Integer Codes Correcting Single Errors within Two Bytes

Aleksandar Radonjic

Institute of Technical Sciences of the Serbian Academy of Sciences and Arts, Belgrade, Serbia
E-mail: sasa_radonjic@yahoo.com

Abstract: **This paper presents a class of integer codes suitable for use in optical networks with low error rates. The presented codes are constructed with the help of a computer and have two important features: first, they can correct single errors affecting one or two *b*-bit bytes, and second, they use processor-friendly operations to encode/decode data bits. The effectiveness of the presented codes is demonstrated on theoretical models of four-core and six-core processors. The obtained results show that the decoder throughput reaches 14.70 Gbps, which is above the operating speed of 10G networks. Finally, the paper compares the proposed codes with BCH codes of similar properties. The comparison is made in terms of redundancy and the number of decoding operations.**

Keywords: **Integer codes, error correction, single errors, optical networks.**

1. Introduction

In most communication systems, the choice of a particular error correcting code depends upon the channel characteristics. In wireless networks, for example, channel errors occur in the form of bursts [1], [2]. For this reason, packets, transmitted over these networks, are protected with very powerful codes, such as Reed-Solomon (RS) codes. On the other hand, in optical networks (ONs) channel errors mostly affect isolated bits. Moreover, the experiments [3]-[5] have shown that, under normal conditions, the vast majority of corrupted packets contain only single errors. However, if ONs operate with receiver power less than the ideal, there will be an increase in the number of packets having two random errors [4]. This can lead to a significant deterioration in the quality of service (QoS), especially in cases where data is delay-sensitive and protected by cyclic redundancy codes (CRCs).

In order to avoid such a scenario, it is necessary to replace the CRCs with codes correcting two random errors. The most optimal candidates for that purpose are the Zetterberg codes [6], the Bose-Chaudhuri-Hocquenghem (BCH) codes [7] and the Orthogonal Latin Square (OLS) codes [8]. All these codes can correct single and double errors, but at different prices. So, for example, the Zetterberg and BCH codes have low redundancy and complex [7] or very complex [6] decoding procedure. On the other hand, the OLS codes are extremely inefficient in terms of redundancy, but can be decoded faster than all other double error correcting (DEC) codes.

Besides the mentioned codes, the potential candidates could be DEC codes detecting all unidirectional errors [9]-[13]. Although these codes are much more complex than standard DEC codes, they could be very useful when transmitting both real-time and non-real-time data. The same goes for RS codes, as well as for all other codes that can correct double (spotty) byte errors [14]-[16].

However, whichever of the above codes we choose, we will face the problem of their implementation. The reason is that these codes use finite field (FF) arithmetic, which is not supported by modern processors. Hence, to achieve high throughputs, the codes from [6]-[16] must be implemented in dedicated hardware (e.g. the software-based DEC-BCH decoders need several tens of clock cycles to process one bit [17], [18]). This, however, would lead to a significant increase in the cost of the network, since all nodes would have to be equipped with additional encoding/decoding (E/D) circuits.

Having this in mind, in this paper we present a class of codes whose implementation does not require the use of dedicated E/D hardware. The reason for this lies in the fact that network nodes are processor-based devices, and that already have hardware support for operations used by the proposed codes (integer and lookup table (LUT) operations). This means that all operations can be performed in the software, including data E/D and code rate change. Besides this feature, the proposed codes can correct single errors within one or two *b*-bit bytes. Given that these errors are dominant in most ONs, it is easy to conclude that the application of the proposed codes would allow the receiver to recover the majority of the corrupted packets. This would also lead to improving the QoS for delay-sensitive applications such as voice and video.

The organization of this paper is as follows: Section 2 deals with the construction of integer codes capable of correcting single errors within one or two bytes. The error correction procedure for these codes is described in Section 3. In Section 4, the presented codes are evaluated and compared with DEC-BCH codes, while Section 5 concludes the paper.

2. Codes Construction

Definition 1. [23] *Let* $Z_{2^b-1} = \{0, 1, \ldots, 2^b - 2\}$ *be the ring of integers modulo* $2^b - 1$ *and let* $B_i = \sum_{n=0}^{b-1} a_n \cdot 2^n$ *be the integer representation of a b-bit byte, where* $a_n \in \{0, 1\}$ *and* $1 \le i \le k$. *Then, the code* $C(b, k, c)$, *defined as*

$$C(b,k,c) = \left\{ (B_1, B_2, ..., B_k, B_{k+1}) \in Z_{2^b-1}^{k+1} : \sum_{i=1}^{k} C_i \cdot B_i \equiv B_{k+1} \pmod{2^b - 1} \right\} \quad (1)$$

is an $(kb + b, kb)$ *integer code, where* $c = (C_1, C_2, ..., C_k, 1) \in Z_{2^b-1}^{k+1}$ *is the coefficient vector and* $B_{k+1} \in Z_{2^b-1}$ *is an integer*.

Definition 2. [23] *Let* $x = (B_1, B_2,\ldots, B_k, B_{k+1}) \in Z_{2^b-1}^{k+1}$, $y = (\underline{B}_1, \underline{B}_2,\ldots, \underline{B}_k, \underline{B}_{k+1}) \in Z_{2^b-1}^{k+1}$ *and* $e = (\underline{B}_1 - B_1, \underline{B}_2 - B_2,\ldots, \underline{B}_k - B_k, B_{k+1} - \underline{B}_{k+1}) = (e_1, e_2,..., e_k, e_{k+1}) \in Z_{2^b-1}^{k+1}$ *be respectively, the sent codeword, the received codeword and the error vector. Then, the syndrome S of the received codeword is defined as*

$$S = \sum_{i=1}^{k} C_i \cdot \underline{B}_i - \underline{B}_{k+1} \pmod{2^b-1} = \sum_{i=1}^{k+1} e_i \cdot C_i \pmod{2^b-1} \tag{2}$$

The first step in the construction of the proposed codes is to determine the integer values of single errors corrupting one and two *b*-bit bytes. For that purpose, we will rely on the analysis from [22]. In that paper, it was shown that the integer value of a single error is equal to $e_i = \pm 2^r$, where $0 \le r \le b - 1$ and $1 \le i \le k + 1$. From this it is easy to conclude that two single errors will change the integer values of two *b*-bit bytes by $e_i = e_j = \pm 2^r$, where $0 \le r \le b - 1$ and $1 \le i < j \le k + 1$. Having this in mind, we can give the following definitions and theorems.

Definition 3. *The set of syndromes corresponding to single errors corrupting one b-bit byte is defined as*

$$s_1 = \left\{ \pm 2^r \cdot C_i \pmod{2^b-1} : 0 \le r \le b-1, 1 \le i \le k+1 \right\} \tag{3}$$

Definition 4. *The set of syndromes corresponding to single errors corrupting two b-bit bytes is defined as*

$$s_2 = \left\{ \pm 2^r \cdot C_i \pm 2^s \cdot C_j \pmod{2^b-1} : 0 \le r, s \le b-1, 1 \le i < j \le k+1 \right\} \tag{4}$$

Theorem 1. *The codes defined by (1) can correct all single errors corrupting one and two b-bit bytes if there exist k mutually different coefficients* $C_i \in Z_{2^b-1} \setminus \{0,1\}$ *such that*

1. $|s_1| = 2 \cdot b \cdot (k+1)$
2. $|s_2| = 2 \cdot b^2 \cdot k \cdot (k+1)$
3. $s_1 \cap s_2 = \varnothing$

where $|X|$ *denotes the cardinality of X.*

Proof. From [22] we know that Condition 1 is the necessary and sufficient condition for correcting single errors. On the other hand, Conditions 2 implies that single errors corrupting two *b*-bit bytes generate $2 \cdot b^2 \cdot k \cdot (k+1)$ nonzero syndromes. To prove this, observe that the set s_2 can be expressed as

$$s_2 = \bigcup_{i=1}^{k} d_i$$

where

$$d_1 = \left\{ \pm 2^r \cdot C_1 \pm 2^s \cdot C_j \pmod{2^b-1} : 0 \le r, s \le b-1, 2 \le j \le k+1 \right\}$$

$$d_2 = \left\{ \pm 2^r \cdot C_2 \pm 2^s \cdot C_j \pmod{2^b - 1} : 0 \le r, s \le b-1, 3 \le j \le k+1 \right\}$$
$$\vdots$$
$$d_k = \left\{ \pm 2^r \cdot C_k \pm 2^s \pmod{2^b - 1} : 0 \le r, s \le b-1 \right\}$$

Obviously, if the coefficients C_i have values such that

$$d_1 \cap d_2 \cap \cdots \cap d_k = \varnothing,$$
$$|d_1| = 4 \cdot b^2 \cdot k,$$
$$|d_2| = 4 \cdot b^2 \cdot (k-1),$$
$$\vdots$$
$$|d_k| = 4 \cdot b^2,$$

then

$$|s_2| = \sum_{i=1}^{k} |d_i| = 4 \cdot b^2 \cdot \sum_{i=1}^{k} (k-1+i) = 2 \cdot b^2 \cdot k \cdot (k+1).$$

Finally, Condition 3 is a necessary condition for distinguishing single errors corrupting one *b*-bit byte from those corrupting two *b*-bit bytes. Hence, the codes satisfying the conditions 1 to 3 are able to correct all single errors corrupting one or two *b*-bit bytes. □

Theorem 2. *Let* $\xi = s_1 \cup s_2$ *be the error set for (kb + b, kb) integer codes correcting single errors within one or two b-bit bytes. Then,*

$$|\xi| = |s_1| + |s_2| = 2 \cdot b \cdot (b \cdot k + 1) \cdot (k+1).$$

Proof. This theorem follows directly from Theorem 1.

Theorem 3. *For any* (*kb + b*, *kb*) *integer code correcting single errors within one or two b-bit bytes it holds that*

$$k \le \left\lfloor \frac{\sqrt{2^{b+1} + (b-1)^2 - 4} - b - 1}{2b} \right\rfloor.$$

Proof. From Definition 1 we know that the total number of nonzero syndromes is equal to $2^b - 2$. On the other hand, from Theorem 2 we know that the error set has $2 \cdot b \cdot (k+1) \cdot (b \cdot k + 1)$ nonzero elements. Obviously, we have the inequality

$$2 \cdot b \cdot (k+1) \cdot (b \cdot k + 1) \le 2^b - 2$$

wherefrom it follows that

$$k \le \left\lfloor \frac{\sqrt{2^{b+1} + (b-1)^2 - 4} - b - 1}{2b} \right\rfloor. \quad \square$$

The last step in constructing the presented codes is to find the C_i's that satisfy conditions of Theorem 1. For that purpose, it is necessary to use a computer. Through several experiments we investigated how the number of the *Ci*'s depends on the byte length (Table 1). Besides this, we

have focused on finding the coefficients for 16-bit and 32-bit codes (Table 2). The reason is that these codes are best suited for implementation on 32/64-bit processors.

Table 1. Number of coefficients for some byte lengths.

	$b = 8$	$b = 9$	$b = 10$	$b = 11$	$b = 12$	$b = 13$	$b = 14$	$b = 15$	$b = 16$
Theoretical bound	0	1	1	2	3	4	5	8	10
Computer-search result	0	1	1	1	2	2	3	3	3

Table 2. Coefficients for codes with parameters $b = 16, 32$ and $k \leq 32$.

$b = 16$							
19	213	537					
$b = 32$							
19	213	377	667	1905	3927	4387	6251
8885	9603	11453	14335	14707	22503	25869	29893
31985	36665	43669	67325	69505	69705	81097	86685
95069	98609	103547	122631	132627	159785	195623	210897

3. Error Correction Procedure

From Theorem 2 we know that the number of correctable errors is equal to $|\xi|$. Besides this, from the same theorem we implicitly know that the relationship between the syndrome (element of ξ), error location(s) (i, j) and error vector(s) (E_1, E_2) can be described using (3)-(4). From this it is easy to conclude that the syndrome table (ST) requires $|\xi| \times \left| 3 \cdot b + 2 \cdot \lceil log_2(k+1) \rceil \right|$ bits (Fig. 1) to store the error correction data.

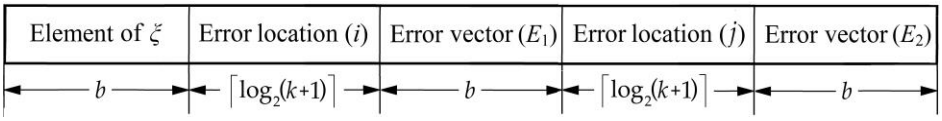

Fig. 1. Bit-width of one ST entry.

Given this, suppose that the codeword is received in error ($S \neq 0$) and that the elements of ξ are sorted in increasing order. In that case, the decoder will perform n_1 table lookups and n_1 comparisons $(1 \leq n_1 \leq \lfloor log_2|\xi| \rfloor + 2)$ [19]-[24] to find the entry where the first b bits match that of the syndrome S. After that, it will execute the operation(s):

- *for single errors corrupting one b-bit byte*

$$B_i = \underline{B}_i + E_1 \pmod{2^b - 1},\ 1 \leq i \leq k+1; \tag{5}$$

$$E_1 \in \{ \mp 2^r \pmod{2^b - 1} : 0 \leq r \leq b-1 \}$$

- *for single errors corrupting two b-bit bytes*

$$B_i = \underline{B}_i + E_1 \pmod{2^b - 1},\ 1 \leq i \leq k; \tag{6}$$

$$B_j = \underline{B}_j + E_2 \pmod{2^b - 1},\ i < j \leq k+1; \tag{7}$$

$$E_1, E_2 \in \{ \mp 2^r \pmod{2^b - 1} : 0 \leq r \leq b-1 \}$$

To make this procedure more clear, let us consider it on example of the (18, 9) code. (Remark: the ST for this code has $|\xi| = 360$ entries, and hence, it will be shown partially.)

Table 3. The ST for the (18, 9) code.

	Element of ξ	i	E_1	j	E_2		*Element of* ξ	i	E_1	j	E_2		*Element of* ξ	i	E_1	j	E_2
1	1	2	1	0	0	**37**	44	1	507	2	479	**215**	306	1	495	2	2
2	2	2	2	0	0	**38**	45	1	1	2	64	**216**	307	1	4	2	383
3	3	1	510	2	495	**39**	46	1	509	2	8	**217**	308	1	495	2	4
4	4	2	4	0	0	**40**	48	1	495	2	255	**218**	309	1	64	2	503
5	5	1	383	2	128	**41**	49	1	495	2	256	**219**	310	1	256	2	64
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
32	37	1	509	2	510	**210**	301	1	64	2	495	**356**	506	1	128	2	383
33	38	1	509	0	0	**211**	302	1	495	2	509	**357**	507	2	507	0	0
34	39	1	509	2	1	**212**	303	1	495	2	510	**359**	508	1	1	2	16
35	40	1	509	2	2	**213**	304	1	495	0	0	**359**	509	2	509	0	0
36	42	1	509	2	4	**214**	305	1	495	2	1	**360**	510	2	510	0	0

Example 1. Let $b = 9$, $k = 1$ and $C_1 = 19$. Now, suppose that we want to transmit 9 bits of data, $B_1 = 111010001_2 = 465$. In that case, the integer value of the check-byte will be equal to

$$B_{k+1} = \sum_{i=1}^{k} C_i \cdot B_i \pmod{2^b - 1} = 19 \cdot 465 \pmod{511} = 148 = 010010100_2$$

and the codeword will have 18 bits, $x = (B_1, B_2) = (111010001_2, 010010100_2) = (465, 148)$.

Scenario 1: Assume that during data transmission an error on the 8th bit has occurred, $y = (11101000\underline{1}1_2, 010010100_2) = (467, 148)$. In that case, using the expression (2), the decoder will calculate the syndrome S

$$S = \sum_{i=1}^{k} C_i \cdot \underline{B}_i - \underline{B}_{k+1} \pmod{2^b - 1} = 19 \cdot 467 - 148 \pmod{511} = 38$$

and conclude that the value $S = 38$ indicates an error within the first byte (Table 3). On the basis of this information, it will perform the error correcting by using

$$B_1 = \underline{B}_1 + E_1 \pmod{2^b - 1} = 467 + 509 \pmod{511} = 465.$$

Scenario 2: Suppose that during data transmission an error occurs on the 9th and 14th bits, $y = (11101000\underline{0}_2, 0100\underline{0}0100_2) = (405, 111)$. As in the previous case, the decoder will calculate

$$S = \sum_{i=1}^{k} C_i \cdot \underline{B}_i - \underline{B}_{k+1} \pmod{2^b - 1} = 19 \cdot 464 - 132 \pmod{511} = 508$$

and conclude that the value $S = 508$ indicates an error within the first and second byte (Table 3). As a result, the following procedure will take place

$$B_1 = \underline{B}_1 + E_1 \pmod{2^b - 1} = 464 + 1 \pmod{511} = 465$$

$$B_2 = \underline{B}_2 + E_2 \pmod{2^b - 1} = 132 + 16 \pmod{511} = 148.$$

4. Evaluation and Comparison with DEC-BCH Codes

Among all DEC codes, the most studied are DEC-BCH codes [25]-[30]. One of the reasons for this lies in their redundancy, which is very close to the minimum. In this respect, DEC-BCH codes outperform the presented ones as well. From Table 4 we see that, for practical data lengths

up to 2048 bits, the proposed codes require 2 to 4 check bits more than DEC-BCH codes. This result, however, can be considered satisfactory if we take into account the redundancy of other DEC codes (e.g. DEC-OLS codes).

Table 4. Check-bit lengths of the proposed, DEC-BCH and DEC-OLS codes.

Codes	Data word length (bits)						
	32	64	128	256	512	1024	2048
DEC-BCH codes	12	14	16	18	20	22	24
DEC-OLS codes	24	32	48	64	100	128	184
Proposed codes	14	17	19	21	24	26	28

Table 5. Comparison of the proposed and DEC-BCH codes.

Main characteristics	Proposed codes	DEC-BCH codes
Error correction capabilities	Correction of single errors in one or two bytes	Correction of single and double errors
Processing of data bits	Integer and LUT operations	FF operations
Number of check bits per 256/512/1024-bit data word	21/24/26	18/20/22
Number of decoding operations per 256/512/1024-bit data word	31/34/37	88/115/152
Memory requirements per 256/512/1024-bit data word	1.9/7.4/29.2 MB	None
Type of implementation	Software	Hardware

When it comes to the processing of data bits, the presented codes have a clear advantage over DEC-BCH codes. This is confirmed by the fact that the fully parallel decoder uses $log_2 K/b + \lfloor log_2 |\xi| \rfloor + 6$ operations per K-bit data word [19], while the optimized DEC-BCH decoder performs $4 \cdot \sqrt{K} + 24$ operations [31], [32] (Table 5). An additional drawback of DEC-BCH codes is that they use FF operations, which are not supported by modern processors. As already stated, this means that DEC-BCH codes must be implemented in dedicated hardware, which increases the cost of the network. On the other hand, the presented codes use integer and LUT operations, which are supported by all processors. Thanks to this, they have the potential to be implemented "for free" (in software). In order to show this, we will rely on the results from [21], [23]. In the first paper, for example, it was shown that any theoretical decoder, implemented on the four-core processor, needs one second to decode

$$G_4 = \frac{(3.5 \cdot 10^9) \cdot 128 \cdot k}{9 \cdot k + 29 \cdot n_1 + 4} \tag{8}$$

bits. On the other hand, the analysis from [23] has shown that any theoretical decoder, implemented on the six-core processor, requires one second to decode

$$G_6 = \frac{(3.3 \cdot 10^9) \cdot 192 \cdot k}{9.5 \cdot k + 35 \cdot n_1 + 3} \tag{9}$$

data bits. Now, if we apply these results to the presented theory, we can easily be convinced that even for smaller values of *k*, the proposed decoder achieves the throughput above the operational speed of 10G networks (Table 6). Intuitively, it can be concluded that better results can be obtained using longer codes and/or more powerful processor [33]. The only prerequisite, in this regard, is that the size of the syndrome table ($\left|\xi\right| \times \left| 3 \cdot b + 2 \cdot \left\lceil log_2(k+1) \right\rceil \right|$ bits) does not exceed the capacity of the last level cache (for more details, see [20]).

Table 6. Memory requirements and theoretical decoding throughputs for some 32-bit codes.

Code	*k*	Memory requirements for storing the syndrome table	$n_{1\max.}$	Minimum theoretical decoding throughput	
				Four-core processor	Six-core processor
(608, 576)	18	9.30 MB	21	10.41 Gbps	12.55 Gbps
(640, 608)	19	10.33 MB	21	10.86 Gbps	13.11 Gbps
(672, 640)	20	11.41 MB	21	11.30 Gbps	13.66 Gbps
(704, 672)	21	12.56 MB	21	11.73 Gbps	14.19 Gbps
(736, 704)	22	13.75 MB	21	12.15 Gbps	14.72 Gbps
(768, 736)	23	14.99 MB	22	12.14 Gpbs	14.70 Gpbs

5. Conclusion

This paper has presented a new class of integer codes. It has been shown that the presented codes have two important features: first, they can correct single errors within one or two *b*-bit bytes, and second, they have the ability to be implemented "for free" (in software). Thanks to this, they can be viewed as a low-cost alternative to DEC-BCH codes, especially in ONs with low error rates.

References

[1] A. Willig *et al.*, “Measurements of a Wireless Link in an Industrial Environment Using an IEEE 802.11-Compliant Physical Layer,” *IEEE Trans. Ind. Electron.*, vol. 49, no. 6, pp. 1265–1282, Dec. 2002.

[2] S. Khayam *et al.*, “Performance Analysis and Modeling of Errors and Losses Over 802.11b LANs for High-Bit-Rate Real-Time Multimedia", *Signal Process.: Image Commun.*, vol. 18, no. 7, pp. 575-595, Aug. 2003.

[3] T. Ono *et al.*, “Bit Error Statistical Analysis of Optical Transmission Systems,” in D.W. Faulkner and A.L. Harmer (Eds.), pp. 43-49, IOS Press, 2000.

[4] L. James, “Error Behaviour in Optical Networks”, PhD thesis, Dept. of Engineering, Univ. Cambridge, 2005.

[5] P. Anslow and O. Ishida, “Error Distribution in Optical Links”, *IEEE 802.3 HSSG Interim Meeting*, Nov. 2007.
[6] L. H. Zetterberg, “Cyclic Codes from Irreducible Polynomials for Correction of Multiple Errors”, *IEEE Trans. Inform. Theory*, vol. 8, no. 1, pp. 13-20, Jan. 1962.
[7] R. Bose and D. Ray-Chaudhuri, “On a Class of Error Correcting Binary Group Codes”, *Inf. Control*, vol. 3, no. 1, pp. 68-79, Mar. 1960.
[8] M. Y. Hsiao, D. C. Bossen and R. T. Chien, “Orthogonal Latin Square Codes”, *IBM J. Res. Develop.*, vol. 14, no. 4, pp. 390-394, Jul. 1970.
[9] F. Boinck and H. Van Tilborg, “Constructions and Bounds for Systematic *t*-EC/AUED Codes,” *IEEE Trans. Inform. Theory*, vol. 36, no. 6, pp. 1381-1390, Nov. 1990.
[10] J. Bruck and M. Blaum, “New Techniques for Constructing *t*-EC/AUED Codes,” *IEEE Trans. Comput.*, vol. 41, no. 10, pp. 1318-1324, Oct. 1992.
[11] R. Katti and M. Blaum, “An Improvement on Constructions of *t*-EC/AUED Codes,” *IEEE Trans. Comput.*, vol. 45, no. 5, pp. 607-608, May 1996.
[12] S. Al-Bassam, “Another Method for Constructing *t*-EC/AUED Codes,” *IEEE Trans. Comput.*, vol. 49, no. 9, pp. 964-966, Sep. 2000.
[13] I. Naydenova and T. Klove, “Some Optimal Binary and Ternary *t*-EC-AUED Codes,” *IEEE Trans. Inf. Theory*, vol. 55, no. 11, pp. 4898-4904, Nov. 2009.
[14] G. Feng, X. Wu and T. R. N. Rao, “New Double-Byte Error-Correcting Codes for Memory Systems,” *IEEE Trans. Inform. Theory*, vol. 44, pp. 1152-1163, Mar. 1998.
[15] K. Suzuki, T. Kashiyama and E. Fujiwara, “A General Class of *m*-Spotty Byte Error Control Codes,” *IEICE Trans*., vol. 90-A, no. 7, pp. 1418-1427, Jan. 2007.
[16] J. Bhaumik and D. Roy Chowdhury, “New Architectural Design of CA-Based Codec,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*., vol. 18, no. 7, pp. 1139-1144, Jul. 2010.
[17] J. Cho and W. Sung, “Efficient Software-Based Encoding and Decoding of BCH Codes,” *IEEE Trans. Comput.*, vol. 58, no. 7, pp. 878-889, July 2009.
[18] A. Subbiah and T. Ogunfunmi, “A Flexible Hybrid BCH Decoder for Modern NAND Flash Memories Using GPGPUs,” *Micromachines*, vol. 10, no. 6, pp. 1-15, Jun. 2019.
[19] A. Radonjic, K. Bala and V. Vujicic, “Integer Codes Correcting Double Asymmetric Errors,” *IET Commun.*, vol. 10, no. 14, pp. 1691-1696, Sep. 2016.
[20] A. Radonjic and V. Vujicic, “Integer Codes Correcting High-Density Byte Asymmetric Errors,” *IEEE Commun. Lett.*, vol. 21, no. 4, pp. 694-697, Apr. 2017.
[21] A. Radonjic and V. Vujicic, “Integer Codes Correcting Burst and Random Asymmetric Errors within a Byte,” *J. Franklin Inst.*, vol. 355, no. 2, pp. 981-996, Jan. 2018.
[22] A. Radonjic, “(Perfect) Integer Codes Correcting Single Errors,” *IEEE Commun. Lett.*, vol. 22, no. 1, pp. 17-20, Jan. 2018.
[23] A. Radonjic and V. Vujicic, “Integer Codes Correcting Sparse Byte Errors,” *Cryptogr. Commun.*, vol. 11, no. 5, pp. 1069-1077, Sept. 2019.
[24] A. Radonjic and V. Vujicic, “Integer Codes Correcting Burst Asymmetric Errors within a Byte and Double Asymmetric Errors,” *Cryptogr. Commun.*, vol. 12, no. 2, pp. 221-230, Mar. 2020.
[25] C. Hartmann, “A Note on the Decoding of Double-Error-Correcting Binary BCH Codes of Primitive Length,” *IEEE Trans. Inform. Theory*, vol. 17, no. 6, pp. 765-766, Nov. 1971.
[26] T. P. Berger, “The Automorphism Group of Double-Error-Correcting BCH Codes,” *IEEE Trans. Inform. Theory*, vol. 40, no. 2, pp. 538-542, Mar. 1994.
[27] P. Charpin, “Weight Distributions of Cosets of Two-Error-Correcting Binary BCH Codes, Extended or Not,” *IEEE Trans. Inf. Theory*, vol. 40, no. 5, pp. 1425–1442, Sep. 1994.
[28] E.-H. Lu, T. Chang, "New Decoder for Double-Error-Correcting Binary BCH Codes", *IEE Proc. Commun.*, vol. 143, no. 3, pp. 129-132, Jun. 1996.

[29] P. Crepeau, “Classification of Error Locator Polynomials for Double Error Correcting BCH Codes", *IEEE Trans. Commun.*, vol. 46, no. 8, pp. 977-980, Aug. 1998.
[30] T. A. Gulliver, W. Lin and F. Dehne, “Fast Parallel Decoding of Double Error Correcting Binary BCH Codes,” *Appl. Math. Lett.*, vol. 11, no. 6, pp. 11-14, Nov. 1998.
[31] R. Blahut, *Algebraic Codes for Data Transmission*, Cambridge Univ. Press, 2003.
[32] D. Schipani, M. Elia and J. Rosenthal, “On the Decoding Complexity of Cyclic Codes up to the BCH Bound,” *Proc. IEEE ISIT 2011*, pp. 835–839, July 2011.
[33] A. Fog, “The Microarchitecture of Intel, AMD and via CPUs: An Optimization Guide for Assembly Programmers and Compiler Makers,” Tech. Univ. Denmark, Mar. 2020.