 14, no. 1, pp. 574-576, March 2014, doi: 10.1109/TDMR.2012.2204753.
- [2] S. Tambatkar, S. N. Menon, V. Sudarshan, M. Vinodhini, and N. S. Murty, "Error detection and correction in semiconductor memories using 3D parity check code with hamming code," *2017 International Conference on Communication and Signal Processing (ICCSP)*, 2017, pp. 0974-0978, doi: 10.1109/ICCSP.2017.8286516.
- [3] K. Neelima and C. Subhas, "Efficient Adjacent 3D Parity Error Detection and Correction Codes for Embedded Memories," *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 2020, pp. 1-5, doi: 10.1109/CONECCT50063.2020.9198452.
- [4] S. Sharma and P. Vijayakumar, "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)*, 2012, pp. 563-567, doi: 10.1109/ICDCSyst.2012.6188771.
- [5] V. Badole and A. Udawat, "Implementation of multidirectional parity check code using hamming code for error detection and correction," *International Journal of Research in Advent Technology*, vol. 2, pp. 1-6, 2014 [Online]. Available: <https://1library.net/document/zx25v8oq-implementation-multidirectional-parity-check-using-hamming-detection-correction.html>.
- [6] M. S. Rahman, M. S. Sadi, S. Ahammed, and J. Jurjens, "Soft error tolerance using Horizontal-Vertical-Double-Bit Diagonal parity method," *2015 International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*, 2015, pp. 1-6, doi: 10.1109/ICEEICT.2015.7307411.
- [7] L. -J. Saiz-Adalid, P. Gil, J. -C. Ruiz, J. Gracia-Morán, D. Gil-Tomás, and J.-C. Baraza-Calvo, "Ultrafast Error Correction Codes for Double Error Detection/Correction," *2016 12th European Dependable Computing Conference (EDCC)*, 2016, pp. 108-119, doi: 10.1109/EDCC.2016.28.
- [8] K. Neelima and C. Subhas, "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*, 2019, pp. 357-371, doi: 10.1007/978-981-33-4058-9_32.
- [9] J. Athira and B. Yamuna, "FPGA Implementation of an Area Efficient Matrix Code with Encoder Reuse Method," *2018 International Conference on Communication and Signal Processing (ICCSP)*, 2018, pp. 0254-0257, doi: 10.1109/ICCSP.2018.8524371.
- [10] A. J. Olazábal and J. P. Guerra, "Multiple Cell Upsets Inside Aircrafts. New Fault-Tolerant Architecture," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 1, pp. 332-342, Feb. 2019, doi: 10.1109/TAES.2018.2852198.
- [11] "IEEE Standard for Error Correction Coding of Flash Memory Using Low-Density Parity Check Codes," in *IEEE Std 1890-2018*, pp. 1-51, 2019, doi: 10.1109/IEEESTD.2019.8654228.
- [12] J. Samanta, J. Bhaumik, and S. Barman, "Compact and power efficient SEC-DED codec for computer memory," *Microsystem Technologies*, vol. 27, pp. 359-368, 2019, doi:10.1007/s00542-019-04366-7.
- [13] K. N. Dang and X. T. Tran, "An Adaptive and High Coding Rate Soft Error Correction Method in Network-on-Chips," *VNU Journal Of Science: Computer Science And Communication Engineering*, vol. 35, no. 1, Jun. 2019, doi:10.25073/2588-1086/vnucsce.218.
- [14] A. Radonjic, "(Perfect) Integer Codes Correcting Single Errors," in *IEEE Communications Letters*, vol. 22, no. 1, pp. 17-20, Jan. 2018, doi: 10.1109/LCOMM.2017.2757465.
- [15] A. Radonjic and V. Vujicic, "Integer codes correcting burst and random asymmetric errors within a byte," *Journal of the Franklin Institute*, vol. 355, no. 2, pp. 981-996, Jan. 2018, doi: 10.1016/j.jfranklin.2017.11.033.
- [16] A. Radonjic and V. Vujicic, "Integer codes correcting sparse byte errors," *Cryptography and Communications*, vol. 11, no. 5, pp. 1069-1077, Sep. 2019, doi: 10.1007/s12095-019-0350-9.
- [17] A. Subbiah and T. Ogunfunmi, "A Flexible Hybrid BCH Decoder for Modern NAND Flash Memories Using General Purpose Graphical Processing Units (GPGPUs)," *Micromachines*, vol. 10, no. 6, 2019, doi: 10.3390/mi10060365.
- [18] S. Lin, K. A.-Ghaffar, J. Li, and K. Liu, "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," in *IEEE Transactions on Information Theory*, vol. 66, no. 9, pp. 5358-5378, Sept. 2020, doi: 10.1109/TIT.2020.2978383.
- [19] U. S. Himaja, M. Vinodhini, and N. S. Murty, "Multi-Bit Low Redundancy Error control with Parity Sharing for NoC Interconnects," *2018 3rd International Conference on Communication and Electronics Systems (ICES)*, 2018, pp. 61-65, doi: 10.1109/CESYS.2018.8724118.
- [20] A. Das and N. A. Toubia, "Low complexity burst error correcting codes to correct MBUs in SRAMs," *Proceedings of the 2018 on*




- Great Lakes Symposium on VLSI*, 2018, pp. 219-224, doi: 10.1145/3194554.3194570.
- [21] J. Li, P. Reviriego, L. Xiao, C. Argyrides, and J. Li, "Extending 3-bit Burst Error-Correction Codes with Quadruple Adjacent Error Correction," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 2, pp. 221-229, Feb. 2018, doi: 10.1109/TVLSI.2017.2766361.
- [22] L. S.-Adalid, J. G.-Morán, D. G.-Tomás, J. C. B.-Calvo, and P. G.-Vicente, "Ultrafast Codes for Multiple Adjacent Error Correction and Double Error Detection," in *IEEE Access*, vol. 7, pp. 151131-151143, 2019, doi: 10.1109/ACCESS.2019.2947315.
- [23] J. G.-Morán, L. J. S.-Adalid, D. G.-Tomás, and P. J. G.-Vicente, "Improving Error Correction Codes for Multiple-Cell Upsets in Space Applications," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 10, pp. 2132-2142, Oct. 2018, doi: 10.1109/TVLSI.2018.2837220.
- [24] J. G.-Moran, L. J. S. -Adalid, J. C. B. -Calvo, and P. Gil, "Correction of Adjacent Errors with Low Redundant Matrix Error Correction Codes," *2018 Eighth Latin-American Symposium on Dependable Computing (LADC)*, 2018, pp. 107-114, doi: 10.1109/LADC.2018.00021.
- [25] S. Liu, P.Reviriego, J.A.Maestro, and L.Xiao, "Fault tolerant encoders for Single Error Correction and Double Adjacent Error Correction codes," *Microelectronics Reliability*, vol. 81 .pp. 167-173, Feb. 2018, doi: 10.1016/j.microrel.2017.12.017.
- [26] W. Zhou, H. Zhang, H. Wang, and Y. Wang, "Designing scrubbing strategy for memories suffering MCUs through the selection of optimal interleaving distance," *International Journal of Computational Science and Engineering*, vol. 19, no. 1, Jan. 2019, doi: 10.1504/IJCSE.2019.099639.
- [27] S. Dey, Aurangozeb, and M. Hossain, "Low-Latency Burst Error Detection and Correction in Decision-Feedback Equalization," in *IEEE Open Journal of Circuits and Systems*, vol. 2, pp. 91-100, 2021, doi: 10.1109/OJCS.2020.3039256.
- [28] A. Radonjic, "Integer Codes Correcting Double Errors and Triple-Adjacent Errors Within a Byte," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 8, pp. 1901-1908, Aug. 2020, doi: 10.1109/TVLSI.2020.2998364.
- [29] A. M. A. Hamdoon, Z. G. Mohammed, and E. A. Mohammed, "Design and implementation of single bit error correction linear block code system based on FPGA," *TELKOMNIKA Telecommunication, Computing, Electronics and Control*, vol. 17, no. 4, pp.1785-1795, Aug. 2019, doi: 10.12928/telkomnika.v17i4.12033.
- [30] W. Toghuj, "Modifying Hamming code and using the replication method to protect memory against triple soft errors," *TELKOMNIKA Telecommunication, Computing, Electronics and Control*, vol. 18, no. 5, pp. 2533-2542, Oct. 2020, doi: 10.12928/telkomnika.v18i5.13345.

BIOGRAPHIES OF AUTHORS



Neelima Koppala    received her M.Tech degree in Embedded Systems from JNTUA Ananthapuramu in 2015, and is now currently pursuing Ph.D in ECE from JNTUA Ananthapuramu. Her current research interests include Information Theory, VLSI, and Image Processing. She can be contacted at email: neelumtech17@gmail.com.



Chennapalli Subhas    received M.Tech Degree in Electronics Instrumentation and Communication Systems from Sri Venkateswara University and Ph. D degree in wireless communications from JNTUA Ananthapuramu. He is a member of IEEE, ISTE, IETE and IEICE. He is currently working as Professor of ECE and Vice-Principal in JNTUA College of Engineering, Kalikiri. He has published over 35 journal papers, 12 papers in conference proceedings and has guided 4 Ph.D Scholars. His research interests are in Error Detection and Correction Codes, and Communication Systems. He can be contacted at email: schennapalli@gmail.com.